Algorithms for Optimizing Multiple Routes Through Constrained Geometric Domains

A Dissertation Presented
by
Joondong Kim
to
The Graduate School
in Partial Fulfillment of the
Requirements
for the Degree of

Doctor of Philosophy
in
Applied Mathematics and Statistics
(Operations Research)

Stony Brook University
August 2010

Stony Brook University

The Graduate School

Joondong Kim

We, the dissertation committee for the above candidate for the Doctor of Philosophy degree,

hereby recommend acceptance of this dissertation.

Joseph S. B. Mitchell - Dissertation Advisor
Professor, Department of Applied Mathematics and Statistics

Esther M. Arkin - Chairperson of Defense
Professor, Department of Applied Mathematics and Statistics

Jiaqiao Hu - Committee Member
Assistant Professor, Department of Applied Mathematics and Statistics

Jie Gao - Outside Member
Assistant Professor, Department of Computer Science

This dissertation is accepted by the Graduate School

Lawrence Martin

Dean of the Graduate School

Abstract of the Dissertation

Algorithms for Optimizing Multiple Routes Through Constrained Geometric Domains

by
Joondong Kim

Doctor of Philosophy
in
Applied Mathematics and Statistics
(Operations Research)

Stony Brook University
2010

We study optimization problems associated with routing multiple moving objects through a two- or three-dimensional geometric domain. We focus primarily on trajectories for moving disks, which can be used to model motion of objects (e.g., aircraft) that are required to remain separated by certain distances. Our primary motivation comes from air traffic management. We examine three main types of problems; for each type, we consider variations, given hardness results, and present algorithms for approximation or special cases.

The first type of problem is that of finding multiple routes for moving objects through a given geometric domain. The moving objects must avoid obstacles (holes) within the domain. The objective is to maximize the number of possible routes through the domain, for moving objects that enter through a source (portion of the boundary) and exit through a sink (portion of the boundary). We study two variations of the problem: that in which trajectories of moving objects are required to be straight and that in which trajectories form a bounded degree tree, such as arises in arriving aircraft that must merge as they approach an airport.

A second type of problem is to estimate the "capacity" of a domain, defined to be the maximum number of possible routes through the domain from source to sink. We are not required to compute the actual routes, but only to determine the maximum number possible. The exact solution can be found using a geometric version of the theory of maximum flows and minimum cuts in a network. We demonstrate that one can compute an approximate solution using the notion of geometric spanner graphs, such as the Delaunay diagram. Further, we perform a sensitivity analysis of capacity estimation, under probabilistic models of uncertainty in the description of the domain. This problem arises in applications to air traffic management in the face of uncertainty in weather forecasts.

The third type of problem involves scheduling the dispatch of the moving objects, each of which is required to follow a pre-established route. In this situation, we consider the domain to be decomposed into a number of sub-regions (e.g., sectors, in the air traffic control scenario), and the goal is to determine dispatch times for each object such that we minimize the maximum number of objects ever simultaneously in a single sub-region. Typically, the dispatch times are constrained to be within some time interval. The problem is motivated by flight scheduling in order to optimize capacity in the air traffic control system. We examine the algorithmic complexity, propose algorithms, and conduct performance experiments for instances using actual and synthetic air traffic and weather data.

Contents

List of Figures

List of Tables

Acknowledgments

The past few years are full of memorable days. It is because of support from others.

First and foremost, I would like to thank my advisor Professor Joseph S. B. Mitchell for his merciful aid. I owe much to Joe in academic guidance and lots of other affairs.

I appreciate Professor Esther M. Arkin for exciting discussions, fascinating lectures and giving me breakthroughs when they were necessary. I also thank Professor Jie Gao and Professor Jiaqiao Hu for serving as committee members on my dissertation committee and for constructive comments.

My growth as a researcher was made possible due to my associations with faculty members and fellow graduate students of the Algorithms Reading Group, including Professor Steven Skiena, Professor George Hart, Professor Michael Bender, Girishkumar Sabhnani, Jason Zou, Shang Yang, Dr. Valentin Polishchuk, Irina Kostitsyna, Justin Iwerks, Mohammad Irfan, Ashish Lohia, Dr. Eli Packer and Rik Sakar. Thank you all for the fun discussions we had.

I am grateful to Dr. Jimmy Krozel and Dr. Joseph Prete of Metron Aviation and Professor Sándor P. Fekete and Dr. Alexander Kröller of Braunschweig University of Technology for sharing their valuable knowledge.

Finally, to my family and friends, thank you so much for being patient with me.

List of Publications

[1] E. M. Arkin, S. P. Fekete, J. Kim, J. S. B. Mitchell, G. R. Sabhnani, and J. Zou. The pencil packing problem. In *The 19th Fall Workshop on Computational Geometry*, Medford, MA, November 2009.

[2] E. M. Arkin, G. Hart, J. Kim, I. Kostitsyna, J. S. B. Mitchell, G. Sabhnani, and S. S. Skiena. The embroidery problem. In *Proceedings of the 20th Canadian Conference on Computational Geometry (CCCG2008)*, page 135138, 2008.

[3] M. Irfan, J. Iwerks, J. Kim, and J. S. B. Mitchell. Guarding polyominoes. In *Proc. 19th Fall Workshop on Computational and Combinatorial Geometry*, Tufts University, Nov 13-14, 2009.

[4] J. Kim, Z. Jingyu, J. Krozel, and J. S. B. Mitchell. Sensitivity of capacity estimates given convective weather constraints. In *AIAA Guidance, Navigation, and Control Conference*, Chicago, Illinois, United States, Aug. 2009.

[5] J. Kim, A. Kroeller, J. S. B. Mitchell, and G. Sabhnani. Scheduling aircraft to reduce controller workload. In *Proceedings of the 9th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS 09)*, 2009.

[6] J. Kim, J. S. B. Mitchell, V. Polishchuk, and A. Vihavainen. Routing a maximum number of disks through a scene of moving obstacles. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 230–231, College Park, MD, USA, 2008. ACM.

[7] J. Kim, J. S. B. Mitchell, and J. Zou. Approximating maximum flow in polygonal domains using spanners. In *Proceedings of the 21st Canadian*

Conference on Computational Geometry (CCCG2009), page 115118, UBC, Vancouver, Canada, Aug. 2009.

[8] J. Kim, J. S. B. Mitchell, and J. Zou. Routing parallel and merging lanes. In *The 19th Fall Workshop on Computational Geometry*, Medford, MA, November 2009.

[9] J. Krozel, J. Prete, J. S. B. Mitchell, J. Kim, and J. Zou. Capacity estimation for super-dense operations. In *AIAA Guidance, Navigation, and Control Conf.*, Aug 2008.

[10] P. Montes, H. Memelli, C. B. Ward, J. Kim, J. S. B. Mitchell, and S. Skiena. Optimizing restriction site placement for synthetic genomes. In A. Amir and L. Parida, editors, *CPM*, volume 6129 of *Lecture Notes in Computer Science*, pages 323–337. Springer, 2010.

[11] J. Prete, J. Krozel, J. S. B. Mitchell, J. Kim, and J. Zou. Flexible, performance-based route planning for Super-Dense operations. In *AIAA Guidance, Navigation, and Control Conference*, Aug. 2008.

[12] S. Yang, J. Kim, J. Krozel, J. S. B. Mitchell, V. Polishchuk, and J. Zou. Maximum flow rates for capacity estimation in level flight with hard and soft constraints. Manuscript (submitted), 2009.

[13] S. Yang, J. Kim, J. S. B. Mitchell, V. Polishchuk, and J. Zou. Routing multi-class traffic flows in the plane. page submitted, 2010.

[14] J. Zou, J. Kim, J. W. Krozel, J. Krozel, and J. S. B. Mitchell. Two methods for computing directional capacity given convective weather constraints. In *AIAA Guidance, Navigation, and Control Conf.*, Aug 2009.

Chapter 1

Introduction

This thesis addresses some algorithmic problems in computational geometry related to routing and scheduling. Most of the problems we discuss here are motivated by challenges arising in the *Air Traffic Management (ATM)* domain. In the ATM system, a large number of aircraft move through an airspace. The aircraft must be properly monitored by one or more air traffic controllers in order to avoid traffic congestion, delays in departures/arrivals, and, most importantly for safety, "conflicts" between aircraft, when two aircraft come too close to each other. In recent years, demands on the ATM system have increased, as the demand for more flights between more pairs of cities has risen. As air traffic demands increase, it becomes more important to devise effective automation techniques for capacity enhancement in the ATM system. There is limited space that must be effectively managed in order to optimize its utilization. The problem becomes especially challenging when disruptions, such as hazardous weather, impact part of the airspace, effectively rendering it useless, or at least decreasing its capacity. Safety is paramount; thus, all decisions must err on the side of caution.

In this thesis, we investigate some of the challenges facing the near-term and far-term future of ATM. In particular, we apply mathematical and algorithmic methodologies to specific problems that arise in our model of certain ATM challenges. We utilize some techniques from *computational geometry*, *graph theory*, *scheduling* and *optimization*. Also, through our modeling efforts, we introduce new theoretical problems, which reflect certain aspects of real ATM problems.

We use mathematical terminology in describing our ATM models. We use

Euclidean space for the airspace. General airspace is modeled as 3-dimensional space, with coordinates (x, y, z). However, in many situations the scope of problem is on ground operations or on flights at a specific altitude; in such cases, we often find that 2-dimensional Euclidean space, (x, y), suffices for our model for the problem. While actual 2-dimensional space should be on the surface of a (near) sphere, representing constant altitude above the earth, for most of our problems we simplify the model to consider the 2-dimensional Euclidean plane, ignoring the curvature of the earth. Our 2-D methods can usually be generalized to apply to a manifold, such as a sphere,

In the case that 2-dimensional obstacles (e.g., hazardous weather "constraints", where aircraft should avoid flying) are in motion in 2-dimensional space, we consider an additional time dimension and speak of 3-dimensional *space-time*, (x, y, t). In the case that our airspace is modeled in full 3-D, moving aircraft and obstacles are represented in 4-dimensional space-time, (x, y, z, t). Obstacles within the airspace are typically modeled as polygons in 2-D or polyhedral regions (e.g., polygonal-based cylinders) in 3-D.

Within the airspace, an aircraft is generally represented by a point at the center of a moving (2-D) disk or a (3-D) cylinder ("hockey puck") representing the *protected airspace zone* (PAZ) around the aircraft, into which no other aircraft should venture (an encroachment represents a "conflict"). The PAZ of an aircraft moving along a trajectory sweeps out a region of space-time that we often refer to as a "tube" or a "thick path". The trajectories of real aircraft are constrained by dynamic considerations (minimum and maximum speed and acceleration). Further, actual routes often obey certain geometric properties, such as *monotonicity* with respect to some direction (e.g., the vector from origin to destination), straightness, or near straightness (having a few number of non-sharp turns). These model simplifications for routes may help to make the problems we study tractable (e.g., in our study of the "pencil packing problem").

One fundamental class of problems in the ATM domain is that of computing routes for aircraft. The problem is, in general, quite hard from an algorithmic complexity point of view. In particular, even finding a single shortest route for a point moving among 3-dimensional polyhedral obstacles is known to be NP-Hard [7]); thus, we do not expect to find efficient (polynomial-time) algorithms for such routing problems. In order to address the multi-aircraft routing problems in ATM, we consider simpler cases, such as straight routes within a restricted airspace, or piecewise-linear routes that lie on a discrete network embedded in space.

Of course, straight routes are generally preferred over highly contorted routes, for efficiency's sake. (More realistically, we are interested in wind-optimized routes; we can think of the metric space as being distorted so that such wind-optimized routes are "straight".) Thus, we study a specific multi-aircraft routing problem in the form of an optimal packing problem for cylinders ("pencils") in space-time. Chapter 2 addresses this problem by first showing that even in its simplest form the routing of multiple aircraft (corresponding to the packing of multiple cylinders) is NP-hard. We complement these negative results with positive results, including approximation algorithms for maximizing the number of routes that can be packed, and experimental results on the performance of our proposed heuristics. Chapter 2 is based on joint work with Esther M. Arkin, Sándor P. Fekete, Joseph S. B. Mitchell, Girishkumar Sabhnani and Jingyu Zou; a preliminary version was presented at the Fall Workshop on Computational Geometry 2009 [1].

In Chapter 3, we address the complexity of the problem of computing multiple thick paths (air lanes) at a common flight level (or on a manifold) that are pairwise-disjoint (parallel flows) or that form a "merge tree" for traffic arriving to an airport or an air portal (cluster of nearby airports). A merge tree is an embedded directed tree whose edges are thick paths (avoiding weather hazards and other constraints). At internal nodes of the tree, two (or more) thick paths come together; this represents flows of air traffic that merge together to form a single flow out of the node. In typical current practice, the in-degree of each node is two, representing the fact that controllers typically merge only two arriving flows together at a time; merging three or more active flows represents a situation of high "complexity", which is generally avoided. The merge tree has a sink node (out-degree zero) at an "arrival fix" at an airport. There are source nodes (in-degree zero) at locations on the outer boundary of the region of interest (e.g., the "transition airspace") where flows of arriving aircraft enter the region on their way to the sink node at the airport. In Chapter 3, we prove that it is NP-hard to determine if there exists a merge tree on k sources among a set of obstacles (polygons having a total of n vertices) in the plane, where k and n are part of the input. This chapter is based on joint work with Joseph S. B. Mitchell and Jingyu Zou; a preliminary paper describing these results was presented at the Fall Workshop on Computational Geometry 2009 [26]. Related work on computing merge trees in practical settings (e.g., when k is considered to be constant) is investigated in our joint work with Joseph Prete and Jingyu Zou. [45, 61].

In ATM, one often wants to be able to estimate the "capacity" of a specified airspace to accommodate air lanes of traffic flow. The air lanes (thick paths) must avoid intersection with each other and with the obstacles (e.g., hazardous weather constraints). In fact, we may require that there be at least a certain separation between air lanes and between an air lane and an obstacle; such constraints can be handled by adjusting the widths of the thick paths and by "growing" the obstacles, using a Minkowski sum with a disk of radius equal to the required offset. The problem of computing how many thick paths can be routed through a polygonal domain, entering the domain along a "source" edge and departing along a "sink" edge, is an instance of the geometric maximum flow problem, as studied by Mitchell and Strang [36, 56]. The classic theory of max-flow/min-cut in networks has been extended to geometric domains (the "continuum"), and efficient (roughly quadratic time) algorithms are known for computing minimum length *cuts*, paths that separate source from sink, where the cost of the path is zero for portions of the path that lie within a hole (obstacle) of the domain. The further development of this theory for exact capacity computation and the application of these techniques to the ATM domain has been investigated in prior related work [43, 38, 29].

In Chapter 4, we study the computation of *approximate* minimum cuts, in nearly linear time, using the notion of geometric spanner graphs. A subgraph G' of a graph G is a *t-spanner* for G if for any pair of nodes of G, there exists a path within G' joining the two nodes such that the length of the path is at most t times the length of a shortest path between them in G. We utilize the theory of geometric spanner graphs to prove results about computing approximate lane capacity of an airspace, taking into account the integrality of the number of air lanes that can be routed. In addition to theoretical bounds on the approximation, we give experimental results that show that the methods are practical, yielding a very fast method of computing approximate capacity of an airspace.

Many of the obstacles that arise in our model of routing problems in ATM are given by weather forecast data, representing regions that are predicted to contain hazardous weather. Since there can be significant uncertainty associated with these obstacles, it is important to investigate how sensitive our capacity estimates (for the number of air lanes that can be accommodated) are to changes in the weather forecast. In Chapter 5, we conduct a sensitivity study. Specifically, we model three sources of uncertainty in hazardous weather constraints that arise from weather forecast data: (1) uncertainty in the severity of the weather, which may cause the predicted intensity values of a convective weather forecast to be shifted up or down, resulting in the associated obstacles being larger or smaller than nominally predicted; (2) uncertainty in the translational position of the predicted storms; and, (3) uncertainty in the timing of the forecast, including the speed with which the storm moves through the region. We present simple models of each type of uncertainty and provide experimental analysis of the sensitivity of capacity estimates that are based on minimum cut computations (considering the capacity to be the number of air lanes that can be routed through the impacted airspace).

The management of airspace is typically done by considering a partition of airspace into management units. In North America, this means that the National Airspace System (the "NAS") is partitioned into about twenty *centers*, each of which is partitioned into about 20-30 *sectors*, each of which is monitored by one or more air traffic controllers. The partitions are of three-dimensional airspace, but they are often organized by altitude for simplicity, particularly for the regions of airspace that are "en route", between major airports. Thus, we often consider the two-dimensional case and examine at a fixed altitude the set of polygonal regions into which space is partitioned. Within a sector there can be multiple aircraft at any given time; however, for safety, there is an upper bound (a "peak aircraft count" or "capacity") on the number of aircraft permitted within a single sector at any moment, since the controller(s) monitoring the sector must be able to communicate with each pilot and be able to reroute flights as needed, to avoid conflicts between pairs of aircraft. The capacity of a sector is related to the estimated "workload" that a controller can handle at any given moment. An estimate of workload is based solely on the count of how many aircraft are in a sector; more sophisticated notions of workload are based on the actual traffic patterns, weather events, climbing and descending aircraft, etc.

In Chapter 6 we study a scheduling problem that arises in the context of ATM and capacitated sectors, treating the capacity simply as a maximum aircraft count. By shifting, within limits, the departure times of scheduled flights, we are able to decrease the maximum number of aircraft scheduled to be in certain sectors. The problem is to compute a set of shifts in departures (e.g., delays) in order to minimize the maximum workload (aircraft count) in any given sector of the NAS. (Here, we consider the partition into sectors to be fixed.) We study the complexity of the problem and propose algorithmic solutions that we test experimentally on real data.

This thesis touches on a few of the many fascinating algorithmic problems that arise in the study of the air traffic system. Many of the problems generalize to other areas of transportation research, e.g., for ground-based travel on roads and rails. Our goal has been to make progress on a few important specific problems and to suggest future challenges where the techniques of computational geometry and algorithmic methods can be applied effectively.

Chapter 2

The Pencil Packing Problem

Abstract

We consider the following three-dimensional packing problem that arises in multi-body motion planning in an environment with obstacles: Given an $n \times n \times n$ regular grid of voxels (cubes), with each voxel labeled as "empty" or "occupied", determine the maximum number of "pencils" that can be packed within the empty voxels, where a pencil is a union of n voxels that form an axis-parallel strip (i.e., a $1 \times 1 \times n$ box). No pencil is allowed to contain an occupied voxel, and no two pencils can have a shared voxel (since they form a packing). We consider also the dual (covering) problem in which we want to find the minimum number of empty "covering" voxels such that every pencil is intersected by at least one covering voxel. We show that both problems are NP-Hard and we give some approximation algorithms. We have evaluated our approximation algorithms experimentally and found that they perform very well in practice.

Keywords: Set Packing Problem, Set Cover Problem, Voxel Cube

This chapter is based on joint-work with Esther M. Arkin of Stony Brook University, Sándor P. Fekete of Braunschweig University of Technology, Joseph S. B. Mitchell, Girishkumar Sabhnani and Jingyu Zou of Stony Brook University. A preliminary version was presented at the Fall Workshop on Computational Geometry, 2009 [1]

2.1 Introduction

We study a problem, which is motivated by *Air Traffic Management (ATM)* domain. For a given piece of airspace, the goal is to determine safe trajectories for aircraft, with no two trajectories overlapping in space-time, and each trajectory staying a safe distance from constraints (i.e., regions that are to be avoided, such as “no-fly zones”and regions that are impacted by hazardous weather).

Our focus in this chapter is on algorithms to compute *straight* (or essentially straight) trajectories in space-time. This is an important first step in any analysis of the fully general problem. In most cases, too, we desire essentially straight routes for flights through an airspace, for their efficiency and ease of control. (More generally, we may desire great circular routes or wind-optimized routes, which are most fuel efficient.) Further, we expect that our techniques can be extended to cases in which routes are polygonal and allowed to have a small number of bends at specified “waypoints”.

While it may seem that determining the existence of straight routes through a constrained airspace is relatively straightforward, we demonstrate that it is not trivial, as it is necessary to coordinate the timing and placement of multiple crossing flows of aircraft on straight trajectories. In fact, even if the aircraft are flying in just two orthogonal directions (e.g., x-parallel and y-parallel) through an airspace cluttered with (time-dependent) constraints, we demonstrate that it is NP-hard to decide if k aircraft can be routed through an airspace in a given window of time. We complement this negative result with positive results on the approximation of the optimization problem that seeks to maximize the number of straight trajectories, and we give experimental evidence that our proposed heuristics work well in practice. The model we study here is that of placing “pencils” (cylinders) through a cluttered three-dimensional space, where the third dimension represents time.

2.1.1 Motivation

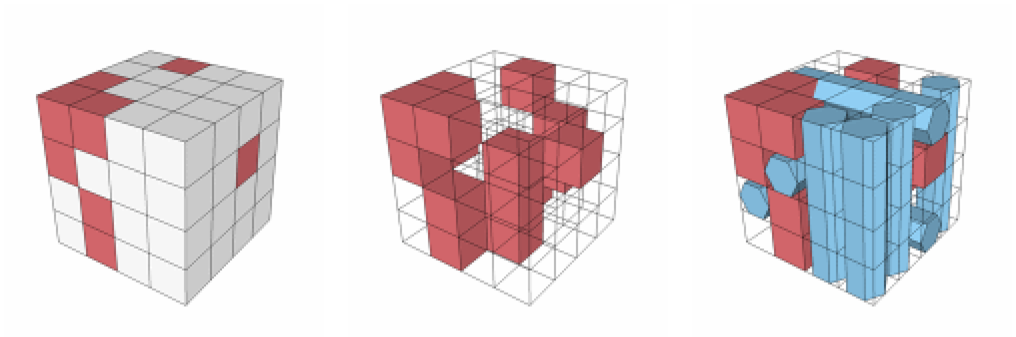
If the given airspace is depicted as 2 dimensional Euclidean space, then it could be approached by Max-Flow/Min-Cut technique[43]; but, finding multiple paths is generally hard problem. For example, It is known to NP-hard to find a shortest and obstacle avoiding path in 3D. Therefore, we give additional restriction to wanted paths, which is straightness.

Instead of arbitrary multiple of thick path in 2D or tube in 3D, we look for straight trajectories for moving objects, which are aircraft in airspace. Usually Airspace is partitioned as manageable size of sub regions, called sectors. Even non-straight path could be considered as chain of smaller straight paths, so segments of a path in smaller sectors look like straight.

We create a theoretic problem, which depict following situation in ATM domain. An ATM controller is required to place multiple aircraft in its responsible sector, so it looks for number of obstacle avoiding and non-overlapping paths from one side to another side. Following is formal statement of the problem.

2.1.2 Problem Statement

We are given an $n \times n \times n$ regular grid, C, of voxels (cubes), with each voxel labeled as "empty" or "occupied". An axis-parallel strip of n face-adjacent empty voxels is called a *pencil*; it is a $1 \times 1 \times n$ block of n empty voxels that passes all the way through C, from one facet of the cube to the opposite facet. Fig 2.1 shows an example with $8 \times 8 \times 8$ cube.



(a) A Cube consists of occupied and empty voxels (b) Red voxels are occupied (c) 8 Pencils are packed in the cube

Figure 2.1: Example of Pencils Packing Problem

Our goal is to pack within C as many pairwise-disjoint pencils as possible. The problem arises in a multi-agent motion planning problem for moving as many disks as possible along discrete-orientation straight trajectories among a set of possibly moving obstacles in the plane; by lifting to space-time (3D) and transforming the coordinates, we arrive at the pencil packing problem.

For convenience, we place C aligned to the x, y and z axes. We let c_{ijk} be a binary variable indicating if voxel indexed (i, j, k) is occupied ($c_{ijk} = 1$) or not ($c_{ijk} = 0$). We let x_{jk} be a binary variable indicating the presence of the x-parallel pencil at position (j, k) in (y, z)-space. We similarly define variables y_{ik} and z_{ij}. Then, the *Pencil Packing Problem* is defined as follows: For given $c_{ijk} \in \{0, 1\}$, where $i, j, k \in N = \{1, \ldots, n\}$,

$$\begin{aligned} \text{maximize} \; & \sum_{j\in N, k\in N} x_{jk} + \sum_{i\in N, k\in N} y_{ik} + \sum_{i\in N, j\in N} z_{ij} \\ \text{s.t.} \; & x_{jk} + y_{ik} + z_{jk} + c_{ijk} \leq 1 \quad i, j, k \in N \end{aligned}$$

Related to the pencil packing problem is a covering problem, defined as follows. Let v_{ijk} be a binary variable associated with voxel (i, j, k), indicating if this voxel is part of the cover. The *Voxel Cover Problem* is, for given $c_{ijk} \in \{0, 1\}$, where $i, j, k \in N$,

$$\begin{aligned} \text{minimize} \; & \sum_{i\in N, j\in N, k\in N} v_{ijk} \\ \text{s.t.} \; & \sum_{i\in N} (v_{ijk} + c_{ijk}) \geq 1 \quad j, k \in N \\ & \sum_{j\in N} (v_{ijk} + c_{ijk}) \geq 1 \quad i, k \in N \\ & \sum_{k\in N} (v_{ijk} + c_{ijk}) \geq 1 \quad i, j \in N \end{aligned}$$

The Voxel Cover Problem is the dual problem of Pencil Packing Problem.

2.1.3 Related Work

Our problems are geometric variants of Set Packing and Minimum Set Cover, which are known to be APX-hard even if the cardinality of each set is bounded ([18], [20], [39]). One previous work [10] shows that the maximum independent set in 3-partite graphs is NP-hard and not approximable within 26/25 unless P=NP. The pencil packing problem can be viewed as a special case of the independent set problem in 3-partite graphs.

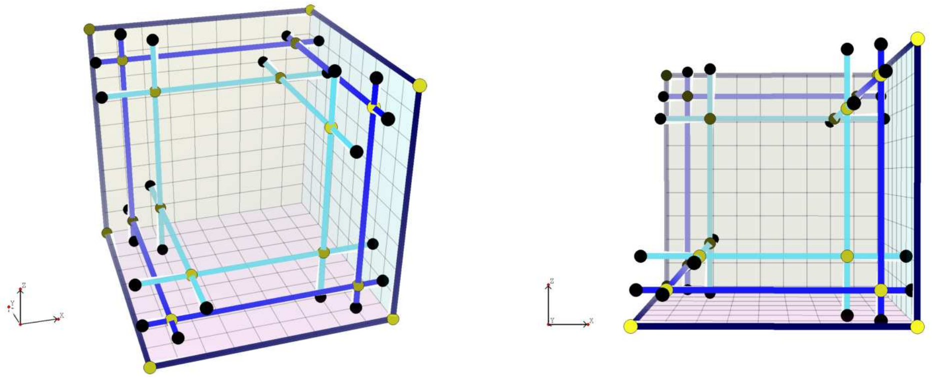


Figure 2.2: 3 variable gadgets are placed. Straight lines represent possible pencils and balls represent intersections and end points of pencils

2.2 Hardness

Using reductions from a variation of 3SAT Problem where each clause has exactly 3 variables [19], we prove the following hardness results:

Theorem 2.2.1. *NP-Hardness of Pencil Packing: Let C be a given voxel cube and k be an integer; then it is NP-complete to determine whether k pencils can be packed in C.*

Proof. Let P be a 3SAT problem instance with a variables and b clauses and each clause exactly 3 variables in it. We construct a voxel cube C, size of which is polynomial to a and b, and then show that finding $3a + 4b$ pencils (it is maximum possible number) in C is equivalent to solve P.

For each variable in P, we have a cycle of possible pencils (empty voxel strip size n) length 6. Fig. 2.2 shows gadgets for 3 variables. Colored straight lines represent possible pencils. It is possible to pack at most 3 pencils among 6 possible ones in each cycle. There are 2 different ways of packing 3 pencils for a variable gadget, so we adopt that aspect as variable assignment. One alternating 3 among 6 possible pencils cycle is true pencils and the rest is false pencils.

A clause gadget consist with 9 possible pencils like Fig. 2.3. 3 possible pencils (red in the figure) have a intersection voxel and each is relayed to variable gadgets with two additional possible pencils (green and purple). Fig. 2.4 shows a connected structure of a clause gadget and three variable gadgets.

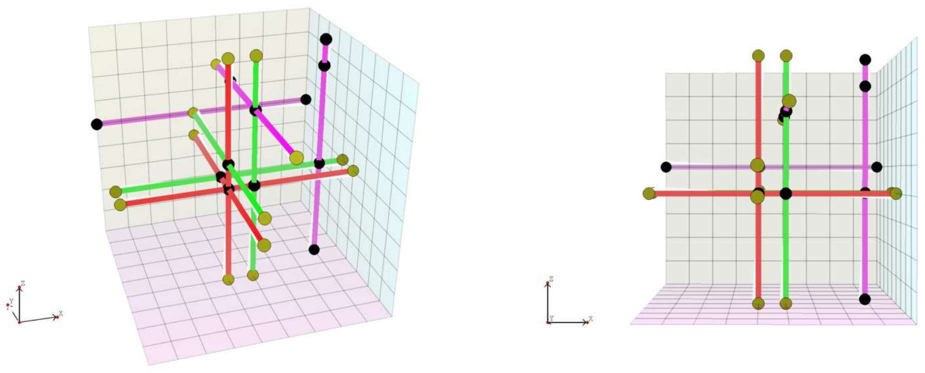


Figure 2.3: A clause gadget consists of 9 possible pencils.

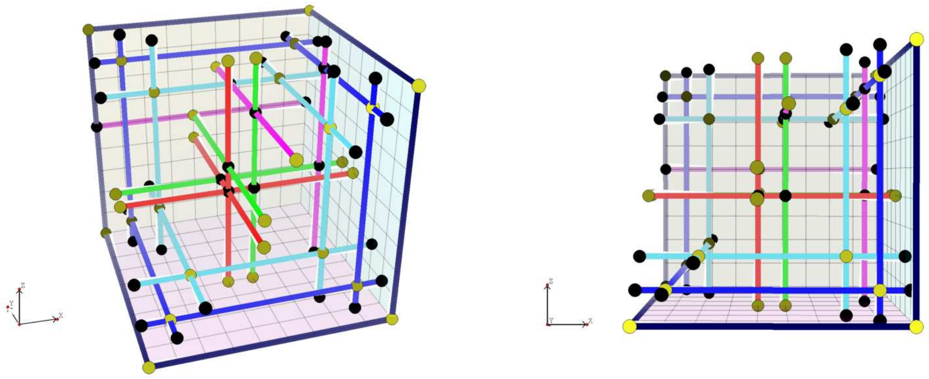


Figure 2.4: A clause gadget is connected to 3 variables gadgets.

One is able to pack 4 pencils in a single clause gadget and this is the maximum possible. When there are 4 pencils in a clause gadget, then at least one of them intersects the variable gadget. Otherwise (without using pink possible pencils), only 3 pencils could be packed. Therefore, if a clause has true literal of a variable, then the pink possible pencil should be connected to the false pencils of the variable.

A variable gadget has 2 voxel layers and a clause gadget has 3 voxel layers in each dimension. Since one of 3 clause possible pencils is place on the layer defined by variable possible pencils, one clause gadget introduces 2 new layers. In order to isolate possible pencils of gadgets, we also place additional layers as barriers. Hence, $4(a+b)-1$ layers are required to build C, which is corresponding to given P.

If we are able to pack $3a + 4b$ pencils in C, each variable gadget has 3 pencils in it and each clause gadget has 4 pencils in it. It implies that we are able to valid variable assignment and satisfy all the clause.

□

Theorem 2.2.2. *NP-Hardness of Voxel Cover: Let C be a given voxel cube and l be an integer; then it is NP-complete to determine whether l empty voxels of C exist that cover all pencils of C.*

Proof. We use a reduction from 3SAT problem to prove. Let P be an instance of 3SAT problem with a variables and b clauses. There are exactly 3 variables in each clause. We construct a voxel cube C corresponding to given P, and show finding minimum voxel cover of C is equivalent to solve P.

For a variable of P, we construct a cycle of possible pencils in C, whose length is multiple of 8. The smallest one is illustrated in Fig. 2.5(a). It has two intersections denoted as T and F, which are exclusively used to express how the variable participate in clauses. The length of a variable gadget cycle is $8l$, there are l pairs of T and F intersections. In such case, the gadget requires $3l$, $3l$ and $2l$ layers in dimensions and the variable is used l times in P totally. Since a variable gadget is even number cycle, one can cover it with the minimum number of voxel (which is the half of the cycle length) in exactly 2 different ways. If that minimum voxel cover contains T intersections, then we regard it as true assignment for corresponding variable. Otherwise, it is false.

A clause gadget consists of 7 possible pencils shown as blue in Fig. 2.5(d) and Fig. 2.6. It intersects variable gadgets 3 different possible pencils. Intersecting point should be T or F intersections and it shows how the variable participate in the particular clause. If none of intersecting points of variable and clause gadget are covered by voxels, which cover variable gadget cycles, then there are 7 possible pencils in clause gadget should be covered by additional 4 voxels. Otherwise (if any possible pencil of clause gadget is covered by a voxel, which is originally for variable gadget cycle), then we need 3 additional voxels for the clause gadget. Each clause gadget requires $0, 1$ and 1 voxel layers in each dimension.

The number of required voxel layer is proportional to the number of variable participation in all clauses. Since all clause has exactly 3 variables, variable gadgets require $9b, 9b$ and $6b$ voxel layers. And $0, 1b$ and $1b$ layers are required for clauses. If we consider the separation layers, then $20b - 1$ is the dimension of C. If we get voxel cover for all possible pencils in C where the cardinality of

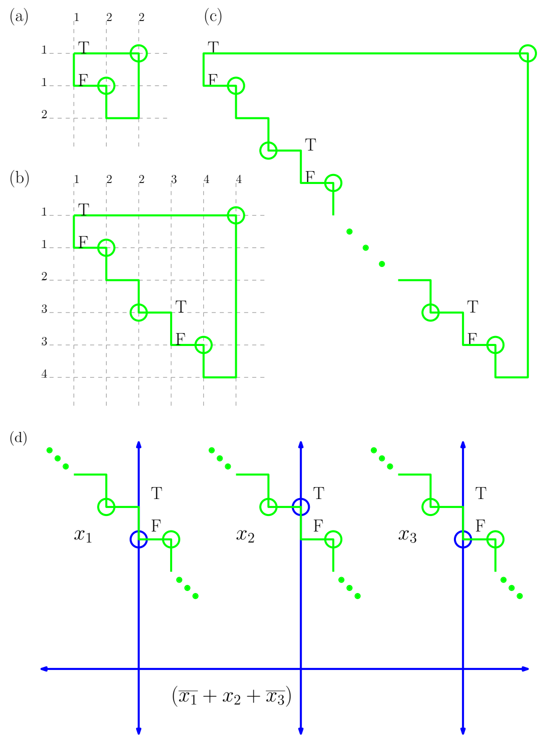


Figure 2.5: Gadgets for voxel cover problem. All graphics are top-viewed. Circles represent possible pencils perpendicular to the planes, where different oriented pencils are placed. (a) a variable gadget form a cycle of 8 possible pencils (shown as green). Number along the pencils identify the depth of voxel layers, which have green pencils (b) a variable gadget forms a 16 cycle (c) generalized variable gadget forms $8l$ cycle, where l is an integer (d) 3 variable gadgets (shown as green) interacting with a clause gadget (blue)

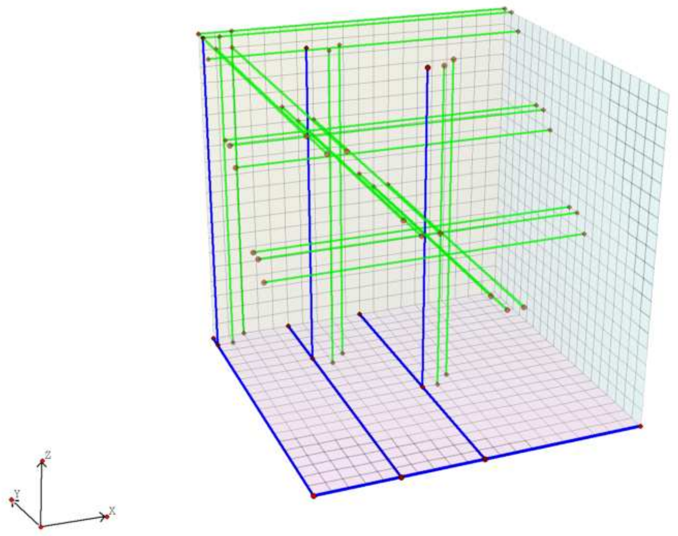


Figure 2.6: variable and clause gadgets in perspective view point

the cover is $4 \cdot 3 \cdot b + 3 \cdot b = 15b$ (which is the minimum possible number), then we are able to get the solution for P.

$\square$

2.3 Bound

If we have any solution for the voxel cover problem of a given C, then we know that at least one pencil is required to cover each voxel in the covering; thus, any voxel cover solution gives an upper bound for pencil packing. Also. LP relaxations of the above IP problems give upper and lower bounds on the optimal solutions. For a given cube C, let $OPT_p(C)$, $OPT_v(C)$ be the optimal solution of the Pencil Packing, Voxel Cover Problems, and let $LP_p(C)$, $LP_v(C)$ be the LP relaxations. We have the following relationship.

$$|OPT_p(C)| \leq |LP_p(C)| = |LP_v(C)| \leq |OPT_v(C)|$$

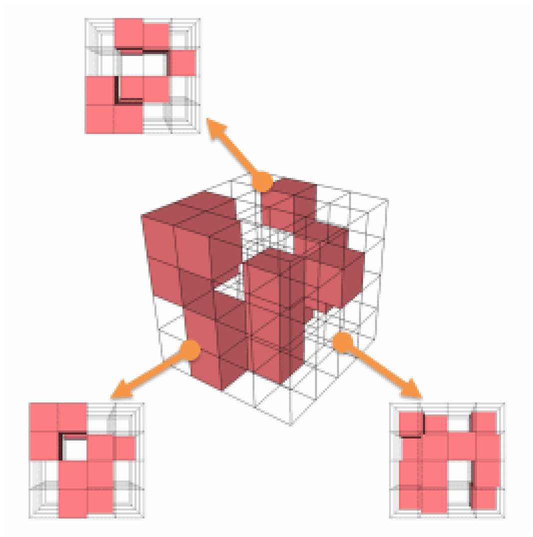
For a given cube C, the *duality gap* is the difference between $|OPT_p(C)|$ and $|OPT_v(C)|$.

2.4 Approximation

A very simple approximation is immediate: Restrict attention to pencils with the same orientation and pack as many as possible. This gives a 1/3-approximation, since one of the three orientations has at least $(1/3)|OPT(C)|$ pencils packed where $OPT(C)$ is optimal solution for given cube C. Fig 2.7(a) shows the idea.

2.4.1 Layered Approach

A better approximation comes from a layered approach. Consider each pair of orientations of pencils (e.g., x- and y-), and pack each layer (corresponding to different z-coordinates) optimally, with purely x-parallel or purely y-parallel pencils, whichever gives more pencils for that layer (which is easy to determine for each layer, by projecting onto a single axis). This gives a 2/3-approximation, since, in one choice of orientation pairs, we will be able to obtain at least as many pencils as OPT has in those two orientations. In fact, unless the number


(a) Naive Approximation

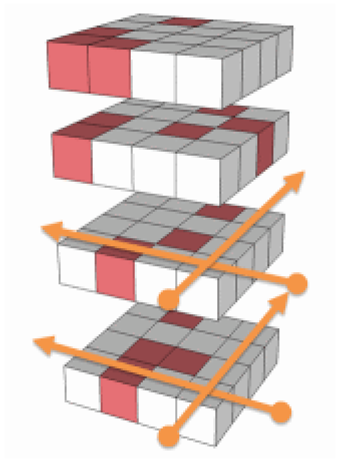

(b) Layered Approach

Figure 2.7: Approximation for Pencil Packing Problem

of pencils of OPT is about the same in all three orientations, this method gives better than a 2/3-approximation. Fig 2.7(b) gives an illustration example.

We have another 2/3-approximation with using matching, which is essentially same with layered approach. Let consider the tri-partite graph G of possible pencils. Each partite consists of pencils with same orientation. We get a bi-partite subgraph G_x of G after removing possible pencils of x-parallel. Then we are able to get maximum non-crossing pencils from G_x by matching. They are either y- or z-parallel pencils. Like the layered approach, there are two other ways of getting subgraph G_y and G_z, so one will take one of them, which provides the most number of pencils.

The worst performance of layered approach is exactly $\frac{2}{3}k$ pencils when the optimum packing gives $3k$ pencils. This situation happens when $3k$ pencils of optimum solution consists of three k pencils in each x-, y- and z-parallels. If one of three orientations gives strictly more than k in optimum solution, let's assume it is x-parallel without loss of generosity, it implies that one of orientation gives strictly less than k pencils (let's assume in z-parallel) in optimum solution. Then, layered approach in x- and y-parallels gives you strictly more than $2k$ pencils at least. It means that worst situation for layered approach gives exactly same number of pencils exist in optimum solution. The

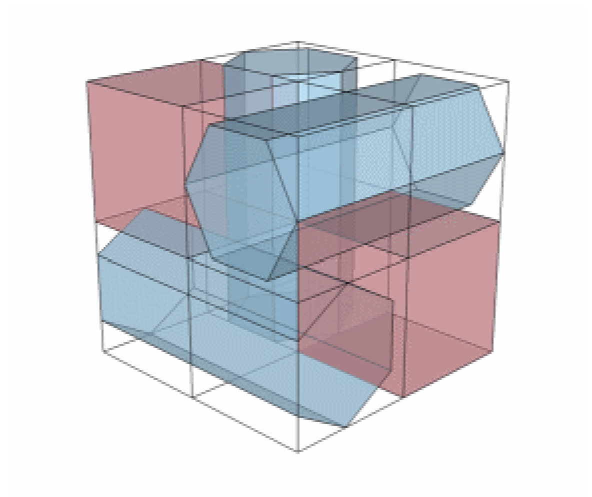


Figure 2.8: $2 \times 2 \times 2$ cube with 2 occupied red voxels. Optimum solution is shown as 3 light blue pencils, however Layered Approach only gives 2 pencils

smallest example, where layered approach gives $2k$ of $|OPT|$ is shown in Fig 2.8.

2.4.2 Random Packing

Let's consider the case, where Layered approach gives exactly $2k$ pencils when $|OPT(C)| = 3k$. We know the number of x-parallel possible pencils is at least k, otherwise we are not able to get $3k$ pencils in optimum solution, And also, the number of x-parallel possible pencils can not exceed $2k$, otherwise we may have more than $2k$ pencils. Therefore, the number of x-parallel possible pencils is between k and $2k$. Same argument holds for y- and z-parallel pencils.

Following strategy gives strictly more than $2k$ pencils for any cube C, where $|OPT(C)| = 3k$; randomly packing a pencils among x-parallel possible pencils, then do Layered approach in y- and z-parallels. If the optimum solution has different number of pencils in each orientation, then Layered approach gives more than $2k$. Otherwise, (if $OPT(C)$ has k pencils in each orientation), then one can have additional pencil in x- with probability more than half. Because k among less than $2k$ pencils are included in $OPT(C)$. This additional pencil is certainly obtained if one try $O(k)$ times in x-parallel pencils. If one tries $O(\binom{2k}{r})$ times of $r \leq k$ pencils random packing, then additional $2k + r$ pencils could be packed.

Without random pencil packing, we guarantee $2/3|OPT(C)|$ pencils in worst case, however we look for better approximation algorithm, which guarantee strictly better ratio. Since we only use the combinatorial property and no use geometric property of the problem, one may get other algorithms exploit those aspect.

2.5 Back to the Applications

The original motivation of this problem comes from ATM system. In a certain period of time and geometric region, there are demands for routing aircraft from one side to another side of airspace. Therefore controller needs to assign airways to aircraft while preventing potential conflicts among them. We discuss how we apply approximation schemes of the theory to practice.

2.5.1 Different oriented straight route in airspace

Suppose we are looking for routes of aircraft traveling airspace with 3 different orientations (for example, from East to West, South West or South) with constant speed. Objective is to maximize the number of routes. It could be affine transformed into pencil packing problem like Fig 2.9.

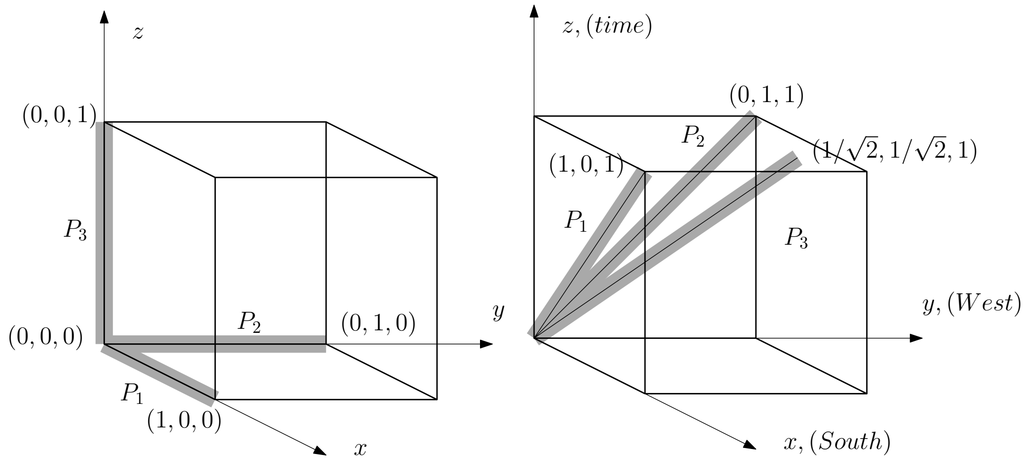
In Fig 2.9, for a coordinate p in axis parallel pencil could be transformed into speed 1 slant pencil with three orientations. $x \mapsto Ax + b$, where

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1/\sqrt{2} & 1/\sqrt{2} & 1 \end{pmatrix}, \quad b = 0$$

Since A is non-singular matrix, this affine transformation is invertible.

2.6 Experiments

We have implemented several methods for experimental comparison. In particular, we implemented a simple 1/3-APX, the layered 2/3-APX, a greedy algorithm to pack pencils, and a greedy algorithm to cover pencils (for use in an upper bound). We also used CPLEX to solve an IP formulation, as well



(a) Axis Parallel Pencils (b) Slant Pencils with Different Orientation

Figure 2.9: Translation between two kinds of Pencil Routing. One could be computed by an affine transformation of the other and vice versa.

$\lvert LayeredALG\rvert/\lvert OPT\rvert$	$< 70\%$	$< 80\%$	$< 90\%$	$< 100\%$
#Cases	1	13	80	261

Table 2.1: Number of pencils from Layered Approach and Optimum Solution

as the corresponding LP relaxation, for both the pencil packing and the pixel covering problems.

We ran the experiments on 1800 instances. We used grid sizes having $n = 5, 10, 15, 20, 25, 30$. We selected voxels to be occupied independently with probabilities $p = 0.2, 0.1, 0.05, 0.01, 0.005, 0.001$. We also considered random clusters of occupied voxels, generated as L_1 balls of radius 0, 1, 2, 3, 4 centered on a randomly selected voxel. For each choice of parameters, we ran 10 randomly generated instances. We found that the 2/3-APX works remarkably well in practice, most often yielding the optimal solution. We also found that the duality gap is usually 0. A detailed comparison is given in the full paper.

Acknowledgments

We thank Alexander Kröller and Valentin Polishchuk for helpful discussions. This work was partially supported by NSF (CCF-0528209, CCF-0729019), NASA Ames, and Metron Aviation.

Chapter 3

Routing Parallel and Merging Lanes

Keywords: Tree Routing, Graph Drawing, Air Traffic Management

3.1 Introduction

We study a graph embedding problem that arises in air traffic management (ATM): Given a planar polygonal domain Ω (with holes), a set of source terminals $\{s_i\}_{i=1}^{k}$ and a sink terminal t in Ω, and two constants $r, \epsilon > 0$, find a (r, ϵ)-"fat graph embedding" of a tree T in Ω such that the number of source terminals s_i that are connected to t by directed paths in T is maximized. Such a tree T can serve as the "fan-in" tree for merging air traffic flows within the transition airspace, when multiple flows of arriving aircraft must merge into a single flow crossing an arrival "metering fix" at the boundary of the terminal airspace.

We first consider a related problem of routing through Ω a maximum number of non-crossing paths in a particular drawing of a layered DAG in Ω, from a set of source terminals $\{s_i\}_{i=1}^{k}$ on the boundary of Ω (the first layer) to a set of sink terminals $\{t_i\}_{i=1}^{k}$ on the boundary of Ω (the final layer). We show that this "parallel lane routing" problem is NP-hard for polygonal domains Ω.

This chapter is based on joint-work with Joseph S. B. Mitchell and Jingyu Zou of Stony Brook University. A preliminary version was presented at Computational Geometry Fall Workshop 2009 [26]

We show that the tree routing problem in a given drawing of a layered DAG is NP-hard. We then consider the special case (typical in practice) in which there are a constant number of arcs in each path from s_i to t. For this case, we give a polynomial-time dynamic programming algorithm for the tree routing problem. We have implemented the dynamic programming algorithm and have performed experiments applying it to terminal airspace instances in which the obstacles are given by hazardous weather regions or special use airspace regions.

3.1.1 Motivation

As flows of arriving aircraft enter the "transition airspace", from en route to the terminal airspace in the vicinity of an airport, they are merged according to a fan-in tree, with each merge point having in-degree at most a small constant (typically 2). The routes must avoid certain "constraints", which include restricted airspace and hazardous weather cells. In order for a controller to have sufficient time to monitor the safe separation of aircraft between merges, there should be a minimum separation between one merge point and the next along a route.

Given a transition airspace cluttered with "constraints" (obstacles), and given a "demand" of desired arrival flows that need to merge to the arrival metering fix, our objective is to accommodate as many arriving flows as possible, given the constraints on the fan-in tree. This problem arising in air traffic management (ATM) motivates a tree embedding problem that we formalize next.

3.1.2 Problem Statement

The input to our problem is a planar polygonal domain Ω with a set of polygonal holes, $\mathcal{H} = \{H_i\}_{i=1}^{h}$, and a total of n vertices. There is a set of k points $\{s_i \in \Omega\}_{i=1}^{k}$ designated as the source terminals, and one point $t \in \Omega$ designated as the sink terminal. For any $r, \epsilon > 0$ given, our objective is to find a (r, ϵ)-"fat graph embedding" of a tree, $T = (V, E)$, in Ω connecting all of the source terminals (or as many terminals as possible) to the sink terminal. For each vertex $v \in V$, its "fat embedding" is an open disk of radius r, $D_r(v) \subset \Omega$. If we let D_ϵ be the open disk of radius ϵ centered at the origin, and for any path π in $\mathbb{R}^2$, let $(\pi)^\epsilon$ be the Minkowski sum of π and

D_ϵ, then for every edge $e = (u, v) \in E$ its fat embedding is an ϵ-thickened path $(\pi_e)^\epsilon \subset \Omega$ for some reference path π_e from $D_r(u)$ to $D_r(v)$. (The path π_e is $\pi_{(u,v)} \setminus (D_r(u) \cup D_r(v))$, for a path $\pi_{(u,v)}$ that embeds the edge linking point u to point v.) We require that the disks $\{D_r(v)|v \in V\}$ be pairwise disjoint, that the fat paths, $\{(\pi_e)^\epsilon|e \in E\}$, be pairwise disjoint, and that $(\pi_e)^\epsilon$ intersects only those disks, $D_r(u)$ and $D_r(v)$, that correspond to the endpoints of edge $e = (u, v)$. It should be noted that the maximum degree of T has to be bounded by some constant that is $O(r/\epsilon)$ because the thickened paths have to stay disjoint, even around some disk $D_r(v)$ where they are merging into.

3.1.3 Related Work

Our model of the fat graph embedding is similar to the model of "bold graph drawing" in [58], except that we allow non-straight edges and require stronger separation condition than the list of conditions given in [58] for good drawings. Our problem is also related to the thick edges drawing problem in [15]. For related work on the routing of thick paths and their application to ATM, see [2, 43].

3.2 Routing Parallel Lanes

We begin by considering a related problem involving routing of a set of non-crossing lanes. We are given a particular drawing of a layered DAG, $G = (V, E)$, in Ω, with $\{s_i\}_{i=1}^k \subset V$ on $\partial\Omega$ the source terminals and $\{t_i\}_{i=1}^k \subset V$ on $\partial\Omega$ the sink terminals. Given any integer $l \leq k$, we want to decide if there exists in G at least l paths connecting pairs of source terminals and sink terminals such that the drawings of the paths are pairwise disjoint. We prove this problem to be NP-hard using a reduction from INDEPENDENT SET. For any given graph G and given integer l, we construct an instance of our problem, shown in Figure 3.1, such that there exists an independent set of size l in G if and only if there exist at least l non-crossing paths in our instance.

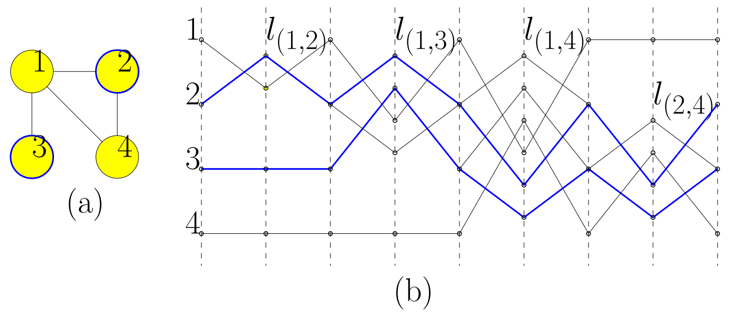


Figure 3.1: (a) An instance of INDEPENDENT SET, and (b) the corresponding layered DAG.

3.3 Routing Merging Lanes

In our motivating application, the portion of transition airspace under consideration is typically modeled as a portion of an annulus, corresponding to flights approaching an arrival metering fix. (Typically, there are four arrival metering fixes for a major airport.) Specifically, we work with a quadrant-shaped region $\Omega = \{(r\cos\theta, r\sin\theta) \in \mathbb{R}^2 | \theta_{\min} < \theta < \theta_{\max}, r_{\min} < r < r_{\max}\}$. Our application also requires that along each flight trajectory the distances between consecutive merge points be at least L and the in-degree of each merge point be at most a small constant (typically 2). We establish within Ω a set of concentric circular arcs ("layers") with radial distance increments of L and place along each such arc a set of candidate merge points (e.g., uniformly distributed). These candidate merge points are the locations where internal nodes of the merge tree (i.e., merge points) may be placed. We connect two merge points on consecutive layers with an edge if there is a feasible (obstacle-avoiding) thick route between them. (For simplicity, we consider only straight routes in the current implementation.) The resulting graph is a layered DAG (with a particular drawing), with layers corresponding to the nodes on the circular arcs (layers).

By a variant of the hardness construction for parallel lane routing (Figure 3.1), it is NP-hard to decide if there exists a non-crossing tree that routes at least l of the source terminals $\{s_i\}$ to the sink terminal t in the drawing of the layered DAG (assuming no transitions between embedded edges can occur except at their endpoints).

In practice, it may be that the number of layers in the DAG is bounded; such is the case for our ATM application, since there is a small upper bound (approximately 5) on the number of merges points along any one arrival route, and, further, $(r_{\max} - r_{\min})/L$ is expected to be bounded. In this case, we give an exact algorithm for computing an arrival merge tree, based on dynamic programming. The algorithm's running time is polynomial in the input size, n (the number of candidate merge points and the number of vertices describing the domain Ω), for fixed number of layers (which appears in the exponent of n in the running time). Figure 3.2 shows an example of a merge tree computed with our software tool, the "Tree-Based Route Planner" (TBRP). The TBRP allows for various objectives in the optimization, including maximizing the number of source terminals linked to t and minimizing the lengths of the routes from sources to sink.

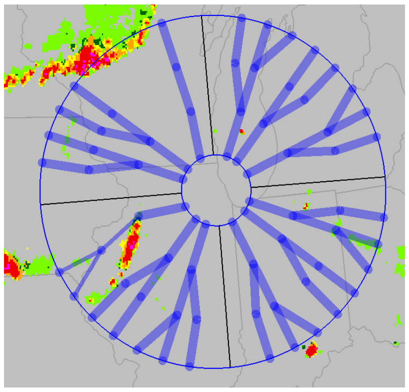


Figure 3.2: An example output of the implemented algorithm.

Acknowledgments

We thank Esther M. Arkin, Jimmy Krozel, and Girishkumar R. Sabhnani for helpful discussions. This work was partially supported by NSF (CCF-0528209, CCF-0729019), NASA Ames, and Metron Aviation.

Chapter 4

Approximating Maximum Flow in Polygonal Domains using Spanners

Abstract

We study a maximum flow problem in a polygonal domain P: Determine the maximum number of disjoint "thick" paths (of specified width w) through P from a source edge to a sink edge of P.

We show that Euclidean spanners offer a means of computing approximately optimal solutions. For a polygonal domain with n vertices and h point holes, we give a 1/2-approximation algorithm that runs in time $O(n + h\log(nh))$; this is to be contrasted with the known exact methods that take time $O(nh + n\log n)$. Further, we show experimentally that using a spanner (e.g., Delaunay graph) yields approximation ratios very close to one.

Keywords: Approximation; Delaunay diagram; Spanner; Critical graph; Air traffic management

This chapter is based on joint-work with Joseph S. B. Mitchell and Jingyu Zou of Stony Brook University. It was appeared in Proceedings of the 21st Canadian Conference on Computational Geometry(CCCG2009) [25]

4.1 Introduction

We consider the problem of routing multiple disjoint "thick" paths through a polygonal domain in the plane. The problem arises in various application domains, including VLSI wiring, robotics, sensor networks, and air traffic management (ATM). Our motivation comes from an ATM application in which the goal is to compute the "capacity" of an airspace: find the maximum number of disjoint "air lanes" avoiding hazardous weather and other "constraints" (obstacles) within an airspace of interest (e.g., a "flow-constrained area") [28, 31]. The goal is to provide computer-automated decision support tools to perform "capacity estimation" on an airspace to determine its maximum throughput, which measures how constrained the airspace is.

4.1.1 Problem Formulation

The input to our problem is a polygonal domain P, consisting of an outer polygon and a set H of h holes. Let n denote the total number of vertices of P. In this paper we focus on the special case in which H consists of a set of *point holes*; thus, the outer boundary of P is a simple polygon with $n - h$ vertices. This is the special case that arises in our ATM application, since the weather data is typically given as a set of points (pixels) at which there is hazardous weather predicted,

Two edges, Γ_s and Γ_t of ∂P are designated as the *source* and the *sink*. A *w-thick path*, or *lane of width w*, is the Minkowski sum of a usual ("thin") source-to-sink path and a disk of radius $w/2$ centered at the origin. (Refer to [37] for more definitions and background.) We consider the parameter w to be fixed and refer to a w-thick path simply as a *thick path*. Our goal is to compute (approximately) the maximum number of pairwise-disjoint thick paths within P from Γ_s to Γ_t.

4.1.2 Related Work

Algorithms for computing maximum flows and minimum cuts in the continuum (2D geometric domains) were first studied in [36]. Recent results have examined the problems of minimizing path lengths for multiple thick paths (minimum-cost flow) [37] and routing thick paths in dynamic environments (e.g., moving weather systems) [2]. See also [42]. The application of max-flow techniques in ATM is addressed, e.g., in [28, 31].

Related to our experiment where we study the stretch factor of Delaunay graph under the "rounded" Euclidean metric, [16, Section 6] studied the average and maximum stretch factor, among other spanner properties, of several well-known geometric spanners for random point sets.

4.1.3 Summary of Results

Using the propagation algorithm of [36], appropriately modified to handle discrete thick paths (versus continuous flow fields), our optimization problem can be solved exactly in time $O(nh+n\log n)$ (see [2, Thm. 2.1]), for a polygonal domain with h polygonal holes, having a total of n vertices. In this paper we propose a simple 1/2-approximation algorithm for the case that P is a polygonal domain with point holes. Our algorithm searches a Euclidean spanner graph for an approximate min-cut, in time $O(n + h\log(nh))$. We show that this results in an approximation for the problem of maximizing the number of disjoint thick paths. We also conduct experiments, using the Delaunay graph as spanner, to validate the effectiveness of the approximation in practice on both randomly generated data and actual weather data.

4.2 An Approximation Bound

Let B (resp., T) be that portion of ∂P counterclockwise between Γ_s and Γ_t (resp., between Γ_t and Γ_s). We define $G = (V, E)$ where $V = H \cup \{T\} \cup \{B\}$, $E = \{(i, j)|i, j \in V, i \neq j\}$. The weight of edge (i, j) is $c(i, j) = \lfloor d(i, j)/w \rfloor$, where $d(i, j)$ is the (minimum) Euclidean distance between i and j, and w is the path width (thickness). This graph G is called the *thresholded critical graph.* Let G_H be the subgraph of G induced by nodes H. Finally, we let G_H^ε denote the subgraph of G_H whose edge set consists of the union of the set of Delaunay edges of H and the edges of a Euclidean $(1+\varepsilon)$-spanner of H. (Edge weights remain as in G.)

Proposition 4.2.1. *The function $f(x) = \frac{\lfloor(1+\varepsilon)x\rfloor}{\lfloor x\rfloor}$, $x > 0$ is bounded above by 2 when $\varepsilon \leq 0.5$.*

Proof. Note that $1+\varepsilon \le 3/2$ and that $f(x)$ is a step function whose value changes only when $x = n$ or $x = \frac{n}{1+\varepsilon}$, for $n \in \mathbb{Z}^+$.

When $x = n$, $n \in \mathbb{Z}^+$, $f(x) \le \frac{(1+\varepsilon)n}{n} = 1+\varepsilon \le 3/2$.

When $x = \frac{n}{1+\varepsilon}$, $n \in \mathbb{Z}^+$, $f(x) = \frac{\lfloor n \rfloor}{\lfloor n/(1+\varepsilon) \rfloor} \le \frac{\lfloor n \rfloor}{\lfloor 2n/3 \rfloor}$. Let $g(n) = \frac{\lfloor n \rfloor}{\lfloor 2n/3 \rfloor}$. If $n = 3k$, $g(n) = \frac{3}{2}$. If $n = 3k-1$, $g(n) = \frac{3k-1}{2k-1}$ and it achieves its maximum of 2 at $k = 1$. If $n = 3k+1$, $g(n) = \frac{3k+1}{2k}$ which also achieves its maximum of 2 at $k = 1$. Thus $f(x) \le 2$ in this case. $\square$

Lemma 4.2.2. *For $\varepsilon \le 0.5$, G_H^ε is a 2-spanner for G_H.*

Proof. It suffices to show that for any edge (u, v) in G_H, there is a path from u to v in G_H^ε of length at most $2 \cdot c(u, v)$. There are two cases to consider,

Case 1: $c(u, v) = \lfloor d(u, v)/w \rfloor = 0$. Since G_H^ε includes Delaunay edges, we know from [6] and [14] that there exists a path π_1 in G_H^ε with vertices $u = v_1, v_2, \ldots, v_k = v$ such that $\forall i, d(v_i, v_{i+1}) \le d(u, v)$. Since $c(u, v) = 0$ implies that $d(u, v) < w$, we have that $\forall i, c(v_i, v_{i+1}) = 0$, so the path π_1 has zero length in G_H^ε and the claimed stretch factor automatically holds.

Case 2: $c(u, v) > 0$. Since G_H^ε contains an Euclidean $(1+\varepsilon)$-spanner, there exists a path π_2 in G_H^ε such that $\sum_{i=1}^{k-1} d(v_i, v_{i+1}) \le (1+\varepsilon)d(u, v)$. Thus, the stretch factor of G_H^ε is at most

$$\begin{aligned}
\frac{\sum_{i=1}^{k-1} c(v_i, v_{i+1})}{c(u, v)} &= \frac{\sum_{i=1}^{k-1} \lfloor d(v_i, v_{i+1})/w \rfloor}{\lfloor d(u, v)/w \rfloor} \\
&\le \frac{\lfloor \sum_{i=1}^{k-1} d(v_i, v_{i+1})/w \rfloor}{\lfloor d(u, v)/w \rfloor} \\
&\le \frac{\lfloor (1+\varepsilon) d(u, v)/w \rfloor}{\lfloor d(u, v)/w \rfloor}.
\end{aligned}$$

In Proposition 4.2.1, we show that if $f(x) = \frac{\lfloor (1+\varepsilon)x \rfloor}{\lfloor x \rfloor}$ and $\varepsilon \le 0.5$, then $f(x) \le 2$, for $x > 0$. $\square$

Theorem 4.2.3. *The maximum number of disjoint w-thick paths in a polygonal domain with n vertices and h point holes can be $\frac{1}{2}$-approximated in time $O(n + h \log(nh))$*

Proof. G^ε_H is constructed in time $O(h \log h)$, the time needed to build the Delaunay graph or a $(1+\varepsilon)$-spanner (with $\varepsilon = 0.5$ and $O(h)$ edges) for h points. We then construct a graph G^ε from G^ε_H by adding nodes for B and T, and linking these nodes to each point of H. We compute the distance from each point of H to the polygonal chains B and T in time $O(\log n)$ per point of H (after spending time $O(n)$ to construct the Voronoi diagrams, and a corresponding point location data structure, of the simple chains B, T [9]). This augmented graph G^ε has $O(h)$ edges and is a 2-spanner for the critical graph G.

By the continuous max-flow min-cut theorem in [2, 36, 56], we know that the maximum number, OPT, of thick paths from source to sink is equal to the length, $|\pi_G|$, of a shortest path from B to T in G. Since G^ε is a 2-spanner for G, we know that the length, $|\pi_{G^\varepsilon}|$, of a shortest B-to-T path in G^ε is at most $2|\pi_G|$: $OPT \le |\pi_{G^\varepsilon}| \le 2 \cdot OPT$, i.e., $(1/2) \cdot OPT \le (1/2)|\pi_{G^\varepsilon}| \le OPT$. Thus, $(1/2)|\pi_{G^\varepsilon}|$ is a $\frac{1}{2}$-approximation to the maximum number of source-to-sink thick paths.

Our algorithm takes time $O(h \log h)$ to build G^ε_H, $O(n + h \log n)$ to build G^ε from G^ε_H, and another $O(h \log h)$ to search for a shortest path in G^ε. Altogether, the time bound is $O(n + h \log(nh))$. $\square$

4.3 Experiments

We did experiments based on computing a specific spanner – the Delaunay graph of the points H. The Delaunay graph is a Euclidean spanner, with stretch factor known to be between 1.581 ($> \pi/2$ [5]) and $\frac{4\pi}{3\sqrt{3}} \le 2.42$ [22]. Theorem 4.2.3 tells us that if we use a spanner with stretch factor at most 1.5 ($\varepsilon = 0.5$), then our approach gives a $\frac{1}{2}$-approximation. Since the Delaunay graph does not have the required property, we do not have a theoretical guarantee that the Delaunay-based results give a $\frac{1}{2}$-approximation; however, we will see that, in practice, the Delaunay performs very well. In the experiments here, we report only our experience with the Delaunay spanner; further experiments are underway with other spanners.

We use a unit square box as the outer boundary of the polygonal domain P, and use two types of input data for the point holes H: (1) uniformly generated points, and (2) real weather data, scaled to the unit square. For both sets of input data, we examine the relationship between the stretch factor (ratio

$|\pi_{G^\epsilon}|/|\pi_G|$) as a function of the "average density" of the point set H. We define the *average density* of the points H to be the average edge length in the nearest neighbor graph of H; the smaller this average edge length is, the denser the point set is.

In the experiments with random point sets, we vary the number, h, of points and the lane width, w.

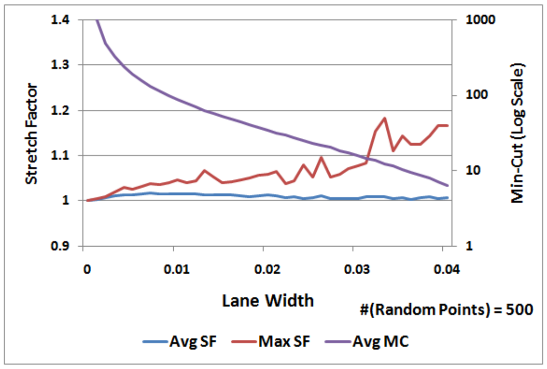


Figure 4.1: Average and maximum stretch factor (SF) and length of min-cut (MC) as a function of lane width w for random points.

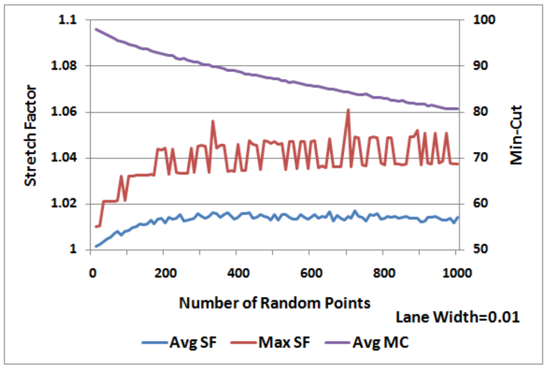


Figure 4.2: S
tretchAverage and maximum stretch factor (SF) and length of min-cut (MC) as a function of the number of random points.

For the experiments with varying lane width w, we fix the number, $h = 500$, of random points and vary w from 0 to 0.4. For each w, we generated 100 random instances and compiled simple statistics – the average and the maximum stretch factor. Figure 4.1 shows that the stretch factor is usually very close to

1. Even the maximum stretch factor (over all 100 instances) is low when the min-cut length is large. For example, if the min-cut length is greater than 20, then the maximum stretch factor is always less than 1.1. This means that the difference between exact min-cut length and our approximation is less than 2 even in the worst instance.

For the experiments with varying number h of points, we fix the width, $w = 0.01$, and vary h from 0 to 1000 in increments of 10. For each h, we generated 100 random instances and recorded the average and maximum stretch factor. Figure 4.2 shows that stretch factor is close to 1 in most cases. The stretch factor is close to 1 even when w is relatively small (i.e., the min-cut is large).

	h	min-cut	avg dist	stretch factor
Set1	576	179	0.0039	1.0056
Set2	502	182	0.0039	1.0110
Set3	430	184	0.0048	1.0163
Set4	752	177	0.0038	1.0113
Set5	820	180	0.0028	1.0000

Table 4.1: Results for real weather data with $w = 0.005$. Here h is the number of point obstacles and “avg dist” is the average nearest neighbor edge length.

Table 4.1 shows the stretch factor data for real weather data. Since real weather tends to have clusters of weather points (pixels), the average nearest neighbor distance is much smaller than for random point sets; accordingly, we set the lane width to be very small, $w = 0.005$. (For $w = 0.05$, we found that the stretch factor is always 1.) The results show that the stretch factor is very close to 1, even if the average nearest neighbor distance is comparable to w.

4.4 Conclusion

Our goal has been to explore the use of spanners in computing approximations for maximizing the number of pairwise disjoint thick paths that can be routed through a polygonal domain in the plane. The advantage of using a spanner (e.g., Delaunay graph) for computing minimum cut values approximately is that it gives a linear space and near-linear time simple algorithm in place of the far more complex exact $O(nh + n \log n))$ algorithm, or the naive $O(n^2)$ algorithm that is easiest to implement. We have seen experimentally that the Delaunay graph does very well in most cases; thus, the Delaunay-based approximation is likely an effective and practical means of doing capacity estimation for ATM.

The exact min-cut problem is a shortest path problem in the plane that may be solvable in $O(n \log n)$ exactly, e.g., by a variant of the continuous Dijkstra paradigm that solves obstacle-avoiding shortest paths in $O(n \log n)$ time. Also, we are examining possible improved approximations possible using spanner techniques, and we are now developing algorithms to produce a set of disjoint thick paths that achieve the capacity determined by our approximation algorithms. Finally, we are doing further experimentation with other Euclidean spanner graphs, with stretch factors approaching 1.

Acknowledgments

The data used for the experiments was provided by Metron Aviation. We thank Valentin Polishchuk and Girishkumar Sabhnani for helpful discussions. This work was partially supported by NSF (CCF-0528209, CCF-0729019), NASA Ames, and Metron Aviation.

Chapter 5

Sensitivity of Capacity Estimation Results subject to Convective Weather Forecast Errors

Nomenclature

ATM	Air Traffic Management
CWAM	Convective Weather Avoidance Model
DST	Decision Support Tool
NAS	National Airspace System
NextGen	Next Generation Air Transportation System
nmi	nautical miles
NWS	National Weather System
pdf	probability distribution function
RNP	Required Navigation Performance
TFM	Traffic Flow Management
VIL	Vertically Integrated Liquid

This chapter is based on joint-work with Joseph S. B. Mitchell and Jingyu Zou of Stony Brook University and Jimmy Krozel of Metron Aviation Inc. It was appeared in AIAA Guidance, Navigation, and Control Conference 2009 (GNC2009). [23]

5.1 Introduction

In the design of key components of the Next Generation Air Transportation System (NextGen) [57, 55, 40, 41], there is an emphasis on weather assimilated into the Decision Support Tools (DSTs). These NextGen DSTs will automate many aspects of the control of the National Airspace System (NAS). Since Air Traffic Management (ATM) decisions must be made ahead of time, the integration of weather into DSTs must take on the form of reasoning about weather forecasts and their inherent uncertainties; furthermore, one must transform the weather forecast into resulting ATM impacts in order to understand the relationship between weather forecasts, their uncertainties, and the robustness of ATM decisions to such uncertainties. A critical component of ATM impact assessment is to estimate the capacity of an airspace subject to hazardous weather constraints ahead of time, in order to set up Traffic Flow Management (TFM) plans for scheduling and routing traffic demand up to but not exceeding that capacity estimate.

In this paper, we investigate the estimation of capacity for en route airspace and transition airspace. For en route airspace, we consider level flight. For the transition airspace, we consider a quadrant of airspace within the annulus from roughly 200 nmi away from the airport, where most aircraft are en route and beginning their top of descent, to approximately 40 nmi from the airport, where traffic transitions into the terminal airspace. Our goal is to understand how sensitive the capacity estimate is to weather forecast uncertainties. This has a significant impact on requirements for weather forecast update rates (in order to monitor the uncertainties and their impacts), and how DSTs should transition between strategic plans (where uncertainties may require probabilistic reasoning) and tactical plans (where uncertainties are smaller and a more deterministic view of the ATM situation can be established).

5.2 Related Work

In previous work [29, 31], we have described many aspects of the literature related to capacity estimation. We refer the reader to those papers for a more detailed review of related work. We mention here briefly a few references that describe recent trends in this research area. Song et al. [52, 53, 51] study the problem of predicting sector capacity for sectors in today's NAS through a pattern recognition technique recognizing the traffic flow pattern is included in their technique. Song et al. [54, 51] have also performed a comparison of three techniques for capacity estimation that compares and contrasts their performance, including the performance of the min-cut technique that we use in this paper. We note that [59] studies a related problem of characterizing the uncertainty in demand estimates for TFM applications.

Our previous papers [38, 29] establish a theoretical analysis of the airspace capacity as a function of hazardous weather constraints, independent of workload considerations and independent of today's jet routes. The analysis is based on maximum flow concepts in geometric domains, based on the min-cut theory given by [36, 43, 56]. Our work is in support of the design of new roles for controllers and pilots in the NextGen, and addresses the maximum throughput of an airspace, assuming that workload is not a constraint.

In order to set up the capacity estimation problem, researchers must define aviation weather hazards for the airspace being analyzed. For the transition airspace, this typically includes convection, turbulence, and icing hazards (see for instance, the surveys [30, 27]). For convective weather constraints (the hazard with greatest aviation impact), researchers have been studying pilot behavior during convective weather events to build pilot deviation models [48, 47, 34]. Some researchers threshold the weather severity at National Weather Service (NWS) Level 3 or above to define the weather hazard [48, 47]; however, they note that pilots are more likely to penetrate NWS Level 3 or above as they get closer to the metering fixes of an airport and there is pressure to land the aircraft. Others invoke the Convective Weather Avoidance Model (CWAM) [8], which considers both the reflectivity (i.e. intensity) of the weather and the aircraft clearance above the echo tops of a severe weather cell. Other papers focus on "route availability" given weather hazards that may interfere with traffic on selected jet routes [13, 34].

5.3 Approach to the Sensitivity Analysis

Next, we discuss capacity estimation algorithms that allow us to estimate the capacity of an airspace. Then we introduce our method of generating a set of weather forecast errors relative to a nominal "truth" weather forecast. Given these erroneous weather forecasts, we use our capacity estimation technique to transform each erroneous weather forecast in the sensitivity study into an ATM impact. Finally, we characterize the ATM impacts in terms of probability distribution functions (pdfs).

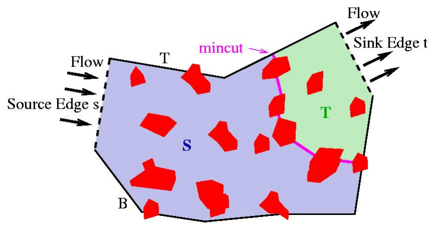
5.3.1 Capacity Estimation Technique

Our solutions are based on the assumption that permeability of an airspace that is, the number of well-separated routes that can be safely flown through that airspace is a reasonable estimator of (e.g., proportional to) capacity. This is not strictly true for current operations since the limiting factor is often based on controller workload the ability of an air traffic controller to monitor and control multiple flights. However, in NextGen, we envision this factor becoming less important as computer-assisted ATM DSTs become prevalent. Therefore, it is important to investigate capacity values for an airspace in terms of their permeability, which can become the limiting factor in hazardous-weather situations.

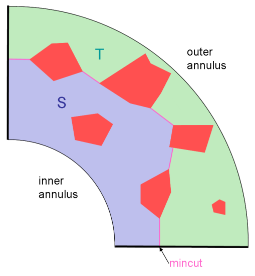
The theory of capacity estimation for static deterministic constraints is well established [36, 38, 29]. By applying the continuous max-flow/min-cut theory to a two-dimensional (2D) domain, we can determine an upper bound on the airspace permeability for a given flight level or on a two-dimensional manifold (e.g., a surface determined by a descent profile for flights arriving at an air portal). While capacity is a function of more than simply permeability, the permeability of an airspace is a strong determining component of capacity, since the permeability is a direct measure of the number of air lanes that can be flown as well as an indirect measure of the likelihood of any given air lane being impacted by hazardous weather. While real weather is neither static nor deterministic, the static-deterministic case for weather constraints represents an important basis in the development of solutions to more general capacity estimation problems. By integrating capacity estimates across the set of 2D manifolds that could reasonably be used for arrivals and departures to a metroplex, we can develop an estimate for capacity of that metroplex as a function of weather independent of runway capacity.

A max-flow/min-cut theory for capacity estimation gives a provable metric for throughput analysis, giving upper bounds on the capacity for an airspace, as a function of time. For long-term planning, we are able to provide probability distributions of the capacity for some portion (e.g., a quadrant) of a metroplex, or the capacity of any specified en route airspace (e.g., a rectangular flow-constrained area).

We briefly review the max-flow/min-cut theory for polygonal domains, as



(a) Continuous polygon domain



(b) Continuous annulus domain

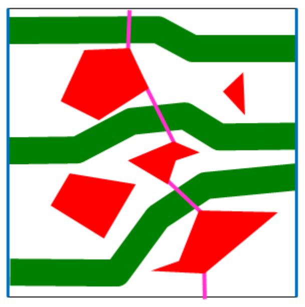
Figure 5.1: Theoretical capacity of a continuous flow field is determined by the *min-cut*

originally studied in [56, 36]. We consider a simple polygon P (Figure 5.1). Two boundary edges, s and t, of the polygon are designated as the source and the sink. A *flow* f in P is a vector field. The constraints $H_1, \ldots, H_k$ are pairwise-disjoint simple hazard polygons that lie fully inside P; the flow is not allowed to pass through any of the constraints, i.e., for any point x within a constraint, $f(x) = 0$. The polygon P is assumed to be uniformly capacitated, i.e., the length of the flow vector must nowhere exceed 1. The value of the flow is defined as an integral of the normal component, $f - n$, over t. There are no sources or sinks inside P; i.e., for any x in P, div $f(x) = 0$.

The max-flow problem is to find an s-t flow of maximum value. A *cut* in P is a partitioning of the polygon into two parts (shaded blue and green in Figure 5.1) so that s is in one of the parts, and t is in the other. The capacity of a cut is the length of the boundary (a magenta line in Figure 5.1) between the parts, where only the part of the boundary that is interior to P (and not on the boundary of P or within a constraint) is included in the length. We use the term *min-cut* to refer to the path(s) within P that comprise the boundary of the cut (magenta in Figure 5.1 and Figure 5.2), as well as to refer to the capacity (length) of the min-cut.

In capacity estimation, we are concerned with computing a min-cut in a polygonal domain, since it represents the maximum theoretical capacity of an airspace with respect to a given set of constraints [29, 31]. The source and sink edges, s and t, split the boundary of P into two polygonal chains, which we denote by B and T (Figure 5.1). If s and t represent the west and east boundaries, then B and T are the bottom and top. The min-cut is computed using the *critical graph*, which we describe in detail in previous publications [29, 31].

As shown in Figure 5.2, the min-cut determines the *bottleneck* for TFM: the length of the min-cut determines the number of air lanes of constant Required Navigation Performance (RNP) that can be routed from source to sink across an en route airspace (Figure 5.2(a)) or a transition airspace (Figure 5.2(b)). The max-flow/min-cut theory applies to general shapes of airspace and choices of B and T on the boundary of a region of interest. For en route airspace, we considered a 100nmi-by-100nmi square-shaped region, at two different orientations, each with two different choices of source/sink, resulting in flows at headings $0\,^\circ$ (S to N), $45\,^\circ$ (SE to NW), $90\,^\circ$ (E to W), $135\,^\circ$ (NE to SW); see Figure 5.3(a). For transition airspace, our study considered a 90-degree quadrant-shaped transition airspace, within an annulus between 40 and 200 nmi from the center of a metroplex, with source arc at 200 nmi and sink arc at 40 nmi, and orientations of flows at $0\,^\circ, 45\,^\circ, 90\,^\circ, 135\,^\circ, 180\,^\circ, 225\,^\circ, 270\,^\circ, 315\,^\circ$, corresponding to arrival flows from the NE, N, NW, W, SW, S, SE, E, respectively.


(a) En route Airspace

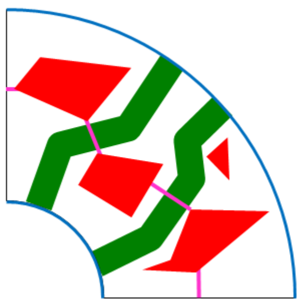
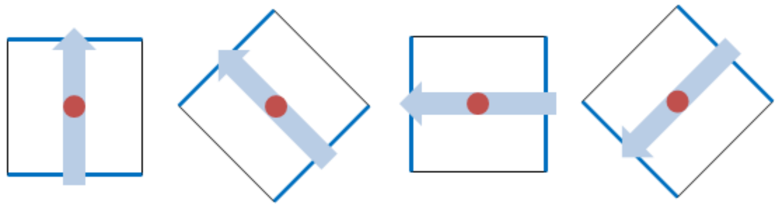

(b) Transition Airspace

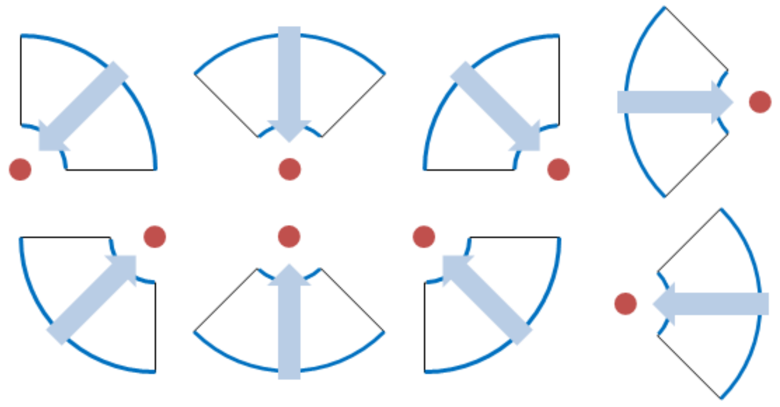
Figure 5.2: The relationship between the Min-Cut and the number of air lanes that can permeate an airspace

5.3.2 Dynamic Weather

Weather forecasts give a sequence of weather constraints over a time horizon. For each time slice of the forecast, we could apply our capacity estimation algorithm for static weather, thereby obtaining capacity estimates for each



(a) En route Airspace: 100 × 100 nmi square, with four flow directions.



(b) Transition Airspace, at various orientations, within an annulus (from 40 to 200 nmi from center), with eight flow directions for arriving traffic.

Figure 5.3: Airspace regions at various orientations for different directions of flow

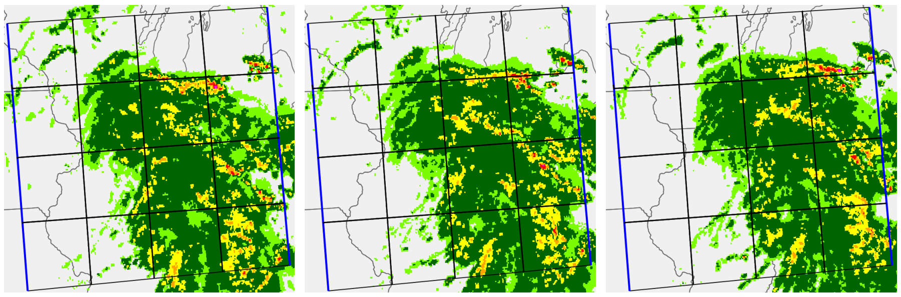
static time slice. However, the capacity estimation problem in the presence of dynamic constraints is a harder problem, which cannot be solved simply by applying the static analysis on a slice by slice basis. This is discussed in further detail in our previous capacity estimation work [31]. See also [44, 46] for related algorithms for finding routes through dynamic weather environments; these algorithms comprise the Flow-Based Route Planner, which can also be used for capacity estimation of a weather-impacted airspace with time-varying weather. Results of a more theoretical nature appear in the recent work of Arkin et. al [2].

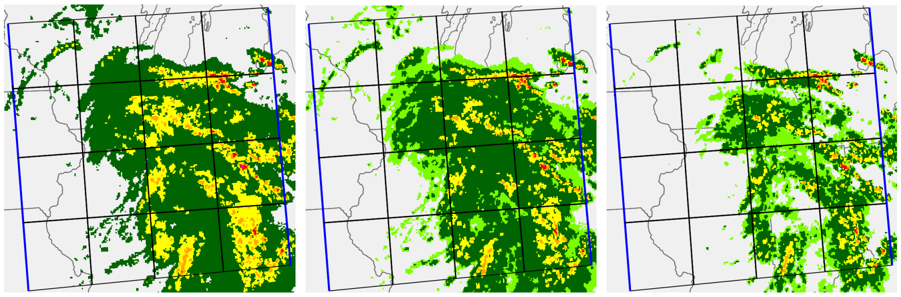
5.3.3 Generation of Weather Errors

The errors inherent in convective-weather forecasts can be measured and quantified on several dimensions, including the error in time at which the weather arrives, coverage (intensity), and the error in its position. Shown in Figure 5.4 is an example of a 400 x 400 nmi region, with the convective weather map shown for variations in time (a), shifts in intensity (b), and variations in position/translation (c). In our experiments, we generate synthetic ensembles of forecasts by varying the time, intensity threshold, and position of a given nowcast (actual observed weather). (Similarly, our same method applies to generate ensembles of forecasts from a single deterministic forecast.)

In particular, we generated "Δ time" ensemble forecasts for time t by using the (single) nowcast at time $t + \Delta t$, for a random variable Δt, modeled with a (truncated) Gaussian distribution. We generated "Δ threshold" ensemble forecasts for time t by using the (single) nowcast at time t but setting the intensity threshold for hazardous weather at level W, a random variable, modeled with a (truncated) Gaussian distribution. Different values of W yield different levels of coverage of the hazardous weather constraints. Finally, we generated "Δ translation" ensemble forecasts for time t by using the (single) nowcast at time t but shifting (translating) the nowcast map by longitude and latitude shifts modeled as a (truncated) Gaussian distribution. See Table 5.1 for parameters used in our experiments.

degree differences of latitude (resp., longitude) around Chicago area correspond to about 60 (resp., 32) nautical miles.


(a) Time errors of -30 minutes, 0 minute, and +30 minutes


(b) Threshold errors, with VIL thresholds set as 2, 3 and 4

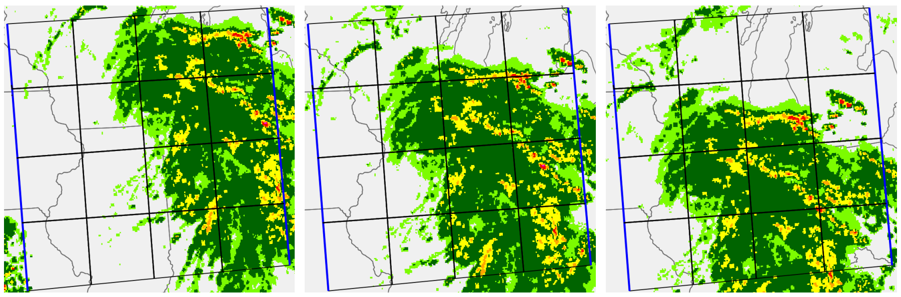

(c) Translation errors of -1.0 deg in both latitude and longitude, no error, and +1.0 deg in both latitude and longitude

Figure 5.4: Weather forecast errors characterized in terms of timing, threshold/intensity, and translational errors. The nominal weather (shown in the middle column) is at 04/30/2009 14:30:00 Z.

Error parameter:	Δ time	Δ threshold	Δ translation
Granularity	5 minutes	0.2	0.2 degrees
Maximum	120 minutes	3.0	2.0 degrees
Std dev	60 minutes	1.5	1.0 degrees

Table 5.1: Error parameters used in experiments

5.3.4 Experiment Setup

We select several days of weather data to represent different weather organizations, each having active convective weather around the Chicago area. For each day, we select a specific time moment to be examined; the days and the particular times were selected based on there being highly active and significant convective weather at the moment. Figure 5.5 are the snapshots of weather data at each time. We use 41.98 degree latitude and -87.91 degree longitude as the center (Chicago O'Hare Airport (ORD)) and VIL value 3 (for comparison, NWS level 3 is VIL value 3.47) as the threshold criterion for our experiments: regions have VIL value above 3 are considered to be constraints.

For each sample point, we perform experiments on two types of regions for capacity estimation. The first is a 100nmi x 100nmi square-shaped region centered on ORD for evaluating the capacity of en route airspace, and the second is a quadrant-shaped region for evaluating the capacity for one quadrant of transition airspace for arrivals leading to the Chicago metroplex area. The quadrant of airspace has 40 miles inner annulus and 200 miles outer annulus and is 90 degrees in angular extent. See Figure 5.2 for these general shapes. Since the capacity depends on the direction of the flow, we use several orientation angles for each choice of airspace. Angles 0, 45, 90, and 135 degrees are used for the square en route airspace (note: symmetric angles have the same values), and angles 0, 45, 90, 135, 180, 225, 270, and 315 degrees are used for the quadrant transition airspace. Refer to Figure 5.3. Table 5.2 shows the capacity values based on actual weather. Capacity values we get from our max-flow/min-cut method are integral, since they are the maximum possible number of lanes.

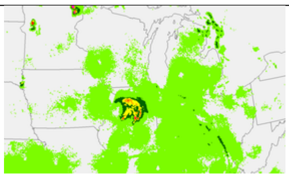
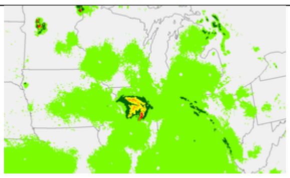
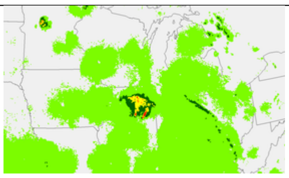
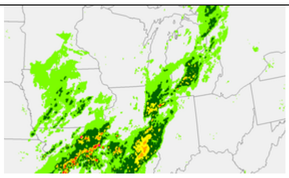
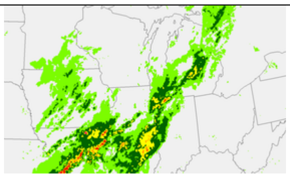
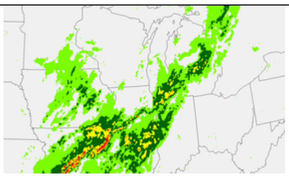
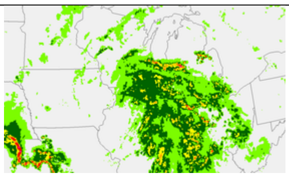
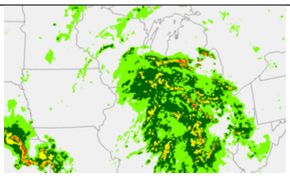
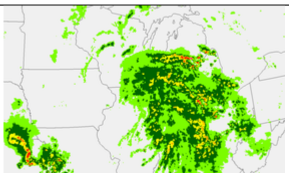
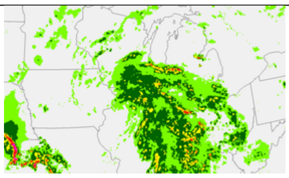
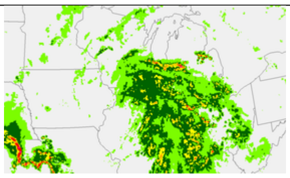
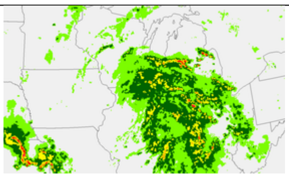
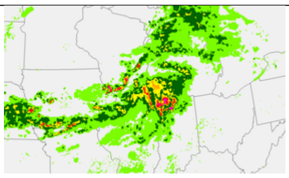
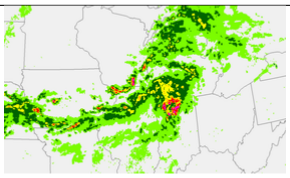
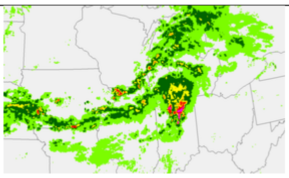
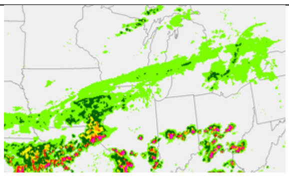
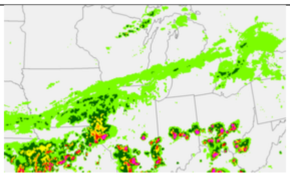
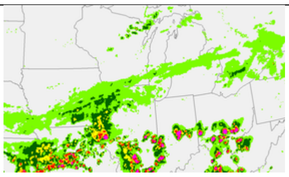
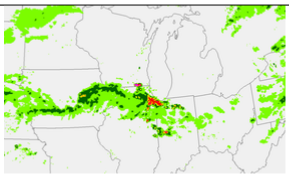
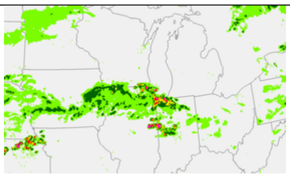
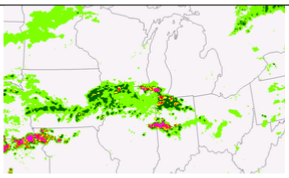
Data/Time (Z)	Snapshot		
	at -30 minutes	at Time	at +30 minutes
4/24/2009 03:00			
4/27/2009 21:30			
4/30/2009 14:30			
4/30/2009 14:00			
6/1/2009 10:30			
6/2/2009 21:00			
6/1/2009 20:35			

Figure 5.5: Weather Snapshots of Sample Data

Date/Time(Z)	Tag on plots	Average capacity and capacities at the 4 or 8 different orientations	
		En route	Transition
4/24/2009 03:00	4/24	7.00 [7, 4, 7, 10]	6.375 [8, 8, 8, 4, 1, 6, 8, 8]
4/27/2009 21:30	4/27	6.75 [7, 8, 7, 5]	6.375 [8, 8, 8, 8, 8, 5, 1, 5]
4/30/2009 14:30	4/30_1	4.50 [4, 5, 5, 4]	4.500 [0, 3, 7, 8, 7, 4, 5, 2]
4/30/2009 14:00	4/30_2	4.75 [5, 4, 5, 5]	5.250 [1, 5, 7, 5, 6, 7, 7, 4]
6/1/2009 10:30	6/1_1	0.50 [0, 2, 0, 0]	4.375 [2, 6, 6, 5, 5, 7, 4, 0]
6/2/2009 21:00	6/2	12.5 [13, 12, 13, 12]	7.875 [8, 8, 8, 8, 7, 8, 8, 8]
6/1/2009 20:35	6/1_2	1.75 [1, 0, 1, 5]	6.375 [8, 6, 6, 8, 8, 7, 4, 4]

Table 5.2: Capacity of Sample Weather Data

5.3.5 Sensitivity of Capacity Estimates to Weather Forecast Errors

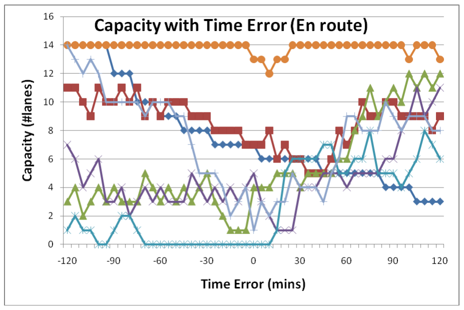
We begin our study of sensitivity to weather forecast errors with a set of results depicted in Figure 5.6, showing how capacity varies with changes in the time, the VIL intensity threshold, and the translation. In each plot of Figure 5.6, we show seven curves, one corresponding to each of the date/time samples used in our study. The vertical axis shows computed capacity (in number of air lanes).

In (a) and (b), we plot the capacity as a function of time shift, for en route (a) and transition (b) airspace, using capacities computed from the actual (observed) weather, going forward and backward in time by 120 minutes, in 5-minute increments. Time "0" refers to the exact time stamp for the sample (e.g., 14:00 on the 4/30_2 sample). As we see, in most cases, the capacity is somewhat diminished at or around the 0 time stamp; however, there is a fair amount of fluctuation on both sides of time 0. For the 6/1_1 data set, we see that the capacity is at or near zero for much of the 90 minutes prior to time 0 (10:30 on 6/1/2009), in both the en route case and the transition airspace case; this is a result of the impact of the severe convective weather. For the 6/2 data set, we see that en route capacity is slightly diminished right around time 0, but is then very high (maximum possible) shortly before time 0, and again from 30 minutes after time 0 onwards. For the same data set, the transition airspace capacity is at a constant maximum possible (8 air lanes) for the entire 4 hour time window.

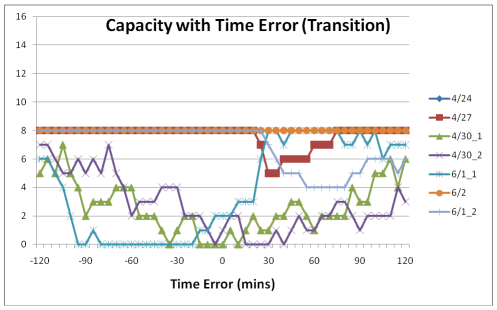
In Figure 5.6 (c) and (d), we plot the capacity as a function of VIL intensity threshold, varied over the range (0,6), centered on the nominal value of VIL 3.0 (which corresponds to threshold error of 0). As expected, these plots all show non-decreasing functions: the higher the threshold, the smaller the coverage region, and the more the capacity. Some data sets (e.g., 4/30_2) show high sensitivity (slope) around error 0, while others (e.g., 6/2) show little sensitivity to threshold changes/errors. (Of course, with threshold error -3, meaning a VIL threshold of 0, there is no capacity, since all points are considered to be weather-impacted constraints.)

In Figure 5.6 (e) and (f), we plot the capacity as a function of translational error (in degrees), for shifts in latitude of the airspace (i.e., an error of -1.0 degree corresponds to translating the airspace to the south, with respect to the weather, or, equivalently, to shifting the weather to the north with respect to a fixed airspace around ORD). For this experiment, we shifted the actual observed weather by latitude, from -2.0 to +2.0 degrees, in increments of 0.2 degrees. Naturally, as the weather map is shifted, there are changes in the computed capacity. There is considerable variation, though, over the different data sets, as the weather systems have different size and structure, and interact differently with the boundaries of the airspace of interest. For example, note that with the 6/2 data set, a translation of -2.0 degrees causes the capacity to drop substantially from its maximum in the en route case. This is because the significant weather for 6/2 occurred to the south of the en route airspace; refer to Figure 5.5, where the active weather cells fall enough south of ORD that they mostly miss the en route airspace (a 100nmi-by-100nmi squared centered on ORD). If there is significant error in the position of the weather with respect to the airspace, this translates to a significant error in the capacity estimation.

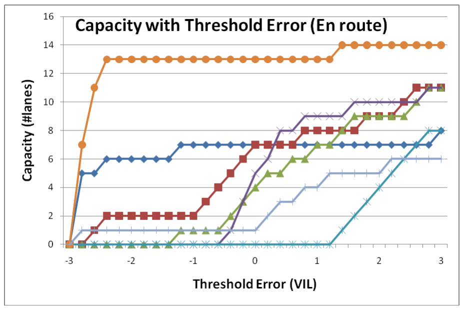
In Figure 5.7, we examine variation also over the orientation of the airspace. Here, we examine one sample, 4/30_1 (04/30/2009 14:30:00 Z), and we plot the capacity as a function of time, VIL threshold, and translation, for each choice of orientation, varying the en route and transition airspace as in Figure 5.3. Note that there can be significant differences in the plots depending on the orientation of the flow with respect to the weather. For example, Figure 5.7 (b) shows a distinctively different (lower) plot of the capacity as a function of time for the orientation at 0 degrees, compared with, e.g., 90 degrees. Also, Figure 5.7(d) shows a very different plot of capacity as a function of VIL threshold for orientation 0 degrees compared with, e.g., 180 degrees. These results are understandable in light of the fact that the weather impacted region for 4/30_1 lies mostly in the region just east of ORD and is mostly clear just west of ORD; refer to Figure 5.5.



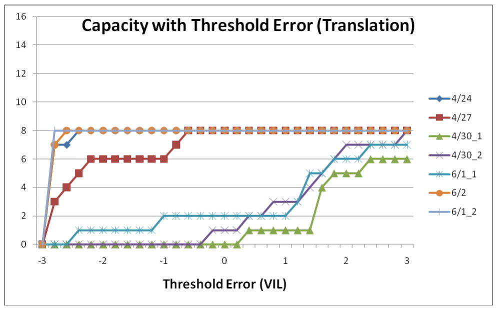
(a) En route with Time Error



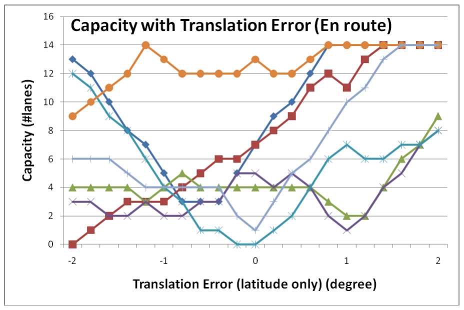
(b) Transition with Time Error



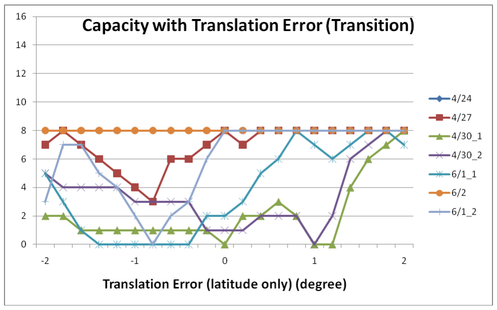
(c) En route with Threshold Error



(d) Transition with Threshold Error

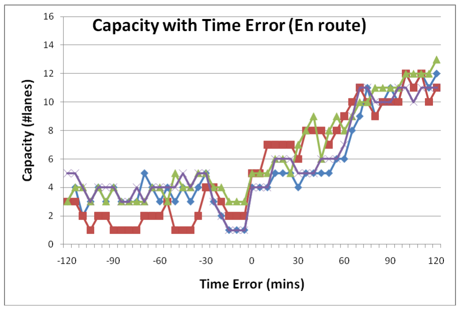


(e) En route with Translation Error

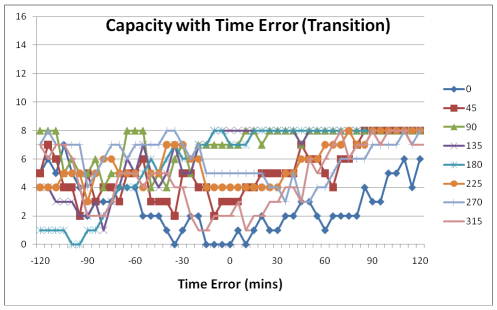


(f) Transition with Translation Error

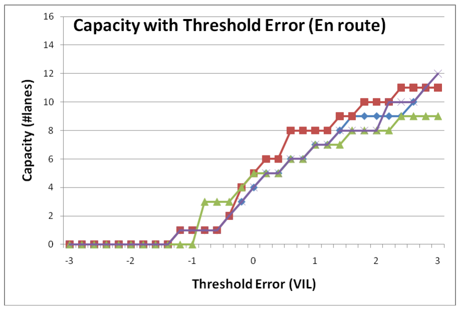
Figure 5.6: Capacity with Time, Threshold and Translation Errors. Fixed orientation (angle 0) is used for all sample points



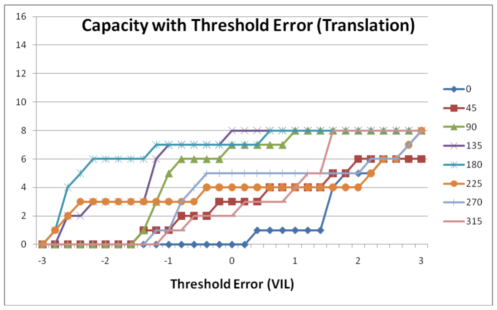
(a) En route with Time Error



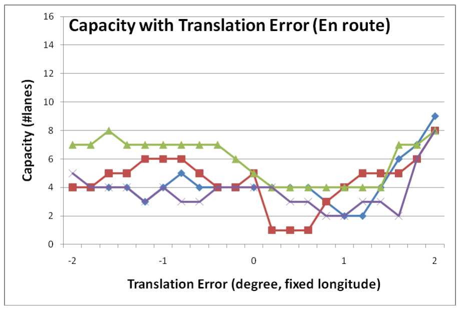
(b) Transition with Time Error



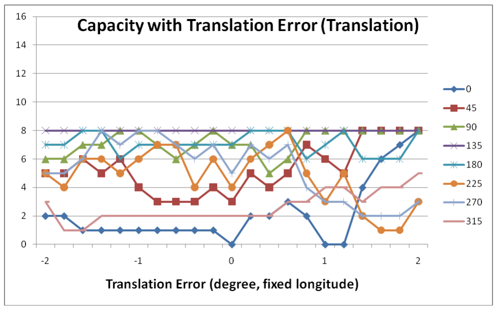
(c) En route with Threshold Error



(d) Transition with Threshold Error



(e) En route with Translation Error



(f) Transition with Translation Error

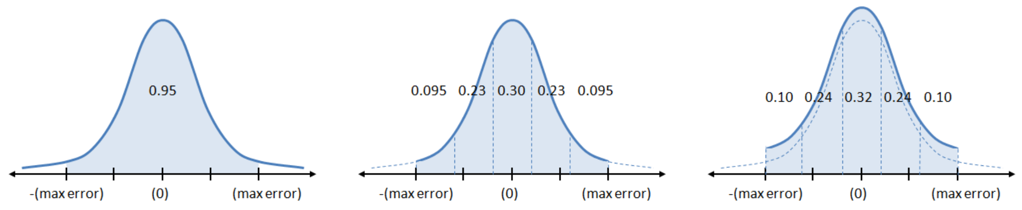
Figure 5.7: Capacity as a function of orientation. Here, the VIL threshold is 3.0, the location is ORD and the date/time is 04/30/2009 14:30:00 Z

5.3.6 Probabilistic Model of Capacity

We use a simple model, based on randomizing the time, threshold, or translation, to synthesize an ensemble forecast from a given weather sample, which can either be a nowcast or a (deterministic) forecast. An ensemble forecast serves as a probabilistic model of uncertain weather: associated with each member of the ensemble is a probability that the member is "selected" as the "true" weather. In the most basic model, each element of the ensemble of k forecasts is equally likely, having probability 1/k. For each member, i, of the ensemble, we run our capacity estimation algorithm, obtaining a capacity estimate, Ci. for forecast i. Then, we compute the probability that the capacity equals some value j by summing the probabilities associated with those ensemble members for which Ci=j. (In the basic model, we simple count the number of members i for which Ci=j, and divide by k.)

We use a truncated Gaussian (normal) to model the probability distribution of the error in time (Δ time), the error in coverage/intensity (Δ threshold), and location (Δ translation). This distribution is bell-shaped, centered on 0 (no error), with a bounded support set (interval of possible values). Since our parameters are chosen among a discrete set of possibilities, we use the continuous model to compute the discrete probabilities of the parameters lying in corresponding discrete intervals. Refer to Figure 5.8. In particular, establish the support set to be between (max error) and +(max error), with the choice of standard deviation, σ, to be such that this interval has probability 0.95 (a user-specified parameter of our model) in the (untruncated) Gaussian model. (Recall that standard (unit) Gaussian distribution has probability approximately 0.95 for the interval between -1.96 and 1.96.) We partition the support set into the required number of equal-sized intervals, and assign probabilities to the corresponding discrete values associated with the intervals, as in Figure 5.8(b). The probabilities are re-normalized so that they sum to 1, as illustrated in Figure 5.8(c).

For example, consider time errors, and assume that the max error is chosen to be 10 minutes. Since we have data in 5-minute increments, we compute the capacity using the weather data of five different data sets: those corresponding to time errors of -10, -5, 0, 5, and 10 minutes. We consider a Gaussian distribution with probability mass of 0.95 inside the range (-10, 10); this implies a standard deviation of 5.10 (=10/1.96). Then, we divide the range (-10,10) into five equal-sized intervals (-10, -6), (-6, -2), (-2, 2), (2, 6) and (6, 10). Probabilities of each interval are shown in Figure 5.8(b), and the rescaled



(a) Gaussian distribution of error with $\mu = 0$ and $\sigma =$(max error)/1.96

(b) Divide error interval by the number of discrete error values and their probabilities

(c) Re-normalize the probabilities of each interval and assign them to discrete error values

Figure 5.8: Truncated Gaussian distribution used to model error for a given maximum error value, "max error". Here, there are five discrete error values(including zero error) in the illustration.

probabilities are shown in Figure 5.8(c). These are the probabilities we assign to each time-stamp. We compute the capacity associated with each time-stamp and weight it by the corresponding probability; this allows us to compute the probability distribution of capacity, assuming that the time error has a discrete, truncated Gaussian distribution. We similarly analyze error models of threshold error and translation error, for different choices of max error.

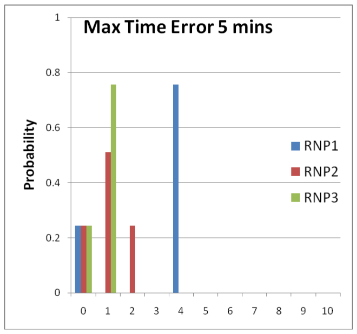
In Figure 5.9 we plot the probability distributions of capacity based on our simple ensemble model, for one particular sample data set, 4/30_1 (04/30/2009 14:30:00 Z). In Figure 5.9(a), (b), and (c), we show the probability distributions for different choices of the max error in time (5 min, 30 min, and 60 min, corresponding to standard deviations of 2.5 min, 15 min, and 30 min). We show three plots, distinguished by color, corresponding to different choices of RNP (lane width). For example, we see that the probability is just over 0.2 that there will be 2 air lanes available at RNP-3 when the time error has standard deviation 15 minutes (max error of Δ time = 30 minutes). Note too, that, as expected, the distribution of capacity "spreads out" (has higher variance) for larger choices of max error. For example, the RNP-1 capacity is concentrated mostly at 4 air lanes for max time error of 5 minutes, but becomes more spread out, ranging from 1 to 6 air lanes, for max time error of 60 minutes. Figure 5.9(d), (e), and (f) shows the probability distribution of capacity for different choices of max error in Δ threshold (0.2, 1.0 and 2.0 VIL), and Figure 5.9(g), (h), and (i) shows the probability distribution for different choices of max error in Δ translation (0.2, 0.6, and 1.0 degrees). We note that, for this particular weather data, the capacity distribution appears to be more dependent on errors in time and threshold than in errors in translation.

In Figure 5.10 we look more closely at the dependence of capacity error, quantified by the standard deviation in the capacity random variable, on the magnitude of error in Δ time, Δ threshold, and Δ translation, quantified by the max error (twice the standard deviation), according to the values in Table 5.1. Translation errors in Figure 5.10 are about latitude and longitude both.

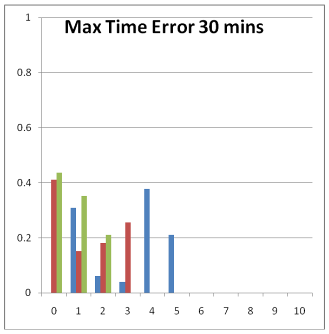
Our min-cut based capacity analysis provides an upper bound on the number of air lanes that are expected to be sent through an en route or transition airspace. The archetypical situation is highly weather-impacted transition airspace around a major metroplex, for example in the Chicago or New York areas. Quadrant-based terminal airspace capacity estimation algorithms can provide a framework for decisions on the best utilization of the airspace, particularly if a probabilistic model is available for reasoning about uncertainty. For example, using probability distributions based on an ensemble model of stochastic weather, such as those shown in Figure 5.9, can assist in ATM decisions about the amount and type (RNP) of traffic that should be permitted to enter the transition airspace for each quadrant, over a given time horizon. For example, this data may be useful in a decision support tool that may suggest, e.g., that all but one fix be restricted to high-performance aircraft (RNP-1), with low-performance aircraft being redirected to alternate fixes, until the weather event is predicted to dissipate.

5.3.7 Comparison of Capacities Based on Forecasts to Capacities Based on Actual Weather

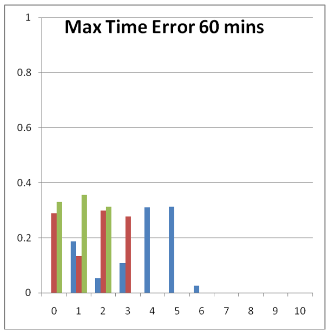
We also compared the capacities computed based on forecast data, of different look-ahead times, with the capacities computed for the actual observed weather (the nowcast, look-ahead 0). In Figure 5.11, we plot, as a function of the look-ahead time of the forecast, the capacity difference: capacity based on forecast, minus capacity based on nowcast. We used forecast data with look-ahead times of 5, 10, 15,. . . ,120 minutes. We see that, as expected, the errors generally grow as the look-ahead time increases. In Figure 5.11(a), we see that, among the seven samples analyzed, the greatest errors in capacity were those corresponding to data set 6/1_2, for which the capacity in the en route airspace based on forecast data is substantially greater than the capacity based on observed weather. In Figure 5.11(b), we see that for the samples analyzed,



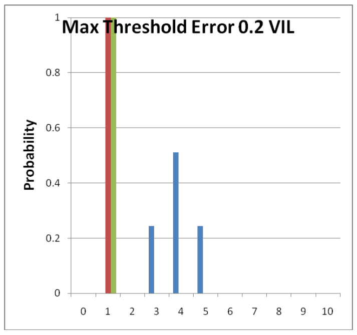
(a) Δtime = 5 mins



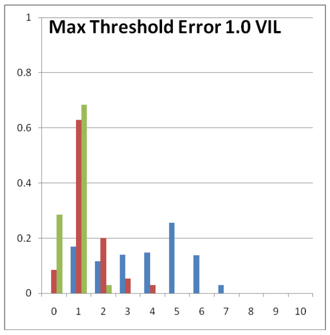
(b) Δtime = 30 mins



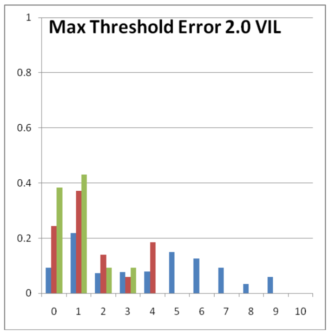
(c) Δtime = 60 mins



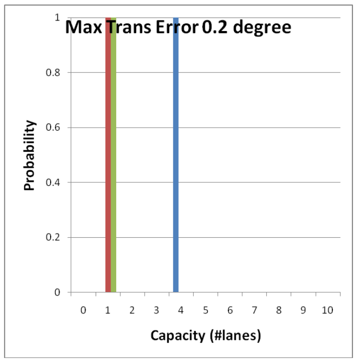
(d) Δthreshold = 0.2 VIL



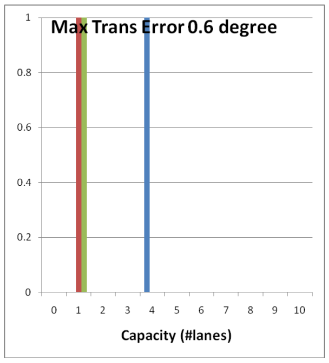
(e) Δthreshold = 1.0 VIL



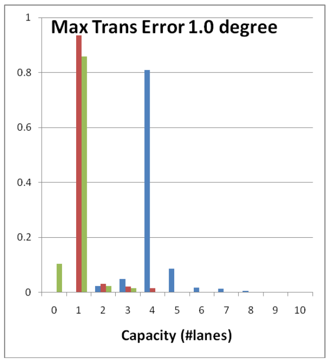
(f) Δthreshold = 2.0 VIL



(g) Δtranslation = 0.2 deg.

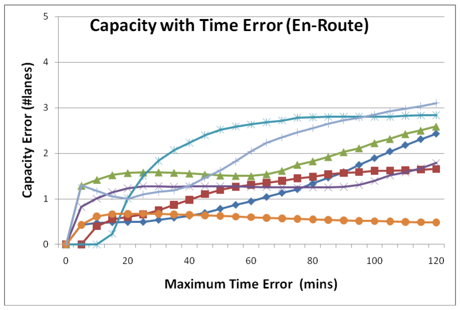


(h) Δtranslation = 0.6 deg.

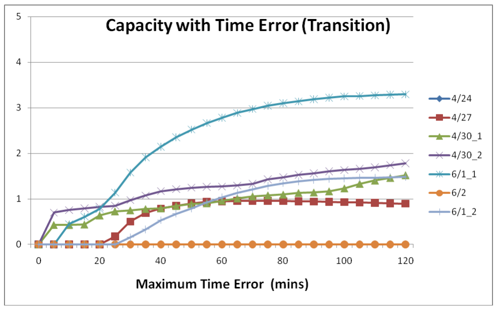


(i) Δtranslation = 1.0 deg.

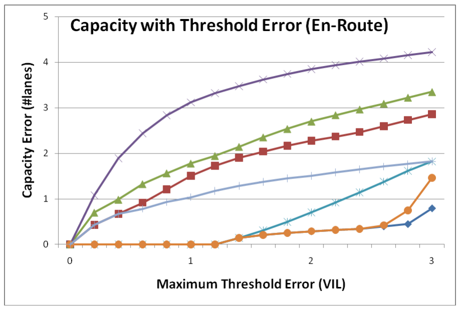
Figure 5.9: Probability distributions of capacity for different choices of max error in Δtime, Δthreshold and Δtranslation, for the en route airspace for 04/30/2009 14:30:00 Z (angle 0, threshold 3.0 VIL). For comparison, the (exact) computed capacity is 4, 1, and 1 air lanes for RNP-1, RNP-2 and RNP-3, respectively.



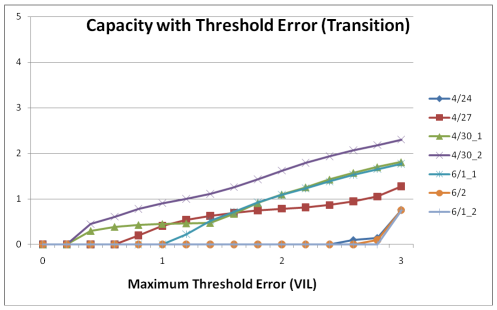
(a) En route with Time Error



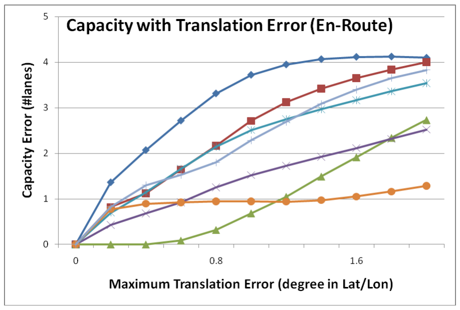
(b) Transition with Time Error



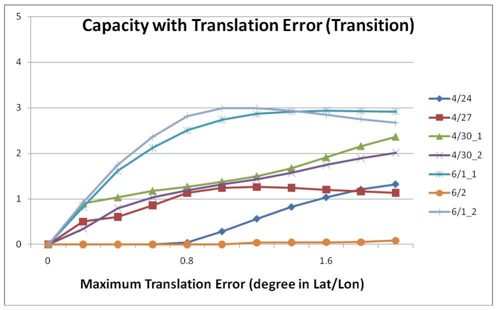
(c) En route with Threshold Error



(d) Transition Threshold Error



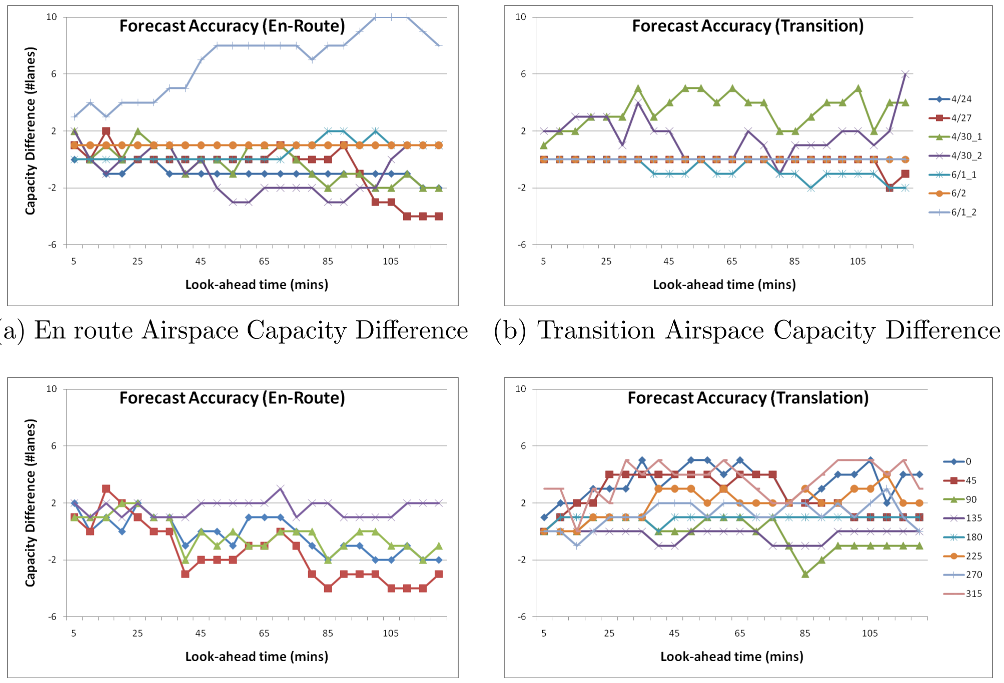
(e) En route with Translation Error



(f) Transition with Translation Error

Figure 5.10: Capacity error with different type of errors. Fixed orientation (angle 0) is used for all sample points.

there was some bias: the capacity based on forecast data was generally higher than the capacity based on the nowcast. Figure 5.11(c) and 11(d) show, for one sample (04/30/2009 14:30:00 Z), how the capacity difference varies with the look-ahead time, for each of the considered orientations of the airspace.



(a) En route Airspace Capacity Difference (b) Transition Airspace Capacity Difference

(c) En route Airspace Capacity Difference (d) Transition Airspace Capacity Difference

Figure 5.11: Difference between capacity based on forecast and capacity based on observed weather. (a) and (b) show differences in computed capacity for the seven samples with fixed orientation (0 degrees). (c) and (d) show differences in computed capacity for the 4 or 8 different orientations of the airspace, for one particular sample, 04/30/2009 14:30:00 Z

5.4 Conclusion

This paper studies the sensitivity of capacity estimates subject to weather forecast uncertainties. En route and transition airspace capacity estimation problems were explored. Capacity was computed using the max-flow/min-cut theory to determine the maximum possible number of air lanes, of fixed width (RNP) that can permeate a constrained airspace. By examining errors in time, in coverage (by means of varying the intensity threshold, in VIL units), and translation, one can compute probability distributions of capacity, under a simple probabilistic error model, as in Figure 5.9. This allows one to relate uncertainty in weather prediction to uncertainty in capacity estimation, leading to new decision support tools to assist TFM. Since weather prediction involves complex models of inherently complex phenomena, these results are only a first step in a quest to develop tools to help managers deal with the difficult task of quantifying the impact of weather forecast uncertainty on air traffic management.

Acknowledgment

This work is being performed under NASA's NGATS Airspace Super Dense Operations (ASDO) research focus area under contract NNA07BA84C, supervised by Doug Isaacson from NASA Ames Research Center Finally, we appreciate the financial support of the sponsor of the research, NASA Ames Research Center and the NextGen Project Manager, Mr. Paramal Kopardekar. J. Mitchell is partially supported by the National Science Foundation (CCF-0528209, CCF-0729019).

Chapter 6

Scheduling Aircraft to Reduce Controller Workload

Abstract

We address a problem in air traffic management: scheduling flights in order to minimize the maximum number of aircraft that simultaneously lie within a single air traffic control sector at any time t. Since the problem is a generalization of the NP-hard no-wait job-shop scheduling, we resort to heuristics. We report experimental results for real-world flight data.

Keywords: Air Traffic Management, trajectory scheduling, flight plan scheduling, no-wait job shop.

6.1 Introduction

In the air traffic control system, the volume of airspace in the altitude range that aircraft utilize is partitioned into a set of *sectors*. We consider the set of all trajectories flown between city pairs. Any one trajectory is modeled

This chapter is based on joint-work with with Alexander Kröller of Braunschweig University of Technology, Joseph Mitchell and Girishkumar Sabhnani of Stony Brook University. It was appeared in Proceedings of the 9th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS2009). [24]

as a polygonal path, with each vertex (*way point*) being specified by a point, (x, y, z, t), in space-time. For a given set of sectors and a given set of trajectories, we can compute the *occupancy count*, $n_\sigma(t)$, of a sector σ at any time t. For purposes of air traffic control, it is important that $n_\sigma(t)$ not be "too large"; often the occupancy count is compared with the *Monitor Alert Parameter* (MAP) value of the sector σ, which is related to the "capacity" of the sector. Depending on the timing and routing of the flights, though, the MAP values of certain congested sectors are often predicted to be exceeded (if current flights remain on filed flight plans), resulting in the rerouting of aircraft to avoid those sectors that are anticipated to be at or near full capacity during some period of time.

We consider the following scheduling problem: For a given set of trajectories and a given sectorization of airspace, determine alternate departure times "close" to the originally scheduled times so that the modified trajectories result in minimizing $\max_{\sigma,t} n_\sigma(t)$, the maximum occupancy count of a sector over a time window of interest.

6.2 Problem Statement

Formally, the Min-Max Sector Workload Problem (MMSWP) is defined as follows. We are given a set Σ of *sectors* and a set Θ of periodic *flight plans*. The common period of all plans is T, e.g., $T = 24$ hours. Corresponding to each flight plan θ is a sequence $\Sigma_\theta = (\sigma_{\theta,1}, \sigma_{\theta,2}, \ldots)$ of the sectors it visits, where $\sigma_{\theta,k} \in \Sigma$, $\forall k$. Flight plan θ also has an associated *departure time* $d_\theta \in [0, T)$, and for each sector $\sigma_{\theta,k}$ it has an associated *dwell time*, $t_{\theta,k}$ (length of time in sector).

Assuming a flight θ departs daily with a delay of Δ_θ, it will therefore be in sector $\sigma_{\theta,k}$ during the intervals

$$I_\theta(\sigma_{\theta,k}, \Delta_\theta) := \left[\sum_{\ell<k} t_{\theta,\ell}, \sum_{\ell\leq k} t_{\theta,\ell}\right) + d_\theta + \Delta_\theta + T\mathbb{Z}. \tag{6.1}$$

Therefore, at time $t \in [0, T)$ (and also $t + zT$ for any $z \in \mathbb{Z}$), a total of

$$n_\sigma(t) := |\{\theta \in \Theta : t \in I_\theta(\sigma, \Delta_\theta)\}| \tag{6.2}$$

flights will be in sector $\sigma \in \Sigma$.

Our goal is to find delays $(\Delta_\theta)_{\theta\in\Theta}$ to minimize the overall maximum occupancy count, $\max_{\sigma,t} n_\sigma(t)$. The delays are constrained to be within the range $[0, D]$ for parameter D. Note that additionally allowing flights to leave early, i.e., $\Delta_\theta < 0$, does not change the problem due to the periodicity of flight plans: A delay range $[-a, b]$ is equivalent to $[0, a + b]$, for $a, b > 0$. Therefore, we just consider the problem where $\Delta_\theta \geq 0$.

6.3 Job-Shop Scheduling and Related Work

No-wait job-shop scheduling is defined as follows (see [17]): We are given a set of m machines and a set of n jobs that have to be processed on these machines. For each job i, we are given a sequence r_{ik} indicating that job i has to be processed on the kth machine. Additionally, we are given the matrix p_{ij} $(1 \leq i \leq n, 1 \leq j \leq m)$, stating the processing time of job i on machine j. Furthermore, the following constraints hold:

- *Sequence*: Each job must be processed in order of its operations and no interruption (preemption) of an operation is allowed.
- *Synchronicity*: No job can be processed by two machines at the same time and no machine can process two jobs at the same time.
- *No-wait*: There must be no waiting time between two consecutive operations of the same job.

When there is no constraint on the maximum delay, i.e., $D \geq T$, our problem is equivalent to "no-wait job-shop scheduling". We represent each flight plan as a job and each sector as a machine. We seek to minimize *makespan*, i.e., the smallest time in which all jobs can be processed, where no two jobs can be on the same machine at the same time. The no-wait constraint ensures that, once started, a job can neither be delayed between machines nor suspended while being processed on one. An optimal solution to the job-shop problem with makespan M can be converted trivially to a flight plan solution with maximum occupancy $\lceil M/T \rceil$. Vice versa, an algorithm for flight plan scheduling also solves job-shop by finding the largest λ for which a flight plan with all processing times scaled by λ can be scheduled with maximum occupancy 1. This can be achieved using binary search.

Lemma 6.3.1. *Minimizing makespan in the no-wait job-shop scheduling problem is polynomially equivalent to the Min-Max Sector Workload Problem (MMSWP).*

No-wait job-shop scheduling has been studied in several papers; see, e.g., [35, 50, 60, 49, 33]. Bansal et al. [3] give a *PTAS* for a special case of the problem and show hardness of approximation for another case. Karger et al. [21] provide a survey of scheduling algorithms, defining the various terms and known results for some of the basic problems. Since the job-shop problem is NP-hard, so is the MMSWP, by Lemma 6.3.1.

Ariano et al. [11] formulate train scheduling as a job shop problem with no-store constraints. Bertsimas et. al [4] solve an optimal combination of flow management actions, including ground holding, rerouting, speed control and airborne holding on a flight-by-flight basis.

6.4 Simplified Cases

In this section, we examine some special cases of the problem. In all the cases here, we consider $D = T$, so that there are no maximum delay constraints.

6.4.1 One-Sector Problem

In the simplest of cases, there is only sector σ_0 and hence all the flight plans just define the time interval the flight remains in this sector. For all $\theta \in \Theta$, $\sigma_{\theta,1} = \sigma_0$.

If we remove periodicity of flight plans, i.e. put a constraint $d_\theta + \Delta_\theta + t_{\theta,1} \leq T$ hours for each flight θ, the optimal re-scheduling problem of minimizing the *max-workload* exactly maps to the bin-packing problem, which is known to be hard (by a reduction from set partition) and and to have an asymptotic PTAS [12].

If we consider periodic flight, then the *one-sector* problem has a trivial solution given by assigning delay to make flights back to back. This gives a max-workload of $\lceil \sum_{\theta \in \Theta} t_{\theta,1}/T \rceil$.

An asymptotic PTAS is an algorithm that, given $\epsilon > 0$, produces a $(1+\epsilon)$- approximate solution provided $OPT > C(\epsilon)$ for some function C, and runs in time polynomial in n for every fixed ϵ.

6.4.2 Two-Sector Problem

The extension of the problem to two sectors, with a periodic schedule of flights, seems like an interesting special case to understand the complications associated with the *no-wait* constraint and also the periodicity of the schedules. It is much easier to understand the *two-sector* problem by considering its exact equivalent below.

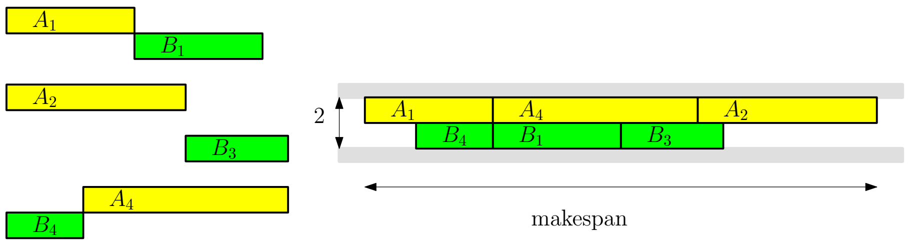


Figure 6.1: Left: 4 kinds of blocks. Right: The tight-fitting in the groove of size 2.

Consider Figure 6.1. Let A, B be the sectors. The yellow rectangles indicate the time interval of flights in A and the green rectangles indicate intervals in B. Yellow to the left of green indicates that flight starts in A and single yellow rectangle indicates the flight is only in A. Thus, the MMSWP corresponds to packing these blocks of rectangles as tightly as possible in the groove of width 2, constraining that yellow rectangles strictly remain in the upper row, green rectangles strictly remain in the lower row and none of the rectangles overlap.

It turns out that periodicity does not really help for this case, as this version of the problem also turns out to be *NP-complete* by reduction from 3-*PARTITION PROBLEM*.

Theorem 6.4.1. *The MMSWP within* 2 *sectors is NP-Complete.*

Proof. $3m$ numbers $a_1, a_2, \ldots, a_{3m}$ are given for a 3-PARTITION PROBLEM instance P. All of these number are between $B/4$ and $B/2$, where mB is the total sum of $a_1, \ldots, a_{3m}$. We show the optimal solution of minimizing workload overall sectors gives us the solution of this problem.

Let's construct the MMSWP problem instance corresponding given input m, B, and a_i's. There are two sectors σ_1 and σ_2. Let time horizon T be $(mB+m)$. For given numbers a_i where $i \in \{1, \ldots, 3m\}$, we generate flights θ_i which visits only σ_1 with staying time a_i, i.e., $\Sigma_{\theta_i} = (\sigma_1)$ and $t_{\theta_i,1} = a_i$ for $i \in \{1, 2, \ldots, 3m\}$.

And we prepare additional m flights $\theta_{3m+1}, \ldots, \theta_{3m+m}$ which visit σ_2 for time $(B+1)$ and then σ_1 for 1. i.e, $\Sigma_{\theta_j} = (\sigma_2, \sigma_1)$ and $t_{\theta_j,1} = (B+1), t_{\theta_j,2} = 1$ for $j \in \{3m+1, \ldots, 3m+m\}$.

Then, we claim that if we minimize maximum workload over all sectors for this problem as 1, then we are able to solve given P.

In order to make workload as 1 for σ_2, we have to arrange $\theta_{3m+1}, \ldots, \theta_{3m+3}$ back-to-back like dark-gray blocks in Figure 6.2. Then there are m intervals with length B in σ_1. Now finding a placement of $\theta_1, \ldots, \theta_{3m}$ (light gray blocks in Figure 6.2) to make workload of σ_1 as 1 is finding a partition of $\{a_1, \ldots, a_{3m}\}$ such that each sum is exactly B.

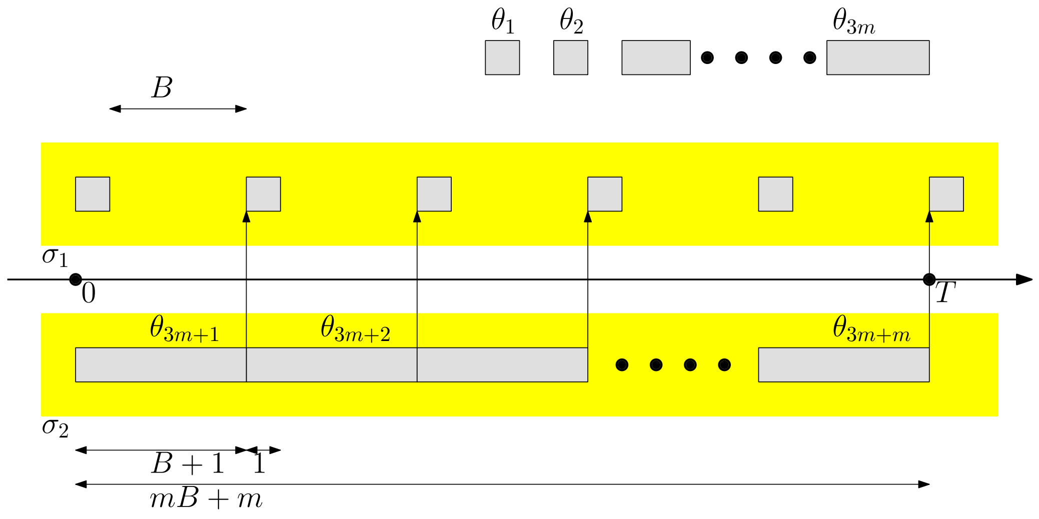


Figure 6.2: 2 sectors workload problem construction for given 3-Partition problem instance

$\square$

6.5 Algorithms

In this section, we present heuristics to solve the MMSWP.

6.5.1 Shifting

Starting with the original flight schedule, we pick the sector with worst max-workload (in case of tie check each one of them), and look at the time interval where the max-workload is worse. All the flights present in the sector in that time interval are considered for re-scheduling (shifting) and the one which gives the "best" improvement is selected greedily. The goodness of a shift is judged by its effect on the workload vector which stores the workloads of all sectors in the sorted order. The flight whose re-scheduling gives the best improvement in lexicographic ordering of the workload vector is selected (in case of ties, we pick the flight which has the least difference in the re-schedule time and the original schedule). The process is repeated till all shifts at a given iteration worsen the workload vector. (Note that shifts keep taking place even when the workload vector remains same).
We constrain the greedy shifting to be of the following three kinds:

- Right Shift - The flights are only allowed to be postponed.
- Left Shift - The flight are only allowed to be preponed.
- Short Shift - The decision of postpone/prepone is decided by the amount of shift, and the shorter one is picked.

It is possible to get into loop if we allow shifts in both directions. In our experiments, we only use right shifts to finish algorithm certainly. Since we allow shifts without strict workload vector improvement, all shifts after the last workload vector change are restored when the algorithm is finished.

We also devise an incremental heuristic, in which flights are added one by one (in a random order). With each new flight addition, we run complete experiment of a shift heuristic considering all the flights previously added along with this one.

6.5.2 Randomized Rounding

The *randomized rounding* algorithm solves a linear problem formulation whose variables describe a probability distribution for each flight plan. Then, a solution is generated by drawing delays from these distributions.

We evenly divide the interval $[0, D]$ into a discrete set of delays $\{0 = d_0, d_1, \ldots, d_m = D\}$. Also we slice the 24h-period T into n pieces $\{0 = t_0, t_1, \ldots, t_n = T\}$.

For each flight θ, the linear formulation has a variable $x_\theta(d_i)$ for each $d_i, 0 \leq i \leq m$. The interpretation (in terms of the finally assigned delay Δ_θ) is

$$x_\theta(d_i) = \Pr[\Delta_\theta \geq d_i] \ .$$

So the $x_\theta(\cdot)$ define a probability function on $[0, D]$ for every flight (the density is constant within each interval $[d_i, d_{i+1})$, that is, the distribution is uniform within each interval). To make sure the $x_\theta(d_i)$ define a proper probability distribution, we use the constraints

$$1 = x_\theta(d_0) \geq x_\theta(d_1) \geq \cdots \geq x_\theta(d_m) = 0.$$

This means the probability that a flight delay is in the range $[d_i, d_j]$ is $x_\theta(d_i) - x_\theta(d_j)$, so the probabilities are nicely encoded in the formulation. Note that

$$\Pr[\text{flight } \theta \text{ is in sector } \sigma \text{ at time } t]$$

is a linear term in the $x_\theta(\cdot)$ variables. To see this, translate t into a range $[\underline{\Delta}_\theta, \overline{\Delta}_\theta]$ of delays where a flight would start to be in σ at t. The probabilities are then:

- Some of the first interval with $d_i \leq \underline{\Delta}_\theta \leq d_{i+1}$, that is,

$$\Pr[\theta \text{ is in } \sigma \text{ at } t \ , \ \Delta_\theta \in [d_i, d_{i+1})] = \frac{d_{i+1} - \underline{\Delta}_\theta}{d_{i+1} - d_i}(x_\theta(d_i) - x_\theta(d_{i+1})) \ .$$

- All of the intervals $\underline{\Delta}_\theta \leq d_i \leq \ldots d_{i+1} \leq \overline{\Delta}_\theta$, in a similar fashion.
- Some interval part around $\overline{\Delta}_\theta$, again analogous to the first case.

By adding the cases, one can see how $\Pr[\theta \text{ is in } \sigma \text{ at } t]$ is a linear term with up to four coefficients. Obviously there are a number of special cases when $[\underline{\Delta}_\theta, \overline{\Delta}_\theta] \not\subseteq [0, D]$; these are easy to resolve and left out in this presentation. So we can now describe the expected load of sector σ at time t by the linear term

$$\mathrm{E}[\text{number of flights in } \sigma \text{ at time } t] = \sum_{\theta \in \Theta} \Pr[\theta \text{ is in } \sigma \text{ at } t].$$

Hence, we solve the following LP:

$$\begin{aligned}
\min\ & C \\
\text{s.t.}\ & \text{E[number of flights in } \sigma \text{ at time } t] \leq C \quad \forall \sigma \in \Sigma, t \in \{T_o, \ldots, T_n\} \\
& 1 = x_\theta(d_0) \geq x_\theta(d_1) \geq \cdots \geq x_\theta(d_m) = 0 \quad \forall \theta \in \Theta\ ,
\end{aligned}$$

which gives us a probability distribution for each Δ_θ, so we now generate actual Δ_θ values following these distributions.

An interesting variant arises when we add integrality constraints to the LP, as this forbids smearing flights over many delay intervals. As the resulting IPs are typically impossible to solve within reasonable time, we employ a different strategy: First, the LP-based heuristic is run. We identify the most crowded sectors, and add integrality constraints for tracks passing these sectors. At the same time, we vary n and m for different sectors and tracks, such that the crowded sectors get a more detailed formulation than the others.

6.6 Lower Bounds

6.6.1 A Simple Bound

The optimal one sector solution for a sector σ (refer to Section 6.4.1), for $D = T$, independent of any other sector, is a naive lower bound to its max-workload attained by any scheduling, for any D. Thus, we can optimize each sector individually, and pick the maximum value over all sectors, to obtain a lower bound on the workload attained by an optimal scheduling.

6.6.2 Linear Programming

The second lower bound algorithm is based on the randomized rounding algorithm. Assume that all the $x_\theta(\cdot)$ are binary, i.e., 0 or 1 (see 6.5.2 for details). If now $x_\theta(d_i) - x_\theta(d_j) = 1$, then flight θ will have a delay $\Delta_\theta \in [d_i, d_j]$.

For a track $\theta \in \Theta$, a sector $\sigma \in \Sigma$ and a time t, we again compute the interval $[\underline{\Delta}_\theta, \overline{\Delta}_\theta]$ of delays for θ under which θ will be in σ at t. Then we determine the smallest $d_i \geq \underline{\Delta}_\theta$ and the largest $d_j \leq \overline{\Delta}_\theta$. Then, when $x_\theta(d_i) - x_\theta(d_j) = 1$, the flight will be in σ at t. So define $g_\theta(\sigma, t) := x_\theta(d_i) - x_\theta(d_j)$.

	No. of Sectors	Alt-Range	Flights	Time Window
Set1	5	$\geq$ 24k feet	1904	$0-24$ hrs
Set2	18	$\geq$ 24k feet	3063	$0-24$ hrs
Set3	57	$\geq$ 0 feet	12123	$0-24$ hrs
Set4	1281	Different	11986	$14-18$ hrs
Set5	16	$\geq$ 24k feet	4994	$0-24$ hrs

Table 6.1: Summary of data sets used for experimentation.

The following IP charges 1 towards the maximum capacity C when a track is guaranteed to be in σ at t:

$$\begin{array}{lll}
\min & C & \\
\text{s.t.} & \displaystyle\sum_{\theta\in\Theta} g_\theta(\sigma,t) \leq C & \forall\sigma\in\Sigma, t\in\{T_o,\ldots,T_n\} \\
& 1 = x_\theta(d_0) \geq x_\theta(d_1) \geq \cdots \geq x_\theta(d_m) = 0 & \forall\theta\in\Theta \\
& x_\theta(d_i)\in\{0,1\} & \forall\theta\in\Theta, i=0,\ldots,m
\end{array}$$

The optimal solution to this IP is a lower bound to the original problem. For efficiency reasons, we do not solve this IP directly, but rather its LP relaxation, which is obtained by dropping the integrality constraint.

6.7 Results

We use real-world flight track data and sector data from the National Airspace System (NAS). The data, as shown in Table 6.1, is divided into 5 sets depending on the number of sectors. The *alt-range* defines the range of altitude for the air-traffic in the sectors. The high-altitude sectors typically have *alt-range* $24,000$ feet and above. **Set1**, **Set2** and **Set3** consider flight tracks for the entire 24 hour time period while **Set4** considers only the flights that overlap a 4 hour time window. Note that the flight times may start or end outside the 4 hour time window. Also, **Set4** includes all the sectors spanned by these flights, thus having high-altitude sectors, low-altitude sectors and some sectors from Canada as well.

Set5 (random data) consists of a 300×300 nautical miles square region divided into 16 sectors in the form of a square grid. Then, 64 (uniform) random cities were generated such that 10% of cities had weight 10, 15% had weight 5,

	Set1			Set2			Set3		
	m	n	Time	m	n	Time	m	n	Time
LP Lower	30	720	1:20	30	720	1:50	30	720	9:10
MIP Lower	30	720	3:04	–	–	–	12	288	10:18
Rand. Rounding	30	720	22:24	12	288	1:05	12	288	30:07
MIP Rounding	12	288	0:28	12	288	0:33	12	288	56:17

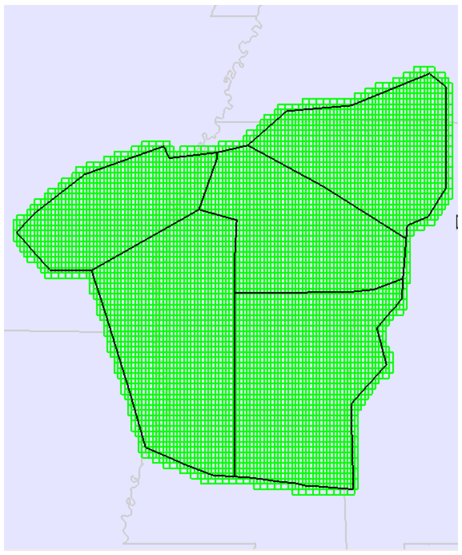
	Set4			Set5		
	m	n	Time	m	n	Time
LP Lower	60	1440	17:19	30	720	10:26
MIP Lower	12	288	14:44	–	–	–
Rand. Rounding	30	720	57:11	12	288	10:18
MIP Rounding	12	288	17:30	12	288	5:13

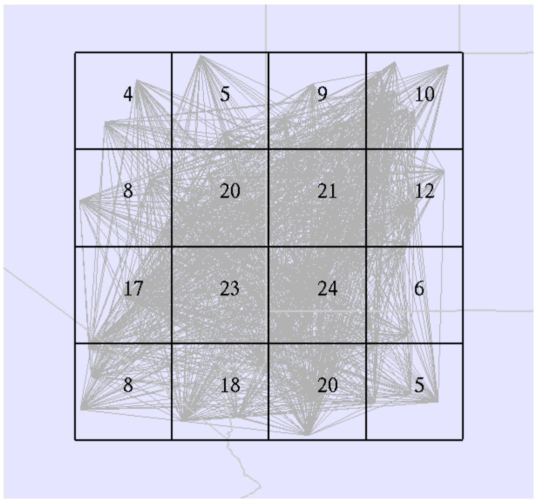
Table 6.2: Details for LP-based heuristics, showing the discretization granularity and total algorithm run-times in minutes.

and the remaining had weight 1. In total, 4994 random flights were generated between (weighted uniform) randomly chosen city pairs, with each city having probability of selection proportional to its weight. The departure-time of a flight was (uniform) randomly generated between $0 - 24$ hours. The (constant) speed of an aircraft was modeled as a (uniform) random variable between 200 and 600 nautical miles per hour. The arrival-time of a flight was calculated from the departure time, the speed of the aircraft, and the distance between the cities in the pair. An additional constraint was added that no two aircraft depart from (or arrive) at a city within 1 minute of each other. A visualization of data sets **Set1**, **Set2** and **Set5** can be seen in Figure 6.3.

We implemented our algorithms and ran them on the five data sets. For the LP-based algorithms, we used CPLEX 10.0 on a 3.0 GHz Linux machine. We solved each instance using a few parameter sets, varying the number of discretizations in delay (i.e., m) and daytime slices (i.e., n). The most often used values of $m = 30$ and $n = 720$ correspond to having one variable per two minutes of delay and one constraint for every other minute of the day. We imposed a run-time limit of 60 minutes on the algorithm. Table 6.2 describes these runs and lists the according algorithm run-times. Run-times for the other heuristics are not listed, as they always finish within a few seconds.

Table 6.3 shows the comparison of max-workload statistics of the given


(a) Set1 sectors and the underlying square grid (and shifted square grid) cover (grid resolution: 0.1x0.1)



(b) Set5 Randomly generated flight tracks with the underlying sectors

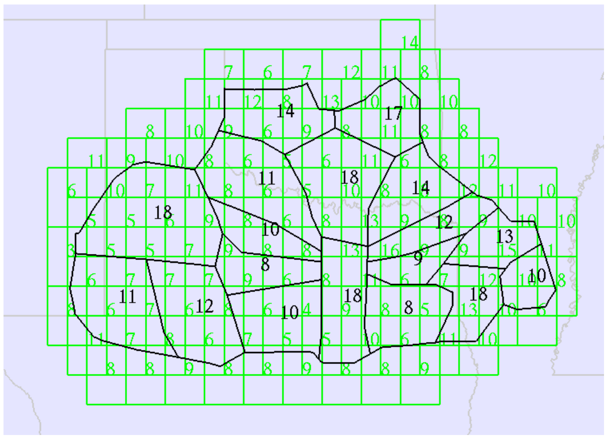

(c) Set2 sectors and grid cover (1x1)

Figure 6.3: Visualization of data sets. (The Numbers in the sectors indicate the max-workload counts for the used flight schedules)

	Set1			Set2			Set3		
	Max	Mean	Var	Max	Mean	Var	Max	Mean	Var
Original plan	22	18.00	6.80	18	12.83	12.25	38	21.56	36.70
Right Shift	18	16.40	1.04	14	11.11	3.99	31	20.77	26.27
Incr. Right Shift	15	13.80	0.96	**12**	10.17	2.25	**26**	18.75	16.40
Rand. Rounding	**14**	13.40	0.24	14	11.67	4.00	28	22.94	19.50
MIP	15	14.40	0.24	14	11.22	4.73	28	23.47	16.18
Lower Bound	Naive	LP	IP	Naive	LP	IP	Naive	LP	IP
	6	**9**	**9**	5	**8**	–	16	**20**	14

	Set4			Set5		
	Max	Mean	Var	Max	Mean	Var
Original plan	58	7.67	37.88	24	13.00	46.13
Right Shift	47	7.61	36.35	19	11.75	29.01
Incr. Right Shift	**39**	7.51	34.50	**17**	10.81	20.66
Rand. Rounding	42	8.04	40.50	19	12.50	25.00
MIP	43	8.22	44.90	19	12.50	30.13
Lower Bound	Naive	LP	IP	Naive	LP	IP
	12	**31**	22	**13**	11	–

Table 6.3: Workload statistics of algorithms. Max: Maximum Workload, Mean: Mean of workload, Var: Variance of workload

	Set1 (1904 flt)			Set2 (3063 flt)			Set3 (12123 flt)		
	Max	Total	Avg	Max	Total	Avg	Max	Total	Avg
Right Shift	6	46	1	9	5:25	1	17	5:18	1
Incr. Right Shift	49	2:00:46	4	52	3:16:21	6	60	18:21:7	6
Rand. Rounding	60	13:22:24	10	60	13:06:48	6	60	35:18:15	4
MIP	60	14:21:48	12	60	15:21:42	7	60	37:10:59	4

	Set4 (11986 flt)			Set5 (4994 flt)		
	Max	Total	Avg	Max	Total	Avg
Right Shift	53	12:53	4	7	3:8	1
Incr. Right Shift	60	14:22:54	17	54	4:18:5	4
Rand. Rounding	58	50:10:59	6	55	59:16:33	17
MIP	55	90:00:38	11	55	60:05:50	17

Table 6.4: Time shift statistics of various methods. Max: Max shift, Total: Sum of absolute value of shift, Avg: Average of absolute value of non-zero shifts. (format 14:21:48 means 14 days 21 hours 48 minutes)

flight plans, the heuristic solutions and the LP based methods. The maximum allowable shift to any flight schedule was constrained to be 1 hour in all methods. The discretization of time for LP/IP methods is 1 minute. The results show a considerable improvement over the workloads of each sector arising due to the original flight schedules. Even the variance values have gone down significantly, indicating more balance of workload across sectors. In particular, the incremental shift heuristic seems to out-perform all the other methods. Note that the shifting heuristics do not discretize the time like LP/MIP methods. The '-' values in Table 6.3 refer to experiments for which no solution was found during more than a week of running time.

Table 6.3 also shows the lower bound calculations for the 5 sets. The best solutions are still not close to the computed lower bounds, but we believe they are very close to optimal solutions. Future work will specifically aim to improve the lower bounds.

Table 6.4 shows the statistics of the amount of time shifts from the original schedule. *Max* indicates the maximum shift in any flight schedule, *Total* indicates the sum of absolute values of shifts, and the *Avg* gives the average time shift of all flights with non-zero shifts. The value of *Total* in the case of the right shift heuristic is noticeably small compared to other methods, possibly because of early termination due to reaching a local minimum. Also, the average time shift is seen to be low for all the methods, suggesting that we can get considerable improvements in workloads with reasonable modification to the schedules.

6.8 Other Workload Considerations

Apart from the *max-workload* of a sector, there are other workload issues which are significant from the controller perspective. One of them, usually referred to as *coordination* workload, deals with the hand-offs between controllers when an aircraft moves from one sector to the other. Another critical issue is the *conflict resolution* workload, which is related to monitoring the aircraft when they are expected to be simultaneously present at (or near) the same geographic point (a "conflict point"). Note that even if two aircraft are flying at different altitudes, at the conflict point, they demand special attention of the controller.

While re-scheduling flights has no effect on the *coordination* workload, it can favorably affect the *conflict resolution* workload, by reducing the number of conflict points. It is easy to incorporate conflict resolution workload in the model, as we now discuss.

We sub-divide the region (spanned by the sectors) into (reasonably) small size cells and compute the max-workload in each cell separately. If the size of the cell is small, a high max-workload cell corresponds to a conflict point, where multiple aircraft are in close proximity simultaneously. We add these cells as new (artificial) sectors to the data set and try to minimize their workload vector separately, thereby (possibly) decreasing the number of conflict points.

The shifting heuristic is now modified to be a two-step procedure. The first step considers the overall maximum value of the max-workload across all cells to be a constraint: The aircraft are re-scheduled to improve the workload vector of the sectors, as before, while keeping the workloads in all cells below a specified W_c. In the second step, the roles of sectors and cells are reversed: The optimized maximum value of the workload of the sectors is treated as a constraint, and the aircraft are re-scheduled with the objective of improving the workload vector of the cells.

For experimentation, these cells come from a uniform (square) grid and a shifted uniform grid as shown in Figure 6.3 covering the region spanned by the sectors. Two different side lengths of square grid cells are used, 0.1×0.1 and 0.2×0.2 (unit latitude/longitude degrees). In Set1, Set2 and Set5, 1 degree corresponds to somewhere in the range of $35 - 60$ nautical miles. Table 6.5 shows the results of the workload improvements with the cell constraints. We observe that the max-workloads of the sectors still improve, compared with the original (18 v/s 22 for Set1), while the number of conflict points are considerably decreased (see Figure 6.4). For Set1, after scheduling there are no grid cells with workload 4, while the number of cells with workload 3 has also decreased by more than 90%.

6.9 Scheduling on Square Cells

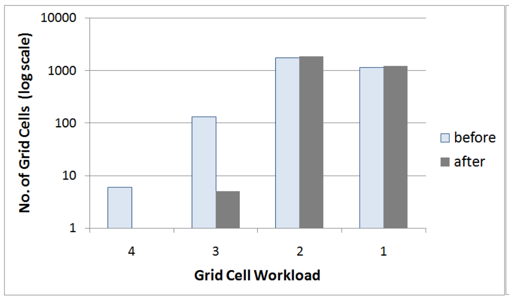
We consider a special case of the problem, whose sectors are same sized square cells. On the 2 dimensional lattice of cells, we are given a set of rectilinear directional paths, each of which is placed on the chain of square cells. Moving objects follow those paths and they proceed from one cell to neighboring cell in an unit time. Paths are required to convey moving objects periodically with given interval. For instance, if a path is assigned interval 3, then there should

Grid Size	Set1 (Given SMax: 22)				
	Given		Shifted		
	GMax	GMean	SMax	GMax	GMean
0.1×0.1	4	1.670	18	3	1.604
0.2×0.2	5	2.446	18	4	2.356

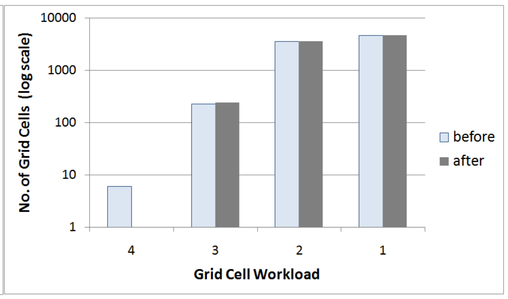
Grid Size	Set2 (Given SMax: 18)				
	Given		Shifted		
	GMax	GMean	SMax	GMax	GMean
0.1×0.1	4	1.467	14	4	1.478
0.2×0.2	5	2.105	14	4	2.083

Grid Size	Set5 (Given SMax: 24)				
	Given		Shifted		
	GMax	GMean	SMax	GMax	GMean
0.1×0.1	11	1.609	19	8	1.598
0.2×0.2	14	2.271	19	10	2.243

Table 6.5: Results of Right-Shift heuristic with additional grid constraints. SMax: Sector Max, SMean: Sector Mean, GMax: Grid Max, GMean: Grid Mean.



(a) Set1



(b) Set2

Figure 6.4: Grid cell max-workload before and after scheduling (Grid size 0.1×0.1)

be new moving object appear on the entry of the path every 3 time unit. The universal capacity threshold is 1. It means that no two objects are placed in same cell simultaneously.

The objective of this variation is finding appropriate phases for all paths. Available phases for a path with interval n are phase 1, phase 2, $\dots$, phase n. If the path takes phase i, then new moving object is scheduled on time i, $i+n$, $i+2n$, $\dots$, and so on. Figure 6.5 is an example. There are 5 thick rectilinear paths in the lattice of grid cells. Paths are crossing others, where the cells are shown as gray. Let's assume that interval period of P_1, P_2 and P_4 are all 2. It means that every second cells along those paths should have moving objects. If C_1 has moving object along P_4, then C_2 also has one along P_4. It forces P_1 and P_2 to have same phase and they are incompatible at C_3. Therefore, a scheduling satisfying the given interval conditions is not available.

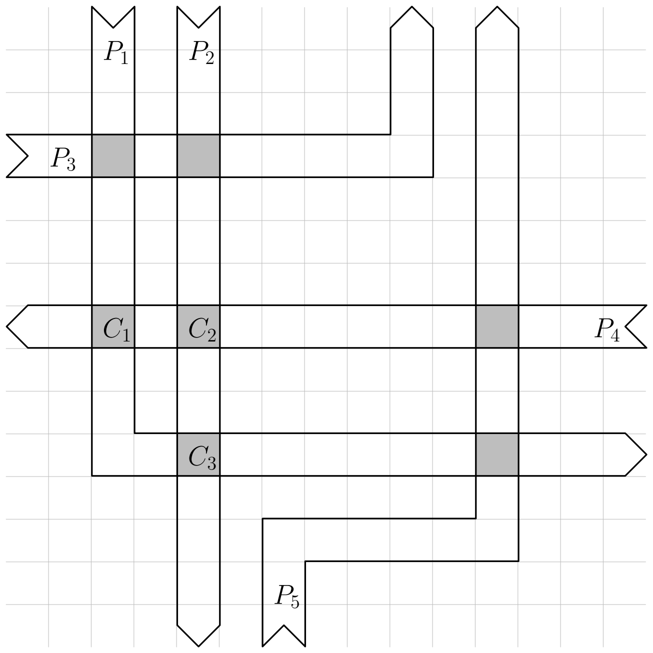


Figure 6.5: Rectilinear Directional Paths on Square Cells

If the interval period of all paths are 2, we refer this case as *every second scheduling*, then we determine whether valid scheduling exists by following observation. For any two paths sharing a common cell, if the phase of one path is determined, then the phase of the other is also determined. Therefore, algorithm for every second scheduling is following; picking a path and assigning a phase, then picking another path, which is intersecting with the phase determined path. Repeating assignment compatible phase to new one. Whenever appropriate assignment is failed, satisfying scheduling is not available.

But, if the interval period of all paths are 3, i.e *every third scheduling*, then it becomes hard to solve.

Theorem 6.9.1. *Every third scheduling is NP-Complete.*

6.9.1 Proof Idea

We use the reduction from Planar Monotone 1-in-3 SAT [32]. We design variable and clause gadget, then create every third scheduling problem instance Q with the gadgets for an instance P of planar monotone 1-in-3 SAT. If we are able to find appropriate phases for all paths in Q, then we also can find variable assignment for P.

An example of variable gadget is shown in Fig 6.6. Corresponding variable used in 3 clauses. Let's consider the time instance, when moving objects from s_1 toward t_1 reach cells denoted as 3. Since two consecutive cells are occupied in perspective of other two paths from s_2 to t_2 and s_3 to t_3, there is only one phase is available for them. Therefore all paths should have same phase in order to schedule every one in third time unit.

Fig 6.7 shows an example of clause gadget. There are three paths and every path intersects with other two paths, so three intersections exist in total. Let's assume a path has moving obstacles at cells denoted $\{0, 3, 6, 9, \ldots\}$. In order to avoiding overlap, other two paths should have either $\{1, 4, 7, 10, \ldots\}$ or $\{2, 5, 8, 11, \ldots\}$ exclusively. Therefore, every path should has different phase with others. In other words, at least one path among three has phase 1.

In P, a variable has two states - true or false, however in Q, variable gadget has three states - phase 1, phase 2 or phase 3. We will refer to phase 1 as true and other phases as false. Since every feasible scheduling has phase 1 path in all clause gadget, solution of Q leads the solution of P. Rest part about placing gadget and connecting them in order to build the instance of Q is abridged.

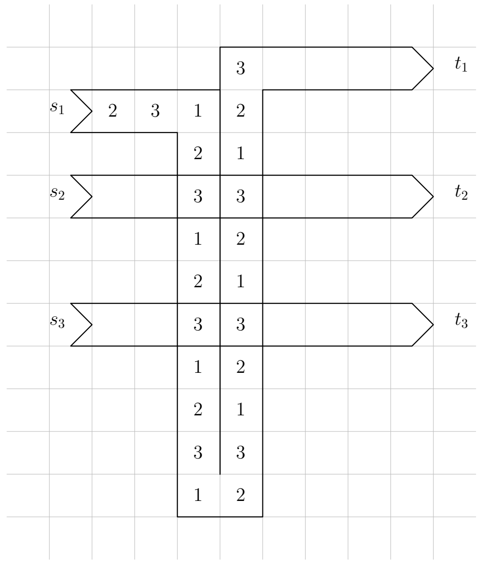


Figure 6.6: Variable gadget for Grid Scheduling Problem

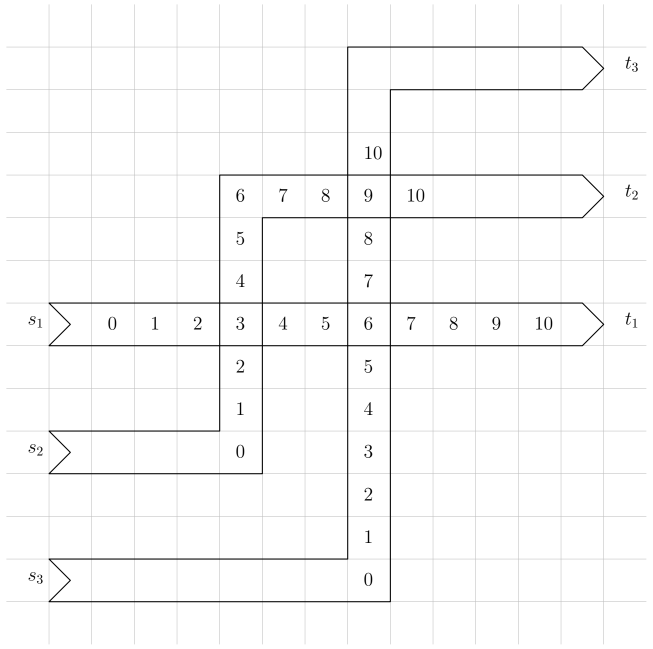


Figure 6.7: Clause gadget for Grid Scheduling Problem

6.10 Conclusion

We presented a periodic flight plan scheduling problem, proved it to be NP-hard, and proposed heuristics for which we reported experimental results on real-world data. The results show a considerable workload improvement over the originally scheduled flight times and come at low computational cost. The reduction in the number of conflict points was also impressive. Future work will specifically aim to improve the lower bound, as we believe that the heuristically produced solutions are already almost optimal. Also, we are interested in combining re-routing with re-scheduling to improve further the workloads.

Acknowledgment

The data used for the experiments was provided by Metron Aviation. We thank Michael Bender and Bob Hoffman for helpful discussions. This work was partially supported by NSF (CCF-0528209, CCF-0729019), NASA Ames, and Metron Aviation.

References

[1] E. M. Arkin, S. P. Fekete, J. Kim, J. S. B. Mitchell, G. R. Sabhnani, and J. Zou. The pencil packing problem. In *The 19th Fall Workshop on Computational Geometry*, Medford, MA, November 2009.

[2] E. M. Arkin, J. S. B. Mitchell, and V. Polishchuk. Maximum thick paths in static and dynamic environments. In *Proceedings 24th ACM Symposium on Computational Geometry*, pages 20–27, 2008.

[3] N. Bansal, M. Mahdian, and M. Sviridenko. Minimizing makespan in no-wait job shops. *Math. Oper. Res.*, 30(4):817–831, 2005.

[4] D. Bertsimas, G. Lulli, and A. Odoni. The air traffic flow management problem: An integer optimization approach. In *13th International Conference on Integer Programming and Combinatorial Optimization, IPCO 2008 Bertinoro*, volume 5035, pages 34–46, May 2008.

[5] P. Bose, L. Devroye, M. Löffler, J. Snoeyink, and V. Verma. The spanning ratio of the Delaunay triangulation is greater than $\pi/2$. In *Proc. 21th Canadian Conference on Computational Geometry*, 2009.

[6] P. Bose, A. Maheshwari, G. Narasimhan, M. Smid, and N. Zeh. Approximating geometric bottleneck shortest paths. *Computational Geometry*, 29(3):233–249, Nov. 2004.

[7] J. Canny and J. H. Reif. New lower bound techniques for robot motion planning problems. In *Proc. 28th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 49–60, 1987.

[8] W. N. Chan, M. Refai, and R. DeLaura. An approach to verify a model for translating convective weather information to air traffic management impact. In *7th AIAA Aviation Technology, Integration and Operations Conference (ATIO)*, Belfast, Northern Ireland, Sept. 2007.

[9] F. Y. Chin, J. Snoeyink, and C. A. Wang. Finding the medial axis of a simple polygon in linear time. In *ISAAC '95: Proceedings of the 6th International Symposium on Algorithms and Computation*, pages 382–391, London, UK, 1995. Springer-Verlag.

[10] A. Clementi, P. Crescenzi, and G. Rossi. On the complexity of approximating colored-graph problems extended abstract. In *Proceedings of the 5th annual international conference on Computing and combinatorics*, pages 281–290. Springer-Verlag, 1999.

[11] A. D'Ariano, D. Pacciarelli, and M. Pranzo. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, 183(2):643–657, December 2007.

[12] W. F. de la Vega and G. Lueker. Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.

[13] R. DeLaura and S. Allan. Route selection decision support in convective weather: A case study of the effects of weather and operational assumptions on departure throughput. In *5th USA/Europe ATM R&D Seminar*, June 2003.

[14] D. P. Dobkin, S. J. Friedman, and K. J. Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete Comput. Geom.*, 5(4):399–407, 1990.

[15] C. Duncan, A. Efrat, S. Kobourov, and C. Wenk. Drawing with fat edges. In *Lecture Notes in Computer Science*, pages 162–177. Springer Berlin, 2002.

[16] M. Farshi. *A Theoretical and Experimental Study of Geometric Networks*. PhD thesis, Eindhoven University of Technology, 2008.

[17] J. M. Framinan and C. Schuster. An enhanced timetabling procedure for the no-wait job shop problem: a complete local search approach. *Comput. Oper. Res.*, 33(5):1200–1213, 2006.

[18] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[19] J. Hastad. Some optimal inapproximability results. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 1–10, El Paso, Texas, United States, 1997. ACM.

[20] V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Inf. Process. Lett.*, 37(1):27–35, 1991.

[21] D. Karger, C. Stein, and J. Wein. Scheduling algorithms. *CRC Handbook of Computer Science*, 1997.

[22] J. M. Keil and C. A. Gutwin. Classes of graphs which approximate the complete euclidean graph. *Discrete Comput. Geom.*, 7(1):13–28, 1992.

[23] J. Kim, Z. Jingyu, J. Krozel, and J. S. B. Mitchell. Sensitivity of capacity estimates given convective weather constraints. In *AIAA Guidance, Navigation, and Control Conference*, Chicago, Illinois, United States, Aug. 2009.

[24] J. Kim, A. Kroeller, J. S. B. Mitchell, and G. Sabhnani. Scheduling aircraft to reduce controller workload. In *Proceedings of the 9th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS 09)*, 2009.

[25] J. Kim, J. S. B. Mitchell, and J. Zou. Approximating maximum flow in polygonal domains using spanners. In *Proceedings of the 21st Canadian Conference on Computational Geometry (CCCG2009)*, page 115118, UBC, Vancouver, Canada, Aug. 2009.

[26] J. Kim, J. S. B. Mitchell, and J. Zou. Routing parallel and merging lanes. In *The 19th Fall Workshop on Computational Geometry*, Medford, MA, November 2009.

[27] J. Krozel, W. M. McNichols, J. Prete, and T. Lindholm. Causality analysis for aviation weather hazards. In *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, Anchorage, AK, Sept. 2008.

[28] J. Krozel, J. S. B. Mitchell, V. Polishchuk, and J. Prete. Airspace capacity estimation with convective weather constraints. In *AIAA Guidance, Navigation, and Control Conference*, Aug 2007.

[29] J. Krozel, J. S. B. Mitchell, P. Valentin, and J. Prete. Maximum flow rates for capacity estimation in level flight with convective weather constraints. *Air Traffic Control Quarterly*, 15(3):209–238, 2007.

[30] J. Krozel and J. T. J. Murphy. Weather hazard requirements for NGATS aircraft. In *Integrated Communications, Navigation, and Surveillance Conf.*, May 2007.

[31] J. Krozel, J. Prete, J. S. B. Mitchell, J. Kim, and J. Zou. Capacity estimation for super-dense operations. In *AIAA Guidance, Navigation, and Control Conf.*, Aug 2008.

[32] P. Laroche. Planar 1-in-3 satisfiability is NP-complete. *Comptes rendus de l'Academie des sciences. Serie II. Fascicule a, Sciences de la terre et des planetes*, 316(4):389, 1993.

[33] P. M. Lennartz. *No-Wait Job Shop Scheduling, a Constraint Propagation Approach.* PhD thesis, UU Universiteit Utrecht, Netherlands, 2006.

[34] B. Martin. Model estimates of traffic reduction in storm impacted en route airspace. In *AIAA Aviation, Technology, Integration, and Operations Conf.*, 2007.

[35] A. Mascis and D. Pacciarelli. Job shop scheduling with blocking and no-wait constraints. *Eur J. Oper. Res.*, 142:498–517, 2002.

[36] J. S. B. Mitchell. On maximum flows in polyhedral domains. *Journal of Computer and System Sciences*, 40(1):88–123, 1990.

[37] J. S. B. Mitchell and V. Polishchuk. Thick non-crossing paths and minimum-cost flows in polygonal domains. In *Proceedings 23rd ACM Symposium on Computational Geometry*, pages 56–65, 2007.

[38] J. S. B. Mitchell, V. Polishchuk, and J. Krozel. Airspace throughput analysis considering stochastic weather. In *AIAA Guidance, Navigation, and Control Conf*, 2006.

[39] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 229–234, Chicago, Illinois, United States, 1988. ACM.

[40] J. Planning and D. Office. Next generation air transportation system: Weather concept of operations, version 1.0. Technical report, Joint Planning and Development Offices, 2006.

[41] J. Planning and D. Office. Concept of operations for the next generation air transportation system, version 2.0. Technical report, Joint Planning and Development Office, 2007.

[42] V. Polishchuk. *Thick Non-Crossing Paths and Minimum-Cost Continuous Flows in Geometric Domains*. PhD thesis, Stony Brook University, Aug 2007. Available at `http://cs.helsinki.fi/~polishch/pages/thesis.pdf`.

[43] V. Polishchuk and J. S. B. Mitchell. Thick non-crossing paths and minimum-cost flows in polygonal domains. In *Proceedings of the twenty-third annual symposium on Computational geometry*, pages 56–65, Gyeongju, South Korea, 2007. ACM.

[44] J. Prete. *Aircraft Routing in the Presence of Hazardous Weather.* Ph.D dissertation, Stony Brook University, NY, Aug. 2007.

[45] J. Prete, J. Krozel, J. S. B. Mitchell, J. Kim, and J. Zou. Flexible, performance-based route planning for Super-Dense operations. In *AIAA Guidance, Navigation, and Control Conference*, Aug. 2008.

[46] J. Prete and J. S. B. Mitchell. Safe routing of multiple aircraft flows in the presence of Time-Varying weather data. In *AIAA Guidance, Navigation, and Control Conf.*, 2004.

[47] D. Rhoda and M. Pawlak. An assessment of thunderstorm penetrations and deviations by commercial aircraft in the terminal area. Technical Report MIT Lincoln Laboratory Tech. Report NAS/A-2, Lincoln Laboratory, MIT, June 1999.

[48] D. Rhoda and M. Pawlak. The thunderstorm penetration / deviation decision in the terminal area. In *American Meteorological Society, 8th Conf. on Aviation, Range, and Aerospace Meteorology*, 1999.

[49] C. J. Schuster. No-wait job shop scheduling: Tabu search and complexity of subproblems. *Mathematical Methods of Operations Research*, 63(3):473–491, July 2006.

[50] C. J. Schuster and J. Framinan. Approximative procedures for no-wait job shop scheduling. *Oper Res Lett*, 31:308–318, 2003.

[51] L. Song, D. Greenbaum, and C. Wanke. The impact of severe weather on sector capacity. In *8th USA/Europe Air Traffic Management R&D Seminar*, July 2009.

[52] L. Song, C. Wanke, and D. Greenbaum. Predicting sector capacity for TFM decision support. In *6th AIAA Technology, Integration, and Operations Conf.*, 2006.

[53] L. Song, C. Wanke, D. Greenbaum, and D. Callner. Predicting sector capacity under severe weather impact for traffic flow management. In *Aviation Technology, Integration, and Operations Forum*, 2007.

[54] L. Song, C. Wanke, D. Greenbaum, S. Zobell, and C. Jackson. Methodologies for estimating the impact of severe weather on airspace capacity. In *26th Intern. Congress of the Aeronautical Sciences*, 2008.

[55] C. Souders, S. McGettigan, E. Dash, and J. May. Weather integration concept of operations for transforming the national airspace system. In *12th Conf. on Aviation, Range, and Aerospace Meteorology (ARAM)*, 2006.

[56] G. Strang. Maximal flow through a domain. *Mathematical Programming*, 26(2):123–143, June 1983.

[57] H. Swenson, R. Barhydt, and M. Landis. Next generation air transportation system (NGATS) air traffic management (ATM)-Airspace project. NASA tech. report, version 6.0, NASA Ames Research Center, 2006.

[58] M. van Kreveld. Bold graph drawings. In *Proceedings of the 21st Canadian Conference on Computational Geometry (CCCG2009)*, pages 119–122, 2009.

[59] C. Wanke, M. Collaham, D. Greenbaum, and A. Masalonis. Measuring uncertainty in airspace demand predictions for traffic flow management applications. In *AIAA Guidance, Navigation, and Control Conf.*, 2003.

[60] G. J. Woeginger. Inapproximability results for no-wait job shop scheduling. *Oper. Res. Lett.*, 32:320–325, 2004.

[61] J. Zou. *Geometric Algorithms for Capacity Estimation and Routing in Air Traffic Management.* PhD thesis, State University of New York at Stony Brook, 2010.