

Managing Uncertainty in the Single Airport Ground Holding Problem
using Scenario-based and Scenario-free Approaches

by

Pei-Chen Liu

B.S. (National Taiwan University) 1997
M.S. (University of California, Berkeley) 1998
M.S. (Stanford University) 1999
M.S. (University of California, Berkeley) 2006

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering-Civil and Environmental Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Mark M. Hansen, Chair
Professor Samer M. Madanat
Professor Andrew E. Lim

Spring 2007

**Managing Uncertainty in the Single Airport Ground Holding Problem
using Scenario-based and Scenario-free Approaches**

Copyright 2007

by

Pei-Chen Liu

Abstract

Managing Uncertainty in the Single Airport Ground Holding Problem using Scenario-based and Scenario-free Approaches

by

Pei-Chen Liu

Doctor of Philosophy in Civil and Environmental Engineering

University of California, Berkeley

Professor Mark Hansen, Chair

The goal of this dissertation is to improve the ability of air traffic managers to handle uncertainty and incorporate probabilistic forecast information in ground delay programs (GDPs). In particular, we investigate ways to advance the support of decision-making under uncertainty in GDPs for a single destination airport. We explore methods to model the stochasticity in GDP operations and mechanisms that respond to conditions dynamically such that the overall system performance is optimized.

Recent developments in solving the single airport ground holding problem (SAGHP) use static or dynamic stochastic programs to manage uncertainty about how airport capacities will evolve. Both static and dynamic models involve the use of scenarios that depict different possible capacity evolutions. Dynamic models also require scenario trees featuring branch points where previously similar capacity profiles become distinct. In this dissertation, we present methodologies for generating and using scenarios and scenario trees from empirical data and examine the performance of scenario-based models in a real-world setting. We find that most U.S. airports have capacity profiles that can be classified into a small number of nominal scenarios, and for a number of airports these scenarios can be naturally combined into scenario trees. The delay costs yielded from

using dynamic optimization are found to be consistently and considerably lower than that from static optimization. However, the costs incurred from applying scenario-based optimization, either static or dynamic, to these airports is considerably higher than what the “theoretical” optimization results suggest, because actual capacities vary around the nominal values assumed in the optimization, and because of uncertainty in navigating scenario trees that the idealized models ignore.

In light of the shortcomings of the scenario-based models, we develop a sequential decision model that is not limited by a small set of scenarios, which we termed as “scenario-free” model. The model is formulated as a dynamic program and the challenge lies in the computational load for solving large-scale problem instances, due to the “curse of dimensionality” of dynamic programming. We present several computational strategies to manage the complexity. We show that the computational strategies reduce the computation time significantly without much loss in optimality in several test cases. We also demonstrate the computational feasibility of the model for problems of realistic scale.

Finally, we compare the performance of the scenario-based and scenario-free models solving identical problems in a real-world setting. We show that the scenario-free model leads to lower average incurred delay cost and lower variation in incurred costs. Moreover, we find the scenario-free model yields solutions that contain more balanced distributions of ground and airborne delay. Though the magnitude of cost reduction was smaller than anticipated based on earlier results, the closeness of the expected and the incurred cost and the narrower spread of the incurred costs from using the scenario-free model make it a more informational and predictable approach.

Professor Mark M. Hansen

Dissertation Committee Chair

CONTENTS

| | |
|---|-----------|
| CHAPTER 1 INTRODUCTION..... | 1 |
| 1.1 OVERVIEW OF THE AIR TRAFFIC SYSTEM | 2 |
| 1.2 THE GROUND DELAY PROGRAM | 4 |
| 1.3 THE PARADIGM OF COLLABORATIVE DECISION MAKING | 5 |
| 1.4 EMERGING TECHNOLOGIES | 7 |
| 1.4.1 <i>Probabilistic Forecasts</i> | 7 |
| 1.4.2 <i>Multi-center Traffic Management Advisor</i> | 7 |
| 1.5 OPTIMIZATION MODELS VERSUS GDP IN PRACTICE | 9 |
| 1.6 RESEARCH OBJECTIVE AND APPROACH | 10 |
| CHAPTER 2 LITERATURE REVIEW | 13 |
| 2.1 MODELS OF SINGLE DESTINATION AIRPORT GROUND HOLDING PROBLEM | 13 |
| 2.1.1 <i>Ball et al. Static Model</i> | 16 |
| 2.1.2 <i>Mukherjee-Hansen Dynamic Model</i> | 18 |
| 2.2 MODELS OF MULTIPLE DESTINATION AIRPORT GROUND HOLDING PROBLEM | 21 |
| 2.3 AIR TRAFFIC FLOW MANAGEMENT PROBLEM WITH EN-ROUTE CAPACITY CONSTRAINTS | 22 |
| 2.4 SCENARIO GENERATION METHODS FOR STOCHASTIC PROGRAMMING | 25 |
| CHAPTER 3 SCENARIO-BASED MODELS FOR SINGLE AIRPORT GROUND HOLDING POLICY PROBLEM: FROM THEORY TO PRACTICE..... | 28 |
| 3.1 USING SCENARIO TREES IN AIR TRAFFIC MANAGEMENT..... | 30 |
| 3.2 DEVELOPING CAPACITY SCENARIO TREES..... | 32 |
| 3.2.1 <i>Identifying Capacity Scenarios</i> | 33 |
| 3.2.2 <i>Constructing Scenario Trees</i> | 36 |
| 3.2.3 <i>Empirical Capacity Scenarios and Scenario Trees</i> | 39 |
| 3.3 PERFORMANCE OF OPTIMIZATION MODELS | 44 |
| 3.4 IMPLEMENTING OPTIMAL STRATEGIES | 48 |

| | | |
|--|---|-----|
| 3.4.1 | <i>Branch Identification Methodology</i> | 49 |
| 3.4.2 | <i>Implementation Performance</i> | 52 |
| 3.5 | SUMMARY..... | 58 |
| | | |
| CHAPTER 4 SCENARIO-FREE MODELS FOR SINGLE AIRPORT GROUND HOLDING | | |
| POLICY PROBLEM.....60 | | |
| 4.1 | SEQUENTIAL DECISION MODEL FOR THE SAGHP..... | 61 |
| 4.1.1 | <i>Algorithmic complexity</i> | 70 |
| 4.2 | COMPUTATIONAL SOLUTION STRATEGIES | 73 |
| 4.2.1 | <i>Memoization</i> | 73 |
| 4.2.2 | <i>Priority Ordering</i> | 79 |
| 4.2.3 | <i>Heuristics for Searching for Best Action</i> | 85 |
| 4.2.4 | <i>Implementation Design</i> | 94 |
| 4.3 | COMPUTATIONAL EXPERIMENTS..... | 96 |
| 4.3.1 | <i>Computation Time</i> | 97 |
| 4.3.1.1 | Effect of Memoization..... | 98 |
| 4.3.1.2 | Effect of Problem Parameters..... | 100 |
| 4.3.2 | <i>Effect of Computational Strategies</i> | 109 |
| 4.3.2.1 | Time Epoch Aggregation | 109 |
| 4.3.2.2 | Effect of Priority Ordering | 112 |
| 4.3.2.3 | Effect of Heuristics..... | 115 |
| 4.3.3 | <i>Application</i> | 117 |
| 4.4 | SUMMARY..... | 122 |
| | | |
| CHAPTER 5 COMPARISON OF SCENARIO-BASED AND SCENARIO-FREE MODELS FOR SAGHP.....124 | | |
| 5.1 | RELATIONSHIP BETWEEN THE SCENARIO-BASED AND SCENARIO-FREE MODELS..... | 124 |
| 5.2 | MODEL PERFORMANCE WITH SIMULATED DATA | 127 |
| 5.2.1 | <i>The Simulation Experiment</i> | 127 |
| 5.2.2 | <i>Results and Discussion</i> | 130 |

| | | |
|------------------------------------|--|------------|
| 5.3 | MODEL PERFORMANCE WITH REAL DATA | 133 |
| 5.3.1 | <i>Experiment Setup</i> | 133 |
| 5.3.2 | <i>Results and Discussion</i> | 135 |
| 5.3.3 | <i>Realism of Markovian Model</i> | 144 |
| 5.4 | SUMMARY..... | 147 |
| CHAPTER 6 CONCLUSIONS | | 150 |
| 6.1 | RECOMMENDATIONS FOR FUTURE WORK | 152 |
| 6.1.1 | <i>Characterization of the Stochastic Process</i> | 153 |
| 6.1.1.1 | Enhancement of Scenario Trees | 153 |
| 6.1.1.2 | Methodology for Transition Matrix Estimation..... | 153 |
| 6.1.1.3 | History-dependent Markov Process..... | 154 |
| 6.1.2 | <i>Assessment of Model Performance</i> | 154 |
| 6.1.2.1 | Justice to the Static Model..... | 154 |
| 6.1.2.2 | Generalizability of the Performance Results | 154 |
| 6.1.3 | <i>Complexity of the Scenario-free Approach</i> | 155 |
| 6.1.3.1 | Estimator for the Computational Cost | 155 |
| 6.1.3.2 | Other Strategies to Manage the Complexity | 155 |
| 6.1.4 | <i>Refinement for the Scenario-Free Model</i> | 156 |
| 6.1.4.1 | Other Priority Ordering Schemes | 156 |
| 6.1.4.2 | Nonlinear Cost | 156 |
| 6.1.5 | <i>Implementation of the Scenario-Free Approach</i> | 156 |
| 6.1.5.1 | Benefits from using the Scenario-Free Approach..... | 156 |
| 6.1.5.2 | Incorporation of the Scenario-Free Approach into CDM | 157 |
| 6.1.6 | <i>Application of Model Interchangeability</i> | 157 |

LIST OF FIGURES

| | |
|---|-----|
| FIGURE 2-1 A SIMPLE SCENARIO TREE FOR THE ILLUSTRATION OF THE COUPLING CONSTRAINTS | 21 |
| FIGURE 3-1 MEAN OF ONE SCENARIO AND 99% CONFIDENCE BAND OF ANOTHER SCENARIO (CI = CONFIDENCE INTERVAL, LB = LOWER BOUND, UB = UPPER BOUND) | 38 |
| FIGURE 3-2 SCENARIOS AT SFO | 40 |
| FIGURE 3-3 SCENARIOS AT BOS | 41 |
| FIGURE 3-4 SCENARIOS AT MIA | 41 |
| FIGURE 3-5 SCENARIOS AT ATL | 42 |
| FIGURE 3-6 SCENARIOS AT DFW | 43 |
| FIGURE 3-7 SCENARIO TREE AT SFO | 44 |
| FIGURE 3-8 OPTIMAL DELAY COST OBTAINED WITH DIFFERENT COST RATIOS, JANUARY 8, 2004, SFO. | 46 |
| FIGURE 3-9 RATIOS OF OPTIMAL DELAY COSTS WITH VARIOUS AIR-TO-GROUND UNIT COST RATIOS | 46 |
| FIGURE 3-10 OPTIMAL TOTAL DELAY COSTS FROM DYNAMIC AND STATIC MODELS, 15 DAYS, SFO. | 47 |
| FIGURE 3-11 BRANCH IDENTIFICATION RESULTS WITH DIFFERENT BRANCHING LAGS | 52 |
| FIGURE 3-12 EFFECTS OF SCALING NOMINAL SCENARIO CAPACITIES | 56 |
| FIGURE 3-13 AVERAGE TOTAL DELAY COST--EXPECTED VERSUS INCURRED COST | 57 |
| FIGURE 4-1 ILLUSTRATION OF THE SEQUENTIAL DECISION MODEL | 62 |
| FIGURE 4-2 ILLUSTRATION OF THE SEQUENTIAL DECISION MODEL FOR THE SAGHP | 63 |
| FIGURE 4-3 SIMPLE EXAMPLE OF SEQUENTIAL DECISION MODEL OF THE SAGHP RENDERED AS AN SDM TREE | 69 |
| FIGURE 4-4 OVERLAPPING SUBPROBLEMS IN AN SDM TREE | 75 |
| FIGURE 4-5 ILLUSTRATION OF COST-TO-GO FUNCTION WITH TWO GROUPS OF FLIGHTS | 88 |
| FIGURE 4-6 SUPERPOSITION OF ELEMENTS IN THE COST-TO-GO FUNCTION | 90 |
| FIGURE 4-7 EFFECT OF PRIORITY ORDERING LGF ON THE COST-TO-GO FUNCTION | 91 |
| FIGURE 4-8 ILLUSTRATION OF THE MVC ARCHITECTURE | 95 |
| FIGURE 4-9 COMPUTATION TIME WITH AND WITHOUT MEMOIZATION | 100 |

| | |
|---|-----|
| FIGURE 4-10 TRANSITION MATRICES FOR DIFFERENT NUMBER OF CAPACITY LEVELS | 102 |
| FIGURE 4-11 COMPUTATION TIME WITH DIFFERENT NUMBER OF CAPACITY LEVELS | 103 |
| FIGURE 4-12 COMPUTATION TIME WITH DIFFERENT TOTAL NUMBER OF FLIGHTS | 104 |
| FIGURE 4-13 COMPUTATION TIME WITH DIFFERENT PLANNING HORIZON..... | 106 |
| FIGURE 4-14 COMPUTATION TIME WITH DIFFERENT NUMBER OF TYPES OF FLIGHT DURATIONS | 108 |
| FIGURE 4-15 COMPUTATION TIME WHEN DIVIDING THE PLANNING HORIZON INTO DIFFERENT NUMBER OF PERIODS..... | 111 |
| FIGURE 4-16 COMPUTATION TIME WITH AND WITHOUT PRIORITY ORDERINGS OF FLIGHTS | 113 |
| FIGURE 4-17 OPTIMAL TOTAL DELAY COST WITH AND WITHOUT PRIORITY ORDERING OF FLIGHTS..... | 114 |
| FIGURE 4-18 COMPUTATION TIME WITH AND WITHOUT THE HEURISTIC | 116 |
| FIGURE 4-19 IMPACT OF THE HEURISTICS ON THE BEST TOTAL DELAY COST | 117 |
| FIGURE 4-20 EFFECT OF COMPUTATIONAL STRATEGIES | 123 |
| FIGURE 5-1 TRANSITION MATRICES FOR THE SIMULATION | 128 |
| FIGURE 5-2 CAPACITY SCENARIOS DERIVED FROM THE GENERATED CAPACITY DATA | 129 |
| FIGURE 5-3 PERFORMANCE OF MODELS IN SIMULATION EXPERIMENT..... | 131 |
| FIGURE 5-4 COMPARISON OF DELAY DISTRIBUTION | 133 |
| FIGURE 5-5 PERFORMANCE OF MODELS IN A REAL-WORLD SETTING WITH REAL DATA | 136 |
| FIGURE 5-6 DELAY DISTRIBUTION OF SCENARIO-BASED AND SCENARIO-FREE MODELS | 137 |
| FIGURE 5-7 DELAY DISTRIBUTION FROM USING SCENARIO-BASED MODEL..... | 138 |
| FIGURE 5-8 DELAY DISTRIBUTION FROM USING SCENARIO-FREE MODEL | 138 |
| FIGURE 5-9 OVERAGE OF DELAY FROM USING THE SCENARIO-BASED MODEL | 139 |
| FIGURE 5-10 MODEL PERFORMANCE BY YEAR..... | 140 |
| FIGURE 5-11 MODEL PERFORMANCE BY MATCHED CAPACITY SCENARIO..... | 141 |
| FIGURE 5-12 MODEL PERFORMANCE WITH ADVANCED INFORMATION | 144 |
| FIGURE 5-13 AVERAGE AAR FROM SIMULATION AND FIELD DATA..... | 145 |
| FIGURE 5-14 STANDARD DEVIATION OF AAR FROM SIMULATION AND FIELD DATA | 146 |
| FIGURE 5-15 AUTOCORRELATION FOR LAGS OF ONE TO FOUR TIME PERIODS | 147 |

LIST OF TABLES

| | |
|---|-----|
| TABLE 3-1 OPTIMAL TOTAL DELAY COSTS AT FOUR AIRPORTS, REPORTED IN GROUND DELAY TIME PERIOD EQUIVALENTS..... | 45 |
| TABLE 3-2. TOTAL DELAY COST UNDER DYNAMIC AND STATIC MODELS..... | 55 |
| TABLE 4-1 CAPACITY (FLIGHTS/TIME PERIOD) TRANSITION MATRIX FOR THE TEST PROBLEM ON THE USE OF MEMOIZATION..... | 99 |
| TABLE 4-2 FLIGHT SCHEDULE FOR THE TEST PROBLEM ON THE USE OF MEMOIZATION..... | 99 |
| TABLE 4-3 FLIGHT SCHEDULE FOR THE TEST PROBLEM OF DIFFERENT CAPACITY LEVELS..... | 101 |
| TABLE 4-4 TRANSITION MATRIX OF TEST PROBLEM FOR DIFFERENT PLANNING HORIZONS..... | 105 |
| TABLE 4-5 FLIGHT SCHEDULE OF THE TEST PROBLEM FOR DIFFERENT PLANNING HORIZONS..... | 105 |
| TABLE 4-6 TEST CASES WITH DIFFERENT FLIGHT DURATION MIXES..... | 107 |
| TABLE 4-7 TRANSITION MATRIX FOR THE EXPERIMENT WITH MULTIPLE FLIGHT DURATION MIXES..... | 107 |
| TABLE 4-8 TRANSITION MATRIX FOR EVERY HALF AN HOUR..... | 110 |
| TABLE 4-9 TRANSITION MATRIX FOR EVERY HOUR..... | 110 |
| TABLE 4-10 TRANSITION MATRIX FOR EVERY TWO HOURS..... | 110 |
| TABLE 4-11 FLIGHT SCHEDULE FOR THE TEST CASES..... | 112 |
| TABLE 4-12 TRANSITION MATRIX FOR EXPERIMENTS IN SUBSECTION 4.3.2..... | 112 |
| TABLE 4-13 ARRIVAL DATA AT SFO FOR YEAR 2006 FROM FAA ASPM DATABASE..... | 119 |
| TABLE 4-14 ARRIVAL WITH EDCT IN MARCH, 2006 AT SFO..... | 119 |
| TABLE 4-15 TRANSITION MATRIX FOR THE TARGET PROBLEM..... | 120 |
| TABLE 4-16 SCHEDULE OF FLIGHTS INCLUDED IN THE TARGET PROBLEM..... | 120 |
| TABLE 4-17 COMPUTATIONAL RESULTS FOR THE TARGET PROBLEM..... | 121 |

Acknowledgements

First, I would like to express my sincere appreciation for the endless advice, guidance, encouragement, and inspiration from my advisor Professor Mark Hansen. I am deeply indebted to Professor Hansen for all that I learned from him, including but not limited to the technical knowledge, research style, writing, the passion to pursue the truth, and good humor. I feel privileged to have had the opportunity to work with him and learn from him. It was the enlightening discussions with him that made my graduate study a hugely positive and enriching experience. His flexibility in my wild scheme to finish up my dissertation while working is greatly appreciated. This work would have been impossible without his help and dedication.

I would like to thank two other members of my committee, Professor Samer Madanat and Professor Andrew Lim. It was from their courses I first learned about dynamic programming and other techniques I used heavily in this research. Professor Madanat's comments about the dissertation were invaluable and truly strengthened it. I am also very grateful to Professor Lim for the helpful discussions along the way during the development of the core of Chapter 4.

I would also like to acknowledge the help and comments on my research from Professor Carlos Daganzo, Professor Shmuel Oren, Professor Martin Wachs, Professor Jacob Sagi, Professor Alexander Skabardonis, and Professor Michael Ball of the University of Maryland. Their comments and suggestions all helped strengthen the research and became part of this work.

I would like to express my appreciation to my fellow graduate students. Especially, I would like thank Avijit Mukherjee for paving the way and allowing me benefit from his

experience on the ground holding problem and being a great resource for discussions ever since. My experience as a graduate student would have been much poorer without the friendship, collaborations, and enlightening discussions with other fellow Berkeley students. I would like to thank particularly Chieh-Yu Hsiao, Wanjira Jirajaruporn, Yu Zhang, Gautam Gupta, Tatjana Bolic, Charles-Antoine Robelin, Helen Yin, Da-Jie Lin, Alex Huang, Wenbin Wei, Byung Heon Lee, Peng-Chu Chen, Yung-Sheng Chen, Wen-Yu Liao, Blake Lin, Jung-Sheng Lin, I-Sheng Yang, Ellen Yeh, and Edward Huang.

I would like to express my gratitude to my managers at IBM. I must thank Maureen Zoric for her encouragement and support at a key moment, and granting me a long educational leave of absence. I would also like to thank Michael Woo and Kiranmayi Potu for carrying me in their departments and looking after me during my leave. I am especially grateful for the support and flexibility Kiran provided me in finishing my dissertation since I returned to work.

I would like to thank my parents and sister for their constant encouragement throughout my graduate study far away from home. It would not have been possible without their love and support.

Finally, I would like to thank my best friend, Matty Chen, for her unwavering support and encouragement. Matty has been the best project partner I ever had, and an unwearied audience to run ideas by. It has been a long road, but we are finally here.

This work was supported by FAA under contract DTFA01-01-C-00029, through the National Center of Excellence for Aviation Operations Research (NEXTOR), and NASA Ames Research Center under grant NCC2-1428.

CHAPTER 1 INTRODUCTION

In the past two decades, air traffic demand has been increasing rapidly in the United States and Europe. As this demand has not been sufficiently supported by developments in the air traffic system, congestion has become a widespread phenomenon. The primary bottlenecks in the system are the major commercial airports, of which many are often operating close to their maximum capacity. Though the recent decrease in traffic has alleviated the congestion temporarily, it is clear that the problem of congestion has come back and will persist in the long run.

Many longer-term approaches could be taken to help alleviate this problem; however, they take much time and capital investment to materialize. For daily operations, the air traffic system administrators strive to minimize the extent and impact of unavoidable delays for given demand and capacity levels. To accomplish this, the air traffic flow should be matched, as closely as possible, to the available capacity over time and across various components in the air traffic system.

The control of air traffic flow is complicated by the uncertainties in the system. The traffic demand would not necessarily arrive as scheduled due to flight cancellations. The capacities at the airports and en-route airspaces are often lowered because of inclement weather and other incidents. Conservative traffic flow management may lead to undesired underutilization of the scarce resources in the system. The more aggressive strategies, on the other hand, need to be planned with caution because the safety of air travel cannot be compromised. Many flow-control strategies have been developed, but there is still much to be done to optimally respond to the uncertainties in a dynamic fashion. Hence, this

research aims to develop methodologies to advance the support of decision-making in air traffic flow management under uncertainties.

In the following sections of this chapter, we provide the background of this research, including a description of the air traffic system, the primary flow-control strategy, current decision-making paradigm, and emerging technologies. At the end of this chapter, we describe the objective and approaches of this study.

1.1 Overview of the Air Traffic System

In regions where the Air Traffic Management (ATM) system is reasonably well-developed, the successive phases of a typical airline flight conducted under instrument flight rules (IFR) between two sizable airports are controlled by three types of facilities. In generic terms, these are the *airport traffic control tower*, the *terminal airspace control center*, and the *en-route control center*. Using this ATM system, air traffic controllers coordinate the requests for use of the airspace, the runways, and the airports based on safety regulations.

In addition to the ATM system whose focus is on maintaining the safety of aviation, air traffic flow management (ATFM) systems are developed to reduce the cost of delays to airlines and other airspace users, and to achieve better utilization of airports and ATM resources. The objective of an ATFM system is to prevent overloading of airports, ATM facilities and services that might affect safety, and to minimize the economic impact on aircraft operators due to air traffic congestion. This is accomplished by adjusting the flows of aircraft such that the demand matches, as well as possible, the capacity at the airports, in terminal airspace, and in en-route airspace. In the United States, the Air

Traffic Control System Command Center (ATCSCC) has the role of national coordinator in the ATFM effort. Under the direction and control of this centralized unit (ATCSCC), local Traffic Management Units (TMU) operate at each of the regional en-route control centers and at the major terminal airspace control facilities. In addition to implementing directives from ATCSCC, the regional and local TMU may also take actions of a more limited scope to relieve problems of a local nature.

As the air traffic is mostly contributed by scheduled air transportation services, the demand-capacity imbalance usually arises when the available capacity is seriously and/or persistently reduced for a period of time. Future operating conditions, such as the available runway capacity at an airport are often subject to a high level of uncertainty. The available capacity for arrivals at an airport can be reduced substantially under adverse weather conditions such as snow, ice, low cloud ceilings, marine stratus, or thunderstorms. Similarly, the en-route sector capacity, which is dependent on the number of aircraft that an air traffic controller can manage at one time, the geographic location, and the weather conditions, can be greatly lowered due to convective weather activities among other factors. Operating in such an inherently stochastic system, ATFM must adopt strategies that take into consideration the level of uncertainty associated with the key parameters and are flexible so they may be easily revised as new information becomes available.

To deal with the demand-capacity imbalance, ATFM employs three principal types of control measures: *ground holding*, i.e. intentionally delaying an aircraft's takeoff for a specified amount of time, trading airborne delay with ground delay; *rerouting*, i.e. changing or restructuring some flight routes to modify the distribution of traffic flows

and possibly avoid regions affected by inclement weather; and *metering*, i.e. controlling the rate at which traffic crosses some specified spatial boundaries by adjusting the spacing between aircraft. Among these control measures, ground holding is the most drastic, as it controls the number of aircraft flowing through the ATM system, while metering is the most tactical in nature.

The development of advanced ATFM systems in the United States and Europe in the past two decades has already had an enormous impact on air traffic management and airport operations. On the other hand, ATFM has also been criticized for occasionally contribute to slowing down air traffic operations and exacerbating, rather than reducing, delays. These deficiencies are gradually being corrected as decision support technologies improve, and this is the area that needs much endeavor.

1.2 The Ground Delay Program

A Ground Delay Program (GDP) is initiated by ATCSCC in the United States to control air traffic volume inbound to an airport where the projected traffic demand is expected to exceed the acceptance rate of an airport for a period of time. Under a GDP, the demand-capacity imbalance is managed through delaying some flights prior to their departure when the destination airport expects shortfall in arrival capacity. The rationale is that it is both less expensive and safer to absorb unavoidable delays on the ground than in the air.

Each GDP is specific to a destination airport, so several GDPs could be run on any single day simultaneously. Boston-Logan, Chicago-O'Hare, New York-LaGuardia, New York-Newark, and San Francisco-International are the airports with the highest GDP

incidence. In 2000, on average about 2.5 GDPs were initiated per day in the United States. The duration of a GDP may be as long as 12 hours or more due to persistent adverse weather condition while a more typical duration is a few hours.

At the onset of a GDP, a list of in-coming flights, ordered by the most recent estimated time of arrival (ETA) is formed. The ATCSCC “run a GDP” by assigning new times of arrival—controlled arrival times (CATs) to flights on the list using a “first scheduled, first served” policy. By this policy, the arrival slots are allocated according to the original schedule of flights (which is called ration by schedule), and by scheduling the CTA’s in the same order as the ETA’s. International flights, general aviation flights, and flights already airborne prior to the GDP are exempt from the program. In addition, other flights, such as flights from certain geographical locations, may be chosen to be exempt from a GDP.

The implementation of GDP was generally met with hostility by the airlines because the centralized resource allocation exposed airline operations to inaccurate estimations on the part of the ATCSCC and represented government intervention. While a GDP is based upon an economic premise, the airlines were against letting the FAA make economic decisions on behalf of them since the FAA does not have the information needed to make certain decisions. Hence, a joint effort between government agencies and the airline industry known as Collaborative Decision Making has arisen.

1.3 The Paradigm of Collaborative Decision Making

The Collaborative Decision Making (CDM) initiative began in the early 1990’s. The philosophy behind CDM is that improved data exchange and communication between air

traffic managers and air traffic system users will lead to better decision-making, and that decisions should be distributed to the parties with the best information and ability to make them. Under this paradigm, air carriers contribute more input into ATFM and have greater control over their own operations.

The first focus of the CDM initiative was GDP enhancement. Under CDM, GDP operations are executed using a cycle of feedback between the service provider and users of the air traffic system. Once a demand-capacity imbalance is forecasted, the ATCSCC implements a GDP and delays are allocated to in-coming flights. Airlines, who assume the ownership of arrival slots assigned to their flights, react to the new situation with a cancellation/substitution process and inform the ATCSCC about their updated schedule. This process allows the airlines greater control over the economic impacts of the GDP. The ATCSCC runs a ration-by-schedule (RBS) algorithm and a compression algorithm to reallocate arrival slots which no flight is assigned. At the end of this process, a new iteration is started. The ATCSCC can make an accurate situational assessment at a data exchange cycle only if the airlines supply correct and up-to-date data of cancellations and revised ETA's. The fairness ensured by the RBS algorithm and the potential benefit of better on-time performance provided by the compression algorithm elicit the incentives from the airlines to participate in the CDM with their most updated information.

Starting with successful pilot programs using GDP with CDM at San Francisco-International and New York-Newark, all GDPs at all airports in the United States have been conducted via CDM since September 1998. As the paradigm of ATFM has moved away from centralized control of air traffic to decentralized decision making, many of the

ATFM models proposed in the literature became obsolete, and better ATFM strategies are needed to fully utilize the ever-growing technologies.

1.4 Emerging Technologies

1.4.1 Probabilistic Forecasts

Airport and en-route airspace capacities are subject to substantial uncertainty as they depend on stochastic weather conditions. Traditionally, the threats of severe weather are predicted by categorical forecast, which depicts the incident in terms of “risk” (low, moderate, high). Unlike categorical forecasts, probabilistic forecasts provide the likelihood of events directly to the users such that they can make decisions without having to first gauge the uncertainty implied in categorical statements. The importance of probabilistic forecast for planning under uncertainty has been widely recognized in recent years. Moving toward this direction, for example, the Storm Prediction Center at the National Oceanic and Atmospheric Administration has been producing experimental subjective probabilistic convective outlooks since 1999. Also, on-going efforts have been going into producing probabilistic forecast of marine stratus impacts at San Francisco International airport. (Wilson, 2004) From the trend and progress of development in forecasting technologies, it is expected that probabilistic forecast for both airport and en-route airspace capacities will be available for ATFM decision making in the near future.

1.4.2 Multi-center Traffic Management Advisor

The Traffic Management Advisor (TMA) is an air traffic control automation system currently in use in eight ARTCCs to enable time-based metering to busy airports within their airspace (FAA, 2005). Time-based metering (TBM) is a traffic flow control

technique used to stretch out arrival demand during periods when the capacity of some National Airspace System (NAS) resources, such as a major airport, is expected to be exceeded. This technique has proven effective in smoothing arrival flow, lessening vectoring for landing sequence and reducing incidence of no-notice holding. As opposed to strategies such as miles-in-trail spacing or managed arrival reservoirs, TBM assigns crossing times for aircraft at points along its route of flight. These assigned times of arrival are intended to delay aircraft so as to provide an efficient and orderly flow of traffic into the constrained resource while also fully utilizing the available capacity of the resource.

The Multi-center Traffic Management Advisor (McTMA), currently a pilot project deployed at four northeast centers in the United States, is an enhanced version of TMA with networking capabilities. ARTCCs that equipped with McTMA can communicate and share information with neighboring centers with McTMA systems. This system environment facilitates collaborative traffic management among traffic managers and controllers on a regional basis. As such, McTMA utilizes distributed scheduling to extend the metering horizon beyond the effective range of a single TMA. McTMA promises to bring time-based metering into the mainstream of ATFM techniques, and enables a more regional or national approach to managing air traffic problems. With the emerging technologies in mind, new ATFM strategies ought to be designed to take advantage of the new capabilities.

1.5 Optimization Models versus GDP in Practice

As the increase of air traffic demand is not sufficiently supported by the infrastructure of the air traffic system, effective and practical air traffic flow management strategies are needed to lower the cost of delay in a safe manner. Although many sophisticated optimization models have been developed for ATFM problems, air traffic managers still direct the system using algorithms which are relatively primitive. The reasons for their reluctance to adopt the optimization-based methods are summarized as follows.

1. Many optimization models are incomplete and obsolete. Many models in the literature do not plan for uncertainties in the system. When demand-capacity imbalance is stochastic, deterministic models for addressing that imbalance are obviously unacceptable for practical use. Moreover, many models were developed for the setting of centralized control in which the output is the optimal action plan for each flight. These models are obsolete for the system which is currently operating under the Collaborative Decision Making paradigm.
2. There is a gap between theory and practice. The lack of research into how the latest optimization models can be applied in real-world operations leaves the applicability of the solutions unclear. For example, a number of the latest models require capacity scenario trees as input. However, the methodology for generating such scenario trees, and how the scenario-based strategies can be implemented were undefined.

3. There is little evidence that these optimization-based strategies can reduce the cost of delay in practice. Many practitioners are concerned that these models may compromise safety or reduce capacity utilization.

In addition to the concerns on the application side, there are also issues of a more fundamental nature in the latest models. First, the latest stochastic models plan for uncertainties based on a finite set of scenarios. The major drawback of a scenario-based strategy lies in the high likelihood of mismatch between the actual situation and the nominal scenarios in plan. This shortcoming almost guarantees a suboptimal decision. Second, the stochastic models optimize the expected cost of delay. Although it has been common practice to plan for uncertainties using expectation, the best average performance not necessarily translates to the objective of the manager of the air traffic system. Operating within a conservative culture, the air traffic manager is likely to put more weight on the down-side risk than on the up-side gain.

1.6 Research Objective and Approach

Air traffic delay is costly to every entity involved in air travel. The economic impact on air traffic service providers, airlines and passengers is profound. The goal of this research is to improve the ability of air traffic managers to handle uncertainty and incorporate probabilistic forecast information in ground delay programs. In particular, we investigate ways to advance the support of decision-making under uncertainty in GDPs for a single destination airport. We explore methods to model the stochasticity in GDP operations and mechanisms that respond to conditions dynamically such that the overall system performance is optimized.

GDPs mitigate the capacity-demand imbalance by assigning ground delays. The ground-holding problem (GHP) is that of ground-holding flights to minimize the overall cost of ground and airborne delays. The ground-holding problem for a single destination airport is termed the Single Airport Ground Holding Problem (SAGHP). We approach the SAGHP by first investigating the implementability of the latest scenario-based SAGHP models. Such models have been studied using hypothesized airport capacity scenarios or scenario trees, but a method to derive scenarios and develop scenario trees from real-world data has yet to be developed. To examine the effectiveness of scenario-based models and to bring them toward practical use, we propose methodologies for developing airport capacity scenario trees. The ultimate value of these methodologies, and of the models they support, depends upon the benefits they provide in a real-world setting. In this research, we assess these benefits by evaluating the delay costs if the air traffic flow management strategies obtained from the models were used in the face of actual arrival demand and capacity evolution.

As mentioned in the previous section, one of the primary shortcomings of the scenario-based models is that they assume a limited number of capacity scenarios when in reality there is a much larger set of possibilities for capacity evolution. The number of scenarios is limited for the integer programming problem to be solvable using the latest solvers. In addition, scenario-tree-based models impose a scenario tree structure when in reality improved information about future capacity is obtained continually rather than at a few discrete branching points. We consider a “scenario-free” Markovian approach for characterizing the capacity evolution to avoid these inadequacies. The process of making ground-holding decisions in a SAGHP can naturally be described as sequential decision

making. In this sequential decision making problem, the ground holding decisions are responses to the capacity uncertainties at the destination airport, and the goal is to find the optimal policy to react to the most updated information. We formulate this sequential decision making model using a dynamic program. One major concern of modeling a large-scale problem using dynamic programming lies in the *curse of dimensionality*. Hence, this research proposes ways to manage this curse.

The remainder of this dissertation is structured as follows. In Chapter 2, recent developments in the literature are briefly reviewed. In Chapter 3, we investigate the implementability of scenario-based models and assess their performance in idealized and real-world settings. Chapter 4 presents the scenario-free sequential decision making model and the solution algorithms. Chapter 5 compares the performance of the scenario-based and scenario-free models. Finally, in Chapter 6, we present our conclusions and provide directions for future research.

CHAPTER 2 LITERATURE REVIEW

Air traffic flow management has been an active area of research since the late 1980s. Most of the early work focuses on the ground holding problem, which arises in the face of constrained airport landing capacity. In recent years, the increase of traffic volume also gradually turns the capacity of the en-route airspace into another constrained resource in the air traffic system, and has drawn attention in the research community. In this chapter, we review the work on single destination airport ground holding problem in Section 2.1. We provide detailed description for a couple of SAGHP models from the literature, since they are heavily referred to in this thesis. In Section 2.2, we summarize the efforts into solving multiple destination airport ground holding problem. As a closely related area of work, we also review recent developments in the air traffic flow management problem with en-route capacity constraints in Section 2.3. Finally, in Section 2.4, we show our findings from studying literatures on scenario generation methods for multi-stage stochastic programming.

2.1 Models of Single Destination Airport Ground Holding Problem

The flow management problem was first proposed by Odoni (1987) as a problem of finding an optimal delay strategy so that the total expected delay cost is minimized. Andreatta and Romanin-Jacur (1987) were the first ones to have investigated in-depth an algorithmic approach to determine the optimal ground delay for the flights during a GDP. The simplified single-time period problem in the study determines optimal ground-holding strategy to minimize total delay costs for n flights to a single destination airport

using a dynamic programming (DP) approach. Following this, Terrab and Odoni (1993) addressed the single destination airport ground holding problem under deterministic and probabilistic airport capacities. For the deterministic case, they formulated the problem as a minimum cost network flow problem in which the total ground delay cost is minimized. For the stochastic case, they extended the single-period dynamic program in Andreatta and Romanin-Jacur (1987) to multiple-period, and proposed several heuristics to ease the computational burden due to the curse of dimensionality of DP. Richetta and Odoni (1993) addressed the stochastic version of SAGHP using a single-stage stochastic linear program. In this model, the decisions are applied to classes of flights, where a class is a set of flights scheduled to arrive during a particular time period. The solution from this model is the number of flights in each class that are subject to ground and/or airborne delay assignments.

As the capacity forecast is updated throughout the day, it is logical to reevaluate decisions made earlier but not yet executed when there is an information update. Terrab and Paulose (1992) proposed a stochastic programming model to minimize delay costs based on probabilistic forecast of the capacity at the destination airport. The system was modeled using a two-stage stochastic program with recourse, where the start of the second stage is marked by the time when the future capacity becomes known with certainty. In this study, they found that the dynamic policies consistently penalize short-haul flights in favor of long-haul flights and that there is little value of early capacity information. The issue of equity was addressed by limiting the amount of delay that can be imposed on a single flight or by assigning arrival times on a first-come, first-served basis. Richetta and Odoni (1994) suggested a dynamic model for the multi-period

stochastic SAGHP. This model is formulated using stochastic linear programming with recourse, where the dynamic evolution of the capacity forecasts and the information updates is modeled through a probabilistic scenario tree. In this model, the ground delays are not assigned once; rather, they are assigned to groups of flights as their scheduled departure times approach. The amount of delay assigned varies according to available information on capacity conditions at the destination airport. However, once the ground delays are assigned to flights, they are not revised later even though some flights may not have departed from their origin airports. Built upon this work, Mukherjee and Hansen (2005) proposed an enhanced model which allows for the revision of ground delays in response to updated information for flights that have not yet departed from their origin airports. This is achieved by relaxing certain constraints in Richetta and Odoni's (2004) model. Numerical studies using hypothesized scenario trees showed favorable performance results for this enhanced model, however, a generic methodology to derive scenario trees from real-world data was not provided.

In addition to the primary lineage of development of the SAGHP models, some presented extensions to the SAGHP while others addressed the SAGHP with different approaches. Hoffman and Ball (2000) suggested five different models of the SAGHP in the presence of banking constraints to accommodate the hubbing operations of major airlines in the United States. These constraints describe the airlines' practice of landing certain groups of flights within fixed time windows. For two of the models, they showed that the constraints induce facets of the convex hull of integer solutions.

Panayiotou and Cassandras (2001) suggested a sample path approach for solving SAGHP. Unlike most of the other models, which are time-driven, they characterized the

system using an event-driven model. Using finite perturbation analysis, they determined the optimal ground delay by minimizing the incremental cost of an additional flight. Willemain (2001) investigated the sensitivity of ground holding assignments to the uncertainty of future airport capacity and the value of cancellation as an alternative to ground delay. Grignon (2002) explored the use of utility function as the objective function for a static stochastic optimization formulation of SAGHP.

Ball, Hoffman, Odoni and Rifkin (2003) proposed a scenario-based static stochastic model for the SAGHP in the CDM environment. This model outputs the aggregate number of flights that should be allowed to arrive for each time period, rather than individual planned arrival time for each flight. Several implementation issues in CDM are addressed by ignoring the identity of individual flights during the strategic planning phase. This approach addressed, for example, the prerogative of airlines to cancel and substitute flights, and the need for fast computation involving thousands of flights.

As the performance of the scenario-based static model of Ball, Hoffman, Odoni and Rifkin (2003) and the scenario-based dynamic model of Mukherjee and Hansen (2005) are discussed extensively in this dissertation, the following subsections provide a detailed review of these two models.

2.1.1 Ball et al. Static Model

The Ball et al. static model requires as input a set of capacity scenarios and their probabilities. Using the scenario probabilities, the static stochastic ground holding (SSGH) problem solves for a profile of planned arrival rates (PARs) that minimizes the expected cost of airborne and ground delay. The following notations are used in the formulation. Let T be the number of time periods considered in the planning horizon, D_t the number of

flights scheduled to arrive in time period t , Q the number of arrival capacity scenarios, p_q the probability of scenario q to occur, $M_{q,t}$ for $1 \leq q \leq Q, 1 \leq t \leq T$, the arrival capacity (AAR) of the airport during t if scenario q were realized, c_g the cost of ground holding per plane per time period, and c_a the cost of airborne holding per plane per time period. The decision variables are A_t , for $1 \leq t \leq T+1$, the number of aircraft planned to land in time period t , *i.e.* the PARs. Also defined are the following auxiliary variables G_t and $W_{q,t}$, for $1 \leq t \leq T+1, 1 \leq q \leq Q$, where G_t is the number of flights that could arrive during time period t but are subject to ground holding at origin airports for one or more time periods, and $W_{q,t}$ is the number of flights to experience airborne holding during time period t . The problem is stated by the following integer program:

$$\text{(SSGH) } \min \sum_{t=1}^T c_g G_t + \sum_{q=1}^Q \sum_{t=1}^T c_a p_q W_{q,t}$$

$$\text{s.t. } A_t + G_t - G_{t-1} = D_t \quad t=1, \dots, T+1 \quad (G_0 = G_{T+1} = 0) \quad (2-1)$$

$$A_t + W_{q,t-1} - W_{q,t} \leq M_{q,t} \quad t=1, \dots, T+1 \quad q=1, \dots, Q \quad (W_{q,0} = W_{q,T+1} = 0) \quad (2-2)$$

$$A_t \in Z_+ \quad \forall t, \quad W_{q,t} \in Z_+ \quad \forall t, q, \quad G_t \in Z_+ \quad \forall t \quad (2-3)$$

The objective is to minimize the sum of ground and the expected airborne delay costs, the first and second terms in the objective function, respectively. Constraint set (2-1) is for flow conservation, and ensures that the current demand (D_t) and the backlogged demand (G_{t-1}) are either served in this period (A_t) or put in the queue (G_t). Constraint set (2-2) can be viewed as flow conservation constraint at equality—it ensures that once the sum of the assigned service load (A_t) and the backlogged demand ($W_{q,t-1}$) exceeds the

capacity ($M_{q,t}$), the excess is put in the queue ($W_{q,t}$), while the inequality allows for the case that the airport could have slacks in the capacity.

Clearly, increasing PARs reduces the amount of ground holding but increases the risk that there will be insufficient capacity when the flights arrive at the destination, necessitating airborne holding. Thus, as the cost of airborne holding increases relative to the cost of ground holding, the optimal strategy becomes more conservative, and features lower PARs.

2.1.2 Mukherjee-Hansen Dynamic Model

Dynamic models such as Mukherjee-Hansen use evolving information about capacity to make ground holding decisions. In the context of a scenario tree, evolving information is represented in the form of a branching point that occurs at a particular time of day. When a branching point is reached, we acquire new information about which of the branches is being realized on a particular day. Based on this information, ground-hold decisions for flights that have not yet taken off can be made or revised. The Mukherjee-Hansen model recognizes these opportunities for recourse in making the original decisions. For example, a flight scheduled to depart just before a branching point might be held so that the release decision can be made with the benefit of the information that will soon become available.

As in previous literature (Richetta and Odoni, 1994), the tree structure is incorporated in the optimization by two means. First, decision variables—release times of individual flights—are scenario-specific. In this respect, the problem is solved as though which scenario will be realized is known. To capture limitations in this information,

coupling constraints are imposed. The constraints force release times for different scenarios to be equal if these scenarios are all still possible at the time of release.

In the Mukherjee-Hansen formulation for the dynamic stochastic ground holding (DSGH) problem, the parameters T , Q , p_q , $M_{q,t}$, c_g , c_a , and $W_{q,t}$ are defined as those in the Ball et al. static model. Additionally, a_f and d_f are used to denote the scheduled arrival and departure time of flight f , respectively. The decision variables $X_{f,t}^q$ and auxiliary variables $Y_{f,t}^q$ are defined as follows:

$$X_{f,t}^q = \begin{cases} 1 & \text{if flight } f \text{ is assigned to arrive in or before time period } t, \\ 0 & \text{otherwise.} \end{cases} \quad \forall f, \forall q, \forall t.$$

$$Y_{f,t}^q = \begin{cases} 1 & \text{if flight } f \text{ is released for departure in or before time period } t, \\ 0 & \text{otherwise.} \end{cases} \quad \forall f, \forall q, \forall t.$$

As such, $X_{f,t}^q - X_{f,t-1}^q$ is 1 only if flight f is assigned to arrive in time period t , and $X_{f,t}^q$

and $Y_{f,t}^q$ are related as follows:

$$Y_{f,t}^q = \begin{cases} X_{f,t+a_f-d_f}^q & \text{if } t + a_f - d_f \leq T, \\ 1 & \text{otherwise.} \end{cases} \quad \forall f, \forall q, \forall t,$$

The DSGH problem is modeled by the following integer program:

$$\text{(DSGH) } \min \sum_{q=1}^Q p_q \left\{ c_g \sum_{f=1}^F \sum_{t=a_f}^{T+1} (t - a_f)(X_{f,t}^q - X_{f,t-1}^q) + c_a \sum_{t=1}^T W_{q,t} \right\}$$

$$\text{s.t. } \sum_{f=1}^F (X_{f,t}^q - X_{f,t-1}^q) + W_{q,t-1} - W_{q,t} \leq M_{q,t} \quad \forall q, t = 1, \dots, T+1, (W_{q,0} = W_{q,T+1} = 0) \quad (2-4)$$

$$X_{f,t-1}^q \leq X_{f,t}^q \quad \forall q, \forall f, t = 1, \dots, T+1, (X_{f,T+1}^q = 1, \forall q, \forall f) \quad (2-5)$$

$$Y_{f,t}^{S_1^i} = Y_{f,t}^{S_2^i} = \dots = Y_{f,t}^{S_{N_i}^i} \quad \forall f, b_i \leq t \leq e_i, i = 1, \dots, n, \quad (2-6)$$

$$X_{f,t}^q \in \{0,1\}, Y_{f,t}^q \in \{0,1\}, \forall q, \forall f, \forall t, \quad W_{q,t} \in Z_+, \forall t, q. \quad (2-7)$$

Constraint set (2-4) is the same as constraints (2-2) in the static model, but expressed with the decision variables $X_{f,t}^q$. Constraint set (2-5) ensures that $X_{f,t}^q$ is non-decreasing in t —that is, once this bit is turned on, it remains on. Also, by $X_{f,T+1}^q = 1$, it is ensured that all flights arrive at the destination airport by the end of the planning horizon.

Constraint set (2-6) is a set of coupling constraints on the release time for flights under different scenarios. These constraints ensure that before a set of scenarios branch out to distinct scenarios, the ground holding assignments are identical across these scenarios. In this set of constraints, we let n denote the number of branches in the scenario tree that contain more than one scenario, b_i the beginning time period of the multi-scenario branch i , e_i the end period of the subset i , N_i the number of scenarios in branch i , and S_k^i the k -th scenario in branch i . For example, for the tree in Figure 2-1, there are two multi-scenario branches—branch 1 contains scenario 1, 2, and 3; branch 2 contains scenario 2 and 3. Each branching point corresponds to the existence of a subset. For scenario 1, 2, and 3, the ground-hold assignments must be the same from $t = 1$ to $t = 3$. For scenario 2 and 3, the ground-hold assignments must still be the same from $t = 4$ to $t = 6$. Expressed mathematically, we have the following set of constraints:

$$\begin{cases} Y_{f,t}^1 = Y_{f,t}^2 = Y_{f,t}^3 & \forall f, 1 \leq t \leq 3 \\ Y_{f,t}^2 = Y_{f,t}^3 & \forall f, 4 \leq t \leq 6 \end{cases}$$

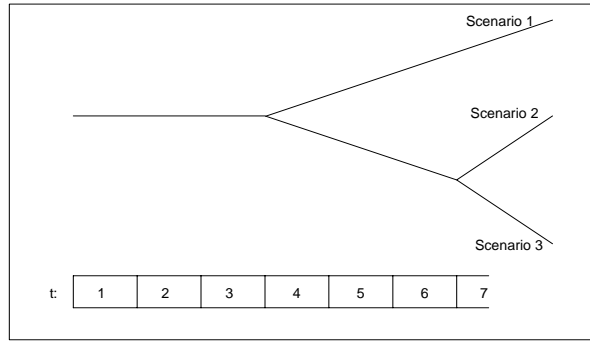


Figure 2-1 A Simple Scenario Tree for the Illustration of the Coupling Constraints

2.2 Models of Multiple Destination Airport Ground Holding Problem

Vranas, Bertsimas, and Odoni (1994) were the first to extend the GHP to the setting of a network of airports. The multi-airport ground holding problem (MAGHP) considers connected flights and the network effect of delay propagation to downstream flight legs. They proposed a dynamic model—a model which provides updated ground holding decisions as weather changes and as aircraft fleet becomes available or unavailable. Integer programming (IP) formulations for the MAGHP with both deterministic and probabilistic capacity forecast were presented. In the deterministic case, the dynamic response is achieved by reassessing delay assignment using the IP model at each decision period. This formulation was also extended to include considerations of flight cancellations and interdependent departure and arrival capacities. In the probabilistic formulation, they consider capacity forecasts that take the forms of scenarios, each coming with the probability of realization. Again, the dynamic response is provided by re-solving the IP model at each decision point, while the decisions that have already been executed being fixed in the model. They acknowledged that their approach will not necessarily give the genuine dynamic optimum, but took this approach rather than

stochastic programming for their concern over the lengthy computation the latter might take.

Navazio and Romanin-Jacur (1998) suggested an IP model to solve the deterministic multi-airport ground holding problem. Their model minimizes the overall delay cost subject to airport arrival capacity, connections, and time constraints imposed by airlines. It is assumed that the flying time of each flight is deterministic, and hence any delay suffered by a flight is due to ground holding at the origin airport. Due to excessive computation time required by the exact method, they provided a heuristic to shorten the average running time based on the limited-resource Critical Path Method. Following this work, Brunetta, Guastalla, and Navazio (1998) improved the heuristic in Navazio and Romanin-Jacur (1998). Later, Filar, Manyem, Visser, and White (2003) presented an enhanced version of the Navazio and Romanin-Jacur (1998) model. This model includes two additional features—cancellations if the arrivals would otherwise be delayed beyond a reasonable upper bound and penalties for arrivals during the curfew hours of the airport. They also proposed a pseudo-dynamic strategy to respond to a change of phase for the airport. This strategy adapts flight schedules to respond to an announced capacity emergency by modifying the objective function such that ground delay and cancellation are only allowed for flights not yet in transit.

2.3 Air Traffic Flow Management Problem with En-route Capacity Constraints

The models in the previous sections do not consider en route congestion. While the NAS in the United States is constrained largely by the capacity of airports and terminal airspace, increasing traffic volume and complexity is leading to more congestion in the en

route airspace. The en route congestion is generally considered at the sector level. The sector capacity is dependent on the number of aircraft that the air traffic controllers can manage at one time, the geographic location and the convective weather condition. Control measures such as miles-in-trail restrictions, speed controls, or rerouting of aircraft can be employed to alleviate the en route airspace congestion. Addressing both the temporal and spatial concerns, the ATFM with en route capacity constraints is a more difficult problem.

Wang (1991) considered the global flow control problem with congested NAS elements and suggested a dynamic programming-based framework for modeling. Helme (1992) formulated a multi-commodity minimum cost flow model over a space-time network. This formulation considers restrictive airway capacities and multiple destinations. However, this method was not fully tested and it was expected that there would be severe dimensionality problems. Lindsay, Boyd, and Burlingame (1993) proposed an IP formulation that tracks a flight as it passes from fix to fix in the airspace. This model has been shown to require long solution time for small problem instances. While uncertainties are not addressed in the model, the authors proposed to treat uncertainties by re-solving the model.

Bertsimas and Stock Patterson (1998) proposed a 0-1 IP model for the ATFM problem with en route capacity constraints. They claimed that this model is a generalization of ATFM problems such that the ground holding problem is a special case of it. This model considers airport capacities, sector capacities, and connected flights, but does not address routing decision problems. The formulation is strong in the sense that many of the constraints are facet-defining for the convex hull of integer solutions.

Andreatta, Brunetta, and Guastalla (2000) proposed to solve the same using an exact algorithm, which is an integration of a heuristic algorithm with an IP model. Through computational experiments, they showed that the exact algorithm provided solutions in a much shorter computation time.

Complementing their previous work, Bertsimas and Stock Patterson (2000) addressed the problem of aircraft rerouting when faced with dynamically changing weather conditions. The problem was modeled using a dynamic network flow formulation with some problem-specific constraints. The model determines the best routes for the aircraft to follow, as well as the amount of ground holding and speed adjustment. However, the problem was formulated in a deterministic setting, where the uncertainties were not modeled.

In addition to the primary lineage of development of the ATFM models with en route capacity constraints, some studies addressed the problem using different approaches. Kleinman, Hill and Ilenda (1998) proposed to apply simultaneous perturbation stochastic approximation (SPSA) to determine the optimal ground delay for each flight in a network of airports. In their solution framework, the SPSA algorithm minimizes the delay cost while the dynamics in the air traffic system is modeled using a detailed air traffic simulation software package, SIMMOD.¹ Oussedik, Delahaye, and Schoenauer (1999) applied genetic algorithm to the air traffic congestion problem. They tried to reduce congestion in the most overloaded sectors by assigning each flight an appropriate departure time and route. Modeling in a microscopic manner, the objectives of the

¹ SIMMOD is an airport and airspace simulation model which takes into account the route and sector capacity constraints as well as wind speeds. It is developed by ATAC Corporation in Sunnyvale, CA.

airlines were incorporated in the decisions. However, this model suffers from high computation time.

So far, the efforts to solve the ATFM problem with en route capacity constraints have not provided a model which characterizes the system realistically and, at the same time, solves the problem within time acceptable for practical use. Hence, some recent studies moved away from flight-specific strategic control to aggregated approaches. Menon, Sweriduk and Bilimoria (2002) explored an Eulerian approach to model air traffic flow using spatial control volumes. Roy, Sridhar, and Verghese (2003) proposed a framework to study the ATFM problem using an aggregate stochastic dynamic model. This model uses Poisson processes to model the uncertainty of the movements of aircraft and tracks the number of aircraft in each Center at each time period. Extending this time-invariant model to a time-varying model, Sridhar, Soni, Sheth, and Chatterji (2004) provided a model with slowly varying transition matrices and Gaussian departure to better represent the traffic behavior at the Center level. However, the use of this class of models for strategic flow management has not been presented.

2.4 Scenario Generation Methods for Stochastic Programming

The generation of a scenario tree, which is the representation of the underlying random process in the system, is a major issue in application of scenario-based multistage stochastic programs. Most methods proposed in the literature were focused on supporting the application of stochastic programs in financial portfolio management.

When the distribution functions of the variables are unknown, some suggested construct discrete distributions that satisfy certain statistical properties, such as the

moments and percentiles. This can be performed through minimizing a measure of distance between the statistical properties of the constructed distribution and the specifications, as described in the work of Hoyland and Wallace (2001) and Gulpinar, Rustem and Settergren (2004). This class of scenario generation method is called “moment matching”. However, as criticized by Kaut and Wallace (2003), when the goal is to solve a stochastic program, a scenario tree should be judged by the quality of the optimization solution it brings to us.

Some others, including Kaut and Wallace (2003), proposed using sampling-based methods to generate scenarios. For example, the *conditional sampling* method samples several values from the stochastic process at every node of a scenario tree. This can be performed by sampling directly from the distribution of the stochastic process. Since sampling is performed at every stage, this method suffers as the scenario tree grows exponentially in size with the planning horizon.

Pflug (2001) pursued a different direction and used a stochastic approximation technique to generate a scenario tree in which the error in the objective function of the optimization model is minimized. However, this methodology works only for univariate processes.

Another breed of methods starts the scenario tree generation process by producing complete scenario paths. From a set of paths, the scenario tree is derived through binding the paths together. Dupacova Consigli and Wallace (2002) described a sequential importance sampling-based method to generate scenarios. In their algorithm, the structure of the scenario tree, i.e. the time of branching and the number of descendent branches, is pre-specified. It is reasonable for their application in finance since the decision stage is

associated with, for example, the expiration date of an option. But clearly it is not the case for our application in air traffic management. Along the same line, Gulpinar et al. (2004) described an approach using simulation and randomized clustering for scenario tree generation. They proposed to simulate price scenarios using sequential or parallel simulations based on the probability density function of the price of tomorrow given the price of today.

As Kaut and Wallace (2003) pointed out, the goodness of a scenario-generation method is coupled with the decision model it serves. Though many methods have been proposed for scenario tree generation, most of them are designed specifically for financial portfolio allocation problems. The stochastic programming model for SAGHP attains its capability to recourse by suggesting actions based on a probabilistic scenario tree. Clearly, the scenario tree is a crucial component in the stochastic programming SAGHP models and the quality of the model solution hinges on it. For the aforementioned reasons, we do not find the scenario tree generation methods developed for the financial portfolio allocation problems appropriate for our application. Hence, it is necessary to develop a method to construct scenario trees that is suitable for the SAGHP.

CHAPTER 3 SCENARIO-BASED MODELS FOR SINGLE AIRPORT GROUND HOLDING POLICY PROBLEM: FROM THEORY TO PRACTICE

The Single Airport Ground Holding Problem has been addressed by deterministic models in the seminal studies (Odoni, 1987) followed by models that consider the uncertainty in airport arrival capacity using stochastic integer programming under static (Ball et al., 2003) and partially dynamic (Richetta and Odoni, 1994) settings. In the Ball et al. (2003) model, ground delays are assigned at the beginning of the planning horizon and not revised later. In the Richetta-Odoni model (Richetta and Odoni, 1994) ground delays are assigned to groups of flights as their scheduled departure times approach, according to the latest information on capacity conditions at the destination airport. However, once the ground delays are assigned to flights, they are not revised later even though some flights may not have departed from their origin airports as planned. In a more recent development, the Mukherjee-Hansen model (Mukherjee, 2004; Mukherjee and Hansen, 2005), allows for the revision of ground delays for flights that have not yet departed in response to updated information. This allows for “wait-and-see” strategies in which flights are held in anticipation of upcoming information.

The Ball et al., Richetta-Odoni, and Mukherjee-Hansen models are all based on the assumption that arrival capacity evolution can be modeled using a limited number of scenarios. The latter two models also assume that scenarios form trees with branch points over the course of the day when previously similar scenarios become distinct. The

scenarios reported in the literature were constructed for purposes of illustration. This begs the questions of whether and how scenario-based models can be implemented for real-world airports, and how the benefits of such implementation compare to those suggested by the idealized applications reported to date.

This chapter addresses these questions. First, we propose and demonstrate methodologies that can be used to determine airport-specific scenario trees at any airport. Unlike previous studies (Inniss and Ball, 2004) on arrival capacity distributions in the literature, we provide a methodology that does not restrict the patterns to a few pre-postulated forms and organizes the arrival capacity profiles into a probabilistic scenario tree. Additionally, we propose methods for “navigating” through a scenario tree so that at a given time we know (or can intelligently guess) what branch of the tree we are on.

The ultimate value of these methodologies, and of the models they support, depends upon the benefits they provide in a real-world setting. In this research, we assess these benefits by simulating what would happen if the air traffic flow management strategies obtained from static and dynamic models were used in the face of actual capacity profiles, which do not match the idealized capacity scenarios perfectly. We compare the results of static (Ball et al.) and dynamic (Mukherjee-Hansen) optimization models in this regard, and also report on experiments using different methods for applying the dynamic model.

The remainder of this chapter is organized as follows. In Section 3.1, we introduce terminology and identify the challenges in implementing scenario-based air traffic management in a real-world setting. Next, we present methods for developing scenario trees from data, and illustrate these methods using data from several major U.S. airports in Section 3.2. In Section 3.3, we use the scenario trees developed for several airports as

input to two of the scenario-based stochastic optimization models, and compare the performance of these two models under idealized conditions. Section 3.4 discusses the challenges to implementing optimal solutions, methods for overcoming these challenges, and how these challenges and their solutions affect model performance. Finally, Section 3.5 provides concluding remarks for this chapter.

3.1 Using Scenario Trees in Air Traffic Management

We first define the related concepts of capacity profile, capacity scenario, and scenario tree, which are the focus of the subsequent discussion. We define a *capacity profile* as a time series of capacity values for an airport over a day or some part of a day. Capacity profiles are derived from airport acceptance rates (AARs), which specify the number of arrivals an airport can accommodate over a 15-minute period. Thus a capacity profile is a time series of AARs for an airport. We use the term *capacity scenario* to refer to a representative profile that is similar, but not identical, to a set of profiles. If profiles are individuals, then scenarios are species. The process of developing scenarios from profiles will be discussed below. Finally, a *scenario tree* is a set of capacity scenarios with associated probabilities of realization. This set is organized into a tree that branches when initially similar scenarios evolve into distinct scenarios.

Our aim is to develop practical methods for creating and using scenarios trees for air traffic management. To do this, we must address four specific challenges:

Identifying Scenarios

For a scenario-based air traffic management solution to be applicable, we must construct capacity scenarios for an airport. Common sense suggests, and our experience

confirms, that it is very rare for two days to have identical capacity profiles. Thus, to construct useful scenarios it is necessary to identify groups of days with similar, but not identical, profiles. To do this efficiently requires an algorithm. In designing that algorithm we must make the trade off between having many scenarios with little intra-scenario variation within them, or few scenarios with more variation. The number of scenarios recognized also influences the number of instances of any particular scenario we observe, and thus the precision of our estimates of their associated probabilities.

Constructing Scenario Trees

When applying dynamic models such as Mukherjee-Hansen, we require more than just a set of scenarios. We must also combine these scenarios into a probabilistic tree. Specifically, we must identify scenarios that are similar to each other early in the day and then branch out into distinct patterns. We must therefore define criteria for grouping several patterns into one over a certain part of the day and criteria for determining when these patterns branch. To construct the scenario trees in a scientific manner, we require statistical criteria to ensure the reliability of the procedure. After the structure of the scenario tree is laid out, the probabilities associated with each branch can be determined from the empirical marginal probabilities of each capacity profile pattern.

Developing “Optimal” Scenario-tree-based Strategies

Next, we must use the scenario tree to develop an “optimal” ground holding strategy. As already indicated, there is a well-developed literature on this. As the first step, we can apply pre-established optimization algorithms derived from that literature. We chose the Ball et al. (static) and Mukherjee-Hansen (dynamic) models for this purpose. Both of these models assume that the scenario-specific capacity profiles are deterministic. In a

setting with variation in the profiles for a given scenario we must decide which scenario-specific profiles should be input in the model. The obvious solution of using profiles derived from the scenario means may not be the best approach, as discussed later in this chapter.

Implementing Optimal Strategies

For static models such as that of Ball et al., implementation is straightforward because decisions are made just once, at the beginning of the planning period. The same cannot be said for dynamic models that use realized information on a given day to evolve the solution. In the idealized setup of these models, we always know which branch of the tree we are on, but in reality we don't. To implement a dynamic solution, we must take realized AAR observations—and/or possibly other information—for a period of time and decide which branch of the tree we are on. We term this process *branch matching*. If our branch matching is incorrect, we may trigger air traffic management actions that are highly suboptimal.

3.2 Developing Capacity Scenario Trees

The generation of a scenario tree is a major issue in application of scenario-based multistage stochastic programs. Most methods proposed in the literature were focused on supporting the application of stochastic programs in finance, as discussed in Chapter 2. As Kaut and Wallace (2003) pointed out, the goodness of a scenario-generation method is coupled with the decision model it serves. Though many methods have been proposed for scenario tree generations, they are designed for financial portfolio allocation problems. Features of that setting that do not exist in ours include pre-specified branching times and

a fixed number of branches. Hence, this section investigates a suitable method to determine scenario trees for the stochastic programs for the single airport ground holding problem. There are two building blocks for the construction of airport-specific scenario trees: first, identification of arrival capacity scenarios, and second, construction of a scenario tree based on these scenarios. The Subsections 3.2.1 and 3.2.2 present the methodologies for these two building blocks.

3.2.1 Identifying Capacity Scenarios

Previous studies on arrival capacity distributions developed models with one or few parameters specific to particular local characteristics (Inniss and Ball, 2004). For the scenario-based ground holding to be widely applicable, we aim to develop an arrival capacity profile pattern recognition mechanism that applies to various localities. Statistical clustering was identified as one such mechanism to classify arrival capacity data into patterns of arrival capacity profiles.

A daily capacity profile, which contains the AAR at n periods in a day, can be considered as a point in an n -dimensional space. The patterns of capacity profiles can then be recognized through clustering in R^n . Among the myriad methods of clustering, K -means clustering (MacQueen, 1967) was found suitable for this application. K -means clustering classifies data into K clusters. Although hierarchical clustering seems to be a tempting choice since a scenario tree has a hierarchical structure, it is not appropriate because it does not necessarily lead to trees that are hierarchical in chronological order, as we require here. The within-cluster means of AAR at these n time periods constitute the nominal arrival capacity scenario for a cluster. Subsequently, the probability associated with each of the scenarios can be estimated from the proportion of days

classified into each of the clusters. Another decision in performing clustering analysis is the choice of the measure of dissimilarity. We chose to use Euclidean distances to gauge the dissimilarity between the points. Hence, the cluster centers are based on least-squares estimation.

One important factor in the success of K -means clustering is the choice of K , the number of clusters. Milligan and Cooper (1985) compared thirty methods for estimating the number of population clusters and found the pseudo F statistic (Calinski and Harabasz, 1974), cubic clustering criterion (CCC) (Sarle, 1983) and pseudo t^2 statistic (Duda and Hart, 1973) performed best in their simulation studies. However, CCC is not valid for correlated variables and pseudo t^2 statistic can be applied only to hierarchical methods (SAS, 2004). Thus, the pseudo F statistic method seems best suited for our application. In this method, the pseudo F statistic is computed for clustering with each of the K values and, ideally, the local peak of the pseudo F statistic would indicate the most suitable K value. However, it is not always the case that a local peak exists—we observe monotonic increasing or decreasing pseudo F statistic over the domain of K values considered in our study. Moreover, we need to consider other objectives besides the pseudo F . On the one hand, more clusters is better, because this will enable traffic management solutions better tailored to the specifics of a scenario. But we also want to avoid clusters associated with a small numbers of days, because we require reliable probability estimates for each cluster. Toward this end, we may ignore clusters below a certain size. In doing this, however, we want to make sure that the clusters we retain account for the vast majority of days in our data set.

Our approach to choosing K takes all these objectives into account. As a preliminary, we define the size of a cluster as the number of days assigned to it. We introduce the following notation. Let

$N(k, m)$ be the number of clusters of size $\geq m$ when the total number of clusters is k . Here we set $m = 20$;

$D(k, m)$ be the total number of days included in clusters of size $\geq m$ when the total number of clusters is k ;

$F(k)$ be the pseudo F statistic when the total number of clusters is k ;

k_{\min}, k_{\max} be the minimum and maximum values of the range of k values to be considered. Here we set $k_{\min} = 3$ and $k_{\max} = 16$;

$N^*(m)$ be the maximum of $N(k, m)$ over the range $k_{\min} \leq k \leq k_{\max}$;

$\Omega(k_{\min}, k_{\max}, m)$ be the set of k values in the range $k_{\min} \leq k \leq k_{\max}$ such that

$$N(k, m) = N^*(m).$$

Our rule for choosing K depends on whether $F(k)$ has an interior maximum (as opposed to a boundary maximum) over the set of values $\Omega(k_{\min}, k_{\max}, m)$. If so, the K is the solution to:

$$\max_{k \in \Omega(k_{\min}, k_{\max}, m)} F(k)$$

Otherwise, we choose K as the solution to:

$$\max_{k \in \Omega(k_{\min}, k_{\max}, m)} D(k, m)$$

Our first aim in this approach is to create as many clusters satisfying the minimum size limit m as possible. We choose among the K values that satisfy the first criterion based on the pseudo F statistic if it gives a clear indication. Otherwise we focus on maximizing the number of days covered by clusters of size greater than m . In either case, to expedite the process, we choose K from a pre-determined range of values that is very likely to contain the “best” one.

3.2.2 Constructing Scenario Trees

The key of formulating the SAGHP with a multistage recourse model lies in the use of a scenario tree. With a scenario tree, as the actual capacities are revealed over time, the ground-holding decisions can be made over multiple periods. To construct a scenario tree based on the capacity scenarios identified, an algorithm was developed to fuse similar scenarios into common trunks and determine when the trunks branch out into subsets of scenarios, and ultimately into individual ones, as follows:

Step 1: Compute the 99% confidence bands for each of the scenarios assuming t -distribution. Let $x_i(t)$ denote the within-scenario mean and $[x_i^l(t), x_i^u(t)]$ the 99% confidence interval of AAR at period t for scenario i . Following this notation, the 99% confidence band for scenario i from time $t = 1$ to T can be characterized by the pair of sequences $\{x_i^l(t) | t = 1, 2, \dots, T\}$ and $\{x_i^u(t) | t = 1, 2, \dots, T\}$.

Step 2: Initially assume all the scenarios are fused together in one common scenario tree trunk. For each pair of scenarios—say, scenario A and B—if Scenario A’s mean is not within scenario B’s 99% confidence band or scenario B’s mean is not within scenario A’s

99% confidence band for four consecutive periods (i.e., 1 hour), the branching is specified to occur on the first of the four periods. Formally, this is stated as follows.

For scenario i and j at time period t , define

$$I_{ij}(t) = \begin{cases} 1 & \text{if } x_i(t) \in [x_j^l(t), x_j^u(t)] \\ 0 & \text{otherwise} \end{cases} \quad \forall i, \forall j, \forall t.$$

The branching time τ_{ij} for each pair of scenarios i and j is determined as follows.

- i. Initialize with $\tau := 1$.
- ii. If $\sum_{t=\tau}^{\tau+3} I_{ij}(t) \cdot \sum_{t=\tau}^{\tau+3} I_{ji}(t) = 0$, stop and we obtain $\tau_{ij} = \tau_{ji} = \tau$; otherwise, go to iii.
- iii. If $\tau = T - 3$, stop and it is found that scenario i and j never branches away; otherwise, go to iv.
- iv. $\tau := \tau + 1$. Go to ii.

Step 3: From Step 2, scenarios obtained may be *isolated* (i.e., they branch away from every other scenario right from the beginning), *affiliated* (i.e., they evolve from common roots), or both. More formally, scenario i is an isolated scenario if $\tau_{ij} = 1$, for all $j \neq i$. Let g denote the set of affiliated scenarios in one such group. Then g is defined such that $\tau_{ij} \neq 1$ for all $i \in g$, $j \in g$ and $i \neq j$. For any affiliated scenario i in g , the time it branches away from the others in the group is determined by

$$\tau_i(g) = \min_{j \in g \setminus \{i\}} \tau_{ij}.$$

Step 4: Construct the scenario tree by using the branching time information obtained from the previous steps. Each of the isolated scenarios found in Step 2 is deemed a group by itself, and let G denote the set of all the groups found. The construction of the scenario tree can be outlined algorithmically as follows:

for each $g \in G$

if $|g| = 1$, the only scenario $i \in g$ is an isolated scenario ;

else, for each scenario $i \in g$, the branching time of scenario i from g is $\tau_i(g)$.

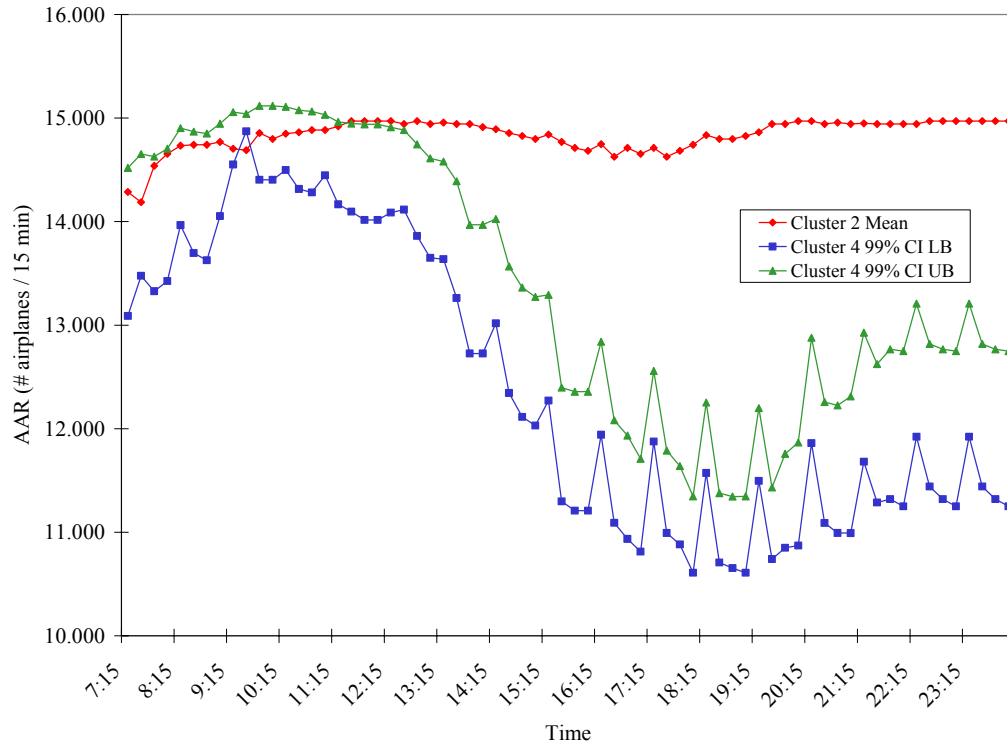


Figure 3-1 Mean of one scenario and 99% confidence band of another scenario (CI = confidence interval, LB = lower bound, UB = upper bound).

The operation in Step 2 is performed for all scenario pairs. For example, with six scenarios, branching times will be computed for 15 pairs of scenarios. This operation is designed so that occasional stray data points do not affect the structure of the scenario tree. One such example is shown in Figure 3-1. The idea behind the procedure in Step 3 is illustrated by the following example. Suppose it is found that a group g of affiliated

scenarios is formed by Scenario 1 (S_1), Scenario 4 (S_4), and Scenario 6 (S_6). S_1 and S_4 branch apart at $\tau = 10$, S_6 and S_4 branch away at $\tau = 15$, and S_1 and S_6 branch away at $\tau = 8$. It is clear that S_4 and S_6 stay together till $\tau = 15$, but the time at which S_1 branches away from the subgroup of S_4 and S_6 needs to be determined because it could be at either $\tau = 8$ or $\tau = 10$. In our algorithm, a pair of subgroups of branches branch away at the earliest suggested branching time ($\tau = 8$ in this case) since this is the earliest time when some members in these subgroups diverge.

3.2.3 Empirical Capacity Scenarios and Scenario Trees

The arrival capacity scenarios at major US airports were identified using the method described above based on FAA ASPM daily airport data from January 1, 2003 to December 31, 2003. Here we present scenarios identified at Dallas-Fort Worth (DFW), Miami (MIA), Boston Logan (BOS), Atlanta (ATL), and San Francisco (SFO) (Figure 3-2, Figure 3-3, Figure 3-4, Figure 3-5, and Figure 3-6). Note that not all clusters are recognized as scenarios. In the case of DFW, for instance, we obtain $K=9$ from the procedure described above, but of the nine clusters only five are of size 20 or more, and thus recognized as scenarios. Together, the discarded clusters account for only 6.6% of the days analyzed. Further, the clustering result statistics indicate that the clusters identified have small root-mean-square standard deviation (less than 2) and big radius (around 20 to 30), and hence, good intra-cluster grouping and inter-cluster separation.

For BOS, MIA, DFW, and SFO, we observe scenarios that are similar at the beginning of the day and later branch out to distinct patterns. In such cases, it is appropriate to construct scenario trees, and models based on such trees may be applicable. Only in the case of ATL are all of the scenarios isolated from the beginning of the day.

Many of the scenarios feature a pattern of oscillation with a periodicity of 1 hour. This is actually an artifact of the reporting system, which converts hourly rates to quarter-hour ones but also requires integer values.

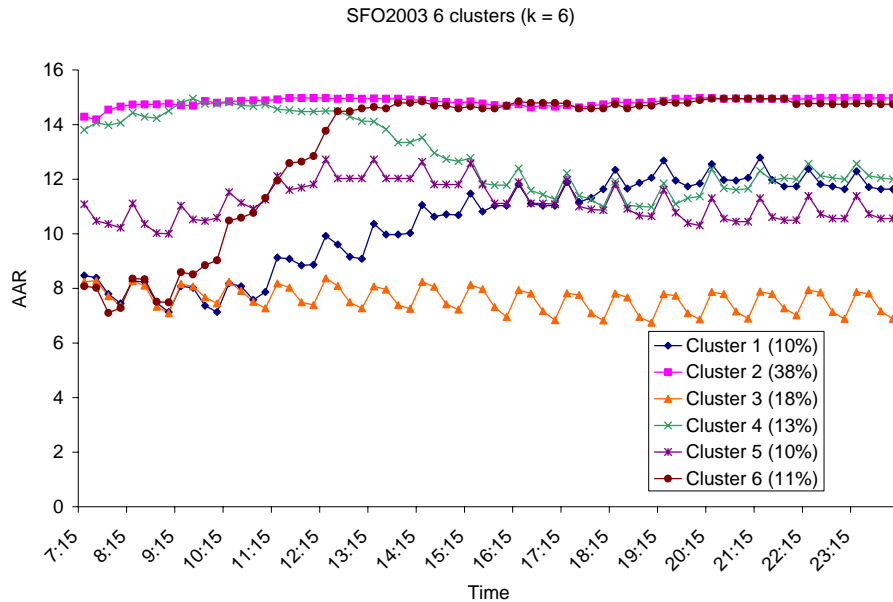


Figure 3-2 Scenarios at SFO

In general, the scenarios reflect weather patterns in the regions where the airports are located. For example, the phenomenon of marine stratus (fog) burn-off at SFO is clearly illustrated in the scenarios. The morning fog at SFO usually clears up around the middle of the day. Cluster 1 and 6 in Figure 6 shows arrival capacity patterns that could be associated with the fog burn-off phenomenon with two burn-off times. Cluster 3 in the same figure has capacity profile similar to those of Cluster 1 and 6 early in the day, but stays with low capacity throughout the day. For SFO, we also verified that the scenarios are closely related to the seasons. The low capacity Cluster 3 is associated with the winter rainy season at San Francisco. The high capacity of Cluster 2, on the other hand, can be

associated with the clear sky in autumn. Similarly, some of the scenarios at BOS and MIA could possibly be attributed to the thunderstorms in those regions, but this is yet to be verified.

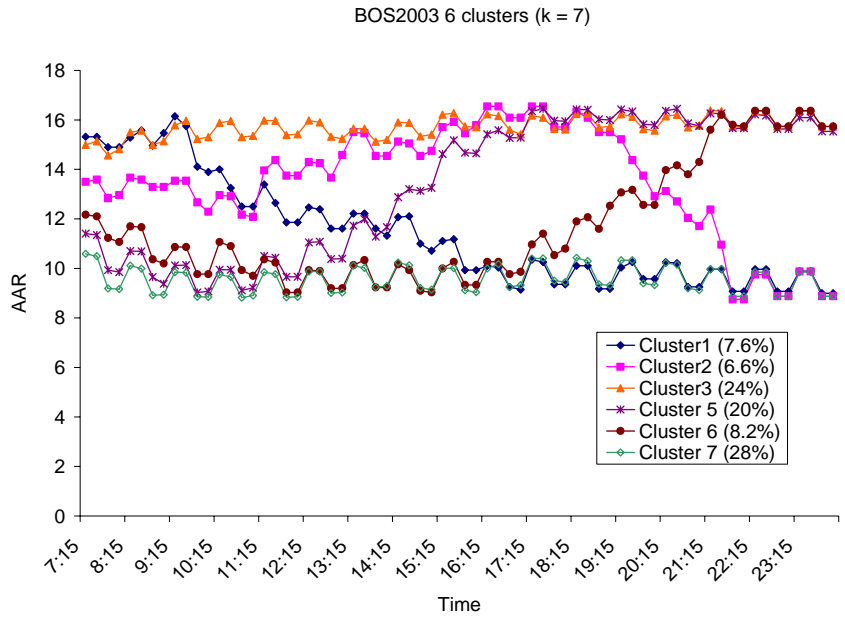


Figure 3-3 Scenarios at BOS

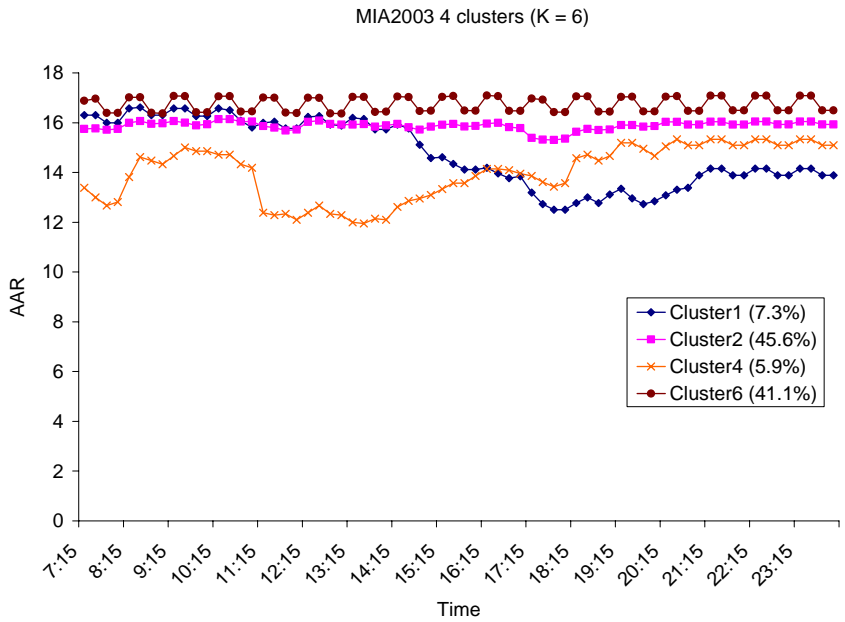


Figure 3-4 Scenarios at MIA

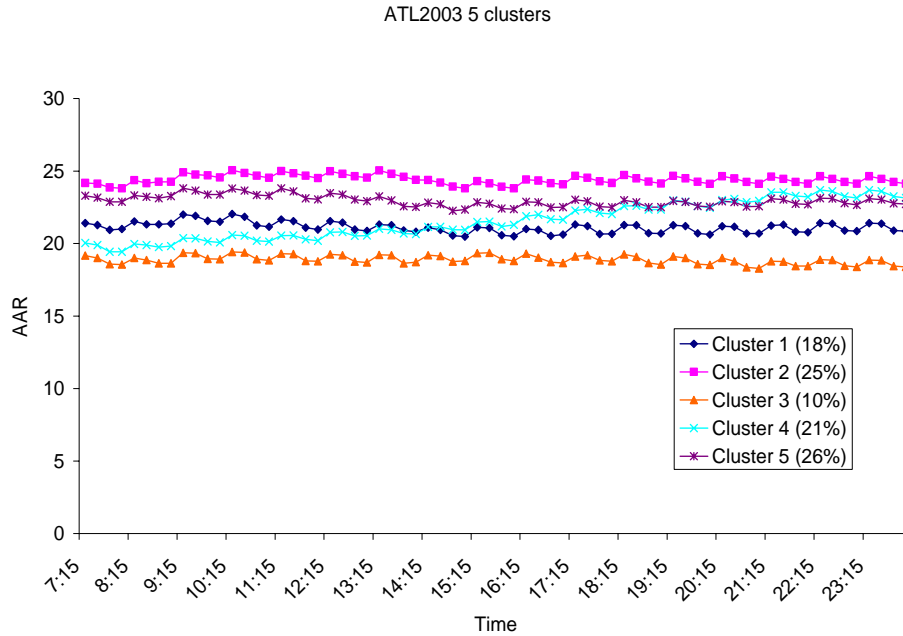


Figure 3-5 Scenarios at ATL

Most of the interesting scenario patterns are from coastal airports—thanks to the variability of oceanic weather. In fact, most of the inland airports in the United States have monotonic scenarios like those at ATL, where the clusters suggest that capacity is generally consistent throughout a given day. For Cluster 2, the capacity is high, as result of clear skies and high visibility. Cluster 3, in contrast, corresponds to days with consistently low cloud ceiling and reduced visibility. The only dynamic profile is Cluster 4, which features improving conditions throughout the day. In some cases, improvement derives from improving visibility as morning fog and haze dissipate, whereas in others it results from the breakup of a cloud ceiling.

Among the interior airports, DFW featured the most interesting scenarios. It features two increasing-capacity scenarios similar to that for ATL, as well as a decreasing capacity scenario (Cluster 9) that arises as a result of afternoon thunderstorms. Of the

increasing capacity scenarios, one (Cluster 2) corresponds to a transition from marginal visual flight rules (VFR) to full VFR, and the other (Cluster 1) to a transition from instrument flight rules (IFR) to marginal VFR.

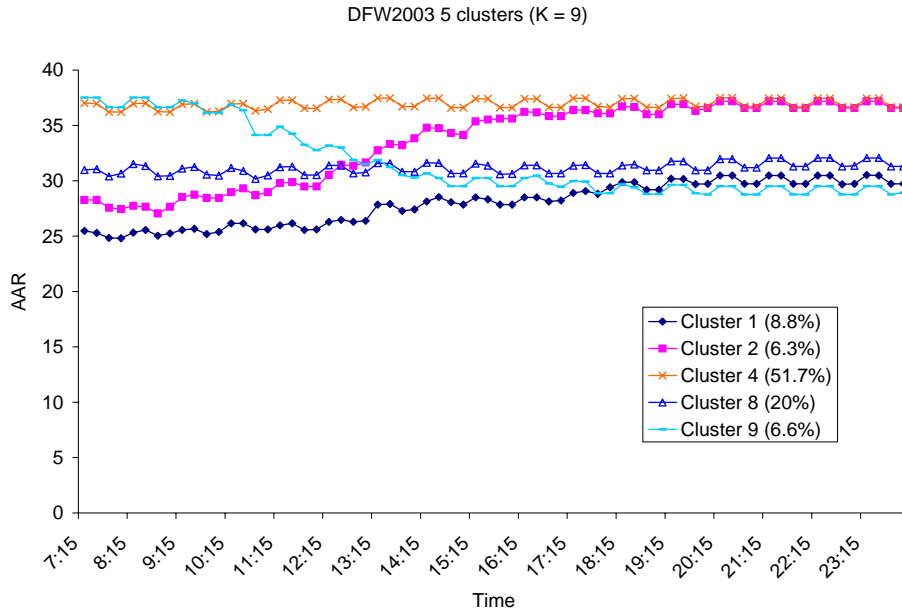


Figure 3-6 Scenarios at DFW

With the capacity scenarios identified, the scenario trees were constructed on the basis of the proposed algorithm proposed. For example, the scenario tree at SFO is constructed as illustrated in Figure 3-7. The conditional probabilities associated with each of the branches were worked out backwards from the empirical marginal probabilities of each of the scenarios. For example, if the early morning AAR is 8, then the system is on the branch containing Scenarios 1, 3, and 6. In that event, the probability that the realized scenario is 1 is calculated based on the frequency of Scenario 1 relative to the other two scenarios associated with that branch. Thus we obtain $P(1) = .10 / (.10 + .18 + .11) = .26$.

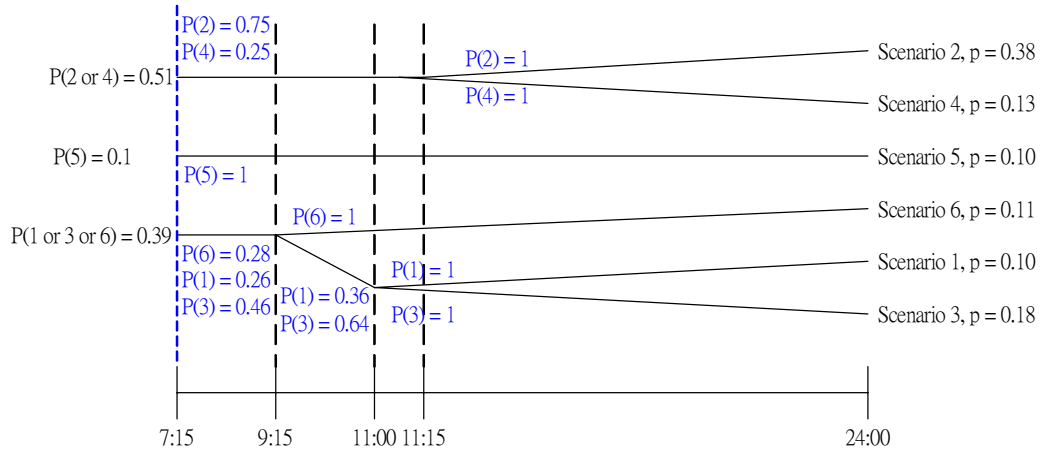


Figure 3-7 Scenario tree at SFO

3.3 Performance of Optimization Models

Theoretically, the dynamic model (Mukherjee-Hansen) clearly would yield a smaller total delay cost than the static model (Ball et al.) because it is essentially a relaxation of the static model for the same minimization problem. However, it would be interesting to learn how big a gap in optimal delay cost this relaxation would create when both models are parameterized with real airport and flight data. Two sets of tests were performed to determine the difference between outcomes from the two models.

First, the optimal delay costs computed from the two models on the same day (January 8, 2004) at four U.S. airports—BOS, MIA, SFO and Seattle (SEA)—were compared. The scenario trees at these airports were constructed based on the AAR data from these airports for the entire year of 2003. The arrival demand information was derived from the flight schedule on January 8, 2004, at these four airports. With the air-to-ground delay cost ratio of 3, the optimal total delay costs are listed in Table 1. Across

these four airports, the relaxation of the static model consistently reduces optimal total delay cost 50% to 70%, which is substantial.

Table 3-1 Optimal Total Delay Costs at Four Airports, Reported in Ground Delay Time Period Equivalents

| Airport | Static Model | Dynamic Model | % Reduction |
|----------------|---------------------|----------------------|--------------------|
| BOS | 1 | 0.3 | 70% |
| MIA | 0.54 | 0.18 | 67% |
| SEA | 8.6 | 4 | 53% |
| SFO | 88.22 | 35.79 | 59% |

1 unit value = 15 min of ground delay or 5 min of airborne delay

The above results could be different if a different air-to-ground delay cost ratio were applied in the optimization. For example, the optimal total delay costs obtained using different cost ratios for January 8, 2004, at SFO are shown in Figure 3-8. The optimization tends to assign more ground delays to avoid costly airborne delays when the unit cost for airborne delay is higher, to avoid costly airborne delays. In the face of high cost for airborne delay, the “wait-and-see” strategy in the dynamic model effectively managed to keep the cost down whereas the static model suffered a high penalty.

The delay cost composition from both the dynamic and static optimization models was examined for the same date at SFO (Figure 3-9). In general, the dynamic model yields solutions with lower ratios of airborne to total delay cost except for c_a/c_g values between 6 and 8. When $c_a/c_g < 6$, the dynamic model allows for some airborne delay, but can keep its cost contribution to a minimum by dynamically revising release times. When $c_a/c_g > 8$, the dynamic model eliminates all airborne delay by releasing flights only when sufficient capacity to land them has been assured.

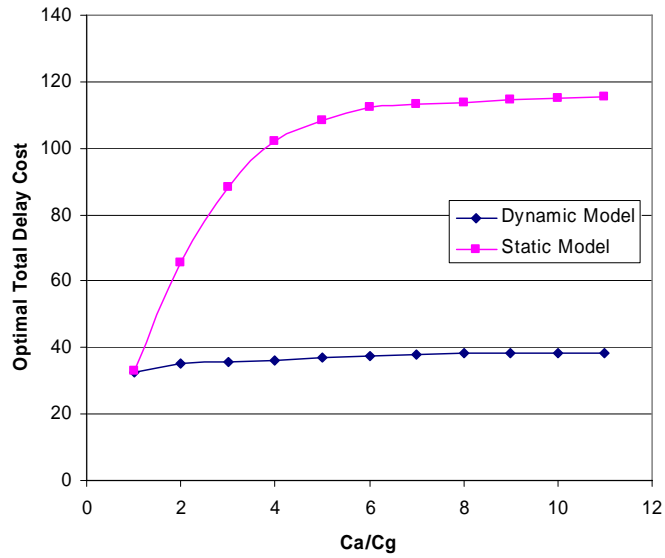


Figure 3-8 Optimal delay cost obtained with different cost ratios, January 8, 2004, SFO.

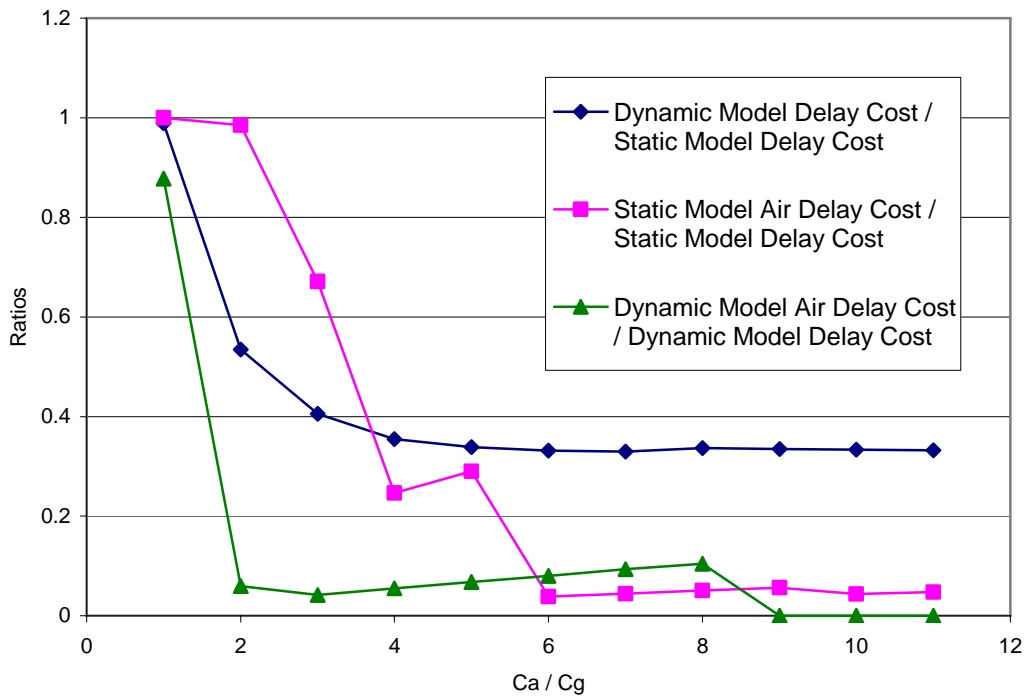


Figure 3-9 Ratios of optimal delay costs with various air-to-ground unit cost ratios.

Second, 15 weekdays in 2004 were randomly selected and the optimal expected total delay costs at SFO were computed using the two models. The only source of variability in these results is the flight schedule. With the air-to-ground delay cost ratio of 3, the optimal total delay cost from the static model is consistently more than twice of that from the dynamic model, as shown in Figure 3-10, where the delay cost is expressed in units of cost per 15 minutes of ground delay. On average, use of the dynamic model yields a cost reduction averaging 63% over these 15 days. Taken together, these empirical results support a “rule of 2/3” for the fractional cost reduction that results from the use of the dynamic model instead of a completely static one. This finding must, however, be tempered by the fact that, in practice, static models would be rerun as new information becomes available.

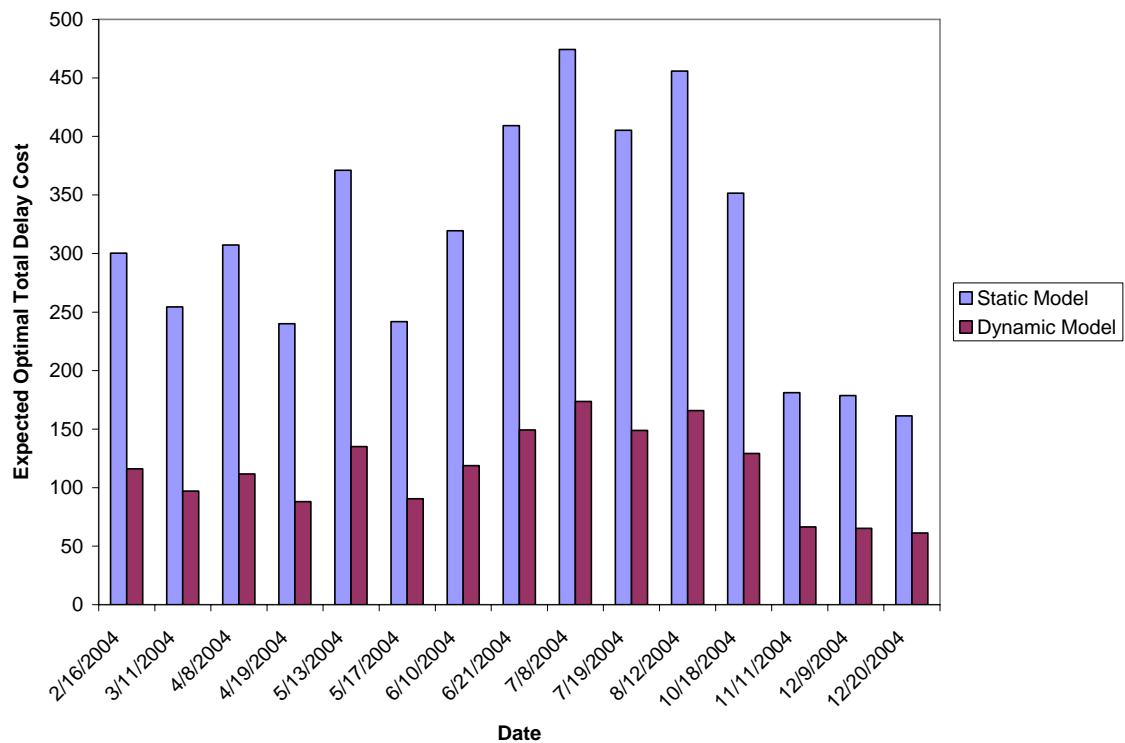


Figure 3-10 Optimal total delay costs from dynamic and static models, 15 days, SFO.

3.4 Implementing Optimal Strategies

While the previous section considered the application of scenario-based air traffic management in the context of real-world airports, it did so in an idealized fashion. We synthesized scenario trees from real-world data, but then idealized the problem by assuming that the capacity profiles follow scenario trees exactly. The deviations from these profiles seen in the real world affect scenario-based air traffic management in two important ways. First, in the case of the dynamic model, the deviations limit the ability of managers to discern which branch of the tree the system is following. In the idealized case branching times are deterministic, and the branch being followed can be known with certainty immediately after a branching point. This knowledge becomes imperfect in the presence of deviations. Implementation thus requires a method for choosing which branch is most closely followed based on the past capacity evolution. Moreover, the branch identified through this method may turn out to be the wrong one in retrospect, leading to air traffic management decisions that may be far from optimal.

The second consequence of differences between actual and idealized scenario capacities affects the performance of both static and dynamic models. In both cases, the differences increase, on average, the incurred cost relative to the expected cost based on the nominal scenario. This is because negative capacity deviations affect operational performance more than positive capacity deviations. The models attach high penalties to airborne delay, resulting in relatively little airborne holding, and consequently few flights that can take advantage of unanticipated capacity surpluses. On the other hand, it is very likely that a capacity deficit will result in additional airborne delay, particularly in the

case of a low capacity scenario. Thus, negative capacity deviations exact a large cost penalty.

It may be possible to “tune” the scenario-based models to improve their performance in a real-world setting. The general approach is to transform the scenario tree developed from real-world data to one that yields better results in the face of real-world deviations. To improve branch identification, the branching times may be delayed so that there is more information to go on when the identification is made. To mitigate the asymmetry of the capacity deviation costs, the nominal capacities assumed in the optimizations may be reduced. The choice of parameters for this tuning, for example, how much branching time delay to introduce, is another implementation issue.

The first step in addressing these implementation issues is to devise an algorithm for branch identification that takes deviations between realized and idealized scenario capacities into account. We then proceed to compare the performance of the static and dynamic models when such deviations are present. Finally, we investigate the effects of scenario tree tuning on the performance of the models.

3.4.1 Branch Identification Methodology

The first issue in branch identification is when to do it. Suppose there is a branching point at time period t . In the idealized problem, we know which branch is being followed as of $t+1$. In practice, we are unlikely to have a definitive indication of this. If we defer the identification time, we are more likely to make a correct identification. But there is also a penalty for deferring: the later we obtain the branch identification, the less valuable

that information becomes. To investigate how different lags in the branch identification time affect operational results, we performed branch identification at $t+1$, $t+3$, and $t+5$.

Given a time period τ in which we want to identify the branch of a scenario tree to which an actual capacity profile belongs, the identification procedure is as follows. Suppose at this time there are $m(\tau)$ candidate scenarios. (As discussed below, a candidate scenario is one that has not been eliminated as the result of a previous application of this procedure.) Scenario i ($i = 1, \dots, m(\tau)$) is defined as a sequence of arrival capacities, $c_{i,t}$, from $t = 1$ to τ : $S_i = (c_{i,1}, c_{i,2}, \dots, c_{i,\tau})$. The actual arrival capacity profile P is defined as $P = (c_{p,1}, c_{p,2}, \dots, c_{p,\tau})$. Define $d_\tau(P, S_i)$ as the distance between P and S_i through time τ . The scenario with which the actual sample path is identified at time τ is:

$$b(\tau) = \arg \min_{i=1, \dots, m(\tau)} d_\tau(P, S_i),$$

$$\text{where } d_\tau(P, S_i) = \sqrt{\sum_{t=1}^{\tau} (c_{i,t} - c_{p,t})^2}.$$

The above distance measure is the Euclidean norm up to time τ . Other distance measures such as L-1 norm and cubic distance were also tested, but the Euclidean distance measure performed the best during the experiments.

The above procedure may result in a tie. The first tie breaker we use is simply the distance between P and the S_i 's at time τ , $\sqrt{(c_{i,\tau} - c_{p,\tau})^2} = |c_{i,\tau} - c_{p,\tau}|$. When there is still a tie when the first tie breaker is applied, we backtrack in time, period by period, and again use $d_{\tau-1}(P, S_i)$ and $\sqrt{(c_{i,\tau-1} - c_{p,\tau-1})^2}$, $d_{\tau-2}(P, S_i)$ and $\sqrt{(c_{i,\tau-2} - c_{p,\tau-2})^2}$, and so on to break

the tie. In all the cases we investigated, we could break the tie by going back no more than three periods. In the unlikely event that even this does not work, we could pick the most likely scenario based on their marginal probabilities.

This algorithm can lead to misidentification, in the sense that the best-match scenario based on the entire capacity profile (which we will term the best retrospective match) may not be the same as that obtained by applying the algorithm at every branching point, as the dynamic model requires. To illustrate, consider Figure 2-1. If the algorithm is applied at $\tau = 4$, it may show that the system is following the lower branch which will eventually devolve into scenarios 2 and 3. Ultimately, however, we may find that the capacity profile most closely matches scenario 1. The dynamic model does not permit such revisionism. If the system is determined to be following the lower branch at $\tau = 4$, then scenario 1 is not a candidate at $\tau = 7$. The optimal dynamic ground holding strategy does not allow a contingency for “branch hopping.”

We tested our branch identification algorithm on the same 15 SFO days considered in the previous section. For each day in the sample, we applied the identification algorithm one, three, and five periods after every branch point. We also applied the algorithm toward the end of the day to determine the scenario that has the best retrospective match. We compared the results of the branch point matching with retrospective matching, obtaining the results summarized in Figure 3-11, which apportions the 15 days according to the quality of the branch point matching compared to the retrospective matching. If the two procedures identify the same scenario, the match is classified as Exact. If the branch point matching procedure yields a different scenario than the retrospective matching, but the difference in the distance measure is small (less

than 20%), the match is classified as Good. For the remaining cases, the match is termed Poor. Figure 3-11 shows that if we applied the identification procedure just after the branch point, the result is Poor in $\frac{2}{3}$ of the cases, but if we wait until 5 periods after the branch point, the match is Exact or Good in $\frac{3}{4}$ of the cases.

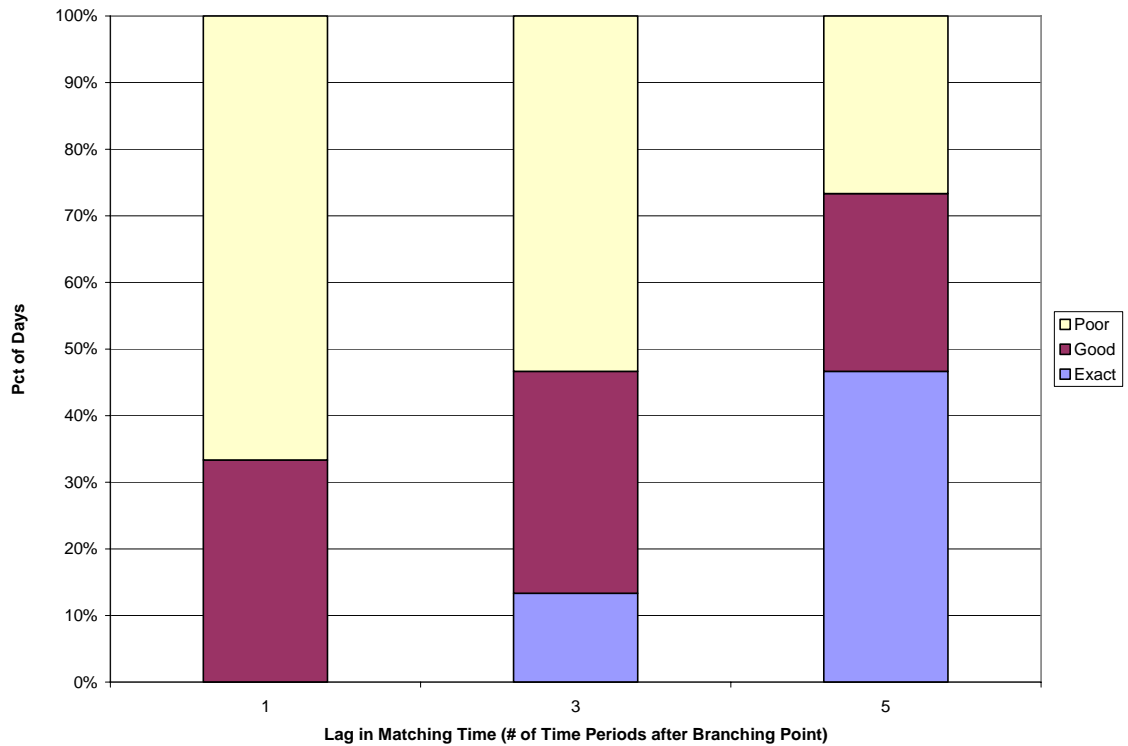


Figure 3-11 Branch Identification Results with Different Branching Lags

3.4.2 Implementation Performance

To address the issue of branch-hopping, we investigated imposing a lag in the information tree relative to the physical tree. We observed that, oftentimes, if we were allowed to wait a few more periods before we make a decision on which branch the

capacity profile is following, the chances of picking the correct branch increases substantially. Thus, we conjectured that imposing a lag in the information tree, which is in effect the set of coupling constraints in the dynamic model, relative to the physical tree, might improve our chances of applying the “optimal” strategy. There are other ways to parameterize the information tree such that the performance of the solution strategy can be improved. From queueing theory, we know that the performance degrades substantially as the utilization is set close to the limit. Hence, applying a scale factor to scale down the capacities is a plausible way to improve the performance of the solution strategy.

We are now able to simulate the performance of both the static and the dynamic models in a real-world setting, and compare this with the performance under the idealized conditions. For each of the 15 SFO days, we simulated the delay cost that would be incurred if we implemented the ground holding decisions found to be optimal under each model. In the case of the dynamic model, implementing the solution required the use of the matching algorithm at each branching point, while in the static model case we merely had to simulate the plan developed at the beginning of the day. The dynamic model optimization was performed for lags in matching time of 1, 3, and 5 time periods since the lag affects the coupling constraints (see Equation 2-6). For comparison, we also report idealized scenario-specific costs based on the idealized case in which capacity profiles follow the nominal scenarios perfectly. (In the idealized case, branches can be identified with perfect accuracy just one time period after the branching point.)

The results appear in Table 3-2. For both models, incurred costs are substantially greater than idealized costs. The ratio (based on the averages over the 15 days) is between

2.5 and 3 for the dynamic model, and 1.5 for the static model. For the latter, incurred costs are higher because of the asymmetric effect of capacity deviations around the nominal scenario. In the dynamic model, errors in branch matching result in additional cost penalties. Nonetheless, the dynamic strategy results in an incurred cost 20-30% less than that obtained from the static model. Of the dynamic models, the one that defers branch identification until 5 periods after the branch point yields the best results. The gain from improved accuracy exceeds the loss from the less timely information over this lag range. Across the days, the variability of the total delay costs is sizable for both models under both idealized and simulated real conditions. The magnitude of dispersion is similar across the models and conditions in terms of the coefficient of variation (c.o.v.). In sum, the results show that the gain from using the dynamic model under real-world conditions is less than that under idealized conditions, but nonetheless sizable, and that dynamic model performance is improved by deferring branch identification until well past the nominal branch points.

Lagging the branch identification time helps reduce the cost of incorrect branch identification. Are there any strategies that can reduce the cost of capacity dispersion? As we have seen, this factor alone increased the incurred cost from implementing the static model by 50 percent. It probably has a similar impact on dynamic model outcomes, although in this case the effect is combined with that of branch misidentification. The potential value of strategies for reducing the capacity dispersion cost is therefore considerable.

Table 3-2. Total Delay Cost under Dynamic and Static Models

| Date | Dynamic | | | | Static | |
|-----------|---------------|-------|-------|----------------|---------------|----------------|
| | Incurred Cost | | | Idealized Cost | Incurred Cost | Idealized Cost |
| | Lag 1 | Lag 3 | Lag 5 | Lag 1 | | |
| 02/16 | 2220 | 1854 | 1853 | 419 | 2564 | 1127 |
| 03/11 | 70 | 70 | 70 | 10 | 71 | 44 |
| 04/08 | 203 | 240 | 203 | 57 | 107 | 38 |
| 04/19 | 91 | 91 | 91 | 1 | 102 | 30 |
| 05/13 | 985 | 935 | 705 | 570 | 1283 | 1616 |
| 05/17 | 729 | 689 | 484 | 349 | 965 | 959 |
| 06/10 | 717 | 659 | 568 | 448 | 1077 | 1230 |
| 06/21 | 336 | 616 | 342 | 73 | 517 | 58 |
| 07/08 | 330 | 366 | 397 | 47 | 433 | 70 |
| 07/19 | 296 | 328 | 296 | 68 | 328 | 58 |
| 08/12 | 307 | 334 | 304 | 63 | 431 | 62 |
| 10/18 | 74 | 74 | 74 | 3 | 93 | 63 |
| 11/11 | 1410 | 1196 | 1199 | 257 | 1650 | 711 |
| 12/09 | 372 | 318 | 327 | 255 | 569 | 707 |
| 12/20 | 511 | 467 | 332 | 233 | 695 | 641 |
| Average | 577 | 549 | 483 | 190 | 726 | 494 |
| Std. Dev. | 586 | 482 | 475 | 186 | 693 | 540 |
| c.o.v. | 1.02 | 0.88 | 0.98 | 0.98 | 0.96 | 1.09 |

Perhaps the simplest such strategy is to scale the scenario capacities. Instead of using the scenario capacity profiles obtained directly from the cluster analysis, we multiply the capacities by some factor less than 1. In effect, we make ground holding decisions more conservative by basing them on lower assumed capacity values. This reduces the potential cost from having realized capacities less than their nominal values, albeit at the cost of underutilizing available capacity at other times.

We evaluated the scaling strategy by simulating its results for the same 15 SFO days used in the previous analyses. We used scale factors ranging from 0.75 to 1.0 in the static model, and the dynamic models with branch identification 1 and 5 periods after the branch point. The results appear in Figure 3-11, which plots the incurred cost averaged over the 15 simulation days against the scaling factor. Scaling yields improvement for

both the static model and dynamic model with a lag of 1. The optimal scale parameter for the static model is about 0.8, which results in an incurred cost 13% less than the unscaled static model. For the lag 1 dynamic model, a scale parameter of 0.85 yields an incurred cost that is 38% less than the unscaled version. Scaling does not improve the results for the lag 5 dynamic model, suggesting that scaling and lagging are more effective when used individually than in combination. Of all the combinations of lagging and scaling tested for the dynamic model, a 1-period lag and 0.85 scaling yielded the best results, with incurred costs 16% less than the runner-up 5-period lag and no scaling, and 35% less than the best static model.

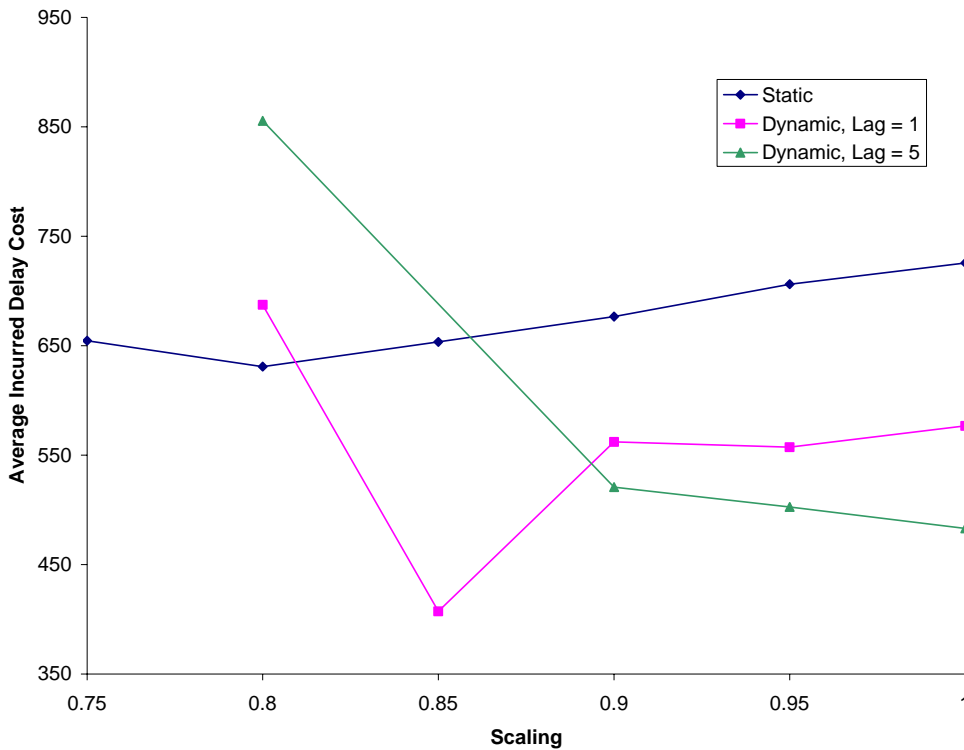


Figure 3-12 Effects of scaling nominal scenario capacities

Finally, we show a comparison between the expected costs computed by the optimization models and the incurred costs from applying the optimal solutions under real conditions. The average total incurred delay cost is much higher than the expected delay cost for both models (Figure 3-13), while the dynamic model has lower cost in either case. The major reason that the models do not perform as expected under real conditions is the discrepancy between the actual capacity profile and the nominal scenarios used in the models. For the dynamic model, another reason is that the scenario tree branch followed early in the day does not always turn out to be the best match retrospectively. Figure 3-13 also shows that both lagging and scaling have positive effect on the performance of the dynamic model, whilst the scale factor is more effective between the two. Although the expected cost is higher with scaled down capacities in the scenarios, it also leads to the desirable property of a smaller gap between expected cost and incurred cost.

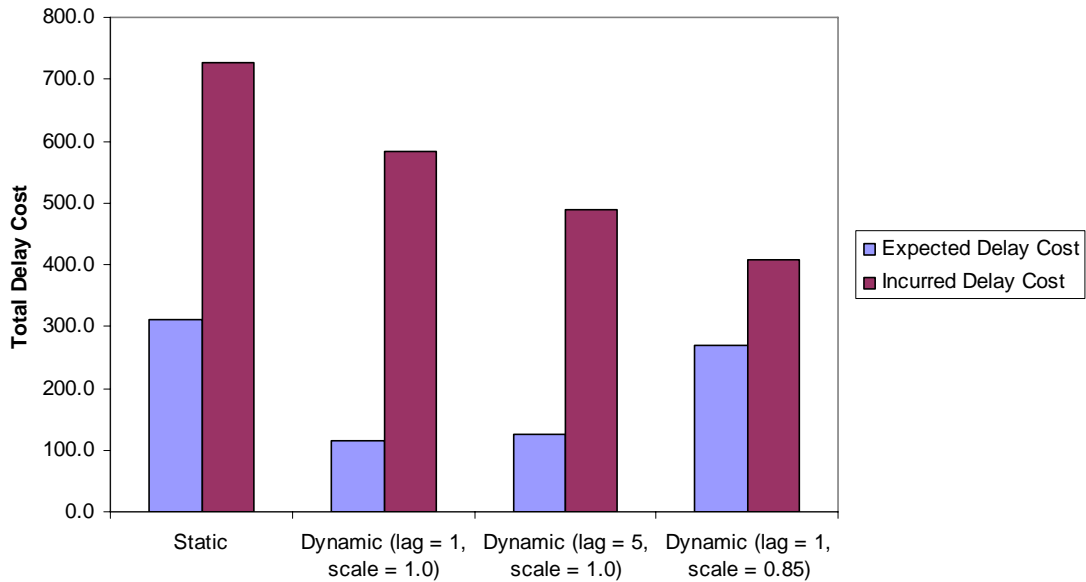


Figure 3-13 Average total delay cost--expected versus incurred cost

3.5 Summary

In this chapter, we have investigated the real-world applicability of scenario-based approaches to the single airport stochastic ground holding problem, including the static model of Ball et al. and the dynamic Mukherjee-Hansen model. Our results demonstrate the feasibility of applying these models to real-world airports. Firstly, we have found that capacity scenarios, which previous studies have assumed but not looked for, exist and can be inferred from historical data. Second, we have found that, for certain airports, the scenarios follow a tree structure in which certain scenarios have similar profiles during the early parts of the day and then branch out later on. We proposed and demonstrated a heuristic for identifying the branching points. Next, we showed that the dynamic model, by anticipating and then using the new information that becomes available after a branching point, can reduce delay costs over 60 percent when compared to the static model.

We then considered the challenges and additional costs of employing these models in a real world in which actual capacity profiles do not follow scenario profiles perfectly, a phenomenon we termed capacity dispersion. First, we proposed a method for matching a realized profile with a nominal scenario profile when dispersion is present. We then compared the incurred costs resulting from applying the dynamic model and static model when dispersion is present. Dispersion results in higher incurred cost for both models, but particularly for the dynamic ones, where it leads to mistakes in identifying the scenario being followed by a capacity profile. Simple methods for reducing the costs of dispersion, including scaling the scenario tree and deferring the branch identification times, were found to be somewhat effective in mitigating the costs of dispersion. Using these, the

dynamic model reduces incurred costs by about one-third compared to the static model, a smaller improvement than in the case without dispersion, but a significant one nonetheless.

Clearly, capacity dispersion exists and one tempting way to handle it is to include more capacity scenarios in these scenario-based optimization models. However, the number of scenarios is limited for the integer programming problem to be solvable using the latest solvers. The primary shortcoming of the scenario-based models is that they assume limited number of capacity scenarios for a reality where there is a much larger set of possibilities for capacity evolution. In addition, scenario-tree-based models impose a scenario tree structure on a reality where the improved information about future capacity is obtained continually rather than at a few discrete branching points. These limitations prompt us to consider a “scenario-free” approach for the SAGHP, and that will be discussed in detail in the next chapter.

CHAPTER 4 SCENARIO-FREE MODELS FOR SINGLE AIRPORT GROUND HOLDING POLICY PROBLEM

The uncertainty in the air traffic system has been included in the stochastic programming models for SAGHP using scenarios or scenario trees. Due to the limited number of scenarios such models can accommodate while staying solvable for the latest solvers, the performance of these models is compromised in a real-world setting where most of the capacity profile realizations are unplanned for. Ideally, problems of planning under uncertainty can be solved optimally using dynamic programming. In the pre-CDM centralized-control environment where the GHP solves for ground holding assignment for each flight, it is likely that a dynamic program would suffer from the *curse of dimensionality* and become computationally intractable. In the current air traffic management environment where a CDM-compliant solution, which need not be flight-specific, is needed, a dynamic programming-based approach has potential to solve the problem optimally since the size of problem is much smaller. Here we propose a sequential decision making model for the SAGHP whose solution is an optimal policy for dynamic control.

The use of dynamic programs for the GHP was explored by Andreatta and Romanin-Jacur (1987) for a simplified single-time period SAGHP and Terrab and Odoni (1993) for SAGHP with multiple time periods. Terrab and Odoni formulated a dynamic program based on the assumption that a fixed landing priority rule has been specified for the flights, and solved for ground-holds for each individual flight. The stochasticity in the problem was accounted for using a set of capacity scenarios. Using the terminology

employed in Chapter 3, it is, however, a *static* model, as it does not allow for recourse actions. Besides, based on their complexity analysis, they had concerns over the practicality in solving problem instances of realistic sizes and devised several heuristics. Since then, dynamic programming has been thought as doomed by the curse of dimensionality and overlooked as a nonviable solution method. A decade after the initial attempt of using dynamic programs to solve SAGHP, we investigate the applicability of a true “dynamic” program for SAGHP in light of the limitations of the scenario-based stochastic programs, the changes in the ATFM environment, and the ever-increasing computing power.

This chapter is organized as follows. Section 4.1 describes the formulation of the SAGHP using a sequential decision model framework and discusses the computational complexity of the dynamic programming algorithm. In Section 4.2, we propose computational solution strategies to manage the curse of dimensionality of dynamic programming. In Section 4.3, we present computational results and discuss the properties of the model. Finally, Section 4.4 gives concluding remarks for this chapter.

4.1 Sequential Decision Model for the SAGHP

The process of making ground-holding decisions in a SAGHP can be naturally described as sequential decision making. In a sequential decision making model, as illustrated in Figure 4-1, the decision maker, or agent, chooses an action at each decision epoch based on the state of the system. The action incurs an immediate cost and affects state and the future evolution of the system. At a subsequent point in time, the system may be in a different state and there may be a different set of actions for the agent to

choose from. In the SAGHP, the goal is to react optimally to the capacity changes at the destination airport such that the expected system-wide cost is minimized. The system includes the destination airport of concern and all the incoming flights to this airport during the planning horizon. The costs include the costs of delaying the flights at their origin airports and the costs incurred when flights experience airborne holding near the destination. In a sequential decision model of the SAGHP, an action is to ground-hold a set of flights at a given decision epoch. The state of the system is defined by the arrival capacity, which is not affected by the action, the flights waiting to depart, the in-transit flights, and the flights in the airborne queue. We assume that the arrival capacity at the destination airport evolves in a probabilistic fashion. A policy for this sequential decision model is a sequence of rules that specify the action to be chosen at a particular time. The rules depend on the present state together with all previous states and actions.

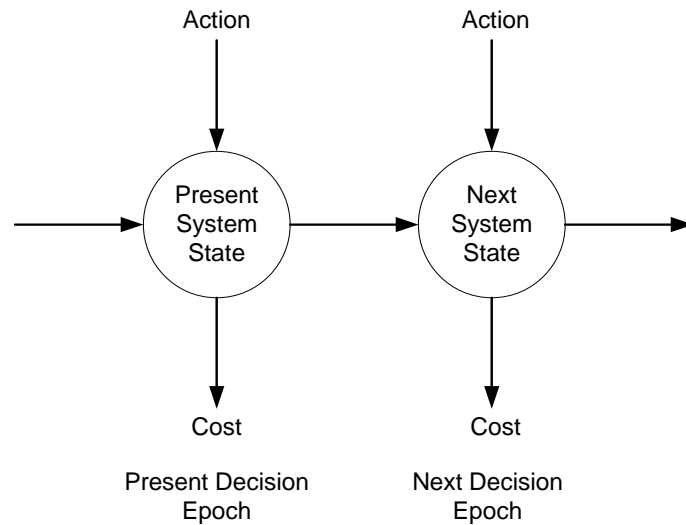


Figure 4-1 Illustration of the sequential decision model

This Sequential Decision Model (SDM) is, in fact, a degenerated finite-horizon Markov Decision Process (MDP) in which the effect of an action is deterministic. In a classical MDP, the transition of system state is probabilistic, depending on both the current state and the action. Though the statement still holds true for this SDM, the stochastic component in the system—the arrival capacity—clearly does not have dependency on the ground-holding decision; that is, the arrival capacity is an exogenous variable for this system. Thus, for SAGHP, we can classify the components of system state into those that are exogenous and stochastic and those that are endogenous and deterministic.² The SDM can be described schematically as in Figure 4-2.

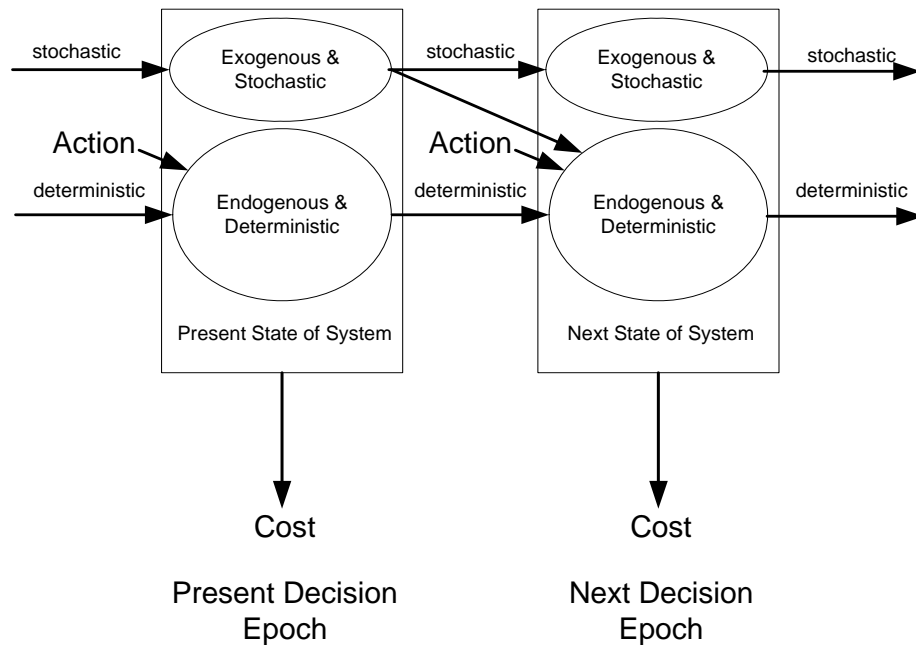


Figure 4-2 Illustration of the sequential decision model for the SAGHP

² Same as in the scenario-based models, we do not consider pop-up flights and flight cancellations and we assume the flight durations are deterministic.

We formulate this sequential decision model using a dynamic program. In this dynamic program, the “state” we respond dynamically to is the exogenous and stochastic component of the system state; i.e., the arrival capacity at the destination airport. The dynamics among the endogenous and deterministic components of the system state is described by a set of equations. We assume the availability of transition probability matrices of airport arrival capacities at each time period, which may be stationary or time-varying. For simplicity, we assume the arrival capacity evolution is a first-order Markov process. We also assume that both the queue on the ground and the queue in the air will clear at the end of the planning horizon. As such, no delay cost will be incurred after the planning horizon. We introduce the following notation:

F = the set of all the flights considered in the planning horizon.

T = the last time period in the planning horizon.

$$X_f^t = \begin{cases} 1 & \text{if flight } f \text{ stays on the ground during time period } t \\ 0 & \text{otherwise} \end{cases} \quad \forall f, \forall t.$$

$$Y_f^t = \begin{cases} 1 & \text{if flight } f \text{ is planned by the model to arrive in time period } t \\ 0 & \text{otherwise} \end{cases} \quad \forall f, \forall t.$$

$$S_f^t = \begin{cases} 0 & \text{if flight } f \text{ is scheduled to depart in or before time period } t \\ 1 & \text{otherwise} \end{cases} \quad \forall f, \forall t.$$

τ_f = the duration of flight of flight f .

A_t = the number of flights planned by the model to arrive in time period t .

L_t = number of flights intending to land at time t .

W_t = number of flights experiencing airborne delay at time t .

K_t = arrival capacity at the destination airport at time t .

$K_{\max}(t)$ = maximum arrival capacity at the destination airport at time t .

$K_{\min}(t)$ = minimum arrival capacity at the destination airport at time t .

$P_{kk'}$ = transition probability from arrival capacity k to k' in the next period.

c_g = cost of ground delay for one time period per flight.

c_a = cost of airborne delay for one time period per flight.

$f_t^p(K_t)$ = the minimal expected cost-to-go at time t , when the state-action path to reach this decision point is p , and the arrival capacity at the destination airport is K_t .

Among these variables, only X_f^t is a decision variable. S_f^t and τ_f are given from the flight schedule and it is assumed that the duration of flight is deterministic. Y_f^t and A_t are auxiliary variables to help describe the dynamics. The optimality equation can be stated using the notation defined above as:

$$f_t^p(K_t) = \min_{X_f^t, f \in F} \left\{ c_g \sum_{f \in F} (X_f^t - S_f^t) + c_a W_t + \sum_{K_{t+1}=K_{\min}(t+1)}^{K_{\max}(t+1)} P_{K_t, K_{t+1}} f_{t+1}^{p'}(K_{t+1}) \right\}, 1 \leq t \leq T-1; \quad (4.1.1)$$

$$\text{s.t.} \quad Y_f^{t+1+\tau_f} = X_f^t - X_f^{t+1} \quad \forall f \in F, t = 0, \dots, T-1; \quad (4.1.2)$$

$$A_t = \sum_{f \in F} Y_f^t \quad t = 0, \dots, T; \quad (4.1.3)$$

$$W_t = (L_t - K_t)^+ \quad t = 1, \dots, T; \quad W_0 = 0;^3 \quad (4.1.4)$$

$$L_t = A_t + W_{t-1} \quad t = 1, \dots, T; \quad (4.1.5)$$

$$X_f^t \geq S_f^t \quad \forall f \in F, t = 0, \dots, T; \quad (4.1.6)$$

$$X_f^t \geq X_f^{t+1} \quad \forall f \in F, t = 0, \dots, T-1; \quad (4.1.7)$$

³ $x^+ \equiv \max(x, 0)$.

$$X_f^t \in \{0, 1\}, \quad Y_f^t \in \{0, 1\} \quad \forall f \in F, t = 0, \dots, T; \quad (4.1.8)$$

$$\text{and} \quad f_T^p(K_T) = c_a W_T. \quad (4.1.9)$$

When the difference $X_f^t - S_f^t$ equals to one, the flight f experiences ground holding during period t . Therefore, the number of flights to hold in period t is the sum of such differences over all the flights. Constraint set (4.1.2) ties the planned arrival time of flight f ($t + 1 + \tau_f$) to its departure time ($t + 1$) and its duration (τ_f) through the auxiliary variable Y_f^t . Equation set (4.1.3) calculates the number of flights planned to arrive in time t , A_t . Constraint set (4.1.5) states that the number of flights desiring to land in period t includes those planned to arrive in period t and those queued in the air in period $t - 1$. Constraint set (4.1.6) ensures that a flight does not take off until its scheduled departure time.

In this formulation, the total delay cost is optimized on the basis of ground holding decisions for each individual flight. Clearly, the action space (to hold or not to hold each flight) is combinatorial with worst case of $2^{|F|}$ per period and the computation is likely to be intractable. The computational load can be greatly reduced when we consider flights in groups classified by flight durations. When a group of flights of the same duration are candidates for release in a given period, it is clear that the objective function depends only on the number of flights from this group to release, not on which individuals to release. We thus reformulate the problem by grouping the flights in F according to their durations. We introduce the following notations for the reformulation.

Γ = the set of groups that represent the classification of flights.

Z_γ^t = number of group γ flights to hold on the ground in time period t .

ξ_γ^t = number of group γ flights planned by the model to arrive in time period t .

S_γ^t = number of group γ flights scheduled for departure at time t .

G_γ^t = number of group γ flights queued on ground from previous periods by time t .

The reformulated optimality equation can be expressed as:

$$f_t^p(K_t) = \min_{\substack{0 \leq Z_\gamma^t \leq G_\gamma^t + S_\gamma^t \\ \gamma \in \Gamma}} \left\{ c_g \sum_{\gamma \in \Gamma} Z_\gamma^t + c_a W_t + \sum_{K_{t+1}=K_{\min}(t+1)}^{K_{\max}(t+1)} P_{K_t, K_{t+1}} \cdot f_{t+1}^{p'}(K_{t+1}) \right\}, \quad 1 \leq t \leq T-1; \quad (4.1.10)$$

$$\text{s.t.} \quad G_\gamma^t = Z_\gamma^{t-1} \quad \forall \gamma \in \Gamma, \quad t = 1, \dots, T; \quad G_\gamma^0 = 0, \quad \forall \gamma \in \Gamma; \quad (4.1.11)$$

$$\xi_\gamma^{t+\tau_\gamma} = G_\gamma^t + S_\gamma^t - Z_\gamma^t \quad \forall \gamma \in \Gamma, \quad t = 0, \dots, T-1; \quad (4.1.12)$$

$$A_t = \sum_{\gamma \in \Gamma} \xi_\gamma^t \quad t = 1, \dots, T; \quad (4.1.13)$$

$$W_t = (L_t - K_t)^+ \quad t = 1, \dots, T; \quad W_0 = 0; \quad (4.1.14)$$

$$Z_\gamma^t \in Z^+ \quad \forall \gamma \in \Gamma, \quad t = 0, \dots, T-1; \quad (4.1.15)$$

$$\text{and} \quad f_T^p(K_T) = c_a W_T. \quad (4.1.16)$$

In the optimality equation (4.1.10), the decision variable is the number of flights to hold for each duration group γ in time period t , Z_γ^t . Constraint set (4.1.11) associates the action in period $t-1$ to its implication for the queue on the ground in period t . Constraint set (4.1.12) links the departure and planned arrival of flights with their flight durations. Constraint set (4.1.13) calculates the number of flights planned to land in time t , A_t , by summing over the number of flights planned to land in period t from all the duration groups.

The computational advantage of the reformulation can be illustrated using a simple example. Consider nine flights waiting for departure in the current time period. Suppose

these nine flights belong to three groups of sizes two, three, and four. Using formulation (4.1.1), there are 512 (29) candidate holding action combinations to evaluate.

When solving for the number of flights to hold per group as in formulation (4.1.10), only 60 ($3 \times 4 \times 5$) action combinations need to be evaluated. The model (4.1.10) reduces the action space significantly; at the same time, the model yields the same optimal policy.

Before we discuss the mathematical properties of the model, let us consider a simple numerical example and illustrate the mechanics of the decision process. Figure 4-3 depicts the sequential decision making process for one of the simplest SAGHP. In this example, there is one flight scheduled to depart at time 0 with flight duration of one time period. Destination capacity levels are 0 and 1 with a stationary transition probability

matrix of $\begin{matrix} & \begin{matrix} 0 & 1 \end{matrix} \\ \begin{matrix} 0 \\ 1 \end{matrix} & \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix} \end{matrix}$. The capacity in the initial period is 0. Unit delay costs are $c_g = 1$

and $c_a = 3$. There is a planning horizon of two periods, and it is guaranteed that the flight can land in the third period. The decision process is described by an acyclic directed graph with two types of nodes: decision nodes and action nodes. We refer to such a graph an *SDM tree*. The decision node with capacity level k at decision period t is marked by “ $K_t = k$.” The action node with the number of flights to hold h at decision period t is marked by “ $H_t = h$.” The associated number of flights in the ground queue (G_t) and number of flights in the airborne queue (W_t) are displayed next to each of the nodes. Also listed are the costs evaluated at each step. At a decision node, the decision maker selects the least costly admissible action. The value of the least-cost action chosen, $f_t^*(K_t)$, is the cost associated with this decision node at period t with capacity level K_t . At an action node, the decision maker evaluates the expected cost-to-go of taking this particular action.

The expected cost-to-go associated with each actions is displayed as “ $V = \text{cost.}$ ” In this example, the optimal policy is to hold the flight at period 0. At period 1, however, depending on the capacity level $K1$, the optimal action could be to hold or not to holding the flight. As this example illustrates that there are quite a few calculations required even for a very small problem. This suggests that the dimension of the SDM tree could be huge for real-world problems.

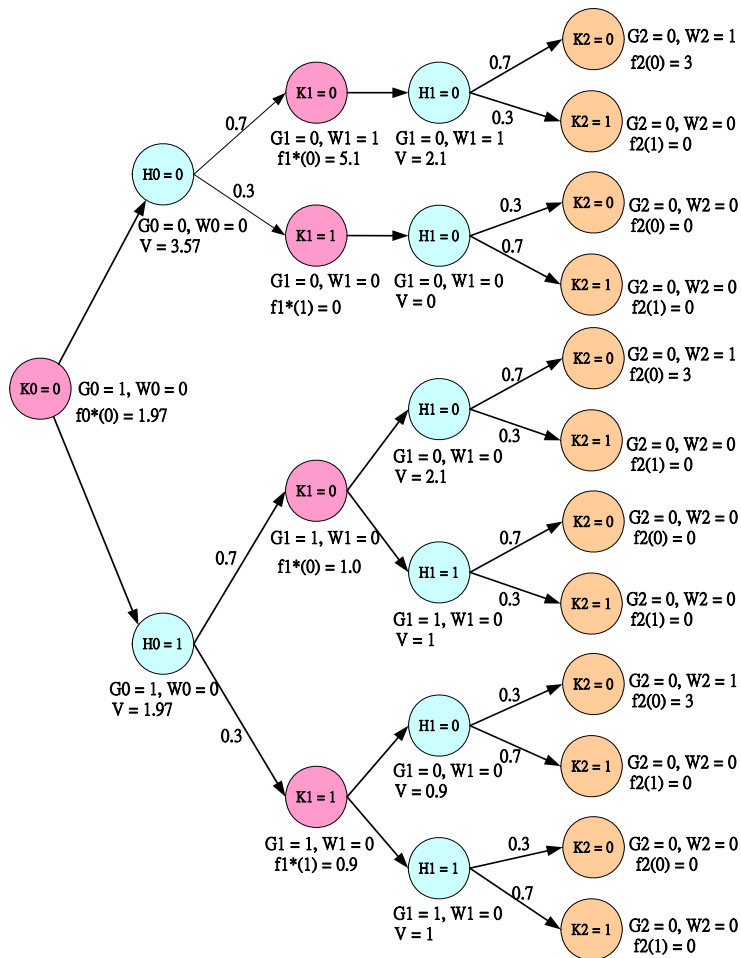


Figure 4-3 Simple example of sequential decision model of the SAGHP rendered as an SDM tree

4.1.1 Algorithmic complexity

We compute the optimal policy for the dynamic program using the *value iteration* algorithm. Traditionally, two algorithms—policy iteration and value iteration—are considered for such problem. In our case, each of the iterations in the policy iteration algorithm involves a protracted iterative computation requiring sweeps through the state set. For that reason, we draw on the value iteration algorithm for our dynamic program. The value iteration algorithm is obtained simply by taking the optimality equation,

(4.1.1) or (4.1.10), as an update rule. In other words, this algorithm performs two operations at each decision point: policy evaluation and policy improvement. The policy evaluation step computes expected cost-to-go and the costs incurred if certain action were taken in the current period. The policy improvement step chooses the action that will lead to the least cost. While the state-action space is traversed recursively, the total number of policy evaluation operations equals to the number of action nodes, and the total number of policy improvement operations equals to the number of decision nodes, in the terminology used in Figure 4-3. As a result, the worst-case complexity of this algorithm can be derived via counting the number of nodes in the tree. Let us define:

T = the number of decision epochs in the planning horizon;

N = the number of capacity levels considered in a period;

M = the number of actions possible in a decision stage;

F = the number of flights to release in a decision stage;

G = the number of groups of flights.

The computational time for this value iteration algorithm is allocated to the following basic operations:

1. *Policy improvement.* The policy improvement operations are performed at each of the decision node. The decision nodes in the tree representation of the sequential decision model include the special nodes—the root node and the leaf nodes—and other ordinary decision nodes. The number of decision nodes in the SDM tree is

$$1 + MN + (MN)^2 + \cdots + (MN)^T = \frac{(MN)^{T+1} - 1}{MN - 1}.$$

The policy improvement step requires $O(1)$ time to retrieve the minimum from a priority queue with values each associated with a capacity level state. Therefore, the total policy improvement time is $O((MN)^T)$.

2. *Policy evaluation.* The policy evaluation operations are performed at each of the action node. The number of the action nodes in the MDP tree is

$$M + M(MN) + M(MN)^2 + \cdots + M(MN)^{T-1} = M \cdot \frac{(MN)^T - 1}{MN - 1}$$

Each policy evaluation step requires $O(1)$ time for the arithmetic operations. Therefore, the total policy evaluation time is $O(M^T N^{T-1})$.

The number of admissible actions in a decision stage, M , however, is a derived information. For the SDM model described by (4.1.10), M is derived from the number of flights waiting to be released in the decision stage (F) and the number of flight groups (G). When $G = 1$, there are $F + 1$ possible actions—holding no flight, holding one flight, holding two flights, and so on up to holding all F flights. When $G = 2$, the admissible actions depend on the number of flights classified into each group. Let F_i denote the number of flights in group i , then the number of possible actions is $(F_1+1) \cdot (F_2+1)$. The number of admissible actions at a decision stage can be generalized to $\prod_{i=1}^G (F_i + 1)$. For

this worst-case complexity analysis, we consider the scenario suggested by the math program

$$\begin{aligned} & \max \prod_{i=1}^N x_i \\ & \text{s.t. } \sum_{i=1}^N x_i = K, \quad x_i \in \mathfrak{R} \quad \forall i. \end{aligned}$$

The solution to this nonlinear program, $x_i = \frac{K}{N}, \forall i$, can be obtained from the optimality condition of its Lagrangian function. This property gives us the following lemma concerning the impact of the number of flight groups on the size of the action set.

Lemma 4.1 *The upper bound of the size of the action set at a decision stage is $(\frac{F}{G} + 1)^G$.*

Between the operations of policy evaluation and policy improvement, we find the computational expense of policy improvement dominating that of policy evaluation. Based on Lemma 4.1, we have established the following result.

Theorem 4.2 *The value iteration algorithm solves the dynamic program for SAGHP described by (4.1.10) in $O(((\frac{F}{G})^G N)^T)$ time.*

Consider the existence of a priority ordering among the flights. Such ordering is summoned in current practice to ensure the equity in assigning delays to flights. From an optimization point of view, a priority ordering imposes constraints for making the optimal ground-holding decisions. Unlike the original problem (4.1.10) that solves for the optimal combination of flights to hold from each group, with a priority ordering the flights to hold are determined based on the ordering and we only need to solve for the optimal combination of flights to hold from each group. In this setup, the problem reduces to the special case of $G = 1$ as stated in the following corollary.

Corollary 4.3 *If there exists a priority ordering among the flights such that the grouping of flights is not needed, the value iteration algorithm solves the dynamic program for SAGHP in $O((FN)^T)$ time.*

The complexity presented above shows that the value iteration algorithm for this problem is an exponential-time algorithm. Note that this worst-case complexity is an upper bound on the running time for sufficiently large values of the parameters inside the big O notation. The running time of a problem depends also on the actual magnitude of these parameters and use of problem-specific computational strategies. In addition, heuristics that have potential to cut down redundant computations can be helpful. In Section 4.2 and 4.3, we explore several strategies to make a typical real-world problem solvable within reasonable time.

4.2 Computational Solution Strategies

The algorithmic complexity results in the previous section indicate the need for computational solution strategies to reduce the time complexity for problems of realistic size. In this section, we describe the use of memoization, priority ordering, and heuristics to alleviate the computational load. In addition, we present a high-level design for implementation of the algorithm at the end of this section.

4.2.1 Memoization

An important feature that an optimization problem must have for dynamic programming to be applicable is that the algorithm encounters the same subproblems over and over; that is, the optimization problem has many overlapping subproblems (Cormen et al., 1985). Fortunately, it is the case for our problem. To illustrate the

overlapping subproblems in the sequential decision model for SAGHP, let us examine a simple example. Consider the problem of one flight departing at time 0 with flight duration of one time period, two possible arrival capacity levels—0 and 1, and three time periods in the planning horizon. The SDM tree of this problem is shown in Figure 4-4.

Due to the setup of a singleton flight with flight duration of one period, the state of the system can be fully characterized by time, capacity level, the number of flight waiting to be released on the ground (Gt), and the size of the airborne queue (Wt). When the algorithm traverses through the tree, we can see that the algorithm repeatedly visits the same states and solves for the same subproblems. For decisions at time period 2, four of the subproblems (Subproblem A, B, C and D) appear more than once. In fact, at time period 2, there are only five distinct subproblems out of the 12 subproblems. Clearly, for bigger problem instances the savings in computation can be significant if the subproblems are not solved over and over again.

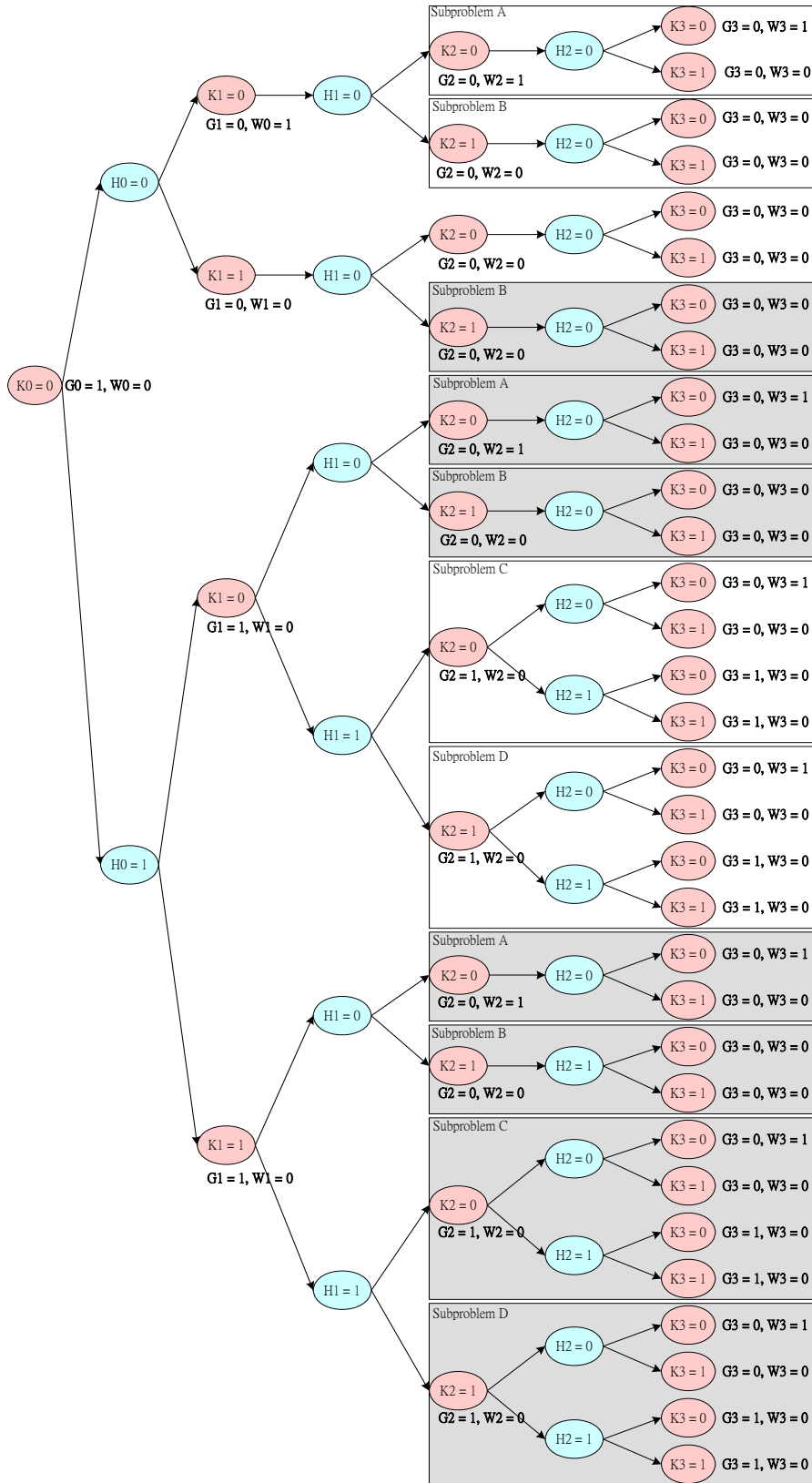


Figure 4-4 Overlapping subproblems in an SDM tree

Memoization is an algorithmic technique used to accelerate computation by storing the results of subproblems for later reuse, rather than re-solving them. Memoizing the top-down recursive algorithm is a variation of dynamic programming which traditionally solves the problems from bottom up. The bottom-up approach starts the computation from identifying and solving all the subproblems at the bottom level. To give an inkling of the predicament in the bottom-up approach, consider a SAGHP of hundreds of flights with various flight durations. The subproblems are characterized by the attributes of the flights waiting in the ground queue and the flights released but yet to land, in addition to the time and the capacity level at the destination airport. The combinatorial effect of these parameters can produce a huge number of subproblems. However, some of the subproblems need not be solved at all because they would arise only from clearly suboptimal upstream decisions that would never be made. Therefore, for the dynamic program for SAGHP, the top-down memoized algorithm has the advantage of only solving those subproblems that are definitely required, and solving them only once.

A memorized recursive algorithm can be implemented by maintaining an entry in a table for the solution to each subproblem. Upon recurrence of a subproblem, the value stored in the table is looked up and retrieved. This approach constructs a table to hold values for the set of all possible subproblems in the subproblem space. Since many subproblems need not be solved at all, a table with all the needed subproblem solutions is actually a sparse high-dimension cube. For compactness in storage and efficiency in retrieval, we memoize by hashing with the subproblem parameters as keys. Hence, the crux of this computational strategy is to identify the necessary and sufficient parameters that fully characterize a distinct subproblem. For clarity, let us restate the notations here.

Notation t denotes the time period, K_t denotes the arrival capacity at period t , G_t denotes the number of flights waiting to be released at the origin in period t , and W_t denotes the length of the airborne queue. From the problem illustrated in Figure 4-4, it is direct to obtain the following.

Proposition 4.4 *For problems of one flight with flight duration of one period, the quadruple (t, K_t, G_t, W_t) is necessary and sufficient to characterize a subproblem.*

The above proposition can be generalized for problems with multiple flights and various flight durations. With multiple flights and multiple possible durations, the lengths of queues are not enough to characterize the agent state. Different mixes of flight durations for the flights in the ground queue pose different problems for the agent. The flights that have been released constitute a sequence of arrival demand in the future periods at the destination airport. Different such sequences also imply different problems for the agent. Let us restate/define the following notations for problems with multiple flights and multiple flight durations.

T = the last time period in the planning horizon.

Φ_t = the set of flights ready to be released in period t , including those queued on the ground from previous periods and those scheduled to take off in period t .

τ_f = the flight duration of flight f .

B_t = the *bag*⁴ of flight durations for the flights intending to be released in period t .
 $= \{\tau_f \mid f \in \Phi_t\}$

⁴ A *bag* is a collection with possible duplicates as opposed to a *set*, which has no duplicate elements.

Alternatively, a bag is called a *multiset*. A vector is, therefore, an ordered bag.

A_t = the number of flights released and planned to arrive in period t .

V_t = the vector of the number of flights released and planned to arrive after period t
= $\langle A_{t+1}, A_{t+2}, \dots, A_T \rangle$.

Also, let us refer to the flights ready to be released as *ready flights*. For the decision maker, five ready flights with durations of 2, 2, 3, 3, and 4 periods presents the same decision-making problem as another five ready flights with durations 2, 3, 4, 2, and 3 periods. Hence, two sets of ready flights that project the same bag of flight durations pose the same problem to the decision maker. At the same time, each unique arrival vector V_t for the in-transit flights presents a different problem for the agent. Besides, the parameters t , K_t , and W_t also take part in defining a distinct subproblem in this multiple flight, multiple duration setup. As a result, we find that the parameters t , K_t , W_t , B_t , and V_t fully characterize a subproblem.

Proposition 4.5 *Quintuple (t, K_t, B_t, V_t, W_t) is necessary and sufficient to characterize a subproblem for problems of multiple flights with various flight durations.*

The quintuple in Proposition 4.6 is, in fact, the characterization of the mathematical state in the sequential decision model, where the same policy is optimal for the same state. This mathematical state is slightly different from the *system state* mentioned earlier with the addition of the time period marker t . In the dynamic program, however, the “state” used in the formulation is the *capacity state*, as it is the only stochastic component considered in the system. We make the distinction between capacity state and system state as follows.

Definition 4.6 The *capacity state* changes due to uncontrollable exogenous factors. The *capacity state* fully reflects the stochastic process considered in the model.

Definition 4.7 The *system state* is defined by all the parameters considered by the agent in the process of making an optimal decision. Every action the agent takes moves the agent to the next system state.

This definition of capacity state extracts out the constituents that are controlled solely by exogenous factors from the mathematical state. This isolation makes the states in the optimality equations (4.1.1) and (4.1.10) more straightforward. Two of the components in the quintuple, B_t and V_t , are collections of elements. To avoid the overhead in preparing these collections beforehand, in actual implementation B_t and V_t are constructed dynamically in the recursive algorithm. Along with t , K_t , and W_t , these five parts make up the key for the solution of a subproblem stored in a hash table. During the traversal of the SDM tree, whenever the algorithm reaches a decision node, it constructs the quintuple of system state and checks the existence of this key in the hash table. If the key has been registered in the hash table, the retrieval of the solution to the overlapped subproblem costs only $O(1)$ time.

The use of memoization has been shown in standard textbooks (e.g. Cormen et al., 1985) to turn an algorithm with exponential asymptotic lower bound $\Omega(2^n)$ into a polynomial-time algorithm for certain simple problem. Due to the intricacy in identifying distinct subproblems in our subproblem space, we do not attempt to estimate the effect of memoization in big- O notation. Instead, we show the effect of memoization by experimental results in Section 4.3.

4.2.2 Priority Ordering

In Subsection 4.1.1 we discussed the value of priority ordering among flights in the reduction of algorithmic complexity. We showed that the existence of priority ordering of

flights reduces the time complexity of the algorithm from $O\left(\left(\frac{F}{G}\right)^G N^T\right)$ to $O((FN)^T)$.

Though the use of memoization changes the complexity developed previously, the savings from transforming an exponential component to polynomial should nevertheless be considerable. Therefore, we consider priority ordering as a second computational solution strategy.

One sensible way to prioritize flights is to release the longer duration flights first. This is justified for two reasons. First, delays to flights with longer duration typically have higher economic cost. For long haul flights, airlines usually use larger aircraft, and carry more passengers. Moreover, holding long haul flights on the ground incurs unrecoverable delay costs which may prove unnecessary if destination airport capacity evolves favorably. Short haul flights, in contrast, can be managed in a more responsive manner as capacity evolves over time. For these reasons, it is sensible to give flights with longer flight durations the priority to take off. This is recognized, albeit somewhat bluntly, in the current practice by “exempting” long-haul flights from GDPs. Here, we do not exempt such flights, but rather allow them to take-off ahead of shorter flights. We term this prioritization scheme *Longest Goes First* or LGF.

A second prioritization scheme, *Ration by Schedule* (RBS), is employed in current practice for non-exempt flights. As its name suggests, RBS gives flights with earlier scheduled arrival time the priority. The airline community considers RBS the most equitable method of allocating capacity. In contrast to LGF, RBS protects short-haul carriers and flights from bearing the brunt of GDPs. Thus, while LGF may be a more efficient scheme, RBS is a more equitable one, at least from the airlines’ point of view.

(One could also employ a weighted priority scheme that considers both the schedule and flight duration, but this will not be investigated here.)

The value iteration algorithm for the sequential decision model needs to be modified when a priority ordering is employed in the decision process. Let us define the following notation:

F_p = the set of *ready flights* at decision node p in the SDM tree;

π_f = the priority score for flight f —the higher the score, the higher the priority;

Ω_p = the set of all possible sets of flights to hold at decision node p

$$= \{ \{\emptyset\}, \{f_1^p\}, \{f_1^p, f_2^p\}, \dots, \{f_1^p, f_2^p, \dots, f_{|F_p|}^p\} \},$$

where $\pi_{f_i^p} \leq \pi_{f_{i+1}^p}, \forall f_j^p \in F_p$.

The optimality equation, or the update rule in the algorithm, can be stated as:

$$f_t^p(K_t) = \min_{\omega \in \Omega_p} \left\{ c_g |\omega| + c_d W_t + \sum_{K_{t+1}=K_{\min}(t+1)}^{K_{\max}(t+1)} P_{K_t K_{t+1}} \cdot f_{t+1}^{p'}(K_{t+1}) \right\} \quad (4.2.1)$$

With a priority ordering, the admissible actions are a set of sets of flights to hold. Each set contains different number of flights, and the sets are constructed based on the priority ordering. The ground delay cost incurred for each action chosen is a function of the number of flights subject to ground holding ($|\omega|$). Let us algorithmically describe the recursive updates and the dynamics between the release of flights and the future airborne queue. Same as before, let A_t denote the number of flights planned by the model to arrive in time period t , and τ_f denote the duration of flight f . In addition, we define the following:

$V_p(\omega)$ = the cost of holding the set of flights ω at decision node p ;

V_p^* = the least possible cost-to-go at decision node p ;

$Q(p, \omega)$ = the set of decision nodes after holding flights in ω at decision node p ;

$A_t^{p, \omega}$ = the number of flights planned to arrive in time period t if flights in ω are held at decision node p .

The algorithm is as follows.

begin

decision node p := the root node of the SDM tree;

$A_t := 0$ and $W_t := 0, \forall t$.

vector $A := \langle A_1, A_2, \dots, A_T \rangle$, vector $W := \langle W_1, W_2, \dots, W_T \rangle$;

minimum cost = *ComputeCost*(p, A, W);

end

The subroutine *ComputeCost*(p, A, W) takes a decision node p , vectors A and W as input and returns the least expected cost evaluated at decision node p . The algorithm of this subroutine is as follows.

begin

if decision node p is a leaf node then

return $V_p^* = c_a \cdot (A_T + W_{T-1} - K_T)^+$;

else

for each set of flights $\omega \in \Omega_p$ do

```


$$A_t^{p,\omega} := A_t \quad \forall t;$$

for each  $f \in F_p \setminus \omega$  do
    
$$A_{t+\tau_f}^{p,\omega} := A_{t+\tau_f}^{p,\omega} + 1;$$

end

$$W_{t(p)} := (A_{t(p)} + W_{t(p)-1} - K_{t(p)})^+;^5$$

for each decision node  $q \in Q(p, \omega)$  do
    
$$A'_t := A_t^{p,\omega} \quad \forall t;$$


$$A' := \langle A'_1, A'_2, \dots, A'_T \rangle; W' := \langle W_1, W_2, \dots, W_T \rangle;$$

    compute  $V_q^* = \text{ComputeCost}(q, A', W')$ ;
end
    compute and record  $V_p(\omega) = c_g |\omega| + c_a W_{t(p)} + \sum_{q \in Q(p, \omega)} E[V_q^*]$ ;
end
return  $V_p^* = \min_{\omega \in \Omega_p} V_p(\omega)$ ;
end

```

As a result of the above algorithm, the nodes in the SDM tree are constructed in the order as a tree is traversed in a depth-first-search manner. Although a sorting operation is involved in the construction of the set Ω_p , the search of optimal policy at each decision stage is reduced from a combinatorial space to one dimension. The assignment of

⁵ Function $t(p)$ gives the time period associated with the decision node p .

$A_t^{p,\omega} := A_t$ passes down a cloned copy of the arrival plan, which the decision node p inherited from its parent, to the descendants of the decision-action pair (p, ω) . This way, the increment of $A_t^{p,\omega}$ wouldn't affect the arrival plan of other paths in the tree.

Note that we present a generic algorithm for the dynamic program for SAGHP with priority ordering instead of one that is specific priority ordering rule such as LGF or RBS. The algorithm is devised such that it is not coupled with a particular priority ordering rule. In this fashion, we provide the flexibility of adopting priority orderings that are calculated based on any scoring scheme. The priority scoring scheme for LGF is simply

$$\pi_f^{LGF} = \text{duration of flight } f = \tau_f.$$

Let S_f denote the scheduled departure time for flight f . Then the priority scoring function for RBS can be specified as

$$\begin{aligned} \pi_f^{RBS} &= \text{the negativity of the scheduled arrival time of flight } f \\ &= -(S_f + \tau_f). \end{aligned}$$

The discussions so far have not addressed the effect of priority ordering on the *keys* used in memoization. In fact, the priority ordering scheme LGF has no effect on the composition of the keys since LGF prioritizes using the duration of flight which is already part of the constituent of a key. On the other hand, the priority ordering scheme RBS introduces an additional element—scheduled arrival time—to the characterization of the system state. Hence, in the case with RBS priority ordering, the quintuple that characterizes a distinct subproblem is (t, K_t, B_t', V_t, W_t) , where B_t' is the bag of the flight duration and scheduled arrival time pairs for the flights ready to be released in period t . That is, $B_t' = \{(\tau_f, S_f + \tau_f) \mid f \in \Phi_t\}$.

4.2.3 Heuristics for Searching for Best Action

During numerical experiments we observe that the cost-to-go functions, oftentimes, are convex in the number of flights to hold. From our experience, it is especially true when all the flights have the same duration. Convexity of the cost-to-go function can be very helpful in cutting down the search for the minimum. Thus, this subsection pursues this opportunity. The following is our theorem.

Theorem 4.8 *The cost-to-go function (4.1.10)⁶*

$$f_t^p(K_t) = \min_{\substack{0 \leq Z_\gamma^t \leq G_\gamma^t + S_\gamma^t \\ \gamma \in \Gamma}} \left\{ c_g \sum_{\gamma \in \Gamma} Z_\gamma^t + c_a W_t + \sum_{K_{t+1}=K_{\min}^{(t+1)}}^{K_{\max}^{(t+1)}} P_{K_t, K_{t+1}} \cdot f_{t+1}^{p'}(K_{t+1}) \right\}$$

is convex in $Z_\gamma^t, \forall \gamma \in \Gamma$, where Γ is the set of duration-based groups of flights.

Proof.

i. First, let us expand several terms in the function where Z_γ appear.

$$\begin{aligned} W_{t+1} &= (L_{t+1} - K_{t+1})^+ \\ &= (A_{t+1} + W_t - K_{t+1})^+ \\ &= \left(\sum_{\gamma \in \Gamma} \xi_\gamma^{t+1} + W_t - K_{t+1} \right)^+ \\ &= \left[\sum_{\gamma \in \Gamma} (G_\gamma^{t+1-\tau_\gamma} + S_\gamma^{t+1-\tau_\gamma} - Z_\gamma^{t+1-\tau_\gamma}) + W_t - K_{t+1} \right]^+ \end{aligned}$$

The cost-to-go function can be expanded slightly into:

$$f_t^p(K_t) = \min_{\substack{0 \leq Z_\gamma^t \leq G_\gamma^t + S_\gamma^t \\ \gamma \in \Gamma}} \left\{ c_g \sum_{\gamma \in \Gamma} Z_\gamma^t + c_a W_t + \sum_{K_{t+1}=K_{\min}^{(t+1)}}^{K_{\max}^{(t+1)}} P_{K_t, K_{t+1}} \cdot f_{t+1}^{p'}(K_{t+1}) \right\}$$

⁶ The notations used here are the same as those described in Section 4.1 in association with (4.1.10).

$$\begin{aligned}
&= \min_{\substack{0 \leq Z_\gamma^t \leq G_\gamma^t + S_\gamma^t \\ \gamma \in \Gamma}} \left\{ c_g \sum_{\gamma \in \Gamma} Z_\gamma^t + c_a W_t + \sum_{K_{t+1}=K_{\min}(t+1)}^{K_{\max}(t+1)} P_{K_t, K_{t+1}} \cdot \min_{0 \leq Z_\gamma^{t+1} \leq Z_\gamma^t + S_\gamma^{t+1}} \left\{ c_g \sum_{\gamma \in \Gamma} Z_\gamma^{t+1} + c_a W_{t+1} \right. \right. \\
&\quad \left. \left. + \sum_{K_{t+2}=K_{\min}(t+2)}^{K_{\max}(t+2)} P_{K_{t+1}, K_{t+2}} \cdot f_{t+2}^{p''}(K_{t+2}) \right\} \right\} \\
&= \min_{\substack{0 \leq Z_\gamma^t \leq G_\gamma^t + S_\gamma^t \\ \gamma \in \Gamma}} \left\{ c_g \sum_{\gamma \in \Gamma} Z_\gamma^t + c_a W_t \right. \\
&\quad \left. + \sum_{K_{t+1}=K_{\min}(t+1)}^{K_{\max}(t+1)} P_{K_t, K_{t+1}} \cdot \min_{\substack{0 \leq Z_\gamma^{t+1} \leq Z_\gamma^t + S_\gamma^{t+1} \\ \gamma \in \Gamma}} \left\{ c_g \sum_{\gamma \in \Gamma} Z_\gamma^{t+1} + c_a \left[\sum_{\gamma \in \Gamma} (G_\gamma^{t+1-\tau_\gamma} + S_\gamma^{t+1-\tau_\gamma} - Z_\gamma^{t+1-\tau_\gamma}) + W_t - K_{t+1} \right]^+ \right. \right. \\
&\quad \left. \left. + \sum_{K_{t+2}=K_{\min}(t+2)}^{K_{\max}(t+2)} P_{K_{t+1}, K_{t+2}} \cdot f_{t+2}^{p''}(K_{t+2}) \right\} \right\}.
\end{aligned}$$

- ii. Consider a specific group of flights γ_0 . Suppose we are solving for the optimal $Z_{\gamma_0}^t$ for the cost-to-go function $f_t^p(K_t)$. This problem can be formulated as

$$\begin{aligned}
f_t^p(K_t) &= \min_{0 \leq Z_{\gamma_0}^t \leq G_{\gamma_0}^t + S_{\gamma_0}^t} \left\{ c_g (Z_{\gamma_0}^t + \sum_{\gamma \in \Gamma \setminus \{\gamma_0\}} Z_\gamma^t) + c_a W_t + \sum_{K_{t+1}=K_{\min}(t+1)}^{K_{\max}(t+1)} P_{K_t, K_{t+1}} \cdot f_{t+1}^{p'}(K_{t+1}) \right\} \\
&= \min_{0 \leq Z_{\gamma_0}^t \leq G_{\gamma_0}^t + S_{\gamma_0}^t} \left\{ c_g Z_{\gamma_0}^t + c_g \sum_{\gamma \in \Gamma \setminus \{\gamma_0\}} Z_\gamma^t + c_a W_t \right. \\
&\quad \left. + \sum_{K_{t+1}=K_{\min}(t+1)}^{K_{\max}(t+1)} P_{K_t, K_{t+1}} \cdot \min_{\substack{0 \leq Z_{\gamma_0}^{t+1} \leq Z_{\gamma_0}^t + S_{\gamma_0}^{t+1} \\ 0 \leq Z_\gamma^{t+1} \leq G_\gamma^{t+1} + S_\gamma^{t+1} \\ \gamma \in \Gamma \setminus \{\gamma_0\}}} \left\{ c_g \sum_{\gamma \in \Gamma} Z_\gamma^{t+1} + c_a \left[\sum_{\gamma \in \Gamma} (G_\gamma^{t+1-\tau_\gamma} + S_\gamma^{t+1-\tau_\gamma} - Z_\gamma^{t+1-\tau_\gamma}) + W_t - K_{t+1} \right]^+ \right. \right. \\
&\quad \left. \left. + \sum_{K_{t+2}=K_{\min}(t+2)}^{K_{\max}(t+2)} P_{K_{t+1}, K_{t+2}} \cdot f_{t+2}^{p''}(K_{t+2}) \right\} \right\}. \tag{4.2.2}
\end{aligned}$$

Since the decision variable for this minimization is $Z_{\gamma_0}^t$, we need to examine the terms associated with $Z_{\gamma_0}^t$ in the arborescent form of the equation. In addition to in the first term $c_g Z_{\gamma_0}^t$, $Z_{\gamma_0}^t$ would appear in a set of $c_a \left[\sum_{\gamma \in \Gamma} (G_\gamma^{t+i-\tau_\gamma} + S_\gamma^{t+i-\tau_\gamma} - Z_\gamma^{t+i-\tau_\gamma}) + W_{t+i-1} - K_{t+i} \right]^+$

terms for the airborne delay cost in period $t + i$ where $Z_{\gamma_0}^{t+i-\tau_{\gamma_0}} = Z_{\gamma_0}^t$. That is, for each K_{t+i} such that $K_{\min(t+i)} \leq K_{t+i} \leq K_{\max(t+i)}$, the following terms need to be considered:

$$c_a[(G_{\gamma_0}^t + S_{\gamma_0}^t - Z_{\gamma_0}^t) + \sum_{\gamma \in \Gamma \setminus \{\gamma_0\}} \xi_{\gamma}^{t+i} + W_{t+i-1} - K_{t+i}]^+ . \quad (4.2.3)$$

Due to (4.1.11), $Z_{\gamma_0}^t$ also appears in the terms in period $t + i + 1$, where $i = \tau_{\gamma_0}$

$$\begin{aligned} & c_a[(G_{\gamma_0}^{t+1} + S_{\gamma_0}^{t+1} - Z_{\gamma_0}^{t+1}) + \sum_{\gamma \in \Gamma \setminus \{\gamma_0\}} \xi_{\gamma}^{t+i+1} + W_{t+i} - K_{t+i+1}]^+ \\ & = c_a[(Z_{\gamma_0}^t + S_{\gamma_0}^{t+1} - Z_{\gamma_0}^{t+1}) + \sum_{\gamma \in \Gamma \setminus \{\gamma_0\}} \xi_{\gamma}^{t+i+1} + W_{t+i} - K_{t+i+1}]^+ . \end{aligned} \quad (4.2.4)$$

The terms associated with $Z_{\gamma_0}^t$ in equation (4.2.2) include $c_g Z_{\gamma_0}^t$, the terms stated by (4.2.3) and the terms in the form of (4.2.4). Apparently $c_g Z_{\gamma_0}^t$ is convex over $Z_{\gamma_0}^t$. It is also obvious that the hockey stick-shaped max function $(.)^+$ is convex. Hence, all of the terms described by (4.2.3) and (4.2.4) are convex over $Z_{\gamma_0}^t$. Note that one of the decision variables in the minimization in period $t + 1$ in (4.2.2), $Z_{\gamma_0}^{t+1}$, has a range of $0 \leq Z_{\gamma_0}^{t+1} \leq G_{\gamma_0}^{t+1} + S_{\gamma_0}^{t+1} = Z_{\gamma_0}^t + S_{\gamma_0}^{t+1}$, where $Z_{\gamma_0}^t$ appears as one of the components of the upper bound. If the upper bound for $Z_{\gamma_0}^{t+1}$ is not binding, then this $Z_{\gamma_0}^t$ is not a parameter in the function minimized in period $t + 1$. In the case that the upper bound is binding, the minimizer $Z_{\gamma_0}^{t+1}$ is equal to $Z_{\gamma_0}^t + S_{\gamma_0}^{t+1}$ and this $Z_{\gamma_0}^t$ appears in the subsequent terms. And again, these terms are all convex for the aforementioned reasons. The terms containing $Z_{\gamma_0}^t$ in (4.2.2), except for the first ground delay cost term $c_g Z_{\gamma_0}^t$, are all within some expectation functions. Expected value of a function over stochastic parameters is a positive linear combination of the function values for each possible realization of the stochastic event. Because positive linear combinations

of convex functions are convex (Zangwill, 1969:32-33), these expectation functions are convex. As a result, $f_t^p(K_t)$ is a convex combination of convex functions and, hence, convex itself.

iii. Since $f_t^p(K_t)$ is convex over $Z_{\gamma_0}^t$ for a specific group of flights $\gamma_0 \in \Gamma$, it is immediate that $f_t^p(K_t)$ is convex over Z_γ^t , for each $\gamma \in \Gamma$. □

The result in Theorem 4.8 highlights the structural properties of our cost-to-go function. For unconstrained minimization problems with continuous decision variables, setting the partial derivatives of the objective function with respect to the decision variables equal to zero and solving them would lead to an optimal solution. However, our decision variables are discrete—non-negative integers—and our function is not differentiable—piece-wise linear due to the max function $(.)^+$. For these reasons, though the cost-to-go function is convex over the decision variable for each flight group, we have a combinatorial optimization problem. A simple example of such cost-to-go function with two groups of flights γ_0 and γ_1 at time period 0 is illustrated in Figure 4-5.

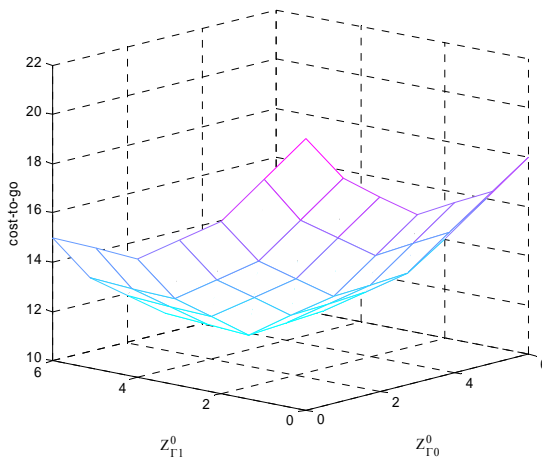


Figure 4-5 Illustration of cost-to-go function with two groups of flights

Earlier we mentioned that we observed the convexity of the cost-to-go function over the number of flights to hold from our numerical experiments when all the flights have the same duration. Based on Theorem 4.8, it is direct to arrive at the following corollary, which formalizes what we observed.

Corollary 4.9 *Consider the case where there is only one duration-based group of flights denoted by γ_0 . If all the flights have the same flight duration, the cost-to-go function*

$$f_t^p(K_t) = \min_{0 \leq Z_{\gamma_0}^t \leq G_{\gamma_0}^t + S_{\gamma_0}^t} \left\{ c_g Z_{\gamma_0}^t + c_a W_t + \sum_{K_{t+1}=K_{\min}(t+1)}^{K_{\max}(t+1)} P_{K_t, K_{t+1}} \cdot f_{t+1}^{p'}(K_{t+1}) \right\} \text{ is convex in } Z_{\gamma_0}^t.$$

Yet, the setup of only one duration-based group of flights in Corollary 4.9 does not happen in the real world. As discussed in Subsection 4.2.2, consideration of priority ordering in the decision-making reduces the problem from a combinatorial space to one dimension. Thus, we examine the structural properties of the cost-to-go function when a priority ordering is adopted in the decision-making process.

Let us point out one important property of the components in the function. The cost-to-go function contains components in the pattern of $f(x) = c_g x + p_1 c_a (a-x)^+ + p_2 c_a (b-x)^+$. The function $f(x)$ can be shown graphically as the superposition of three functions of x in Figure 4-6 when $c_g = 1$, $c_a = 3$, $p_1 = 0.4$, $p_2 = 0.6$, $a = 2$, and $b = 4$. The kinks in the function are at the points where a $(\cdot)^+$ function hits zero. The function is *increasing* to the right of the rightmost kink point. As we increase the number of flights to hold, the total delay cost would go down when the marginal ground delay cost is lower than the marginal airborne delay cost. At a certain number of flights to hold, holding more simply increases the ground delay cost and does not reduce the airborne delay cost in expectation.

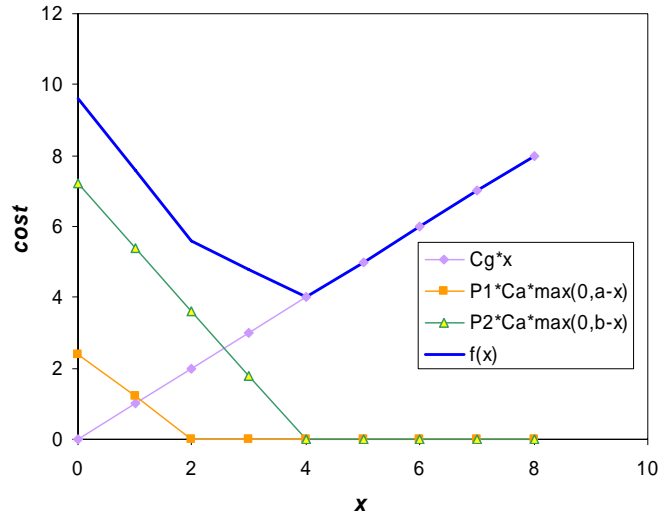


Figure 4-6 Superposition of elements in the cost-to-go function

Consider the case when a duration-based priority ordering such as LGF is in effect. Flights in the group of the shortest duration will be held first. Only after all the flights in the shortest-duration group are held, the flights in the group of the second shortest duration will be subject to holding. The effect of priority ordering on the cost-to-go function can be demonstrated using a simplified example. Consider a function $f(x) = f(x_1+x_2) = f_1(x_1) + f_2(x_2)$, where $f_i(x_i) = c_g x_i + p_{1,i} c_a (a_i - x_i)^+ + p_{2,i} c_a (b_i - x_i)^+$. Suppose $c_g = 1$, $c_a = 3$, $p_{1,1} = 0.4$, $p_{2,1} = 0.6$, $a_1 = 2$, $b_1 = 4$, $p_{1,2} = 0.7$, $p_{2,2} = 0.3$, $a_2 = 1$, and $b_2 = 3$. Also, it is assumed that the feasible range for integers x_1 and x_2 is $0 \leq x_1, x_2 \leq 8$. The variables x_1 and x_2 are analogous to the number of flights to hold for flight group 1 and 2. The priority ordering dictates that x_2 can be greater than 0 only when x_1 equals 8; i.e., $(x_1 - 8) \cdot x_2 = 0$. As illustrated in Figure 4-7, although it is cheaper to hold one flight than to hold no flight from group 2 ($f_2(0) = 4.8$, $f_2(1) = 2.8$), no flight in group 2 can be held until all the flights in group 1 are held. The cost reduction from holding flights from group 2 after holding all

the group 1 flights caused a dip in the cost-to-go function. As such, the cost-to go function is not convex and we have the following corollary.

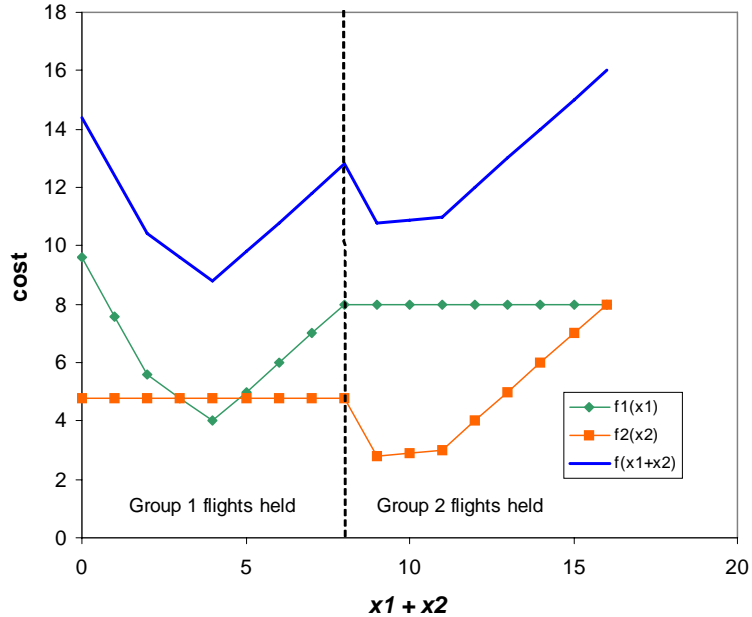


Figure 4-7 Effect of priority ordering LGF on the cost-to-go function

Corollary 4.10 *Upon the employment of a priority ordering rule, the convexity of the cost-to-go function described in Theorem 4.8 and Corollary 4.9 cannot be guaranteed.*

From the reasoning and demonstration above, we can see that the cost-to-go function is *piece-wise convex*—the convexity breaks when the next group of flights is considered for holding. Although it is not as perfect as a convex function, this piece-wise convex function gives us an opportunity to cut down the computations by searching for the minimum as if it were a convex function.

The search for the minimum on a piece-wise linear convex function can be performed from either ends of the domain. The function value is evaluated for each integer input argument. The function values should decrease as we proceed until the

minimum is reached. Once an increase in the function value is observed, the previous input argument is found as the minimizer. For the reason that the cost-to-go function is not truly convex, the computational strategy of treating it as a convex function gives us heuristics that cut back the computation time while providing a suboptimal but good solution. We develop two heuristic algorithms that we refer to as H1 and H2.

To describe these heuristics, we introduce the following notation. Let n be the number of flights ready to be released at period t . Denote the cost-to-go at decision node p when holding x flights by $f_p(x)$. Also, let x^* denote the best number of flights to hold determined by a given algorithm. The heuristic algorithm H1 is described as follows.

Step 1: At decision node p , compute the cost-to-go for the actions at the two ends of the cost function; that is, releasing none of the flights, $f_p(n)$, and releasing all of the flights, $f_p(0)$.

Step 2: If $f_p(0) \leq f_p(n)$, set $x^* := 0$ and go to Step 3; otherwise, set $x^* := n$ and go to Step 4.

Step 3: FOR $x = 1$ to $n-1$ DO:

If $f_p(x) > f_p(x - 1)$, then $f_p(x - 1)$ is a local minimum. Set $x^* := x - 1$ and stop.

Else continue.

Step 4: FOR $x = n-1$ to 1 DO:

If $f_p(x) > f_p(x + 1)$, then $f_p(x + 1)$ is a local minimum. Set $x^* := x + 1$ and stop.

Else continue.

Step 5: x^* is the best number of flights to hold at decision node p and $f_p(x)$ is the lowest cost found by this algorithm.

Step 1 and 2 are devised to cut down the chase and pick one direction to search from. If we were to search from both sides, we would often evaluate the function over the entire range and thus defeat the purpose of reducing computations. This algorithm finds the optimal action when the cost-to-go function is convex in the number of flights to hold. When the function is convex over the domain to either side of the minimum, this algorithm has chance of finding the optimal action as well. When the case is none of the above, this algorithm cannot guarantee finding the optimum and it gives a solution that is a local optimum.

To further reduce the computations and arrive at suboptimal but probably good solutions, we propose heuristic algorithm H2. In this heuristic, we always begin our search from zero. This approach is motivated by the observation that the function tends to be check-mark-shaped (\checkmark) especially when there are many *ready flights*. It is because that the cost is contributed mainly by the ground delay when the number of flights being held is relatively high. This observation hints that the lower end of the domain is likely to be the better side to start searching for the minimum. Thus heuristic algorithm H2 is as follows.

Step 1: At decision node p , compute the cost-to-go $f_p(0)$.

Step 2: FOR $x = 1$ to $n-1$ DO:

If $f_p(x) > f_p(x - 1)$, then $f_p(x - 1)$ is a local minimum. Set $x^* := x - 1$ and stop.

Else continue.

Step 3: x^* is the best number of flights to hold at decision node p and $f_p(x)$ is the lowest cost found by this algorithm.

It seems that Heuristic H2 skips only a few steps in heuristic H1. Considering that these operations are performed at each decision node in the SDM tree, cumulatively these skips can save a sizable number of computations for a large problem. Note that the previous discussion on the effect of priority ordering on convexity considered only the duration-based ordering, LGF. The flights in groups defined under the priority ordering scheme RBS are not as homogeneous as under LGF, in terms of having the same attribute that are intrinsic to the flights (scheduled arrival time is a “derived” attribute). Due to the heterogeneous nature of the flight groups, we do not discuss further on the structural properties of the cost-to-go function with the RBS ordering rule. Yet we expect the cost-to-go functions to behave, in general, similar to those with the LGF ordering. Since the heuristics H1 and H2 are not coupled with a particular priority ordering, they can be applied when RBS is the acting priority ordering rule.

While H2 may seem to reduce computational burden only slightly compared to H1, it can make a considerable difference in computation time. Considering that these heuristics are applied at each decision node in the SDM tree, cumulatively H2 can save a sizable number of computations for a large problem. The effect is multiplicative with the number of time periods, so that a 10% saving over one period becomes a 2/3 saving over 10 periods.

4.2.4 Implementation Design

The algorithm that determines the optimal policy for the dynamic program for SAGHP can be implemented using any general contemporary programming languages. In this subsection, we briefly describe our high-level design for implementation, including the architecture and several key objects.

We employ the Model/View/Controller (MVC) architecture⁷ for our software. In this architectural style, the subsystems of the program are classified into: model—where the domain knowledge is maintained, view—description (graphical or textual) of the system to the user, and controller—the mechanism that manages the interactions with the user. This architecture can be visualized using a UML (Unified Modeling Language) class diagram as in Figure 4-8.

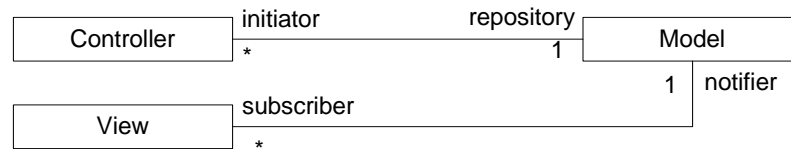


Figure 4-8 Illustration of the MVC architecture⁸

The Controller subsystem collects basic information of the SAGHP from the user, including the unit cost of delays, number of periods in the planning horizon, the flight schedule, the initial capacity level, the capacity transition matrices, and the algorithmic features for computation (e.g. use ordering LGF or RBS, heuristic H1 or H2). Upon user request, the Controller stores the information in the Model and kicks off the computation for the optimal policy. In our design, the Controller functionalities are handled by one user interface object.

The Model subsystem maintains the data store for the basic information collected by the Controller and manages the data structure for the MDP tree. The static problem

⁷ Detailed description of this architecture can be found in standard texts such as *Object-Oriented Software Engineering* by Bruegge and Dutoit, 2003.

⁸ This visual is adopted from Bruegge and Dutoit (2003), page 239.

specification of the SAGHP is maintained in an *Environment* object which provides getter and setter functions for the information. The hash table that supports memoization is also maintained in *Environment*. The SDM tree is composed of two classes of objects: *Decision Node* and *Action Node*. A *Decision Node* object is responsible for generating a set of valid *Action Node* objects as its children (unless it is a leaf node) and performing a policy improvement operation. An *Action Node* object is responsible for generating a set of *Decision Node* objects as its children and performing a policy evaluation operation. The SDM tree is seeded with a root *Decision Node* object. Based on problem specifications, the tree grows and simultaneously computes for optimal policy. By the time the tree is full-grown, the computation for optimal policy is done.

The View subsystem provides interfaces that organize and show the information stored in the model, including the user input and the optimization results. In our design, several objects are constructed to display system information in the format of tables and tree graphs. These interfaces are useful during software development and result analysis.

4.3 Computational Experiments

In this section, we present computational results for our sequential decision model for SAGHP. We programmed the model using Java JDK 5.0. We present the computational results from running the Java program on either a personal computer with Windows XP or a server with Linux operating system. The personal computer runs with a 2.8 GHz Pentium 4 CPU and 504 MB of RAM. The Linux server is equipped with two 3.4 GHz CPUs and 1,034 MB of memory with 2,097 MB of swap memory. On the personal computer, the computational experiments can be conducted in a more controlled

environment where the computing power is not shared with other users. However, our program is memory intensive such that this personal computer runs out of memory during the computation for larger problem instances. Furthermore, it takes longer to solve the same problem on the personal computer because of its slower CPU speed. Hence, for larger problems, we resort to the Linux server for its CPU and memory equipage.

This section is organized as follows. We first discuss the behavior of computation time with respect to different parameters and problem setups. Next, we present the effects of computational solution strategies that are faster, but may lead to compromises on solution quality (such as priority ordering and heuristics). Finally, we show our results from solving a real-world problem instance.

4.3.1 Computation Time

From our discussions on algorithmic complexity in previous sections, we mentioned that when memoization is applied the theoretical time complexity is not easy to estimate and, since it is based on a worst case, may not be a good indicator for the actual computation time. The computation time is of great concern for our model since the worst-case complexity is exponential before the algorithm is memoized. For this reason, we experimentally investigate the behavior of computation time with different parameter setups when memoization is employed.

A set of computational experiments are designed for understanding better the behavior of computation time under different circumstances. In each experiment, we observe the effect of one parameter on the running time while keeping other parameters constant. As noted above, when experiments are run on the Linux server, we cannot ensure dedicated computational power for them. Thus, when we record the running times,

we take the reading of the shortest running time among five to ten runs. The number of runs we conduct depends on the variability of the initial runs—if results vary significantly we carry out more runs. The running time recorded this way is clearly an upper bound for the best computation time that can be achieved with our computing environment.⁹ Our experimental results may thus be slightly distorted by the shared-server environment, but we are satisfied that by using repeated runs as necessary we have made this effect quite small.

4.3.1.1 Effect of Memoization

Memoization is a technique to accelerate computation by trading memory space with speed. The solution to the problem is not affected when solutions to the subproblems are memoized. We present the running time results for problems with different total number of flights to show the effect of memoization. We assume there are three possible arrival capacity levels in each period: one, three, or five flights and the initial capacity level at three flights. For simplicity, we assume the transition matrix, as shown in Table 4-1, is time-invariant in the planning horizon of six time periods.¹⁰ We consider a flight schedule with nine flights of two-period duration (Table 4-2). These test problems, and many others in the subsequent experiments, are devised to have single flight duration so that the computation is less demanding and we get to explore the effect of other problem parameters in a wider range. The effect of multiple flight durations is discussed later in Subsection 4.3.1.2. We increase the number of flights to include in the test problem

⁹ We turn off the graphical rendering of results during the experiments so that the running time would not include the overhead for the graphics.

¹⁰ We assume that the capacity is infinite in the seventh period.

based on the index of the flights. For example, we include Flight 1 for problem with one flight; Flight 1 and Flight 2 for problem with two flights.

Table 4-1 Capacity (flights/time period) Transition Matrix for the Test Problem on the Use of Memoization

| from \ to | 1 | 3 | 5 |
|-----------|-----|-----|-----|
| 1 | 0.3 | 0.5 | 0.2 |
| 3 | 0.2 | 0.6 | 0.2 |
| 5 | 0.1 | 0.4 | 0.5 |

Table 4-2 Flight Schedule for the Test Problem on the Use of Memoization

| Flight | Departure Period | Arrival Period |
|--------|------------------|----------------|
| 1 | 0 | 2 |
| 2 | 1 | 3 |
| 3 | 2 | 4 |
| 4 | 3 | 5 |
| 5 | 0 | 2 |
| 6 | 1 | 3 |
| 7 | 2 | 4 |
| 8 | 3 | 5 |
| 9 | 0 | 2 |

Figure 4-9 shows that without memoization, the computation time goes up exponentially as the number of flights increases. With memoization, the computation time also goes up at a super-linear rate but the increase is more gradual. Most importantly, our results show that the computation time without memorization, even for these small problems, is on the order of thousand-fold greater than that with memoization. Given that we observe problem instances taking several minutes to solve even with memoization, the applicability of the model in practice as well as our ability to assess model performance experimentally obviously hinges on the use of memoization. Accordingly, we use this technique in all of our subsequent experiments.

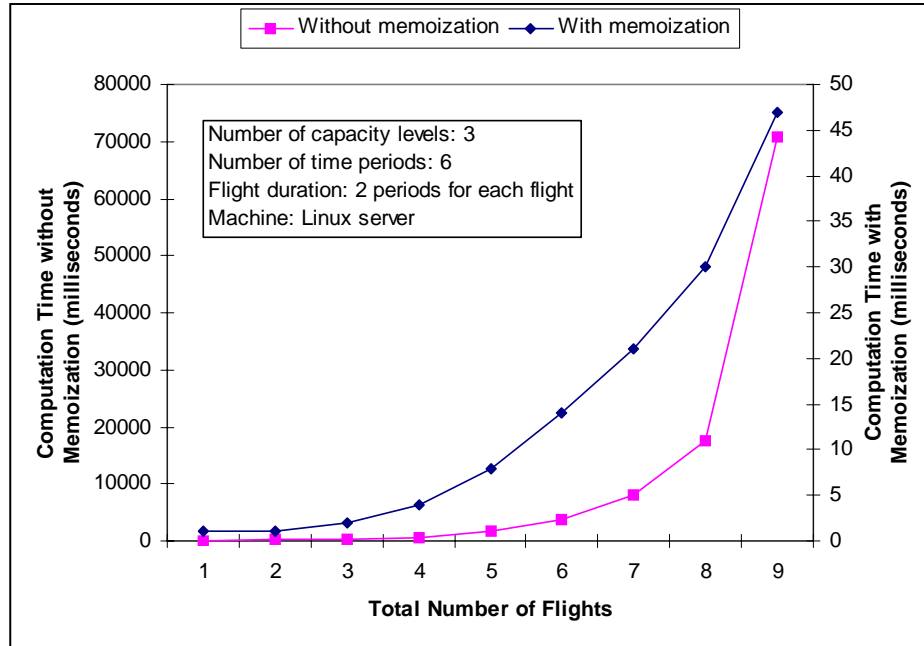


Figure 4-9 Computation time with and without memoization

4.3.1.2 Effect of Problem Parameters

The relationship between computation time and key problem parameters—the number of capacity levels, the total number of flights, and the length of planning horizon—while memoization is applied are exhibited in Figure 4-11, Figure 4-12, and Figure 4-13, respectively. As before, in these three experiments we simplify by assuming the same flight duration for all the flights. At the end of this subsection we show the results for problems with heterogeneous flight durations. In designing the experiments, we set parameters so as to obtain runtime results in a reasonable range—the same setup may result in less than 10 milliseconds in computation time for one experiment but minutes to hours for another. In Subsection 4.1.1, we showed that the theoretical time complexity of the algorithm without memoization is $O((FN)^T)$ when all the flights have

the same flight duration. Here we compare variation in actual running times and theoretical computation time (without memoization) with each parameter of interest.

To show the effect of the number of capacity levels (N) on the running time we consider a test problem with a six period planning horizon and 10 flights, whose schedule is shown in Table 4-3. We solve the problem with different number of capacity levels, ranging from one to ten. We construct ten transition matrices for different number of capacity levels—from 1-by-1 matrix to 10-by-10 matrix (Figure 4-10), and we assume the initial capacity is 2. We performed this experiment in the XP environment.

Table 4-3 Flight Schedule for the Test Problem of Different Capacity Levels

| Flight | Departure Period | Arrival Period |
|--------|------------------|----------------|
| 1 | 0 | 2 |
| 2 | 0 | 2 |
| 3 | 1 | 3 |
| 4 | 1 | 3 |
| 5 | 1 | 3 |
| 6 | 2 | 4 |
| 7 | 2 | 4 |
| 8 | 2 | 4 |
| 9 | 3 | 5 |
| 10 | 4 | 6 |

| | |
|-----------|---|
| from \ to | 2 |
| 2 | 1 |

| | | |
|-----------|-----|-----|
| from \ to | 2 | 3 |
| 2 | 0.6 | 0.4 |
| 3 | 0.4 | 0.6 |

| | | | |
|-----------|-----|-----|-----|
| from \ to | 1 | 2 | 3 |
| 1 | 0.1 | 0.6 | 0.3 |
| 2 | 0.1 | 0.4 | 0.5 |
| 3 | 0.1 | 0.6 | 0.3 |

| | | | | |
|-----------|-----|-----|-----|-----|
| from \ to | 1 | 2 | 3 | 4 |
| 1 | 0.1 | 0.5 | 0.3 | 0.1 |
| 2 | 0.1 | 0.4 | 0.4 | 0.1 |
| 3 | 0.1 | 0.5 | 0.3 | 0.1 |
| 4 | 0.1 | 0.3 | 0.5 | 0.1 |

| | | | | | |
|-----------|-----|-----|-----|-----|-----|
| from \ to | 1 | 2 | 3 | 4 | 5 |
| 1 | 0.1 | 0.4 | 0.3 | 0.1 | 0.1 |
| 2 | 0.1 | 0.4 | 0.3 | 0.1 | 0.1 |
| 3 | 0.1 | 0.3 | 0.4 | 0.1 | 0.1 |
| 4 | 0.1 | 0.3 | 0.4 | 0.1 | 0.1 |
| 5 | 0.1 | 0.2 | 0.4 | 0.2 | 0.1 |

| | | | | | | |
|-----------|-----|-----|-----|-----|-----|-----|
| from \ to | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0.1 | 0.2 | 0.2 | 0.3 | 0.1 | 0.1 |
| 2 | 0.1 | 0.2 | 0.2 | 0.3 | 0.1 | 0.1 |
| 3 | 0.1 | 0.1 | 0.2 | 0.4 | 0.1 | 0.1 |
| 4 | 0.1 | 0.1 | 0.2 | 0.4 | 0.1 | 0.1 |
| 5 | 0.1 | 0.2 | 0.2 | 0.2 | 0.2 | 0.1 |
| 6 | 0.1 | 0.2 | 0.2 | 0.2 | 0.2 | 0.1 |

| | | | | | | | |
|-----------|------|------|------|------|------|------|------|
| from \ to | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 0.10 | 0.20 | 0.20 | 0.30 | 0.10 | 0.05 | 0.05 |
| 2 | 0.10 | 0.20 | 0.20 | 0.30 | 0.10 | 0.05 | 0.05 |
| 3 | 0.10 | 0.20 | 0.20 | 0.30 | 0.10 | 0.05 | 0.05 |
| 4 | 0.10 | 0.10 | 0.20 | 0.40 | 0.10 | 0.05 | 0.05 |
| 5 | 0.10 | 0.10 | 0.20 | 0.40 | 0.10 | 0.05 | 0.05 |
| 6 | 0.10 | 0.20 | 0.20 | 0.20 | 0.20 | 0.05 | 0.05 |
| 7 | 0.10 | 0.20 | 0.20 | 0.20 | 0.20 | 0.05 | 0.05 |

| | | | | | | | | |
|-----------|------|------|------|------|------|------|------|------|
| from \ to | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | 0.05 | 0.05 | 0.20 | 0.20 | 0.30 | 0.10 | 0.05 | 0.05 |
| 2 | 0.05 | 0.05 | 0.20 | 0.20 | 0.30 | 0.10 | 0.05 | 0.05 |
| 3 | 0.05 | 0.05 | 0.20 | 0.20 | 0.30 | 0.10 | 0.05 | 0.05 |
| 4 | 0.05 | 0.05 | 0.10 | 0.20 | 0.40 | 0.10 | 0.05 | 0.05 |
| 5 | 0.05 | 0.05 | 0.10 | 0.20 | 0.40 | 0.10 | 0.05 | 0.05 |
| 6 | 0.05 | 0.05 | 0.20 | 0.20 | 0.20 | 0.20 | 0.05 | 0.05 |
| 7 | 0.05 | 0.05 | 0.20 | 0.20 | 0.20 | 0.20 | 0.05 | 0.05 |
| 8 | 0.05 | 0.05 | 0.20 | 0.20 | 0.20 | 0.20 | 0.05 | 0.05 |

| | | | | | | | | | |
|-----------|------|------|------|------|------|------|------|------|------|
| from \ to | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 | 0.05 | 0.05 | 0.10 | 0.10 | 0.20 | 0.30 | 0.10 | 0.05 | 0.05 |
| 2 | 0.05 | 0.05 | 0.10 | 0.10 | 0.20 | 0.30 | 0.10 | 0.05 | 0.05 |
| 3 | 0.05 | 0.05 | 0.10 | 0.10 | 0.20 | 0.30 | 0.10 | 0.05 | 0.05 |
| 4 | 0.05 | 0.05 | 0.10 | 0.20 | 0.20 | 0.20 | 0.10 | 0.05 | 0.05 |
| 5 | 0.05 | 0.05 | 0.10 | 0.10 | 0.10 | 0.40 | 0.10 | 0.05 | 0.05 |
| 6 | 0.05 | 0.05 | 0.10 | 0.10 | 0.20 | 0.20 | 0.20 | 0.05 | 0.05 |
| 7 | 0.05 | 0.05 | 0.10 | 0.10 | 0.20 | 0.20 | 0.20 | 0.05 | 0.05 |
| 8 | 0.05 | 0.05 | 0.10 | 0.10 | 0.20 | 0.20 | 0.20 | 0.05 | 0.05 |
| 9 | 0.05 | 0.05 | 0.05 | 0.15 | 0.20 | 0.20 | 0.20 | 0.05 | 0.05 |

| | | | | | | | | | | |
|-----------|------|------|------|------|------|------|------|------|------|------|
| from \ to | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 0.05 | 0.05 | 0.10 | 0.10 | 0.10 | 0.10 | 0.30 | 0.10 | 0.05 | 0.05 |
| 2 | 0.05 | 0.05 | 0.10 | 0.10 | 0.10 | 0.10 | 0.30 | 0.10 | 0.05 | 0.05 |
| 3 | 0.05 | 0.05 | 0.10 | 0.10 | 0.10 | 0.10 | 0.30 | 0.10 | 0.05 | 0.05 |
| 4 | 0.05 | 0.05 | 0.10 | 0.10 | 0.10 | 0.20 | 0.20 | 0.10 | 0.05 | 0.05 |
| 5 | 0.05 | 0.05 | 0.10 | 0.10 | 0.10 | 0.20 | 0.20 | 0.10 | 0.05 | 0.05 |
| 6 | 0.05 | 0.05 | 0.10 | 0.10 | 0.20 | 0.20 | 0.10 | 0.10 | 0.05 | 0.05 |
| 7 | 0.05 | 0.05 | 0.10 | 0.10 | 0.20 | 0.20 | 0.10 | 0.10 | 0.05 | 0.05 |
| 8 | 0.05 | 0.05 | 0.10 | 0.10 | 0.20 | 0.20 | 0.10 | 0.10 | 0.05 | 0.05 |
| 9 | 0.05 | 0.05 | 0.05 | 0.15 | 0.20 | 0.20 | 0.10 | 0.10 | 0.05 | 0.05 |
| 10 | 0.05 | 0.05 | 0.05 | 0.15 | 0.20 | 0.20 | 0.10 | 0.10 | 0.05 | 0.05 |

Figure 4-10 Transition matrices for different number of capacity levels

Figure 4-11 shows that as the number of capacity levels increases, the running time increases nearly linearly, far more slowly than is suggested by the complexity $O((FN)^T)$. The theoretical growth of computation time is plotted using the function $f(N) = N^6$ since we keep F constant in the experiment and consider a problem with six time periods. The growth behavior of the computation time suggests that, as the number of capacity levels increases, there is a rise in the number of overlapping subproblems as the number of capacity levels increases. Memoization exploits this to dramatically reduce the computational load.

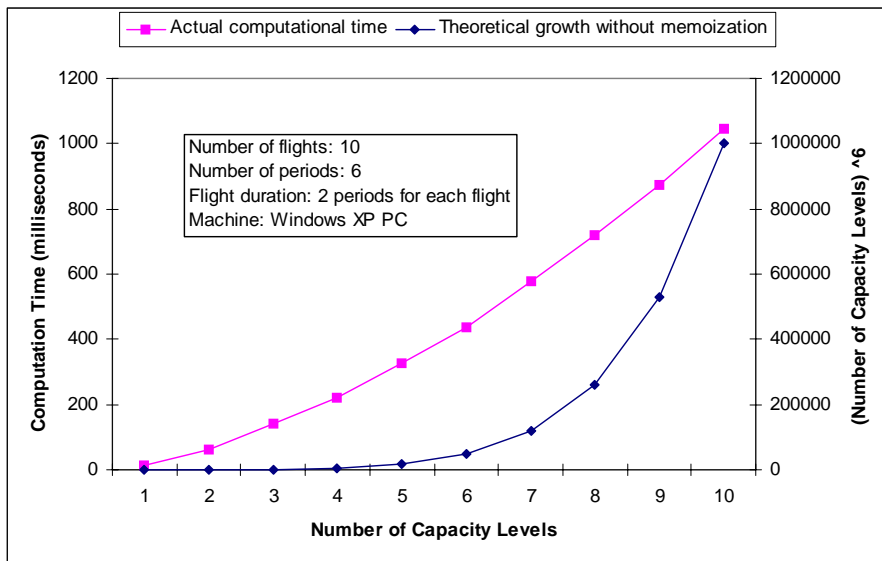


Figure 4-11 Computation time with different number of capacity levels

Next, we examine the effect of the total number of flights on computation time. Remember that the F in the big- O notation is the number of flights to release at a decision point. In this experiment, we schedule all flights for departure at time period 0. Thus, the worst-case F for each time period (holding all flights in the previous period) is the total number of flights. Again, we consider the problem with six periods in the planning horizon and transition matrices in Table 4-1. Also, we assume all flights have the same

flight duration of two periods and that the initial capacity level is 3. Figure 4-12 shows that as the total number of flights increases, the running time increases roughly by the problem size to the power of the number of time periods, as the theoretical complexity calculation suggests. Although the theoretical computation time (which ignores memoization) grows proportionally to $(FN)^T$, our experiments show that, with memoization, this relationship is preserved for F but not N . Apparently, increases in the number of overlapping subproblems do not offset the upsurge in complexity that results from increases in the number of flights.

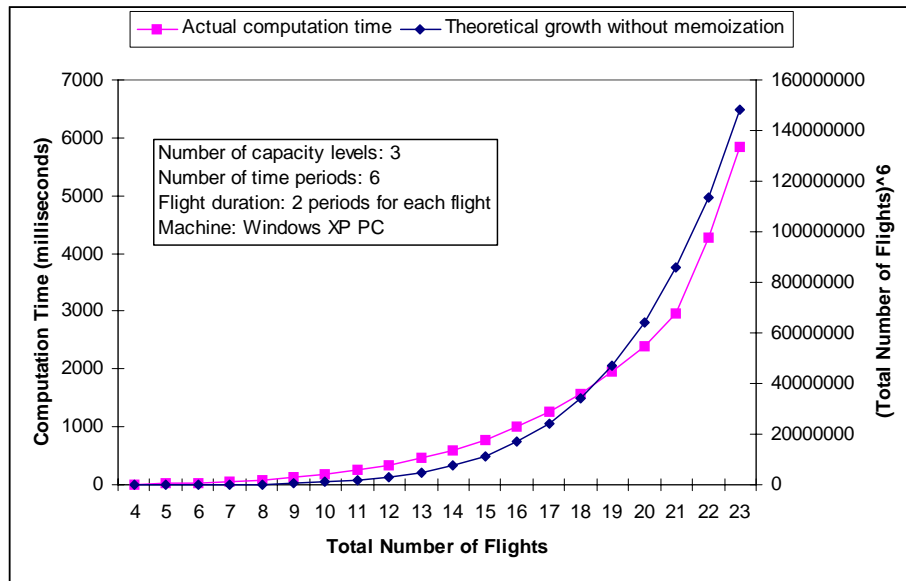


Figure 4-12 Computation time with different total number of flights

According to the worst-case complexity $O((FN)^T)$, the computation time would grow exponentially with the number of time periods in the problem. In our experiment, we set up the problem with fixed capacity transition matrix (Table 4-4), flight schedule (Table 4-5), initial arrival capacity of one flight, and test the effect of different T 's by extending the planning horizon further into the future.

Table 4-4 Transition Matrix of Test Problem for Different Planning Horizons

| from \ to | 1 | 3 | 5 |
|-----------|-----|-----|-----|
| 1 | 0.3 | 0.4 | 0.3 |
| 3 | 0.2 | 0.6 | 0.2 |
| 5 | 0.2 | 0.3 | 0.5 |

Table 4-5 Flight Schedule of the Test Problem for Different Planning Horizons

| Flight | Departure Period | Arrival Period |
|--------|------------------|----------------|
| 1 | 0 | 2 |
| 2 | 0 | 2 |
| 3 | 0 | 2 |
| 4 | 0 | 2 |
| 5 | 1 | 3 |
| 6 | 1 | 3 |
| 7 | 1 | 3 |
| 8 | 1 | 3 |
| 9 | 1 | 3 |
| 10 | 1 | 3 |
| 11 | 2 | 4 |
| 12 | 2 | 4 |
| 13 | 2 | 4 |
| 14 | 2 | 4 |
| 15 | 2 | 4 |
| 16 | 2 | 4 |
| 17 | 3 | 5 |
| 18 | 3 | 5 |
| 19 | 4 | 6 |
| 20 | 4 | 6 |

Figure 4-13 shows that the growth of computation time is not exponential, but rather only slightly super-linear, as more time periods are included in the planning horizon. This can be attributed to the use of memoization, when there is exponential growth in the number of overlapping subproblems while the growth in the number of distinct subproblems is linear. Thus, combining our results up to this point, we find the time complexity of our algorithm, with memorization, and when all flights have the same duration, is roughly proportional to $N \cdot F^T \cdot T$ rather than $(FN)^T$.

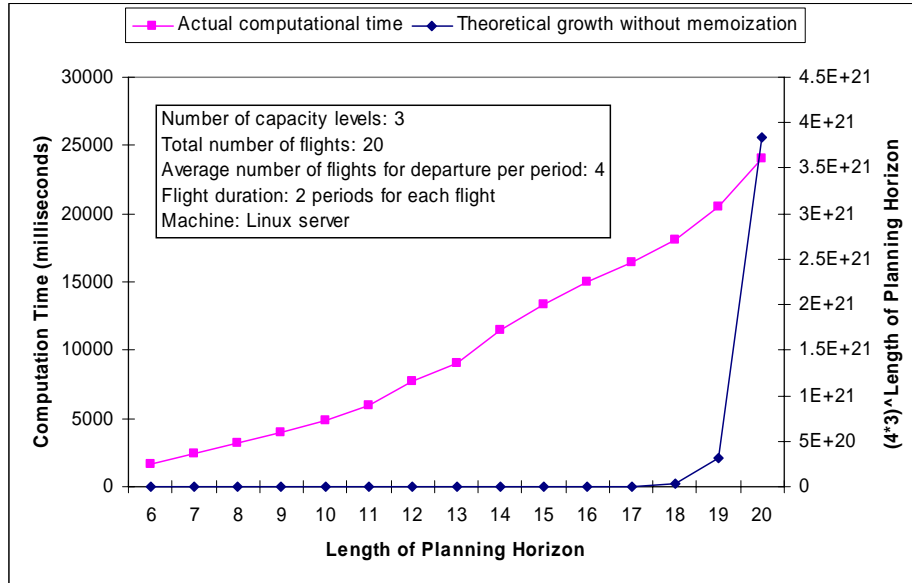


Figure 4-13 Computation time with different planning horizon

The previous experiments keep the flight duration constant for all flights. As the last topic in this subsection, we discuss the effect of multiple flight durations on the computation time. Suppose there are six flights all departing in time period 0. We consider six different mixes of flight durations as listed in Table 4-6. For example, in case 1 all the flights have the same flight duration of six periods; in case 6 each flight has a different flight duration ranging from one to six periods. This setup keeps the total number of flights constant and demonstrates the effect of a greater variety of flight durations. The model (4.1.10) computes for the best number of flights to hold for each flight group of the same flight duration. Similar as before, we assume three possible capacity levels and one capacity transition matrix (Table 4-7) throughout the planning horizon of seven periods. The initial arrival capacity level is set at two flights.

Table 4-6 Test Cases with Different Flight Duration Mixes

| Case | Number of Different Flight Durations | Duration of Flight | | | | | |
|--------|--------------------------------------|--------------------|----------|----------|----------|----------|----------|
| | | Flight 1 | Flight 2 | Flight 3 | Flight 4 | Flight 5 | Flight 6 |
| case 1 | 1 | 6 | 6 | 6 | 6 | 6 | 6 |
| case 2 | 2 | 5 | 6 | 6 | 6 | 6 | 6 |
| case 3 | 3 | 4 | 5 | 6 | 6 | 6 | 6 |
| case 4 | 4 | 3 | 4 | 5 | 6 | 6 | 6 |
| case 5 | 5 | 2 | 3 | 4 | 5 | 6 | 6 |
| case 6 | 6 | 1 | 2 | 3 | 4 | 5 | 6 |

Table 4-7 Transition Matrix for the Experiment with Multiple Flight Duration Mixes

| from \ to | 1 | 2 | 3 |
|-----------|-----|-----|-----|
| 1 | 0.4 | 0.4 | 0.2 |
| 2 | 0.2 | 0.6 | 0.2 |
| 3 | 0.2 | 0.4 | 0.4 |

Figure 4-14 shows the computation time for the test cases with different mix of flight durations. The running time grows considerably as the variety of flight durations increases. This growth is expected because the flight-holding action space is combinatorial when there are multiple flight duration lengths. We also observe a diminishing rate of computation time increase as the number of flight durations goes above four. To better understand this, we record the number of policy improvement operations (non-leaf decision nodes) performed (excluding those memoized) for each case and plot them in the same diagram. As shown in the figure, the computation time tracks very closely with the number of policy improvement steps. The smaller increment of additional policy improvement steps when the number of flight durations increases from five to six explains the slower escalation rate of running time. At the same time, this numerical result is consistent with our complexity analysis in Subsection 4.1.1 in revealing that the algorithmic complexity is dominated by the policy improvement operations.

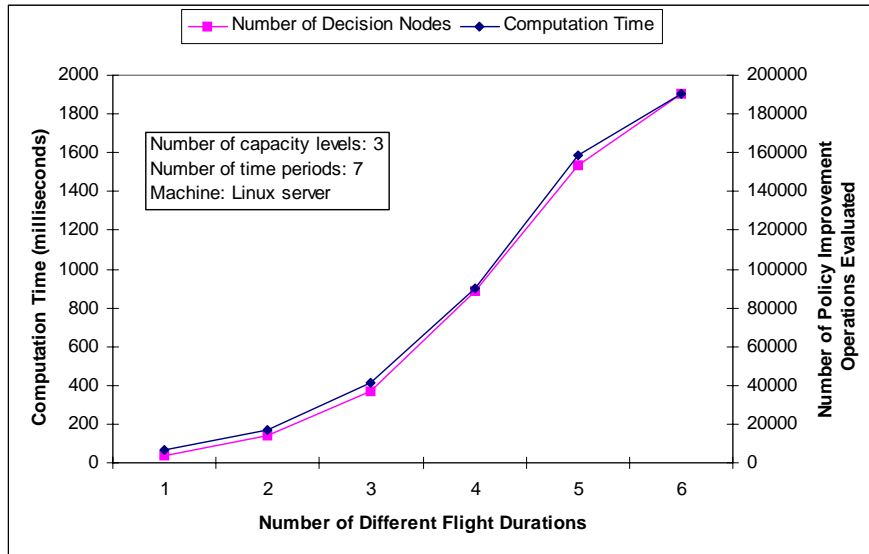


Figure 4-14 Computation time with different number of types of flight durations

To recap this subsection, we started with single flight duration and examined the effect of other problem parameters on the computation time. Though we did not derive the theoretical complexity of the algorithm when memoization is applied, we developed an understanding of the growth of the complexity with respect to the three basic parameters—number of capacity levels, number of flights, and length of planning horizon. The number of flights is found to have the strongest impact on computational load while the number of capacity levels and length of planning horizon affect the load in a more moderate fashion. Finally, the results in Figure 4-14 numerically show how variability in flight duration affects computation time. This suggests a solution strategy that avoids the combinatorial action space induced by the variety of flight durations. We discuss the use and effect of one such strategy—priority ordering—in the next subsection.

4.3.2 Effect of Computational Strategies

The problem sizes in the experiments in Section 4.3.1 are small in comparison with typical real-world problem instances which can take a lot longer to solve. To reduce the computational load so that problems of realistic scale can be handled, we consider several measures. These include time epoch aggregation, priority-based flight release, and abbreviated policy search. Unlike memoization, these measures, while reducing computation time, may also yield suboptimal solutions. Below, we describe each measure and present results of experiments designed to assess its effect on computation time and solution quality.

4.3.2.1 Time Epoch Aggregation

Time epoch aggregation is the strategy of reducing the complexity of a problem by reformulating and solving it at a coarser time scale. Suppose we have a SAGHP with a planning horizon of six hours and two flights scheduled to depart every 15 minutes in the first four hours. Each of these 32 flights has duration of two hours. We examine the effect of aggregation by considering the problem in 16 quarter-hour time periods, eight half-hour time periods, four one-hour time periods, and two two-hour time periods. In this way, the problem with quarter-hour time periods has two eight-period flights to depart per period in the first 16 periods of the 24-period planning horizon. The problem with half-hour time periods has four four-period flights to depart per period in the first eight periods of the 12-period planning horizon. The problem with one-hour and two-hour time periods are set up likewise. As the problem is aggregated in the time scale, the transition matrices are affected as well. We assume that the transition matrix for every 15 minutes

is $\begin{matrix} 0 & 1 \\ 0 & 1 \end{matrix} \begin{bmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \end{bmatrix}$. Based on this elementary transition matrix, we obtain the transition matrices

for every half an hour, one hour, and two hours as shown in Table 4-8, Table 4-9, and Table 4-10. For simplicity, we assume the initial capacity is zero for every setup. Finally, the unit cost per period of delay needs to be set to reflect the different lengths of time period in the problems.

Table 4-8 Transition Matrix for Every Half an Hour

| from \ to | 0 | 1 | 2 |
|-----------|------|-----|------|
| 0 | 0.36 | 0.4 | 0.24 |
| 1 | 0.3 | 0.4 | 0.3 |
| 2 | 0.24 | 0.4 | 0.36 |

Table 4-9 Transition Matrix for Every Hour

| from \ to | 0 | 1 | 2 | 3 | 4 |
|-----------|-------|-------|-------|-------|-------|
| 0 | 0.130 | 0.259 | 0.304 | 0.221 | 0.086 |
| 1 | 0.117 | 0.248 | 0.304 | 0.232 | 0.099 |
| 2 | 0.108 | 0.240 | 0.304 | 0.240 | 0.108 |
| 3 | 0.099 | 0.232 | 0.304 | 0.248 | 0.117 |
| 4 | 0.086 | 0.221 | 0.304 | 0.259 | 0.130 |

Table 4-10 Transition Matrix for Every Two Hours

| from \ to | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|-----------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0 | 0.0168 | 0.0635 | 0.1356 | 0.2015 | 0.2226 | 0.1857 | 0.1146 | 0.0485 | 0.0112 |
| 1 | 0.0159 | 0.0610 | 0.1321 | 0.1989 | 0.2226 | 0.1883 | 0.1181 | 0.0510 | 0.0121 |
| 2 | 0.0152 | 0.0591 | 0.1295 | 0.1969 | 0.2226 | 0.1903 | 0.1207 | 0.0528 | 0.0128 |
| 3 | 0.0146 | 0.0575 | 0.1273 | 0.1952 | 0.2226 | 0.1920 | 0.1230 | 0.0545 | 0.0134 |
| 4 | 0.0140 | 0.0560 | 0.1251 | 0.1936 | 0.2226 | 0.1936 | 0.1251 | 0.0560 | 0.0140 |
| 5 | 0.0134 | 0.0545 | 0.1230 | 0.1920 | 0.2226 | 0.1952 | 0.1273 | 0.0575 | 0.0146 |
| 6 | 0.0128 | 0.0528 | 0.1207 | 0.1903 | 0.2226 | 0.1969 | 0.1295 | 0.0591 | 0.0152 |
| 7 | 0.0121 | 0.0510 | 0.1181 | 0.1883 | 0.2226 | 0.1989 | 0.1321 | 0.0610 | 0.0159 |
| 8 | 0.0112 | 0.0485 | 0.1146 | 0.1857 | 0.2226 | 0.2015 | 0.1356 | 0.0635 | 0.0168 |

The results of this experiment are summarized in Figure 4-15. As expected, the computation time is much longer when the problem is considered in a finer scale. When each time period is 15 minutes long it takes 22,380 milliseconds to solve the problem. That is more than 10 times of the time it takes (2,250 ms) when each time period is 30 minutes long. In the same plot, we show the optimal expected total delay cost obtained from each problem configuration. Ideally all the problem setups would give us the same optimal cost. But there is no free lunch; we need to pay for the time the aggregation buys us. With a coarser scale of two-hour periods, the optimal cost the model finds is more than three-fold of the true optimal cost (assuming the quarter-hour period problem is the

true problem). A higher optimal cost is not surprising given that aggregation in effect imposes constraints on a minimization problem. These additional constraints rule out ground holding actions that are feasible with shorter time steps. For example, if a 15-minute time scale is used, then a flight can be held for 0, 15, 30, ... minutes, while if a 30 minute scale is used only ground holds of 0, 30, 60, ... minutes are possible. This also hints on the loss in solution quality when solving the problem using 15-minute units rather than 5-minute or 1-minute units. In short, we are facing a tradeoff between the feasible computation time and the quality of solution. We observe from Figure 4-15 that when the number of departure periods is reduced from 16 15-minute periods to 8 30-minute ones, the computation time dropped 90% while the optimal cost increased by just 27%. This suggests that for problems with unacceptable solution times, aggregation from 15 to 30 minute periods could drastically reduce runtimes with only a small decrease in the quality of the solution. In brief, from the use of time epoch aggregation, we found an opportunity to shorten the computation time significantly while trading off with the quality of the solution.

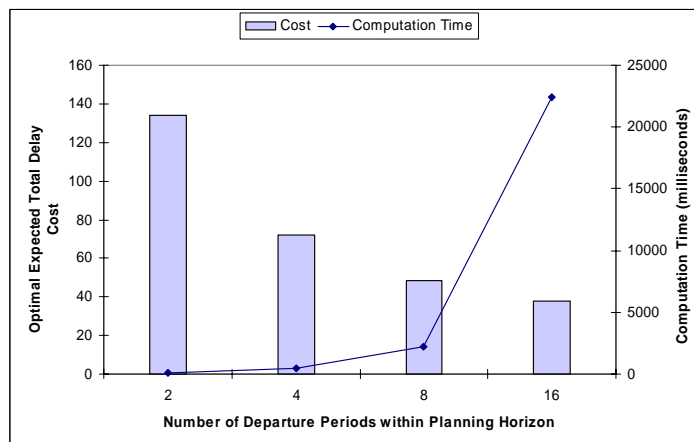


Figure 4-15 Computation time when dividing the planning horizon into different number of periods

4.3.2.2 Effect of Priority Ordering

The two priority ordering schemes—LGF and RBS—presented in Subsection 4.2.2 fold down the combinatorial action space (action of holding some number of flights from each flight duration type) into a much more manageable space where an action is defined by the number of flights to hold. From theoretical analysis, it is clear that the computational complexity should be reduced considerably with the use of priority ordering. But the extent to which the quality of solution would be compromised due to priority ordering is unclear. Here we numerically investigate the performance of the LGF and RBS schemes as compared with the full optimization algorithm.

We experiment with six test cases. The test cases all have eight flights to release but each test case has a different flight duration mix, as described in Table 4-11. The arrival capacity transition matrix (Table 4-12) is fixed throughout the planning horizon of six periods and the initial capacity level is two flights per period. All the experiments are carried out on the Linux server.

Table 4-11 Flight Schedule for the Test Cases

| Flight Index | Case 1 | | Case 2 | | Case 3 | | Case 4 | | Case 5 | | Case 6 | |
|--------------|------------------|----------------|------------------|----------------|------------------|----------------|------------------|----------------|------------------|----------------|------------------|----------------|
| | Departure Period | Arrival Period | Departure Period | Arrival Period | Departure Period | Arrival Period | Departure Period | Arrival Period | Departure Period | Arrival Period | Departure Period | Arrival Period |
| 1 | 0 | 4 | 0 | 2 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 2 | 0 | 4 | 0 | 2 | 0 | 1 | 0 | 2 | 0 | 3 | 0 | 2 |
| 3 | 0 | 4 | 0 | 4 | 0 | 2 | 0 | 2 | 0 | 3 | 0 | 3 |
| 4 | 0 | 4 | 0 | 4 | 0 | 4 | 0 | 4 | 0 | 4 | 0 | 4 |
| 5 | 1 | 5 | 1 | 3 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| 6 | 1 | 5 | 1 | 3 | 1 | 2 | 1 | 3 | 1 | 4 | 1 | 3 |
| 7 | 1 | 5 | 1 | 5 | 1 | 3 | 1 | 3 | 1 | 4 | 1 | 4 |
| 8 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 |

Table 4-12 Transition Matrix for Experiments in Subsection 4.3.2

| from \ to | 2 | 3 | 4 |
|-----------|-----|-----|-----|
| 2 | 0.4 | 0.4 | 0.2 |
| 3 | 0.2 | 0.6 | 0.2 |
| 4 | 0.2 | 0.4 | 0.4 |

The flight duration is the same for all the flights in Case 1. Under this condition, the *exact* computation does not have a combinatorial action space to search from for the best

action; the priority ordering scheme LGF does not take effect since all the flights have the same flight duration; and the scheme RBS does not have actual effect, either, for the same reason. We observe that (Figure 4-16) the *exact* computation takes about the same time as the computation with either of the priority ordering schemes. Not surprisingly, these three algorithms arrive at the same optimal solution for Case 1 (Figure 4-17). As the mix of flight durations gets more heterogeneous, it takes a lot longer for the *exact* algorithm to complete computation while the algorithms with priority ordering take about the same amount of time as they do with Case 1. The algorithms with priority ordering takes around 17% and 3% of the time the exact algorithm takes to solve the problems with two types of flight durations (Case 2) and four types of flight durations (Case 6), respectively. Between LGF and RBS, we do not observe much difference of the time they take to solve the same problem.

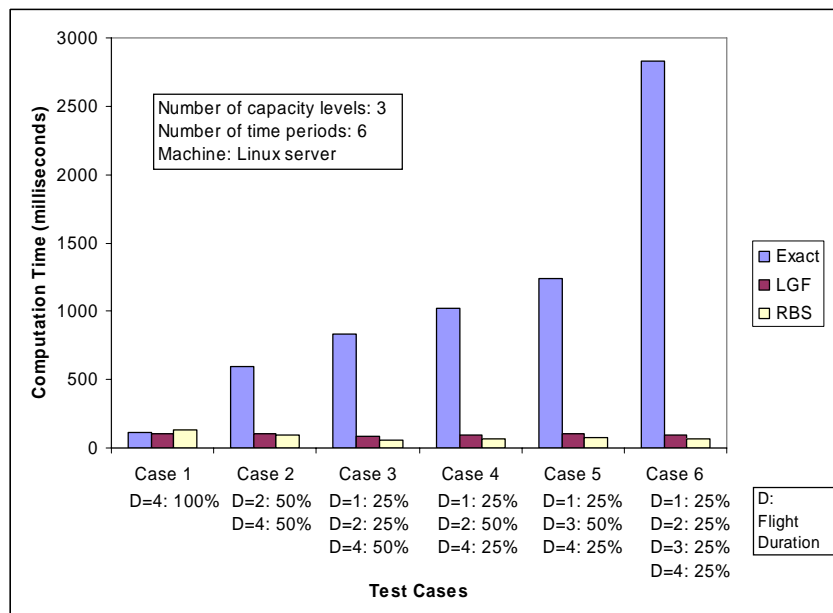


Figure 4-16 Computation time with and without priority orderings of flights

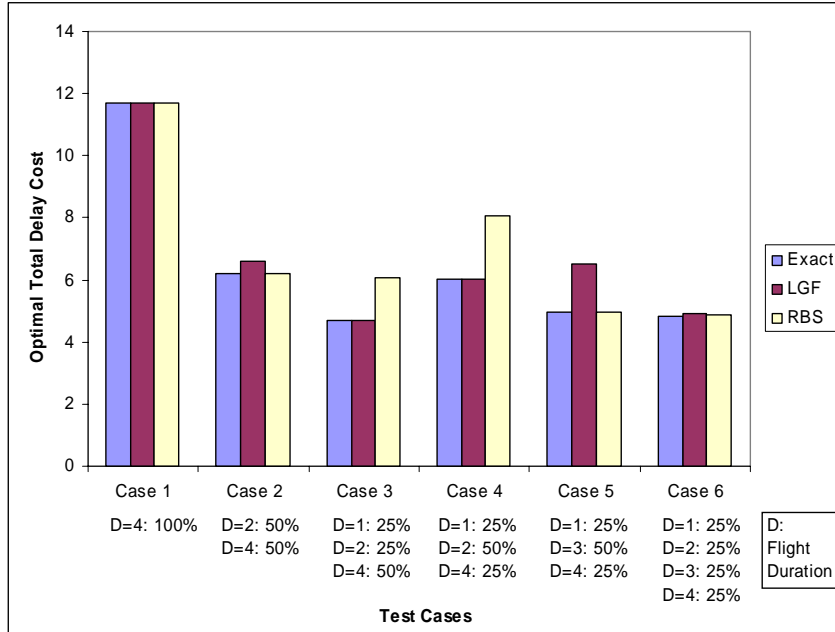


Figure 4-17 Optimal total delay cost with and without priority ordering of flights

On the other hand, the optimal total delay costs computed by the algorithms with priority ordering are not much higher than that computed by the exact algorithm. In fact, in many cases one of the algorithms with priority ordering arrives at the same optimal cost as the exact algorithm. Among the six test cases, the biggest deviations from the exact optimal cost due to LGF and RBS are 29% (RBS, Case 3), 33% (RBS, Case 4), and 32% (LGF, Case 5). There is no clear pattern concerning the relative sub-optimality of LGF and RBS. Intriguingly, in every test case at least one of the two prioritization schemes produces a solution that essentially matches the full optimization in terms of optimality.

In brief, we experimentally confirmed the effectiveness of priority ordering in reducing computation time as the heterogeneity of the flight durations increases. At the same time, we found that the loss in the quality of solution was never very large, and often negligible, in our six test cases.

4.3.2.3 Effect of Heuristics

In Subsection 4.2.3, we proposed two heuristics, H1 and H2, to further reduce the computation required to implement priority ordering schemes. These heuristics are devised for a search of optimum in single dimension—a condition when priority ordering is put into effect. In this subsection, we consider the effect of these heuristics numerically when each of the priority ordering scheme is implemented.

We continue the use of the six cases introduced in 4.3.2.1. Figure 4-18 shows the computation time results from using heuristics H1 and H2 when each of the priority ordering schemes—LGF and RBS—is in effect. Consistently, heuristic H2 takes less time than heuristic H1, and heuristic H1 takes less time than the original algorithm. Heuristic H1 takes 14-42% and 18-41% of the time it takes for the original algorithm with LGF and RBS to solve a problem, respectively. With a more dramatic effect, heuristic H2 takes 1-25% and 6-25% of the time it takes for the original algorithm with LGF and RBS to solve a problem, respectively. It is also notable that the effect of the heuristics on the computation time is not as significant for Case 1. Case 1 is special in that the cost-to-go function is actually convex since all the flights have the same flight duration (Corollary 4.9). As such, the cost-to-go function in Case 1 does not have the opportunities for upsurge as the functions in other cases with a greater variety of flight durations have. Hence, the heuristic algorithms proceed till they reach the bottom of the convex function for one particular flight duration and it unsurprisingly takes longer. By the same token, Case 1 is the only one where the heuristics guarantee the same solution as the complete search.

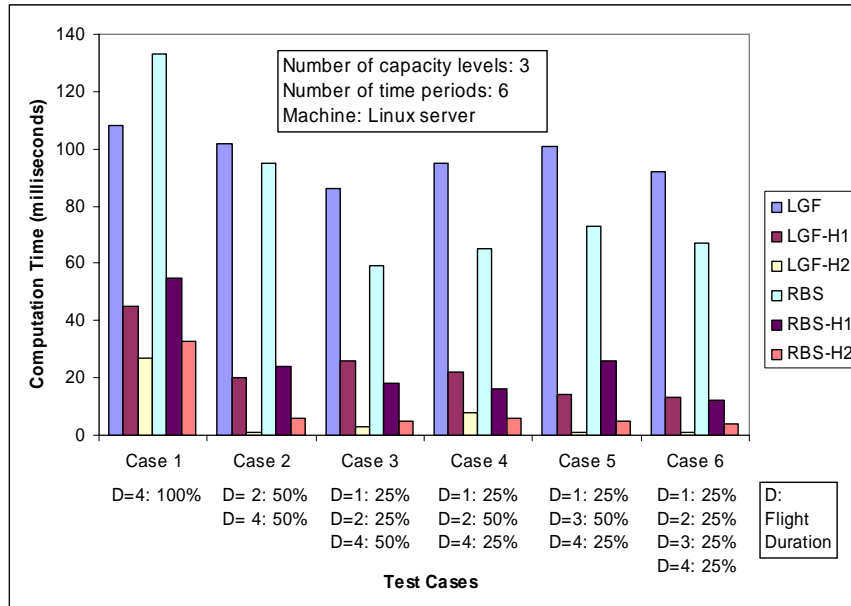


Figure 4-18 Computation time with and without the heuristic

The effect of the heuristics on the best total delay cost is exhibited in Figure 4-19, where the best total delay cost achieved by each algorithm is presented as a percentage of the optimal cost computed by the exact algorithm. As expected, all the algorithms arrive at the same solution for Case 1. For other cases, the best costs attained by the heuristics exceed the optimal cost to varying degrees. Generally, however, the difference is slight. Moreover, the larger differences (in Cases 3, 4, and 5) are mostly caused by the adoption of the priority ordering scheme rather than the heuristics per se. The only appreciable penalty from using H1 is in RBS Case 4, in which the prioritization causes a 33% cost increase which H1 takes up to 40%. H2 does slightly worse in this case, yielding a cost 44% over the true optimum. H2 also performs relatively poorly in RBS Case 3, increasing the cost 49% over the original solution, as compared to the 29% increase with priority ordering without heuristic. On the other hand, across these six cases, the heuristics do not lead to any cost overage for LGF prioritization.

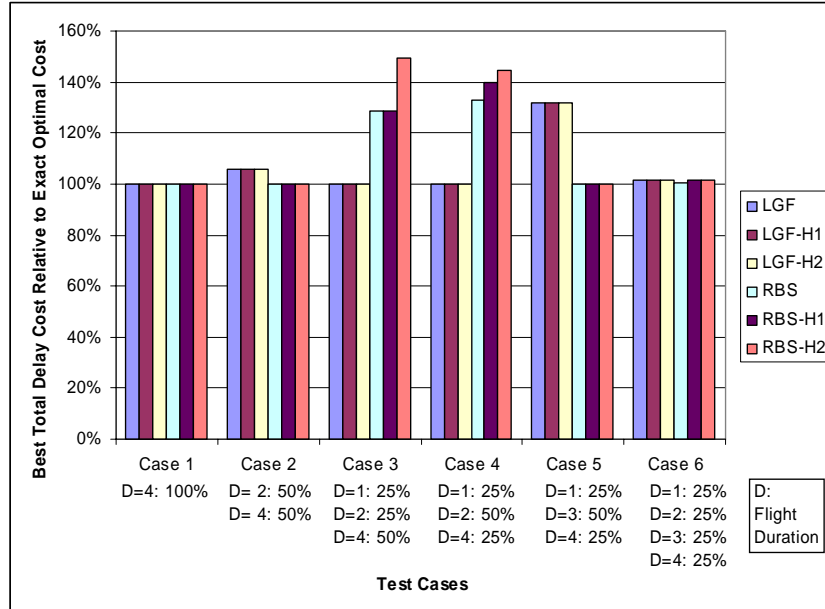


Figure 4-19 Impact of the heuristics on the best total delay cost

In sum, the heuristics reduce the computation time by 40% to 99% in our numerical experiments. Between heuristics H1 and H2, H2 reduces the computation time more dramatically but is somewhat more likely to incur a cost penalty. In these test cases, the heuristics lead to the same optimal solution when the LGF priority rule is used, but from time to time lead to actions that are more costly when the RBS priority rule is used. Another pattern suggested, albeit far from conclusively, from these results, is that the cost penalties from heuristic use are highest in the cases where the penalties from using priority ordering are also highest. While these initial results are promising, more extensive testing and analysis is required before the heuristics are used in practice.

4.3.3 Application

The test cases in the previous subsection are small scale problems formulated so that they can be solved fairly quickly by all models, including the exact model. In this subsection, we investigate the computational feasibility of solving a real-world problem

using our model and algorithms. For this investigation, we first identify a real-world problem as our target problem and then use our algorithm to determine the optimal policy for the SAGHP.

Our case is based on San Francisco International Airport (SFO) on the day of March 2, 2006. In general, SFO has a relatively high proportion of arrivals subject to ground holds, and March 2006 was a month in which ground holding at SFO was particularly common, according to the details given below. The estimated departure clearance time (EDCT), also known as the controlled time of departure (CTD), is the assigned departure delay for flights subject to an airport ground delay program or an airspace flow program (AFP). According to the FAA ASPM database, 14% of the arrivals at SFO were arrivals with EDCT from January to May in 2006. This percentage of arrivals with EDCT puts SFO at the fifth place among the airports in the U.S. during this period.¹¹ Although GDP is not the only cause of EDCT assignment, we take high percentage of arrivals with EDCT as a reflection of high incidence of GDPs for the airport. Out of these five months, March had the highest frequency of arrivals with EDCT (Table 4-13). Thus, we choose to pick our target problem from March. Among the 31 days in March, 22 days have at least one arrival with EDCT. Table 4-14 lists these days, ordered by the percentage of arrivals with EDCT. There are days with fairly low EDCT incidences and days with incidence rate as high as 72%. From this list, we take the situation on the day with the median percentage of arrivals with EDCT, March 2nd, as our target problem, where around 27% of arrivals were subject to EDCT.

¹¹ The airports that have higher percentage of arrivals with EDCT than SFO in the same time frame are ATL, ORD, LGA and EWR, in ascending order.

Table 4-13 Arrival Data at SFO for year 2006 from FAA ASPM Database

| Month | Number of Arrivals | Arrivals with EDCT | % of Arrivals with EDCT |
|----------|--------------------|--------------------|-------------------------|
| January | 13484 | 1419 | 10.52% |
| February | 11513 | 1556 | 13.52% |
| March | 13102 | 2864 | 21.86% |
| April | 12889 | 2520 | 19.55% |
| May | 12872 | 824 | 6.40% |

Table 4-14 Arrival with EDCT in March, 2006 at SFO

| Date | Number of Arrivals | Number of Arrival with EDCT | % of Arrivals with EDCT |
|-----------|--------------------|-----------------------------|-------------------------|
| 3/7/2006 | 427 | 8 | 1.87 |
| 3/6/2006 | 426 | 9 | 2.11 |
| 3/14/2006 | 423 | 9 | 2.13 |
| 3/9/2006 | 437 | 12 | 2.75 |
| 3/25/2006 | 395 | 24 | 6.08 |
| 3/22/2006 | 439 | 60 | 13.67 |
| 3/13/2006 | 429 | 60 | 13.99 |
| 3/15/2006 | 429 | 68 | 15.85 |
| 3/17/2006 | 442 | 85 | 19.23 |
| 3/21/2006 | 427 | 101 | 23.65 |
| 3/2/2006 | 427 | 115 | 26.93 |
| 3/27/2006 | 425 | 123 | 28.94 |
| 3/12/2006 | 413 | 120 | 29.06 |
| 3/20/2006 | 430 | 149 | 34.65 |
| 3/29/2006 | 430 | 158 | 36.74 |
| 3/10/2006 | 432 | 182 | 42.13 |
| 3/30/2006 | 425 | 183 | 43.06 |
| 3/5/2006 | 393 | 248 | 63.1 |
| 3/28/2006 | 408 | 264 | 64.71 |
| 3/31/2006 | 440 | 285 | 64.77 |
| 3/16/2006 | 414 | 293 | 70.77 |
| 3/24/2006 | 425 | 308 | 72.47 |

Our aim is to assess the computation time for a typical real-world problem instance. As such, instead of specifying the problem as an exact mapping to the situation, we simplify the process of converting the actual situation into the problem specification. From the airport data on March 2nd, we observe nearly continuous capacity shortage (arrival demand greater than AAR) from 8:45 a.m. to 1:15 p.m. In addition, most of the flights scheduled to arrive in this period have scheduled departure time from 7 a.m. to 12 p.m. Hence, for this target problem, we include flights with scheduled departure time from 7 a.m. to 12 p.m. inbound to SFO. The planning horizon is set for seven hours in total (7 a.m. to 2 p.m.). The airport data indicate that the arrival capacity at SFO on that day was around six to seven aircraft per quarter hour across the entire planning horizon.

According to the flight schedule, 28 of the flights arriving in between 9 a.m. to 2 p.m. originate from foreign airports. To account for the exemption of the international flights in the SAGHP, we simplify by reducing the arrival capacity level and assuming a constant capacity transition matrix throughout the planning horizon (Table 4-15) with initial arrival capacity level at 5. Though the matrix in is a simplification, a 3-by-3 capacity transition matrix nevertheless reflects the real airport situation. The weather condition at an airport is categorized as VFR, MVFR (marginal VFR), or IFR based on the visibility and ceiling conditions. For the same runway configuration, the capacity level is a function of the weather condition. Therefore, it is reasonable to consider three possible capacity levels due to the three primary weather conditions. Excluding the international flights, there are 116 flights departing between 7 a.m. and 12 p.m. We consider these 116 flights in the target problem (Table 4-16).

Table 4-15 Transition Matrix for the Target Problem

| from \ to | 5 | 6 | 7 |
|-----------|-----|-----|-----|
| 5 | 0.4 | 0.5 | 0.1 |
| 6 | 0.3 | 0.4 | 0.3 |
| 7 | 0.2 | 0.3 | 0.5 |

Table 4-16 Schedule of Flights Included in the Target Problem

| Departure Hour | Departure Quarter | Number of Flights | Flight Duration |
|----------------|-------------------|-------------------|---------------------------|
| 7 | 1 | 6 | 3 5 8 15 17 17 |
| 7 | 2 | 6 | 5 6 7 7 11 5 |
| 7 | 3 | 8 | 4 6 6 8 9 11 16 21 |
| 7 | 4 | 8 | 4 5 5 5 5 8 8 18 |
| 8 | 1 | 6 | 3 5 17 17 17 17 |
| 8 | 2 | 6 | 5 5 5 6 9 26 |
| 8 | 3 | 2 | 12 25 |
| 8 | 4 | 3 | 7 7 25 |
| 9 | 1 | 6 | 2 3 4 5 5 6 |
| 9 | 2 | 10 | 4 6 6 6 7 7 8 11 20 23 |
| 9 | 3 | 5 | 2 3 16 17 17 |
| 9 | 4 | 3 | 5 6 6 |
| 10 | 1 | 12 | 4 4 4 5 6 7 7 7 8 8 18 20 |
| 10 | 2 | 2 | 6 6 |
| 10 | 3 | 5 | 5 6 6 10 21 |
| 10 | 4 | 2 | 2 4 |
| 11 | 1 | 8 | 5 5 5 6 6 7 18 18 |
| 11 | 2 | 4 | 8 11 25 25 |
| 11 | 3 | 8 | 4 4 5 6 6 7 13 24 |
| 11 | 4 | 6 | 3 5 5 6 8 8 |

We solve the target problem with heuristic H2 on the Linux server and present the computational results in Table 4-17. The algorithm reached a slightly lower optimal cost but had a much longer computation time with RBS priority ordering. But even with RBS, the computation is completed in 20 seconds—well within the acceptable range for real-world application.

Table 4-17 Computational Results for the Target Problem

| | Priority Ordering | |
|--|-------------------|--------|
| | LGF | RBS |
| Number of Decision Nodes | 156301 | 415228 |
| Optimal Expected Total Delay Cost | 26.43 | 25.87 |
| Computation Time (milliseconds) | 2324 | 19741 |

On the other hand, it took more than two hours for the algorithm with heuristic H1 to solve this target problem using LGF.¹² This is not surprising since, as discussed above, the modest savings from using heuristic H1 in a given time period grow exponentially with the number of time periods. For a problem with realistic scale, this difference is sufficient to make H2 the only feasible heuristic for our computing resources. Since the computation time is acceptable for the target problem at the quarter-hour level, we did not need to reconfigure the problem using time epoch aggregation. In general, for larger problems that may take an unacceptably long time to solve, we can resort to time epoch aggregation and trade the computational feasibility with the precision of the solution.

Besides, we observed demanding memory usage during computation for larger problems. The computation time is often prolonged for larger problems due to intensive memory swapping and management operations. That is, the algorithmic complexity is not the sole reason for the computation time. Hence, the computational performance can be

¹² We timed-out this algorithm after two hours.

improved on this front with a mightier computing environment and/or by tailored coding that suits the specific memory management needs. We leave these for future exploration since they are not the focus of this research.

4.4 Summary

In this chapter, we have investigated the use of sequential decision models to optimize ground holding decisions in the context of the single airport ground holding problem. We formulated the model using a dynamic program and solved it with a value iteration algorithm. As far as we know, this is the first scenario-free model that can provide a dynamic optimum for the SAGHP.

From complexity analysis, we recognized the need to explore computational strategies so as to manage the curse of dimensionality of dynamic programming. Firstly, we observed numerous overlapping subproblems in the dynamic program, and chose to memoize the top-down recursive algorithm instead of using the traditional bottom-up approach. Memoization cuts down computation time dramatically and provides the same optimal solution.

Next, we explored several strategies that can reduce computation time but may result in suboptimal solutions. From the algorithmic complexity, we observed that priority ordering among flights could reduce the computational load significantly. We proposed two priority ordering schemes, and found that they could help reduce the computation time greatly while providing solutions that were nearly optimal in our test cases. Additionally, the structural property of the cost-to-go function inspired us to devise heuristics for limited search. We found the heuristics helpful in further reducing

computation time while only slightly reducing solution quality. Schematically, our exploration of computational strategies for this computationally intense problem can be summarized by Figure 4-20.

As a proof of concept, we demonstrated the computational feasibility of our model and algorithm for solving a typical real-world SAGHP. We found that the problem can be solved within an acceptable time for real-world application. In this real-world example, the model, in fact, planned for trillions of capacity scenarios. This quantity is clearly out of the capacity of the integer programming solvers today, and demonstrated an advantage of the scenario-free modeling approach. However, it is not clear by how much the computational strategies compromise the quality of solution. We explore this topic by comparing the performance of the scenario-based model and the scenario-free model in a real-world setting in the next chapter.

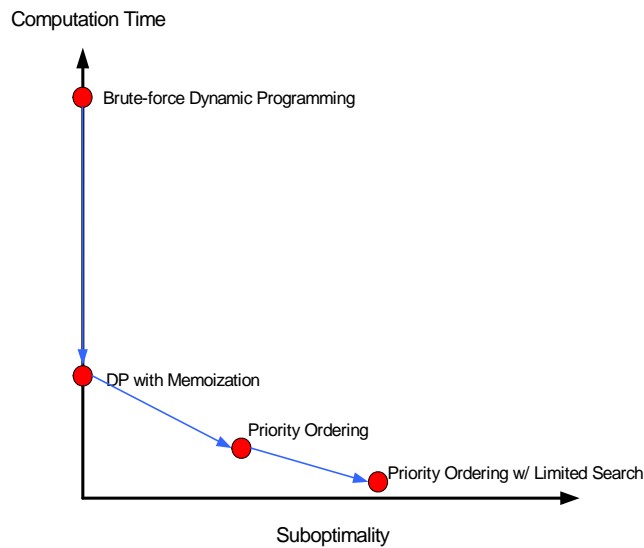


Figure 4-20 Effect of computational strategies

CHAPTER 5 COMPARISON OF SCENARIO-BASED AND SCENARIO-FREE MODELS FOR SAGHP

A problem of optimal decision making under uncertainty can be formulated using various mathematical models, including stochastic programming, Markov decision process, statistical decision theory, and stochastic control (Birge and Louveaux, 1997). The suitability of the modeling approach depends on the match between the emphases of the model and the characteristics of the problem. In the previous chapters, we have discussed the use of scenario-based stochastic programming and scenario-free sequential decision-making dynamic programming models for the single airport ground holding problem. Though the modeling approaches are different, both models can, in principle, provide a dynamic optimum for the SAGHP. In this chapter, we compare the performance of these models. We start by discussing the relationship between the scenario-based Mukherjee-Hansen model and the scenario-free sequential decision model in Section 5.1. Then, we investigate the performance of the models in a real-world setting. In Section 5.2, we test the performance of the models using simulated capacity data. Next, we compare the performance of the models in a real-world setting with real data in Section 5.3. Finally, Section 5.4 offers conclusions on the model comparison.

5.1 Relationship between the scenario-based and scenario-free models

In Chapter 4, we have presented a sequential decision model for SAGHP as an alternative to the stochastic programs. The sequential decision model and the Mukherjee-Hansen stochastic program describe the same dynamics of the same system; the

difference between these two models lies in their decomposition schemes. In effect, the sequential decision model determines the optimal action at each time period given the system state. More computations are involved when the planning horizon is longer. On the other hand, the stochastic program determines the optimal actions for each scenario of capacity state evolution and reports the expected cost over the probability distribution of the scenarios. Additionally, the Mukherjee-Hansen multistage stochastic program models recourse through nonanticipativity constraints (Constraint set 2-6).¹³ The strategy of the stochastic programs is to tackle an intricate planning-under-uncertainty problem by scenario. Therefore, the major difference in modeling between the sequential decision model and the stochastic program is “decomposition by time” versus “decomposition by scenario”.

It is obvious that not every stochastic program has an equivalent sequential decision model counterpart. The nonanticipativity constraints in the Mukherjee-Hansen model enforces the optimal decision to behave in the same way as that from a sequential decision making model. Thus, these two stochastic optimization models give the same optimal solution for the same problem. The interchangeability between these two models is established on the equivalence between the stochastic processes within these two optimization models. In the sequential decision model, the stochastic process of the capacity level evolution is modeled through a set of transition matrices. In the stochastic program, the same stochastic process is modeled using a probabilistic scenario tree. With a probabilistic scenario tree, the capacity state moves forward to the next state according

¹³ The nonanticipativity constraint of multiperiod stochastic programs imposes the requirement that solutions are based only on information known at the time of decisions.

to conditional probabilities. As a transition probability matrix is a way of stating conditional probabilities, the two stochastic processes can be equivalent. We arrive at the following conjecture.

Conjecture 5.1 *The multistage stochastic program with recourse and the sequential decision model are interchangeable when the probabilistic scenario tree in the stochastic program is equivalent to the Markov chain in the sequential decision model.*

Conjecture 5.1 has been verified empirically through many numerical experiments. We found that when the probabilistic scenario tree describes the same stochastic process as what is described by the Markov chain, the two models always give the same optimal solution for the same problem. That is, the dynamic optimum found by one model is the same as that found by the other when the probabilistic scenario tree is equivalent to the Markov chain. Hence, there is no optimality gap between them. However, the stochastic process described by a probabilistic scenario tree need not be memoryless. In fact, the probabilistic scenario tree can carry more information and characterize a history-dependent stochastic process. History-dependent stochastic process can also be characterized through a higher-order Markov chain, but it is not accommodated in the current sequential decision model.

While the two models are equivalent in concept, they differ in application. The stochastic program makes the problem manageable through approximating the capacity evolution by a limited set of scenarios, which in reality are aggregations of many distinct capacity profiles. The sequential decision model, on the other hand, makes the problem manageable through heuristic algorithms that approximate the optimization, while

allowing for a huge set of possible capacity evolutions. For larger problems, the sequential decision model can ultimately resort to the strategy of time epoch aggregation, which manages the problem by aggregation in time. In the end, the choice of model turns on the gains and losses in these aggregations and approximations.

5.2 Model Performance with Simulated Data

In this section and the next, we investigate the performance of the scenario-based and scenario-free models in a real-world setting. The scenario-free model assumes that the capacity evolution follows a set of transition probability matrices; therefore, we have to provide a set of such matrices before we can use the model. Ideally, the capacity transition probability matrices should be developed based on real airport data, and can be conditional on time-of-day, season, weather forecast, runway configuration, and other factors. One way to test the model without developing the matrices is experimenting in a simulated environment. In this section, we conduct a simulation experiment, assuming that the world is Markovian. This simulation environment also helps isolate the experiment from confounding factors such as how well the phenomenon is captured by the Markov process and the loss of information in matrices estimation. Then, we investigate the performance of models using matrices developed from real data in the next section.

5.2.1 The Simulation Experiment

In this experiment, we assume that the airport arrival capacity evolves based on a set of time-varying first-order Markov transition probability matrices (Figure 5-1). The matrices are constructed based on a hypothesized structure of four capacity profile

patterns and the actual arrival capacity levels at SFO. The transition matrices range from 3 x 3 to 7 x 7 in size and the capacity level ranges from 4 to 10 landings per quarter hour. Based on this set of transition matrices, we generate 1000 capacity profiles that span seven hours (28 periods) using Monte-Carlo simulation. We assume the initial capacity level is seven landings per quarter hour.

Period 1-7

| from \ to | 6 | 7 | 8 |
|-----------|-----|-----|-----|
| 6 | 0.4 | 0.3 | 0.3 |
| 7 | 0.3 | 0.4 | 0.3 |
| 8 | 0.3 | 0.3 | 0.4 |

Period 8

| from \ to | 5 | 6 | 7 | 8 | 9 |
|-----------|-----|-----|-----|-----|-----|
| 6 | 0.3 | 0.4 | 0.2 | 0.1 | 0 |
| 7 | 0.1 | 0.2 | 0.4 | 0.2 | 0.1 |
| 8 | 0 | 0.1 | 0.2 | 0.4 | 0.3 |

Period 9-13

| from \ to | 5 | 6 | 7 | 8 | 9 |
|-----------|-----|-----|-----|-----|-----|
| 5 | 0.4 | 0.3 | 0.3 | 0 | 0 |
| 6 | 0.3 | 0.4 | 0.3 | 0 | 0 |
| 7 | 0 | 0.3 | 0.4 | 0.3 | 0 |
| 8 | 0 | 0 | 0.3 | 0.4 | 0.3 |
| 9 | 0 | 0 | 0.3 | 0.3 | 0.4 |

Period 14

| from \ to | 4 | 5 | 6 | 7 | 8 | 9 |
|-----------|-----|-----|-----|-----|-----|-----|
| 5 | 0.3 | 0.3 | 0.2 | 0.1 | 0.1 | 0 |
| 6 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0 |
| 7 | 0 | 0 | 0 | 0.5 | 0.5 | 0 |
| 8 | 0 | 0 | 0 | 0.3 | 0.4 | 0.3 |
| 9 | 0 | 0 | 0 | 0.3 | 0.3 | 0.4 |

Period 15-21

| from \ to | 4 | 5 | 6 | 7 | 8 | 9 |
|-----------|-----|-----|-----|-----|-----|-----|
| 4 | 0.4 | 0.3 | 0.3 | 0 | 0 | 0 |
| 5 | 0.3 | 0.4 | 0.3 | 0 | 0 | 0 |
| 6 | 0.1 | 0.2 | 0.4 | 0.2 | 0.1 | 0 |
| 7 | 0 | 0 | 0.2 | 0.4 | 0.3 | 0.1 |
| 8 | 0 | 0 | 0.1 | 0.2 | 0.4 | 0.3 |
| 9 | 0 | 0 | 0 | 0.3 | 0.3 | 0.4 |

Period 22

| from \ to | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-----------|-----|-----|-----|-----|-----|-----|-----|
| 4 | 0.4 | 0.3 | 0.3 | 0 | 0 | 0 | 0 |
| 5 | 0.3 | 0.4 | 0.3 | 0 | 0 | 0 | 0 |
| 6 | 0.1 | 0.2 | 0.4 | 0.2 | 0.1 | 0 | 0 |
| 7 | 0 | 0 | 0.2 | 0.4 | 0.3 | 0.1 | 0 |
| 8 | 0 | 0 | 0.1 | 0.2 | 0.4 | 0.3 | 0 |
| 9 | 0 | 0 | 0 | 0.1 | 0.2 | 0.4 | 0.3 |

Period 23-28

| from \ to | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-----------|-----|-----|-----|-----|-----|-----|-----|
| 4 | 0.4 | 0.3 | 0.3 | 0 | 0 | 0 | 0 |
| 5 | 0.3 | 0.4 | 0.3 | 0 | 0 | 0 | 0 |
| 6 | 0.1 | 0.2 | 0.4 | 0.2 | 0.1 | 0 | 0 |
| 7 | 0 | 0 | 0.2 | 0.4 | 0.3 | 0.1 | 0 |
| 8 | 0 | 0 | 0.1 | 0.2 | 0.4 | 0.2 | 0.1 |
| 9 | 0 | 0 | 0 | 0.1 | 0.2 | 0.4 | 0.3 |
| 10 | 0 | 0 | 0 | 0 | 0.3 | 0.3 | 0.4 |

Figure 5-1 Transition matrices for the simulation

The flight schedule considered in this experiment is adapted from the schedule on March 2, 2006, at SFO, from 7 a.m. to 2 p.m. The flights included are those with arrival time no later than 12 p.m. In total, 100 flights are included for this experiment.

Following the methodology developed in Chapter 3, four capacity scenarios are identified based on the 1000 capacity profiles (Figure 5-2). The algorithm in Subsection 3.2.2 gives us a scenario tree with three branching points at period 7, 9, and 18, respectively. Using the branch identification algorithm developed in Subsection 3.4.1, a scenario is identified for each of the capacity profile as the best match. An interesting feature of the scenarios is that they “cross” (for example, scenarios 1 and 2 in time period 22). AAR time series that cross each other is possible for the given transition matrices. When the clustering algorithm organizes data series with crossing trends into groups, we can see patterns as shown in Figure 5-2. Thus, the crossing scenarios found in Chapter 3 do no, per se, invalidate the assumption in Chapter 4 that capacity evolution is Markovian.

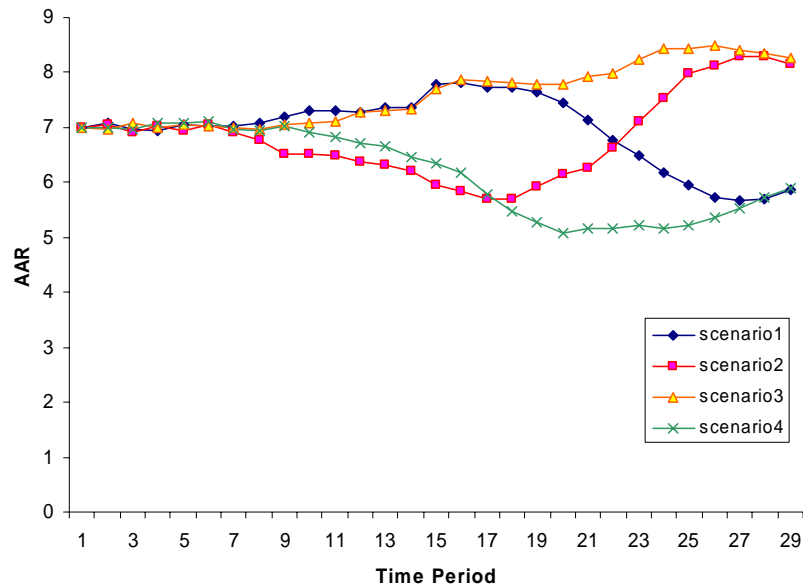


Figure 5-2 Capacity scenarios derived from the generated capacity data

The optimal ground-holding policy is determined using the Mukherjee-Hansen stochastic program, with a lag of five periods in its scenario tree. Simulating the actual application of the ground holding strategy against the 1000 generated “actual” capacity profiles, we obtain the total incurred delay cost as the sum of the ground delay cost planned and the airborne delay cost incurred.

The scenario-free sequential decision model is set up with the transition matrices in Figure 5-1. The optimal ground-holding policy is computed using the algorithm that assumes convexity to the left of the optimum (heuristic H2), for both priority ordering schemes LGF and RBS. The total incurred delay cost for each of the 1000 “actual” capacity profile is computed through navigating and recording the ground-holding policy and airborne waiting for each state-action trajectory in the SDM tree.

5.2.2 Results and Discussion

The performance of the models is summarized in Figure 5-3. The scenario-based model has a lower expected delay cost (5.43) than the scenario-free model (12.96 for LGF; 13.68 for RBS). However, the scenario-based model gives a higher average incurred delay cost (14.18) than the scenario-free model (13.22 for LGF; 13.93 for RBS). Between the scenario-free algorithms we find that the algorithm with the LGF priority ordering performs slightly better than the algorithm with the RBS ordering both in the average cost and the spread. Though the algorithm with LGF leads to higher average incurred delay time (5.5 quarter hours for LGF; 5.3 quarter hours for RBS), ground delay accounts for a higher percentage of the delay time (30% for LGF; 19% for RBS, Figure 5-4). Since the scenario-free algorithm with LGF performs better than the one with RBS, the following discussion focuses on the LGF version of the scenario-free model.

If the expected cost calculated by the models were to be used as an estimate of the delay cost, we find that the scenario-based model seriously underestimates the cost. On the contrary, the expected cost from the scenario-free model is very close to the average cost incurred. As discussed in Chapter 3, the gap between the expected cost and the average incurred cost for the scenario-based model is largely due to capacity dispersion around the nominal scenario. Clearly, capacity dispersion is not an issue for the scenario-free model. In the simulated environment where the transition matrices used in the model tracks the stochastic process exactly, the expected cost from the scenario-free model closely approximates the actual average incurred cost.

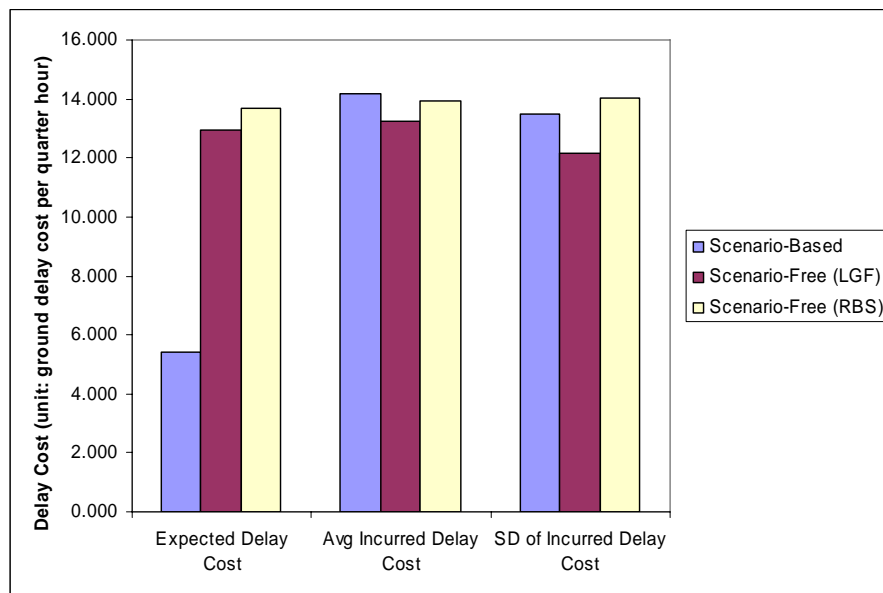


Figure 5-3 Performance of models in simulation experiment

The scenario-free model gives an average incurred delay cost that is 7% lower than the scenario-based model. The cost reduction is significant, but less than expected. The modest improvement can be attributed to the use of heuristics which do not guarantee finding the optimum. Moreover, for the capacity profiles that closely match the scenarios, the optimal solution from the scenario-based model is, since it is exact, at least as good as

that from the scenario-free model. This suggests the need for improved heuristics and better computational strategies in order to advance the performance of the scenario-free model.

Another basis for comparison is the average additional cost when one model leads to higher cost than the other. For the cases where scenario-based model leads to higher cost than the scenario-free model, the average of the additional cost is 5.43 (in units of ground delay per quarter hour). For the cases where scenario-free model leads to higher cost than the scenario-based model, the average of the additional cost is 3.16. On average, it is more costly when the scenario-based model gives a worse solution than the scenario-free model. Out of the 1000 cases, the scenario-free model leads to lower incurred cost in 480 cases, the scenario-based model leads to lower incurred cost in 387 cases, and they tie in 133 cases. In brief, both in the average additional cost and in the frequency of leading to lower cost, the scenario-free model performs better than the scenario-based model in our experiment. Moreover, there is an inherent advantage to have incurred and expected cost be similar. The expected cost is available right at the moment the optimization problem is solved, before any solution is implemented. The air traffic management service provider and the users can take the expected cost as a predictor of the condition of the system, if the operations will be managed following the solution from the model. As the expected cost tracks closely with the incurred cost, it delivers valuable information of what to expect in the future. This characteristic can help the acceptance of the model in the field since it demonstrates that the situation is under control. Also, this information can help the air traffic system users prepare their contingency plans better.

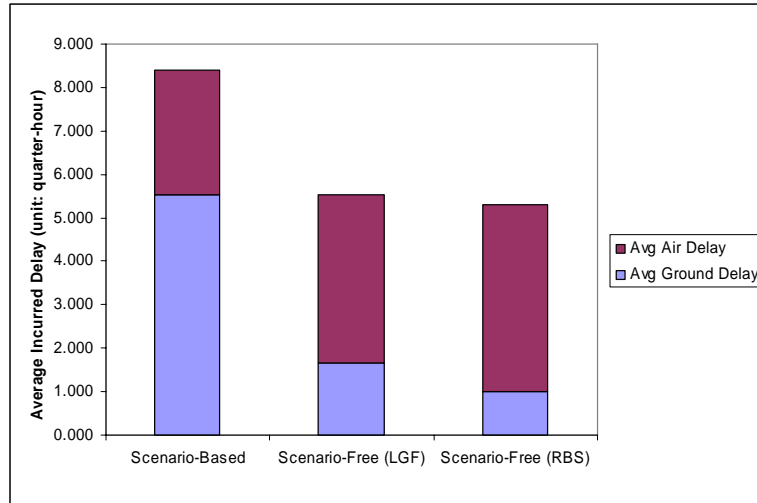


Figure 5-4 Comparison of delay distribution

5.3 Model Performance with Real Data

In this section, we investigate the performance of the models in a real-world setting with real data. We first describe the setup of the experiment in Subsection 5.3.1, and then discuss the results in Subsection 5.3.2.

5.3.1 Experiment Setup

In this experiment, we compare the performance of the scenario-based and scenario-free models for the SAGHP when they face the same problems. We use SFO as the backdrop for this experiment. The scenario tree developed in Chapter 3 for SFO based on 2003 data is used as input for the scenario-based model. In order to have an “apples-to-apples” comparison, we develop the capacity transition probability matrices for the scenario-free model based on the same set of 2003 airport data.

As mentioned in Section 5.2, the transition matrices for the scenario-free model may be conditional on many factors. To simplify the process and maintain comparability with

the scenario-based approach, we develop a set of time-varying transition matrices based simply on the observed evolution of capacity levels. For each quarter-hour time period, we count the number of transitions from each observed capacity level to capacity levels in the next time period and construct the time-specific transition probability matrices from these counts. We consider the SAGHP in the time horizon from 8 a.m. to 1 p.m. in this experiment; therefore, there are 20 transition matrices—one for each successive pair of time epochs. In developing these transition matrices, we eliminated certain very rare transitions, since these increase problem complexity while adding very little information. The transition matrices are included in Appendix A.1. The flight schedule considered in this experiment is adapted from the schedule on March 21, 2006, at SFO, from 8 a.m. to 1 p.m. The flights included are those with arrival time no later than 1 p.m. In total, 122 flights are included in this experiment (Appendix A.2).

The capacity profiles used in this experiment are the observed AAR sequences at SFO extracted from the FAA ASPM database, for all 1096 days in the years 2003-2005. The scenario-based model is the Mukherjee-Hansen model with a lag of five time periods in the scenario tree. Using the branch identification algorithm developed in Subsection 3.4.1, an optimal policy is chosen for each daily capacity profile. The scenario-free model employs the algorithm with LGF priority ordering and the heuristic H2. The optimal policy for each of the days is given by traversing the SDM tree. As an input to the scenario-free model, the initial capacity level should match the first capacity level of each sample capacity profile. Among these 1096 capacity profiles, 97% of them start at capacity level 7, 8, 12, or 15. To focus on the majority of the cases, we discard the profiles starting from other capacity levels, and work with the remaining 1061 of them.

We execute the algorithm with initial capacity at each of the four levels, and obtain four policy trees for the ground holding decisions.

As a result of eliminating very rare transitions, and because our transitions are based on just 2003 while our profiles come from three years, 18% of the 1061 days contain cases not covered by the transition matrices. For one or more time period during these days, the AAR takes a value that is not included in the transition matrix for that time period, and thus for which there is no policy. In such cases, we simply take the policy for the closest capacity level for which one is available. When there is a tie, for example, the AAR is 7 and the available policies are for levels 6 and 8, we choose conservatively by taking the policy for the lower level. Both models use the air-to-ground delay cost ratio of 3.

5.3.2 Results and Discussion

The result of the experiment is summarized in Figure 5-5. The scenario-based model has a lower expected delay cost (7.35) than the scenario-free model (13.68), but the scenario-based model gives a higher average incurred delay cost (15.75) than the scenario-free model (13.55). This result is in general consistent with what we found in the simulation experiment. This is a cost saving of 14%, as compared to the 7% improvement obtained from using the scenario-free model in the simulation experiment. While these experiments are not fully comparable, they suggest that the simulation results are not greatly affected by the assumption of a “Markovian” reality. A conceivable reason is that the time-dependent Markovian transition matrices we estimated are good approximation for the non-Markovian (or higher order Markovian) real world. We will discuss further on this topic in Subsection 5.3.3.

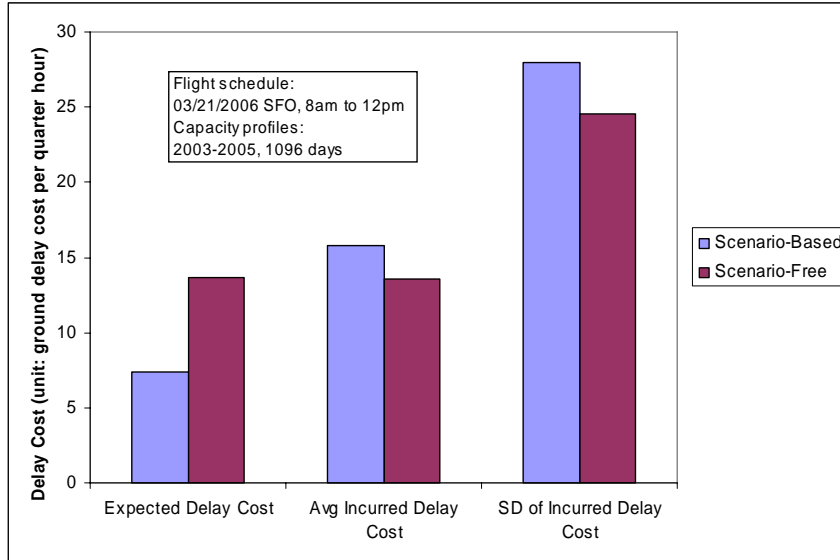


Figure 5-5 Performance of models in a real-world setting with real data

It is also notable that the standard deviation of the incurred cost from the scenario-based model is 12% higher than that from the scenario-free model. The wider spread of incurred cost can be of concern to the users of the models. It suggests that the scenario-based model may do better in the days with higher capacity and do worse in the low capacity days. As the air traffic managers are almost certainly risk-averse, the lower standard deviation of incurred cost is another advantage of the scenario-free approach. As discussed above, yet another advantage of this is that incurred cost matches expected cost.

Figure 5-6 shows the air-ground distribution of incurred delay from the models. On average, the scenario-based model assigns 34 minutes more ground delay than the scenario-free model while the incurred airborne delay from both models is about the same. Thus the cost savings from the scenario-free model come from eliminating unnecessary ground delays.

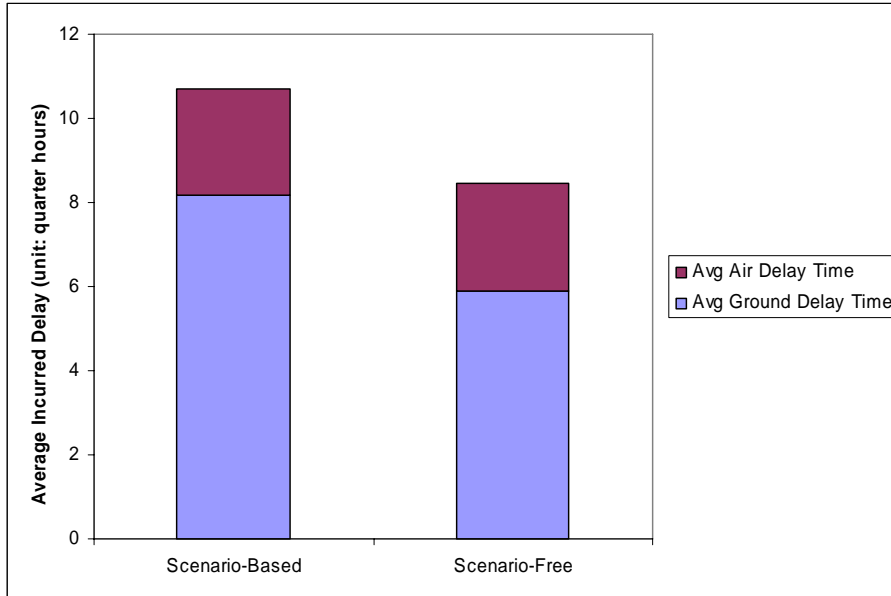


Figure 5-6 Delay distribution of scenario-based and scenario-free models

A detailed view of the delay distribution is provided in the bubble plots in Figure 5-7 and Figure 5-8. Each bubble corresponds to a combination of ground delay and airborne delay indicated by its coordinates. The size of the bubbles represents the number of days, out of the 1061 days in the experiment, in which this combination of ground and airborne delay was incurred. Following the optimal policy from the scenario-based model, many days had no airborne delay but some ground delay. We also observe many days with airborne delay in the range of 30 to 60 quarter hours. On the other hand, the distribution of delay incurred under the scenario-free model is more scattered but concentrated within the 20 quarter-hour radius from the origin, except for a cluster of days with ground delay of 33 quarter hours. This distribution of delay shows the ability of the scenario-free model to contain the delays in a smaller range for most days. The exceptions with high ground delay are all days that belong (based on the branch identification algorithm) to the rainy Cluster 3 in Figure 3-2.

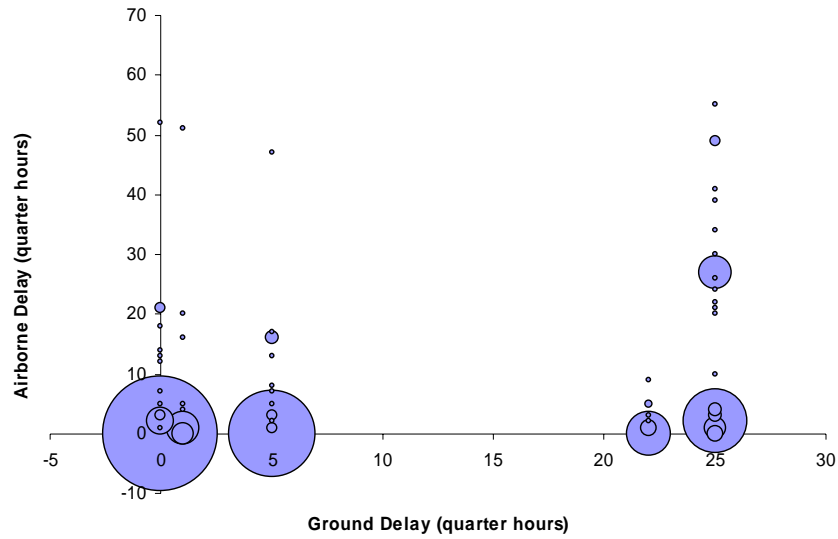


Figure 5-7 Delay distribution from using scenario-based model

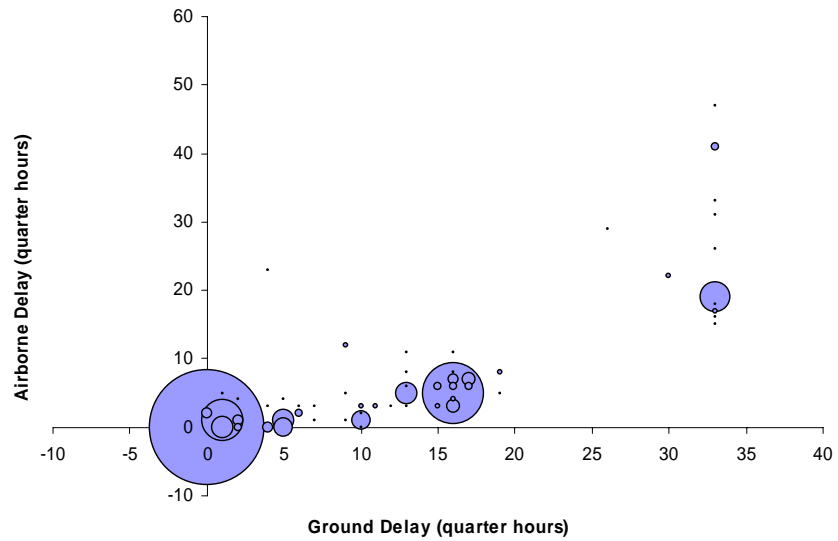


Figure 5-8 Delay distribution from using scenario-free model

In Figure 5-9, we plot the difference in the assigned ground delay and incurred airborne delay between the models for each day. The difference is the overage of delay due to the scenario-based model. The biggest bubble is located at the origin, indicating that it is most likely for the models to cause the same amount of ground and airborne delay (453 cases). The first and third quadrants correspond to days in which one model

dominates the other. We observe few bubbles in the first and no bubble in the third quadrant. Thus, on virtually every day the outcomes from the two models reflect different tradeoffs between ground delay and airborne delay. Bubbles in the second quadrant represent cases when the scenario-based model assigned less ground delay and thus incurred more airborne delay. Many bubbles in the second quadrant fall on a 45° line representing cases where the scenario-free model reduced cost by replacing airborne delay with an equal (but less costly) quantity of ground delay. The fourth quadrant contains days when the scenario-free model assigned less ground delay and thus incurred more airborne delay. Unlike in the second quadrant, the events are distributed near the x -axis and the origin. This pattern indicates that when the scenario-free model assigns less ground delay, it incurs a relatively small penalty in airborne delay. In sum, the scenario-free model is more effective in striking a balance in the tradeoffs between ground and airborne delay.

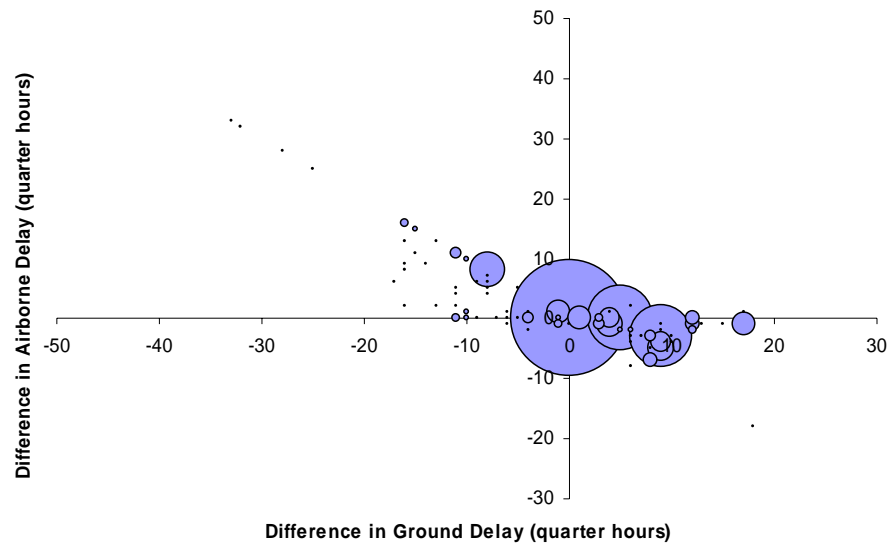


Figure 5-9 Overage of Delay from using the scenario-based model

The previous discussions were based on summary statistics of all 1061 days in the experiment. We now compare results for different subgroups of days. Since both the probabilistic scenario tree and the transition matrices were developed based on the data in 2003, it is possible that the models would perform better with the days in 2003 than the days in other years. Therefore, we group the days by year and compute the average cost for each group. Figure 5-10 shows that the relative performance of the models is fairly consistent across the years. The scenario-free model gives an average cost around 87 % of that given by the scenario-based model for 2003 and 2004, and 84% for 2005. This finding suggests that the capacity pattern is stable across the years and/or that the models are robust to the year-to-year differences that do exist. This is reassuring since, in practice, use of these models will require the projection of historical patterns into the future.

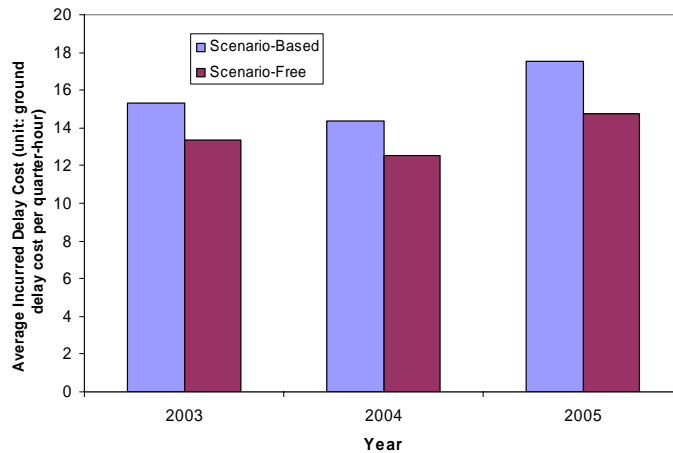


Figure 5-10 Model performance by year

It is possible that the optimal policy provided by a model works better for certain capacity patterns and worse for others. Since each capacity profile is associated with a capacity scenario in the scenario-based model, we classify the 1061 samples based on the associated capacity scenario and compare the performance of the models by the grouping.

The capacity scenarios considered here are the six sequences plotted in Figure 3-2. Scenario 1, 3, and 6 starts the day with low capacity. The capacity level goes up later in the day for Scenario 1 and 6, but stays low for Scenario 3. Scenario 2 and 4 are both high capacity patterns. Scenario 5 is a pattern with capacity in the middle range.

Figure 5-11 shows that the scenario-based model gives the difference between the models is the biggest for Scenario 6 (scenario of fog burn-off), where the cost from the scenario-free model is 48% less than that from the scenario-based model. For the high capacity scenarios (Scenario 2 and 4), the absolute cost difference between the models is small, but the scenario-free model on average gives a 20-26% lower cost than the scenario-based model. The most costly scenario (Scenario 3) is the case with low capacity throughout the planning horizon. Here the performance is determined more by lower capacity than by choice of model, with the scenario-free model reducing cost by less than 5%. This plot shows that the scenario-free model has a greater advantage when capacity evolution patterns are more dynamic.

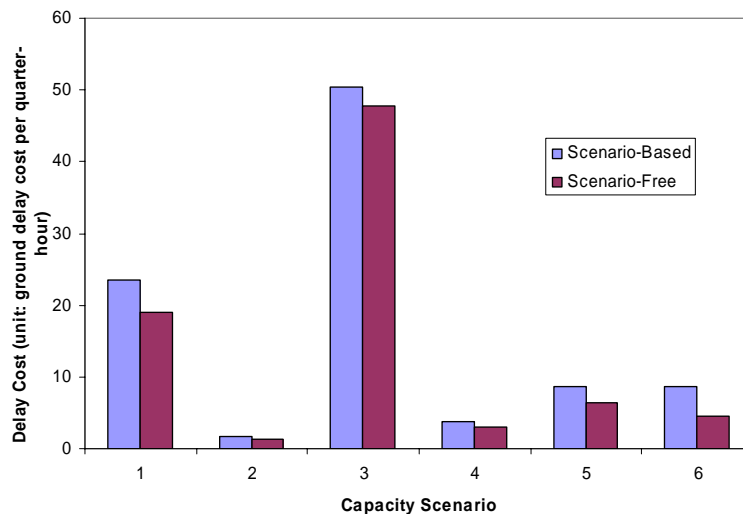


Figure 5-11 Model performance by matched capacity scenario

In the real world, ground holding decisions are made based on a combination of day-of-operation and historical information. To simulate making ground delay plans with day-of-operation information, we conduct the following experiment. Based on the SFO scenario tree, we classify the capacity profiles into three types: Type 1 starts the day with low capacity, Type 2 starts the day with high capacity, and Type 3 starts the day with medium capacity. We develop three sets of transition matrices for each type of capacity profiles using SFO AAR data in 2003. Type 1 matrices are estimated using data from the days associated with Scenario 1, 3 and 6 (Appendix A.3). Type 2 matrices are estimated using data from the days associated with Scenario 2 and 4 (Appendix A.4). Type 3 matrices are estimated using data from the days associated with Scenario 5 (Appendix A.5). The procedure for developing these matrices is the same as that used for developing the generic matrices. We assume that we know which of the three types of profiles we are facing before we apply the ground holding policy from the scenario-free model. In this experiment, we identify the closest scenario match for each capacity profile and use the “type” associated with the scenario for the capacity profile as the advanced information. Then we run the algorithm with the specific set of transition matrices associated with this type.

Arguably, this experiment puts the scenario free model on a more even footing with the scenario-based one. The three groups of scenarios are distinct from the beginning of the planning period, so the scenario-based solution is specific to whichever condition type is realized. Thus in this experiment both models have access to roughly the same amount of day-of-operation information. The one possible difference is that, in implementing the scenario-based solution, the planner could misidentify the type, leading

to costly branch hopping (see Chapter 3). Therefore, for comparison purposes, we compute the average incurred delay cost from the scenario-based model, with and without assured accurate scenario match. The case without assured match is the basis of the earlier results, and contains some cases where cost is very high due to branch misidentification. The assured accurate match case assumes that the condition type is correctly identified and that all subsequent branching is accurately interpreted. In terms of comparability to the type-specific scenario-free model, the former case assumes that the scenario-based model has somewhat less information, while the latter assumes it has slightly more information.

Figure 5-12 summarizes the results of this experiment. The average incurred cost for the scenario-based model without assured accurate match is 36.1, 1.2, and 7.6 (in the unit of ground delay cost per quarter hour) for Type 1, 2, and 3 conditions, respectively. Comparing the average incurred delay cost from using the scenario-based model (without assured accurate match) to that from using the scenario-free model (without advanced information), the cost savings are 11%, 78%, and 18%, for Type 1, 2, and 3 conditions, respectively. Note that the overall costs comparisons reported previously are driven mainly by the Type 1 condition due to its combination of high incidence and high cost.

Turning to the advanced information question, we find that this yields cost reductions for Type 2 and Type 3 capacity conditions for 22% and 5%, respectively. There is virtually no cost reduction resulting from advanced information for Type 1 condition. Because Type 1 conditions are the predominant source of overall cost the overall cost savings from having day-of-operation information is a marginal 0.5%. Assured accurate scenario match has a somewhat larger impact on scenario-based model

performance, although again the largest percentage impacts are for the condition types that generate the least cost. And even when the scenario-based model is given this information advantage, it still does worse overall, for two of the three condition types, than the scenario-free model without advanced information.

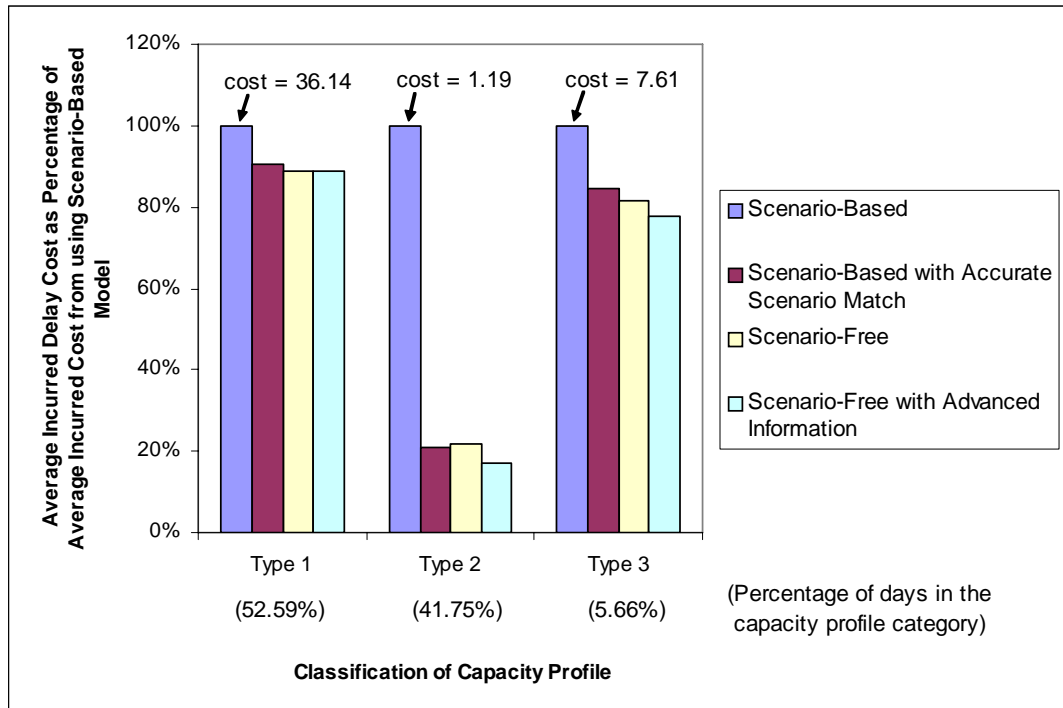


Figure 5-12 Model performance with advanced information

5.3.3 Realism of Markovian Model

We have developed and used time-varying transition matrices for the experiments in Subsection 5.3.2, since it is unlikely that a homogeneous first-order Markov process is sufficient to describe the capacity evolution. But it was not obvious that first-order Markov process, even a time-varying one, is a good representation of airport capacity evolution. While Subsection 5.3.2 has shown encouraging results for a scenario-free model built on such a representation, it is appropriate to examine the matter directly.

In this subsection, we show numerical support for the conjecture above using a simulation experiment. Based on the time-varying transition matrices we estimated (Appendix A.1), we generate 1000 capacity profiles using Monte-Carlo simulation. The capacity level for the first time period is generated based on the probability distribution of capacity levels at the first time period. Each capacity profile contains 20 capacity levels, for each quarter-hour from 8 a.m. to 1 p.m. The actual capacity data we considered for the comparison are the quarter-hour AAR data at SFO in 2003, from 8 a.m. to 1 p.m. For this comparison, we calculate the mean and standard deviation of the capacities at each time period, for both the capacities generated by the simulation and the capacities collected in the field. We also compute the autocorrelation with lags of one to four time periods for both the simulation-generated capacity data and observed AAR data.

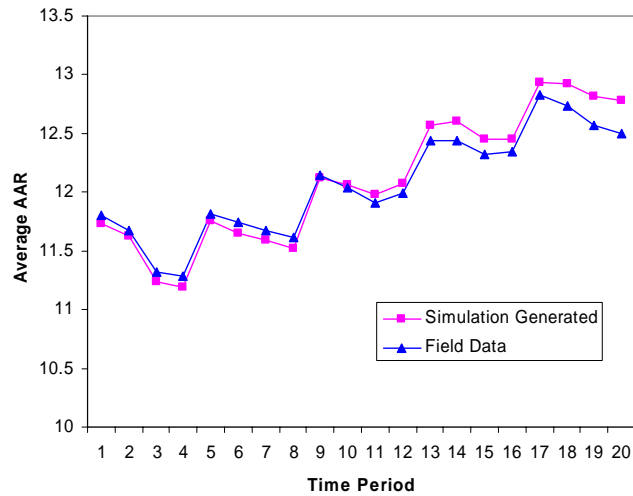


Figure 5-13 Average AAR from simulation and field data

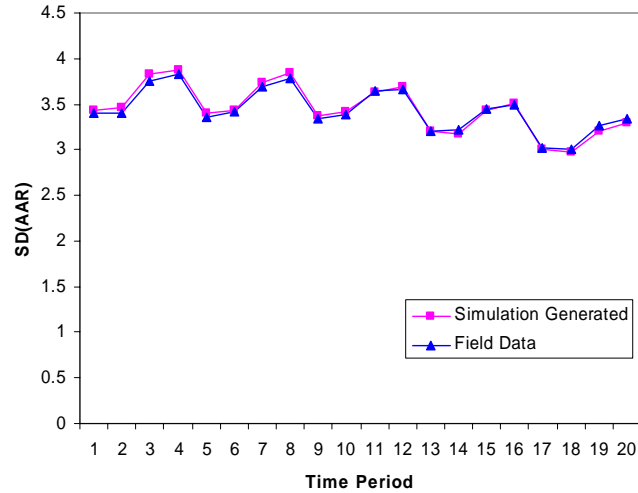


Figure 5-14 Standard deviation of AAR from simulation and field data

We find that the means and standard deviations of simulation-generated data nearly replicate the statistics from the field data (Figure 5-13, Figure 5-14). This finding demonstrates that the time-varying first-order Markovian transition matrices are capable of producing data very similar to the actual phenomenon. From Figure 5-15, we can see that the autocorrelation coefficients from the simulation-generated data track closely with those from the field data. Not surprisingly, the dissimilarity from the real data increases as the lag increases, but the difference in autocorrelations is small—around 0.05—even for the worst case. Even though it is still an open question to determine the best model to characterize the stochastic process of airport capacity evolution, the result of this simulation experiment indicates that time-dependent first-order Markov process can provide a good approximation of the phenomenon. Given the complexity that a higher-order Markov process would impose on the algorithmic development for the optimization model, this variant of first-order Markov process gives us an opportunity to develop

tractable algorithms while offering a good approximation of the stochasticity in the system.

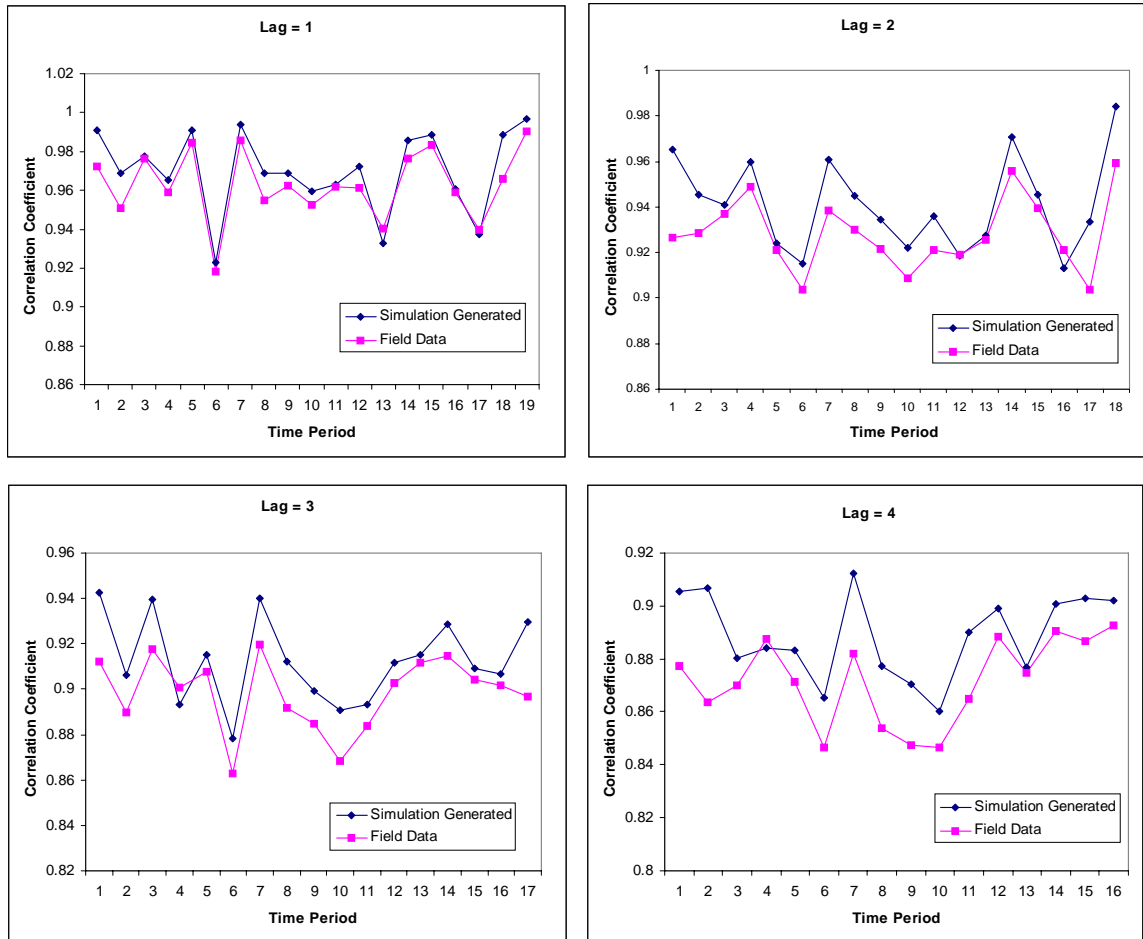


Figure 5-15 Autocorrelation for lags of one to four time periods

5.4 Summary

In this chapter, we compared the scenario-based and scenario-free models for the SAGHP on their mathematical relationship and performance in a real-world setting. When the stochastic processes employed in the models were equivalent, we confirmed the interchangeability of the models empirically. At the same time, this result verified the correctness of the mathematical modeling and computer programming for both models.

While the models can lead to the same dynamic optimum, the key difference between them lies in their approach to decomposing the problem—by scenario or by time period—to cope with a large scale problem instance. The decomposition approach directly affects the approximations needed for the problem to stay manageable. The approximation scheme used—approximate in the scenarios to plan for or in the search for optimality—then directly affects the performance of the models.

The performance of the models has been examined in a real-world setting using simulated data and real data. We found that the scenario-free model led to lower average incurred delay cost and lower variation in costs in both cases. More importantly, the scenario-free model is more flexible in assigning ground delays and thus more effective in striking the balance between the tradeoffs of ground and airborne delay. Results from the experiments suggest that branch misidentification is the primary cause for the lower performance of the scenario-based model while the scenario-free model sidesteps this problem. Moreover, the scenario-free model without advanced knowledge of the future performs almost as well as when advanced information is given.

However, comprehensive testing is required before we can suggest the scenario-free model as the better approach to plan for ground holdings. Our experiments were based on problem instances with typical air traffic load and GDP duration at SFO. Problem instances with higher air traffic load or prolonged GDP duration will require more computing resources to handle and the performance of the models could be different in those conditions. Furthermore, the current scenario-free model assumes a first-order Markovian stochastic process while the capacity evolution is actually history dependent.

Methods for incorporating knowledge of the trend of capacity evolution in the scenario-free model is another area that needs more investigation.

CHAPTER 6 CONCLUSIONS

Optimal decision-making for the single airport ground holding problem has been researched in a number of studies in the past two decades. Nonetheless many issues pertaining to the management of uncertainty in the system and the applicability of the optimization models in a real-world setting remain unresolved. In this dissertation, we explored methods to model capacity uncertainty in the context of a ground delay program and decision making strategies that respond to conditions dynamically such that the overall system performance is optimized.

In one approach, capacity uncertainty in the SAGHP is represented using scenarios and stochastic optimization models. We refer to these as scenario-based models. In Chapter 3, we have identified, and proposed solutions for, some of the challenges faced in applying scenario-based models in the real world. Clearly, scenario-based methods require scenarios or scenario trees, and we have demonstrated the use of clustering to construct them from real-world data. Our results demonstrated the applicability of the scenario tree concept at several airports, but also suggested that at many other airports such trees do not exist.

Using the scenarios and scenario trees as input, we investigated the real-world applicability of scenario-based models to the SAGHP, including the static model of Ball et al. and the dynamic Mukherjee-Hansen model. We showed that the dynamic model, by anticipating and then using the new information that becomes available after a branching point, can reduce delay costs over 60 percent when compared to the static model in the idealized case when actual capacity profiles precisely follow the scenario profiles.

Recognizing that the idealized case does not match reality, we considered the challenges and additional costs of employing these models in a real-world setting where capacity dispersion is present. With some simple methods to mitigate the cost of dispersion, the dynamic model reduces incurred costs by about one-third compared to the static model, a smaller improvement than in the case without dispersion, but a significant one nonetheless.

The primary shortcoming of the scenario-based models is that they assume limited number of capacity scenarios for a reality where there is a much larger set of possibilities for capacity evolution. This assumption has a direct impact on the ease of implementing policies from such models and the quality of the solutions they generate. In light of these limitations, in Chapter 4, we developed a scenario-free sequential decision model for the SAGHP. We formulated dynamic programs for this scenario-free model. The challenge that arose from this approach was how to manage of the curse of dimensionality. From our observation of a large amount of overlapping subproblems in the dynamic program, we chose to memoize the top-down recursive algorithm rather than solving the problem from the bottom up. We found that memoization cuts down the computation time dramatically without any loss of optimality. Moreover, we explored several strategies that can further reduce computation time but may result in suboptimal solutions, including the use of priority ordering in the decision process and heuristics that limit the search for optimum. With these computational strategies, we demonstrated that the dynamic program can solve a typical instance of SAGHP in the real world within acceptable time.

In Chapter 5, we discussed the relationship between the scenario-based and scenario-free model for SAGHP and compared their performance in a real-world setting. We

found that the models are interchangeable when the stochastic processes employed in the models are equivalent. But for large-scale problems, the decomposition schemes used in the formulations for the two models dictate different approximations required for the problems to stay computationally manageable. The scenario-based model approximates in the scenarios to plan for, while the scenario-free approach approximates in the search for optimum. Both approximation schemes involve trades between computability and solution quality. We examined the performance of the models in a real-world setting using simulated data and real data. We found that the scenario-free model led to lower average incurred delay cost and lower variation in costs in both cases. Moreover, we found the scenario-free model flexible in assigning ground delays and thus more effective in striking the balance between the tradeoffs of ground and airborne delay. Our results also suggest that the scenario-free model has a relatively stable performance with or without transition matrices tailored to the day of operation. In addition, the scenario-free model performs better than the scenario-based model, even when the scenario-based strategy is deployed in an ideal condition.

6.1 Recommendations for Future Work

This dissertation lays ground work, and suggests some directions, for future work in air traffic flow management. Many pressing problems still remain. We list some of them in this section.

6.1.1 Characterization of the Stochastic Process

6.1.1.1 Enhancement of Scenario Trees

In this study, the applications are based on “annualized” scenario trees that are unlikely to match the situation on a particular day. The challenge of blending information about the general patterns followed by capacity profiles with information specific to a particular day to form a “customized” tree has yet to be addressed. The benefits of scenario-based air traffic management cannot be adequately assessed until a method for developing such customized trees has been developed. While the methods presented in this research employ only historical capacity information, many other information sources, such as weather forecasts, can be brought to bear on the problem of assigning scenario probabilities and navigating scenario trees on a particular day. For example, one might relate scenario probabilities for a particular day to the 24-hour weather forecast for that day issued the night before. Alternatively, the capacity profile clustering could be performed for different subsets of days defined by different forecast “types” which might themselves be identified through cluster analysis or some other such technique. Methods for leveraging such forecast information for scenario-building and GHP decision making is an important opportunity for further research.

6.1.1.2 Methodology for Transition Matrix Estimation

The capacity transition matrices used in Chapter 5 for the scenario-free model were developed with a rather primitive method. Ideally the matrices should also be conditional on the runway configuration, wind speed and direction, weather condition, among other imaginable factors. Transition matrices estimated using rigorous methodologies could

enhance the performance of the scenario-free model. Multinomial probit model and hidden Markov model are potential candidates for such effort.

6.1.1.3 History-dependent Markov Process

One major simplifying assumption made for the scenario-free model is that the capacity evolution is a first-order Markov process. Though the performance of the model based on such assumption turned out to be competitive as shown in Chapter 5, a history-dependent stochastic process is a better characterization of the phenomenon than a memoryless one. Using a higher-order Markov process that has dependency on the capacity in two or more prior periods is one way to better characterize the capacity evolution. The challenge of employing a higher-order Markov process in the scenario-free model lies in the complexity it brings to the solution algorithm.

6.1.2 *Assessment of Model Performance*

6.1.2.1 Justice to the Static Model

The comparisons between dynamic and static models in Chapter 3 did not take into account that, in application, a static model is unlikely to be used statically. Rather, it would be re-applied either on a regular basis, or when new information required it. While the dynamic model explicitly allows us to “plan to re-plan”, replanning per se is possible with either model. An experiment comparing an explicitly dynamic model with repeated runs of a static model is also a ripe topic for further research.

6.1.2.2 Generalizability of the Performance Results

We have compared the performance of the models using data of the typical air traffic load, capacity condition, and GDP duration at SFO. However, performance of the models

under extreme demand-capacity imbalance conditions and prolonged GDP duration has not been investigated. More empirical studies are needed to confirm the generalizability of our results.

6.1.3 Complexity of the Scenario-free Approach

6.1.3.1 Estimator for the Computational Cost

In Chapter 4, we showed complexity analysis for the algorithm without the use of memoization. The complexity analysis for the algorithm with memoization is more complicated but it can help determine whether a given problem instance can be solved in acceptable time with the available computing resources. Such estimator will be very desirable if the algorithm is considered for real-time application.

6.1.3.2 Other Strategies to Manage the Complexity

We mentioned briefly in Chapter 4 about time epoch aggregation as a way to manage the computational load, but we have not studied the performance of such strategy in detail. Another strategy to alleviate the computational load due to a long planning horizon is to run the algorithm with limited look-ahead at multiple time points. The structural property of the cost-to-go function may suggest other interesting heuristics. For example, treating the discrete decision variable as continuous and approximate the solution to the nearest integer value could help find a better solution. These and other strategies to manage the complexity make another area for future research.

6.1.4 Refinement for the Scenario-Free Model

6.1.4.1 Other Priority Ordering Schemes

We proposed two priority ordering schemes in Chapter 4—LGF and RBS. Clearly, there can be many other ways to prioritize the flights. One simple example could be using a weighted score that accounts for both the LGF and RBS concepts. Methodologies for constructing equitable priority orderings among the flights is another area for further research.

6.1.4.2 Nonlinear Cost

One may argue that it is reasonable to have nonlinear delay cost, as a longer delay should induce a nonlinearly higher penalty. Unlike in the stochastic program, it is more complicated to have quadratic cost or cost in other nonlinear forms for the dynamic program because the cost is computed additively throughout the time periods. However, since the algorithm tracks each of the flights, it is possible to assign higher incremental cost for longer delay. This could be an important extension for the scenario-free model to deliver convincing solutions.

6.1.5 Implementation of the Scenario-Free Approach

6.1.5.1 Benefits from using the Scenario-Free Approach

There are other important benefits from using the scenario-free model that we have not explored in this dissertation. The sequential decision model tree carries abundant information that can be leveraged in making decisions. At each decision point, probabilistic evaluation of the future condition and implications of each action are readily available. Moreover, the sequential decision model provides the infrastructure for

decision-making based on criteria other than the expected cost. One drawback of the models we have discussed is that they assume the decision maker wants to minimize expected cost. The optimal actions suggested by such model might not satisfy a system manager who wants to act more aggressively or conservatively. An alternative objective could be in the following form: an action is preferred if the chance of total cost exceeding C is less than P %. The sequential decision model has potential to accommodate this type of objective, and that is an interesting area for future research.

6.1.5.2 Incorporation of the Scenario-Free Approach into CDM

Before the scenario-free approach can be considered for deployment in the field, thorough investigation of the compatibility with and impact on the current system of operations is needed. More specifically, how existing CDM practices can be adapted to incorporate a scenario-free approach is an important topic for future research. The challenge lies in ensuring the reconciliation between the logic of the optimization model with the business objectives of the participants—both the service provider and users—in the CDM process. The acceptance of the scenario-free resource allocation approach by the aviation community will need mechanisms for ensuring fairness and allowing user flexibility.

6.1.6 *Application of Model Interchangeability*

In Chapter 5, we have identified the condition for the stochastic program and dynamic program to be interchangeable. Many problems in other fields have been formulated in one or the other optimization model. Potentially, there are other applications that may find the other way of model formulation fruitful.

This dissertation has presented methods and analysis that we believe will be useful for air traffic flow management. Despite the promising results we have shown, many interesting and important problems remain. We believe that further efforts to study these problems will pay off richly.

BIBLIOGRAPHY

- Ahuja, R. K., T. L. Magnanti, and J. B. Orlin. 1993. *Network Flows*. Prentice-Hall, Upper Saddle River, NJ.
- Andreatta, G., L. Brunetta. 1998. Multiairport ground holding problem: A computational evaluation of exact algorithms. *Operations Research* **46**(1) 57–64.
- Andreatta, G., L. Brunetta, G. Guastalla. 2000. From ground holding to free flight: An exact approach. *Transportation Science* **34**(4) 394–401.
- Andreatta, G., A. R. Odoni, O. Richetta. 1993. Models for the ground-holding problem. *Large-Scale Computation and Information Processing in Air Traffic Control*. Springer Verlag, 125–168.
- Andreatta, G., G. Romanin-Jacur. 1987. Aircraft flow management under congestion. *Transportation Science* **21**(4) 249–253.
- Andreatta, G., G. Tidona. 1994. A new formulation for the multiairport ground holding problem. Tech. Rep. 3, Dip. di Matematica Pura e Applicata. University of Padova, Italy.
- Ang, A. H-S., and W. H. Tang. 1975. *Probability Concepts in Engineering Planning and Design*. John Wiley & Sons.
- Ball, M., R. Hoffman, D. J. Lovell, A. Mukherjee. 2005. Response mechanisms for dynamic air traffic flow management. Proceedings of the 6th USA/Europe Air Traffic Management R&D Seminar, Baltimore, MD.

- Ball, M., T. Vossen, R. Hoffman. 2001. Analysis of demand uncertainty effects in ground delay programs. Proceedings of the 4th USA/Europe Air Traffic Management R&D Seminar, Santa Fe, NM.
- Ball, M. O., R. Hoffman, A. R. Odoni, R. Rifkin. 2003. A stochastic integer program with dual network structure and its application to the ground-holding problem. *Operations Research* **51** 167–171.
- Ball, M. O., G. Lulli. 2004. Ground delay programs: Optimizing over the included flight set based on distance. *Air Traffic Control Quarterly* **12** 1–25.
- Bertsekas, D. P. 2000. Dynamic Programming and Optimal Control. Athena Scientific, Belmont, MA.
- Bertsekas, D. P., and J. N. Tsitsiklis. 1996. Neuro-dynamic Programming. Athena Scientific, Belmont, MA.
- Bertsimas, D., S. S. Patterson. 1994. The air traffic flow management problem with enroute capacities. Working Paper No. 3726-94 MSA, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA.
- Bertsimas, D., S. S. Patterson. 1998. The air traffic flow management problem with enroute capacities. *Operations Research* **46** 406–422.
- Bertsimas, D., S. S. Patterson. 2000. The traffic flow management rerouting problem in air traffic control: A dynamic network flow approach. *Transportation Science* **34** 239–255.
- Birge, J.R., and F. Louveaux. 1997. Introduction to Stochastic Programming. Springer-Verlag, New York, NY.

- Brennan, M., C. Ermatinger, J. Burke, L. Hathaway, V. Sud. 2005. Evaluating en route congestion management through interactive simulation. Proceedings of the 6th USA/Europe Air Traffic Management R&D Seminar, Baltimore, MD.
- Bruegge, B., and A.H. Dutoit. 2003. Object-Oriented Software Engineering: Using UML, Patterns, and Java. Pearson Prentice Hall, Upper Saddle River, NJ.
- Calinski, T., J. Harabasz. 1974. A Dendrite Method for Cluster Analysis. *Communications in Statistics* **3** 1-27.
- Cormen, T.H., C.E. Leiserson, R.L. Rivest. 1989. Introduction to algorithms. The MIT press.
- Delahaye, D., A.R. Odoni. 1997. Airspace congestion smoothing by stochastic optimization. Peter J. Angeline, Robert G. Reynolds, John R. McDonnell, Russ Eberhart, eds., *Evolutionary Programming VI*. Springer, Berlin, 163–176.
- De Neufville, R., and A. Odoni. 2003. Airport Systems: Planning, Design, and Management. McGraw-Hill.
- Duda, R.O., P.E. Hart. 1973. Pattern Classification and Scene Analysis. John Wiley & Sons, New York.
- Dupacova, J., G. Consigli, S.W. Wallace. 2000. Scenarios for Multistage Stochastic Programs. *Annals of Operations Research* **100** 25-53.
- FAA. 2005. Operational Evolution Plan Version 7 Executive Summary, February.
- Filar, J. A., P. Manyem, M. S. Visser, K. White. 2003. Air traffic management at sidney with cancellations and curfew penalties. *Optimization and Industry: New Frontiers*. Kluwer Academic Publishers, 113–140.

- Filar, J. A., P. Manyem, K. White. 2001. How airlines and airports recover from schedule perturbations: A survey. *Annals of Operations Research* 315–333.
- Gosavi, A. 2003. Simulation-based Optimization: Parametric Optimization Techniques and Reinforcement Learning. Kluwer Academic Publishers, Norwell, MA.
- Gulpinar, N., Rustem, B., Settergren, R., 2004. Simulation and Optimization Approaches to Scenario Tree Generation. *Journal of Economic Dynamics & Control* **28** 1291-1315.
- Helme, M. P. 1992. Reducing air traffic delay in a space-time network. IEEE International Conference on Systems, Man and Cybernetics, Chicago.
- Hoffman, R. 1997. Integer programming models for ground-holding in air traffic flow management. Ph.D. thesis, University of Maryland, College Park.
- Hoffman, R., M. O. Ball. 2000. A comparison of formulations for the single-airport ground holding problem with banking constraints. *Operations Research* **48**(4) 578–590.
- Hoyland, K., S.W. Wallace. 2001. Generating Scenario Trees for Multi-stage Decision Problems. *Management Science* **47** (2) 295-307.
- Inniss, T. R. 2000. Stochastic Models for the Estimation of Airport Arrival Capacity Distributions. Ph.D. thesis, University of Maryland, College Park.
- Inniss, T.R., M.O. Ball. 2004. Estimating One-Parameter Airport Arrival Capacity Distributions for Air Traffic Flow Management. *Air Traffic Control Quarterly* **12** 223-251.

- Kaut, M., S.W. Wallace. 2003. Evaluation of Scenario-generation Methods for Stochastic Programming. Stochastic Programming E-Print Series, 2003-14, <http://hera.rz.hu-berlin.de/speps/>.
- Kleinman, N. L., S. D. Hill, V. A. Ilenda. 1998. Simulation optimization of air traffic delay cost. *Proceedings of the 1998 Winter Simulation Conference*.
- Law, A. M., and W. D. Kelton. 2000. Simulation Modeling and Analysis. McGraw-Hill.
- MacQueen, J.B. 1967. Some Methods for Classification and Analysis of Multivariate Observations. Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, pp. 281-297.
- Marshall, K. T., and R. M. Oliver. 1995. Decision Making and Forecasting. McGraw-Hill.
- Menon, P. K., G. D. Sweriduk, K. D. Bilimoria. 2002. A new approach for modeling, analysis and control of air traffic flow. Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit, American Institute of Aeronautics and Astronautics, Inc., Monterey, California.
- Milligan, G. W. and M. C. Cooper. 1985. An Examination of Procedures for Determining the Number of Clusters in Data Set, *Psychometrika* **50** 159-179.
- Mukherjee, A. 2004. Dynamic stochastic optimization models for air traffic flow management. Ph.D. thesis, University of California, Berkeley.
- Mukherjee, A., M. Hansen. 2005. Dynamic stochastic optimization models for air traffic flow management with en route and airport capacity constraints. Proceedings of the 6th USA/Europe Air Traffic Management R&D Seminar, Baltimore, MD.
- Nash, S. G., and A. Sofer. 1996. Linear and Nonlinear programming. McGraw-Hill.

- Navazio, L., G. Romanin-jacur. 1998. The multiple connections multi-airport ground holding problem: Models and algorithms. *Transportation Science* **32**(3) 268–276.
- Nemhauser, G., L. Wolsey. 1988. Integer and Combinatorial Optimization. John Wiley & Sons, New York, NY.
- Odoni, A.R., 1987. The Flow Management Problem in Air Traffic Control. In: Odoni, A.R., Bianco, L., Szego, G. (Eds.), Flow Control of Congested Networks. Springer Verlag, New York, pp. 269-288.
- Oussedik, S., D. Delahaye, M. Schoenauer. 1999. Dynamic air traffic planning by genetic algorithms. *Proceedings of the Congress on Evolutionary Computation*, vol. 2. IEEE Press, Washington D.C., USA, 1110–1117.
- Panayiotou, C. G., C.G. Cassandras. 2001. A sample path approach for solving the ground-holding policy problem in air traffic control. *IEEE Transactions on Control Systems Technology* **9**(3) 510–523.
- Pflug, G.C. 2001. Scenario Tree Generation for Multiperiod Financial Optimization by Optimal Discretization. *Mathematical Programming* **89** 251-271.
- Porteus, E.L. 2002. Foundations of Stochastic Inventory Theory. Stanford University Press, Stanford, CA.
- Pulugurtha, S. S., S. S. Nambisan. 2001. Using genetic algorithms to evaluate aircraft ground holding policy under static conditions. *Journal of Transportation Engineering* 433–441.
- Puterman, M.L. 2005. Markov Decision Processes. John Wiley & Sons, Hoboken, NJ.
- Richetta, O., A. R. Odoni. 1993. Solving optimally the static ground-holding policy problem in air traffic control. *Transportation Science* (27) 228–238.

- Richetta, O., A. R. Odoni. 1994. Dynamic solution to the ground-holding problem in air traffic control. *Transportation Research Part A* **28** (3) 167–185.
- Ross, S. M. 2000. Introduction to Probability Models. Academic Press, San Diego, CA.
- Rossi, F., S. Smriglio. 2001. A set packing model for the ground holding problem in congested networks. *European Journal of Operational Research* **131** 400–416.
- Roy, S., B. Sridhar, G. C. Verghese. 2003. An aggregate dynamic stochastic model for an air traffic system. Proceedings of the 5th USA/Europe Air Traffic Management R&D Seminar, Budapest, Hungary.
- Sarle, W.S. 1983. Cubic Clustering Criterion. SAS Technical Report A-108. SAS Institute, Cary, NC.
- SAS Institute. 2004. SAS/STAT 9.1 User's Guide. Cary, NC.
- Sridhar, B., T. Soni, K. Sheth, G. Chatterji. 2004. An aggregate flow model for air traffic management. Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit, Providence, Rhode Island.
- Sutton, R. S., and A. G. Barto. 1998. Reinforcement Learning: An Introduction. The MIT Press, Cambridge, MA.
- Terrab, M., A. R. Odoni. 1993. Strategic flow management for air traffic control. *Operations Research* **41**(1) 138–152.
- Terrab, M., S. Paulose. 1992. Dynamic strategic and tactical air traffic flow control. IEEE International Conference on Systems, Man and Cybernetics, Chicago.
- Vranas, P. B. M., D. Bertsimas, A. R. Odoni. 1994. Dynamic ground-holding policies for a network of airports. *Transportation Science* **28**(4) 275–291.

- Wilson, F. W. 2004. Probabilistic Forecasts of Cloud Impacts at San Francisco International Airport. Proceedings of the 20th International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology, Seattle, WA.
- Zangwill, W. 1969. Nonlinear Programming: A Unified Approach. Prentice Hall, Englewood Cliffs, NJ.

APPENDIX A INPUT FOR NUMERICAL EXPERIMENTS

A.1 Transition Matrices Estimated using AAR Data at SFO in 2003

8:00

| from\to | 7 | 8 | 11 | 15 |
|---------|--------|--------|--------|------|
| 7 | 0.96 | 0 | 0 | 0.04 |
| 8 | 0.0500 | 0.9417 | 0.0083 | 0 |
| 12 | 0.0345 | 0 | 0.9655 | 0 |
| 15 | 0 | 0 | 0 | 1 |

8:15

| from\to | 7 | 11 | 15 |
|---------|--------|--------|--------|
| 7 | 0.9032 | 0.0323 | 0.0645 |
| 8 | 1 | 0 | 0 |
| 11 | 0 | 0.9697 | 0.0303 |
| 15 | 0.0225 | 0.0056 | 0.9719 |

8:30

| from\to | 6 | 7 | 11 | 15 |
|---------|--------|--------|--------|--------|
| 7 | 0.1586 | 0.8138 | 0.0276 | 0 |
| 11 | 0 | 0 | 0.9412 | 0.0588 |
| 15 | 0 | 0.0114 | 0 | 0.9886 |

8:45

| from\to | 7 | 8 | 12 | 15 |
|---------|--------|--------|--------|--------|
| 6 | 0.9565 | 0 | 0.0435 | 0 |
| 7 | 0 | 0.975 | 0 | 0.025 |
| 11 | 0 | 0 | 0.9310 | 0.0690 |
| 15 | 0 | 0.0114 | 0.0057 | 0.9829 |

9:00

| from\to | 7 | 8 | 11 | 15 |
|---------|--------|--------|--------|----|
| 7 | 0.9524 | 0.0476 | 0 | 0 |
| 8 | 0.1008 | 0.8824 | 0.0168 | 0 |
| 12 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |

9:15

| from\to | 7 | 11 | 15 |
|---------|--------|--------|--------|
| 7 | 0.6250 | 0.1250 | 0.2500 |
| 8 | 1 | 0 | 0 |
| 11 | 0 | 0.9706 | 0.0294 |
| 15 | 0 | 0 | 1 |

9:30

| from\to | 6 | 7 | 11 | 15 |
|---------|--------|--------|--------|--------|
| 7 | 0.1654 | 0.8346 | 0 | 0 |
| 11 | 0 | 0.0270 | 0.8919 | 0.0811 |
| 15 | 0 | 0 | 0 | 1 |

9:45

| from\to | 7 | 8 | 12 | 15 |
|---------|--------|--------|--------|--------|
| 6 | 1 | 0 | 0 | 0 |
| 7 | 0.0185 | 0.9259 | 0.0278 | 0.0278 |
| 11 | 0 | 0 | 0.8710 | 0.1290 |
| 15 | 0 | 0 | 0 | 1 |

10:00

| from\to | 7 | 8 | 11 | 15 |
|---------|--------|--------|--------|--------|
| 7 | 0.9167 | 0 | 0.0417 | 0.0417 |
| 8 | 0.1569 | 0.8039 | 0.0392 | 0 |
| 12 | 0.0333 | 0 | 0.9667 | 0 |
| 15 | 0 | 0 | 0 | 1 |

10:15

| from\to | 7 | 11 | 15 |
|---------|--------|--------|--------|
| 7 | 0.7949 | 0.1538 | 0.0513 |
| 8 | 1 | 0 | 0 |
| 11 | 0.0270 | 0.8919 | 0.0811 |
| 15 | 0 | 0 | 1 |

10:30

| from\to | 6 | 7 | 11 | 15 |
|---------|--------|--------|--------|--------|
| 7 | 0.1842 | 0.7456 | 0.0439 | 0.0263 |
| 11 | 0 | 0 | 0.9487 | 0.0513 |
| 15 | 0 | 0 | 0 | 1 |

10:45

| from\to | 7 | 8 | 12 | 15 |
|---------|--------|--------|--------|--------|
| 6 | 0.9545 | 0 | 0.0455 | 0 |
| 7 | 0.0588 | 0.9176 | 0 | 0.0235 |
| 11 | 0 | 0 | 0.9268 | 0.0732 |
| 15 | 0 | 0 | 0 | 1 |

11:00

| from\to | 7 | 8 | 11 | 15 |
|---------|-------|------|-------|-----|
| 7 | 0.8 | 0 | 0 | 0.2 |
| 8 | 0.10 | 0.85 | 0.05 | 0 |
| 12 | 0.025 | 0 | 0.975 | 0 |
| 15 | 0 | 0 | 0 | 1 |

11:15

| from\to | 7 | 11 | 15 |
|---------|--------|--------|--------|
| 7 | 0.8214 | 0.1429 | 0.0357 |
| 8 | 1 | 0 | 0 |
| 11 | 0 | 0.9773 | 0.0227 |
| 15 | 0 | 0 | 1 |

11:30

| from\to | 6 | 7 | 11 | 15 |
|---------|--------|--------|--------|--------|
| 7 | 0.1978 | 0.7582 | 0.0330 | 0.0110 |
| 11 | 0 | 0 | 0.9787 | 0.0213 |
| 15 | 0 | 0 | 0 | 1 |

11:45

| from\to | 7 | 8 | 12 | 15 |
|---------|--------|--------|--------|--------|
| 6 | 1 | 0 | 0 | 0 |
| 7 | 0.0597 | 0.8657 | 0.0299 | 0.0448 |
| 11 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |

12:00

| from\to | 7 | 8 | 11 | 15 |
|---------|--------|--------|--------|--------|
| 7 | 0.8182 | 0 | 0.0455 | 0.1364 |
| 8 | 0.0172 | 0.9138 | 0.0690 | 0 |
| 12 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |

12:15

| from\to | 7 | 11 | 15 |
|---------|------|-------|-------|
| 7 | 0.95 | 0.05 | 0 |
| 8 | 1 | 0 | 0 |
| 11 | 0 | 0.949 | 0.051 |
| 15 | 0 | 0 | 1 |

12:30

| from\to | 6 | 7 | 11 | 15 |
|---------|--------|--------|----------|----------|
| 7 | 0.2432 | 0.7568 | 0 | 0 |
| 11 | 0 | 0 | 0.983051 | 0.016949 |
| 15 | 0 | 0 | 0 | 1 |

12:45

| from\to | 7 | 8 | 12 | 15 |
|---------|---|--------|----|--------|
| 6 | 1 | 0 | 0 | 0 |
| 7 | 0 | 0.9818 | 0 | 0.0182 |
| 11 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |

A.2 Adapted Schedule of Incoming Flights to SFO on March 21, 2006

| Departure Hour | Departure Quarter | Number of Flights | Flight Duration | | | | | | | | | | | | |
|----------------|-------------------|-------------------|-----------------|----|----|----|----|----|----|----|----|----|----|--|--|
| 8 | 1 | 11 | 4 | 5 | 7 | 8 | 9 | 9 | 10 | 10 | 13 | 13 | 14 | | |
| 8 | 2 | 9 | 3 | 3 | 5 | 5 | 5 | 6 | 9 | 9 | 11 | | | | |
| 8 | 3 | 8 | 4 | 6 | 7 | 11 | 11 | 11 | 13 | 14 | | | | | |
| 8 | 4 | 4 | 4 | 7 | 10 | 14 | | | | | | | | | |
| 9 | 1 | 10 | 2 | 3 | 4 | 5 | 6 | 7 | 9 | 9 | 12 | 14 | | | |
| 9 | 2 | 11 | 6 | 6 | 6 | 6 | 7 | 7 | 7 | 8 | 8 | 9 | 13 | | |
| 9 | 3 | 5 | 3 | 6 | 8 | 9 | 11 | | | | | | | | |
| 9 | 4 | 5 | 5 | 6 | 6 | 10 | 10 | | | | | | | | |
| 10 | 1 | 5 | 3 | 4 | 4 | 6 | 13 | | | | | | | | |
| 10 | 2 | 5 | 5 | 6 | 6 | 7 | 11 | | | | | | | | |
| 10 | 3 | 7 | 4 | 5 | 6 | 6 | 7 | 9 | 11 | | | | | | |
| 10 | 4 | 2 | 2 | 4 | | | | | | | | | | | |
| 11 | 1 | 10 | 3 | 3 | 4 | 4 | 5 | 6 | 6 | 7 | 9 | 12 | | | |
| 11 | 2 | 2 | 8 | 14 | | | | | | | | | | | |
| 11 | 3 | 11 | 4 | 5 | 5 | 6 | 6 | 7 | 8 | 9 | 9 | 9 | 13 | | |
| 11 | 4 | 6 | 3 | 4 | 5 | 6 | 8 | 8 | | | | | | | |
| 12 | 1 | 3 | 4 | 4 | 8 | | | | | | | | | | |
| 12 | 2 | 4 | 4 | 5 | 7 | 8 | | | | | | | | | |
| 12 | 3 | 2 | 6 | 6 | | | | | | | | | | | |
| 12 | 4 | 2 | 6 | 6 | | | | | | | | | | | |

A.3 Transition Matrices for Type 1 Condition

8:00

| from\to | 7 | 8 | 11 | 15 |
|---------|--------|--------|--------|----|
| 7 | 1 | 0 | 0 | 0 |
| 8 | 0.0187 | 0.9720 | 0.0093 | 0 |
| 12 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |

8:15

| from\to | 7 | 11 | 15 |
|---------|------|----|------|
| 7 | 1 | 0 | 0 |
| 8 | 1 | 0 | 0 |
| 11 | 0 | 1 | 0 |
| 15 | 0.25 | 0 | 0.75 |

8:30

| from\to | 6 | 7 | 11 | 15 |
|---------|--------|--------|--------|--------|
| 7 | 0.1692 | 0.8154 | 0.0154 | 0 |
| 11 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0.6667 | 0 | 0.3333 |

8:45

| from\to | 7 | 8 | 12 | 15 |
|---------|---|--------|----|--------|
| 6 | 1 | 0 | 0 | 0 |
| 7 | 0 | 0.9815 | 0 | 0.0185 |
| 11 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |

9:00

| from\to | 7 | 8 | 11 | 15 |
|---------|--------|--------|----|----|
| 7 | 0.9524 | 0.0476 | 0 | 0 |
| 8 | 0.0755 | 0.9245 | 0 | 0 |
| 12 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |

9:15

| from\to | 7 | 11 | 15 |
|---------|--------|--------|--------|
| 7 | 0.7143 | 0.1071 | 0.1786 |
| 8 | 1 | 0 | 0 |
| 11 | 0 | 0.875 | 0.125 |
| 15 | 0 | 0 | 1 |

9:30

| from\to | 6 | 7 | 11 | 15 |
|---------|--------|--------|-----|-----|
| 7 | 0.1765 | 0.8235 | 0 | 0 |
| 11 | 0 | 0.1 | 0.7 | 0.2 |
| 15 | 0 | 0 | 0 | 1 |

9:45

| from\to | 7 | 8 | 12 | 15 |
|---------|--------|--------|--------|--------|
| 6 | 1 | 0 | 0 | 0 |
| 7 | 0.0101 | 0.9394 | 0.0202 | 0.0303 |
| 11 | 0 | 0 | 0.8571 | 0.1429 |
| 15 | 0 | 0.0909 | 0 | 0.9091 |

10:00

| from\to | 7 | 8 | 11 | 15 |
|---------|--------|--------|--------|----|
| 7 | 0.9565 | 0 | 0.0435 | 0 |
| 8 | 0.1383 | 0.8298 | 0.0319 | 0 |
| 12 | 0.125 | 0 | 0.875 | 0 |
| 15 | 0 | 0 | 0 | 1 |

10:15

| from\to | 7 | 11 | 15 |
|---------|--------|--------|--------|
| 7 | 0.7778 | 0.1667 | 0.0556 |
| 8 | 1 | 0 | 0 |
| 11 | 0.0909 | 0.8182 | 0.0909 |
| 15 | 0 | 0 | 1 |

10:30

| from\to | 6 | 7 | 11 | 15 |
|---------|--------|--------|--------|--------|
| 7 | 0.1963 | 0.7570 | 0.0374 | 0.0093 |
| 11 | 0 | 0 | 0.8667 | 0.1333 |
| 15 | 0 | 0 | 0 | 1 |

10:45

| from\to | 7 | 8 | 12 | 15 |
|---------|--------|--------|--------|--------|
| 6 | 0.9545 | 0 | 0.0455 | 0 |
| 7 | 0.0370 | 0.9383 | 0 | 0.0247 |
| 11 | 0 | 0 | 0.875 | 0.125 |
| 15 | 0 | 0.05 | 0 | 0.95 |

11:00

| from\to | 7 | 8 | 11 | 15 |
|---------|--------|--------|--------|--------|
| 7 | 0.8696 | 0 | 0 | 0.1304 |
| 8 | 0.0909 | 0.8571 | 0.0519 | 0 |
| 12 | 0.0667 | 0 | 0.9333 | 0 |
| 15 | 0 | 0 | 0 | 1 |

11:15

| from\to | 7 | 11 | 15 |
|---------|--------|--------|--------|
| 7 | 0.8519 | 0.1111 | 0.0370 |
| 8 | 1 | 0 | 0 |
| 11 | 0 | 0.9474 | 0.0526 |
| 15 | 0 | 0 | 1 |

11:30

| from\to | 6 | 7 | 11 | 15 |
|---------|--------|--------|--------|--------|
| 7 | 0.2022 | 0.7528 | 0.0337 | 0.0112 |
| 11 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |

11:45

| from\to | 7 | 8 | 12 | 15 |
|---------|--------|--------|--------|--------|
| 6 | 1 | 0 | 0 | 0 |
| 7 | 0.0615 | 0.8769 | 0.0308 | 0.0308 |
| 11 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |

12:00

| from\to | 7 | 8 | 11 | 15 |
|---------|--------|--------|--------|--------|
| 7 | 0.8182 | 0 | 0.0455 | 0.1364 |
| 8 | 0.0175 | 0.9298 | 0.0526 | 0 |
| 12 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |

12:15

| from\to | 7 | 11 | 15 |
|---------|--------|--------|--------|
| 7 | 0.95 | 0.05 | 0 |
| 8 | 1 | 0 | 0 |
| 11 | 0 | 0.9355 | 0.0645 |
| 15 | 0.0571 | 0 | 0.9429 |

12:30

| from\to | 6 | 7 | 11 | 15 |
|---------|--------|--------|--------|--------|
| 7 | 0.2432 | 0.7568 | 0 | 0 |
| 11 | 0 | 0 | 0.9667 | 0.0333 |
| 15 | 0 | 0 | 0 | 1 |

12:45

| from\to | 7 | 8 | 12 | 15 |
|---------|---|--------|--------|--------|
| 6 | 1 | 0 | 0 | 0 |
| 7 | 0 | 0.9818 | 0 | 0.0182 |
| 11 | 0 | 0.1071 | 0.8929 | 0 |
| 15 | 0 | 0 | 0 | 1 |

A.4 Transition Matrices for Type 2 Condition

8:00

| from\to | 7 | 8 | 11 | 13 | 15 |
|---------|------|------|-----|----|----|
| 7 | 0 | 0 | 0 | 0 | 1 |
| 8 | 0.75 | 0.25 | 0 | 0 | 0 |
| 11 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0.2 | 0 | 0.8 | 0 | 0 |
| 13 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0 | 1 |
| 16 | 0 | 0 | 0 | 0 | 1 |

8:15

| from\to | 7 | 11 | 12 | 15 |
|---------|--------|--------|----|--------|
| 7 | 0.5 | 0 | 0 | 0.5 |
| 8 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0.8571 | 0 | 0.1429 |
| 13 | 0 | 0 | 1 | 0 |
| 15 | 0.0116 | 0.0058 | 0 | 0.9826 |

8:30

| from\to | 7 | 11 | 12 | 15 |
|---------|-----|--------|----|--------|
| 7 | 0.6 | 0.4 | 0 | 0 |
| 11 | 0 | 0.7143 | 0 | 0.2857 |
| 12 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |

8:45

| from\to | 8 | 11 | 12 | 13 | 15 | 16 |
|---------|--------|--------|--------|----|--------|--------|
| 7 | 0.6667 | 0 | 0 | 0 | 0.3333 | 0 |
| 11 | 0 | 0.4286 | 0.2857 | 0 | 0.2857 | 0 |
| 12 | 0 | 0 | 0 | 1 | 0 | 0 |
| 15 | 0.0115 | 0 | 0.0057 | 0 | 0.9770 | 0.0057 |

9:00

| from\to | 7 | 8 | 11 | 13 | 15 |
|---------|------|------|----|----|----|
| 8 | 0.75 | 0.25 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 1 |
| 12 | 0 | 0 | 1 | 0 | 0 |
| 13 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0 | 1 |
| 16 | 0 | 0 | 0 | 0 | 1 |

9:15

| from\to | 7 | 11 | 12 | 15 |
|---------|-------|----|----|--------|
| 7 | 0 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 11 | 0 | 1 | 0 | 0 |
| 13 | 0 | 0 | 1 | 0 |
| 15 | 0.006 | 0 | 0 | 0.9944 |

9:30

| from\to | 7 | 11 | 12 | 15 |
|---------|--------|----|----|--------|
| 7 | 1 | 0 | 0 | 0 |
| 11 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 1 | 0 |
| 15 | 0.0056 | 0 | 0 | 0.9944 |

9:45

| from\to | 8 | 11 | 12 | 13 | 15 |
|---------|--------|--------|--------|----|--------|
| 7 | 0.6667 | 0 | 0.3333 | 0 | 0 |
| 11 | 0 | 0.3333 | 0 | 0 | 0.6667 |
| 12 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0.0056 | 0 | 0 | 0 | 0.9944 |

10:00

| from\to | 7 | 8 | 11 | 13 | 15 |
|---------|--------|--------|--------|----|--------|
| 8 | 0.3333 | 0.3333 | 0.3333 | 0 | 0 |
| 11 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 1 | 0 | 0 |
| 13 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0.0056 | 0 | 0.9944 |

10:15

| from\to | 7 | 11 | 12 | 15 |
|---------|---|------|----|------|
| 7 | 1 | 0 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0.75 | 0 | 0.25 |
| 13 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |

10:30

| from\to | 7 | 10 | 11 | 12 | 15 |
|---------|-----|--------|----|----|--------|
| 7 | 0.5 | 0 | 0 | 0 | 0.5 |
| 11 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0.0056 | 0 | 0 | 0.9944 |

10:45

| from\to | 7 | 8 | 10 | 12 | 13 | 15 |
|---------|---|--------|----|--------|----|--------|
| 7 | 1 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0.6667 | 0 | 0.3333 |
| 12 | 0 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0.0056 | 0 | 0.0056 | 0 | 0.9888 |

11:00

| from\to | 8 | 10 | 11 | 13 | 15 |
|---------|---|----|----|----|----|
| 7 | 0 | 0 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 | 0 |
| 12 | 0 | 0 | 1 | 0 | 0 |
| 13 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0 | 1 |

11:15

| from\to | 7 | 10 | 11 | 12 | 15 |
|---------|---|----|----|----|----|
| 8 | 1 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 1 | 0 | 0 |
| 13 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0 | 1 |

11:30

| from\to | 7 | 10 | 11 | 12 | 15 |
|---------|---|----|----|----|----|
| 7 | 1 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0 | 1 |

11:45

| from\to | 8 | 10 | 11 | 12 | 13 | 15 |
|---------|---|----|--------|--------|----|--------|
| 7 | 1 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0.3333 | 0.6667 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0.0056 | 0 | 0.9944 |

12:00

| from\to | 10 | 11 | 13 | 15 |
|---------|----|--------|----|--------|
| 8 | 0 | 1 | 0 | 0 |
| 10 | 1 | 0 | 0 | 0 |
| 11 | 0 | 1 | 0 | 0 |
| 12 | 0 | 1 | 0 | 0 |
| 13 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0.0056 | 0 | 0.9944 |

12:15

| from\to | 10 | 11 | 12 | 15 |
|---------|----|--------|----|--------|
| 10 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0.8333 | 0 | 0.1667 |
| 13 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0.0113 | 0 | 0.9887 |

12:30

| from\to | 10 | 11 | 12 | 15 |
|---------|----|--------|----|--------|
| 10 | 1 | 0 | 0 | 0 |
| 11 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0.0170 | 0 | 0.9830 |

12:45

| from\to | 8 | 10 | 12 | 13 | 15 |
|---------|-------|----|-------|----|--------|
| 10 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0.006 | 0 | 0.006 | 0 | 0.9884 |

A.5 Transition Matrices for Type 3 Condition

8:00

| from\to | 7 | 8 | 10 | 11 | 12 | 15 |
|---------|--------|--------|----|-----|----|-----|
| 7 | 1 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0.1111 | 0.8889 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 1 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 1 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0.5 | 0 | 0.5 |
| 16 | 0 | 0 | 0 | 0 | 0 | 1 |

8:15

| from\to | 7 | 10 | 11 | 12 | 15 |
|---------|-----|----|-----|----|-----|
| 7 | 0.5 | 0 | 0.5 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0.5 | 0 | 0 | 0 | 0.5 |

8:30

| from\to | 6 | 7 | 10 | 11 | 12 | 15 |
|---------|-----|-----|----|----|----|----|
| 7 | 0.1 | 0.9 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 1 |

8:45

| from\to | 8 | 10 | 11 | 12 | 13 | 15 |
|---------|---|----|------|------|----|----|
| 6 | 0 | 0 | 0 | 1 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0.05 | 0.95 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 1 |

9:00

| from\to | 7 | 8 | 10 | 11 | 12 | 15 |
|---------|--------|--------|----|--------|----|----|
| 8 | 0.1111 | 0.6667 | 0 | 0.2222 | 0 | 0 |
| 10 | 0 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 0 | 1 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 1 |

9:15

| from\to | 7 | 10 | 11 | 12 | 15 |
|---------|---|----|----|----|----|
| 7 | 0 | 0 | 1 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0 | 1 |

9:30

| from\to | 7 | 10 | 11 | 12 | 15 |
|---------|---|----|--------|----|--------|
| 7 | 1 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0.9583 | 0 | 0.0417 |
| 12 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0 | 1 |

9:45

| from\to | 7 | 8 | 10 | 11 | 12 | 13 | 15 | 16 |
|---------|--------|--------|----|--------|--------|----|--------|-----|
| 7 | 0.1667 | 0.8333 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0.0435 | 0.9130 | 0 | 0.0435 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0.5 | 0.5 |

10:00

| from\to | 7 | 8 | 10 | 11 | 12 | 15 |
|---------|-----|-----|-----|----|----|-----|
| 7 | 0 | 0 | 0 | 0 | 0 | 1 |
| 8 | 0.4 | 0.6 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0.5 | 0 | 0 | 0.5 |
| 11 | 0 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 0 | 1 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 1 |
| 16 | 0 | 0 | 0 | 0 | 0 | 1 |

10:15

| from\to | 6 | 7 | 10 | 11 | 12 | 15 |
|---------|-----|---|----|--------|----|--------|
| 7 | 0 | 1 | 0 | 0 | 0 | 0 |
| 8 | 0 | 1 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0.9545 | 0 | 0.0455 |
| 12 | 0 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0.2 | 0 | 0 | 0 | 0 | 0.8 |

10:30

| from\to | 6 | 7 | 10 | 11 | 12 | 15 |
|---------|---|-----|----|-----|----|-----|
| 6 | 1 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0.6 | 0 | 0.2 | 0 | 0.2 |
| 10 | 0 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 1 |

10:45

| from\to | 7 | 8 | 10 | 12 | 13 | 15 |
|---------|--------|--------|----|----|----|----|
| 6 | 0 | 0 | 1 | 0 | 0 | 0 |
| 7 | 0.3333 | 0.6667 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 1 |

11:00

| from\to | 7 | 8 | 10 | 11 | 12 | 15 |
|---------|-----|-----|----|----|----|----|
| 7 | 0 | 0 | 0 | 0 | 0 | 1 |
| 8 | 0.5 | 0.5 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 1 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 1 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 1 |

11:15

| from\to | 7 | 10 | 11 | 12 | 15 |
|---------|---|----|----|----|----|
| 7 | 0 | 0 | 1 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0 | 1 |

11:30

| from\to | 7 | 10 | 11 | 12 | 15 |
|---------|---|----|--------|----|--------|
| 7 | 1 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0.9565 | 0 | 0.0435 |
| 12 | 0 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 0 | 1 |

11:45

| from\to | 10 | 12 | 13 | 15 |
|---------|----|----|----|----|
| 7 | 0 | 0 | 0 | 1 |
| 10 | 1 | 0 | 0 | 0 |
| 11 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |

12:00

| from\to | 10 | 11 | 12 | 15 |
|---------|----|----|----|----|
| 10 | 1 | 0 | 0 | 0 |
| 12 | 0 | 1 | 0 | 0 |
| 13 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |

12:15

| from\to | 10 | 11 | 12 | 15 |
|---------|----|----|----|----|
| 10 | 1 | 0 | 0 | 0 |
| 11 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |

12:30

| from\to | 10 | 11 | 12 | 15 |
|---------|----|----|----|----|
| 10 | 1 | 0 | 0 | 0 |
| 11 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |

12:45

| from\to | 10 | 12 | 13 | 15 |
|---------|----|----|----|----|
| 10 | 1 | 0 | 0 | 0 |
| 11 | 0 | 1 | 0 | 0 |
| 12 | 0 | 0 | 1 | 0 |
| 15 | 0 | 0 | 0 | 1 |