Shaping the Next Generation Air Transportation System with an Airspace Planning and Collaborative Decision Making Model

Justin M. Hill

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Industrial and Systems Engineering

Hanif D. Sherali, Chair
Ebru K. Bish
C. Patrick Koelling
Antonio A. Trani

September 8, 2009
Blacksburg, Virginia

Keywords: Air traffic management, flight plan generation, restricted shortest path problems, airspace sectorization, dynamic airspace configuration

Shaping the Next Generation Air Transportation System with an Airspace Planning and Collaborative Decision Making Model

Justin M. Hill

(ABSTRACT)

This dissertation contributes to the ongoing national project concerning the *Next Generation Air Transportation System* (NextGen) that endeavors, in particular, to reshape the management of air traffic in the continental United States. Our work is part of this effort and mainly concerns modeling and algorithmic enhancements to the Airspace Planning and Collaborative Decision-Making Model (APCDM).

First, we augment the APCDM to study an *Airspace Flow Program* (AFP) in the context of weather-related disruptions. The proposed model selects among alternative flight plans for the affected flights while simultaneously (a) integrating slot-exchange mechanisms induced by multiple Ground Delay Programs (GDPs) to permit airlines to improve flight efficiencies through a mediated bartering of assigned slots, and (b) considering issues related to sector workloads, airspace conflicts, as well as overall equity concerns among the involved airlines in regard to accepted slot trades and flight plans. More specifically, the APCDM is enhanced to include the following:

a. The revised model accommodates continuing flights, where some flight cannot depart until a prerequisite flight has arrived. Such a situation arises, for example, when the same aircraft will be used for the departing flight.

b. We model a slot-exchange mechanism to accommodate flights being involved in multiple trade offers, and to permit slot trades at multiple GDP airports (whence the flight connection constraints become especially relevant). We also model flight cancelations whereby, if a flight assigned to a particular slot is canceled, the corresponding vacated slot would be made available for use in the slot-exchange process.

c. Alternative equity concepts are presented, which more accurately reflect the measures used by the airlines.

d. A reduced variant of the APCDM, referred to as **APCDM-Light**, is also developed. This model serves as a fast-running version of APCDM to be used for quick-turn analyses, where the level of modeling detail, as well as data requirements, are reduced to focus only on certain key elements of the problem.

e. As an alternative for handling large-scale instances of APCDM more effectively, we present a *sequential variable fixing heuristic* (SFH). The list of flights is first partitioned into suitable subsets. For the first subset, the corresponding decision variables are constrained to be binary-valued (which is the default for these decision variables), while the other variables are allowed to vary continuously between 0 and 1. If the resulting solution to this relaxed model is integral, the algorithm terminates. Otherwise, the binary variables are fixed to their currently prescribed values and another subset of variables is designated to be binary constrained. The process repeats until an integer solution is found or the heuristic encounters infeasibility.

f. We experiment with using the APCDM model in a *dynamic, rolling-horizon framework*, where we apply the model on some periodic basis (e.g., hourly), and where each sequential run of the model has certain flight plan selections that are fixed (such as flights that are already airborne), while we consider the selection among alternative flight plans for other imminent flights in a look-ahead horizon (e.g., two hours).

These enhancements allow us to significantly expand the functionality of the original APCDM model. We test the revised model and its variants using realistic data derived from the *Enhanced Traffic Management System* (ETMS) provided by the *Federal Aviation Administration* (FAA). One of the new equity methods, which is based on average delay per passenger (or weighted average delay per flight), turns out to be a particularly robust way to model equity considerations in conjunction with sector workloads, conflict resolution,

and slot-exchanges. With this equity method, we were able to solve large problem instances (1,000 flights) within 30 seconds on average using a 1% optimality tolerance. The model also produced comparable solutions within about 20 seconds on average using the Sequential Fixing Heuristic (SFH). The actual solutions obtained for these largest problem instances were well within 1% of the best known solution. Furthermore, our computations revealed that APCDM-Light can be readily optimized to a 0.01% tolerance within about 5 seconds on average for the 1,000 flight problems. Thus, the augmented APCDM model offers a viable tool that can be used for tactical air traffic management purposes as an airspace flow program (particularly, APCDM-Light), as well as for strategic applications to study the impact of different types of trade restrictions, collaboration policies, equity concepts, and airspace sectorizations.

The modeling of slot ownership in the APCDM motivates another problem: that of generating detoured flight plans that must arrive at a particular slot time under severe convective weather conditions. This leads to a particular class of network flow problems that seeks a shortest path, if it exists, between a source node and a destination node in a connected digraph $G(N, A)$, such that we arrive at the destination at a specified time while leaving the source no earlier than a lower bounding time, and where the availability of each network link is time-dependent in the sense that it can be traversed only during specified intervals of time. We refer to this problem as the *reverse time-restricted shortest path problem* (RTSP). We show that RTSP is NP-hard in general and propose a dynamic programming algorithm for finding an optimal solution in pseudo-polynomial time. Moreover, under a special regularity condition, we prove that the problem is polynomially solvable with a complexity of order $O(|N||A|)$. Computational results using real flight generation test cases as well as random simulated problems are presented to demonstrate the efficiency of the proposed solution procedures.

The current airspace configuration consists of sectors that have evolved over time based on historical traffic flow patterns. Kopardekar et al. (2007) note that, given the current airspace configuration, some air traffic controller resources are likely under-utilized, and

they also point out that the current configuration limits flexibility. Moreover, under the free-flight concept, which advocates a relaxation of waypoint traversals in favor of wind-optimized trajectories, the current airspace configuration will not likely be compatible with future air traffic flow patterns. Accordingly, one of the goals for the *NextGen Air Transportation System* includes redesigning the airspace to increase its capacity and flexibility. With this motivation, we present several methods for defining sectors within the *National Airspace System* (NAS) based on a measure of sector workload. Specifically, given a convex polygon in two-dimensions and a set of weighted grid points within the region encompassed by the polygon, we present several mixed-integer-programming-based algorithms to generate a plane (or line) bisecting the region such that the total weight distribution on either side of the plane is relatively balanced. This process generates two new polygons, which are in turn bisected until some target number of regions is reached. The motivation for these algorithms is to dynamically reconfigure airspace sectors to balance predicted air-traffic controller workload. We frame the problem in the context of airspace design, and then present and compare four algorithmic variants for solving these problems. We also discuss how to accommodate monitoring, conflict resolution, and inter-sector coordination workloads to appropriately define grid point weights and to conduct the partitioning process in this context. The proposed methodology is illustrated using a basic example to assess the overall effect of each algorithm and to provide insights into their relative computational efficiency and the quality of solutions produced. A particular competitive algorithmic variant is then used to configure a region of airspace over the U.S. using realistic flight data.

The development of the APCDM is part of an ongoing *NextGen* research project, which envisages the sequential use of a variety of models pertaining to three tiers. The *Tier 1* models are conceived to be more strategic in scope and attempt to identify potential problematic areas, e.g., areas of congestion resulting from a severe convective weather system over a given time-frame, and provide aggregate measures of sector workloads and delays. The affected flow constrained areas (FCAs) highlighted by the results from these *Tier 1* models would then be analyzed by more detailed *Tier 2* models, such as APCDM, which

consider more specific alternative flight plan trajectories through the different sectors along with related sector workload, aircraft conflict, and airline equity issues. Finally, *Tier 3* models are being developed to dynamically examine smaller-scaled, localized fast-response readjustments in air traffic flows within the time-frame of about an hour prior to departure (e.g., to take advantage of a break in the convective weather system). The APCDM is flexible, and perhaps unique, in that it can be used effectively in all three tiers. Moreover, as a strategic tool, analysts could use the APCDM to evaluate the suitability of potential airspace sectorization strategies, for example, as well as identify potential capacity shortfalls under any given sector configuration.

This research has been supported by the National Aeronautics and Space Administration (NASA) under Grant Number NNA06CN21A (Subcontract Z627302).

Dedication

For Aaron.

Acknowledgments

First I would like to thank my advisor and committee chair, Dr. Hanif Sherali. His wisdom, guidance, encouragement, and support were invaluable, and his dedication and work-ethic can serve as examples for us all.

I am also grateful for the personal and professional insights offered by the rest of my committee. Drs. Ebru Bish, Patrick Koelling, and Antonio Trani each approach the important things in their lives with such enthusiasm and passion that I could not help but grow from my experiences with them.

Thanks also to the helpful and friendly staff in the Grado Department of Industrial & Systems Engineering, especially Kim Ooms and Hannah Swiger. It seems like one of them had the answer to every question I asked about "how things work" at Virginia Tech.

I offer my heartfelt thanks to my fellow students. In particular, Brian Lunday and his wife Erin were always quick to offer assistance or a listening ear, and Evrim Dalkiran was always there to discuss, well, anything! I wish all of you the best life has to offer.

I am deeply indebted to my wife, Jennifer. Her love and understanding during my work on this dissertation were beyond compare. Every day I am in awe that we found each other, and I could not imagine spending my life with anyone else.

Lastly, I would like to express my sincerest gratitude to all of the others who helped me during my years here at Virginia Tech. You are numerous and wonderful, and I hope that the future is a happy one for you all.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

By 2025, U.S. air traffic is predicted to increase two- to three-fold, and modernizing the air traffic control system is a major priority for the United States. To that end, the Joint Planning and Development Office (JPDO) was established in 2003 to coordinate the effort to develop the Next Generation Air Transportation System (NextGen) (Public Law 108-176 2003). NextGen is a collaborative effort across the Departments of Transportation, Defense, Homeland Security, Commerce, FAA, NASA, and the White House Office of Science and Technology Policy. Additionally, researchers at several institutions around the country are developing tools for use in NextGen. One such tool is the *Airspace Planning and Collaborative Decision-Making Model* (APCDM), originally proposed by Sherali, Hobeika and Kangwalklai (2003).

The U.S. National Airspace System (NAS) is a complex system in which thousands of people and pieces of equipment interact each day to ensure that 50,000 flights and 1.8 million passengers are managed safely and efficiently. The NAS currently involves approximately 14,500 air traffic controllers, 4,500 aviation safety inspectors, and 5,800 technicians, who operate and maintain services at more than 19,000 airports and 600 air traffic control facilities (FAA 2009). Moreover, the Federal Aviation Administration (FAA) and others in the aviation industry forecast that air traffic operations will increase 150 to 250 percent

over the next two decades (NASA 2006), which implies a significant deficit in existing and planned capacity.

NASA researchers note that there are severe limits on operational flexibility and overall capacity in today's air traffic management system because of the "*non-integrated distributed control paradigm*" and "*the cognitive human limitation that restricts the number of aircraft an individual air traffic controller can handle.*" As such, they call for a movement towards "*full automation*" for tactical separation decisions so that humans can concentrate on more strategic decision-making processes (NASA 2006).

To that end, the U.S. government is proposing approximately $800 million in funds for fiscal year 2010 to support the continuing development of the *Next Generation Air Transportation System* (**NGATS**, or more recently, referred to as **NextGen**), calling NextGen a "*long-term effort to improve the efficiency, safety, and capacity of the air traffic control system*" (OMB 2009). These three elements – efficiency, safety, and capacity – constitute the focal basis of the APCDM model.

1.1 Scope of Research

The development of the *NextGen Air Transportation System* is part of an ongoing collaborative effort between researchers at the *Massachusetts Institute of Technology*, the *University of California at Berkeley*, the *University of Maryland*, *Metron Aviation*, and *Virginia Tech*, with the goal of developing tools to design and implement a future air traffic management system. The overall vision is to develop a variety of models pertaining to three tiers. The *Tier 1* models are macroscopic in scope and are meant to be implemented at a national level for longer term planning purposes. The purpose of such a model is to attempt to identify potential problematic areas, e.g., areas of congestion resulting from a severe convective weather system over a given time-frame, and to provide aggregate measures of sector workloads and delays. Such *Tier 1* models would therefore identify potential problem ares where further

analysis might be necessary, which in turn would be conducted by more detailed *Tier 2* models, such as APCDM, which consider more specific alternative flight plan trajectories through the different sectors along with related sector workload, aircraft conflict, and airline equity issues. These *Tier 2* models are intended to provide analysis at a mesoscopic level, providing guidance at a flight-specific level while considering a relevant subsection of the NAS over a suitable time-window (typically, 2 to 3 hours). Finally, *Tier 3* models are being developed to dynamically examine smaller-scaled, localized fast-response readjustments in air traffic flows within the time-frame of about an hour prior to departure (e.g., to take advantage of a break in the convective weather system). These microscopic models will manage more intricate operations such as takeoffs and landings in a terminal area that is affected by severe weather.

The APCDM, initially conceived and further augmented by Sherali, Staats and Trani (2003, 2006), is a mixed-integer programming model that is intended to manage mesoscopic-level air traffic decisions within the NAS. Given a set of potential trajectories for each flight, the objective of the APCDM is to select an optimal set of flights subject to workload, safety, and equity considerations. The objective function is the overall system cost, which includes penalties for fuel, delay, and flight cancelations, as well as penalties designed to enforce an equitable solution with regard to each airline, and to maintain ATC workload measures at manageable levels.

In this dissertation, we propose and evaluate several important extensions and modifications to the APCDM, as well as investigate various effective solution methodologies for optimizing APCDM. More specifically, the goals of this dissertation are three-fold:

First, we enhance the APCDM to accommodate continuing and connecting flights. We also model a slot-exchange mechanism (McCrea 2006) to manage the trading of arrival slots at an airport at which a *Ground Delay Program* (GDP) has been invoked. The revised model includes the following capabilities with regard to slot exchanges: flights may be involved in multiple trade offers; slot trades can occur at multiple GDP airports; and the ar-

rival slots for cancelled flights are made available for exchange. Alternative equity concepts are additionally presented, which are more in line with the decision-making processes of airline companies. To provide the capability for the APCDM to be more flexible when modeling certain scenarios, we propose a model variant called **APCDM-Light**, which enables reducing the level of detail in situations when high fidelity is unnecessary, while maintaining the ability, through the full APCDM, to model air traffic operations in more detail where appropriate. We also make available a suite of solution methodologies to further the ability to use APCDM in various situations. These include traditional mixed-integer optimization methods and heuristic procedures, as well as implementing APCDM in a dynamic rolling-horizon framework that might be applied to better accommodate lengthy scenarios. The APCDM was designed for both strategic and tactical purposes, and we envisage that this additional functionality will enable its use in a wide variety of analytical contexts.

Second, we analyze an underlying open problem that arises in the context of generating flight trajectories. A severe convective weather system might render certain routes unusable for a given period of time. As such, we consider a particular class of network flow problems that seeks a shortest path, if it exists, between a source node s and a destination node d in a connected digraph, such that we arrive at node d at a specified time τ, while leaving node s no earlier than a lower bounding time LB, and where the availability of each network link is time-dependent in the sense that it can be traversed only during specified intervals of time. We refer to this problem as the *reverse time-restricted shortest path* problem (RTSP). The problem is shown to be NP-hard in general, and we develop a pseudo-polynomial time dynamic programming algorithm to solve it. We also exhibit that this problem is polynomially solvable under a special regularity condition.

Finally, we study a strategic issue related to airspace design and dynamic resectorization. The current airspace configuration is both inefficient and inadequate in capacity to satisfy future forecasted demand. Moreover, the lack of flexibility with the current design impedes automated air traffic control paradigms such as free-flight, whereby flight plans adopt optimized trajectories in lieu of traditional waypoint traversals. We consider a sub-region

of the national airspace represented as a polygon in 2-D space, onto which we superimpose a set of grid points in which each grid point has an associated weight that represents an aggregate ATC workload measure for some neighborhood about the point. We then begin generating sectors by constructing a *separating plane* that partitions the given polygonal region in a manner that balances the sum of weights assigned to either half-space. Then, at each subsequent step, we select the sub-polygon that has the largest total weight assigned to it, and repeat the foregoing bisection process, continuing this sequential partitioning until we have produced the required number of sectors. We delineate four algorithmic variants using this basic concept, and we describe a method for determining grid point weights as well as accommodating inter-sector coordination workload in this partitioning process. The proposed methodology is illustrated using a basic example to assess the overall effect of each algorithm and to provide insights into their relative computational efficiency and the quality of solutions produced. A particular competitive algorithmic variant is then used to configure a region of airspace over the U.S. using realistic flight data.

1.2 Summary of Contributions

This dissertation makes the following specific contributions in further enhancing the APCDM model and its components:

1. We develop an algorithm for solving a *reverse time-restricted shortest path* (RTSP) problem that arises when generating flight plans under severe convective weather conditions. We show that this problem is NP-hard in general, but that it can be solved in polynomial time under a special regularity condition. Moreover, we design a dynamic programming algorithm that solves RTSP in pseudo-polynomial time in general. Computational results, using both random problems and certain real flight path generation instances, are provided.

2. We augment the APCDM to accommodate multiple airport Ground Delay Programs

(GDPs), which are imposed to force flights to incur delay on the ground, where the passengers and crew are safe, rather than in the air. Additionally, we propose slot-exchange mechanisms to allow for inter-airline trading of arrival slots as well as the possibility that, if a flight assigned to a particular slot is cancelled, the corresponding slot would be made available for use in the slot-exchange process. We evaluate the efficacy of these enhancements with respect to solution time and several different solution quality metrics.

3. We enhance the scope of APCDM by incorporating the ability to model continuing flights, in which a flight cannot depart unless a designated prerequisite flight has arrived. These situations occur regularly when an airline uses the same aircraft for both flights. Modeling continuing flights becomes especially important when, for example, there exist multiple GDPs and delays are propagating through the system. Failing to account for these circumstances would yield ineffective solutions.

4. We describe a reduced variant of the APCDM, which we refer to as APCDM-Light, where the objective function retains the fuel, delay, and cancellation costs, along with certain penalties associated with solution equity, and where the model complexity is reduced by eliminating the consideration of detailed aspects of ATC workload restrictions, while maintaining the fundamental capacity and safety representations. APCDM-Light serves as a fast-running version of APCDM to be used for quick-turn analyses, where the level of modeling detail as well as data requirements are reduced to focus only on certain key elements of the underlying problem.

5. To handle large-sized problems that might exceed the algorithmic capability to solve APCDM to opimality, we design a *sequential variable fixing heuristic* (SFH) in which the APCDM is optimized by partitioning flights into suitable subsets. The decision variables corresponding to each subset considered in sequence are constrained to be binary-valued (which is the default for these decision variables), while the other variables are allowed to vary continuously between 0 and 1. If the resulting solution is

integral, the algorithm terminates. Otherwise, the binary variables are fixed to their currently prescribed values and another subset of variables is designated to be binary constrained. This process repeats until an integer solution is found, or the heuristic encounters infeasibility. We provide details for designing such a heuristic, and present computational results to demonstrate its effectiveness in practice.

6. We study the integration of APCDM as a *Tier-2* model within the *NextGen* project by designing its implementation in a *dynamic, rolling-horizon framework*. Here, we apply the model on some periodic basis (e.g., hourly), where each sequential run of the model fixes certain flight plan selections (e.g., airborne flights), while searching among alternative flight plans for other flights that overlap a designated look-ahead horizon (e.g., two hours). A subset of new flight plans corresponding to certain imminent flights as prescribed by the model output are then chosen for implementation, and the horizon is advanced for the next run with updated information. This framework additionally permits air traffic managers to optimize larger problem instances by focusing on more imminent routing decisions, while incorporating a sufficient look-ahead time-window to accommodate a subsequent set of decisions that might interact with, and thereby influence, the currently relevant decisions.

7. Finally, we investigate the strategic issue of dynamically redefining sector boundaries to conform with existing air traffic patterns in order to more equitably balance sector workloads, and thereby enhance the efficiency of air traffic flow patterns. In this context, we present a process for partitioning regions of airspace represented as polygons in two dimensions, and within each such polygon, several grid points are used to represent air traffic controller workload at various locations. We present four mixed-integer-programming-based algorithms to generate a plane (or line) bisecting the region such that the total weight distribution on either side of the plane is relatively balanced. This process generates two new polygons, which are in turn bisected until some target number of regions is reached. We also discuss how to accommodate monitoring and

conflict resolution workloads using output from the APCDM to appropriately define grid point weights, and how to incorporate inter-sector coordination workloads into the partitioning process. The proposed methodology is illustrated using a basic example to assess the overall effect of each algorithm and to provide insights into their relative computational efficiency and the quality of solutions produced. A particular competitive algorithmic variant is then used to configure a region of airspace over the U.S. using realistic flight data.

1.3 Organization of Dissertation

The remainder of this dissertation is organized as follows. In Chapter 2 we review the literature on air traffic management models, shortest path problems, and methods for airspace sectorization. This chapter also includes a review of the APCDM.

In Chapter 3 we describe an algorithm for solving the reverse time-restricted shortest path problem (RTSP) and show that, while the problem is NP-hard in general, it is solvable in pseudo-polynomial time and, in particular, under a certain regularity condition, we can solve the problem in polynomial time. We then provide computational results using the algorithm to solve both random and real flight generation instances of Problem RTSP.

Chapter 4 details several enhancements and modifications to the APCDM. We introduce two new sets of constraints for the APCDM: one that enables the modeling of continuing and connecting flights and another that permits *at-most, at-least* (AMAL) types of slot-exchanges, wherein flights might be involved in multiple exchange offers, including possible flight cancelations, and where there might exist multiple GDP airports. We also design a more practice-oriented APCDM-Light model to be used, for example, in the context of an *Airspace Flow Program* (AFP), and we compare its utility with that of the original APCDM using random problem instances.

Furthermore, in order to facilitate the solution of practical large-scale instances using

all the features inherent within the full APCDM model, we design a *sequential fixing heuristic* (SFH) in Chapter 5, and empirically study the resulting quality of solutions and the effort required in comparison with APCDM. We also implement the APCDM in a dynamic, rolling-horizon framework, and investigate impacts of such a policy on solution times and its ability to evolve solutions based on dynamically varying, uncertain weather patterns.

In Chapter 6 we study the strategic problem of airspace resectorization or dynamic airspace configuration. A fundamental methodology is developed for partitioning a region of airspace based on certain ATC workload measures, and four algorithmic variants are then presented using this basic concept. We also discuss how to accommodate monitoring, conflict resolution, and inter-sector coordination workloads to appropriately define grid point weights and to conduct the partitioning process in this context. The proposed methodology is illustrated using a basic example to assess the overall effect of each algorithm and to provide insights into their relative computational efficiency and the quality of solutions produced. A particular competitive algorithmic variant is then used to configure a region of airspace over the U.S. using realistic flight data.

Finally, in Chapter 7, we summarize our findings and recommend several directions for future research.

Chapter 2

Literature Review

In this chapter, we provide a brief review of the existing literature that pertains to the subject matter herein. We begin by discussing the evolution of air traffic management models and how such models have grown to incorporate various air traffic management initiatives over time. We next discuss shortest path problems and their time-dependent variants as well as algorithms that have been developed to solve such problems. Finally, various methods for measuring ATC workload are reviewed, along with the literature related to airspace sectorization.

2.1 Air Traffic Management Models

Amadeo Odoni conceptualized the problem of scheduling flights to minimize the costs associated with a congested airspace (Odoni 1987). He noted that published research concerning similar topics was sparse. However, a review of the literature since reveals several models that have been proposed to address specific elements of the flight scheduling and routing problems. We review below some contemporary air traffic management policies such as ground delay programs (GDPs) and airspace flow programs (AFPs), as well as recent air

traffic flow management models, including the APCDM.

2.1.1 Ground Delay Programs

Vossen and Ball (2006b) provide an excellent description of GDPs, which are issued by the FAA "*to manage temporary reductions in an airport's arrival capacity.*" They further note that, "*The underlying motivation is that, as long as a delay is unavoidable, it is both safer and less costly for the flight to absorb this delay on the ground.*" In a related *Ground-Holding Problem* (**GHP**) one chooses the flights to delay so that the overall cost of airline delays is minimal and airport arrival and departure capacities are not exceeded. The simplest variant of the GHP is the *Single-Airport Ground-Holding Problem* (SAGHP) in which flights are assigned delays to flights a single airport. Terrab and Odoni (1993) solved the deterministic SAGHP as a network flow problem, and Richetta and Odoni (1993, 1994) solved the stochastic SAGHP using stochastic programming.

An obvious extension to the SAGHP is the *Multi-Airport Ground-Holding Problem* (MAGHP), wherein flights in a network of airports are assigned delays. Vranas et al. (1994a,b) formulated the deterministic MAGHP as a binary integer programming problem and Terrab and Paulose (1992) considered the MAGHP as a stochastic programming problem. Both the SAGHP and MAGHP can be solved as special cases of the model due to Bertsimas and Stock Patterson (1998), which is discussed in Section 2.1.3.

2.1.2 Airspace Flow Programs

Another important air traffic management initiative sponsored by the FAA in 2006 is that of *Airspace Flow Programs* (AFPs) (see FAA 2006 and Sud et al. 2009). Here, due to a severe convective weather pattern, certain affected *flow constrained areas* (FCAs) or sub-regions of the airspace are identified, and flights passing through these FCAs over a specified duration are given a controlled departure time and are efficiently routed in an equitable fashion,

subject to FCA capacity constraints in terms of the maximum number of flights managed per hour.

Very little of the published research focuses exclusively on AFPs. Rather, AFPs are most often regarded as part of a larger air traffic management problem. For example, Krozel et al. (2006) proposed a heuristic approach to manage an AFP, including flight scheduling, route selection, and airborne holding, as well as an estimation of FCA capacities based on weather forecasts. Mukherjee and Hansen (2009) presented a stochastic integer programming model for managing traffic that is inbound to a particular airport whose capacity is adversely affected by weather, thus creating an AFP. They conclude that making rerouting decisions dynamically allows traffic managers to exploit updated weather information in this process. Abdelghany et al. (2007) developed heuristic methods for a single airline to decide which flights to reroute and how to govern slot trades involving the airline's own flights.

Wanke and Greenbaum (2007) designed a simulation model to study the impact of uncertainty within the air traffic management process. Air traffic managers attempt to ensure that the demand (number of aircraft) within a sector does not exceed the sector capacity, where this capacity can be reduced by various events, particularly due to convective weather (i.e., thunderstorms). This is precisely the scenario created by an AFP. Wanke and Greenbaum noted that congestion control is "*primarily through manual processes, relying on experience and limited traffic prediction data*," and they proposed a quantitative strategy for congestion management. The authors further noted that a number of factors that influence such decisions, such as weather forecasts and traffic demand, are stochastic. As such, their proposed strategy for traffic management attempts to account for this uncertainty.

The simulation model of Wanke and Greenbaum (2007) addresses a series of decisions. Once the air traffic manager decides that congestion at some time T is likely, the model focuses on decisions at several points in time leading up to T. At each decision point, the model simulates three different courses of action to relieve the congestion: 1) do nothing; 2) partially resolve the congestion, or 3) fully resolve the congestion. The same three decisions

occur at each decision point up to the final decision (by which time, all congestion must be resolved). This results in a decision tree, which the authors use to compute the expected system congestion, the expected system delay, the number of flights rerouted, and other metrics in order to prescribe a best course of action. Embedded within this model are other simulation sub-models for generating traffic demand and studying the outcome of the rerouting decisions. There is also an algorithm for the congestion resolution process in which the flights are prioritized and considered in order (airborne flights with earlier landing times are given higher priority). If the flight's currently scheduled path does not cause the probability of congestion to exceed a maximum allowable level, the flight is not altered. If the maximum congestion probability is exceeded, an alternative route is chosen from a list of predetermined routes. The algorithm then considers the next flight in the prioritized list. The congestion resolution algorithm proposed by Wanke and Greenbaum (2007) is somewhat unsophisticated. The "greedy" manner in which the highest priority flights are first rerouted can easily lead to a suboptimal solution. For example, if some high-priority flight is particularly slow-moving, it could stay in a sector for a significant amount of time, possibly causing many other lower-priority flights to be rerouted even though it might be possible to safely reroute this single flight simultaneously with only a few of the others. A more sophisticated algorithm could ensure that high-priority flights are heeded while being more concerned with the performance of the overall system. Results of a test case using two intermediate decision points indicated that the best course of action was to partially resolve the congestion at the first decision point, do nothing at the second, and wait until the final decision to resolve any remaining congestion. This strategy resulted in the least "Total Positive Delay" per flight while rerouting relatively few flights. The conclusions reached were similar to those of Mukherjee and Hansen (2009) in that both studies experimentally realized benefits by effectively postponing decisions.

2.1.3 The Air Traffic Flow Management Problem

The *Air Traffic Flow Management Problem* (TFMP) is a more general version of the GHP in which one also considers airborne delay. Typically, FCA capacity is not addressed in models designed to solve the TFMP. In the TFMP, airborne delay is often modeled by assigning an average speed to the flight or by including nodes in the network that represent delay points. Several researchers have proposed models for the TFMP (see Lindsay et al. (1993) and Helme (1992)), but the model most often referenced in the literature is due to Bertsimas and Stock Patterson (1998).

Bertsimas and Stock Patterson (1998) modeled the TFMP as a binary integer programming problem where the objective is to minimize the total cost of air and ground delays. The minimal cost is constrained by the arrival and departure capacities at each airport, and the capacity within each flight sector. Other constraints are included to ensure connectivity of the airports, sectors, and the flow of time within the model. The flight plan trajectories are assumed to be fixed and conflict-free, and no equity considerations are addressed. Their model is noteworthy in that the linear relaxation is particularly tight and often returns integer solutions; thus, the model runs quickly and is favorable from a computational standpoint.

Bertsimas and Stock Patterson (1998) also defined the *Air Traffic Flow Management Rerouting Problem* (TFMRP) as the TFMP with the additional flexibility of assigning each flight to a different route. They suggested a modification of the TFMP model to incorporate rerouting; however, they noted that additional route options will increase the size of the problem by a factor of the number of available routes for each flight. They also recommended a modification in which each flight decides at the time it enters each sector in its route the particular sector to enter next. The authors provided no experimental results for either modification, but relegated this for further investigation.

In a follow-on study, Bertsimas and Stock Patterson (2000) addressed the TFMRP for a single airline using a dynamic network flow approach. The authors reduced the dimension-

ality of the problem by using variables to represent the overall flow between airports rather than to model individual flights. The problem was modeled as a dynamic, multicommodity, integer network flow problem with side-constraints, and was solved using Lagrangian relaxation to generate aggregate flows. The aggregate flows were then decomposed into groups of flight paths using a heuristic, and individual flights were assigned to paths using an integer program. The authors noted that the solutions generated are near-optimal (within 1% of the derived lower bounds), and they proposed a methodology for using this construct to consider the TFMRP for multiple airlines.

Kaufhold et al. (2007) formulated a binary integer programming model to solve the *Generalized Air Traffic Flow Management Problem* (GATFMP), a TFMP combined with a runway assignment problem. The airport capacity was modeled using a resource counter network that tracks arrivals, departures, and movements (arrivals plus departures) separately. The resulting model allows for some flexibility within the capacity constraints by constraining the flow through the capacity resource counter network. For example, a user can set limits on individual runways and on total movements in the airspace without incorporating additional variables. The proposed model was solved using a branch-cut-and-price algorithm, and the solution obtained was prescribed to serve as a starting point for tactical planners at an airport to formulate actual decisions.

ATM in Europe is certainly a case in which necessity breeds invention. Lulli and Odoni (2007) noted that the airspace capacity in and around Europe is more constrained because of airspace ownership and due to major airports being located in relatively close proximity to one another. The result is that, in addition to capacity constraints around airports, any model for the EU TFMP must account for en route sector capacity more carefully than models for the United States. Lulli and Odoni presented extensions to their earlier deterministic model (Lulli and Odoni 2006). Their model is essentially a binary integer program (with non-binary artificial variables) that minimizes a linear combination of the costs of airborne and ground delays, and the authors pointed out that it can be considered as a macroscopic version of the model due to Bertsimas and Stock Patterson (1998).

The model of Lulli and Odoni (2007) accounts for a nonlinear airborne holding cost via a piecewise linear approximation driven by artificial variables in the constraints. The authors indicated that this modification increases the computational complexity, but they noted that run-times are not an issue for their application. Their model also attempts to ensure that delays are equitable by using superlinear cost coefficients that penalize tardiness. These cost coefficients institute a preference for assigning one unit of delay to each of two flights rather than assigning two units of delay to a single flight. The formulation remains linear, but the authors were able to account for two significant nonlinearities. Lulli and Odoni used their model to demonstrate that straightforward algorithms, such as the Ration-by-Schedule algorithm (see, e.g., Vossen and Ball (2006a) for a description), might lead to highly cost-inefficient solutions in the European version of the TFMP. Their results also indicated a tradeoff between efficient solutions (minimal cost) and equitable solutions (shared delay).

As another modification of the earlier work presented in Bertsimas and Stock Patterson (1998, 2000), Bertsimas et al. (2009), proposed an alternative model to solve the TFMP. This model considers both ground and airborne delay, speed control, and rerouting and turns out to be very computationally efficient, even for large-scale problem instances involving 3000 flights. The model seeks to minimize delay costs subject to airport capacity, airspace sector capacity, routing restrictions, and flight time limits. Similar to its predecessor, sector occupancy is computed at discrete points in time, and the routes defined by sector entry and exit times and implied speeds are determined by the time spent in each sector. However, the trajectories implied by the model solution might not be contiguous or realizable, which is an intended compromise. Indeed, this model is part of ongoing research related to the *NextGen Air Transportation System*, which envisages the sequential use of a variety of models pertaining to three tiers. The *Tier 1* models, such as the one described by Bertsimas et al. (2009), are conceived to be more strategic in scope and attempt to identify potential problematic areas, e.g., areas of congestion resulting from a severe convective weather system over a given time-frame, and provide aggregate measures of sector workloads

and delays. The affected flow constrained areas (FCAs) highlighted by the results from these *Tier 1* models would then be analyzed by more detailed *Tier 2* models, such as the APCDM, which consider more specific alternative flight plan trajectories through the different sectors along with related sector workload, aircraft conflict, and airline equity issues. Finally, *Tier 3* models are being developed to dynamically examine smaller-scaled, localized fast-response readjustments to air traffic flows within the time-frame of about an hour prior to departure (e.g., to take advantage of a break in the convective weather system). An example of a Tier 3 model is the aforementioned model by Mukherjee and Hansen (2009).

2.1.4 Airspace Planning and Collaborative Decision-Making Model

The Airspace Planning and Collaborative Decision-Making Model (APCDM), which was introduced by Sherali, Staats and Trani (2003) and further developed in Sherali, Staats and Trani (2006), is a large-scale mixed-integer programming model designed to enhance the management of air traffic in the National Airspace System (NAS). Given a set of potential 4-D (space-time) trajectories for each flight, the objective of the APCDM is to select an optimal set of flight plans subject to sector workload, collision safety, and airline equity considerations. McCrea (2006) provides a thorough review of the APCDM modeling framework. The discussion here is devoted to the enhancements discussed in McCrea et al. (2008), followed by an overview of the APCDM model.

McCrea et al. (2008) developed a tool for generating flight plans in the presence of a severe convective weather system. Given a set of reporting stations from the (continental) United States Model Output Statistics (MOS), the authors constructed weather-specific probability-nets that are dynamic with respect to time and space, and are defined as follows. The nodes of the probability-nets are the MOS reporting sites having associated point-by-point forecast probabilities. Connections between the MOS reporting sites form the links or strands within the probability-nets, and are constructed based upon a user-specified adjacency threshold, which is defined as the maximum allowable great circle distance between

any such pair of sites for which an interpolation of severe weather probabilities is approximately valid. When a flight plan traverses through a probability-net, its probability footprint corresponding to the 4-D time-space locations where the flight plan and the probability-net strands intersect is extracted.

Next, a flight-trajectory-grid network is superimposed upon the probability-net(s). Using the U.S. Navigational Aids (NAVAIDS) as the network nodes, alternative flight plans are generated pertaining to specified slot allocations or departure/arrival time specifications via a process of determining restricted, time-dependent shortest paths between the origin and destination airports such that the generated trajectory intersects no strand with an interpolated probability exceeding a specified threshold value. Note that whenever an arrival time-slot value is specified at a destination airport (e.g., a GDP-restricted airport), a reverse, time-dependent path needs to be generated that would accordingly determine the corresponding departure time at the origin airport. In this context, links have fixed travel times, but their availability is time-dependent based on the weather pattern and the specified threshold probability. A more detailed analysis and an algorithm for solving this specialized shortest path problem is provided in Chapter 3 of this dissertation. Observe also that this process automatically accommodates long-haul flights that would likely not be affected by the severe weather phenomenon due to its dissipation by the time the flight reaches the currently weather-prone region. Based on the probability footprint of the generated flight plan, and an estimation of delay distribution, McCrea et al. (2008) also designed a methodology for computing appropriate expected weather delays and related disruption factors for inclusion within the APCDM model.

McCrea (2006) also addressed certain equity-related modeling constructs and preliminary ideas for modeling some basic elements for considering slot-exchanges within the APCDM. These are developed further in more detail and modeled with additional considerations in Chapter 4.

We next present an overview of the APCDM model (Sherali, Staats and Trani 2003,

2006). The notation for the model and the mathematical formulation are given below:

Notation

(a) Index Sets:

- $s = 1, \ldots, S$: Sectors involved in the model analysis.
- $\alpha = 1, \ldots, \overline{\alpha}$: Airlines involved in the model analysis.
- $f = 1, \ldots, F$: Flights to be scheduled.
- $p \in P_f$: Alternative flight plans or surrogates for flight f.
- $P_{f0} = P_f \cup \{0\}$, with $p = 0$ representing the flight cancelation surrogate. (Note that canceling a flight is the prerogative of the particular airline and could be offered as an alternative in exchange for improving the status of other flights within the CDM framework. This surrogate ($p = 0$) is ascribed an inordinately high penalty if canceling the corresponding flight is not tendered as a possible option. Also, naturally, the surrogate $p = 0$ is not an option for airborne flights.)
- A_α: Set of flights belonging to airline α, for $\alpha = 1, \ldots, \overline{\alpha}$.

(b) Decision Variables:

(i) Principal Decision Variables:

- x_{fp}: Binary variable, which equals one if flight plan $p \in P_{f0}$ is selected for flight f, and zero otherwise, for $f = 1, \ldots, F$. (These are sometimes denoted by x_P or x_Q (etc.) where each P and Q (etc.) represent some actual flight plan combination (f, p) for $p \in P_f, f \in \{1, \ldots, F\}$.)

(ii) Auxiliary Decision Variables:

- n_s: Peak occupancy level (number of flights) for sector s.
- w_s: Average occupancy level (number of flights) for sector s.

- $y_{sn} \in [0,1], n = 0, \ldots, \overline{n}_s$: Convex combination weights attached to the breakpoints of a piecewise linear increasing convex penalty function, which represents the penalty ascribed to the difference between the peak and the average workload in sector s.
- z_{PQ}: Binary variable, which equals one whenever conflicting flight plans P and Q are selected. (It is assumed in the model description that $z_{PQ} \equiv z_{QP}$.)
- $d_\alpha(x)$: Relative performance ratio for airline α.
- $E_\alpha(x)$: Collaboration efficiency for airline α.
- $E_\alpha^{\text{equity}}(x)$: Collaboration equity for airline α.

  Note: The previous three functions of x are notationally designated as variables in the model formulation to ease representation and to facilitate structure:
- $x^{\text{equity}} \equiv \sum_{\alpha=1}^{\overline{\alpha}} \omega_\alpha \left| E_\alpha^{\text{equity}}(x) \right|$: ω-mean collaboration inequity.
- v_α: Variable used to represent the term $\left| E_\alpha^{\text{equity}}(x) \right|$ for airline α.

(c) Model Parameters and Coefficients:

- ω_α: The weight factor ascribed to each airline α (e.g., based on $|A_\alpha|$).
- c_{fp}: The cost to execute flight plan $p \in P_{f0}$ for flight f.
- $c_f^* = \min\{c_{fp} : p \in P_f\}$, for each flight $f = 1, \ldots, F$.
- H: The length of the time horizon under consideration (in minutes).
- t_{fp}^s: The length of time (in minutes) that flight plan $p \in P_f$ of flight f occupies sector s.
- Ω_s: The set of flight plans that occupy sector s during some subset of the time horizon.
- $\overline{n}_s$: The maximum allowable peak monitoring workload (simultaneous flight occupancies) in sector s.
- r_s: The maximum number of simultaneous conflict resolutions permitted to exist in sector s.

- γ_s: The airspace monitoring (penalty) cost factor for sector s, per unit average occupancy level workload.
- ψ_{sn}: The (penalty) cost assessed when peak monitoring workload in sector s exceeds the average workload by $n \in \{0, 1, \ldots, \overline{n}_s\}$.
- φ_{PQ}: The (penalty) cost ascribed to resolve an en-route conflict between flight plans P and Q.
- μ: The (penalty) cost factor associated with the equity terms in the objective function. (Following Sherali, Staats and Trani (2006), we take $\mu = 0.1\sum_{f=1}^{F} c_f^*$.)
- W_f: Relative priority weight attached to flight $f \in A_\alpha$ by airline α, where $\sum_{f \in A_\alpha} W_f = 1$, $W_f \geq 0$, $\forall f \in A_\alpha$.
- v^{equity}: Maximal limit imposed on x^{equity}, the ω-mean collaboration inequity.
- $E_{\max}^{\text{equity}}$: Maximal limit imposed on each weighted inequity $\omega_\alpha \left| E_\alpha^{\text{equity}}(x) \right|$, $\forall \alpha = 1, \ldots, \overline{\alpha}$. (As recommended in Sherali, Staats and Trani (2006), we take $E_{\max}^{\text{equity}} = 0.07/\overline{\alpha}$).
- $C_{si}, \forall i = 1, \ldots, I_s$: The maximal overlapping sets of occupying flight plans for sector s.
- $M_{sk}, \forall k = 1, \ldots, K_s$: The maximal overlapping sets of conflicting pairs of flight plans for sector s.

(d) Equity Measures:

The measure below, introduced in Sherali, Staats and Trani (2006), is indexed as Equity Method 1; other measures $d_\alpha^i(x)$ and $E_\alpha^i(x), i = 2, 3$ are proposed later in Chapter 4.

- $d_\alpha^1(x) = \dfrac{\sum_{f \in A_\alpha} \sum_{p \in P_{f0}} W_f c_{fp} x_{fp}}{\sum_{f \in A_\alpha} W_f c_f^*}, \forall \alpha = 1, \ldots, \overline{\alpha}.$
- $E_\alpha^1(x) = \dfrac{d_{\max}^1 - d_\alpha^1(x)}{d_{\max}^1 - 1}, \forall \alpha = 1, \ldots, \overline{\alpha}.$

- $d^1_{\max} = 1.2$.

APCDM Model:

Minimize

$$\sum_{f=1}^{F} \sum_{p \in P_{f0}} c_{fp} x_{fp} + \mu \left(\sum_{\alpha=1}^{\overline{\alpha}} \omega_\alpha \left[1 - E_\alpha(x) \right] + x^{\text{equity}} \right) + \sum_{s=1}^{S} \gamma_s w_s + \sum_{s=1}^{S} \sum_{n=0}^{\overline{n}_s} \psi_{sn} y_{sn} + \sum_{(P,Q) \in A} \varphi_{PQ} z_{PQ} \tag{2.1a}$$

subject to:

$$\sum_{p \in P_{f0}} x_{fp} = 1, \quad \forall f = 1, \ldots, F \tag{2.1b}$$

$$\sum_{(f,p) \in C_{si}} x_{fp} \leq n_s, \quad \forall i = 1, \ldots, I_s, s = 1, \ldots, S \tag{2.1c}$$

$$w_s = \frac{1}{H} \sum_{(f,p) \in \Omega_s} t^s_{fp} x_{fp}, \quad \forall s = 1, \ldots, S \tag{2.1d}$$

$$n_s - w_s = \sum_{n=0}^{\overline{n}_s} n y_{sn}, \quad \forall s = 1, \ldots, S \tag{2.1e}$$

$$\sum_{n=0}^{\overline{n}_s} y_{sn} = 1, \quad \forall s = 1, \ldots, S \tag{2.1f}$$

$$x_P + x_Q \leq 1, \quad \forall (P,Q) \in FC \tag{2.1g}$$

$$\sum_{(P,Q) \in M_{sk}} z_{PQ} \leq r_s, \quad \forall k = 1, \ldots, K_s, s = 1, \ldots, S \tag{2.1h}$$

$$x_P + x_Q - z_{PQ} \leq 1, \quad \forall (P,Q) \in A \tag{2.1i}$$

$$\sum_{Q \in J_{sk}(P)} z_{PQ} \leq r_s x_P, \quad \forall P \in N_{sk} : |J_{sk}(P)| \geq r_s + 1, \quad \forall k = 1, \ldots, K_s, s = 1, \ldots, S \tag{2.1j}$$

$$d_\alpha(x) \equiv d^i_\alpha(x) \text{ and } E_\alpha(x) \equiv E^i_\alpha(x) \text{ relationships, } i = 1 \text{ (or } i = 2, 3 \text{ as in Ch. 4)} \tag{2.1k}$$

$$E_{\alpha}^{\text{equity}}(x) = E_{\alpha}(x) - \left(\sum_{\alpha=1}^{\overline{\alpha}} \omega_{\alpha} E_{\alpha}(x) \right), \quad \forall \alpha = 1, \ldots, \overline{\alpha} \tag{2.1l}$$

$$v_{\alpha} \geq -E_{\alpha}^{\text{equity}}, \quad v_{\alpha} \geq E_{\alpha}^{\text{equity}}, \quad \forall \alpha = 1, \ldots, \overline{\alpha} \tag{2.1m}$$

$$x^{\text{equity}} = \sum_{\alpha=1}^{\overline{\alpha}} \omega_{\alpha} v_{\alpha} \tag{2.1n}$$

$$x_{fp} \text{ binary}, \forall p \in P_{f0}, f = 1, \ldots, F; \quad z_{PQ} \geq 0, \forall (P, Q) \in A;$$

$$y_{sn} \geq 0, \forall n = 0, \ldots, \overline{n}_s, s = 1, \ldots, S; \quad n_s \leq \overline{n}_s, \forall s = 1, \ldots, S;$$

$$E_{\alpha}(x) \geq E_{\min}^{\text{efficiency}}, \forall \alpha = 1, \ldots, \overline{\alpha}; \quad x^{\text{equity}} \leq v^{\text{equity}};$$

$$\text{and } \omega_{\alpha} v_{\alpha} \leq E_{\max}^{\text{equity}}, \forall \alpha = 1, \ldots, \overline{\alpha}. \tag{2.1o}$$

The first term in the objective function (2.1a) is the summation of the fuel, delay, and cancellation costs for the selected flight plans. The next term in (2.1a), associated with the parameter μ, imposes a penalty on the attained levels of the ω-mean collaboration inefficiency and the ω-mean collaboration inequity. Note that the first term in (2.1a) represents the total system cost, whereas the second term is dimensionless and includes weighting priority factors among airlines, as well as weighting priorities for flights pertaining to each individual airline. The third term in the objective function (2.1a) ascribes a penalty to the average sector workload. The fourth term penalizes the differential between the peak and average sector workloads according to a piecewise linear increasing convex function, and the final term penalizes the resolvable conflicts that will need to be addressed by the air traffic controllers.

Constraints (2.1b) require exactly one flight plan to be selected from among the set of available surrogates. The next four constraints are associated with sector occupancies, where (2.1c) limits the maximum simultaneous occupancy for each sector s, (2.1d) evaluates the average sector workloads, (2.1e) computes the peak-average workload differential, and (2.1f) requires the convex combination weights ascribed to the breakpoints of the peak-average workload differential penalty function to sum to one. All fatal conflicts are prohibited via

(2.1g). The conflict constraint formulation, described in detail in Sherali, Staats and Trani (2003), is expressed via (2.1h)-(2.1j), where (2.1h) and (2.1i) jointly ensure that no more than r_s resolvable conflicts coexist within sector s at any point in time, and (2.1j) further tightens this representation via a set of star-subgraph convex hull based valid inequalities. The (alternative) relative performance ratios and collaboration efficiencies are represented in (2.1k). Constraints (2.1l) compute the collaboration equity, and Constraints (2.1m) and (2.1n) provide linearized relationships for determining the ω-mean collaboration inequity. Finally, (2.1o) imposes the necessary logical and desired bounding conditions.

2.2 Shortest Path Problems

The classical shortest path problem involves finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized. Probably the most well-known algorithm for solving a shortest path problem was introduced by Dijkstra (1959), which determines the shortest path from a single node to all other nodes in the network, assuming non-negativity of the edge weights. Since then, a plethora of algorithms have been developed to solve this problem as well as many other variants (see Ahuja et al. 1993 and Bazaraa et al. 2004). One particular problem variant involves graphs in which the edge weights might change with respect to time. Such problems are traditionally referred to as time-dependent shortest path problems (TDSP).

Cooke and Halsey (1966) developed the first dynamic programming algorithm to solve the time-dependent shortest path problem (TDSP). Their algorithm determines the fastest travel time between two nodes based on a given starting time and with no waiting allowed. Dreyfus (1969) noted that the same problem could also be solved using Dijkstra's algorithm (Dijkstra 1959) on a time-expanded representation of the underlying network. Halpern (1977) incorporated additional rules to allow finite waiting times at nodes. Subsequently, several authors proved that applying Dijkstra's labeling scheme to the time-expanded net-

work is guaranteed to return an optimal solution only when the first-in-first-out (FIFO) condition holds (Ahn and Shin 1991, Kaufman and Smith 1993, Chabini 1998). It was Kaufman and Smith (1993) who formalized the definition of the FIFO condition, which basically states that flow entities cannot pass one another while traversing an edge.

Orda and Rom (1990) provided a more in-depth look at how waiting affects the TDSP by generalizing Cooke and Halsey's model to include arbitrary travel time functions, including those that do not meet the FIFO condition. They also considered several waiting policies: unlimited waiting allowed, waiting forbidden, and waiting allowed only at the source node. In addition, they developed efficient solution methods for the case with unlimited waiting, and showed that permitting waiting at the source node often allows for efficient solutions as well, and that shortest paths in non-FIFO networks might be non-simple and possibly non-concatenated. Sherali et al. (1998) showed that the latter problem is NP-hard even if only a single link in the network has such a non-FIFO, time-dependent delay structure.

Chabini (1998) discussed the differences between the *one-to-all* and the *all-to-one* versions of the problem and pointed out that the all-to-one version, in which the focus is on the destination rather than the origin, is sometimes more applicable. Chabini provided algorithms for the all-to-one and one-to-all versions, both with and without waiting policies, and noted that the variant of the problem without the FIFO assumption and with no waiting allowed at the nodes is particularly challenging.

Several authors have addressed different variations of the TDSP. These include finding multiple shortest paths (Ziliaskopoulos and Mahmassani 1993), optimal disjoint paths (Sherali et al. 1998), shortest paths when the path is constrained by arc labels (Sherali, Hobeika and Kangwalklai 2003), label-constrained shortest paths under approach-dependent travel times (Sherali, Jeenanunta and Hobeika 2006), shortest paths when the arc lengths are fuzzy (Ji et al. 2007), and stochastic time-dependent shortest paths (Hall 1986, Thomas and White III 2007).

The paper most similar to the present research is that by Daganzo (2002), who

considered the reversibility of the TDSP by addressing the problem of finding the latest departure time at which one can depart an origin so as to arrive at a specific destination at a given time. Link travel times were permitted to be time-dependent but the FIFO condition was assumed, and the departure time from the origin was not constrained. Daganzo showed that an algorithm that generates a shortest path on a strictly FIFO network will also generate a reverse shortest path. Daganzo also showed that an algorithm designed for solving any forward (reverse) TDSP also solves some conjugate reverse (forward) TDSP, regardless of whether the travel time function satisfies the FIFO assumption.

2.3 Air Traffic Controller Workload

Air traffic controller (ATC) workload is often a limiting factor with regard to the real-time control of air traffic. As such, any strategic or operational models for air traffic flow management should consider some suitable measure of ATC workload.

The Federal Aviation Administration (FAA 2008) currently uses Monitor Alert Parameters (MAP), which are based on the number of aircraft within a sector, as the primary indicator of ATC workload. Workload measures based on sector count are intuitive and easy to implement, and even though the FAA has supported the development of more sophisticated ATC workload metrics, research efforts focusing on aircraft count within a sector are ongoing. Lee (2005) used three different regression models – linear, exponential, and an S-curve – to fit perceived workload against sector count. The S-curve yielded the best fit in all of his experiments, implying a categorical relationship between ATC workload and sector count. Most of the current research, however, is based on more sophisticated means of measuring sector complexity.

Dynamic Density (DD) was first proposed in an RTCA Radio Technical Commission for Aeronautics report as an aggregate metric to pull together all the information essential for assessing air traffic complexity. In fact, Kopardekar and Magyarits (2003) listed 40

different factors that contribute to air traffic monitoring complexity, and presented results of a multi-year research initiative in which multiple organizations identified possible DD metrics. The authors then used the results from four previous DD studies as inputs in a linear regression model to measure complexity using DD variables. They developed such regression models of complexity for four Air Route Traffic Control Centers (ARTCCs), using data collected over two-minute intervals. A model composed of inputs unified from all four of the previous studies was able to account for 39% of the variability of the observed complexity data, whereas a model based simply on aircraft count only accounted for 23%. Kopardekar and Magyarits (2003) then performed a principal components analysis on the 23 significant variables from the unified regression model to identify 12 general factors that seem to contribute to complexity. The next step was to assess the efficacy of using DD data to *predict* future complexity (up to 120 minutes) using past data. A certain look-ahead model performed well, but not significantly better than a model based on the predicted aircraft count.

Loft et al. (2007) argue that, although empirically derived metrics such as DD are useful, we cannot accurately model ATC workload until we understand the controllers' strategies for processing tasks in a high-workload setting. To that end, Boag et al. (2006) used *relational complexity* to predict workload. For example, two aircraft traveling at the same altitude on the same vector at the same speed are essentially unrelated in the ATC's mind (relational complexity is zero), because if nothing changes, these two aircraft cannot conflict with one another. On the other hand, if two aircraft are approaching one another on merging paths at different speeds while one is climbing through the altitude at which the other is flying, the controller has to account for a potential conflict in multiple dimensions while accounting for a difference in speed. This is obviously a situation with a higher workload, and the relational complexity is higher as well. Boag et al. (2006) surveyed air traffic controllers, asking them to rank the complexity of various scenarios, and ran a regression model of complexity rankings against minimum separation criteria and relational complexity, and found that the resulting model accounts for 53% of the variance of the complexity rankings.

However, the experimental results pertain to a sector-based air traffic monitoring system and might not translate well to a sector-free system.

All of the research mentioned above assumes a sector-based structure similar to the one in which ATCs currently operate. However, the FAA is planning to move away from the presently defined sector-based partitioning of the NAS, and some researchers have attempted to address the problem of measuring congestion in a sector-free airspace. Yousefi and Donohue (2004) imposed a hexagonal, multi-layered grid onto the NAS and used an optimization model to create new sectors based on historical traffic observations. Their work seems to be in line with the concept of *dynamic resectorization* (Kopardekar et al. 2007), but they noted that their solution process is too slow to be implemented in real-time.

Ishutkina et al. (2005) assessed complexity by generating a smooth vector field based on a snapshot of the NAS. They used each aircraft's position, speed, and heading as inputs and formulated a linear programming model to determine if it is feasible to build a smooth vector field of aircraft velocities. If such a field is feasible, they argued, the complexity of the system is low. When no such field exists, they solved a problem to determine the minimal number of constraints that should be relaxed in order to create such a field. The authors asserted implicitly that this is related to ATC workload, but provided no experimental results to that effect.

Bilimoria and Lee (2005) rightly noted that DD metrics are sector-based, and proposed a method for determining congestion without considering sectors. In fact, some of the salient DD variables are based on sector geometries, which would be non-existent within a sector-less NAS. The authors first identified clusters of aircrafts, and then computed a density measure (called *Gaggle Density*) within these clusters. This method could be used to identify regions requiring more ATC attention, where the clusters essentially become ad hoc sectors without boundaries.

2.4 Airspace Sectorization

Airspace has traditionally been divided into sectors, each of which is monitored by a team of air traffic controllers. Kopardekar et al. (2007) noted that, given the current airspace configuration, some air traffic controller resources might be under-utilized, and they also pointed out that the current configuration limits flexibility. They emphasized the necessity of reorganizing the airspace, not only to better use scarce resources, but also to allow for other advanced concepts such as automated separation assurance. Another related research area is that of *dynamic airspace configuration* (DAC), in which the (most likely redesigned) airspace is adjusted in real-time to accommodate demand patterns that might change throughout a given day. Researchers have taken several approaches to solving various combinations of these two problems.

Delahaye et al. (1994) proposed a sector definition method employing genetic algorithms and Voronoi diagrams. Given a set of K generating points, a Voronoi diagram is the partitioning of a plane into K convex polygons such that each polygon contains exactly one generating point and every point in a given polygon is closer to its generating point than to any other (Weisstein 2009). Delahaye et al. (1994) initialized their method by randomly assigning the generating points in the airspace and partitioning the space by building the associated Voronoi diagram. Several random sectorizations were delineated to compose an initial population for the genetic algorithm. Various workload measures, including the number of conflicts between flights and the dwell time for each flight within a sector, were then computed for each resulting sectorization. Two sets of sector definitions were chosen at each iteration as parents, and traditional crossover and mutation operations were applied to the locations of the generating points to design two new partitionings. These new sectorization schemes were compared to the parent schemes, and two of the four were removed from the population based on a decision that combines the workload measures and some element of randomness. Delahaye and Puechmorel (2006) modified this method to work in three dimensions, and Xue (2008) presented a similar algorithm that includes an iterative deepening

step to simplify the computations for the genetic algorithm and to reduce the computational effort. The implementation of Xue used several different cost-based objective functions that balance sector workloads, minimize sector crossings, maximize dwell times, or maximize sector capacity. A significant benefit of these approaches is that all sectors are guaranteed to be convex, and convex sectors are desirable to minimize the likelihood that a flight revisits the same sector. None of these authors reported the experimental computational efficiency of their respective algorithms.

Trandac et al. (2002) developed a constraint programming formulation to define convex sectors subject to sector monitoring workload, conflict workload, and ATC coordination workload. This formulation can be used to solve smaller problem instances directly, but they proposed a two-phase approach to solve larger problem instances that begins with a recursive bisection algorithm to generate an initial solution. A constraint programming formulation is then iteratively applied to smaller subsets of the sectors identified by the initial solution. The authors were able to solve 94% of randomly generated problems with 100 nodes (representing airports and U.S. Navigational Aids (NAVAIDS)) to optimality within 30 minutes.

Martinez et al. (2007) generated a graph based on the current sector architecture in which each node represents a sector boundary crossing and an edge between two nodes implies a direct path between these two boundary crossings. Thus, each edge is contained within a sector, and the graph is used to track current traffic flow patterns. A grid was then superimposed onto the graph, and a heuristic procedure was utilized to create sectors by first placing each node and its cell into a sector and then assigning empty cells from the grid to one of the sectors based on proximity. Workload metrics were computed for each of the sectors in this initial solution, and the sectors were further divided if the workload was higher than allowed. The resulting sectors were found to be satisfactory with respect to peak and average workload, but they were often oddly shaped and rarely convex.

Klein et al. (2007) proposed an *Airspace Playbook* to work in conjunction with the *National Severe Weather Playbook* (FAA 2005), but their procedure turned out to be difficult to automate. A later paper by Klein et al. (2008) cited several challenges with respect to dynamic airspace configuration, and proposed a more flexible approach in which certain subsections of the current airspace configuration were designated as *shareable*. Then, at 15 minute intervals, the airspace was scanned to determine if any sector was overloaded, in which case a shareable sector module was tentatively reassigned to another team of controllers until the traffic subsided. Conversely, under-utilized sectors were permitted to take on additional workload if necessary. The authors noted that this method could be used in conjunction with optimization-based sectorization algorithms in which the initial airspace represents some optimized configuration, and dynamic adaptations are handled with sharing.

In Chapter 6, we pull together ideas from several of these works and propose alternative sectorization schemes in which workload measures are computed at numerous grid points throughout the airspace and accordingly, convex sectors are then generated by a sequential partitioning process which attempts to balance ATC workload. In contrast to the methodologies that apply genetic algorithms to generate Voronoi diagrams (Delahaye et al. 1994, Delahaye and Puechmorel 2006, Xue 2008), we compute workload measures only once for each grid point as opposed to recomputing workload measures for each newly generated set of sectors for comparison with previously generated sets. Moreover, unlike the strategies by Martinez et al. (2007) and Klein et al. (2008), these new schemes do not depend in any way on the current sector configuration.

Chapter 3

Reverse Time-Restricted Shortest Paths

In this chapter, we consider a particular class of network flow problems that seeks a shortest path, if it exists, between a source node s and a destination node d in a connected digraph, such that we arrive at node d at a specified time τ while leaving node s no earlier than a lower bounding time LB, and where the availability of each network link is time-dependent in the sense that it can be traversed only during specified intervals of time. We refer to this problem as the *reverse time-restricted shortest path* problem (RTSP), and it arises, for example, in the context of generating flight plans within air traffic management approaches under severe convective weather conditions. We show that this problem is NP-hard in general, but is polynomially solvable under a special regularity condition. A pseudo-polynomial time dynamic programming algorithm is developed to solve Problem RTSP for the general case, along with an effective heap implementation strategy. Computational results using real flight generation test cases as well as random simulated problems are presented.

Consider a connected digraph $G(N, A)$ with node set N and a directed arc set A. Let $s \in N$ and $d \in N$ be a pair of designated source and destination nodes, respectively, and suppose that associated with each arc $(i, j) \in A$ is a specified (integer-valued) travel time

$f_{ij} \geq 1$. (Note that rational values can be integerized by scaling, if necessary.) However, the availability of each arc (i, j) for possible travel use is time-dependent. Specifically, this availability is characterized by the set

$$F_{ij} = \{t : \text{arc } (i, j) \text{ is available for traversal if we arrive at node } i \text{ at time } t\}, \forall (i, j) \in A. \tag{3.1}$$

Typically, each F_{ij} might be described as a union of closed intervals over some horizon. Furthermore, we are given a particular (integral) arrival time τ at the destination node d, along with a lower bound LB on the permissible departure time at the source node s. The problem, then, is to determine a time-dependent shortest path that leaves node s as late as possible (but no earlier than LB) in order to reach node d at time τ, with no waiting permitted at any nodes, and while respecting the arc availabilities. Note that such a path may not exist, in which case we would like the prescribed algorithm to ascertain this fact, and moreover, if it exists, then it need not be *simple* (i.e., it may contain loops). We refer to this problem as the *reverse time-restricted shortest path problem* (**RTSP**).

This problem is motivated by an air traffic management application, as for example using the Airspace Planning and Collaborative Decision-Making model developed by Sherali, Staats and Trani (2003, 2006). One principal ingredient in this model is the generation of alternative flight plans for each flight that adequately circumvent a time-dynamic severe convective weather system. McCrea et al. (2008) describe a framework for superimposing a flight grid network on a representation of the severe weather system in such a manner that the resulting network is of the form $G(N, A)$, with the availability of each arc $(i, j) \in A$ being restricted by some F_{ij} as in Equation (3.1) so that the probability of encountering severe weather is held below a specified threshold. Moreover, in such a context, under a Ground Delay Program (GDP) imposed at a certain destination airport (node), we are given an arrival time slot for each affected flight and we are required to determine the latest possible corresponding time (exceeding some lower-bounding value) for this flight to take off from its source or origin airport (node), while being cognizant of the severe weather system,

i.e., while respecting the aforementioned flight network arc availabilities. This results in the reverse time-restricted shortest path problem described above, which was heuristically solved in McCrea et al. (2008) via an approximated static shortest path problem.

In the next section, we establish that Problem RTSP is NP-hard, despite the specially-structured arc travel time functions. Thereafter, in Section 3.2, we propose a dynamic programming algorithm for solving Problem RTSP that has a pseudo-polynomial time complexity in general, but that can be implemented in polynomial time under an additional *regularity condition*. An alternative Heap-Dijkstra type procedure that conceptually implements the foregoing dynamic programming routine more efficiently is described in Section 3.3. Finally, Section 3.4 provides computational results using both random problems as well as certain real flight generation instances.

3.1 Complexity Analysis

In this section, we establish that Problem RTSP is NP-hard, despite having special arc travel time functions that are essentially either constant or infinite, depending on the arrival times at the corresponding head nodes.

Proposition 3.1. Problem RTSP is NP-hard.

Proof. We employ a polynomial reduction from the NP-Complete Partition Problem, which is defined as follows (see Garey and Johnson 1979): Given positive integers $a_1, \ldots, a_n$, where $\sum \equiv \sum_{j=1}^{n} a_j$ is even, does there exist a subset $J \subseteq \{1, \ldots, n\}$ such that $\sum_{j \in J} a_j = \sum /2$? Given any instance of the Partition Problem, we construct a corresponding instance of Problem RTSP as follows. Let the underlying graph $G(N, A)$ for Problem RTSP be as depicted in Figure 3.1, with the travel times $f_{ij} \geq 1$ displayed against the arcs.

Furthermore, let $F_{nd} = \left\{ \frac{\Sigma}{2} + n + 1 \right\}$, $F_{so} = \{0\}$, and $F_{ij} = (-\infty, \infty)$, otherwise, with $LB = 0$ and $\tau = \frac{\Sigma}{2} + n + 2$. Then, it follows that for any feasible solution to this instance of Prob-

lem RTSP, if it exists, we must commence at node s at time 0, thereby reaching node 0 at time 1, and then reach node n at time $\frac{\Sigma}{2} + n + 1$, in order to reach node d at time τ. Hence, it follows that for any such feasible solution, if we define $J = \{j : \text{arc } (j-1, j) \text{ having the designated travel time } (a_j+1) \text{ is included in the given solution}\} \subseteq \{1, \ldots, n\}$, and let $\overline{J} = \{1, \ldots, n\} - J$, then we must have

$$\sum_{j \in J}(a_j + 1) + \sum_{j \in \overline{J}} 1 = \frac{\Sigma}{2} + n \Rightarrow \sum_{j \in J} a_j = \frac{\Sigma}{2}.$$

Therefore, the Partition Problem has a solution if and only if the foregoing instance of Problem RTSP returns an optimum starting time at node s of value 0. Hence, Problem RTSP is NP-hard. □

3.2 Dynamic Programming Algorithm (DP)

In this section, we develop a dynamic programming (DP) routine to solve Problem RTSP in pseudo-polynomial time. Notationally, given $G(N, A)$, let the *forward-star* and the *reverse-star* of any node $i \in N$ be respectively defined as $FS(i) = \{j \in N : (i, j) \in A\}$ and $RS(i) = \{j \in N : (j, i) \in A\}$. Without loss of generality, we assume that $RS(s) = FS(d) = \emptyset$. Also, we denote the discrete time horizon of interest as $H \equiv \{LB, LB + 1, \ldots, \tau\}$.

To begin with, consider the following result that provides a computational expedient for curtailing the state-space in the DP algorithm.

Proposition 3.2. Let SP_i be the shortest path value from node s to node $i \in N$ using *static*

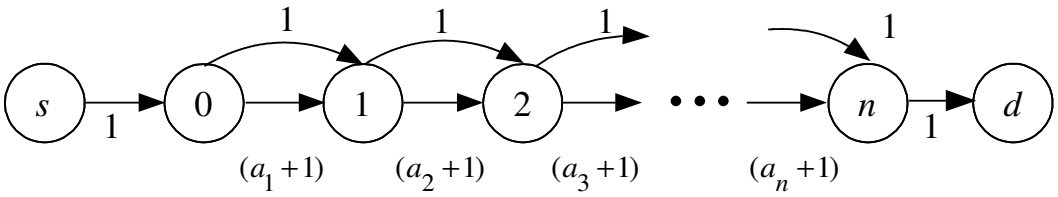


Figure 3.1: Graph $G(N, A)$ for the proof of Proposition 3.1.

travel costs $f_{ij}, \forall (i,j) \in A$, and let $LB_i \equiv LB + SP_i, \forall i \in N$. Then any time-dependent shortest path solution $\{s \equiv m_1, m_2, \ldots, m_h \equiv d\}$ to Problem RTSP that visits nodes $m_1, \ldots, m_h$ at respective times $T_1, \ldots, T_h$ must satisfy $T_k \geq LB_{m_k}, \forall k = 1, \ldots, h$.

Proof. By the feasibility of the path solution, we must have $T_1 \geq LB_{m_1} = LB$. Hence, for each $k = 2, \ldots, h$, since $T_k - T_1 \geq SP_{m_k}$, we have that $T_k \geq T_1 + SP_{m_k} \geq LB + SP_{m_k} = LB_{m_k}$. □

Remark 3.1. Whenever we determine an incumbent solution to Problem RTSP that starts at node s at time $T_s^* > LB$, we can update $LB \leftarrow T_s^*$, and accordingly by Proposition 2, update $LB_i = T_s^* + SP_i, \forall i \in N$. □

We now describe a pseudo-polynomial-time dynamic programming routine (DP) to solve Problem RTSP, based on the following definitions of stages, states, and decisions. This procedure has $K + 1$ stages, where *stage* $k \in \{0, 1, \ldots, K\}$ denotes an epoch when we are located at a node from which it is possible to reach node d in exactly k *steps* (arc transitions), and where

$$K = \min\{\tau - LB, \text{ maximum number of steps possible in } G \text{ in traversing from } s \text{ to } d\}. \tag{3.2}$$

The overall *state-space* S is defined by two-tuples of the type (i_j, t), where *state* (i_j, t) denotes the status of being at node i at time t such that it is then possible to transition along arc (i,j) en-route to reaching node d at the specified time τ.

The set of *states at stage* $k \in \{0, 1, \ldots, K\}$, denoted S_k, represents the subset of S that is comprised of states of the type (i_j, t) such that it is possible to be at node i at stage k. Hence, we have that $S_0 \equiv \{(d, \tau)\}$, where $(d, \tau) \equiv (d_\emptyset, \tau)$ to conform with the general

notation of a state, and where given S_{k-1} for $1 \leq k \leq K$, we compute S_k as follows:

$$S_k = \bigcup_{(j_r,t') \in S_{k-1}} \{(i_j, t), \text{ for each } i \in RS(j) \text{ such that } t \equiv (t' - f_{ij}) \in F_{ij} \text{ and } t \geq LB_i\}, \tag{3.3a}$$

and where we reduce the resultant S_k according to

$$S_k \leftarrow S_k - \{(i_j, t) : \text{ there exists some resident } (i_{j'}, t') \text{ in } S_k \text{ with } t' = t\}. \tag{3.3b}$$

Note that (3.3b) reduces the state-space by not keeping track of alternative paths from node $i \in N$ to node τ having the same travel duration.

The set of *decisions* D_k at stage k represents the set of states to which we could possibly transition; hence, $D_k \equiv S_{k-1}$. We can then follow a backward recursion scheme based essentially on (3.3) in order to solve Problem RTSP as follows.

3.2.1 Algorithm DP

Initialization: Set $S_0 = \{(d, \tau)\}$, and compute $LB_i, \forall i \in N$, as defined in Proposition 3.2. If $\tau < LB_d$, then stop; there does not exist a solution to Problem RTSP. Else, put $k = 1$, and $(k^*, t^*, j^*) = (\emptyset, -\infty, \emptyset)$ (this will record the incumbent solution as indicated below).

Step 1: Given S_{k-1}, compute S_k using Equation (3.3). For each (i_j, t) included within S_k via some $(j_r, t') \in S_{k-1}$, put (a *pointer* or *label*) $DOWN[(i_j, t)] = (j_r, t')$. Additionally, if any state of the type (s_j, t) corresponding to the source node s is generated for inclusion within S_k for which $t > t^*$, then update, $(k^*, t^*, j^*) = (k, t, j)$, $LB = t^*$, and $LB_i = t^* + SP_i, \forall i \in N$.

Step 2: If $S_k = \emptyset$ or $k = K$, go to Step 3. Else, increment $k \leftarrow k + 1$ and repeat Step 1.

Step 3: If $(k^*, t^*, j^*) = (\emptyset, -\infty, \emptyset)$, then there does not exist any solution to Problem RTSP. Else, an optimal solution having a starting time of t^* at node s can be traced by commencing

at state (s_{j^*}, t^*) at stage k^* and following the $DOWN(\cdot)$ pointers up to state (d, τ).

Proposition 3.3. Algorithm DP finitely determines an optimal solution to Problem RTSP, if it exists, with pseudo-polynomial complexity $O(K|H||A|)$, where $|H| = \tau - LB + 1$.

Proof. Finite convergence follows from the finiteness of the state-space S, noting that for any state (i_j, t) generated, we must have $t \in \{LB, \ldots, \tau\}$. The validity of the algorithm follows from Bellman's Principle of Optimality (see Bellman and Dreyfus 1962) based on the definition of the stages, states, and decisions, where given any stage k and any state $(i_j, t) \in S_k$, the best path from this state onward (to node d) depends only on the fact that we are currently located at node i at time t and will transition next to node j via arc $(i, j) \in A$, where $t \in F_{ij}$, and does not depend on any previous states or decisions involved in arriving to this state.

As far as the complexity is concerned, the computation of $LB_i, \forall i \in N$, can be performed in $O(|A|)$ time (see Ahuja et al. 1993). For the main DP scheme, there are $K + 1$ stages. For each stage $k \geq 1$, any node j appears within a state $(j_r, t') \in S_{k-1}$ at most $|H|$ times. For each such (j_r, t'), scanning for $i \in RS(j)$ to possibly include (i_j, t) in S_k takes $O(|RS(j)|)$ time. Performing this for all states in S_{k-1} involving node j therefore takes $O(|H||RS(j)|)$ time, and scanning this over all the possible nodes $j \in N$ gives a complexity per stage of $O(|H||A|)$. Hence, the overall algorithmic complexity is $O(|K||H||A|)$. □

Now, suppose that the travel time functions described for $G(N, A)$ satisfy the assumption that for each $(i, j) \in A$, we have that $F_{ij} = [a_{ij}, b_{ij}]$, where

$$b_{ij} \geq \tau - (\text{shortest static path value from node } i \text{ to node } d), \text{ and } a_{ij} \leq b_{ij}.$$

We refer to this as the *regularity condition*.

Proposition 3.4. Under the regularity condition, Algorithm DP can be terminated after Stage $|N|$, and furthermore, when we generate S_k via (3.3a) at any stage k, we can reduce

this set of states according to

$$S_k \leftarrow S_k - \{(i_j, t) : \text{ there exists some resident } (i_{j'}, t') \text{ in } S_k \text{ with } t' \geq t\}. \tag{3.4}$$

In this case, Algorithm DP has a polynomial time complexity of $O(|N||A|)$.

Proof. Consider any node $i \in N$, and suppose that there exist times t_1 and t_2 with $t_1 > t_2$ such that it is possible to reach node d in G at time τ by starting at node i at either time t_1 or t_2. Then, in any optimal solution to Problem RTSP, we will not visit node i at time t_2. This follows because, on the contrary, if an optimal solution to Problem RTSP begins at node $s \equiv m_1$ at time T_1, then visits some (not necessarily distinct) nodes $m_2, \ldots, m_h$ at respective times $T_2, \ldots, T_h$, and next arrives at node i at time t_2 to finally reach node d at time τ, then by following the same path from node s to node i but visiting each node m_j at time $T_j + (t_1 - t_2)$, for $j = 1, \ldots, h$, we would arrive at node i at time t_1 and thus discover an improved solution that would be feasible by the regularity condition. By this same argument, it follows that an optimal solution cannot contain any loops (i.e., an optimal path must be *simple*), else, an improved feasible solution would result by eliminating any loops in a given solution (note that $RS(s) = FS(d) = \emptyset$). Hence, we can terminate Algorithm DP after Stage $|N|$, and furthermore, (3.4) is valid. Consequently, at any stage $k \geq 1$, any node j appears within a state $(j_r, t') \in S_{k-1}$ at most once, and so, following the proof of Proposition 3.3, the effort per stage is $O(|A|)$, leading to an overall polynomial-time complexity of $O(|N||A|)$. □

Remark 3.2. Under the regularity condition, although the complexity of Algorithm DP is the same as that of standard label-correcting algorithms for static shortest path problems (see Ahuja et al. 1993), it is insightful to note that solving the underlying *static* shortest path problem (e.g., using Dijkstra's algorithm of complexity $O(|A|)$) would *not* solve Problem RTSP. For example, consider the network depicted in Figure 3.2, where the f_{ij}-values are specified on the arcs. The shortest static path is $\{s, d\}$, which commences at node s at time 8. However, if $F_{sd} = [9, 10]$, and $F_{s1} = F_{1d} = [7, 10]$, then we have the regularity condition holding true, but the optimal solution to Problem RTSP is $\{s, 1, d\}$, with a starting time of

7 at node s. □

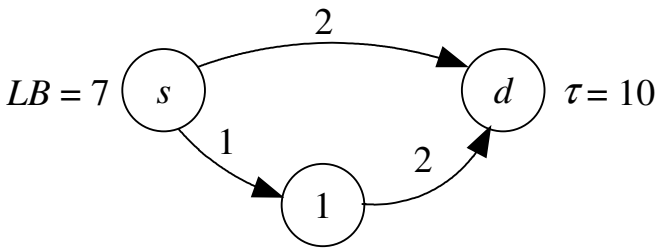
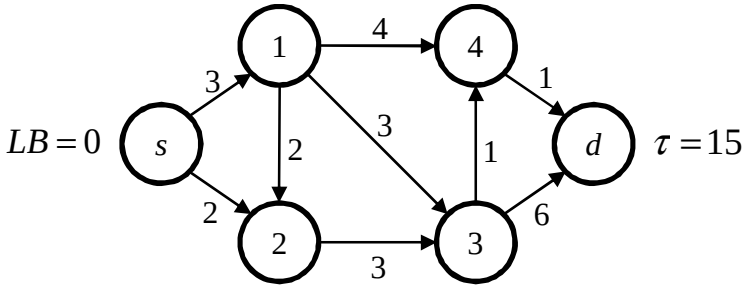


Figure 3.2: Example for Remark 3.2.

3.2.2 Illustrative Example

Consider the network displayed in Figure 3.3, where the f_{ij}-values are shown against the arcs, and the sets F_{ij} are as specified on the right of the graph. Algorithm DP would then proceed as follows, where the arrows indicate the $DOWN(\cdot)$ pointers.



- $F_{14} = [0, 11]$
- $F_{s2} = [0, 5]$
- $F_{s1} = [0, 5]$
- $F_{ij} = [0, 15]$, otherwise

Figure 3.3: Illustrative example.

Initialization. We readily compute (as per Proposition 3.2):

$$LB_s = 0, LB_1 = 3, LB_2 = 2, LB_3 = 5, \text{ and } LB_4 = 5,$$

and we set $(k^*, t^*, j^*) = (\emptyset, -\infty, \emptyset)$. The following computations are performed, where the arrows indicate the DOWN-pointers.

Stage 0: $S_0 = \{(d, 15)\}$

Stage 1: $S_1 = \{(4_d, 14), (3_d, 9)\}$

Stage 2: $S_2 = \{(1_4, 10), (3_4, 13), (1_3, 6), (2_3, 6)\}$

Stage 3: $S_3 = \{(1_3, 10), (2_3, 10), (s_1, 3), (1_2, 4), (s_2, 4)\}$, with $(k^*, t^*, j^*) = (3, 4, 2)$

$\Rightarrow LB_s = 4, LB_1 = 7, LB_2 = 6, LB_3 = 9,$ and $LB_4 = 9.$

Stage 4: $S_4 = \{(1_2, 8)\}$ (**Note:** $(s_1, 1)$ is suppressed because $1 < LB_s = 4$.)

Stage 5: $S_5 = \{(s_1, 5)\}$, with $(k^*, t^*, j^*) = (5, 5, 1) \Rightarrow LB_s = 5, LB_1 = 8, LB_2 = 7,$

$LB_3 = 10,$ and $LB_4 = 10.$

Stage 6: $S_6 = \emptyset \Rightarrow$ stop; optimal starting time at source s is 5, with the corresponding optimal path being $\{s, 1, 2, 3, 4, d\}$.

3.3 Heap Implementation (HI)

In lieu of the dynamic programming routine discussed in Section 3.2, we can essentially execute the same algorithmic approach more efficiently using a *heap implementation* (HI) (see Ahuja et al. 1993, for example). Here, we maintain a heap (HP) of states arranged in nonincreasing order of their times of visitation, commencing with $HP = \{(d, \tau)\}$. At any stage in the algorithmic process, we *extract* (select and delete) the topmost element from HP, and we scan the reverse-star of the corresponding node for this state (terminating the procedure if this node is s), in order to possibly include new states (i_j, t) within HP. The details of this scheme are specified below, where the function $INSERT[(i_j, t), HP]$ denotes the insertion of the state (i_j, t) within HP while maintaining the aforementioned nonincreasing order of times.

3.3.1 Algorithm HI

Initialization: Set $HP = \{(d, \tau)\}$, and compute $LB_i, \forall i \in N$, as defined in Proposition 3.2.

Main Step:

(a) Extract the topmost element from HP, say, (j_r, t'). If $j = s$, go to Step (d). Else, proceed to Step (b).

(b) Scan $RS(j)$:
For each $i \in RS(j)$, if $t \equiv (t' - f_{ij}) \in F_{ij}$ and $t \geq LB_i$, and if $(i_{j'}, t) \notin HP$ for some j', then $INSERT[(i_j, t), HP]$ and put $DOWN[(i_j, t)] = (j_r, t')$. Additionally, if $i = s$ and $t > LB$, then set $LB = t$ and $LB_i = t + SP_i, \forall i \in N$, and then delete from HP any state (p_q, δ) having $\delta < LB_p$.

(c) If $HP = \emptyset$, then stop; there does not exist any solution to Problem RTSP. Else, repeat the Main Step.

(d) Put $t^* = t'$ as the optimal starting time at node s, and recover an optimal solution to Problem RTSP by commencing at state (s_r, t^*), and following the $DOWN(\cdot)$ pointers up to state (d, τ).

Remark 3.3. Note that at any step, we can maintain within HP at most a single state of the type (s_j, t) that corresponds to the source node s, which has the highest t-value among all such states generated. Likewise, under the regularity condition, whenever a particular state (i_j, t) is generated for insertion within HP and there currently exists a state $(i_{j'}, t') \in HP$, then we can include the former state only if $t > t'$, whence we accordingly delete the latter state from HP. $\square$

3.3.2 Illustrative Example

Consider the problem instance displayed in Figure 3.3. Algorithm HI applied to this example would then proceed as follows.

Initialization:

$HP = \{(d, 15)\}$, with $LB_s = 0, LB_1 = 3, LB_2 = 2, LB_3 = 5$, and $LB_4 = 5$.

Main Step 1:

$HP = \{(4_d, 14), (3_d, 9)\}$, with $DOWN[(4_d, 14)] = DOWN[(3_d, 9)] = (d, 15)$.

Main Step 2:

$HP = \{(3_4, 13), (1_4, 10), (3_d, 9)\}$, with $DOWN[(3_4, 13)] = DOWN[(1_4, 10)] = (4_d, 14)$.

Main Step 3:

$HP = \{(1_3, 10), (2_3, 10), (1_4, 10), (3_d, 9)\}$, with

$DOWN[(1_3, 10)] = DOWN[(2_3, 10)] = (3_4, 13)$.

Main Step 4:

$HP = \{(2_3, 10), (1_4, 10), (3_d, 9)\}$.

Main Step 5:

$HP = \{(1_4, 10), (3_d, 9), (1_2, 8)\}$, with $DOWN[(1_2, 8)] = (2_3, 10)$.

Main Step 6:

$HP = \{(3_d, 9), (1_2, 8)\}$.

Main Step 7:

$HP = \{(1_2, 8), (1_3, 6), (2_3, 6)\}$, with $DOWN[(1_3, 6)] = DOWN[(2_3, 6)] = (3_d, 9)$.

Main Step 8:

$HP = \{(1_3, 6), (2_3, 6), (s_1, 5)\}$, with $DOWN[(s_1, 5)] = (1_2, 8)$. Also, we update $LB_s = 5$, $LB_1 = 8, LB_2 = 7, LB_3 = 10$, and $LB_4 = 10$. Accordingly, we update $HP = \{(s_1, 5)\}$.

Main Step 9:

We select $(s_1, 5)$ and terminate with $t^* = 5$, and with the corresponding optimal solution to Problem RTSP given by tracing the $DOWN(\cdot)$ pointers starting at $(s_1, 5)$ to yield the sequence of states $\{(s_1, 5), (1_2, 8), (2_3, 10), (3_4, 13), (4_d, 14), (d, 15)\}$, i.e., the path

$\{s, 1, 2, 3, 4, d\}$.

3.4 Computational Experience

In this section, we provide the results of several sets of experiments concerning Algorithm HI. For the experiments in Section 3.4.1 we generated 800 random graphs and simulated three different types of severe weather systems: small storms, larger storms, and storms designed specifically to render one of the static shortest paths unavailable. We refer to the latter as *targeted* storms. For each of these three, we first applied the algorithm without assuming regularity. Then we transformed the travel time function, as described below, in order to test the algorithm under the regularity condition. In Section 3.4.2, we used real flight generation test cases based on the network induced by the U.S. Navigational Aids. The algorithm was implemented in MATLAB 7.4 using a 1.66 GHz Intel Core Duo T2300 processor with 2.5 GB of RAM.

3.4.1 Simulated Weather Systems

Without the Regularity Condition

For our first set of experiments we randomly generated connected digraphs having at least one directed path from the source s to the destination d. The static travel times f_{ij} were taken as integers proportional to the Euclidean distances between nodes, and the arrival time τ was randomly chosen as an integer between 25% and 75% of the travel time for the *a priori* known path. The lower bound on the departure time LB was assumed to be zero. Then, to define the arc availability function F_{ij}, we simulated small and large convective weather systems by generating a starting time, path, duration, and radius for the storm (all random), with the storm duration being randomly chosen on the interval $[0.75\tau, \tau]$ and the starting time randomly chosen on the interval $[0, \tau - \text{storm duration}]$. An arc was declared

to be temporarily unavailable when the simulated storm crossed over it, and the length of time for the outage was set depending on the angle of incidence between the storm and the arc (i.e., the arc was unavailable for a longer period of time if the storm's movement was relatively parallel to the arc).

The targeted storms were generated in a similar manner, but the path of the storm in each case was designed such that the computed static shortest path for this instance was rendered unavailable. Thus, for any optimal departure time t^* for problem RTSP we have $t^* \leq \tau - SP_d$, with equality only when there exists an alternative static shortest path that is feasible to the problem. Moreover, any solution with $t^* < \tau - SP_d$ could not have been obtained using Dijkstra's algorithm (as illustrated in Remark 3.2).

Assuming $RS(s) = FS(d) = \emptyset$, the maximum number of directed arcs in each graph is given by $A_{max} = 2\binom{|N|-2}{2} + 2(|N| - 2)$ for a given number of nodes $|N|$. We generated 50 instances for each combination of $|N| \in \{50, 100, 200, 400\}$ and $|A| \in \{0.05A_{max}, 0.10A_{max}, 0.25A_{max}, 0.50A_{max}\}$. The results obtained are summarized in Table 3.1, which provides the median, mean, standard deviation, minimum, and maximum CPU times (in seconds) over the 50 instances for each $(|N|, |A|)$ pair and for each of the three types of storms.

The vast majority of the 2,400 problem instances were solved to optimality in under a minute, but 13 instances required more effort. Table 3.2 shows the distribution of solution times. Plotting the median CPU times versus $|N||A|$ indicates a strong linear relationship between the two, with a correlation coefficient greater than 0.9997 for all three storm types. In 39 of the 800 graphs the length of the static shortest path exceeded $\tau - LB$. All of these instances were infeasible even without a storm. In addition to these problems, there were 5 infeasible instances for small storms, 5 for large storms, and 2 for targeted storms. The solution times required for infeasible instances were comparable to those for the feasible instances.

Given the dissimilarities between the storm types (e.g., the starting points, durations, and directions of movement), we chose to compare the algorithmic performance relative to

Table 3.1: CPU times (seconds) for random graphs with a simulated weather-induced availability function.

		Small Storm				**Large Storm**				**Targeted Storm**			
		% A_{max}				% A_{max}				% A_{max}			
Nodes		**5**	**10**	**25**	**50**	**5**	**10**	**25**	**50**	**5**	**10**	**25**	**50**
50	Median	0.009	0.016	0.036	0.069	0.009	0.017	0.037	0.069	0.009	0.018	0.037	0.072
	Mean	0.015	0.020	0.039	0.071	0.014	0.022	0.043	0.075	0.011	0.022	0.039	0.089
	S.D.	0.032	0.010	0.008	0.011	0.019	0.020	0.025	0.031	0.005	0.013	0.006	0.089
	Min	0.007	0.014	0.033	0.064	0.007	0.014	0.032	0.064	0.007	0.014	0.033	0.065
	Max	0.237	0.061	0.072	0.127	0.140	0.150	0.208	0.277	0.029	0.083	0.061	0.699
100	Median	0.057	0.113	0.266	0.551	0.057	0.111	0.267	0.532	0.059	0.115	0.276	0.549
	Mean	0.078	0.152	0.341	0.712	0.065	0.114	0.299	0.545	0.071	0.117	0.286	0.606
	S.D.	0.093	0.160	0.354	1.111	0.040	0.011	0.175	0.039	0.060	0.010	0.025	0.205
	Min	0.052	0.101	0.251	0.523	0.051	0.102	0.256	0.511	0.053	0.106	0.261	0.519
	Max	0.636	0.994	2.582	8.406	0.324	0.156	1.502	0.727	0.414	0.146	0.361	1.901
200	Median	0.452	0.880	2.160	4.297	0.425	0.836	2.101	4.204	0.442	0.858	2.144	4.287
	Mean	0.462	0.884	2.185	14.250	0.441	0.847	30.927	194.279	0.451	0.882	2.263	4.393
	S.D.	0.043	0.035	0.090	70.307	0.056	0.030	154.678	1 343.956	0.043	0.072	0.359	0.446
	Min	0.414	0.838	2.112	4.217	0.408	0.819	2.033	4.101	0.419	0.830	2.077	4.181
	Max	0.695	0.982	2.692	501.456	0.742	0.966	1 046.363	9 507.416	0.659	1.295	4.207	7.235
400	Median	3.438	6.899	16.966	32.295	3.367	6.635	16.432	33.059	3.414	6.908	16.971	34.048
	Mean	3.476	7.095	2 299.711	32.303	3.393	6.701	2 276.423	45.457	3.463	7.130	2 244.596	42.752
	S.D.	0.114	1.028	16 142.745	0.390	0.118	0.187	15 962.297	83.042	0.172	1.284	15 740.757	27.171
	Min	3.380	6.753	15.872	31.672	3.288	6.468	16.099	32.681	3.348	6.693	16.742	33.452
	Max	4.065	14.111	114 163.228	33.255	3.974	7.318	112 889.454	620.272	4.396	15.831	111 322.468	171.532

Table 3.2: Distribution of CPU times for the original problems.

	Small	**Large**	**Targeted**
0-1 min	798	793	795
1-5 min	0	2	4
5-10 min	1	1	0
10-60 min	0	2	0
1-5 hr	0	1	0
$>$ 5 hr	1	1	1

storm size by examining the run-times over instances for each pair of $(|N|, |A|)$-values. Since the CPU times are, by inspection, not normally distributed, we compared the median solution times for each $(|N|, |A|)$ pair using a Kruskal-Wallis test (see, e.g., Wackerly et al. 2007) in order to assess the impact of the storm type on the solution time. The p-values for each pair are given in Table 3.3. These results indicate that the type of storm can impact the distribution of the solution time, and in many of the cases, it seems that the run-times are lower for larger storms, but this pattern is not consistent enough to draw any conclusions. Nevertheless, it is plausible that more constrained problems yield shorter run-times because of smaller state-spaces.

Table 3.3: Kruskal-Wallis test results: average ranks for each storm type and resulting p-values.

		A_{max}			
Nodes		**5**	**10**	**25**	**50**
50	Small	73.7	71.5	69.4	59.9
	Large	79.0	72.7	77.8	69.9
	Targeted	73.8	82.3	79.2	96.7
	p-value	0.7805	0.3964	0.4746	< 0.0001
100	Small	71.7	73.4	60.2	85.6
	Large	71.0	66.0	67.2	54.1
	Targeted	83.8	87.1	99.2	86.9
	p-value	0.2534	0.0479	< 0.0001	0.0001
200	Small	96.7	97.6	89.6	95.8
	Large	51.5	45.4	55.4	45.3
	Targeted	78.2	83.5	81.6	85.4
	p-value	< 0.0001	< 0.0001	0.0002	< 0.0001
400	Small	97.8	96.0	76.9	27.9
	Large	49.4	41.9	43.9	77.5
	Targeted	79.4	88.7	102.1	121.1
	p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001

With the Regularity Condition

The availability set F_{ij} for a disrupted arc in the first set of experiments was typically a union of two closed intervals of the type $F_{ij} = [LB, t_{ij}^{(1)}] \cup [t_{ij}^{(2)}, t_{ij}^{(3)}]$, where $LB \leq t_{ij}^{(1)} \leq t_{ij}^{(2)} \leq t_{ij}^{(3)}$, which cannot meet the regularity condition when $t_{ij}^{(1)} < t_{ij}^{(2)}$. However, suppose that we transform F_{ij} to an alternative arc availability set $\tilde{F}_{ij}$ as follows:

$$\tilde{F}_{ij} = [a_{ij}, b_{ij}] = \text{cl}(\text{conv}[F_{ij} \cap [LB, \tau)]), \forall (i,j) \in A, \tag{3.5}$$

where cl(conv[$\cdot$]) denotes the closure of the convex hull of [$\cdot$]. Notice that when $\tau \in [LB, t_{ij}^{(1)}]$ we have $\tilde{F}_{ij} = [LB, \tau]$, and when $\tau \in (t_{ij}^{(1)}, t_{ij}^{(2)}]$ we have $\tilde{F}_{ij} = [LB, t_{ij}^{(1)}]$. When either of these cases apply, or when $t_{ij}^{(1)} = t_{ij}^{(2)}$, $\tilde{F}_{ij}$ simply gives an *equivalent* definition of the feasible set of times at which one can traverse arc (i,j). However, when $\tau > t_{ij}^{(2)} > t_{ij}^{(1)}$, we have $\tilde{F}_{ij} = [LB, \min\{t_{ij}^{(3)}, \tau\}]$, which relaxes the availability set for that arc. Now, suppose that we solve RTSP using $\tilde{F}$, where, *in case* regularity is satisfied, we invoke it within the algorithm. If the resulting solution yields node visitation times along the optimal path that satisfy the F_{ij} -availability sets, then this solution is optimal to the original problem because $\tilde{F}$ corresponds to a relaxation to the original problem. Otherwise, we re-solve the problem using the original data.

For each of the 2,400 problem instances, we transformed F to $\tilde{F}$ and then checked $\tilde{F}$ for regularity. We then solved RTSP, invoking the regularity condition when it was valid. The results, including CPU times for verifying the regularity condition and solving RTSP, are provided in Table 3.4.

Table 3.4: CPU times (seconds) for verifying the regularity condition (Verify) and then solving RTSP (Solve) under $\tilde{F}$.

		Small Storm								**Large Storm**								**Targeted Storm**							
		% A_{max}								% A_{max}								% A_{max}							
		5		**10**		**25**		**50**		**5**		**10**		**25**		**50**		**5**		**10**		**25**		**50**	
Nodes		**Verify**	**Solve**	**Verify**	**Solve**	**Verify**	**Solve**	**Verify**	**Solve**	**Verify**	**Solve**	**Verify**	**Solve**	**Verify**	**Solve**	**Verify**	**Solve**	**Verify**	**Solve**	**Verify**	**Solve**	**Verify**	**Solve**	**Verify**	**Solve**
50	Median	0.007	0.008	0.013	0.015	0.032	0.035	0.063	0.066	0.007	0.009	0.014	0.016	0.032	0.035	0.065	0.068	0.007	0.009	0.013	0.016	0.032	0.036	0.064	0.070
	Mean	0.008	0.011	0.014	0.017	0.032	0.035	0.063	0.067	0.007	0.009	0.014	0.017	0.033	0.036	0.065	0.069	0.007	0.010	0.014	0.017	0.032	0.037	0.064	0.071
	S.D.	0.006	0.009	0.001	0.004	0.001	0.002	0.001	0.005	0.000	0.002	0.002	0.004	0.001	0.003	0.004	0.005	0.000	0.003	0.002	0.004	0.001	0.003	0.001	0.004
	Min		0.007		0.013		0.032		0.062		0.007		0.013		0.032		0.064		0.007		0.014		0.034		0.066
	Max		0.068		0.028		0.041		0.094		0.017		0.030		0.042		0.096		0.017		0.033		0.045		0.087
100	Median	0.051	0.055	0.101	0.106	0.248	0.256	0.495	0.511	0.053	0.056	0.103	0.109	0.257	0.263	0.515	0.524	0.052	0.057	0.102	0.110	0.253	0.268	0.508	0.538
	Mean	0.051	0.057	0.101	0.107	0.248	0.259	0.498	0.512	0.053	0.059	0.104	0.111	0.258	0.266	0.517	0.526	0.053	0.060	0.104	0.113	0.255	0.272	0.510	0.539
	S.D.	0.002	0.010	0.002	0.007	0.002	0.010	0.010	0.010	0.002	0.014	0.003	0.007	0.005	0.009	0.009	0.015	0.003	0.010	0.003	0.008	0.005	0.012	0.007	0.009
	Min		0.050		0.100		0.247		0.492		0.052		0.102		0.257		0.506		0.053		0.105		0.260		0.522
	Max		0.105		0.142		0.308		0.541		0.145		0.142		0.301		0.577		0.111		0.153		0.336		0.567
200	Median	0.406	0.423	0.802	0.818	1.987	2.016	3.971	4.030	0.412	0.424	0.816	0.822	2.042	2.071	4.061	4.093	0.408	0.436	0.816	0.850	2.033	2.120	4.053	4.221
	Mean	0.417	0.426	0.805	0.827	1.986	2.026	3.974	4.031	0.413	0.430	0.818	0.827	2.051	2.075	4.073	4.101	0.411	0.438	0.815	0.855	2.034	2.127	4.053	4.234
	S.D.	0.063	0.022	0.016	0.022	0.016	0.036	0.043	0.051	0.010	0.020	0.012	0.018	0.032	0.047	0.061	0.063	0.006	0.016	0.005	0.017	0.010	0.037	0.011	0.044
	Min		0.395		0.791		1.962		3.937		0.405		0.804		2.008		3.987		0.418		0.829		2.080		4.172
	Max		0.524		0.908		2.135		4.194		0.499		0.880		2.233		4.238		0.485		0.902		2.242		4.367
400	Median	3.199	3.248	6.388	6.502	16.070	16.229	32.470	32.883	3.241	3.259	6.466	6.533	16.424	16.505	32.646	32.883	3.248	3.396	6.491	6.773	16.240	16.865	32.464	33.809
	Mean	3.200	3.256	6.404	6.521	16.115	16.889	32.675	33.102	3.247	3.277	6.474	6.560	16.459	17.182	32.862	33.085	3.249	3.408	6.495	6.789	16.245	17.547	32.461	33.819
	S.D.	0.029	0.060	0.051	0.103	0.418	4.441	0.886	1.027	0.030	0.059	0.054	0.097	0.341	4.321	1.316	0.857	0.012	0.061	0.017	0.079	0.043	4.405	0.065	0.202
	Min		3.164		6.345		15.813		31.624		3.185		6.404		16.103		32.383		3.342		6.683		16.704		33.405
	Max		3.462		6.767		47.611		35.971		3.468		6.799		46.992		37.360		3.621		6.973		46.091		34.413

Table 3.5 gives a breakdown of the problem instances that were determined to satisfy the regularity condition. These are divided into three categories. Most often, solving the relaxation returned a solution that was feasible, and hence optimal, to the original problem. Some of the original instances were feasible, but the corresponding relaxation solution was infeasible under the original availability function. In the remaining cases the original problems were infeasible, and any solution obtained from the relaxation would be infeasible to the original problem.

Table 3.5: Breakdown of problem instances that satisfy the regularity condition and corresponding results.

		Small Storm					**Large Storm**					**Targeted Storm**				
		% A_{max}				**Total**	% A_{max}				**Total**	% A_{max}				**Total**
Nodes		**5**	**10**	**25**	**50**		**5**	**10**	**25**	**50**		**5**	**10**	**25**	**50**	
50	Regularity Condition	48	50	50	48	196	48	49	50	50	197	48	47	44	42	181
	Relax Soln. Opt. to Orig.	34	41	43	45	163	31	38	37	39	145	25	27	32	16	100
	Relax Soln. Infeas. to Orig.	4	4	4	1	13	8	5	10	8	31	13	16	10	24	63
	Original Problem Infeasible	10	5	3	2	20	9	6	3	3	21	10	4	2	2	18
100	Regularity Condition	50	50	50	50	200	50	50	50	50	200	50	46	49	44	189
	Relax Soln. Opt. to Orig.	41	42	47	47	177	37	45	48	42	172	27	32	18	19	96
	Relax Soln. Infeas. to Orig.	2	6	3	1	12	5	3	2	6	16	16	12	31	24	83
	Original Problem Infeasible	7	2	0	2	11	8	2	0	2	12	7	2	0	1	10
200	Regularity Condition	50	50	50	50	200	50	50	50	48	198	47	46	45	46	184
	Relax Soln. Opt. to Orig.	48	44	46	47	185	47	44	39	46	176	25	20	23	27	95
	Relax Soln. Infeas. to Orig.	1	3	1	2	7	2	3	8	1	14	21	23	19	18	81
	Original Problem Infeasible	1	3	3	1	8	1	3	3	1	8	1	3	3	1	8
400	Regularity Condition	50	50	50	50	200	50	50	50	50	200	42	47	44	47	180
	Relax Soln. Opt. to Orig.	47	48	48	48	191	47	46	47	47	187	27	24	29	27	107
	Relax Soln. Infeas. to Orig.	2	1	2	1	6	2	4	3	2	11	14	23	15	19	71
	Original Problem Infeasible	1	1	0	1	3	1	0	0	1	2	1	0	0	1	2
Total	Regularity Condition	198	200	200	198	796	198	199	200	198	795	187	186	182	179	734
	Relax Soln. Opt. to Orig.	170	175	184	187	716	162	173	171	174	680	104	103	102	89	398
	Relax Soln. Infeas. to Orig.	9	14	10	5	38	17	15	23	17	72	64	74	75	85	298
	Original Problem Infeasible	19	11	6	6	42	19	11	6	7	43	19	9	5	5	38

The most conspicuous outcome is that the problem instances that consumed more than 5 CPU hours in the previous section were solved to optimality in just over a minute. Also, it is noteworthy that, whereas Algorithm HI executed relatively faster in most cases without the aforementioned transformation and check for the regularity condition, this latter strategy was critical for efficiently solving the extreme outlier cases and other difficult instances. These results might suggest first attempting to solve the problem without verifying regularity, and then, once a specified time threshold has passed, attempting to invoke the regularity condition if applicable (or alternatively, running the two procedures in parallel).

3.4.2 Actual Flight Paths

For generating flight paths in the real instances solved, we used the U.S. Navigational Aids (NAVAIDS) as the underlying set of nodes, and we constructed the network using the Ellipsoidal Region Technique (ERT) derived by Sherali, Staats and Trani (2003) and Sherali, Hobeika and Kangwalklai (2003). Thus, we considered only those nodes that are contained in a defined ellipse for which the foci represent the origin and destination airports – in this case Dallas/Fort Worth International Airport (DFW) and Hartsfield-Jackson Atlanta International Airport (ATL), respectively. We also chose not to allow arcs shorter than 100 km nor longer than 500 km. Short arcs result in frequent changes in direction that add to the workload of the pilots and to the discomfort of the passengers, and longer arcs might result in the aircraft getting off-course (the larger the distance between the NAVAIDS, the larger the error). Both of these techniques served to limit the size of the problem to 385 nodes and 26,963 arcs. Another method to reduce the number of arcs would be to apply the ERT between each node i and the destination d, and to allow only arcs from node i that lie within the defined ellipse.

Several times each day, multiple carriers fly this route and the typical scheduled duration is approximately two hours. Figure 3.4 shows a typical path for this route obtained by downloading historical flight routes from flightaware.com (2008). We address the scenario where convective severe weather prevents the aircraft from flying its normal route.

The distance traveled on the typical path is 1,208.9 km. For the purpose of illustration in this paper, we consider only distance traveled as the *cost* of each arc; whereas, in reality, flight planners take into account other features such as wind and accompanying air traffic when determining arc costs. However, this does not affect the implemented algorithm. Paths A and B were derived based on two possible positions of the weather system. The lengths of Path A and B are 1,429.9 km and 1,247.8 km, respectively. Path A is a significant detour, whereas Path B is only slightly longer than that of the typical path. If one assumes that the aircraft's block speed is approximately 800 km/hr, both trips are completed in less than

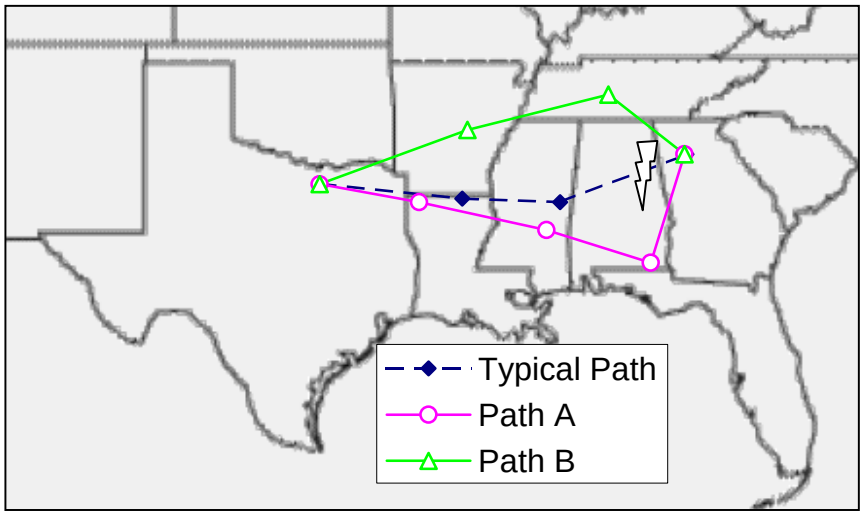


Figure 3.4: An example of two flight paths generated using Algorithm HI.

two hours. (Block speed is the average speed that includes takeoff and landing phases.) In fact, the pilot could depart DFW approximately 13 minutes after the scheduled takeoff time when taking Path A, or approximately 26 minutes later when taking Path B, and arrive on time at ATL.

Chapter 4

Enhancements to the APCDM

In this chapter we discuss several enhancements and modifications to the APCDM. The APCDM is enhanced to include (a) the selection of slot-exchange trade offers suggested by participating airlines based on their allotted slots under a GDP; (b) connections between continuing flights, and (c) several alternative equity concepts. Thus, the proposed model selects among alternative flight plans for the affected flights while integrating slot-exchange mechanisms induced by multiple Ground Delay Programs (GDPs) to permit airlines to improve flight efficiencies through a mediated bartering of assigned slots, and simultaneously considering issues related to continuing flights, sector workloads, airspace conflicts, as well as overall equity concerns among the involved airlines in regard to accepted slot trades and flight plans. Both full and light versions of this model are developed and tested using realistic data derived from the *Enhanced Traffic Management System* (ETMS) provided by the FAA.

Section 4.1 describes a hypothetical illustrative example to elucidate the slot-exchange concept. Section 4.2 develops the proposed slot-exchange mechanism for inclusion within the APCDM model, and Section 4.3 models continuing flight restrictions. Section 4.4 delineates various alternative approaches for modeling equity with respect to the selected trade offers. Section 4.5 discusses the resulting proposed full and light versions of the APCDM model, and Section 4.6 presents computational results and additional insights using test instances

based on realistic data derived from the ETMS.

4.1 Illustrative Trade Offer Example and Insights

Suppose that each airline delineates alternative arrival times for each of its flights that are later than the originally scheduled times, based on the slots it owns, as well as based on slots that it could possibly acquire through trade offers.

As an illustrative example, consider the following situation at a particular airport in the overall problem at which a GDP has been imposed. Suppose that the execution of the RBS procedure produces the arrival slot allocations for Airlines A, B, and C as depicted in the left-hand blocks of Figure 4.1. For example, Flight 1 for Airline A (designated as A1) has been allotted the 0800 arrival time slot, and so forth. Under the enhancements to the GDP, Airline A owns the 0800-slot and thus, may either utilize it for another one of its flights through a swapping process, or may consider offering that slot to another airline in return for a slot that reduces the delay for one of its subsequent flights. In this spirit, consider the AMAL trade offers from Airlines A, B, and C as shown in Figure 4.1, where the flights operated by the particular airline are shaded for clarity in each case.

For example, Airline A has offered to increase the delay of Flight 1 with an arrival time no later than the 0816 slot in return for moving Flight 3 up to an arrival time ranging between the 0824 and 0816 slots. It is insightful to note here the importance of considering inter-airline trading. Namely, airlines cannot always accomplish slot exchanges within their own operations. For example, airline A cannot (or is not willing to) trade slots between flights A1 and A3. Using the slot times as nodes, and the transition of flights from current to new slots as arcs, we can represent the slot offers as a directed network (see Figure 4.2). For notational simplicity, we designate slot time 0800 as Node 1, slot time 0804 as Node 2, and so on. The potential movements of a flight from its current slot to the newly proposed slots are designated by directed arcs. Acceptable trades, subject to the aforementioned

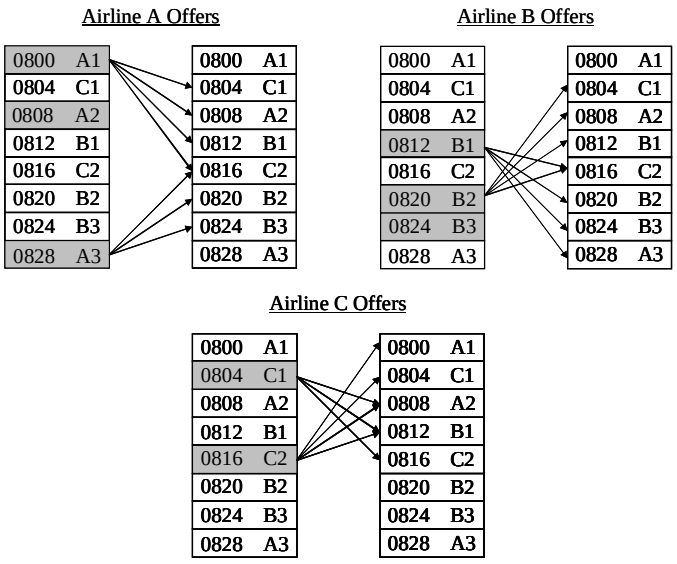


Figure 4.1: Airline slot allocations and slot offers.

trade restrictions, are in the form of directed cycles in this network. For example, referring to Figure 4.2, some resulting directed cycles that correspond to sets of possible trades that preserve feasibility to the trade restriction include $\{(1,4,8,6,2,5,1)\}$, $\{(4,6,4),(2,5,2)\}$, $\{(1,5,1),(2,4,8,6,2)\}$, and $\{(4,8,6,4),(1,2,5,1)\}$, where, for instance, the first foregoing set corresponds to the swaps A1 $\rightarrow$ B1, B1 $\rightarrow$ A3, A3 $\rightarrow$ B2, B2 $\rightarrow$ C1, C1 $\rightarrow$ C2, and C2 $\rightarrow$ A1. Observe also that the second set involves intra-airline swaps, which are also included within this modeling framework (any such enforced or previously declared exchange is assumed to be done *a priori*). Note that the final resulting mix of flight plans need to be collectively compatible with respect to sector workload and conflict resolution restrictions, and also with the governing equity measures.

Finally, note that when offers are made as conditional queries that do not commit an airline to accept any trades until receiving feedback on the results of the exchange, this would require possibly multiple runs of the proposed model that would entail some additional acceptance conditions to ensure convergence of this process. To avoid this iterative process, we assume that airlines make firm trade offers that they commit to adopt if accepted as part of the overall exchange mechanism. The following two sections respectively describe

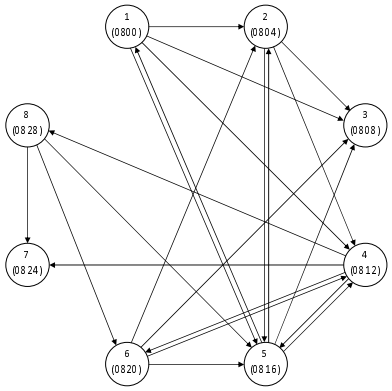


Figure 4.2: Slot offer network.

further modeling details for representing and automatically selecting feasible trades, and for enforcing equity, within the APCDM model.

4.2 Modeling Feasible Slot-Exchanges within APCDM

In this section, we augment the APCDM model to account for slot exchanges between participating airlines. McCrea (2006) modeled slot trades using a theoretical exchange graph G, in which nodes represented slot times and a directed arc between two nodes u and v was defined if the flight assigned to slot u has offered to move to slot v. If a flight plan corresponding to such a trade offer is selected, then the flight originally assigned to slot v must move to another slot as well. Consequently, any set of accepted trade offers must be a part of some directed cycle within G. McCrea (2006) also proposed an optional constraint requiring the net reduction in passenger minutes to be nonnegative. However, each flight was restricted to be involved in at most a single trade offer; flight cancellations were not modeled, and only a single GDP airport was considered with no flight connection constraints. We therefore extend APCDM to incorporate all of these foregoing additional features.

Toward this end, consider any particular airport in the problem that is concerned with such trades under a GDP. For this airport, suppose that based on various offer schemes, we have a collection of connected *exchange graphs* $G_k, k = 1, \ldots, K$, where K represents the

number of (separable, connected) components resulting from the trade offers, and where each exchange graph G_k has node set N_k corresponding to the related slots with their currently occupying flights, and has directed arcs with each arc corresponding to an altered arrival time, and therefore corresponds to a proposed flight plan (f, p) for a flight f, which switches from its current tail-slot to the new head-slot. Accordingly, a variable x_{fp} is associated with each arc. For example, Figure 4.2 illustrates a particular exchange graph in which, for instance, arc (1, 4) corresponds to Flight A1 adopting a designated flight plan that arrives at the GDP destination airport at time 0812. Hence, in effect, valid trades will be represented by directed circuits within any such graph G_k. Note that in case we wish to consider multiple flight plans corresponding to a given slot-trade (or arc) that might differ in their trajectories and departure times, we will maintain multiple corresponding arcs, each representing a particular plan. Observe that we do not construct self-loop arcs pertaining to plans that retain the associated assigned slot for the corresponding flight in this approach. The x_{fp}-variable associated with any such flight plan would be directly accommodated within Constraint (a) for the APCDM model, and if set equal to one, would therefore automatically imply the retention of the allotted slot for the particular flight. Furthermore, we accommodate flight cancellation offers that, if accepted, would create vacant slots for other flights to possibly occupy as follows. Given any exchange graph G_k, suppose that some flight f that currently occupies slot s has an associated cancellation plan $p = 0$ as a possible option. We then create a dummy companion f_ϕ, say, for flight f and let it currently occupy a dummy blank slot s_ϕ, where s_ϕ now represents a new node in G_k. Next, we create a directed arc (with the associated variable x_{f0}) from the node representing slot s to that representing slot s_ϕ in G_k. Hence, node s_ϕ will be involved in an exchange circuit if and only if $x_{f0} = 1$, whence the dummy flight f_ϕ would end up moving to some newly vacated real slot, depending on the particular resulting exchange circuit (including possibly to slot s). Therefore, we also generate forward arcs going out of the slot s_ϕ node to all the other nodes corresponding to real slots in G_k and associate with each such arc a variable $x_{f_\phi p}$ for a corresponding distinct dummy plan $p \in P_{f_\phi}$. In addition, we model the case when there exist some real currently

vacant slots by creating fictitious flights to occupy such slots and then for each such slot (node in G_k), we generate incident directed arcs coming from, and going to, each of the real occupied slots within G_k, along with the appropriate designation of flight plan variables for each such arc as before. Now, because the selected flight plans that constitute a valid exchange scheme within any graph G_k are represented by directed cycles (or *circuits*), with a node being involved in at most one such circuit, we formulate the following constraints:

$$\sum_{(f,p)\in RA_n^k} x_{fp} = \sum_{(f,p)\in FA_n^k} x_{fp}, \quad \forall n \in N_k, \quad \forall k = 1, \ldots, K \tag{4.1a}$$

$$\sum_{(f,p)\in FA_n^k} x_{fp} \leq 1, \quad \forall n \in N_k, \quad \forall k = 1, \ldots, K \tag{4.1b}$$

where for each $k = 1, \ldots, K$, and $n \in N_k$, we have

$$RA_n^k = \{(f,p) : x_{fp} \text{ corresponds to a "reverse" arc coming into node } n\}$$
$$FA_n^k = \{(f,p) : x_{fp} \text{ corresponds to a "forward" arc going out of node } n\}.$$

Hence, (4.1a) requires flows to be circulatory, and (4.1b) enforces at most a unit flow in each circuit, with at most one circuit involving any particular node.

However, we also want to ensure that the trades prompted by the selected circuits satisfy the trade restrictions. Toward this end, let

$$D = \{(f,\#) : \text{ flight } f \text{ is offered to be delayed in offer indexed by } \#\}$$

and for each $(f,\#) \in D$, let

$$\begin{aligned} H_{(f,\#)} = \{(f',p') : &\text{ at least one of these designated flight plans } p' \text{ of flight} \\ &f' \text{ belonging to the same airline must be selected in order to accept} \\ &\text{delaying flight } f \text{ in offer } \#\}. \end{aligned}$$

Also, let

$$DP_{(f,\#)} = \{p : \text{ plan } p \text{ corresponds to the delay of } f \text{ in offer } \#\}, \forall (f,\#) \in D.$$

Note that by the foregoing notation, a given flight $f \in D$ can be possibly involved in *multiple offers*, i.e., we could have $(f,\#) \in D$ for more than one #-value, where the different corresponding H-sets might possibly intersect. However, in such a case, we assume that $DP_{(f,\#)} \cap DP_{(f,\#\#)} = \emptyset$ for distinct offers # and ##, so that implicitly, no more than one offer can be accepted for a given flight. Likewise, for any distinct flights f_1 and f_2, we assume that $H_{(f_1,\#)} \cap H_{(f_2,\#\#)} = \emptyset$, for any indices # and ##. Hence, a delay-reducing move cannot by itself compensate for more than one delay-increasing move. Accordingly, in any of the exchange graphs, if for some $(f,\#) \in D$, we have that $x_{fp} = 1$ for any $p \in DP_{(f,\#)}$, then at least one of $x_{f'p'}$ for $(f',p') \in H_{(f,\#)}$ must also be 1. This is enforced by the constraints

$$\sum_{p \in DP_{(f,\#)}} x_{fp} \leq \sum_{(f',p') \in H_{(f,\#)}} x_{f'p'}, \quad \forall (f,\#) \in D. \tag{4.2}$$

Furthermore, while some slot exchanges might result in a significant improvement in terms of reducing the overall net delay in passenger-minutes, the structure of the individual trade offers could possibly result in a net increase in passenger-minutes of delay for one or more airlines. In some instances, this might be acceptable for the corresponding airline when the airline's motivation was something other than passenger delay reduction, such as reducing the downstream effects of airframe or crew delays in their network of flights. Nonetheless, we also impose an additional restriction that for each airline, the realized net reduction in passenger-minutes of delay at the GDP-restricted airport due to slot exchanges should be nonnegative. Note that whereas the information regarding the actual number of passengers on any flight is not readily available, and is in fact guarded by the airlines, we could use a fixed (agreed-upon) value to estimate this quantity (referred to as PAX_f below) based on applying an assumed load-factor to the type of aircraft, perhaps using historical

data. To model this (optional) restriction, we define the following additional entities, where all delays are measured with respect to the published schedule:

- A^{trade} : The set of airlines involved in the trade offers.
- PAX_f : The estimated number of passengers associated with flight f.
- δ_{fp} : The delay (minutes) for plan p of flight f relative to the published schedule. (Note that $\delta_{fp} \geq 0, \quad \forall (f,p)$.)
- DL_f^{GDP} : The delay (minutes) for flight f based on the GDP assigned slot.

The restriction on achieving a nonnegative *net reduction in passenger-minutes* ($NRPM$) of delay at the GDP airport for each airline can be formulated as follows:

$$NRPM_\alpha \equiv \sum_{(f,\#)\in D: f\in A_\alpha} \left\{ \sum_{p\in DP_{(f,\#)}} PAX_f \left[DL_f^{GDP} - \delta_{fp}\right] x_{fp} + \sum_{(f',p')\in H_{(f,\#)}} PAX_{f'} \left[DL_{f'}^{GDP} - \delta_{f'p'}\right] x_{f'p'} \right\} \geq 0, \quad \forall \alpha \in A^{\text{trade}}. \tag{4.3}$$

Alternatively, to complement (4.2), we could require that the net reduction in passenger-minutes of delay corresponding to each trade, given by $\{\cdot\}$ in (4.3), should be nonnegative. However, this might be too restrictive, although it would permit the corresponding airline making the offer to specify additional delay-increasing move slots for the related $f \in D$, while being assured that any such accepted delay will be adequately compensated by an associated delay-reducing move in terms of the resulting net reduction in passenger-minutes of delay.

To summarize, we formulate slot exchanges within Model APCDM by incorporating (4.1) and (4.2), and optionally, (4.3).

4.3 Modeling Continuing Flights

In this section we address the issue of *continuing flights*. Given a pair of flights f_1 and f_2 where flight f_2 must necessarily follow flight f_1 (designated by $f_1 \rightarrow f_2$), we need to ensure that the departure time of flight f_2 is no earlier than the arrival time of flight f_1 at the connecting airport. (This is also relevant when considering multiple GDP-restricted airports.) Toward this end, let us define:

$\tau^{\text{dep}}_{fp}, \tau^{\text{arr}}_{fp}$: Respectively, the departure and arrival times of flight plan $(f,p), p \in P_f, f = 1, \ldots, F$.

Observe that if for some pair of continuing flights f_1 and f_2, where $f_1 \rightarrow f_2$, we have that

$$\max_{p \in P_{f_1}} \left\{ \tau^{\text{arr}}_{f_1 p} \right\} \leq \min_{p \in P_{f_2}} \left\{ \tau^{\text{dep}}_{f_2 p} \right\}, \tag{4.4}$$

then we do not need to further restrict the choices of flight plans for f_1 and f_2, except that we need to tie their cancellation surrogate plans by equating $x_{f_1 0} = x_{f_2 0}$. Hence, let us define the set

$$F^{\text{cont}} = \{(f_1, f_2) : f_1 \rightarrow f_2, \text{ where } f_1, f_2 \in \{1, \ldots, F\} \text{ are such that (4.4) does not hold}\}.$$

Accordingly, we incorporate the following set of constraints within APCDM:

$$\sum_{p \in P_{f_1}} \tau^{\text{arr}}_{f_1 p} x_{f_1 p} \leq \sum_{p \in P_{f_2}} \tau^{\text{dep}}_{f_2 p} x_{f_2 p}, \quad \forall (f_1, f_2) \in F^{\text{cont}} \tag{4.5a}$$

$$x_{f_1 0} = x_{f_2 0}, \quad \forall (f_1, f_2) \in F^{\text{cont}}. \tag{4.5b}$$

Remark 4.1. Note that in lieu of (4.5), we could have combined f_1 and f_2 into a single "block-flight" f_{12}, say, having $|P_{f_1}|\,|P_{f_2}|+1$ surrogate flight plans (where the "+1" represents the cancellation surrogate). However, this might entail a substantial increase in flight plans, and moreover, would preclude (or complicate) the consideration of multiple jointly connecting

flights such as $f_1 \rightarrow f_2$ and $f_3 \rightarrow f_2$, besides inconveniencing data manipulations. Hence, we utilize (4.5).

Remark 4.2. We can tighten the model representation by incorporating the following set of valid inequalities, which are implied by (4.5) and Constraint (2.1b):

$$x_{f_1 p} \leq \sum_{\substack{p' \in P_{f_2}: \\ \tau^{\text{dep}}_{f_2 p'} \geq \tau^{\text{arr}}_{f_1 p}}} x_{f_2 p'}, \quad \forall p \in P_{f_1}, \quad \forall (f_1, f_2) \in F^{\text{cont}} \tag{4.6a}$$

$$x_{f_2 p} \leq \sum_{\substack{p' \in P_{f_1}: \\ \tau^{\text{arr}}_{f_1 p'} \leq \tau^{\text{dep}}_{f_2 p}}} x_{f_1 p'}, \quad \forall p \in P_{f_2}, \quad \forall (f_1, f_2) \in F^{\text{cont}}. \tag{4.6b}$$

Observe that any one of the constraint sets (4.5a), or (4.6a), or (4.6b) by itself correctly enforces the precedence relationship $f_1 \rightarrow f_2$ in concert with (4.5b), but jointly, they assist in tightening the continuous relaxation of the model representation. Hence, we incorporate (4.6) into the model to accompany (4.5).

4.4 Equity Constraints

When addressing equity issues, Vossen et al. (2003), Tadenuma (2002), and others agree that efficiency and equity habitually conflict in that attempts to improve equity amongst the participants vying for a resource typically results in a reduction in the efficient distribution of that resource. The original APCDM model attempts to trade off efficiency and equity through the objective function (2.1a), where the equity related terms are based on certain collaboration efficiency and collaboration equity functions defined for each airline as described below. The general framework given below will be used as a foundation for the original APCDM equity concept as well as for proposing two alternative equity formulations in this section.

Each of these equity measures is based on a specifically defined *relative performance*

ratio $d_\alpha(x)$ (e.g., delay per passenger), for each airline $\alpha \in \{1, \ldots, \overline{\alpha}\}$, as a linear function of the binary flight plan choice variables x (vector of the x_{fp}-variables). Associated with this we define a particular *collaboration efficiency function* $E_\alpha(x) \in [0, 1]$, which is also linear in x, where $E_\alpha(x) = 1$ represents the best possible outcome (a 100% efficiency). Contingent of these functions, we then define the *collaboration equity function* as the deviation of the efficiency value from its (weighted) mean as given by:

$$E_\alpha^{\text{equity}}(x) = E_\alpha(x) - \left(\sum_{\alpha=1}^{\overline{\alpha}} \omega_\alpha E_\alpha(x) \right), \tag{4.7}$$

where $\omega_\alpha > 0$ is a weight factor ascribed to airline $\alpha, \forall \alpha = 1, \ldots, \overline{\alpha}$, with $\sum_{\alpha=1}^{\overline{\alpha}} \omega_\alpha = 1$ (for example, ω_α might be taken in proportion to the number of flights in A_α). For efficiency and equity, we would like $E_\alpha(x)$ to be close to one and $E_\alpha^{\text{equity}}(x)$ to be close to zero, $\forall \alpha$. Accordingly, the "equity-related term" in the objective function (2.1a) is given by

$$\mu \left[\sum_\alpha \omega_\alpha \left[1 - E_\alpha(x) \right] + \sum_\alpha \omega_\alpha \left| E_\alpha^{\text{equity}}(x) \right| \right], \tag{4.8}$$

where μ is designated as $0.1 \sum_{f=1}^{F} c_f^*$ as motivated by Sherali, Staats and Trani (2006). In addition, for a specified constant $E_{\max}^{\text{equity}}$, (designated as $0.07/\overline{\alpha}$ by Sherali, Staats and Trani 2006), Constraint (f) of APCDM also restricts

$$\omega_\alpha \left| E_\alpha^{\text{equity}}(x) \right| \leq E_{\max}^{\text{equity}}, \quad \forall \alpha = 1, \ldots, \overline{\alpha}. \tag{4.9}$$

The three alternative equity concepts proposed next in Sections 4.4.1-4.4.3 specify different relative performance ratios $d_\alpha(x)$ and their related efficiency functions $E_\alpha(x)$. Constraint set (d) of APCDM is then embodied by the respective formulas for $d_\alpha(x)$ and $E_\alpha(x)$ and their stated bounding relationships, along with Equations (4.7) and (4.9). The first of these equity measures is from Sherali, Staats and Trani (2006), whereas the other two are

new and focus on delays, assuming that the wind/weather optimized trajectory costs are compatible among the different surrogate plans for each flight.

4.4.1 Equity Method 1 (EM1)

This method, adopted by Sherali, Staats and Trani (2006), defines the *relative performance ratio* as

$$d^1_\alpha(x) = \frac{\sum_{f\in A_\alpha}\sum_{p\in P_{f0}} c_{fp}x_{fp}}{\sum_{f\in A_\alpha} c^*_f}, \forall \alpha = 1,\ldots,\overline{\alpha}, \tag{4.10}$$

where (4.10) measures the total fuel and delay costs incurred by airline α as a multiple of the best possible (unconstrained) cost. Note that by virtue of the scale-invariance of the ratio in (4.10), airlines could internally prioritize their own flights in this regard by appropriately weighting or adjusting the relative c_{fp}-values. Sherali, Staats and Trani (2006) restrict $d^1_\alpha(x) \leq d^1_{\max}, \forall \alpha$, where $d^1_{\max} = 1.2$ is selected based on some empirical sensitivity analyses. Contingent on this ratio, we define the *collaboration efficiency* as a linear function for each airline α according to

$$E^1_\alpha(x) = \frac{d^1_{\max} - d^1_\alpha(x)}{d^1_{\max} - 1}, \forall \alpha = 1,\ldots,\overline{\alpha}, \tag{4.11}$$

so that the efficiency $E^1_\alpha(x) = 1$ if $d^1_\alpha(x) = 1$, and $E^1_\alpha(x) = 0$ if $d^1_\alpha(x) = d^1_{\max} \equiv 1.2$ (i.e., exceeds the minimal possible total cost by 20%).

4.4.2 Equity Method 2 (EM2)

The *relative performance ratio* in this case measures the total *average delay realized per passenger* and is given by

$$d^2_\alpha(x) = \frac{\sum_{f \in A_\alpha} (PAX_f) DL^{CDM}_f(x)}{\sum_{f \in A_\alpha} (PAX_f)}, \forall \alpha = 1, \ldots, \overline{\alpha}, \tag{4.12}$$

where $DL^{CDM}_f(x) \equiv \sum_{p \in P_{f0}} \delta_{fp} x_{fp}$ is the CDM-realized delay function for flight f, and where PAX_f represents the aforementioned passenger count estimate, $\forall f$ (McCrea 2006). In lieu of using such passenger count estimates based on historical load factors, which might be controversial for airlines, and also, because FAA subscribes to the general policy of treating all aircraft uniformly, we could assume that $PAX_f = 1, \forall f$, so that the focus in this equity measure would then be on the *average aircraft delays*, as opposed to the average delay per passenger. As another option, we could treat PAX_f as a normalized ordinal weight provided by airlines for prioritizing their different flights based on criticality with respect to downstream operations (and/or profits), if this information is available. In this lattermost case, any relative scale can be used since the constraints described below are all scale-invariant, and $d^2_\alpha(x)$ would then represent a *weighted average delay per flight* for each airline α. We note here that this could also be problematic in that it requires typically guarded information, but might be workable if such information regarding priority weights is provided to the FAA in a confidential parameter data file to be perceived and utilized by the model only. Using any of these alternative interpretations, we define the collaboration efficiency in this method as the linear function

$$E^2_\alpha(x) = \frac{d^2_{\max} - d^2_\alpha(x)}{d^2_{\max}}, \forall \alpha = 1, \ldots, \overline{\alpha}, \tag{4.13}$$

where $d^2_{\max}$ is given as follows, based on some preliminary empirical analysis (see McCrea 2006):

$$d^2_{\max} = 1.5\Big(\min_{\alpha=1,\ldots,\overline{\alpha}} \big\{ d^2_\alpha(x) : x \text{ corresponds to selecting the highest delay surrogate option for each flight} \big\} \Big).$$

Hence, the efficiency $E^2_\alpha(x) = 1$ if $d^2_\alpha(x) = 0$, and $E^2_\alpha(x) = 0$ if $d^2_\alpha(x) = d^2_{\max}$, where we also restrict $d^2_\alpha(x) \leq d^2_{\max}$.

4.4.3 Equity Method 3 (EM3)

This method is inspired by the discussion in Vossen and Ball (2006b), which indicates that a feature of important relevance to airlines is *on-time performance* (e.g., delays not exceeding 15 minutes). In this case, we define a binary on-time performance index $\delta^{\text{on-time}}_{fp}$ for each flight plan $p \in P_{f0}$ of flight $f \in \{1, \ldots, F\}$ based on the estimated delay δ_{fp} and a permissible tolerance δ^{tol} (e.g., the 15-minute allowance mentioned above) according to

$$\delta^{\text{on-time}}_{fp} = \begin{cases} 1 \text{ if } \delta_{fp} \leq \delta^{\text{tol}} \\ 0 \text{ otherwise.} \end{cases}$$

Accordingly, the relative performance ratio and collaboration efficiency functions for this method are defined as follows:

$$d^3_\alpha(x) = \frac{\sum_{f \in A_\alpha} \sum_{p \in P_{f0}} \delta^{\text{on-time}}_{fp} x_{fp}}{|A_\alpha|} \tag{4.14}$$

and

$$E^3_\alpha(x) = d^3_\alpha(x), \quad \forall \alpha = 1, \ldots, \overline{\alpha}. \tag{4.15}$$

4.5 Enhanced Formulation of APCDM, APCDM-Light, and Algorithmic Issues

The enhanced version of APCDM inherits the following additional constraints to augment the model delineated in Equation (2.1):

- Constraints representing slot exchanges, given trade offers (Equations (4.1), (4.2), and optionally (4.3)).
- Constraints for tying-in continuing flights (Equations (4.5) and (4.6)).

For the sake of simplicity and computational expediency from the viewpoint of airspace flow programs, we also study a reduced version of APCDM, which we refer to as **APCDM-Light**. This model has the following characteristics: it retains within the objective function (2.1a) only the first term (total system fuel, delay, and cancellation costs) along with the efficiency-based first term from (4.8); it imposes only the basic sector (or FCA) capacity restrictions in Constraints (2.1c); it eliminates the above mentioned slot-exchange constraints (4.1), (4.2), and (4.3)), as well as omits the connection constraints (4.5) and (4.6) (assuming that, barring slot exchanges, the delays associated with the different alternative flight plans are compatible with flight connections; else, the relevant constraints (4.5) and (4.6) may be incorporated); and it retains Constraints (2.1b) and (2.1g) - (2.1o).

In a direct implementation, we employ the MIP option in CPLEX 11.1 to solve APCDM to (ϵ-)optimality, where we explore $0 \leq \epsilon \leq 5\%$ in Section 4.6 to assess the effect of ϵ on the quality of the solution produced and the required effort.

4.6 Computational Experience

We now provide results of several computational experiments. The enhanced APCDM model, the different supporting modules that preprocess the raw data for providing the requisite

inputs for APCDM, and the APCDM-Light model were all coded from scratch as stand-alone software to facilitate their use in the NextGen project. As we shall see in the sequel, this re-coding has resulted in a far more efficient software that has enabled the solution of substantially larger problems with reasonable computational effort, in comparison with the results reported in McCrea (2006), despite the more complex features modeled herein. In Section 4.6.1 we present a realistic test case derived from the FAA's Enhanced Traffic Management System (ETMS), and we use this test set to investigate the effects of the different equity methods given in Section 4.4, including the consequences with respect to solution equity when slot exchanges are considered. We also demonstrate the impact of the sector workload and conflict resolution constraints on the permissible slot exchanges. Finally, Section 4.6.2 studies the computational effort required to solve APCDM and APCDM-Light for randomly generated problems of various sizes, and presents insights into the effect of utilizing different equity measures. All computations were performed on a Dell Precision T7400 workstation with an Intel Xeon E5410 2.33 GHz CPU having 3.25 GB of RAM, and using CPLEX 11.1 to solve the different optimization problems.

4.6.1 Illustrative Test Case

We constructed a test scenario using data obtained from the FAA based on the Enhanced Traffic Management System (ETMS) flight information pertaining to the Miami and Jacksonville Air Route Traffic Control Centers (ARTCCs). From this data, we selected flight trajectories that traverse an FCA comprised of a subset of the 88 sectors that define the two aforementioned ARTCCs. Some of the arrival times were modified slightly to account for specific slot allocations.

The FAA (2004) Benchmark Report for MIA details the airport's reported capacity under optimal, marginal, and Instrument Flight Rules (IFR) operating conditions. The test set was generated to study the case in which MIA is operating within IFR maximum capacity rates (40 arrivals and 40 departures per hour). Each of the flights in this set was ascribed six

surrogate flight plans that represent alternative routes as well as revised arrival and departure times as necessary, based on imposed ground delays. The test set also features eight AMAL trade offers that we used to demonstrate the effects of slot exchanges on overall system cost and solution equity. The details of these trade offers, including passenger estimates, are found in Table 4.1. A summary of the results, including the mean collaboration equity (MCE) across all airlines (using equal weights) and the total NRPM for the system, is provided in Table 4.2. Recall from Section 4.4 that values of collaboration efficiency closer to one, and values of collaboration equity closer to zero, are more desirable. CPLEX returned solutions after 20 minutes or when the LP/IP gap reached 0.01%, whichever came first.

Table 4.1: APCDM trade offers.

	Induce Delay				Reduce Delay			
		Scheduled	Delay Until			Scheduled	Move Up To	
Airline	Flight	Arrival	At Most	PAX	Flight	Arrival	At Least	PAX
AAL	1	600.0	610.5	98	3	616.5	615.0	160
AAL	8	642.0	649.5	111	10	652.5	651.0	200
COA	2	630.0	637.5	118	3	636.0	630.0	170
DAL	1	606.0	613.5	122	2	610.5	609.0	179
NWA	1	601.5	612.0	142	3	621.0	619.5	230
NWA	4	645.0	654.0	155	5	651.0	646.5	320
UAL	1	607.5	615.0	95	2	609.0	606.0	215
USA	1	612.0	621.0	112	2	618.0	616.5	217

Comparison of Equity Methods

Columns (a)-(c) in Table 4.2 display collaboration efficiencies and NRPM values for the different airlines at the optimal solution under each of the three equity methods when applying Constraint (4.3), and Columns (d)-(f) provide the corresponding information when Constraint (4.3) is not applied. The collaboration efficiency values are normalized to allow for more direct comparisons. Equity Method 2 consistently yields the most equitable solutions across all airlines, both with and without Constraint 4.3. Applying Constraint 4.3 ensures airline acceptability within the slot exchanges by way of nonnegative reduction in overall passenger delays for each airline; however, it is noteworthy that the overall collaboration eq-

uity measure and the collaboration efficiency for most airlines improves when this constraint is ignored. In addition, the objective value decreased by 1.04% under EM1, which is the emphasis of EM1, with decreases of 0.21% and 0.30% under EM2 and EM3, respectively.

Effect of Slot Exchanges

We next studied the impact of including slot exchanges in the APCDM modelling construct by solving variants of APCDM with and without certain constraint sets and examining the results (see Columns (g)-(o) in Table 4.2). Ignoring sector workloads (Columns (g)-(i)) rendered slot exchanges unnecessary for AAL, i.e., no AAL trades were accepted while the model generally prescribed more overall efficient and equitable solutions in this capacity-free environment. (Note that the objective function compromises between cost, efficiency, and equity.) Omitting the conflict resolution constraints also impacts the solution with respect to slot trades (Columns (j)-(l)). One of the traded flight plans in this case that belongs to COA is actually involved in a fatal conflict, which goes undetected without imposing the conflict resolution constraints. This underscores the importance of considering conflict risks and conflict resolution workload limits in selecting a mix of flight plans. Optimizing the test case without permitting slot exchanges (Columns (m)-(o)) somewhat affected the aforementioned compromise between cost, efficiency, and equity under EM2 and EM3, but rendered the test problem infeasible when using EM1 because the value of $d^1_{\max} = 1.2$ was too stringent to permit a feasible solution in the absence of slot exchanges. In other words, the average cost to at least one airline would have been more than 20% higher than the minimum possible cost for that airline. Moreover, the solutions obtained using EM2 and EM3 produced objective values that were 3.44% and 4.16% higher, respectively. As expected, permitting slot exchanges between airlines seems to allow for more overall cost effective, although not necessarily more equitable, solutions.

Table 4.2: Collaboration efficiency and NRPM by airline.

	Column Labels														
	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)	(m)	(n)	(o)
Applied Constraints															
- Sect. Wkld.	*	*	*	*	*	*				*	*	*	*	*	*
- Conflict Res.	*	*	*	*	*	*	*	*	*				*	*	*
- Slot Exch.	*	*	*	*	*	*	*	*	*	*	*	*			
- NRPM ≥ 0	*	*	*				*	*	*	*	*	*			
- Eq. Method	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
Normalized Collaboration Efficiency													**(infeas)**		
- AAL	0.960	1.000	0.866	0.998	1.000	0.912	1.000	1.000	0.866	0.963	1.000	0.866	-	0.985	0.898
- COA	0.917	1.000	1.000	0.947	1.000	0.955	0.955	1.000	1.000	0.978	1.000	1.000	-	0.987	1.000
- DAL	1.000	0.992	0.955	0.998	0.992	0.929	0.963	0.992	0.883	0.898	0.992	0.955	-	0.990	0.974
- NWA	0.937	1.000	0.909	0.991	1.000	0.955	0.954	1.000	0.909	1.000	1.000	0.909	-	1.000	1.000
- UAL	0.982	1.000	0.921	1.000	1.000	0.967	0.962	1.000	0.988	0.922	1.000	0.921	-	0.971	0.945
- USA	0.894	0.990	0.955	0.924	0.990	1.000	0.883	0.990	0.955	0.961	1.000	0.955	-	0.987	0.962
MCE	**0.032**	**0.004**	**0.035**	**0.027**	**0.004**	**0.022**	**0.023**	**0.004**	**0.047**	**0.029**	**0.002**	**0.035**	**-**	**0.006**	**0.028**
Net Reduction in Passenger Minutes															
- AAL	0	501	501	-621	-48	99	0	0	0	501	501	246			
- COA	1863	1332	1332	1863	1332	1332	1332	1863	1863	312	0	0			
- DAL	879	86	879	-12	0	86	342	86	86	1428	86	1428			
- NWA	51	1769	1769	1717	1718	1718	0	51	51	917	1718	1769			
- UAL	1650	1185	0	1507	1793	1650	360	1185	1185	503	360	1793			
- USA	1449	620	620	1449	-504	-504	1449	620	620	1124	0	1124			
System	**5892**	**5492**	**5100**	**5904**	**4290**	**4380**	**3483**	**3804**	**3804**	**4784**	**2664**	**6359**			

4.6.2 Computational Effort Analysis

To examine the computational effort required to solve APCDM and APCDM-Light we generated 120 random problems. All of the problems included flights between the 40 busiest airports in the United States through a notional airspace comprising 26 sectors. In an effort to make these random problems as realistic as possible with respect to actual traffic patterns, we sampled ETMS data for flights between these airports and scheduled the flights accordingly. The 40 busiest airports were considered in this process to allow a wide variety of potential flights. The relatively low number of sectors when compared to the actual NAS was chosen for two reasons: (1) to emulate a smaller FCA region and (2) to increase the likelihood that sector capacity constraints would be active. In each of the problems, we designated the number of flights considered (125, 250, 500, or 1000 flights, with 30 problems each), and each flight was assigned a random number of surrogate flight plans (20% with two surrogates, 50% with three, and 30% with four). We formulated the models APCDM and APCDM-Light and obtained solutions to varying levels of optimality (0.01%, 1%, and 5%).

Table 4.3 provides details on the computational effort required (averaged over the 30 instances of each size) to optimize APCDM to different optimality tolerances, along with the average quality of the resulting solutions. Here, we refer to *solution quality* as the percent difference between the final objective function value produced and the best known objective function value for that problem instance. The average time required to generate the constraints is also listed for each problem size. Note that the runs in Tables 4.3 and 4.4 used equity method EM2. This was the only equity method for which there existed a feasible solution to each of the randomly generated problems, and as seen in Section 4.6.1, it was the most robust with respect to slot exchanges. EM1 was the most restrictive, producing 15 infeasible instances. These instances could be made feasible by suitably increasing the value of $d^1_{\max}$. EM3 produced three infeasible instances, which are more difficult to resolve because of the discrete manner in which the collaboration efficiency ($E^3_\alpha(x)$) is computed in this case. Since on-time performance is a discrete value, there might exist no solutions in which the resulting collaboration equity lies within the imposed bounds. A possible resolution might be to increase (or relax) $E^{\text{equity}}_{\max}$ and depend primarily on the objective function penalties to achieve equity. Each run was terminated after 20 minutes, and the number that exceeded this time limit (which occurred only with the tight optimality tolerance of 0.01%) is provided in the table. Nonetheless, the solution for every problem that reached the 20 minute time limit under EM2 was within 1% of optimality. (Results using all three equity methods are compared subsequently.)

It is noteworthy that the solutions obtained for APCDM when CPLEX exits within the specified tolerance of 1% of optimality are actually of a relatively high quality (ranging from 0.161% - 0.267% of the best known solution value, on average), and are generally obtained much faster in comparison with the results derived using CPLEX's default optimality tolerance of 0.01%. However, while increasing the optimality tolerance to 5% substantially reduces the solution effort, the solution quality is significantly degraded.

Table 4.4 provides similar average solution time and quality results for APCDM-Light using different optimality tolerances. Here again, *solution quality* refers to the percentage

Table 4.3: APCDM solution time and quality (Equity Method 2).

		Optimality Tolerance					
	Constraint	0.01%		1%		5%	
	Generation	Solution	# Exceeded	Solution	Solution	Solution	Solution
Flights	Time (sec)	Time (sec)	Time Limit	Time (sec)	Quality (%)	Time (sec)	Quality (%)
125	0.184	4.204	0	1.256	0.161	0.349	1.904
250	0.511	217.999	2	3.320	0.166	0.586	2.309
500	1.518	859.028	18	5.620	0.196	1.771	2.465
1000	5.427	1200.000	30	28.696	0.267	9.281	2.213

deviation in the objective function value with respect to the best known objective value for solving that particular instance of APCDM-Light. In this case, solutions to within 0.01% of optimality are obtained expeditiously.

Table 4.4: APCDM-Light solution time and quality (Equity Method 2).

		Optimality Tolerance				
	Constraint	0.01%	1%		5%	
	Generation	Solution	Solution	Solution	Solution	Solution
Flights	Time (sec)	Time (sec)	Time (sec)	Quality (%)	Time (sec)	Quality (%)
125	0.134	0.032	0.034	0.001	0.038	0.001
250	0.365	0.050	0.049	0.000	0.063	0.126
500	1.082	0.081	0.090	0.004	0.092	0.335
1000	4.088	0.174	0.173	0.024	0.185	0.503

Tables 4.5 and 4.6 present comparisons of the computational effort required to solve APCDM and APCDM-Light, respectively, using each of the three equity methods and employing the specified optimality tolerances (1% for APCDM and 0.01% for APCDM-Light). In addition to the average solution times, these tables include the minimum and maximum times required to solve each problem size. As before, times are reported only for feasible problem instances. Note that some problem instances modeled using EM3 were particularly challenging; even three of the APCDM-Light runs using EM3 reached the 20 minute time limit. Perhaps, because of the discrete nature of EM3, there exist several alternative (near-) optimal solutions that the branch-and-bound solver enumerates in the search process before being able to verify optimality.

Table 4.5: APCDM solution time by equity method (1% opt. tolerance).

	Equity Method 1			Equity Method 2			Equity Method 3		
Flights	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.
125	0.062	0.125	0.082	0.218	8.625	1.256	0.859	233.258	16.769
250	0.094	0.625	0.273	0.375	21.187	3.320	4.171	1118.090	107.578
500	0.156	16.844	3.422	0.921	25.375	5.620	1.265	1200.000	215.550
1000	0.390	1200.000	342.598	2.625	221.400	28.696	3.469	1200.000	159.817

Table 4.6: APCDM-Light solution time by equity method (0.01% opt. tolerance).

	Equity Method 1			Equity Method 2			Equity Method 3		
Flights	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.
125	0.031	0.094	0.045	0.015	0.062	0.032	0.031	0.047	0.034
250	0.047	0.125	0.074	0.031	0.109	0.050	0.046	0.234	0.068
500	0.078	74.170	2.782	0.062	0.110	0.081	0.093	1200.000	80.103
1000	0.188	65.467	2.632	0.140	0.235	0.174	0.188	1200.000	80.208

Chapter 5

A Sequential Fixing Heuristic and Implementation Issues

Whereas the results in Chapter 4 demonstrate a capability to solve moderate to large problem instances in a reasonable amount of time, there is still a need for developing a more effective methodology to solve the more detailed full APCDM model when the number of flights exceeds 1000 (see Table 4.3). Accordingly, we design a heuristic solution procedure in this chapter for the APCDM to handle such large-sized instances.

In Section 5.1 we describe the proposed *sequential variable fixing heuristic* (SFH) in which the APCDM is optimized by partitioning flights into suitable subsets. Then, in Section 5.2, we study the integration of APCDM as a *Tier-2* model within the NextGen project by designing a *dynamic, rolling-horizon framework* implementation. Here we apply the model on some periodic basis (e.g., hourly), where each sequential run of the model has certain flight plan selections that are fixed (such as flights that are already airborne), while we consider the selection among alternative flight plans for certain other flights in a look-ahead horizon (e.g., two hours).

5.1 A Sequential Fixing Heuristic

As an alternative to a direct implementation of the MIP option in CPLEX 11.1 to solve APCDM to (ϵ-)optimality, we propose and implement in this section a *sequential variable-fixing heuristic* (**SFH**) to solve APCDM effectively for large-scale instances. (A similar procedure can be applied to APCDM-Light, although this model is itself more readily solvable directly, even for large-scale problems.)

Given the estimated arrival time data information $\left(\tau_{fp}^{\text{arr}}\right)$ for the different flight plans $(f,p), p \in P_f, f = 1, \ldots, F$, let

$$\tau_{f*}^{\text{arr}} \equiv \min_{p \in P_f} \left\{\tau_{fp}^{\text{arr}}\right\}, \quad \forall f = 1, \ldots, F,$$

and accordingly, construct a list L of the flights $f = 1, \ldots, F$ arranged in nondecreasing values of τ_{f*}^{arr}. Partition L (in its given order) into subsets $L_1, \ldots, L_U, U \geq 1$, such that $|L_u|$ is roughly equal to $F/U, \forall u = 1, \ldots, U$ (we advocate $F/U \approx 30$). Denote

$$L_u^* = \bigcup_{u'=1}^{u} L_{u'}, \quad \forall u = 1, \ldots, U.$$

Now, let $\mathbf{Alg}[L_u^*, \epsilon]$ connote an application of CPLEX-MIP to solve APCDM to $\epsilon\%$ of optimality, while restricting $x_{fp}, \forall p \in P_f, f \in L_u^*$ to be binary-valued (with x_{fp} fixed at designated binary values as described below, $\forall p \in P_f, f \in L_{u-1}^*$, where $L_0^* \equiv \emptyset$), and while relaxing the remaining x-variables $x_{fp}, \forall p \in P_f, f \in L \setminus L_u^*$ to be continuous (nonnegative) variables. Note that by including the cancellation surrogate (possibly a dummy one if this is not a real option) and adjusting the right-hand-side of (4.9) if necessary, we assure that each updated version of APCDM solved below is feasible.

5.1.1 Heuristic SFH

Initialization. Set $u = 1$ and choose $0 \leq \epsilon \leq 5\%$.

Step 1. Run **Alg**$[L_u^*, \epsilon]$, and let $\overline{x}$ be the solution obtained.

Step 2. If $\overline{x}$ is binary-valued, then stop with this as the prescribed solution.

Step 3. Else, update APCDM by permanently fixing $x_{fp} \equiv \overline{x}_{fp}, \forall p \in P_f, f \in L_u^*$. Replace $u \leftarrow u + 1$, and return to *Step 1.*

Note that once we get $u = U$ in Heuristic SFH, we shall necessarily stop at *Step 2.* Of course, it is possible for this procedure to terminate earlier than this juncture.

5.1.2 Computational Experience with SFH

We now provide some computational experience for implementing the sequential fixing heuristic (SFH) described in the previous section. We attempted partitioning the list of flights L into U subsets where $L/U \approx 30$. It is pertinent to note that the sequential fixing of variables in such relatively small subsets can sometimes induce infeasibility in later iterations of the heuristic, when currently relaxed variables are subsequently enforced to be binary-valued. Indeed, as expected, we observed this phenomenon to occur most often when the optimality tolerance used for the subproblems was larger (1% or 5%), but it also occurred when the subproblems were solved to within 0.01% of optimality. Increasing the size of the subsets resulted in fewer infeasibilities, but processing times were naturally increased. Equity Method 2 seemed to be the most robust with respect to SFH. Under EM2, the heuristic was unsuccessful only once (with 1,000 flights and $\epsilon = 0.01$). The values for solution time and quality when optimizing APCDM using the SFH are provided in Table 5.1. Comparing Tables 4.3 and 5.1, we see that using the heuristic SFH with a 0.01% optimality tolerance yields comparable results with solving APCDM with a 1% optimality tolerance.

We also generated five larger problems, each including 3000 flights. These problems included 26 sectors, and we solved each assuming four different sector capacity levels. The first instance for each problem was solved assuming an unlimited sector capacity for each

Table 5.1: APCDM solution time and quality using SFH (Equity Method 2).

	Optimality Tolerance					
	0.01%		1%		5%	
Flight	Solution Time (sec)	Solution Quality (% above optimal)	Solution Time (sec)	Solution Quality (% above optimal)	Solution Time (sec)	Solution Quality (% above optimal)
125	0.442	0.451	0.300	0.702	0.156	4.406
250	1.591	0.502	0.584	0.837	0.408	7.590
500	4.932	0.387	1.531	1.577	1.336	11.073
1000	20.519	0.358	46.321	3.523	6.124	19.601

sector, and each subsequent instance was made more difficult by lowering the uniform sector capacity to 24, then 22, and finally 20. We solved each problem without SFH, imposing a two-hour time limit and an optimality tolerance of 1%. We then solved each problem using SFH with $\epsilon = 0.01\%$. Solution time and quality data is provided in Table 5.2.

Solving APCDM using SFH was faster than CPLEX alone for all but one problem instance, and as problem difficulty increased (i.e., sector capacity decreased), the solutions returned by SFH were often superior to those obtained using CPLEX alone. Moreover, on the most difficult instances when sector capacity was equal to 20, SFH returned superior solutions for all five problems. To address the overall quality of solutions obtained using the SFH, the LP/IP gap for each of the twenty problem instances was less than 2%, so the SFH solutions were always within 2% of optimal.

Table 5.2: Solution time and quality using SFH on Larger Problems (Equity Method 2).

	Sector Capacity											
	Unlimited			24			22			20		
Problem	Solution Time (s)	SFH Time (s)	SFH Pct Improved	Solution Time (s)	SFH Time (s)	SFH Pct Improved	Solution Time (s)	SFH Time (s)	SFH Pct Improved	Solution Time (s)	SFH Time (s)	SFH Pct Improved
1	1564.19	160.55	-0.19	7201.61	179.65	36.48	7201.53	664.73	35.17	7201.75	178.34	36.38
2	742.91	159.51	-0.17	2979.95	146.29	-0.26	1555.60	142.75	-0.27	7201.17	249.62	30.28
3	889.11	135.02	-0.85	7201.63	127.42	41.61	7201.59	180.69	7.57	7201.32	180.14	36.74
4	813.40	74.31	-0.07	195.71	77.14	-0.03	7201.01	75.89	30.74	6675.95	90.73	0.15
5	53.79	77.61	-0.11	353.54	69.07	-0.40	196.01	67.31	-0.30	392.09	82.81	0.04

5.2 Implementing APCDM in a Dynamic Rolling-Horizon Framework

We study the integration of APCDM as a *Tier-2* model within the NextGen project by designing its implementation in a *dynamic, rolling-horizon framework* (DRHF). Here we apply the model on some periodic basis (e.g., hourly), where each sequential run of the model has certain flight plan selections that are fixed (such as flights that are already airborne), while we consider the selection among alternative flight plans for certain other flights in a look-ahead horizon (e.g., two hours). A subset of new flight plans corresponding to certain imminent flights as prescribed by the model output are then chosen for implementation, and the horizon is advanced for the next run with updated information. This framework additionally permits air traffic managers to optimize larger problem instances by focusing on more imminent routing decisions, while incorporating a sufficient look-ahead time-window to accommodate a subsequent set of decisions that might interact with, and thereby influence, the currently relevant decisions.

Note that on a tactical level, we envisage that the proposed model would be executed in a DRHF on, say, an hourly basis. Here, in each sequential or periodic run of the model, certain flight plan selections might be fixed (for some of both airborne flights and imminent, not-as-yet-departed flights), while for a set of other flights covering a specified affected region over a two-hour look-ahead horizon, we would consider the selection among alternative flight plans as usual, based on the most recent available weather data. A subset of new flight plans corresponding to certain imminent flights as prescribed by the model output would then be chosen for implementation, and the horizon would be advanced for the next run with updated weather information.

In addition to the operational advantages gained by having access to more recent weather information, implementing APCDM in a DRHF should be advantageous with respect to computational times. Specifically, the conflict detection routines AEM and PAEM are

based on pairwise comparisons between plans from different flights. If a particular flight plan has previously been fixed, there is no need to determine potential conflicts for any other surrogate flight plans. So, for example, if some Flight A originally had 4 surrogate flight plans and Flight B has 3, then in a direct implementation of APCDM we would analyze 12 pairs of routes to determine the existence of fatal or resolvable conflicts. However, if the plan for Flight A was fixed in some previous horizon, we need only analyze 3 pairs of flights.

5.2.1 Equity Considerations in a Dynamic, Rolling-Horizon Framework

Executing the model in a DRHF can present difficulties with respect to Equity Method 1 (see Section 4.4.1). If a particular flight f is fixed, there is only one surrogate plan p for that flight and we will have $c_{fp} = c_f^*$. This might cause an unrealistically low relative performance ratio $d_\alpha^1(x)$ for that airline, and the resulting collaboration efficiency $E_\alpha^1(x)$ would be unrealistically high. Thus, the airline in question might be unduly penalized with respect to some other flight(s).

One way to counteract this effect in Equity Method 1 is to pass the value of c_f^* from one run to the next. This would ensure that the relative performance ratio considers the cost of the fixed flight plan relative to, perhaps, some alternative minimal cost flight plan from a previous model run. However, this would also require storing, and possibly updating, c_f^* for each flight in each run of the model. One could also counteract the effect of having $c_{fp} = c_f^*$ for fixed flight plans by designating quantities q_α^{num} and q_α^{den} for each airline to be added to the numerator and denominator, respectively, when computing the relative performance ratio (where $q_\alpha^{\text{num}} > q_\alpha^{\text{den}}$). These values should effectively adjust the numerator and denominator to reflect the appropriate cost terms to be used for the fixed flights (e.g., the aforementioned previous c_f^*-values, say $c_{f(\text{old})}$, could be accommodated by including in q_α^{den} the term $(c_{f(\text{old})} - c_{fp})$ for each fixed flight), as well as possibly incorporate cost terms pertaining to previously scheduled flights as desired (i.e., a set of other fixed flights

immediately prior to the current horizon). This approach would therefore simply introduce two additional parameters for each airline. Equity Methods 2 and 3 do not present such problems, because they are based on performance relative to the published schedule and not on performance relative to other surrogate flight plans. Nonetheless, if so desired, one could incorporate a similar memory effect into Equity Methods 2 and 3. In any event, determining suitable memory lags and effective schemes for continually updating these quantities for each sequential run would require experimentation with very specific, realistic data that included time-series data with respect to forecasted and actual weather information as well as real-time updated aircraft position information.

5.2.2 Computational Experience for the Dynamic-Rolling Horizon Framework

Here we provide the results of an experiment implementing APCDM in a DRHF. To make optimal routing decisions, air traffic managers would need to correctly forecast capacity levels before assigning routes using a program such as APCDM. However, weather is highly unpredictable, especially with the precision necessary for routing aircraft. In practice, under these circumstances, air traffic controllers sometimes employ a wait-and-see approach; so implementing APCDM in this scenario under a DRHF might be analogous in that it would allow for better capacity predictions and possibly more cost-effective routing.

Metron Aviation provided a data set in which 1936 flights are routed through a section of the national airspace during a four-hour window of interest. During this four-hour window, the capacity in each of 25 sectors is reduced because of severe weather. When solving APCDM for this scenario, only two flights are canceled due to excess air-traffic controller workload. To get a better idea of the impact that a DRHF implementation might have, we reduced the sector capacities to half of their original values, rounding the resulting capacity to the nearest integer when necessary.

We generated 100 problem instances by forcing random perturbations of the sector capacity values at uniformly distributed times throughout the scenario. The sector capacity was increased by one for approximately 25% of the sectors, decreased by one for 25%, and the capacity remained unchanged for the remaining sectors. Thus, the expected value of the change in overall capacity was zero. We solved APCDM for each instance using the forecasted capacity, and we then implemented APCDM under a DRHF, hourly, with a look-ahead horizon of two hours. Plans chosen for flights occurring within the first hour of each horizon were fixed for subsequent runs, and the revised capacities were used in the APCDM constraints if the capacity for a given sector changed before the start of the horizon. In 47 of the problems, the objective value increased, but the increase was attributable to lower sector capacities (i.e., when the nominal capacity was reduced by one). Thus, the higher objective values were actually a more accurate reflection of the scenario and the original situation would not have been feasible. For eight instances, the rolling horizon scheme resulted in higher objective values. Here, a less expensive flight plan was chosen and fixed for a flight from an earlier horizon, forcing the selection of a significantly more expensive flight plan for a later flight that would not have otherwise been selected (the same condition experienced when employing the SFH in Section 5.1.2). These increases, however, were always less than 3% of the original objective value and in only one of the eight instances did the fixed flights force a cancellation. On the other hand, objective values improved in 34 of the problem instances, because better capacity information allowed more efficient aircraft routing decisions. The objective function values remained unchanged for 11 of the problem instances.

Chapter 6

Strategic Analysis: Dynamic Airspace Sector Definitions

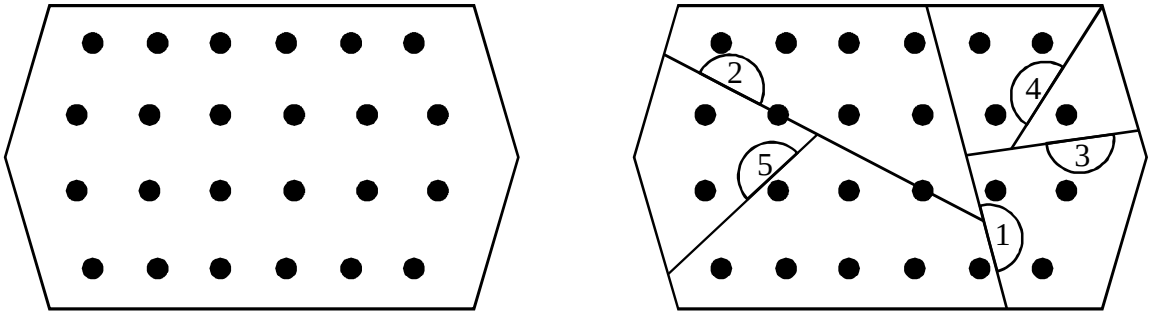
6.1 Introduction

Given the current inflexible airspace sector configuration, some air traffic controller resources might be under-utilized, while others might be over-burdened (Kopardekar et al. 2007). Reorganizing the partitioning of the airspace will allow for more effective use of scarce resources, increase safety through reduced workload stress, and facilitate certain advanced concepts such as automated separation assurance. Moreover, an efficient sectorization algorithm might allow for *dynamic airspace configuration* (DAC), in which the airspace is adjusted in real-time to accommodate demand patterns that change throughout the day.

Consider the national airspace over the continental United States (or a subset thereof) represented as a polytope (polygon in 2-D space). Figure 6.1(a) depicts this region with a set of grid points indexed by $i \in I$, where each grid point i has an associated weight $w_i, i \in I$. For example, in the context of the sectorization schema of Yousefi and Donohue (2004), the grid points might correspond to centers of hexagonal-cells, with the weights representing an

aggregate workload measure over the corresponding cell. In the spirit of Yousefi and Donohue (2004), but adopting a more computationally manageable process, we propose the following methodology to partition the given polygonal airspace into some $m \geq 2$ convex polygonal sectors that have a balanced workload measure. We begin by constructing a *separating plane* that partitions the given polygonal region in a manner that balances the sum of weights lying on either side of the plane, i.e., minimizes the absolute difference between the total weight of points lying in each of the two halfspaces defined by the constructed plane, where the points lying on the plane can be assigned to either halfspace. We refer to this as the *bisection process*. This produces two *sub-polygons*. Then, at each subsequent step, we select the sub-polygon that has the largest total weight assigned to it, and repeat the foregoing bisection process, continuing this sequential partitioning until we have produced m sectors. Note that if $m = 2^q$ for some integer q, then this would produce a near-perfectly balanced *sectorization* or segmentation of workload. Figure 6.1(b) depicts a possible sectorization for $m = 6$, following $(m-1) = 5$ bisection processes, where the sequentially generated separating planes at each step are consecutively numbered as $1, \ldots, 5$.

Remark 6.1. Note that the original region can be specified as a union of polytopes, in which case, we adopt the same procedure as above by simply treating the given polytopes as an initial partition, each having some corresponding cumulative weight. □



(a) Original Polygon (b) Resulting Sectors for $m = 6$

Figure 6.1: Sectorization strategy.

The remainder of this chapter is organized as follows. Section 6.2 describes the

concept and methodology for conducting the aforementioned bisection process. Section 6.3 then delineates four possible algorithmic variants using this basic concept, which impose certain additional restrictions for practical purposes. Section 6.5 provides computational results and illustrations for these four procedures, and Section 6.6 concludes the chapter.

6.2 Bisection Process

We begin this section by analyzing a particular restricted version of the bisection process, which will be useful in designing our different algorithmic variants, and then readily extend this to consider the general version of the bisection process.

6.2.1 Restricted Bisection Process (RBP)

At any general step of the bisection process, suppose that we are presently examining a polygon X containing assigned grid points located at (a_i, b_i) for $i \in I_X$, with associated workload measures (weights) $w_i, \forall i \in I_X$. Consider any pair of edges AB and CD of X, where A, B, C, and D are vertices of X (possibly $B \equiv C$) that occur in an ordered subset of the clockwise sequence of vertices of X (see Figure 6.2). Let α_1 and α_2 be the respective coordinates (in R^2) of the vertices A and C, and let the edges AB and CD be described by $\alpha_e + \lambda_e d_e, 0 \leq \lambda_e \leq 1$, for $e = 1$ and 2, respectively. Let $L_{AB,CD}$ denote the set of planes (lines) in R^2 that intersect both the edges AB and CD of X. Figure 6.2 depicts one such plane $\nabla \cdot x = \beta$ as a dashed line. We now define the *restricted bisection problem* **RBP** as follows:

> Among all planes in $L_{AB,CD}$, find one of the type $\nabla \cdot x = \beta$ such that this induces
> a partition $\left\{I_X^+, I_X^-\right\}$ of I_X (i.e., with $I_X^+ \cup I_X^- = I_X$ and $I_X^+ \cap I_X^- = \varnothing$),
> where $\nabla \cdot (a_i, b_i) \geq \beta, \forall i \in I_X^+$, and $\nabla \cdot (a_i, b_i) \leq \beta, \forall i \in I_X^-$, (6.1a)

and where the absolute difference:

$$\left| \sum_{i \in I_X^+} w_i - \sum_{i \in I_X^-} w_i \right| \text{ is minimized.} \tag{6.1b}$$

Denoting the components of any defined vector $v \in R^2$ as (v_1, v_2), note that the vector

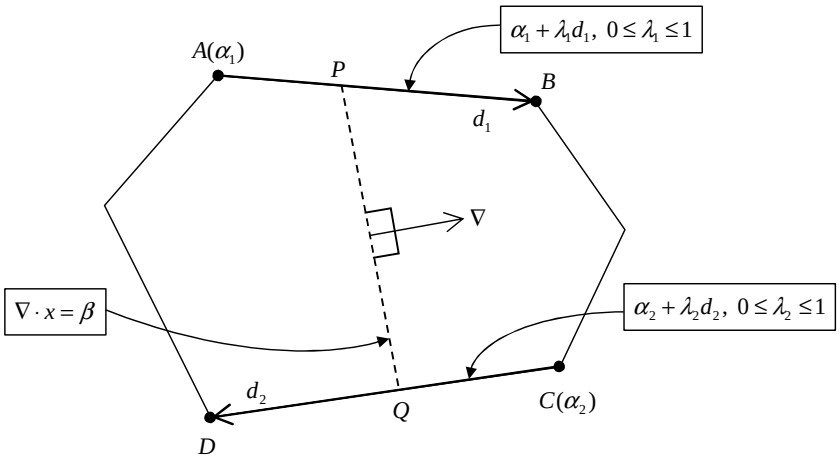


Figure 6.2: Illustration of X along with a pair of edges AB and CD.

$\overrightarrow{PQ}$ illustrated in Figure 6.2, corresponding to some $0 \le \lambda_e \le 1, e = 1, 2$, is given by $(\alpha_2 + \lambda_2 d_2) - (\alpha_1 + \lambda_1 d_1)$, and the orthogonal vector $\nabla = (\nabla_1, \nabla_2)$ that is rotated 90° anticlockwise with respect to the orientation of $\overrightarrow{PQ}$ is given by

$$\nabla = \begin{bmatrix} \nabla_1 \\ \nabla_2 \end{bmatrix} = \begin{bmatrix} (\alpha_{12} + \lambda_1 d_{12}) - (\alpha_{22} + \lambda_2 d_{22}) \\ (\alpha_{21} + \lambda_2 d_{21}) - (\alpha_{11} + \lambda_1 d_{11}) \end{bmatrix}. \tag{6.2}$$

Hence, the equation of the plane containing the line segment PQ is given by $\nabla \cdot x = \nabla \cdot (\alpha_1 + \lambda_1 d_1) \equiv \beta$, where from (6.2), we get,

$$\begin{aligned} \beta = {} & (\alpha_{12}\alpha_{21} - \alpha_{11}\alpha_{22}) + \lambda_1(\alpha_{21} d_{12} - \alpha_{22} d_{11}) \\ & + \lambda_2(\alpha_{12} d_{21} - \alpha_{11} d_{22}) + \lambda_{12}(d_{12} d_{21} - d_{11} d_{22}), \end{aligned} \tag{6.3a}$$

$$\text{where } \lambda_{12} \equiv \lambda_1 \lambda_2. \tag{6.3b}$$

Given that $0 \leq \lambda_e \leq 1, e = 1, 2$, we can use (6.2) and (6.3) to readily compute implied lower and upper bounds on $\nabla_1, \nabla_2,$ and β, denoted as follows:

$$\nabla_{j\ell} \leq \nabla_j \leq \nabla_{ju}, j = 1, 2, \beta_\ell \leq \beta \leq \beta_u. \tag{6.4}$$

For example, we can take

$$\nabla_{1\ell} = (\alpha_{12} - \alpha_{22}) + \min\{0, d_{12}\} + \min\{0, -d_{22}\} \tag{6.5a}$$

$$\nabla_{1u} = (\alpha_{12} - \alpha_{22}) + \max\{0, d_{12}\} + \max\{0, -d_{22}\} \tag{6.5b}$$

$$\nabla_{2\ell} = (\alpha_{21} - \alpha_{11}) + \min\{0, d_{21}\} + \min\{0, -d_{11}\} \tag{6.5c}$$

$$\nabla_{2u} = (\alpha_{21} - \alpha_{11}) + \max\{0, d_{21}\} + \max\{0, -d_{11}\} \tag{6.5d}$$

$$\beta_\ell = \min\{f(0,0), f(1,0), f(0,1), f(1,1)\} \tag{6.5e}$$

$$\beta_u = \max\{f(0,0), f(1,0), f(0,1), f(1,1)\} \tag{6.5f}$$

where $f(\lambda_1, \lambda_2)$ represents the expression on the right-hand side of (6.3a), and where β_ℓ and β_u are valid since the minimal and maximal values of the bilinear function f over $0 \leq \lambda_e \leq 1, e = 1, 2$, are attained at extreme point solutions of this latter set.

To further constrain ∇ and β, note that the left closed half-space of any plane in $L_{AB,CD}$ will contain the points A and D, while the right closed half-space will simultaneously contain the points B and C. That is, the following relationships must hold true:

$$\nabla \cdot \alpha_1 \leq \beta \tag{6.6a}$$

$$\nabla \cdot (\alpha_2 + d_2) \leq \beta \tag{6.6b}$$

$$\nabla \cdot (\alpha_1 + d_1) \geq \beta \tag{6.6c}$$

$$\nabla \cdot \alpha_2 \geq \beta. \tag{6.6d}$$

For defining Problem RBP, we hence characterize

$$L_{AB,CD} \equiv \{\nabla \cdot x = \beta : (\nabla_1, \nabla_2, \beta) \text{ satisfy (6.6), (6.2), (6.3a), and (with (6.3b) relaxed to) } \lambda_{12} \leq \lambda_1, \lambda_{12} \leq \lambda_2, \lambda_{12} \geq \lambda_1 + \lambda_2 - 1, \text{ and } \lambda_{12} \geq 0\}. \tag{6.7}$$

Note that the constraints imposed in (6.7) imply $0 \leq \lambda_e \leq 1, e = 1, 2$, as well as the bounds in (6.4). Accordingly, to determine the partition satisfying (6.1a), define the binary variables

$$y_i = \begin{cases} 1 & \text{if } \nabla \cdot (a_i, b_i) > \beta \\ 0 & \text{if } \nabla \cdot (a_i, b_i) < \beta \end{cases}, \forall i \in I_X, \tag{6.8}$$

with $y_i = 0$ or 1 whenever $\nabla \cdot (a_i, b_i) = \beta$ for any $i \in I_X$.

Observe that the y_i-variables can be determined *a priori* for certain grid points. As depicted in Figure 6.3, notice first that the (hollow) grid points lying in the left open half-space with respect to the plane passing through A and D will always satisfy $\nabla_1 a_i + \nabla_2 b_i < \beta$ for any plane in $L_{AB,CD}$; hence, we can fix $y_i \equiv 0$ for all such grid points. Likewise, we can fix $y_i \equiv 1$ for all grid points lying in the open half-space to the right of the plane passing through B and C.

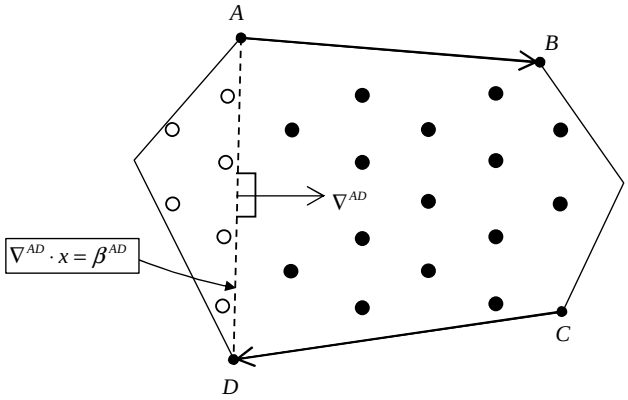


Figure 6.3: Illustration of some y_i-values that can be determined *a priori*.

To model this, construct a partition $\{I_X^0, I_X^{AB,CD}, I_X^1\}$ of I_X as follows. Compute

$$\nabla^{AD} = \begin{bmatrix} \alpha_{12} - \alpha_{22} - d_{22} \\ \alpha_{21} + d_{21} - \alpha_{11} \end{bmatrix} \text{ and } \beta^{AD} = f(0,1).$$

If $\nabla^{AD} = 0$ (whence $A = D$), then I_X^0 is empty. Otherwise, $\nabla^{AD} \cdot x = \beta^{AD}$ represents the aforementioned plane passing through A and D, and we designate $I_X^0 \equiv \{i \in I_X : \nabla_1^{AD} a_i + \nabla_2^{AD} b_i < \beta^{AD}\}$. Similarly, if $B \neq C$ (else, $I_X^1 = \varnothing$), compute

$$\nabla^{BC} = \begin{bmatrix} \alpha_{12} + d_{12} - \alpha_{22} \\ \alpha_{21} - \alpha_{11} - d_{11} \end{bmatrix} \text{ and } \beta^{BC} = f(1,0),$$

and define the set $I_X^1 \equiv \{i \in I_X : \nabla_1^{BC} a_i + \nabla_2^{BC} b_i > \beta^{BC}\}$. Then we can set $y_i \equiv 0, \forall i \in I_X^0$, and $y_i \equiv 1, \forall i \in I_X^1$, and the remaining *free grid points* constitute the elements of $I_X^{AB,CD} = I_X \setminus (I_X^0 \cup I_X^1)$.

Next, we model the two implications in (6.8) for the free grid points as the disjunctions

$$\{y_i = 1\} \vee \{\nabla \cdot (a_i, b_i) \leq \beta\} \text{ and } \{y_i = 0\} \vee \{\nabla \cdot (a_i, b_i) \geq \beta\}, \forall i \in I_X^{AB,CD},$$

and accordingly construct the convex hull representation for each disjunction over (6.4) (see Balas (1974), or Sherali and Shetty (1980), or the RLT method of Sherali and Adams (1990, 1994)). This yields the following representation, which is directly verified by Proposition 6.1 below to enforce the desired partitioning property (6.1a). Here, $\{\nabla_1, \nabla_2, \beta,$ and $y_i, i \in I_X^{AB,CD}\}$ are the *principal variables* as defined above, and $\{\nabla_j^i$ for $i \in I_X^{AB,CD}, j = 1, 2,$ and β^i for $i \in I_X^{AB,CD}\}$ are *auxiliary variables* that respectively represent the RLT product terms $\{\nabla_j y_i$ for $i \in I_X^{AB,CD}, j = 1, 2,$ and βy_i for $i \in I_X^{AB,CD}\}$, which are used to generate the required convex hull representation in a higher dimensional space.

$$\nabla_1 a_i + \nabla_2 b_i - \beta \leq \nabla_1^i a_i + \nabla_2^i b_i - \beta^i, \quad \forall i \in I_X^{AB,CD} \tag{6.9a}$$

$$\nabla_1^i a_i + \nabla_2^i b_i \geq \beta^i, \quad \forall i \in I_X^{AB,CD} \tag{6.9b}$$

$$\nabla_{j\ell}(1-y_i) \leq \nabla_j - \nabla_j^i \leq \nabla_{ju}(1-y_i) \text{ and } \nabla_{j\ell} y_i \leq \nabla_j^i \leq \nabla_{ju} y_i, \quad \forall i \in I_X^{AB,CD}, j = 1,2 \tag{6.9c}$$

$$\beta_\ell(1-y_i) \leq \beta - \beta^i \leq \beta_u(1-y_i) \text{ and } \beta_\ell y_i \leq \beta^i \leq \beta_u y_i, \quad \forall i \in I_X^{AB,CD} \tag{6.9d}$$

$$y_i \text{ binary}, \quad \forall i \in I_X^{AB,CD}. \tag{6.9e}$$

Proposition 6.1. For any feasible solution $(\nabla_1, \nabla_2, \beta)$ to (6.9), along with accompanying values of $(y_i, i \in I_X^{AB,CD}; \nabla_j^i, i \in I_X^{AB,CD}, j = 1,2; \beta^i, i \in I_X^{AB,CD})$, we have that (6.4) is satisfied, and that $\nabla_1 a_i + \nabla_2 b_i \geq \beta$ if $y_i = 1$ and $\nabla_1 a_i + \nabla_2 b_i \leq \beta$ if $y_i = 0$.

Proof. Consider any feasible solution to (6.9), and examine any $i \in I_X^{AB,CD}$. If $y_i = 1$, then (6.9c) and (6.9d) imply that $\nabla_j = \nabla_j^i, j = 1,2$, and $\beta = \beta^i$ with (6.4) holding true, and (6.9b) then yields that $\nabla_1 a_i + \nabla_2 b_i \geq \beta$ (with (6.9a) being null). Likewise, if $y_i = 0$, then (6.9c) and (6.9d) imply that $\nabla_j^i = 0, j = 1,2$, and $\beta^i = 0$ with (6.4) holding true, and (6.9a) then yields that $\nabla_1 a_i + \nabla_2 b_i \leq \beta$ (with (6.9b) being null). □

Based on these constructs, we formulate the restricted bisection problem (RBP) as the following linear mixed-integer program (MIP):

RBP: Minimize $s^+ + s^-$ (6.10a)

subject to

$$\left(\sum_{i \in I_X^{AB,CD}} w_i y_i - \sum_{i \in I_X^{AB,CD}} w_i (1-y_i) \right) + \left(\sum_{i \in I_X^1} w_i - \sum_{i \in I_X^0} w_i \right) = s^+ - s^- \tag{6.10b}$$

$(s^+, s^-) \geq 0$ (6.10c)

Equations (6.9a) – (6.9e) (6.10d)

Equation (6.2) (6.10e)

Equation (6.3a) (6.10f)

Equation (6.6) (6.10g)

$$\lambda_{12} \leq \lambda_1, \quad \lambda_{12} \leq \lambda_2, \quad \lambda_{12} \geq \lambda_1 + \lambda_2 - 1, \quad \lambda_{12} \geq 0. \tag{6.10h}$$

Based on Proposition 6.1 and Constraint (6.10d), the objective function (6.10a), in concert with Constraints (6.10b) and (6.10c), minimizes the absolute difference defined in (6.1b), where

$$I_X^+ \equiv \{i \in I_X : y_i = 1\} \text{ and } I_X^- \equiv \{i \in I_X : y_i = 0\}. \tag{6.11}$$

The remaining constraints (6.10e) – (6.10h) define the set $L_{AB,CD}$ as in Equation (6.7), where note that (6.10d) implies (6.4) by Proposition 6.1, and that (6.10h) implies the bounding restrictions $0 \leq \lambda_e \leq 1, e = 1, 2$.

6.2.2 Avoiding Degenerate and Undesirable Solutions in Problem RBP

A *degenerate solution* to Problem RBP can occur when the edges AB and CD are such that $B = C$ (see Figure 6.4(a)). The solution where $\lambda_1 = 1$ and $\lambda_2 = 0$ implies via (6.3a) that

$$\beta = (\alpha_{12}\alpha_{21} - \alpha_{11}\alpha_{22}) + (\alpha_{21}d_{12} - \alpha_{22}d_{11}).$$

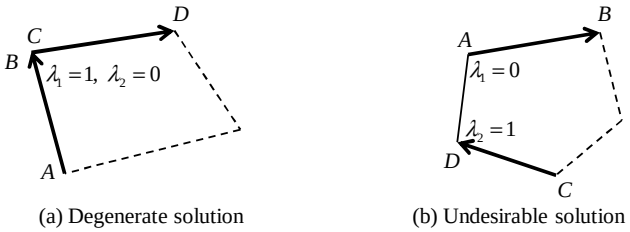
However, since $B = C$ we have $\alpha_2 = \alpha_1 + d_1$, and so,

$$\beta = \alpha_{12}(\alpha_{11} + d_{11}) - \alpha_{11}(\alpha_{12} + d_{12}) + (\alpha_{11} + d_{11})d_{12} - (\alpha_{12} + d_{12})d_{11} = 0.$$

For the same reason, (6.2) yields $\nabla = (0, 0)$, and therefore, the y_i-variables can be arbitrarily assigned binary values, since any such values will satisfy (6.9).

Another undesirable solution can occur when the two edges are separated by a single edge. For example, in Figure 6.4(b), the solution corresponding to $(\lambda_1, \lambda_2) = (0, 1)$ does not divide the polygon, but rather reproduces a side that already exists, although such a solution might yet optimize Problem RBP based on the weight-density of gridpoints lying

on the edge AD.



(a) Degenerate solution (b) Undesirable solution

Figure 6.4: Examples of degenerate and undesirable solutions for Problem RBP.

Thus, in order to prevent such solutions, we recommend constraining each λ_e to fall within $[\epsilon_1, 1 - \epsilon_2]$ for some appropriate values of ϵ_1 and ϵ_2, with $0 < \epsilon_1 + \epsilon_2 \leq 1$. For example, the following procedure validly alters the lower bound for a particular λ_e to obviate a degenerate solution. The inputs are two edges, AB and BD, and the output is a positive lower bound for the λ_e-parameter associated with the edge BD. An analogous procedure can be applied to decrease the upper bound for the λ_e-parameter associated with the edge AB. (For convenience, we let A, B, and D respectively denote the coordinate vectors representing these points.)

Procedure for Updating the λ-Lower Bound: ULLB(AB, BD, I_X):

– Define θ_i as the angle between AB and the vector $v_i = (a_i, b_i) - A, \forall i \in I_X$, and determine $i^* \in \arg\min_{i \in I_X} \{\theta_i : \theta_i > 0\}$.

– Define λ_e^* as the value of λ_e corresponding to the intersection between $B + \lambda_e \overrightarrow{BD}$ and the line containing (a_{i^*}, b_{i^*}) and A.

– If $\lambda_e^* \in [0, 1]$, whence $\lambda_e^* > 0$, update the lower bound for λ_e to impose $\lambda_e \geq \lambda_e^*$. If $\lambda_e^* < 0$ or $\lambda_e^* > 1$, set $\lambda_e^* = 1$, because there are no grid points in the triangle ABD except those that might lie on edge AB.

Likewise, to obviate an undesirable solution of the type depicted in Figure 6.4(b), noting that both A and D lie in the half-space $\nabla \cdot x \leq \beta$ of the constructed plane, we could

impose the restriction $\nabla \cdot A + \nabla \cdot D \leq 2\beta - \epsilon$ for some small positive perturbation ϵ. In such a case, we also modify the definition of the set I_X^0 so that the inequality is no longer strict (i.e., $I_X^0 \equiv \{i \in I_X : \nabla_1^{AD} a_i + \nabla_2^{AD} b_i \leq \beta^{AD}\}$), since any point along AD will, in this case, necessarily have $y_i = 0$. Even so, following the procedure discussed in Section 6.2.3 next, the overall algorithm will automatically prefer some alternative solution where the partitioning line intersects edge AD in this case.

6.2.3 General Bisection Problem (BP)

We propose and test two options for solving the general bisection problem (BP). In **Option 1**, we solve Problem RBP for all pairs of edges of X and select the best resulting solution. In this case, given the symmetry in the problem, there will likely exist several alternative optimal solutions that are identified by this process. For example, if X is an elongated rectangle as depicted in Figure 6.5 having a uniform distribution of weights, either of the solutions $\nabla \cdot x = \beta$ displayed in Figures 6.5(a) and 6.5(b) would solve Problem BP, although from a practical or aesthetic point of view, we would prefer the partition given in Figure 6.5(a). Hence, in general, given a set of identified alternative optimal solutions to Problem BP, we can discriminate among these designated planes by preferring the one that yields the shortest intersection with X. We refer to the resulting preemptive priority bi-objective problem solved as **Problem BP1**.

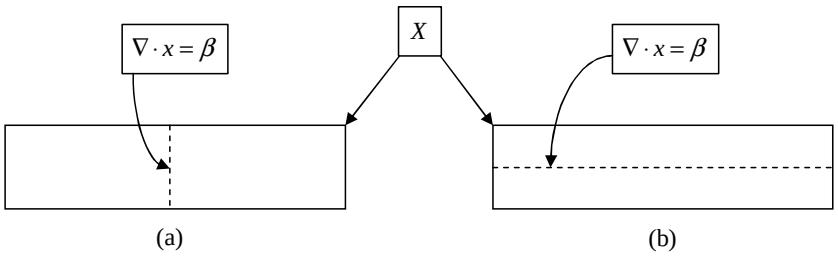


Figure 6.5: Discriminating among alternative optimal solutions to Problem BP.

In the second option (**Option 2**), we directly compute the required partitioning plane $\nabla \cdot x = \beta$ as follows. First of all, note that a known issue with linear and integer programming

problems in which the goal is to generate a separating plane is that the trivial solution, in which $\nabla = 0$ and $\beta = 0$, will allow an optimal solution without actually generating a separating plane (Koehler 1990). This is not a problem when BP1 is solved after updating the bounds on λ_e via a procedure such as ULLB, because ∇ is then guaranteed to have a non-zero magnitude. In the present case, Koehler notes that constraining the orientation vector to have a constant magnitude will avoid this problem without cutting off any feasible solutions; however, since the Euclidean norm is non-linear, we implement a similar idea using the L_1-norm $\|\nabla\|_1 \equiv |\nabla_1| + |\nabla_2| = 1$. This constraint makes necessary an alternative process for determining the bounds for ∇ and β.

First we restrict ∇_1 to lie in $[-1, 1]$, and we define ∇_2 to lie in $[0, 1]$. These values allow for any orientation and will always allow $\|\nabla\|_1 = 1$. Define non-negative variables ∇_{1+} and ∇_{1-} such that $\nabla_1 = \nabla_{1+} - \nabla_{1-}$, and define a binary variable y_{∇_1} to act as a switch between positive and negative values of ∇_1. We model the associated restrictions on ∇ as follows:

$$\nabla_1 = \nabla_{1+} - \nabla_{1-} \tag{6.12a}$$

$$\nabla_{1+} \leq y_{\nabla_1} \tag{6.12b}$$

$$\nabla_{1-} \leq 1 - y_{\nabla_1} \tag{6.12c}$$

$$\nabla_{1+} + \nabla_{1-} + \nabla_2 = 1 \tag{6.12d}$$

$$(\nabla_{1-}, \nabla_{1+}, \nabla_2) \geq 0, y_{\nabla_1} \in \{0, 1\}. \tag{6.12e}$$

For determining the bounds on β in the relationship $\nabla \cdot x = \beta$, consider the following:

$$\beta_l^* = \min\{\nabla_1 x_1 + \nabla_2 x_2 : |\nabla_1| + \nabla_2 = 1, \nabla_2 \geq 0, x \in X\} \tag{6.13a}$$

$$\beta_u^* = \max\{\nabla_1 x_1 + \nabla_2 x_2 : |\nabla_1| + \nabla_2 = 1, \nabla_2 \geq 0, x \in X\}. \tag{6.13b}$$

Let $x_{1\ell} = \min\{x_1 : x \in X\}$, $x_{1u} = \max\{x_1 : x \in X\}$, $x_{2\ell} = \min\{x_2 : x \in X\}$, and $x_{2u} = \max\{x_2 : x \in X\}$. Then from (6.13), noting the bilinear objective function, we readily

obtain

$$\beta_l^* = \min\{x_{1\ell}, -x_{1u}, x_{2\ell}\} \text{ and } \beta_u^* = \max\{-x_{1\ell}, x_{1u}, x_{2u}\}. \tag{6.14}$$

Next, note that the goal-programming type weight-balance constraint in (6.10b) is defined based on a pair of edges AB and CD. Since Option 2 considers all edges simultaneously, we must instead impose

$$\sum_{i \in I_X} w_i y_i - \sum_{i \in I_X} w_i (1 - y_i) = s^+ - s^-. \tag{6.15}$$

Now, we can generate the required partitioning plane by solving the following MIP:

BP2: Minimize $\{(6.10a) : (6.15), (6.10c), (6.12),$ and (6.9) re-defined for all $i \in I_X$ (in lieu of $i \in I_X^{AB,CD}$) and using $\nabla_{1\ell}^* = -1, \nabla_{1u}^* = 1, \nabla_{2\ell}^* = 0, \nabla_{2u}^* = 1,$ and β_ℓ^* and β_u^* given by $(6.14)\}$. (6.16)

Note that because Problem BP2 ignores the inherent symmetry in the problem, we might expect the MIP (6.16) for Option 2 to be more computationally difficult to solve than each Problem RBP given by (6.10), although we would be solving several such latter problems for Option 1. On the other hand, Option 1 provides an added control to avoid constructing long skinny sectors as illustrated in Figure 6.5. We shall investigate this computational effort versus quality aspect with respect to Options 1 and 2 in Section 6.5, in addition to the alternative algorithmic variants discussed in the next section.

Additionally, solving Problem BP2 might result in an undesirable solution similar to the one depicted in Figure 6.4(b). Such solutions can be prevented by incorporating constraints imposing $|\nabla \cdot x^i - \beta| + |\nabla \cdot x^j - \beta| \geq \epsilon$ for all adjacent pairs of vertices x^i and x^j of X, for example. In lieu of adding these additional constraints to the problem, upon encountering an undesirable solution we solve an alternative problem like those described in the sequel for Algorithms 2 and 3.

6.3 Algorithmic Variants

In this section, we discuss three alternative algorithms for performing the partitioning of the given airspace into m polygonal sectors. The first of these described below formalizes the procedures proposed in Section 6.2.

Algorithm 1.

Initialization: Let $S = \{X_1, \ldots, X_r\}$ represent the initial specified set of polytopes, where $1 \leq r < m$, and where $\mathbf{int}(X_k) \cap \mathbf{int}(X_\ell) = \varnothing, \forall k \neq \ell$ in $\{1, \ldots, r\}$. Correspondingly, let $I_{X_k}, k = 1, \ldots, r$, be a *partitioning* of the index set of grid points $i = 1, \ldots, n$, where $(a_i, b_i) \in X_k, \forall i \in I_{X_k}, k = 1, \ldots, r$. Define $w_{X_k} \equiv \sum_{i \in I_{X_k}} w_i, \forall k = 1, \ldots, r$.

Step 1: Select $k^* \in \arg\max\{w_{X_k}, k = 1, \ldots, r\}$ and delete X_{k^*} from S.

Step 2: Use **Option 1** or **Option 2** of Section 6.2 (respectively yielding **Algorithm 1.1** and **Algorithm 1.2**) to generate the separating plane $\nabla \cdot x = \beta$.

Step 3: Determine the intersection of $\nabla \cdot x = \beta$ with X_{k^*} to partition it into two sub-polygons X_{k^*1} and X_{k^*2}, with respective grid point sets $I^+_{X_{k^*}}$ and $I^-_{X_{k^*}}$ as defined in (6.11), and having respective cumulative weights given by

$$w_{X_{k^*1}} = \frac{w_{X_{k^*}} + s^+ - s^-}{2}, \text{ and } w_{X_{k^*2}} = w_{X_{k^*}} - w_{X_{k^*1}}.$$

Rename X_{k^*1} and X_{k^*2} as X_{k^*} and X_{r+1}, respectively, and increment $r \leftarrow r + 1$. If $r = m$, then stop; else, go to Step 2. □

The second algorithmic variant is similar to Algorithm 1.1 (i.e., using Option 1), except that it solves (a relaxed version of) Problem RBP for only a single pair of edges identified as follows, where X is the given polytope to be partitioned having a total weight w_X, E is the set of edges of X, and where $0 \leq \mu < 1$ is a selected parameter.

Procedure Edge-Pair EP(X, μ):

– For each edge $e \in E$ described by $\alpha_e + \lambda d_e, 0 \leq \lambda \leq 1$, determine its perpendicular bisector given by the plane $\nabla_e \cdot x = \beta_e$, when $\nabla_e \equiv d_e$ and $\beta_e = d_e \cdot (\alpha_e + \frac{d_e}{2})$. Determine

$$I_e^+ = \{i \in I_X : \nabla_e \cdot (a_i, b_i) > \beta_e\} \text{ and } I_e^- = \{i \in I_X : \nabla_e \cdot (a_i, b_i) < \beta_e\}$$

and compute

$$w_e^+ = \sum_{i \in I_e^+} w_i, \; w_e^- = \sum_{i \in I_e^-} w_i, \text{ and } w_e^0 = w_X - (w_e^+ + w_e^-).$$

– Define the *imbalance* for edge e as

$$\theta_e \equiv \min_{0 \leq \gamma \leq 1} \left\{ \left| \left[w_e^+ + (1-\gamma) w_e^0 \right] - \left[w_e^- + \gamma w_e^0 \right] \right| \right\}.$$

Note that the optimal value of γ for computing θ_e is given by $\gamma = 0$ if $w_e^0 = 0$ or $(w_e^+ + w_e^0 - w_e^-) \leq 0$; else, $\gamma = \min\{1, (w_e^+ + w_e^0 - w_e^-)/2w_e^0\}$.

– Compute $\theta^* \equiv \min\{\theta_e : e \in E\}$. Let l_e be the length of the line segment obtained by intersecting $\nabla_e \cdot x = \beta_e$ with X, and denote the edge other than e that determines this intersection as p_e.

– Determine $e^* \in \arg\min_{e \in E}\{l_e : \theta_e \leq \theta^* + \mu w_X\}$. Output the pair of edges e^* and p_{e^*}. □

Algorithm 2.

This algorithm is identical to Algorithm 1 except that Step 2 is replaced by the following, where $0 \leq \mu < 1$ is a specified parameter (we recommend $\mu = 0.1$).

Step 2: Invoke Procedure EP(X_{k^*}, μ) to determine the pair of edges e^* and p_{e^*}. Solve RBP corresponding to this pair of edges (incorporating the bounds on the λ_e-parameters as

prescribed in Section 6.2.3, as necessary) to generate the partitioning plane $\nabla \cdot x = \beta$. □

The third proposed procedure imitates Algorithm 2, but solves a simpler restricted version of Problem RBP as follows:

Algorithm 3.

This is identical to Algorithm 2 with Step 2 replaced by the following:

Step 2. Invoke Procedure EP(X_{k^*}, μ) to determine the pair of edges e^* and p_{e^*}. Let $\alpha_e + \lambda_e d_e, 0 \leq \lambda_e \leq 1, e = 1, 2$ characterize these two edges (as in Figure 6.2). Compute

$$\nabla = \frac{d_1}{2\|d_1\|} - \frac{d_2}{2\|d_2\|}. \tag{6.17}$$

(Note that if d_1 and $-d_2$ are collinear, then ∇ coincides with this (scaled) direction; otherwise, ∇ is oriented along the direction that bisects the angle between the pair of edges e^* and p_{e^*}.) Consider the separating plane $\nabla \cdot x = \beta$ with ∇ *fixed* as computed by (6.17), and where β is restricted to lie in the interval

$$\beta_\ell \leq \beta \leq \beta_u, \text{ where } \beta_\ell = \min\{\nabla \cdot x : x \in X\} \text{ and } \beta_u = \max\{\nabla \cdot x : x \in X\}. \tag{6.18}$$

Accordingly, solve the following problem, prompted by (6.9) and (6.10), to compute the separating plane $\nabla \cdot x = \beta$, where β_ℓ and β_u are as specified in (6.18), and $\nabla = (\nabla_1, \nabla_2)$ is determined by (6.17) (so then, $\nabla^i_j \equiv \nabla_j y_i, \forall i \in I_X, j = 1, 2$ in (6.9)):

$$\begin{aligned}
\text{Minimize } \{&\text{(6.10a) : (6.15) and (6.10c), along with:} \\
&(\nabla_1 a_i + \nabla_2 b_i)(1 - y_i) \leq \beta - \beta^i, \quad \forall i \in I_X \\
&(\nabla_1 a_i + \nabla_2 b_i) y_i \geq \beta^i, \quad \forall i \in I_X \\
&\beta_\ell (1 - y_i) \leq \beta - \beta^i \leq \beta_u (1 - y_i) \text{ and } \beta_\ell y_i \leq \beta^i \leq \beta_u y_i, \quad \forall i \in I_X \\
&y_i \text{ binary}, \forall i \in I_X\}.
\end{aligned} \tag{6.19}$$

6.4 Workload Measures

Air traffic controllers characterize their own workload as divided into three basic tasks as follows (Delahaye et al. 1994):

1. Each aircraft trajectory is checked periodically so that the controller can maintain a sense of the overall traffic flow within the sector and hence be able to anticipate problems. This is called the *monitoring workload*.

2. When two aircraft will likely collide or move dangerously close to one another, the controller must resolve this conflict. We refer to this as the *conflict workload*.

3. An aircraft leaving one air traffic controller's *area of responsibility* (AOR) necessarily moves into the AOR for another controller. The information exchange between the aircraft and the controllers induces a *coordination workload*.

Sherali, Staats and Trani (2003) describe modelling constructs for ascertaining the number of aircraft simultaneously occupying a sector, as well as the number of simultaneously occurring conflicts. Since monitoring workload in a given sector is, by definition, correlated with the number of aircraft in the sector, we can use these methods in the context of the APCDM to measure the conflict and monitoring workloads for a given set of flights over a hypothetical airspace comprising, for example, the grid of hexagonal cells described by Yousefi and Donohue (2004). The resulting workload measures distributed over such cells can be used as the weights for the grid points in the RBP. The coordination workload, however, is a function of the sector definition process, and as such, must be approached in a different way. First we discuss the monitoring and conflict workload measures, and then we present a framework for incorporating coordination workload measures into the airspace partitioning process.

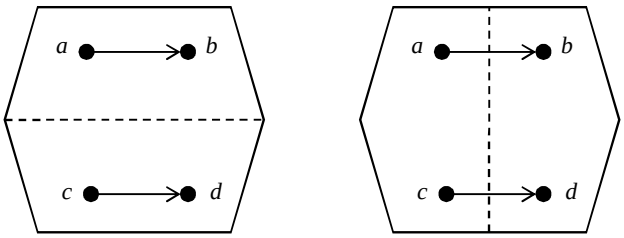
6.4.1 Monitoring and Conflict Workloads

Suppose that we superimpose onto the airspace a grid of hexagonal cells to represent hypothetical sectors. The *Airspace Occupancy Model* (AOM) of Sherali et al. (2000) provides sector occupancy times for a given flight trajectory, and this information can be used in the *Aircraft Encounter Model* (AEM) (Sherali et al. 2000) or in the *Probabilistic Aircraft Encounter Model* (PAEM) (Sherali, Staats and Trani 2003) to determine potential conflicts between flight plans. We can then generate an instance of the APCDM, which, when solved, will return the peak and average workloads for these hypothetical sectors, and will also identify any conflicts that must be resolved among the prescribed set of flight plans. Alternatively, since the hypothetical sector geometries might affect the optimal solution, we can relax the x_{fp}-variables to vary continuously over $[0, 1]$ and use the resulting solution to estimate the *expected* sector occupancies and resolvable conflict durations. Using the continuous relaxation of binary variables to estimate probabilities is not unprecedented, and has been used previously by Raghavan and Thompson (1987) and Rubin (1997), for example. We utilize this technique to compute the grid point weights in Section 6.5.

6.4.2 Coordination Workload

Accounting for the coordination workload in the bisection process is less straightforward, because the coordination workload is a function of the resulting partition. Consider the example in Figure 6.6, which shows four airports (a, b, c, and d), where there exists one flight from a to b and another from c to d. The balanced bisection in Figure 6.6(a) indicates a coordination workload of zero since both flights stay in one sector for the duration, whereas the bisection in Figure 6.6(b) indicates a coordination workload proportional to two, since each flight crosses exactly one sector boundary.

Ideally, we would like to minimize the coordination workload while simultaneously balancing the gridpoint weights as in the RBP. This is akin to the *minimum bisection problem*,



(a) Bisection with zero coordination workload.

(b) Bisection with nonzero coordination workload.

Figure 6.6: Coordination workload example.

which is known to be NP-hard (Garey et al. 1974). Rather than solve the minimum bisection problem, we incorporate the coordination workload into Problem RBP as follows. Define v_{ij} as a binary variable that equals one if grid points i and j are not members of the same subset in the partition and zero otherwise (i.e., $v_{ij} = 1$ if gridpoint i is in I_X^+ and gridpoint j is in I_X^-, or vice-versa), and let a_{ij} represent the number of aircraft moving from hexagonal cell i to cell j. Also define the set $A_i = \{j \in I_X : \text{ cell } j \text{ is adjacent to cell } i \text{ with } a_{ij} \geq 1\}, \forall i \in I_X$. Finally, let w_i^c be the total number of aircraft leaving hexagonal cell i that need coordination (i.e., cross between I_X^+ and I_X^-), and let w^c represent the overall coordination workload for the polytope X. We model coordination workload via the following set of constraints:

$$v_{ij} \geq y_i - y_j, \quad \forall i \in I_X, j \in A_i \tag{6.20a}$$

$$v_{ij} \geq y_j - y_i, \quad \forall i \in I_X, j \in A_i \tag{6.20b}$$

$$v_{ij} \leq y_i + y_j, \quad \forall i \in I_X, j \in A_i \tag{6.20c}$$

$$v_{ij} \leq 2 - y_i - y_j, \quad \forall i \in I_X, j \in A_i \tag{6.20d}$$

$$w_i^c = \sum_{j \in A_i} a_{ij} v_{ij}, \quad \forall i \in I_X \tag{6.20e}$$

$$w^c = \sum_{i \in I_X} w_i^c \tag{6.20f}$$

$$0 \leq v_{ij} \leq 1, \quad \forall i \in I_X, j \in A_i. \tag{6.20g}$$

Note that the v_{ij}-variables are declared to be continuous but would automatically take on binary values, given any binary vector y. We can incorporate coordination workload into the RBP or BP2 in one of two ways: (i) we can use the coordination workload non-preemptively to select among alternative optimal solutions as in **Algorithm 1.1**, or (ii) we can incorporate an appropriate penalty for coordination workload into the objective function. We investigate the latter, more proactive modeling approach in Section 6.5 (we used a penalty parameter value of $0.01w^c$ in our experiments). Also, when coordination workload is penalized in the objective function, Constraints (6.20c) and (6.20d) are unnecessary, because v_{ij} would then automatically be zero whenever (y_i, y_j) equals $(0, 0)$ or $(1, 1)$ (given $a_{ij} > 0$).

6.5 Illustrative Examples and Computational Experience

In this section, we provide results for two different sets of experiments concerning the algorithms discussed above. For the experiments in Section 6.5.1, we examine a preliminary example in which we subdivide a square. The initial polygon is populated with 371 grid points having randomly assigned integer weights, and each of the aforementioned algorithms are used to divide the region into 32 sectors. These experiments illustrate that the proposed bisection problems might be ill-conditioned, and we provide suggestions for decreasing the processing times of the competitive algorithms. In Section 6.5.2, applying the lessons learned in Section 6.5.1, we employ the most competitive algorithm to generate sectorization plans for the U.S. airspace. All algorithms were implemented using a custom-built application, written in C, which employs the CPLEX 11.1 callable libraries. Initially, all of the CPLEX options were set at their default values. The experiments were performed on a Dell Precision workstation with dual quad-core Intel Xeon E5345 2.33 GHz processors and 3.25 GB of RAM. However, only a single processor thread was used.

6.5.1 Initial Computational Experience

In this section, we examine the computational performance and solution quality for a preliminary example, where the computational performance is measured using the CPU time consumed by each algorithm, whereas solution quality is measured by finding the *mean absolute deviation* (MAD) of the sector weight values. We begin with a single polygon, a square having vertices at $(0, 0)$, $(1, 0)$, $(1, 1)$, and $(0, 1)$. Onto this region we superimpose a diamond-shaped lattice of 371 grid points. The horizontal distance between any two neighboring grid points in the same row is 0.05, and the vertical distance between rows is 0.025. An integral weight between 1 and 10 was randomly assigned to each grid point, and random integer flows on $[1, 10]$ were generated between neighboring grid points. We divided the square into 32 sectors using each of the Algorithms 1.1, 1.2, 2, and 3.

Solution Time

We first generated sectors without accounting for coordination workload. The problems, especially Problem BP2 in Algorithm 2, seemed to exhibit numerical difficulties (e.g., dwelling at the same solution value without progressing toward optimality), and so we enabled the CPLEX option to emphasize numerical precision (ENP), which eliminated the numerical problems and enhanced performance. The resulting solution times are included in Table 6.1.

It is worth noting the impact of the y_i-variable fixing procedure. Generating sectors using Algorithm 1.1 without fixing the y_i-variables whose values could be determined *a priori* took more than 3500 seconds (and more than 7600 seconds without the ENP option enabled). Solving the same set of problems after fixing the pre-determined y_i-values took just over 236 seconds.

Generating sectors while minimizing coordination workload proved to be more difficult. Even with the y_i-fixing procedure and an emphasis on numerical precision in CPLEX, Algorithm 1.1 required 1878 seconds to generate 32 sectors. Accordingly, we imposed a time

Table 6.1: Solution Time (seconds) and Quality.

	Coordination Workload					
	Neglected			Incorporated		
Algorithm	Time	MAD	CW	Time	MAD	CW
1.1	236.63	0.50	2841	*1089.60	0.50	1844
1.2	523.62	0.50	3241	1968.34	0.50	1612
2	78.22	0.69	2791	392.05	0.50	1921
3	2.92	1.31	2175	14.66	1.31	1942

* With time-limits imposed on subproblems.

limit on the subproblems for each pair of edges. This time limit was dependent on the quality of the best incumbent solution for that particular polytope. A lower bound on the optimal objective function value is zero (i.e., a perfectly balanced workload, with zero coordination workload). We computed the *solution quality* by dividing the objective function value by the total weight, which represents the difference between the incumbent solution value and the ideal value as a percentage of the total weight. For the first subproblem, there was no time limit imposed. If the incumbent solution quality for a pair of edges was within 0.1%, we set the time limit for subsequent problems to 60 seconds. If the incumbent solution quality was greater than 1% but less than 5%, the time limit was set to 600 seconds. We imposed a 1200 second limit when solution quality was between 5% and 10%, and there was no time limit when the incumbent solution quality was more than 10%. This rule reduced the Algorithm 1.1 solution time for generating sectors in combination with minimizing the coordination workload from 1878 seconds to 1090 seconds, while reaching an identical solution.

As Figure 6.7 indicates, most of the time spent by each of the algorithmic variants occurs at the beginning of the process. In fact, generating Sectors 5–32 took considerably less time in general than generating Sectors 1–4.

Solution Quality

Each of the algorithmic variants yielded sectors with relatively balanced weight values. However, the results of the more relaxed Algorithms 2 and 3 do indicate a tradeoff between

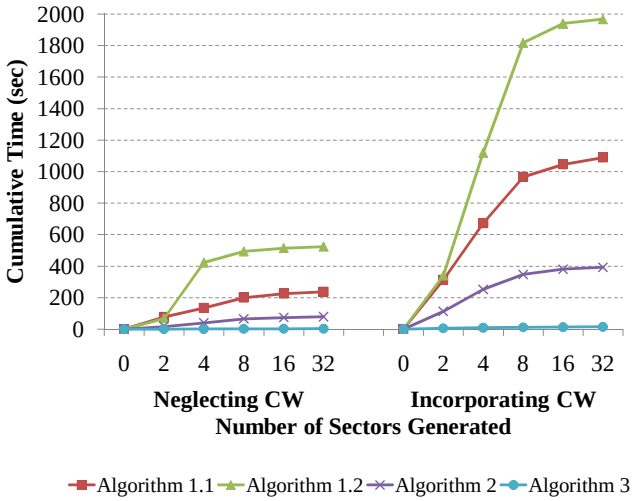


Figure 6.7: Algorithmic progression.

computational time and weight balance (and when applicable, computational time and coordination workload). Table 6.1 provides the MAD values for the various solution procedures and problem combinations. Note that a MAD of 0.5 is the best possible value for this particular problem. Also included is the total coordination workload (CW) generated by each solution, including those for which coordination workload was not minimized.

Solution Comparisons

In this section, we provide solution plots to compare the different algorithmic variants. Penalizing coordination workload impacts the solution for every algorithmic variant, and Figure 6.8 provides a comparison for each. Note that Algorithms 1.1 and 1.2 produce very different solutions when coordination workload is neglected (see Figures 6.8(a) and 6.8(c), respectively); however, when minimizing coordination workload, the solutions are relatively more similar. In fact, the bottom halves of the solutions in Figures 6.8(b) and 6.8(d) are almost identical, while the top halves are still quite distinct. The similarities are somewhat expected because the alternative balanced-weight solutions for the pairs of edges considered in Algorithm 1.1 yielded different coordination workload values, which therefore discriminated among these solutions. Consequently, the non-preemptive selection rule of choosing

the shortest new edge did not come into play. A possible corollary is that long, skinny sectors might sometimes be desirable when accounting for coordination workload. This idea is in line with the *corridors-in-the-sky* concept mentioned by Kopardekar et al. (2007).

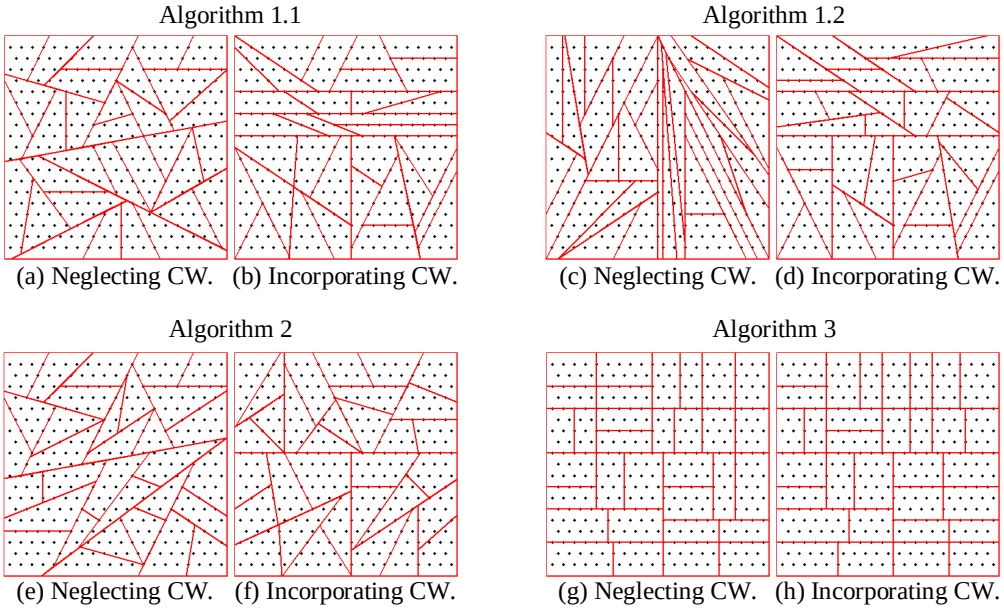


Figure 6.8: Sector results.

Algorithms 2 and 3 solve more relaxed versions of Problem RBP, and the results are provided in Figures 6.8(e)–6.8(h). Note that the sectors resulting from Algorithm 3 depend heavily upon the shape of the original polygon. Also noteworthy is that including coordination workload in these options had a relatively smaller effect on the resulting sector layouts.

Other Examples

To investigate the significance of the given initial airspace region (i.e., the initial union of polytopes defining the airspace), we used the same set of grid points, weights, and flow values, but instead of the previous square, the given airspace section was modified by constructing a hexagon within this square. Thus, the initial region comprised five polygons: four triangles and a hexagon. We henceforth refer to the first example as the *square* scenario and to this

second example as the *hex-square* scenario. We again ran each algorithm until there were 32 distinct regions; the solution times and quality measures are presented in Table 6.2. Notice that even though there were fewer bisections necessary (27 instead of 31), the problems incorporating coordination workload took longer to solve. Additionally, the lack of balance in the total weights for the initial union of polytopes yielded a final solution that was much less balanced than the solutions summarized in Table 6.1. One noteworthy observation is that due to the imbalance in the initial solution, and the sub-optimal nature of Algorithm 3 with regard to weight balance, the overall solution from Algorithm 3 was actually the most balanced (i.e., having lower MAD-values). The resulting sectors are displayed in Figure 6.9.

Table 6.2: Solution Time (seconds) and Quality.

	Coordination Workload					
	Neglected			Incorporated		
Algorithm	Time	MAD	CW	Time	MAD	CW
1.1	148.03	17.25	2814	*1,525.69	17.25	1806
1.2	350.88	17.25	3591	3,176.88	17.25	1806
2	34.31	17.19	2845	563.50	17.25	2194
3	2.42	16.38	2235	10.59	16.25	2207

* With time-limits imposed on subproblems.

As in the *square* example, Algorithm 1.2 yielded several long, skinny sectors when we neglected coordination workload, and the results from Algorithms 1.1 and 1.2 were relatively more similar when incorporating coordination workload. An interesting difference when starting with this *hex-square* sector configuration is that the sector layout resulting from Algorithm 3 was relatively more complex (see Figures 6.9(g)–(h)), when compared with the configurations produced in Figures 6.8(g)–(h), owing to the shape of the given initial airspace region.

Overall, in the *square* example, Algorithm 1.1 was the fastest to yield an ideally balanced solution for both CW scenarios, generating sectors in approximately half the time of Algorithm 1.2. Although Algorithm 2 derived reasonably good quality solutions (increased MAD values 0% and 35% with and without CW consideration), while significantly curtailing

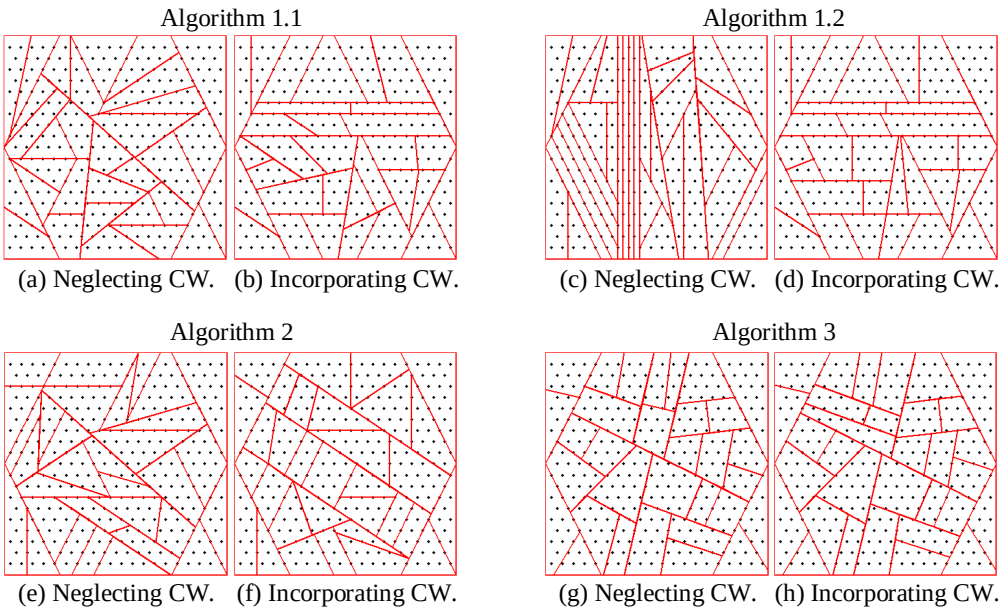


Figure 6.9: Sector results.

computational effort (64% and 67%, respectively), its heuristic nature resulted in increased coordination workload. In the *hex-square* problem, Algorithms 2 and 3 out-performed Algorithm 1.1 with respect to both MAD and computational effort, but Algorithm 1.1 was better at simultaneously controlling the coordination workload. Due to the relative robustness of Algorithm 1.1 and its acceptable computational requirements in practice, we prescribe it for consideration and utilize it in the next section for a realistic scenario.

6.5.2 U.S. Airspace Example

The demand for U.S. airspace is constantly changing. Even though the current sectorization scheme was designed to take advantage of historical air traffic flow patterns, certain circumstances (for example, severe weather conditions) can adversely impact sector capacities, which might in turn reduce the traffic flow. Hence, the Federal Aviation Administration (FAA) is interested in exploring dynamic sector configurations, or a redefinition of existing sector boundaries based on current air traffic flows, in order to better utilize air traffic control resources (FAA 2002). For illustrative purposes, we considered some 1936 flights routed

through a section of the national airspace in the vicinity of Cleveland, Ohio, as depicted in Figure 6.10. Due to an assumed convective weather system, the capacity of most of the sectors in the area was reduced. The current sector configuration is displayed in Figure 6.11(a). The airspace in question comprises 25 sectors, but Figure 6.11(a) shows a 2-D projection of only 11; the remaining sectors are similarly defined, but differ in the floor and ceiling altitudes.

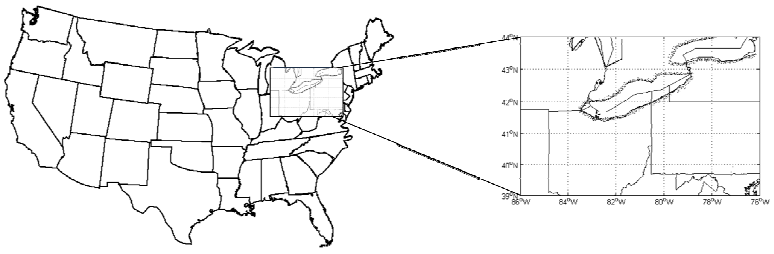
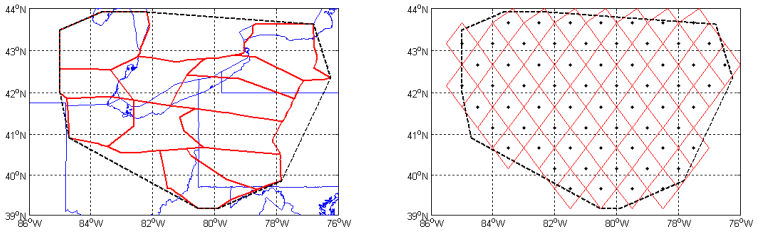


Figure 6.10: Scenario location.

We solved APCDM under the current sector configuration, and extracted the sector workload information resulting from the optimal solution. In solving APCDM, the number of resolvable conflicts and ATC workload were not limited so that me might obtain worst-case workload performance measures. Then, we generated 65 generically-shaped dummy sectors, depicted in Figure 6.11(b), to cover the area contained within the convex hull of the original sector layout. These were constructed at each of three altitude levels so as to encompass the entire volume enclosed by the original sector configuration. Based on the floor and ceiling altitudes of the original sectors, the lowest layer of dummy sectors was generated between 24,000 and 31,000 feet, the middle layer between 31,000 and 36,000 feet, and the highest layer spanned the volume from 36,000 feet upward.

We again solved APCDM without limiting the number of resolvable conflicts, but this time using the 195 dummy sectors. Based on the results obtained, we computed the peak and average aircraft counts in each dummy sector, as well as the number and duration of any resolvable conflicts. Several combinations of these four workload measures were then analyzed, and one such combination was chosen as follows to assign weight values to grid



(a) Original sector configuration. (b) Overlaid dummy sectors.

Figure 6.11: Sector configurations.

points at the center of each of the dummy sectors. The conflict-related measures were highly correlated with one another, as were the aircraft count-related measures. Hence, we considered combinations involving one conflict-related measure and one count-related measure. A weighted sum of normalized conflict duration values and normalized peak workload values for each dummy sector was chosen, because the distribution of the resulting weights was similarly shaped at all three altitude levels and was relatively smooth.

The convex hull of the original sector layout was then used as the initial polygon to generate sectors at all three altitude levels. We chose to generate eight sectors at each level to approximate the total number of sectors in the original layout (25), while keeping the weight-values balanced within each layer. The resulting sectors after implementing Algorithm 1.1 under both conditions, neglecting and incorporating coordination workload, are displayed in Figures 6.12(a)–(c) and 6.12(d)–(f), respectively.

Finally, we evaluated each new sector layout by running APCDM and comparing the resulting sector workload measures with those obtained for the original sector configuration. The distributions of the average and peak aircraft counts are presented in Figure 6.13 (with frequencies along the ordinate axes), and summary statistics are provided in Table 6.3. Notice that even with fewer sectors, 24 as opposed to 25, both new sector layouts provided lower, more balanced average aircraft occupancy statistics. Moreover, peak workload was more equitably distributed, although the mean peak workload was slightly lower for the original sector configuration than for the sectors generated while neglecting CW. Penalizing coordination workload gave rise to even lower peak and average workload values, while

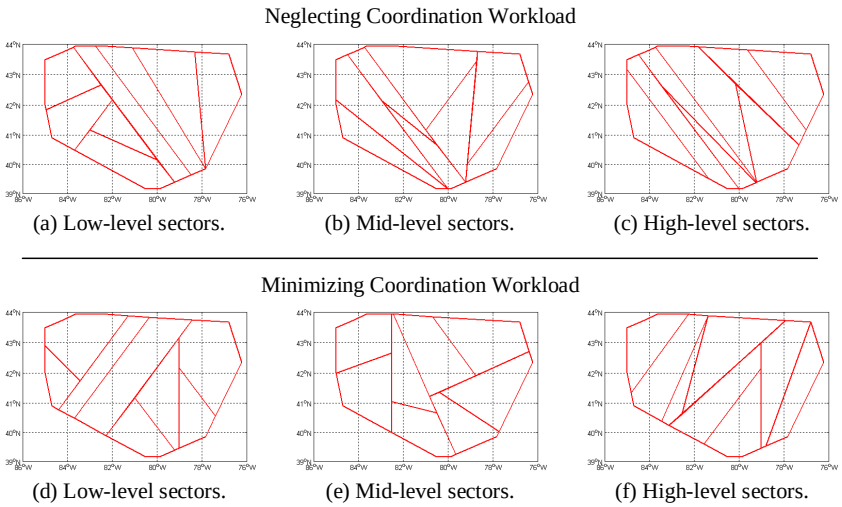


Figure 6.12: Generated sectors.

reducing the number of hand-offs by approximately 26% when compared with the sectors generated while neglecting coordination workload.

Table 6.3: Solution quality measures.

	Aircraft Count			
	Average		Peak	
Scenario	Mean	MAD	Mean	MAD
Original Sectors	1.38	0.44	7.36	1.50
Neglecting CW	1.16	0.29	7.88	1.31
Incorporating CW	1.06	0.35	7.13	1.07

Nonetheless, there are certain drawbacks to the new sector layouts. First off, the original sector design is relatively homogeneous between altitude levels, whereas, for example, our mid-level sector layout looks very different from its high-level counterpart. This might affect an air traffic controller's ability to transition easily between managing sectors at different flight levels during the same shift. Additionally, in the spirit of Yousefi and Donohue (2004) our methodology generates sectors separately within each layer of airspace, whereas the current airspace contains sector definitions that span multiple layers in cases where the traffic density does not merit defining multiple sectors. Whether these potential disadvantages are actually barriers to designing a functional airspace with respect to

human-in-the-loop considerations is left as an open question.

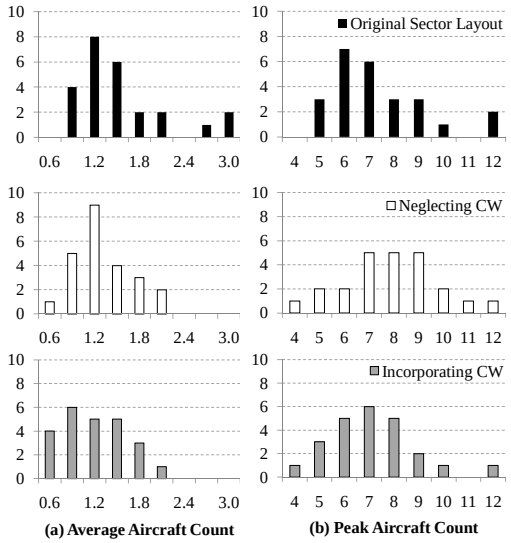


Figure 6.13: Average and Peak Aircraft Count Distributions from APCDM.

6.6 Summary and Conclusions

We have developed and implemented several algorithms designed to bisect a polygonal region such that the total weight of grid points assigned to either resulting half-space is balanced, with the motivation of generating sector configurations within an airspace. Both hypothetical and realistic examples were solved to illustrate the algorithms and provide insights into their relative performance with respect to computational effort and the quality of solutions provided. Among the proposed methods, Algorithms 1.1 and 2, both of which were designed based on suitable pairwise edge considerations, yielded relatively competitive results. Algorithm 1.1, in which we solve problem RBP for each pair of edges in a polytope, required more computational effort, but produced superior solutions depending on the importance placed on reducing coordination workload. On the other hand, Algorithm 2, in which a pair of edges is determined beforehand, provided relatively balanced solutions quickly but was not as successful at reducing the coordination workload.

Possible areas for future research include determining alternative methods for computing the ϵ-perturbations used to avoid degeneracy, and developing heuristic approaches for finding good initial solutions to pass to the optimization software. The development of rules for triggering sector redefinitions is also a potential research area.

Another important topic for future research is to derive techniques for assigning grid point locations. The idea would be similar to that of quadrature in numerical integration. A relatively low number of well-placed grid points might be able to allow for an accurate workload representation while yielding faster computational times for the AOM and (P)AEM, as well as for a more refined partitioning process.

Chapter 7

Conclusions and Recommendations for Future Research

By 2025, U.S. air traffic is predicted to increase two- to three-fold, and modernizing the air traffic control system is a major priority for the United States. The *NextGen Air Transportation System* is being developed to allow the U.S. National Airspace System (NAS) to accommodate the projected increased air traffic and to more safely and effectively manage it.

In this dissertation, with the motivation of generating alternative flight plans for consideration in an airspace flow program, we first developed a pseudo-polynomial time dynamic programming algorithm for finding an optimal solution to a reverse time-restricted shortest path problem with a specially structured, but non-FIFO, travel time function. We proved that the problem is NP-hard in general, but under a special regularity condition, we showed that the problem can be solved in polynomial time of computational complexity $O(|N||A|)$. Experimental results using both randomly generated networks and an actual network induced by the U.S. Navigational Aids amply exhibited the efficacy of the proposed algorithm. For future research, as a further expedient to solve large problem instances, we recommend incorporating additional heuristic techniques such as the triangle-inequality

based method of Sedgewick and Vitter (1986).

The notion of slot ownership, formalized via the FAA's Collaborative Decision Making initiative under the enhancements to the GDP, has spawned new research efforts pertaining to slot-trading opportunities that could provide additional benefits in terms of flight efficiencies and desirable airline schedules. Given that airlines submit at most-at least trade offers, we have significantly expanded the functionality of the original APCDM model of Sherali, Staats and Trani (2003, 2006) in this work to automatically accommodate the consideration of such trades in terms of related slot exchanges, along with appropriate flight connection constraints, while simultaneously considering the impact of the resultant overall mix of flight plans on sector workloads, safety with respect to conflict risk, and equity among the involved airlines. We also proposed two alternative equity formulations that produce solutions comparable to those obtained using the original APCDM model's equity concept, but far more efficiently. One of the new methods, EM2, which is based on average delay per passenger (or weighted average delay per flight), turns out to be a particularly robust way to model equity considerations in conjunction with sector workloads, conflict resolution, and slot exchanges. Moreover, employing EM2 allowed us to solve the relatively more difficult 1,000 flight problem instances for APCDM within 30 seconds on average using a 1% optimality tolerance, and yielded comparable solutions within about 20 seconds on average using the *Sequential Fixing Heuristic* (SFH) with an optimality tolerance of 0.01%. The actual solutions obtained for these largest problem instances were well within 1% of the best known solution. In addition, we designed a simplified model formulation, APCDM-Light, which is more amenable for use as an *airspace flow program*. This model was readily optimized to a 0.01% tolerance within about 5 seconds on average for the largest set comprising 1,000 flights. The augmented APCDM model therefore offers a viable tool that can be used for tactical air traffic management purposes as an airspace flow program (particularly, APCDM-Light), as well as for strategic applications to study the impact of different types of trade restrictions, collaboration policies, equity concepts, and airspace sectorizations.

The development of the APCDM model is part of ongoing research related to NASA's

Next Generation Air Transportation System (NextGen), which envisages the sequential use of a variety of models pertaining to three tiers. The *Tier 1* models, such as the one described by Bertsimas et al. (2009), are more strategic in scope and attempt to identify potential problematic areas, e.g., areas of congestion resulting from a severe convective weather system over a given time-frame, and provide aggregate measures of sector workloads and delays. The affected flow constrained areas (FCAs) highlighted by the results from these *Tier 1* models would then be analyzed by more detailed *Tier 2* models, such as APCDM, which consider more specific alternative flight plan trajectories through the different sectors along with related sector workload, aircraft conflict, and airline equity issues. Finally, *Tier 3* models are being developed to dynamically examine smaller-scaled, localized fast-response readjustments in air traffic flows within the time-frame of about an hour prior to departure (e.g., to take advantage of a break in the convective weather system). Implementing APCDM in a dynamic rolling-horizon framework showed promise as a means of optimizing routes for imminent flight plans while taking into account future air traffic as well, and showed the efficacy for using APCDM as a Tier 2 model under NextGen. Better, more up-to-date information seemed to allow for lower system-wide costs without sacrificing run times.

We developed and implemented several algorithms designed to bisect a polygonal region such that the total weight of grid points assigned to either resulting half-space is balanced, with the motivation of generating sector configurations within an airspace. Both hypothetical and realistic examples were solved to illustrate the algorithms and provide insights into their relative performance with respect to computational effort and the quality of solutions provided. Among the proposed methods, Algorithms 1.1 and 2, both of which were designed based on suitable pairwise edge considerations, yieled relatively competitive results. Algorithm 1.1, in which we solve problem RBP for each pair of edges in a polytope, required more computational effort, but produced superior solutions depending on the importance placed on reducing coordination workload. On the other hand, Algorithm 2, in which a pair of edges is determined beforehand, provided relatively balanced solutions quickly but was not as successful at reducing the coordination workload.

Possible areas for future research with respect to dynamic resectorization include determining alternative methods for computing the ϵ-perturbations used to avoid degeneracy, and developing heuristic approaches for finding good initial solutions to pass to the optimization software. The development of rules for triggering sector redefinitions is also a potential research area.

Another important topic for future research is to derive techniques for assigning grid point locations. The idea would be similar to that of quadrature in numerical integration. A relatively low number of well-placed grid points might be able to allow for an accurate workload representation while yielding faster computational times for the AOM and (P)AEM, as well as for a more refined partitioning process.

Future research areas for APCDM include a two-stage stochastic extension of the APCDM model. In such a scenario, the imminent flight plan selection decisions could be considered as first-stage variables, and other future time period flight plan selections could be conditioned on several alternative scenarios pertaining to the stochastic dynamic weather pattern and predicted sector capacities, and would constitute second-stage variables. It would be of interest to examine differences in the nature of solutions produced under a single versus multiple weather pattern scenarios, and to investigate if some equivalent aggregate weather pattern scenario can capture the observed effects of considering multiple scenarios in such a two-stage stochastic programming framework. Other future research topics of interest include the evaluation of side-payments as part of the trade offer and equity structure, as suggested by Vossen (2002), and the use of S-shaped value functions (Kirkwood 1997) to more generally reflect airlines' assessments of delay impacts within the defined relative performance ratios (see McCrea (2006) for some results in this vein).

Bibliography

Abdelghany, K., Abdelghany, A. and Niznik, T.: 2007, Managing severe airspace flow programs: The airlines' side of the problem, *Journal of Air Transport Management* **13**(6), 329–337.

Ahn, B. and Shin, J.: 1991, Vehicle routing with time windows and time-varying congestion, *Journal of the Operational Research Society* **42**, 393–400.

Ahuja, R. K., Magnanti, T. L. and Orlin, J. B.: 1993, *Network Flows: Theory, Algorithms and Applications*, Prentice-Hall, Upper Saddle River, NJ.

Balas, E.: 1974, Disjunctive programming: Properties of the convex hull of feasible points, Management Science Research Report 348, GSIA, Carnegie Mellon University.

Bazaraa, M. S., Jarvis, J. J. and Sherali, H. D.: 2004, *Linear Programming and Network Flows*, 3rd edn, John Wiley & Sons, Inc., New York, NY.

Bellman, R. and Dreyfus, S.: 1962, *Applied Dynamic Programming*, Princeton University Press, Princeton, NJ.

Bertsimas, D. J. and Stock Patterson, S.: 1998, The multi-airport flow management problem with en route capacities, *Operations Research* **46**(3), 406–422.

Bertsimas, D. J. and Stock Patterson, S.: 2000, The traffic flow management rerouting problem in air traffic control: A dynamic network flow approach, *Transportation Science* **34**(3), 239–255.

Bertsimas, D., Lulli, G. and Odoni, A.: 2009, An integer optimization approach to large-scale air traffic flow management, *Manuscript, Sloan School of Management and Operations Research Center, MIT, Cambridge, MA* .

Bilimoria, K. D. and Lee, H. Q.: 2005, Analysis of aircraft clusters to measure sector-independent airspace congestion, *AIAA 5th Aviation, Technology, Integration, and Operations Conference (ATIO)* .

Boag, C., Neal, A., Loft, S. and Halford, G. S.: 2006, An analysis of relational complexity in an air traffic control conflict detection task, *Ergonomics* **49**(14), 1508–1526.

Chabini, I.: 1998, Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time, *Transportation Research Record* **1645**, 170–175.

Cooke, K. L. and Halsey, E.: 1966, The shortest route through a network with time-dependent internodal transit times, *Journal of Mathematical Analysis and Applications* **14**(3), 493–498.

Daganzo, C.: 2002, Reversibility of the time-dependent shortest path problem, *Transportation Research Part B* **36**, 665–668.

Delahaye, D., Alliot, J.-M., Schoenauer, M. and Farges, J.-L.: 1994, Genetic algorithms for partitioning air space, *10th IEEE Conference on Artificial Intelligence Applications*, San Antonio, TX, March 1–4.

Delahaye, D. and Puechmorel, S.: 2006, 3D airspace sectoring by evolutionary computation, *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, Seattle, WA, July 8–12, pp. 1637–1644.

Dijkstra, E.: 1959, A note on two problems in connection with graphs, *Numerische Mathematik* **1**, 269–271.

Dreyfus, S.: 1969, An appraisal of some shortest path algorithms, *Operations Research* **17**, 395–412.

FAA: 2002, *National Airspace Redesign Strategic Management Plan*, Federal Aviation Administration, Internet website: `http://www.faa.gov/air_traffic/nas_redesign/`.

FAA: 2004, *Airport Capacity Benchmark Report*. Miami International Airport.

FAA: 2005, *National Severe Weather Playbook*, Federal Aviation Administration, Internet website: `http://www.fly.faa.gov/playbook/pbindex.html`.

FAA: 2006, *Advisory Circular 90-102: Airspace Flow Program*.

FAA: 2008, *Order JO 7210.3V, Facility Operation and Administration*, Federal Aviation Administration.

FAA: 2009, *FAA NAS Architecture*, Federal Aviation Administration, Internet website: `http://nas-architecture.faa.gov`.

flightaware.com: 2008, Internet website, `http://www.flightaware.com`.

Garey, M. R. and Johnson, D. S.: 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, NY.

Garey, M. R., Johnson, D. S. and Stockmeyer, L.: 1974, Some simplified NP-complete problems, *STOC '74: Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, ACM, New York, NY, USA, pp. 47–63.

Hall, R.: 1986, The fastest path through a network with random time-dependent travel times, *Transportation Science* **20**, 182–188.

Halpern, H.: 1977, Shortest route with time dependent length of edges and limited delay possibilities in nodes, *Mathematical Methods of Operations Research* **21**(3), 117–124.

Helme, M.: 1992, Reducing air traffic delay in a space-time network, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Chicago, IL, pp. 236–242.

Ishutkina, M. A., Feron, E. and Bilimoria, K. D.: 2005, Describing air traffic complexity using mathematical programming, *AIAA 5th Aviation, Technology, Integration, and Operations Conference (ATIO)*, Arlington, Virginia.

Ji, X., Iwamura, K. and Shao, Z.: 2007, New models for shortest path problem with fuzzy arc lengths, *Applied Mathematical Modelling* **31**, 259–269.

Kaufhold, R., Marx, S., Müller-Berthel, C. and Nachtigall, K.: 2007, A pre-tactical generalised air traffic flow management problem, 7th USA/Europe Air Traffic Management R&D Seminar, Barcelona, Spain.

Kaufman, D. and Smith, R.: 1993, Fastest paths in time-dependent networks for intelligent vehicle-highway systems application, *Intelligent Vehicle-Highway Systems Journal* **1**, 1–11.

Kirkwood, C.: 1997, *Strategic Decision Making: Multiobjective Decision Analysis with Spreadsheets*, Duxbury Press, New York, NY.

Klein, A., Kopardekar, P., Rodgers, M. D. and Kaing, H.: 2007, Airspace playbook: Dynamic airspace reallocation coordinated with the national severe weather playbook, *AIAA Aviation Technology, Integration and Operations (ATIO) Conference*, Belfast, UK.

Klein, A., Rodgers, M. D. and Kaing, H.: 2008, Dynamic FPAs: A new method for dynamic airspace configuration, *Integrated Communications, Navigation and Surveillance Conference*, Bethesda, MD, May 5–7, pp. 1–11.

Koehler, G. J.: 1990, Considerations for mathematical programming models in discriminant analysis, *Managerial and Decision Economics* **11**(4), 227–234.

Kopardekar, P., Bilimoria, K. and Sridhar, B.: 2007, Initial concepts for dynamic airspace configuration, *7th AIAA Aviation Technology, Integration and Operations Conference (ATIO)*, Belfast, Northern Ireland, September 18–20.

Kopardekar, P. and Magyarits, S.: 2003, Measurement and prediction of dynamic density, *5th USA/Europe ATM R&D Seminar*, Budapest, Hungary.

Krozel, J., Jakobovits, R. and Penny, S.: 2006, An algorithmic approach for airspace flow programs, *Air Traffic Control Quarterly* **14**(3), 203–230.

Lee, P. U.: 2005, A non-linear relationship between controller workload and traffic count, *Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting*, Orlando, Florida.

Lindsay, K., Boyd, E. and Burlingame, R.: 1993, Traffic flow management modeling with the time assignment model, *Air Traffic Control Quarterly* **1**(3), 255–276.

Loft, S., Sanderson, P., Neal, A. and Mooij, M.: 2007, Modeling and predicting mental workload in en route air traffic control: Critical review and broader implications., *Human Factors* **49**(3), 376–399.

Lulli, G. and Odoni, A.: 2006, The European air traffic flow management problem, *Manuscript, Department of Aeronautics and Astronautics and Operations Research Center, Room 33-219, MIT, Cambridge, MA 02139* .

Lulli, G. and Odoni, A.: 2007, Characteristics of delay assignments in European air traffic flow management, 11th World Conference on Transport Research, Berkeley, CA.

Martinez, S., Chatterji, G., Sun, D. and Bayen, A.: 2007, A weighted-graph approach for dynamic airspace configuration, *AIAA Guidance, Navigation, and Control Conference*, Hilton Head, SC, August 20–23.

McCrea, M. V.: 2006, *Slot-Exchange Mechanisms and Weather-Based Rerouting within an Airspace Planning and Collaborative Decision-Making Model*, Ph.D. dissertation, Virginia Polytechnic Institute and State University, Blacksburg, VA.

McCrea, M. V., Sherali, H. D. and Trani, A. A.: 2008, A probabilistic framework for weather-based rerouting and delay estimations within an airspace planning model, *Transportation Research Part C* **16**(4), 410–431.

Mukherjee, A. and Hansen, M.: 2009, A dynamic rerouting model for air traffic flow management, *Transportation Research Part B* **43**(1), 159–171.

NASA: 2006, Next generation air transportation system (NGATS) air traffic management (ATM)-airspace project, *Reference material*, National Aeronautics and Space Administration, Ames Research Center.

Odoni, A.: 1987, *Flow Control of Congested Networks*, Springer-Verlag, Berlin, chapter The Flow Management Problem in Air Traffic Control.

OMB: 2009, Budget documents for fiscal year 2010, *Budget of the United States Government*, Office of Management and Budget.

Orda, A. and Rom, R.: 1990, Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length, *Journal of the Association for Computing Machinery* **37**(3), 607–625.

Public Law 108-176: 2003, Vision 100 – Century of Aviation Reauthorization Act, `http://www.jpdo.gov/vision_100_law.asp`.

Radio Technical Commission for Aeronautics (RTCA): 1995, RTCA Task Force 3 free flight implementation report. RTCA: Washington, DC.

Raghavan, P. and Thompson, C. D.: 1987, Randomized rounding: A technique for provably good algorithms and algorithmic proofs, *Combinatorica* **7**, 365–374.

Richetta, O. and Odoni, A.: 1993, Solving optimally the static ground-holding policy problem in air traffic control, *Transportation Science* **27**(3), 228–238.

Richetta, O. and Odoni, A.: 1994, Dynamic solution to the ground-holding policy problem in air traffic control, *Transportation Research, Part A* **28**(3), 167–185.

Rubin, P. A.: 1997, Solving mixed integer classification problems by decomposition, *Annals of Operations Research* **74**, 51–64.

Sedgewick, R. and Vitter, J.: 1986, Shortest paths in Euclidean graphs, *Algorithmica* **1**(1), 31–48.

Sherali, H. D. and Adams, W. P.: 1990, A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems, *SIAM Journal on Discrete Mathematics* **3**(3), 411–430.

Sherali, H. D. and Adams, W. P.: 1994, A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems, *Discrete Applied Mathematics* **52**, 83–106.

Sherali, H. D., Hobeika, A. G. and Kangwalklai, S.: 2003, Time-dependent, label-constrained shortest path problems with applications, *Transportation Science* **37**(3), 278–293.

Sherali, H. D., Jeenanunta, C. and Hobeika, A. G.: 2006, The approach-dependent, time-dependent, label-constrained shortest path problem, *Networks* **48**(2), 57–67.

Sherali, H. D. and Shetty, C. M.: 1980, Optimization with Disjunctive Constraints, *Lecture Notes in Economics and Mathematical Systems*, Vol. 181, Springer Verlag, Berlin-Heidelberg, New York, NY.

Sherali, H. D., Smith, J. C., Trani, A. A. and Sale, S.: 2000, National airspace sector occupancy and conflict analysis models for evaluating scenarios under the free-flight paradigm, *Transportation Science* **34**(4), 321–336.

Sherali, H. D., Staats, R. W. and Trani, A. A.: 2003, An airspace planning and collaborative decision-making model: Part I - probabilistic conflicts, workload, and equity considerations, *Transportation Science* **37**(4), 434–456.

Sherali, H. D., Staats, R. W. and Trani, A. A.: 2006, An airspace planning and collaborative decision-making model: Part II - cost model, data considerations, and computations, *Transportation Science* **40**(2), 147–164.

Sherali, H., Ozbay, K. and Subramanian, S.: 1998, The time-dependent shortest pair of disjoint paths problem: Complexity, models, and algorithms, *Networks* **31**, 259–272.

Sud, V., Tanino, M., Wetherly, J., Brennan, M., Lehky, M., Howard, K. and Oiesen, R.: 2009, Reducing flight delays through better traffic management, *Interfaces* **39**(1), 35–45.

Tadenuma, K.: 2002, Efficiency first or equity first? Two principles of rationality of social choice, *Journal of Economic Theory* **20**, 462–472.

Terrab, M. and Odoni, A.: 1993, Strategic flow management for air traffic control, *Operations Research* **41**(1), 138–152.

Terrab, M. and Paulose, S.: 1992, Dynamic strategic and tactical air traffic flow control, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Chicago, IL, pp. 243–248.

Thomas, B. and White III, C.: 2007, The dynamic shortest path problem with anticipation, *European Journal of Operational Research* **176**, 836–854.

Trandac, H., Baptiste, P. and Duong, V.: 2002, A constraint-programming formulation for dynamic airspace sectorization, *Proceedings of the 21st Digital Avionics Systems Conference*, Irvine, CA, October 27–31, pp. 1.C.5–1 – 1.C.5.–11.

Vossen, T.: 2002, *Fair Allocation Methods in Air Traffic Management*, Ph.d. dissertation, University of Maryland, College Park, MD.

Vossen, T. and Ball, M. O.: 2006a, Optimization and mediated bartering models for ground delay programs, *Naval Research Logistics* **53**, 75–90.

Vossen, T. and Ball, M. O.: 2006b, Slot trading opportunities in collaborative ground delay programs, *Transportation Science* **40**(1), 29–43.

Vossen, T., Ball, M. O., Hoffman, R. L. and Wambsganss, M. C.: 2003, A general approach to equity in traffic flow management and its application to mitigating exemption bias in ground delay programs, *Air Traffic Control Quarterly* **11**, 277–292.

Vranas, P., Bertsimas, D. and Odoni, A.: 1994a, Dynamic ground-holding policies for a network of airports, *Transportation Science* **28**(4), 275–291.

Vranas, P., Bertsimas, D. and Odoni, A.: 1994b, The multi-airport ground-holding problem in air traffic control, *Operations Research* **42**(2), 249–261.

Wackerly, D., Mendenhall, W. and Scheaffer, R. L.: 2007, *Mathematical Statistics with Applications*, 7th edn, Duxbury Press.

Wanke, C. and Greenbaum, D.: 2007, Incremental, probabilistic decision making for en route traffic management, 7th USA/Europe ATM R&D Seminar, Barcelona, Spain, July 2–5.

Weisstein, E. W.: 2009, Voronoi diagram, From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/VoronoiDiagram.html`.

Xue, M.: 2008, Airspace sector redesign based on Voronoi diagrams, *AIAA Guidance, Navigation, and Control Conference*, Honolulu, HI, August 18–21.

Yousefi, A. and Donohue, G. L.: 2004, Temporal and spatial distribution of airspace complexity for air traffic controller workload-based sectorization, *AIAA 4th Aviation Technology, Integration and Operations (ATIO) Forum*, Chicago, IL, September 20–22.

Ziliaskopoulos, A. K. and Mahmassani, H.: 1993, Time-dependent shortest path algorithm for real-time intelligent vehicle/highway system, *Transportation Research Record* **1408**, 94–104.