UNIVERSITY OF CALIFORNIA,
IRVINE

Strategic Air Traffic Planning Using Eulerian Route Based Modeling and Optimization

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Mechanical and Aerospace Engineering

by

Alessandro Bombelli

2017

DEDICATION

To Elisa, my parents, my grandma Nonna Franca, and my extended family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Thank you Professor Mease for your patience during these years. Due to our lack of expertise in Air Traffic Management, the learning curve has been very steep. As a consequence, you tried to focus on a more general research aspect, i.e., the ability to always find the "why" of research. This is something I still need to improve, but I really appreciated your academic rigor and efforts to make me a better researcher.
Thanks to my committee members Professor Tryphon Georgiou and Professor Solmaz Kia, for taking the time to go through my work and provide me with useful feedbacks.
Thank you to Lluis Soler Segura, who started the project and has been a good office mate and fierce opponent when playing racquetball. Thanks to Adrià Segarra Torné and Arnau Francí Rodon, who worked on different parts of this work with me. The quality of the work would have been worse without your ideas. The biggest thank you goes to Eric Trumbauer, my unofficial advisor and a very funny labmate. Also, thanks to all the labmates who were not directly involved in this work, for creating a fun working environment.
Thank you to University Affiliated Research Center for partially supporting this work under Prime Contract Number NAS2-03144 awarded to the University of California, Santa Cruz, University Affiliated Research Center by NASA. A big thank to all the people there, for making me feel at home during my internships at NASA Ames. Esther, Angela and Debbie, thank you for being such nice ladies. Thank you to everybody in the Aviation Systems Division at NASA Ames for being an incredible source of expertise. In particular, thank you Heather and Tony for being awesome mentors and such easygoing people as well. I hope our paths will cross again in the future.
Thank you to all my friends from Società Sportiva Calcio San Morosini. You made me rediscover the passion for played football (I never lost the passion for watched football).
Thank you to all my international friends from UCI. The friends from the Intramural teams, for all the fun games we played together. Elisa's labmates, who adopted me as a lab member. I enjoyed our basketball games, even if we started playing right before I left. A special thank to Omid for being a true friend and politely crashing our wedding.
Thank you to all my Italian friends from UCI. Marco, for introducing me to the life within UCI. Michele, for being an amazing housemate with some issues with the toilet brush. Minno, for being the most proactive and adventurous among us. Biasio, for living the American dream like nobody else. Paolo, for being the cool office mate Marco has never been able to be (judgement based only on the time spent in the office!). Enrico, for being my best friend here, and for all the tough times we have been through together.
Thank you Anto for being an awesome friend since we were fourteen, and for visiting us in the United States more often than anybody else.
Thank you to Caterina, Nino, Anna, Mauri, and the latest member of the family, Alice, for being the best second family I could possibly ask for. For the same reason, thank you to the Sicilian branch of the family as well.
Thank you mom, dad and nonna Franca for being constantly overconcerned although my teenage years are way over, but for always being there for me when needed.
Thank you Elisa for having been my sweetheart and best friend all these years. We will keep getting better together as time goes by, like a good red wine.

CURRICULUM VITAE

Alessandro Bombelli

EDUCATION

Doctor of Philosophy in Aerospace Engineering	**2017**
University of California Irvine	*Irvine, CA*
Master of Science in Space Engineering	**2012**
Polytechnic University of Turin	*Turin, Italy*
Master of Science in Space Engineering	**2012**
Polytechnic University of Milan	*Milan, Italy*
Bachelor of Science in Aerospace Engineering	**2009**
Polytechnic University of Milan	*Milan, Italy*

RESEARCH EXPERIENCE

Graduate Research Assistant	**2012–2017**
University of California, Irvine	*Irvine, California*

TEACHING EXPERIENCE

Teaching Assistant	**2013–2017**
University of California, Irvine	*Irvine, California*

REFEREED JOURNAL PUBLICATIONS

Strategic Air Traffic Planning with Fréchet Distance Aggregation and Rerouting **2017**
Journal of Guidance, Control, and Dynamics

REFEREED CONFERENCE PUBLICATIONS

Analysis of Convective Weather Impact on Pre-Departure Routing of Flights from Fort Worth Center to New York Center **2017**
AIAA AVIATION Conference

Automated Route Clustering for Air Traffic Modeling **2017**
AIAA SCITECH Conference

Strategic Air Traffic Planning with Fréchet Distance Aggregation and Rerouting **2015**
AIAA AVIATION Conference

Debris Removal Mechanism Based on Tethered Nets **2012**
International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)

Optimal Design of a Net-Shaped Space Debris Removal System **2012**
5th International Conference on Astrodynamics Tools and Techniques (ICATT)

ABSTRACT OF THE DISSERTATION

Strategic Air Traffic Planning Using Eulerian Route Based Modeling and Optimization

By

Alessandro Bombelli

Doctor of Philosophy in Mechanical and Aerospace Engineering

University of California, Irvine, 2017

Professor Kenneth D. Mease, Chair

Due to a soaring air travel growth in the last decades, air traffic management has become increasingly challenging. As a consequence, planning tools are being devised to help human decision-makers achieve a better management of air traffic. Planning tools are divided into two categories, strategic and tactical. Strategic planning generally addresses a larger planning domain and is performed days to hours in advance. Tactical planning is more localized and is performed hours to minutes in advance. An aggregate route model for strategic air traffic flow management is presented. It is an Eulerian model, describing the flow between cells of unidirectional point-to-point routes. Aggregate routes are created from flight trajectory data based on similarity measures. Spatial similarity is determined using the Fréchet distance. The aggregate routes approximate actual well-traveled traffic patterns. By specifying the model resolution, an appropriate balance between model accuracy and model dimension can be achieved. For a particular planning horizon, during which weather is expected to restrict the flow, a procedure for designing airborne reroutes and augmenting the traffic flow model is developed. The dynamics of the traffic flow on the resulting network take the form of a discrete-time, linear time-invariant system. The traffic flow controls are ground holding, pre-departure rerouting and airborne rerouting. Strategic planning - determining how the controls should be used to modify the future traffic flow when local capacity violations are anticipated - is posed as an integer programming problem of minimizing a weighted sum of flight delays subject to control and capacity constraints. Several tests indicate the effectiveness of the modeling and strategic planning approach. In the final, most challenging, test, strategic planning is demonstrated for the six western-most Centers of the 22-Center national airspace. The planning time horizon is four hours long, and there is weather predicted that causes significant delays to the scheduled flights. Airborne reroute options are computed and added to the route model, and it is shown that the predicted delays can be significantly reduced. The test results also indicate the computational feasibility of the approach for a planning problem of this size.

Chapter 1

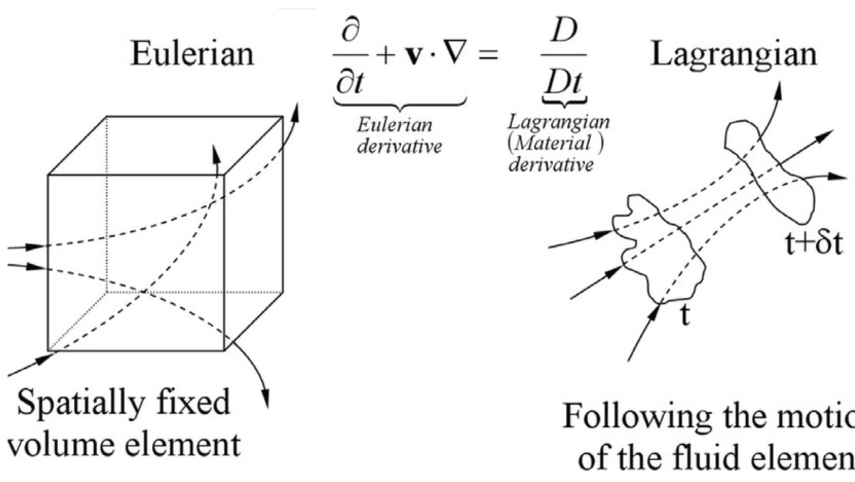
Introduction

With the growing number of aircraft in the National Airspace System (NAS), Air Traffic Management (ATM) becomes increasingly challenging. Research over the past decade and a half has been directed at devising support tools to assist human decision-makers. ATM is typically separated hierarchically into strategic planning of traffic flow in the NAS with a time horizon of 2-8 hours, and tactical control which is more local in both space and time. There is also a third lower control authority, i.e., aircraft. The hierarchical structure is described in Table 1.1, with a focus on the different decision makers, the spatial domain, the time horizon and control variables of each layer.

Table 1.1: Hierarchical structure of ATM.

ATM Hierarchy	Decision Maker	Spatial Domain	Time Horizon	Control Variables
Strategic	Committee: FAA, ATCs, Airlines	NAS	2-8 h ahead	departure schedule/routes; airborne reroutes
Tactical	ATCs	Sectors	1 h ahead	holding patterns, speed, reroutes
Aircraft	Pilot	Local airspace	minutes	maneuvers

The role of strategic planning is to anticipate airspace demand exceeding the capacity of airports and sectors of the NAS, and determine how the air traffic flow should be adjusted nationally or regionally to mitigate the situation [8]. Two fundamentally distinct modeling



Eulerian
$\underbrace{\frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla}_{\text{Eulerian derivative}} = \underbrace{\frac{D}{Dt}}_{\text{Lagrangian (Material) derivative}}$
Lagrangian
t+δt
t
Spatially fixed volume element
Following the motion of the fluid element

Figure 1.1: Lagrangian and Eulerian approach to track aircraft (particles).

approaches derived from fluid mechanics have been pursued for strategic Traffic Flow Management (TFM): (i) the Lagrangian approach of tracking each aircraft (particle) in the flow and (ii) the Eulerian approach of tracking the distribution of aircraft over spatially fixed cells in the airspace. Figure 1.1 depicts how the two different approaches keep track of the temporal spatial evolution of particles (aircraft in the ATM context).

In [8, 9], a Lagrangian model is presented that optimizes the traffic flow of a set of flights in the NAS. The problem is formulated as a 0-1 Integer Programming (IP) problem, where decision variables determine the sequence of sectors a flight transverses from origin to destination. Sectors are local units that divide the NAS, where each sector is controlled by an Air Traffic Controller (ATC) and can accommodate a limited amount of aircraft at the same time. The objective of the Traffic Flow Management Problem (TFMP) is to decide how long each flight is going to be held on the ground and in the air in order to minimize the total delay cost. Capacity constraints are imposed on departures from and arrivals to airports [34, 80], and on sector capacity. Since each flight is assigned a sequence of sectors that are crossed from origin to destination, the granularity of the problem is directly connected to the size of the different sectors. If the number of aircraft within each sector is less than the upper bound, it is assumed that ATCs will be able to handle the air traffic flow.

Each flight is not characterized by a flight plan, but by a sequence of sectors, which might be a too coarse approximation even in a strategic planning framework.

In [84], another Lagrangian model is presented that addresses the air traffic planning problem from a graph-oriented perspective. In the graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, the set of nodes $\mathcal{N}$ is a set of waypoints (latitude-longitude-altitude triplets) in the NAS, while the set of edges $\mathcal{E}$ is a set of directed airways connecting waypoints. The graph-oriented framework is chosen to address one of the issues identified in [8, 60, 80]. Those models only determine ground delays or reroutes expresses as different sequences of sectors, rather than designing the entire flight path. Modifications to flight plans to better utilize airspace resources are not explored since the granularity of these models is bounded to the size of sectors, and does not account for actual flight plans. In [84], a decentralized problem is solved that is based on a priority queue where all the flights of interest are stored. Authors suggest a First-Come, First-Served policy. Once the path planning problem for the current flight is solved, the sector capacity of sectors occupied by the aircraft is decreased by one unit. If, for a certain time instance, a sector reaches full capacity, that sector becomes a no-fly zone for all the remaining flights in the queue for that time instance.

References [8, 9, 60, 80, 84] all identify complexity and scalability as issues. An increase in the expected air traffic flow might turn the TFMP from tractable to intractable. In [9], a very accurate analysis of the size of the formulation is provided. In particular, it is stated that if the number of flights in the formulation doubles, the number of variables doubles and the number of constraints nearly doubles.

Reduced model complexity and scalability has been one of the main drivers for the development of Eulerian models, which focus on the computation of a graph-oriented structure that defines the dynamics of aircraft. In Lagrangian models, each aircraft is tracked singularly. In Eulerian models, the concept of single aircraft is lost, and the state of the model is the number of aircraft in each node of the graph-oriented structure.

The Eulerian approach for traffic control was inspired by the Lighthill-Whitham-Richards models [59] and the cell-transmission model [23] for ground transportation. In this work, we develop an Eulerian model for strategic TFM called the Aggregate Route Model (ARM). A challenge for Eulerian modeling is to determine the minimum level of resolution, i.e., the size and location of the units of discretization, that allows effective strategic planning.

Eulerian modeling of air traffic began with the work in [56, 57, 70]. In [70], 23 air traffic centers (one for international flights) are the basic units of discretization, resulting in a Linear Time-Varying (LTV) flow model with 23 state variables. In [57], the airspace is covered with a grid of 2D cells, each cell traversed by 8 paths; this results in a LTV model where the state dimension is 8 times the number of cells. Rather than cover a 3D or even 2D airspace with cells, several previously developed models [16, 53, 72] and the model developed in this work account for aircraft in segments of one-way aggregate routes. Since aircraft follow similar paths for portions of their flight, these paths may be aggregated together into a reduced number of routes. Modeling flow along aggregate routes limits the airspace that has to be considered and can reduce the model dimension without sacrificing accuracy.

The route aggregation method used in this work is distinctive relative to previous one-way route type models [53, 72]. In [72], for each air sector modeled, a straight line aggregate link is defined for common entry/exit paths connecting neighboring sectors. Thus, potential distinct patterns from the same upstream sector to the same downstream sector cannot be detected inside the current sector. In [53], both k-means and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [52] algorithms are used to determine aggregate routes, but a constant speed is assumed everywhere. Here, we use the Fréchet distance [4, 5, 18] as the spatial similarity measure for aggregating routes. It is a distance that is particularly appropriate when comparing flight trajectories. (i) It does not depend on the overall length of the trajectories. This is in contrast to additive metrics that make it difficult to compare trajectories with considerably different lengths. (ii) It does not allow backtracking, which is a natural requirement when considering aircraft trajectories. To the best of our knowledge, the Fréchet distance has been applied in the literature to planar cases [4, 5, 18], or convex polyhedra [51] (i.e., intersections of planes). We adapt the Fréchet distance to compare great circle arc sequences on a spherical surface; for our application these paths are projections of the actual flight tracks.

We also have a strategy to guarantee trajectories belonging to the same cluster have similar dynamical properties, and use the average ground speed as the driver. For our graph-oriented approximation of the air traffic flow, we need to detect distinct spatial patterns, but also distinct dynamical patterns (if any) within the same spatial pattern. Grouping together trajectories that are spatially similar, but dynamically dissimilar would create an aggregate route that is not representative of the modeled air traffic flow.

To construct an aggregate route model, we first identify well-traveled routes by clustering historical flight data in a planning domain of interest. The planning domain is a connected subset of the NAS. For each cluster, we then compute an aggregate route where aircraft can move. This modeling strategy is primarily Eulerian, but captures some Lagrangian features. In [72] the modeling accounts for air traffic in high sectors using aggregate routes with constant speed inside each sector. In [53] the aggregate routes extend from origin to destination with constant speed throughout. Although our aggregate routes are divided into segments (cells) with constant transit time, the segment lengths can vary and thus the speed in each segment can be different. Additionally, there may be multiple segments in a sector. We also consider flight paths from take off to landing, which allows us to account directly for ground holding and pre-departure rerouting as the strategic air traffic controls that are naturally available once the Eulerian model is built, another distinction relative to the modeling in [16, 53, 72]. Our extensions to the Eulerian approach have counterparts in the Lagrangian model in [8]. Relative to the Lagrangian approach in [8], our approach has the potential for resolving the flow more accurately with a lower order model.

The identification of well-traveled routes is the first part of a two-step process. The goal of the first step is to define the basic structure of the network flow model, which characterizes the planning domain regardless of the specific weather scenario. In the second part, when bad weather is reducing capacity in the planning domain, additional routes are computed that are specifically designed for the weather situation. Other Lagrangian [84] and Eulerian models [53] also describe techniques to design reroutes when bad weather is affecting the planning domain. At a more tactical level, research has been done [7, 43, 64] that addresses the issue of designing efficient reroutes avoiding obstacles that might impend the planned route. References [7, 43] and [74, 75, 76] emphasize the operational feasibility of reroutes from different perspectives. In [7, 43], operational feasibility is addressed imposing dynamical constraints on the turning radius of a rerouting maneuver. In [74, 75, 76], operational feasibility is assessed by determining how often a certain connection between two waypoints has been historically flown. The nature of our model makes operational feasibility, as stated in [7, 43], a more natural fit, keeping in mind the strategic and thus coarser level of precision we can guarantee.

In the second step of the two-step process, we (i) detect if some of the scheduled departures are expected to experience significant ground delays because of severe weather, and (ii) design airborne reroutes that avoid the weather fronts. These reroutes are needed for strategic planning. The other alternative is to increase the dataset with well-traveled routes by adding all possible weather situations, or possibly finding particular days that had similar weather and building an ARM for the particular situation. The first option would increase the model dimension more than what we are doing. The way we do it is more tailored to the specific situation, including departure schedules.

Using the Eulerian model, the strategic planning problem is formulated as a constrained optimization problem. A weighted sum of the flight delays is minimized subject to airport and sector capacity constraints. The air traffic flow dynamics are discrete-time and linear time-invariant, and the optimization problem is posed as an IP problem. The cost function has the following features: the delays associated with the ground holding, pre-departure rerouting and airborne rerouting control actions are included; the cost is zero if and only if the scheduled flight plan does not violate capacity constraints; and the weighting factors favor control actions later in the planning horizon when possible. The latter feature limits early actions based on uncertain forecasts and buys time for replanning based on updated information. Given the Eulerian framework of the model, the allocation of the flow directives from the strategic planning to specific aircraft is left to other decision makers.

The constrained optimization problem is integer, since we controls are applied to aircraft and not portions of aircraft. The solution of an IP problem is computationally more demanding than the solution of the associated relaxed Linear Programming (LP) problem. Under some specific circumstances, an IP problem can be solved with a LP solver, and the solution is guaranteed to be integer. In particular, if the inequality matrix of the LP problem is unimodular [81] and the simplex method is used, the solution is guaranteed to be integer. Reference [9] shows that, although the inequality matrix is not guaranteed to be totally unimodular, solutions obtained by solving LP relaxations of the original IP problem are integral in most of the cases. In this work, the structure of the inequality matrix is such that unimodularity cannot be guaranteed. We will demonstrate that solving the problem is still tractable with IP techniques.

The work is organized as follows. Chapter 2 describes how a historical dataset of flown trajectories is processed and translated into an Eulerian model of the air traffic flow. Chapter 3 treats the airborne rerouting problem and shows how it fits in with the Eulerian model previously introduced. Chapter 4 describes the dynamics of the uncontrolled model, introduces controls and shows how they modify the dynamics, while Chap. 5 describes how the IP problem is posed and solved. Chapter 6 shows how the ARM is tested on scenarios of different sizes and complexity. Conclusions are stated in Chap. 7.

Chapter 2

Aggregate Routes via Clustering

To determine the aggregate routes that will comprise the base traffic flow model, we cluster trajectory data for clear weather days to identify the well-traveled routes. This chapter presents the work reported in [12, 13, 14] and extensions of that work. In the next chapter, we describe how additional routes are added to accommodate convective weather for a particular day.

For clear weather days, flight trajectories are generally characterized by smooth and well-defined patterns. Tactical maneuvers such as holding patters and path stretches, can still characterize a clear weather day, as a consequence of some operational constraints ATCs must satisfy. It is important to have a systematic method of identifying and discarding trajectories that are unusual in this sense; for this purpose an outlier detection approach is presented.

Clustering techniques can be found in Refs. [12, 19, 53, 58, 63, 72]. In [72], routes are initially divided into O-D (origin-destination) pairs. Then, routes that share the same sequence of sectors are clustered. In each sector, portions of routes are approximated with a segment connecting the two centroids associated with the entry and exit points from the specific sector. This approach lacks flexibility, especially if airspace redesign [82] is taken into account. As [58] points out, air traffic patterns can be analyzed to suggest modifications of the current sector partitioning scheme. A clustering algorithm that does not constrain patterns according to sector sequences is invariant with respect to a re-sectorization.

In [53], flight tracks are clustered using Principal Component Analysis and DBSCAN [52]. The clustering technique adopted does not characterize routes according to speed. In fact, in the resulting graph all aircraft move at a uniform constant speed. This feature might fail to catch dynamical differences between routes.

In [58], trajectories are initially resampled to display the same number of features (i.e., latitude-longitude pairs), and then clustered using DBSCAN [52]. The aim of the work is the identification of spatial patterns to assess airspace occupancy, and thus the inability to detect speed properties due to the resampling is not a major issue. In a context where also speed properties should be considered, this approach would not be applicable.

In [63], a weighted sum of the pairwise distances between flight tracks (intended as latitude-longitude-altitude triplets) of two different flight routes is used as measure of dissimilarity. The weight is the inverse of the number of the accumulated flight tracks, in order not to penalize longer routes. Since the two routes can have different lengths, it is suggested to begin with the last track, and then to go backwards to the first track of the shorter of the two routes. Although this method seems applicable to scenarios where trajectories are similar in length, it does not seem applicable to more general scenarios.

We address the limitations in the methods in [53, 58, 63, 72] to achieve a clustering algorithm that (i) detects air traffic flow patterns without using the domain sectorization as input, (ii) efficiently computes dissimilarity between routes with a different length, and (iii) provides a physical interpretation of the clustering parameters. We use the Fréchet distance [4, 5, 18, 51] between two trajectories to quantify their spatial similarity and hierarchical clustering to select the appropriate number of clusters. In comparison to the approach in [13], we add the following features: (i) coarse clustering to avoid the unnecessary computation of distances between tracks, (ii) flight track undersampling to reduce the computational time of each distance computation without loss in accuracy, and (iii) outlier detection to remove trajectories that are not representative of well-traveled routes. Additionally, we provide bounds on the inflow/outflow that each airport can handle with a geometric data-driven interpretation of the departure and arrival capacity inspired by [34].

The combination of the distance metric and the clustering method, the outlier detection step, and the automation in the choice of clusters makes the routine applicable to scenarios of different sizes and type, with little manual tuning.

The chapter is organized as follows. Section. 2.1 defines the general framework, describes the dataset we process to cluster flight tracks and the different flight categories we consider, and states some modeling assumptions. Section 2.2 introduces the concept of coarse clusters, and describes how they are computed according to the specific flight category. Section 2.3 describes how outliers are identified and labeled. Section 2.4 highlights how fine clustering determines the optimal number of fine clusters for each coarse cluster, while Sec. 2.5 shows how a fine cluster of routes is translated into an aggregate route representative of the cluster. Section 2.6 describes how departure and arrival capacity is estimated for each airport of interest. Section 2.7 shows an application example.

2.1 General Framework, Historical Dataset and Flight Categories for Route Clustering

The automated clustering algorithm we present is summarized by the block diagram shown in Fig. 2.1. The input is a historical dataset of recorded aircraft trajectories inside the NAS.

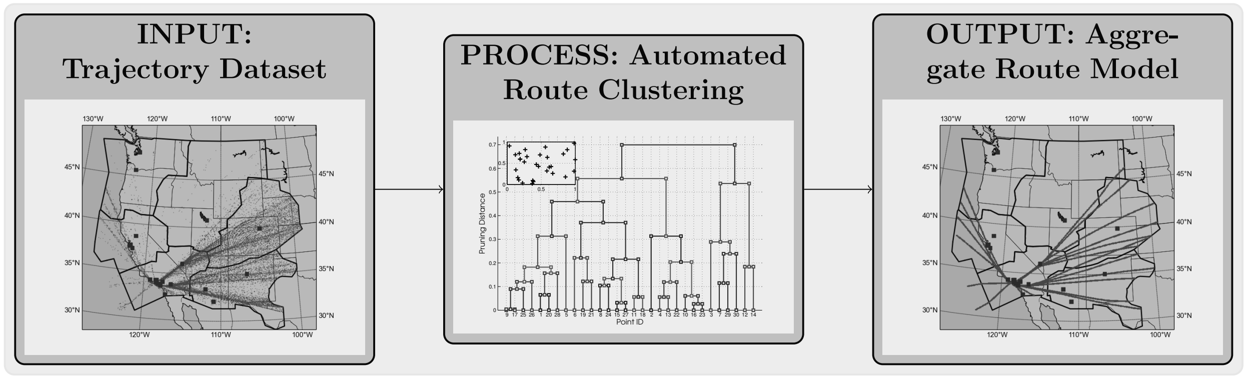


Figure 2.1: Route clustering as a process.

The dataset is processed with the automated route clustering routine, and the output is a set of aggregate routes, i.e., a graph-oriented approximation of the original dataset. In this section, the input is described. The core of the chapter, i.e., Sec. 2.2, 2.3, 2.4 shows how the input is processed. Section 2.5 focuses on the output instead.

Our historical dataset is a set of Future ATM Concept Evaluation Tool (FACET) Track (TRX) files [10]. For each day (from 00:00 to 23:59 Coordinated Universal Time (UTC)), a TRX file contains all information of aircraft flying inside the NAS, with a temporal resolution of 1 minute. The TRX file is divided into 1,440 blocks, one for every minute in a day. Each block starts with a time stamp, in unix time[1], and then contains information of all aircraft recorded inside the NAS at that time. For each aircraft, information is stored as follows

```
TRACK SCX135 B738 350313 1143050 249 36 283 ZLA ZLA06 160
FP_ROUTE KHRL./.DRK103025..KIFP
```

where the first line contains the aircraft identifier, the aircraft type, latitude, longitude, ground speed, heading, altitude, Center, sector, and flight level[2]. The second line is the filed flight plan.

[1]https://en.wikipedia.org/wiki/Unix_time
[2]https://en.wikipedia.org/wiki/Flight_level

The dataset is first rearranged into a structure that stores the full temporal history of each flight. We address strategic planning for a portion of the NAS, referred to as the planning domain D, and for a time period referred to as the planning horizon. The 3-dimensional planning domain D is a subset of the NAS and is defined as follows. Let A be a connected set on the Earth's surface which may be parameterized by latitude and longitude. A will be taken as the projection of a Center or a union of neighboring Centers onto the Earth's surface, i.e., their footprint. A natural selection could be the union of some of the 20 Centers forming the continental portion of the NAS. The planning domain D is the airspace above the set A from ground level up to some maximum altitude. Inside D, we only consider airports belonging to the Aviation System Performance Metrics (ASPM)[3] ASPM-77 list. This is a list containing the airports that account for the majority of the overall air traffic frow.

Flight trajectories are then divided into subsets, according to the flight category. We define four flight categories. Refer to Fig. 2.2 for examples.

- Internal flights: Flights between two airports, with the airports and the entire flight path in the domain. Examples are flight F3 from origin airport A1 to destination airport A2 and F4 from A2 to A3, but not F6 because its flight path leaves and re-enters the domain.

- Exiting flights: Flights with origin airport in the domain, destination airport outside the domain, and no re-entry into the domain after the first exit. An example is flight F2 from airport A1 to domain boundary point B2 to destination airport A6. From the planning domain perspective B2 is considered the destination for F2.

- Entering flights: Flights with destination airport in the domain and origin airport outside the domain. An example is flight F1 from origin airport A7 to boundary point

[3] `http://aspmhelp.faa.gov/index.php/ASPM_Airports`

B1 to destination airport A1. From the planning domain perspective, B1 is considered the origin for F1. If a flight with destination and origin airports as stated enters the domain more than once, the entry flight is defined from the last entry point and the earlier domain segments are discarded.

- Overflights: Flight paths crossing the domain with origin and destination airports outside the domain. An example is F5 with origin airport A10 and destination airport A11; from the planning domain perspective the origin is B6 and the destination is B7. If a flight with origin and destination outside the domain crosses the domain multiple times, each interior segment is accounted for as an overflight.

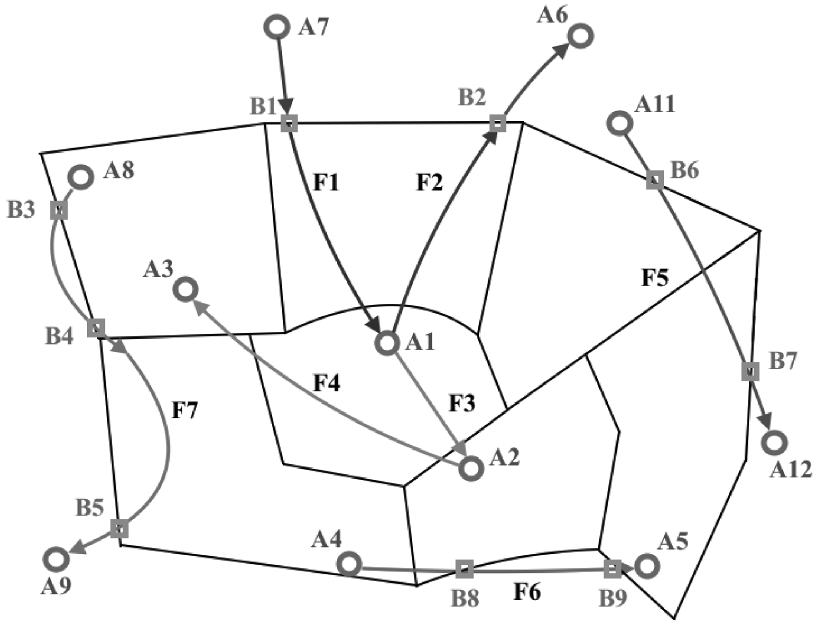


Figure 2.2: Schematic of planning domain composed of sectors and examples of flights from the four categories described in the text.

We also assume that we have a sufficient number of complete trajectories that partial trajectories can be discarded. A partial trajectory is a trajectory in any of the four categories that does not extend, within D, to its origin or destination points. For example, an internal flight is partial if either it was already airborne at the start of the dataset timespan or did not reach its destination by the end of the dataset timespan. Note that for internal flights, origin and destination points are the origin and destination airports, respectively. For the other flight categories, the origin and/or the destination point is identified as the boundary crossing point on the boundary of D, i.e., for each subset, trajectory points outside D are discarded.

We also discard flight trajectories with average cruise speeds less than 400 kts. This threshold is chosen to filter out short-haul regional airliners whose cruise speeds are significantly smaller than the cruise speeds of commercial airliners (such as the Boeing B737, the Airbus A320 and the McDonnell Douglas MD-80), and that minimally contribute to the overall air traffic flow.

For each airport in the planning domain, there are potentially three subsets of the trajectory dataset corresponding respectively to internal flights for which the airport is the origin, exiting flights for which the airport is the origin, and entering flights for which the airport is the destination. There is one additional subset for overflights. The overall number of subsets is at most $3n_a + 1$, where n_a is the number of airports inside D.

Note that the same flight trajectory might be categorized differently with a different planning domain selection, i.e., the distribution of flights into these subsets is dependent on the size of the planning domain. For example, some overflights for a smaller domain may become internal, exiting, or entering if the planning domain is expanded.

2.2 Coarse Clustering

The processing of the trajectory dataset to produce the aggregate route model consists of four main steps that are shown in a flowchart in Fig. 2.3. The clustering phase consists of three sequential steps: coarse clustering, outlier identification, and fine clustering. Then, the fourth step translates each fine cluster into an aggregate route. In this section, coarse

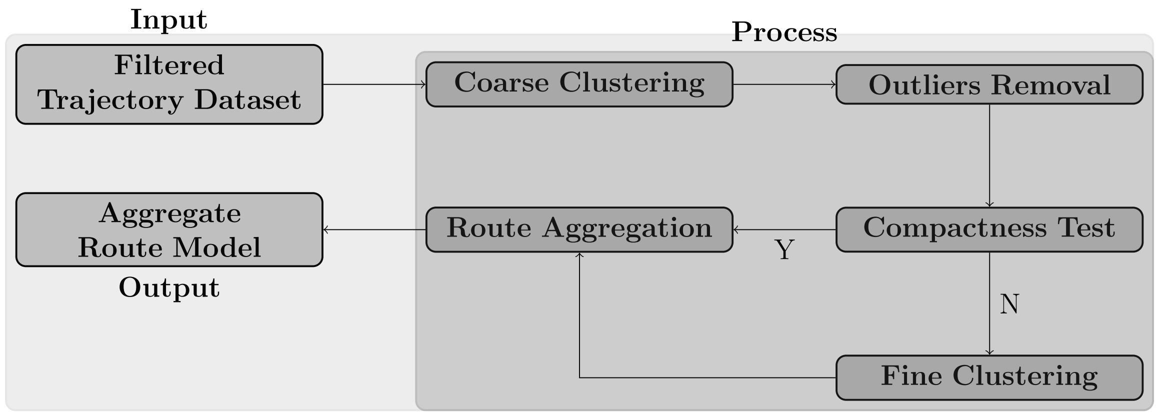


Figure 2.3: Flowchart describing the route clustering and aggregation process.

clustering will be described. The remaining clustering steps are discussed in Sec. 2.3 and Sec. 2.4 respectively.

The evaluation of distances between flight tracks (represented as sequences of latitude-longitude coordinates projected onto a sphere) is the computationally most demanding operation of the clustering process. The most precise clustering is based on the Fréchet distance between flight tracks, however computing the Fréchet distance is computationally costly. By first performing lower resolution (coarse) clustering, we reduce the number of Fréchet distances that need to be computed.

There are two steps in coarse clustering: (i) for internal flights, there is a coarse cluster for each O-D pair, while for entering, exiting, and overflights, there are clusters based on the planning domain boundary crossing points, and (ii) the clusters generated from (i) are subdivided if there are two or more sufficient distinct ranges of average ground speed.

For internal flights, there is straightforward coarse clustering based on the destination airport. With n_a airports in the domain, there are at most $n_a - 1$ destination airports and thus the internal flights for each origin airport are separated into at most $n_a - 1$ coarse clusters.

For the other three flight categories, the origin and/or the destination airports are outside the domain, thus an alternative approach is required. In such cases, we assess the spatial similarity of the boundary crossing points. The coarse clusters for the exiting flights from each airport are determined by clustering the boundary crossing points of all flights taking off from the airport and exiting the planning domain. Coarse clustering for entering flights is the same except that the entry boundary points are clustered for each destination airport. For overflights, we cluster boundary crossing points associated with entry points first, and then we cluster boundary crossing points associated with exit points of trajectories belonging to the same cluster of entry points. When clustering boundary crossing points, two distinct features are targeted. Clusters (i) should identify groups of boundary crossing points that are compact (small intracluster distances) and well-separated (high intercluster distances), and (ii) a cluster (even if compact and well-separated) should not have a maximum intracluster distance too large with respect to the size of D. An example provided later will better clarify this concept.

The boundary crossing point clustering process is as follows. Exit/entry points are projected onto a sphere, and the haversine [79] distance is selected as the distance metric. We select a sphere with a $35,000\ ft$ altitude above Earth's surface, because it is the average cruise altitude of most aircraft. We use the haversine distance, instead of the Euclidean distance, to account for the curvature of D and be consistent with the distance metric used in Sec. 2.4. Agglomerative hierarchical clustering is used to cluster points. In agglomerative hierarchical clustering, each element is initialized as a single cluster. Then a tree structure called a dendrogram is built, which links elements as the allowed intracluster distance is increased. When comparing elements at the lowest level of the dendrogram, the distance between the points is directly used. To assess the spatial similarity of two clusters, based on the set of distances between pairs of points, one in each cluster, previous works have used one of three intercluster distances [33]: complete-linkage, single-linkage, or average linkage. We use the average-linkage, which is a trade-off between intercluster distance allowed and number of clusters.

Given the same set of elements to be clustered, a different intercluster distance choice would generally result in a different dendrogram. Once a dendrogram is created, the number of clusters is obtained by pruning the dendrogram selecting a pruning distance. The pruning distance is a horizontal line that cuts the dendrogram in as many points as the number of clusters. For each cut, an isolated portion of the full dendrogram is obtained. All elements of the set that belong to the isolated portion define the resulting cluster.

An example of a dendrogram for clustering points is shown in Fig. 2.4. Thirty points shown in the top left corner of Fig. 2.4 are randomly generated inside a unit square in 2D Cartesian space, and all pairwise Euclidean distances are computed. Then, using average-linkage distance, the resulting dendrogram is shown in Fig. 2.4. Analyzing Fig. 2.4, pruning distances of 0.01, 0.5 or 0.8 will result in 30, 4 or 1 cluster respectively.

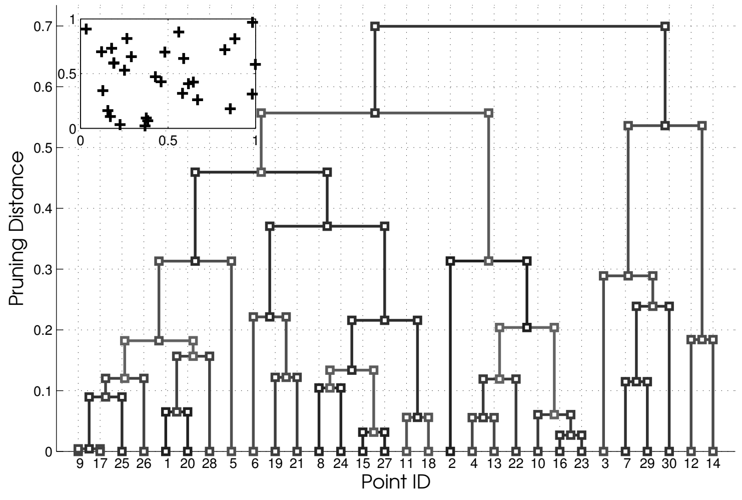


Figure 2.4: Example of a dendrogram applied to a distribution of points, using the Euclidean distance.

Note that there is always a range of pruning distances that result in the same number of clusters. As example, considering Fig. 2.4 again, any distance between 0.47 and 0.53 results in 4 clusters.

We now address the following issues: (i) how we select a range of pruning distances (and, as a consequence, of clusters), and (ii) given a range of clusters, how to choose the "best" number of clusters N_c.

The range of interest for N_c is defined by setting a lower and upper bound on the pruning distance. The upper bound sets the lowest number of clusters N_c^{min} (since the dendrogram is cut in fewer points), while the lower bound sets the highest number of clusters N_c^{max}. N_c will be searched within the interval defined by N_c^{min} and N_c^{max}. The lower bound and upper bound pruning distances are, respectively, $\eta_{cl} d_m$ and $\eta_{cu} d_m$. d_m is the maximum pairwise haversine distance of the set of exit/entry points, while $\eta_{cl} = 0.1$ and $\eta_{cl} = 0.25$. This choice of parameters, for the data processed, has proven effective to (i) identify clusters of boundary crossing points that are compact and well-separated, and (ii) sub-divide a cluster of boundary crossing points with an excessive maximum intracluster distance.

To decide the number of clusters N_c, we use majority voting. The three voters are three performance indices, which are each applied to all the feasible numbers of clusters. In general, an efficient clustering process guarantees clusters that are compact and well-separated, but different methods might provide different solutions for the same set of elements. With a different strategy, our three voters all try to identify the number of clusters N_c that best guarantees both requirements. How N_c is actually determined will be further clarified after introducing the three indices.

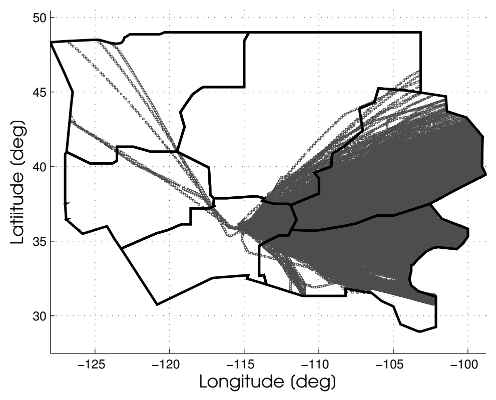
- **Average Silhouette Index** [65]: The silhouette value expresses how similar a trajectory is to its own cluster, compared to other clusters. The silhouette value ranges from -1 to 1, where a high value indicates that the trajectory is well-matched to its own cluster and poorly matched to neighboring clusters. Averaging the silhouette values of all trajectories, the average silhouette index is obtained.

- **Davies-Bouldin Index** [25]: For each cluster, a specific ratio between an intracluster distance and an intercluster distance (see [25] for details) is computed. Then, the average among the different ratios is computed. An index value close to zero indicates an efficient clustering process.

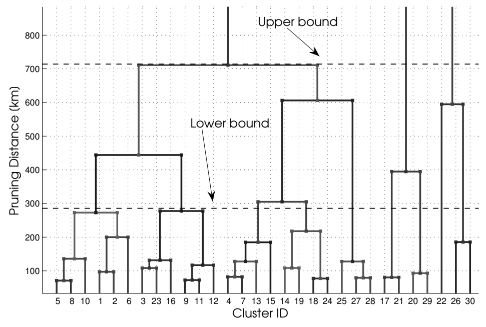
- **Dunn Index** [29]: This index is a ratio between the smallest intercluster distance among all pairs of clusters and the highest intracluster distance. An index value considerably greater than one indicates an efficient clustering process.

The three performance indices are computed for all the cluster configurations between N_c^{min} and N_c^{max}. Then, local maxima of the average silhouette index and the Dunn index, and local minima of the Davies-Bouldin index are identified. By selecting an odd number of voters, if at least two indices identify (i.e., vote for) the same number of clusters, that value is selected. Otherwise, for each index we compute the percentile loss in performance when considering the optimal number of clusters suggested by both the other indices, and average the two values. The number of clusters introducing the smallest loss in performance is selected as N_c.

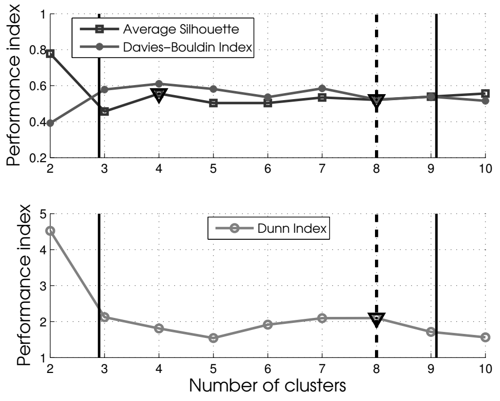
Coarse clustering is applied to all the exiting flights for the planning domain comprised of the six western-most Centers of the NAS, for 14 consecutive days spanning from July 1st, 2014 to July 14th, 2014. The outflow from McCarran International airport (LAS) is analyzed. Figure 3.6(a) shows all the flight tracks considered. In this case, $d_m = 2856.4\ km$, thus the slice of the dendrogram between 285.6 km and 714.1 km is considered. Figure 3.6(b) shows the dendrogram (for visual clarity, only the region of interest is shown), where the two bounds are highlighted with dashed lines. N_c^{min} is 3 and N_c^{max} is 9. Figure 3.6(c) shows the evolution of the three performance indices as the number of clusters is changed. The selected number of clusters, indicated by a vertical dashed line, is determined to be 8. This is a global maximum for the Dunn index and a global minimum for the Davies-Bouldin index in the region of interest. For the average silhouette value, $N_c = 4$ was the suggested optimal number of clusters. Note that, without distance limitations, all indices would identify the optimal number of clusters to be 2. If Fig. 3.6(a) is analyzed, there are a few trajectories



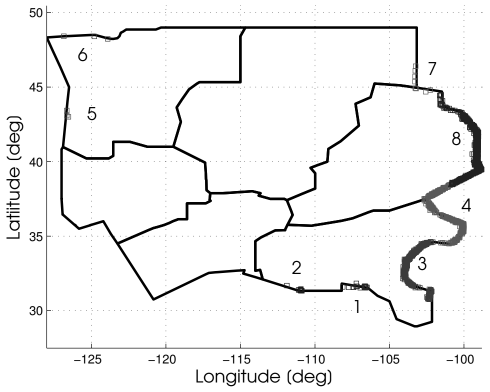
(a) All flight tracks.



(b) Dendrogram mapping distances between boundary crossing points.



(c) Performance indices.



(d) Coarse clusters identified.

Figure 2.5: Computation of the best number of coarse clusters for flights departing from LAS and exiting the planning domain.

headed West, while the majority of the exiting flow is towards East. Thus, from a clustering perspective, the exit points of the two flows define two clusters that are compact and well-separated. On the other hand, the maximum intradistance of the denser cluster would require the computation of distances between tracks that should not be clustered together. The imposition of the two distance thresholds avoids this undesired effect by splitting the denser cluster. Figure 3.6(d) shows, in different shades of gray, the 8 clusters.

In the framework of the ARM, we are interested in approximating the aircraft traffic flow dynamics with a set of aggregate routes. Each aggregate route should represent a well-traveled route also from a dynamical perspective. In fact, we do not want to aggregate trajectories that are spatially similar, but that are characterized by significantly different speeds. Thus, if in the same coarse cluster groups of trajectories with a considerably different ground speed are present, we treat those groups as different coarse clusters. For each trajectory belonging to a given coarse cluster, the average ground speed of the cruise phase is computed. The average speed distribution is then analyzed to assess if the coarse cluster is dynamically consistent. By dynamically consistent, we mean a coarse cluster where the average ground speed roughly follows a normal distribution. In this case, the majority of trajectories is characterized by a speed close to the mean μ of the distribution, while trajectories considerably slower or faster are less frequent (i.e., the standard deviation σ is generally low).

To understand if a coarse cluster is characterized by different dynamical patterns or not, the gap statistic [77] is used to compute the best number of clusters for the average ground speed distribution. The gap statistic is used because it is one of the few clustering indices that is defined for $N_c = 1$. The gap statistic is computed to determine the number of clusters that best represents the distribution of the average ground speed. If the output is one, then the coarse cluster is not further divided. Otherwise, the coarse cluster is split into as many coarse clusters as the gap statistic suggested.

An example of average ground speed distribution is shown in Fig. 2.6, with the flight tracks processed shown in the upper right corner. The average ground speed of recorded trajectories from Los Angeles International airport (LAX) to Seattle-Tacoma International airport (SEA) is plotted. The gap statistic suggests that the coarse cluster is dynamically consistent.

To have enough flight tracks to assess dynamic consistency of a coarse cluster, we discard coarse clusters with fewer trajectories than a lower threshold N_{traj}^{min} (in this work, $N_{traj}^{min} = 15$). A coarse cluster with less than N_{traj}^{min} trajectories is considered a cluster with a very low-frequency of flights, and is considered not representative enough of the aircraft traffic flow inside D. Examples of low-frequency clusters are coarse clusters 5, 6, and 7 in Fig. 3.6(d).

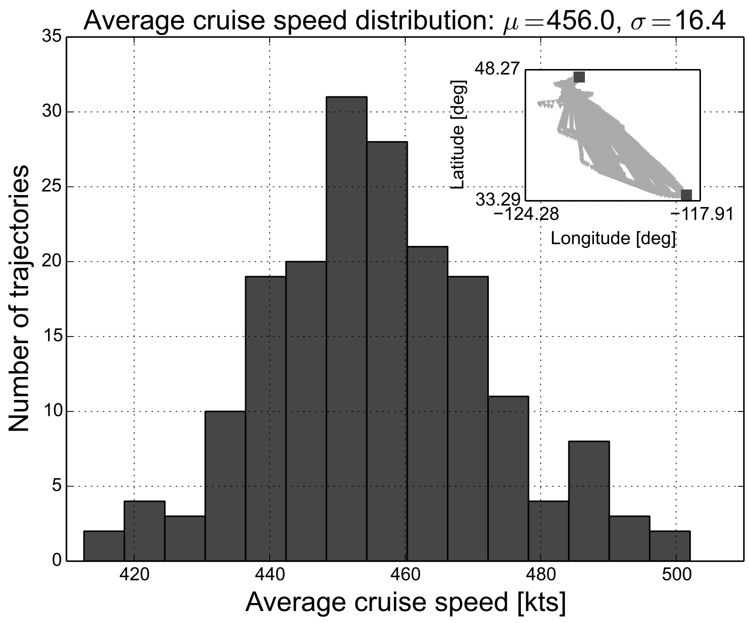


Figure 2.6: Average cruise speed for the LAX-SEA coarse cluster.

2.3 Flight Track Undersampling, Fréchet Distance Computation and Outlier Detection

In this section, we present a second strategy to reduce the computational burden of the algorithm. This time, the goal is to reduce the computational burden of each Fréchet distance computation.

The clustering algorithm presented is not designed to be a real-time process. The ARM is computed in advance by processing the historical dataset described in Sec. 2.1, while online corrections are computed as Chap. 3 will describe. Even if the ARM assembly is an offline procedure, using a large dataset implies the computation of hundreds of thousands of distances, and a reduction of the overall computational burden can still be a desirable achievement. We can use the properties of the Fréchet distance to identify, for each flight track, a reduced sequence of latitude-longitude pairs that are representative of the trajectory. The procedure to identify the sequence, which we refer to as flight track undersampling, is described in Sec. 2.3.1.

Then, Sec. 2.3.2 describes how the Fréchet distance between two undersampled tracks is computed. Once all pairwise distances are computed, in Sec. 2.3.3 we justify the necessity to have some form of outlier detection and removal [3, 52] in order to improve the accuracy of clustering processes, as well as to find more representative aggregate routes.

2.3.1 Flight Track Undersampling

For each coarse cluster, spatial similarity between flight tracks must be assessed. Distances between tracks, rather than between points, need to be computed. For this purpose, we use the Fréchet distance [4, 5, 18]. In this section, we describe the undersampling process that is used to reduce the computational cost for the Fréchet distance without a significant loss in accuracy.

The computational cost to determine each pairwise Fréchet distance increases quadratically with the number of points of each track [4]. The Ramer-Douglas-Peucker (RDP) algorithm [28, 62] is a means of reducing the number of discrete points representing each track, referred to as undersampling, without compromising the accuracy of the Fréchet distance. The accuracy is not compromised because data points that define significant changes in the heading of the flight track, are the ones also needed for an accurate evaluation of the Fréchet distance. Given a distance threshold η, the RDP algorithm undersamples portions of tracks that are quasi-linear with respect to the threshold η, while it preserves original data points in regions that are not quasi-linear.

In the planar case, the algorithm recursively splits the initial set of points in half. The splitting point is the point with the maximum Euclidean distance from the line connecting the endpoints of the segment considered. If all the intermediate points of a segment are within a $\pm\eta$ corridor around the segment, only the endpoints are kept. We adapt this algorithm to the spherical case, which requires distances between points and great circle arcs to be computed. We consider latitude-longitude coordinates projected onto a sphere with a $35,000$ ft altitude above Earth's surface. We translate flight tracks (where each latitude-longitude pair is characterized by a different altitude) into ground tracks (where all latitude-longitude pairs share the same altitude, i.e., they have been projected onto a sphere).

Let P_c be a latitude-longitude pair, and P_a and P_b the two endpoints of a great circle arc. We define $\hat{n}_a$, $\hat{n}_b$ and $\hat{n}_c$ as the versors going from the center of the sphere to points P_a, P_b and P_c, respectively. Then, we compute $\hat{n}_g = \hat{n}_a \times \hat{n}_b$, that is the versor normal to the plane containing the great circle arc, and $\hat{n}_f = \hat{n}_c \times \hat{n}_g$. $\hat{n}_f$ is the versor normal to a plane that passes through P_c and that is perpendicular to the plane containing the great circle arc. This plane cuts the great circle in two points (P_1 and $-P_1$). Computing $\hat{n}_{p_1} = \hat{n}_f \times \hat{n}_g$, we obtain the versor from the center of the sphere to one of those two points, while $-\hat{n}_{p_1}$ defines the other point. The last step is to compute the haversine distances η_1=h(P_c,P_1) and η_2=h(P_c,$-P_1$), where with h(·,·) we mean the haversine distance between two points on a sphere. The smallest value between η_1 and η_2 is selected. Note that when the two distances η_1 and η_2 are equivalent, the distance between the point and the great circle arc is $\pi/2$ radians, which is the maximum distance that can be achieved on the sphere between a point and a great circle arc. Figure 2.7 shows the geometrical interpretation of the problem. In this case η_1 is the distance between P_c and the great circle arc from P_a to P_b.

We use $\eta = 7.8 \cdot 10^{-5}$ rad, which corresponds to 500 m on a sphere with the radius defined above. Figure 2.8 shows an example of the procedure applied to two ground tracks from LAX to Phoenix Sky Harbor International airport (PHX). The number of data points for track $\mathcal{P}$ is reduced from 61 to 23, while the number of data points for track $\mathcal{Q}$ is reduced from 64 to 32. The Fréchet distance, computed using the routine described in Sec. 2.3.2, is

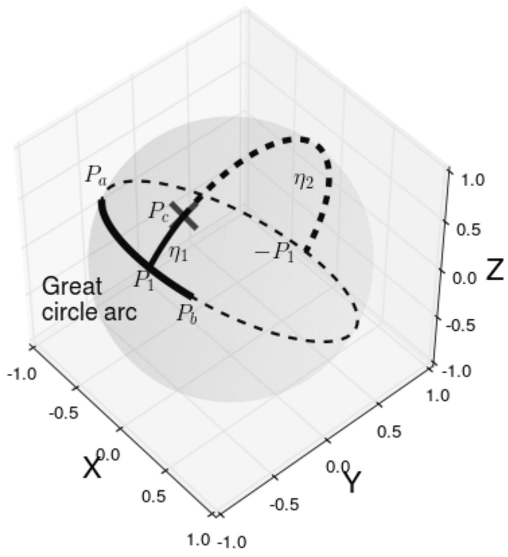
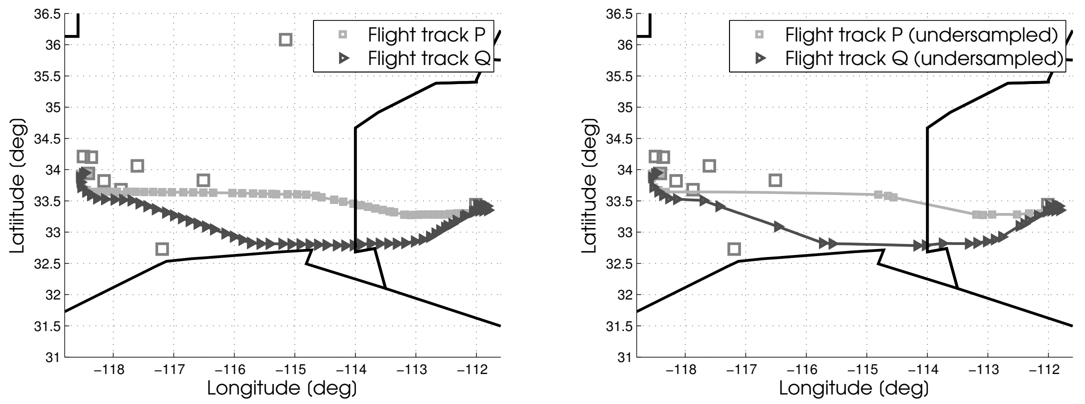


Figure 2.7: Graphical interpretation of the distance η_1 between a latitude-longitude pair and a great circle arc.

84.22 km and 84.35 km when using the original and the undersampled tracks, respectively. The percentile error is 0.15%, with the computational time dropping from 4.2 s to 0.5 s.

2.3.2 Fréchet Distance Computation

Reference [4] describes into details how the Fréchet distance $\mathcal{F}$ between two polygonal curves (i.e., piecewise-linear planar curves) is computed. A first polygonal curve $\mathcal{P}$ with p edges (and $p + 1$ data points), and a second polygonal curve $\mathcal{Q}$ with q edges (and $q + 1$ data points) are given. Then, a matrix with q rows of p horizontally stacked unitary squares is created. The unitary square in position (i,j) maps distances (generally Euclidean in planar applications) between the i-th normalized edge of $\mathcal{Q}$ and the j-th normalized edge of $\mathcal{P}$. Given a candidate distance $\mathcal{F}_\epsilon$, for each unitary square the locus of points whose distance is less or equal to $\mathcal{F}_\epsilon$ is computed. This locus is called free space [4], and is convex. The process is repeated for all unitary squares, and the free space diagram characterizing the two



(a) Initial flight tracks. (b) Undersampled flight tracks.

Figure 2.8: Two flight tracks from LAX to PHX, before and after undersampling.

polygonal curves is obtained. Note that, while each free space is convex, the union of all free spaces is not necessarily convex. In the free space diagram, the computation of $\mathcal{F}$ is subject to the solution of a decision process. A monotone path from the upper left corner (first data point of both curves) to the lower right corner (last data point of both curves) is searched, since curves need to be entirely spanned [4]. Due to a monotonicity requirement that arises from a backtracking constraint, we are only interested in the intersections of each free space with the four sides of the associated unitary square. They are defined $L^F_{i,j}$, $L^F_{i,j+1}$, $B^F_{i,j}$, $B^F_{i+1,j}$ in [4]. The free space diagram can be interpreted as a directed graph. A different candidate distance $\mathcal{F}_\epsilon$ will result in a different set of $L^F_{i,j}$, $L^F_{i,j+1}$, $B^F_{i,j}$, $B^F_{i+1,j}$ $(1 \leq i \leq q,\ 1 \leq j \leq p)$, and in different adjacency properties of the directed graph. The smallest distance such that a monotone path compatible with these adjacency properties is found, is the Fréchet distance $\mathcal{F}$. Figure 2.9 shows an example of free space diagram with $p = 33$ and $q = 18$.

In most applications, the Fréchet distance is applied to polygonal curves. $L^F_{i,j}$, $L^F_{i,j+1}$, $B^F_{i,j}$, $B^F_{i+1,j}$ are computed as intersections of circles with lines, i.e., as solutions of a quadratic equation. For the purpose of calculating the Fréchet distance in our application, flight track points in a $3D$ position space are projected onto a sphere at a fixed altitude to define a ground track; for each track, consecutive projected points are connected by a great circle arc to obtain

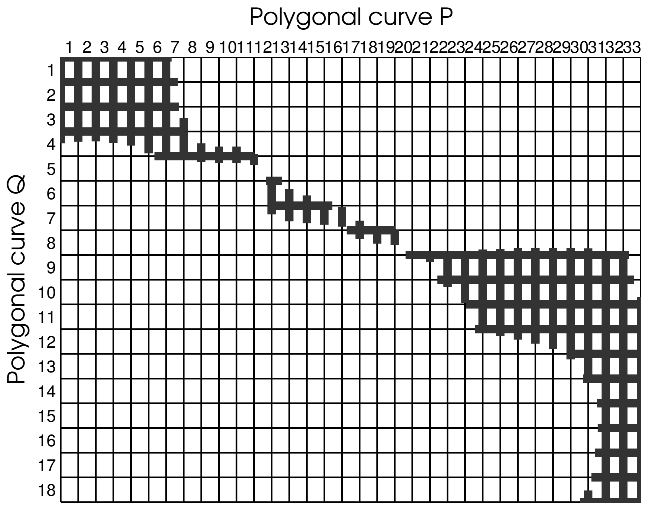
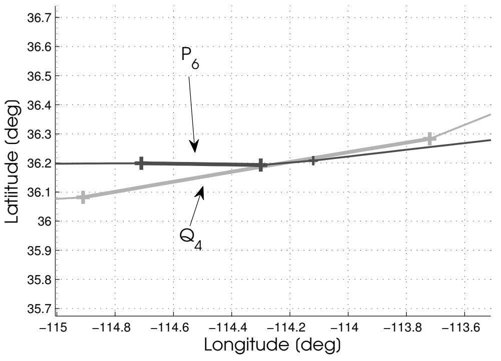


Figure 2.9: Example of a free space diagram. A monotone path from the upper left corner to the lower right corner within the gray region is identifiable.

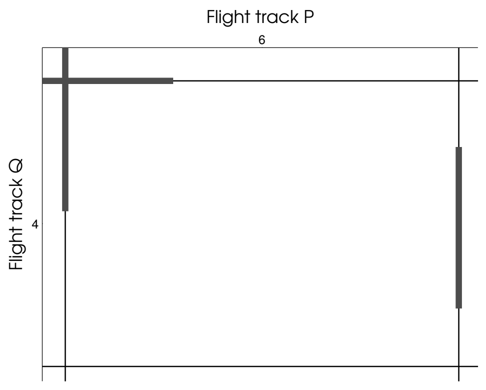
a continuous curve. The different framework implies a different approach in the computation of $L^F_{i,j}$, $L^F_{i,j+1}$, $B^F_{i,j}$, $B^F_{i+1,j}$. Polygonal curves in the Cartesian formulation become great circle arc sequences. Likewise, the Euclidean distance is replaced with the haversine distance. The solution of the decision process follows the same procedure mentioned above instead.

Each edge of a free space, as the one shown in Fig. 2.9, maps the mutual distance between a great circle arc of a track, and a latitude-longitude pair of the other track. For each unitary square, determining the existence and extension of $L^F_{i,j}$, $L^F_{i,j+1}$, $B^F_{i,j}$, $B^F_{i+1,j}$ involves two latitude-longitude pairs per track, and one great circle arc per track connecting the latitude-longitude pairs.

$L^F_{i,j}$, $L^F_{i,j+1}$, $B^F_{i,j}$, $B^F_{i+1,j}$ are computed detecting a possible intersection between a circle centered in a latitude-longitude pair of curve $\mathcal{P}$ and a great circle arc of curve $\mathcal{Q}$ (or vice versa). The radius of the circle is the candidate distance $\mathcal{F}_m$. Because each square is unitary, each intersection ranges between 0 and 1 (meaning respectively that the tail/head node of the great circle arc is part of the intersection).



(a) Two great circle arcs, belonging to different ground tracks.



(b) Associated unitary square with $\mathcal{F}_m = 30.5\ km$.

Figure 2.10: Example of how the haversine distance along two great circle arcs is mapped into the associated free space.

Figure 2.10 shows a visual example of the four intersections $L^F_{i,j}$, $L^F_{i,j+1}$, $B^F_{i,j}$, $B^F_{i+1,j}$. On the left, the sixth great circle arc $\mathcal{P}_6$ of ground track $\mathcal{P}$ and the fourth great circle arc $\mathcal{Q}_4$ of ground track $\mathcal{Q}$ are highlighted in gray. On the right, the associated unitary square is depicted. For the upper edge ($B^F_{4,6}$), there is an intersection ranging from 0 to roughly 0.25. Thus, a circle with radius $\mathcal{F}_m$, centered in the left vertex of $\mathcal{Q}_4$, contains the first quarter of $\mathcal{P}_6$. For the right edge ($L^F_{4,7}$), the intersection spans roughly from 0.25 to 0.75. Thus, the same circle, centered in the right vertex of $\mathcal{P}_6$, contains the central part of $\mathcal{Q}_4$. No intersection is identified for the lower edge.

In the spherical formulation, the intersection of circles with great circle arcs is computed as solution of a trigonometric equation. Given a point on a sphere of radius r_s defined with the triplet latitude-longitude-radius, the equivalent position in Cartesian coordinates is $\vec{x} = [x\, y\, z]'$. The position of the center of the circle (i.e., a latitude-longitude pair of one track) is defined by the vector $\vec{x}_c$. The position of the endpoints of a great circle arc belonging to the other curve is defined by vectors $\vec{x}_1$ and $\vec{x}_2$ respectively. The length of the great circle arc is d. The goal is to determine if the great circle containing the great circle arc between $\vec{x}_1$ and $\vec{x}_2$ intersects the locus of points that (i) belong to the sphere and (ii) have a haversine distance of $\mathcal{F}_m$ with respect to $\vec{x}_c$.

The great circle arc coordinates $\vec{x}$ are parametrized with f, such that $\vec{x} = \vec{x}_1$ if $f = 0$ and $\vec{x} = \vec{x}_2$ if $f = 1$. Using spherical trigonometry properties, we can write

$$\vec{x} = \frac{\sin((1-f)d)}{\sin(d)}\vec{x}_1 + \frac{\sin(fd)}{\sin(d)}\vec{x}_2 \tag{2.1}$$

Requirements (i) and (ii) are translated into the following set of equations

$$x^2 + y^2 + z^2 = r_s^2 \tag{2.2a}$$

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = 4r_s^2 \sin^2\left(\frac{\mathcal{F}_m}{2}\right) \tag{2.2b}$$

Substituting Eq. 2.2a into Eq. 2.2b, the following locus is obtained

$$xx_c + yy_c + zz_c = r_s^2\left[1 - \sin^2\left(\frac{\mathcal{F}_m}{2}\right)\right] \tag{2.3}$$

and combined with Eq. 2.1. Expanding $\sin((1-f)d)$ using trigonometric identities, the final trigonometric equation is written in compact form as

$$C_1 \cos(fd) + C_2 \sin(fd) = C_3 \tag{2.4}$$

where

$$\begin{cases} C_1 = x_1 x_c + y_1 y_c + z_1 z_c \\ C_2 = \dfrac{x_2 x_c + y_2 y_c + z_2 z_c - \cos(d)(x_1 x_c + y_1 y_c + z_1 z_c)}{\sin(d)} \\ C_3 = r_s^2 \left[1 - \sin^2\left(\dfrac{\mathcal{F}_m}{2}\right)\right] \end{cases}$$

Equation 2.4 can be rewritten as

$$R\cos(fd - \gamma) = C_3, \text{ with } R = \sqrt{C_1^2 + C_2^2}, \ \gamma = \tan^{-1}\left(\frac{C_2}{C_1}\right) \tag{2.5}$$

For a free space like the one shown in Fig. 2.9, the overall number of $L_{i,j}^F$ to be computed is $(p+1)q$. The overall number of $B_{i,j}^F$ is $(q+1)p$. Thus, Eq. 2.5 needs to be solved $2pq+q+p$ times. The existence of feasible values for $L_{i,j}^F$ and $B_{i,j}^F$ depends on $\cos^{-1}\left(\frac{C_3}{R}\right)$ as follows

1. $\cos^{-1}\left(\frac{C_3}{R}\right) > 1$: the great circle containing the great circle arc from $\vec{x}_1$ to $\vec{x}_2$ does not intersect the locus defined by Eq.2.3. No feasible interval for f is obtained.
2. $\cos^{-1}\left(\frac{C_3}{R}\right) = 1$: the great circle containing the great circle arc from $\vec{x}_1$ to $\vec{x}_2$ is tangent to the locus defined by Eq.2.3. A single solution $f_1 = f_2 = f$ is obtained. If $0 \leq f \leq 1$, the solution is feasible, otherwise the point of tangency is external with respect to the great circle arc.
3. $\cos^{-1}\left(\frac{C_3}{R}\right) < 1$: the great circle containing the great circle arc from $\vec{x}_1$ to $\vec{x}_2$ intersects the locus defined by Eq.2.3 in two distinct points. Two solutions f_1 and f_2 are obtained. If

(a) $0 \leq f_1 \leq 1$ and $0 \leq f_2 \leq 1$: both solutions are feasible, and the interval of validity for the parameter is $f \in [\min(f_1, f_2)\ \ \max(f_1, f_2)]$.

(b) $0 \leq f_1 \leq 1$ and $f_2 > 1$ or vice versa: one solution is feasible. Define $\bar{f} = \min(f_1, f_2)$ and d_1 and d_2 the haversine distances between $\vec{x}_c$ and $\vec{x}_1$, and $\vec{x}_c$ and $\vec{x}_2$ respectively. If $d_1 < d$, the interval of validity for the parameter $f \in [0\ \ \bar{f}]$, otherwise $f \in [\bar{f}\ \ 1]$.

(c) $f_1 > 1$ and $f_2 > 1$: the great circle arc from $\vec{x}_1$ to $\vec{x}_2$ is either completely contained in the locus, or totally external. If $d_1 > d$ and $d_2 > d$, no interval of validity for the parameter is obtained. If $d_1 < d$ and $d_2 < d$, $f \in [0\ \ 1]$.

The $2pq + q + p$ intervals computed solving Eq. 2.5, as well as the monotonicity requirement, determine the adjacency properties of the directed graph mapping the free space diagram. A depth-first-search solver [22, 39] is then used to determine the smallest distance value such that, in the associated free space diagram, a path between node 1 (upper left corner) and node $3pq + 2(p + q) + 1$ (lower right corner) exists. That value identifies the Fréchet distance $\mathcal{F}$.

2.3.3 Outlier Detection

For each coarse cluster, all pairwise Fréchet distances are computed as described in Sec. 2.3.2. The set of distances is used as input for the outlier detection algorithm. To detect outliers, two different methods are used: (i) DBSCAN [52] and (ii) the method described in [3]. Both methods are characterized by two steps, namely, outlier detection and clustering. In this context, the methods are used to detect outliers, while the clustering technique introduced in Sec. 2.2 is used to cluster the remaining ground tracks.

DBSCAN is based on two parameters, *MinPts* and ϵ. Using these two parameters, a cluster is defined to be a set of density-connected points which is maximal with respect to density-reachability [52]. Each cluster is characterized by some core points, that define the denser region of the cluster, and some border points. The set of points not belonging to any cluster defines outliers.

The method presented in [3] is based on a similar concept. Outliers can be viewed as elements or small groups of elements located in low-density zones, contrasting with the denser intracluster structure. The detection of outliers is based on an iterative process that computes, at each iteration, a radius that is a certain multiple of the average nearest-neighbor distance. For each element of the dataset, the associated connectivity (i.e., how many elements are within the selected radius) is computed, and the average connectivity $\bar{c}_j$ is also computed. Then, every element characterized by a connectivity less than $\beta\bar{c}_j$ is labeled as an outlier and discarded. The process is repeated until no more outliers are identified. In the original paper, the authors propose $\beta = 1/3$.

When a coarse cluster is processed with the two methods, all the outliers identified by at least one method are selected. As an example, Fig. 4.1 shows all the flight tracks from LAX to Salt Lake City International airport (SLC). In light gray are the tracks not labeled as outliers. In dark gray are the tracks labeled as outliers. Out of the 121 original flight tracks, 3 are labeled as outliers. Some path stretches and an evident holding pattern can be seen. This motivates the relevance of outlier detection also for clear weather days, to discard tracks with significant tactical maneuvers such as path stretches and holding patterns. It is also important to remark that, while we discard outliers when computing aggregate routes, their analysis can still provide useful insights. Appendix C describes a procedure for outlier categorization.

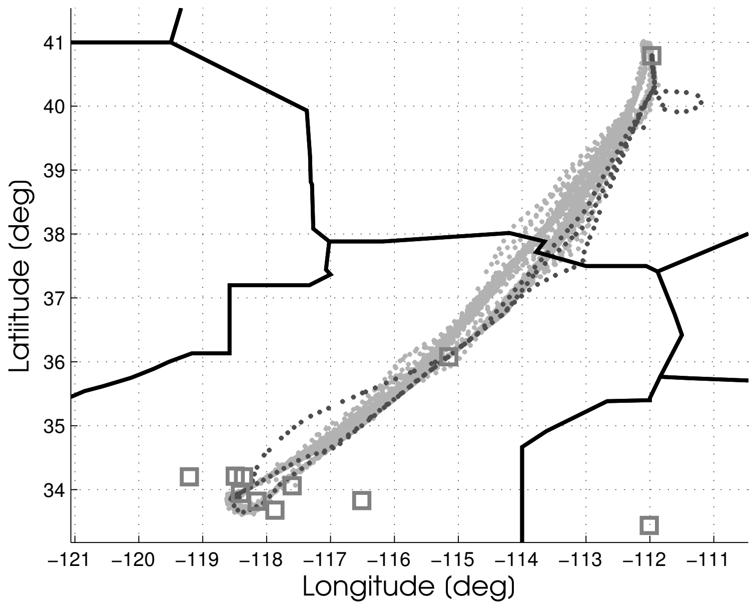


Figure 2.11: Flight tracks forming the LAX-SLC coarse cluster. In light gray are shown the filtered tracks, while in dark gray are highlighted the outliers.

2.4 Fine Clustering

For each coarse cluster, after outlier removal, ground tracks undergo a fine clustering process. The goal of this process is (i) to assess if a coarse cluster needs to be further divided into fine clusters, and (ii) to compute the optimal number of fine clusters, if necessary.

To assess if a coarse cluster is compact enough (i.e., it does not need to be further divided), we want to understand if $N_c = 1$ defines an optimal clustering configuration for the coarse cluster. Since the three indices introduced in Sec. 2.2 are not defined for $N_c = 1$, the optimality of the single cluster case must be assessed differently. We use the properties of the dendrogram to assess the compactness of the coarse cluster. Given a dendrogram, the pruning distance to move from a number of clusters to the next one is easily identifiable. We are interested in $d_{2\to1}$, i.e., the pruning distance where the number of clusters changes from two to one. We set a threshold $\eta_{lb} = \min(d_{min}, 0.05 d_{OD})\ km$, where d_{OD} is the haversine distance between the two airports (internal flight tracks), or the distance between the airport and the centroid of the exit/entry points (exiting/entering flight tracks). d_{min} is set equal to 40 km instead. The threshold represents the maximum allowed intracluster distance to consider a coarse cluster as compact. It can be interpreted as a maximum allowed lateral deviation with respect to the great circle arc connecting the origin and destination points. For an O-D pair closer than 800 km, the maximum intracluster distance allowed is the constant value $\eta_{lb} = d_{min}$, which is considered already significant to justify fine clustering. Otherwise, $\eta_{lb} = 0.05 d_{OD}$. The threshold η_{lb} has been tested on coarse clusters of different sizes, and has proven to be generally successful in assessing the compactness of coarse clusters.

For a non-compact coarse cluster, the optimal number of fine clusters is computed as described in Sec. 2.2. Due to the different granularities required, the lower and upper bounds for pruning the dendrogram are chosen differently. The lower bound is selected consistently with the coarse cluster compactness measure η_{lb}, while the upper bound is η_{ub}. In a similar fashion, we set $\eta_{ub} = \max(d_{max}, 0.15 d_{OD})$ km, where $d_{max} = 90$ km. For O-D pairs closer than 600 km, $\eta_{ub} = d_{max}$, otherwise $\eta_{ub} = 0.15 d_{OD}$. Given the possible number of clusters within these bounds, the three performance indices are computed, and the optimal number of clusters is selected. The process is summarized in the flowchart shown in Fig. 2.12.

Figure 2.13 shows an example of the process applied to the LAX - SEA O-D airport pair. The coarse cluster comprised 185 flight tracks. The outlier detection algorithm identified outliers. The 22 outliers and the remaining 163 tracks are shown in dark and light gray, respectively, in Fig. 2.13(a). Figure 2.13(b) shows the dendrogram computed using the 13,203 pairwise Fréchet distances. The compact coarse cluster case option is discarded. The upper bound is set at 231 km, and intersects the dendrogram in one single point. Since the compactness test identified the need to further divide the coarse cluster, the minimum allowed number of clusters is 2. The lower bound cuts the dendrogram in 6 different points. The range for N_c spans from 2 to 6. Figure 2.13(c) shows the evolution of the performance indices. Note that $N_c = 1$ and $N_c = 7$ are used as auxiliary values to identify local maxima/minima for

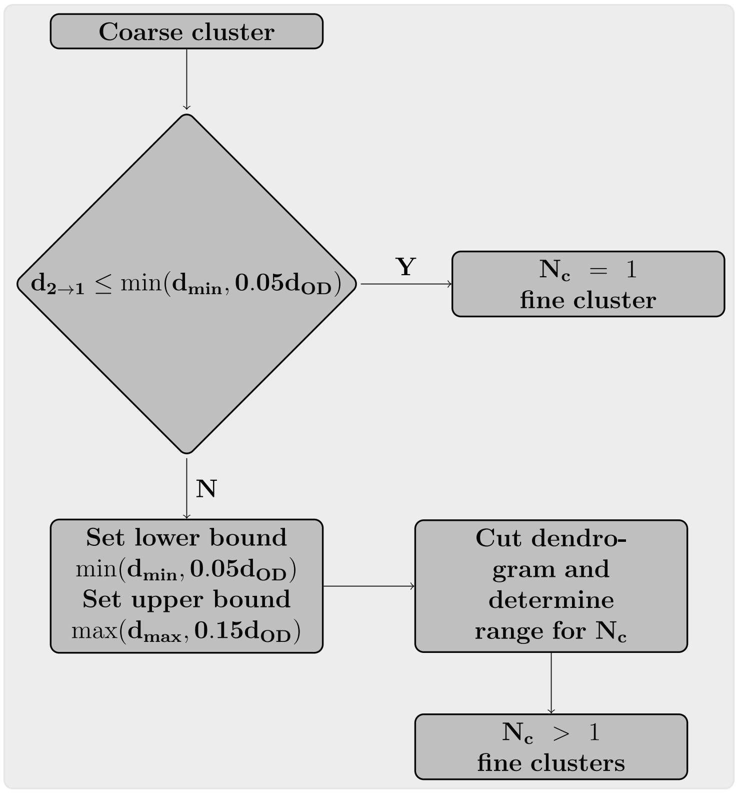
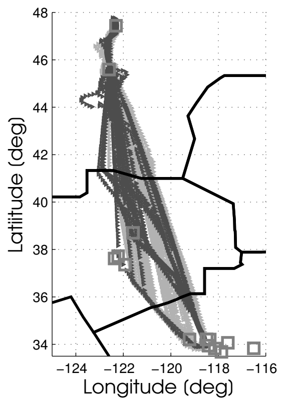
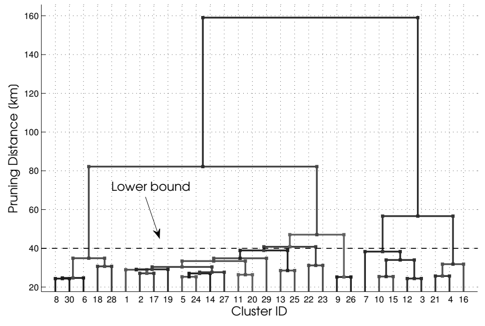

Coarse cluster
$\mathbf{d_{2\to1}} \leq \min(\mathbf{d_{min}}, \mathbf{0.05d_{OD}})$
Y
$\mathbf{N_c} = 1$ fine cluster
N
Set lower bound $\min(\mathbf{d_{min}}, \mathbf{0.05d_{OD}})$ Set upper bound $\max(\mathbf{d_{max}}, \mathbf{0.15d_{OD}})$
Cut dendrogram and determine range for $\mathbf{N_c}$
$\mathbf{N_c} > 1$ fine clusters

Figure 2.12: Flowchart describing the selection of the optimal number of fine clusters given a coarse cluster.

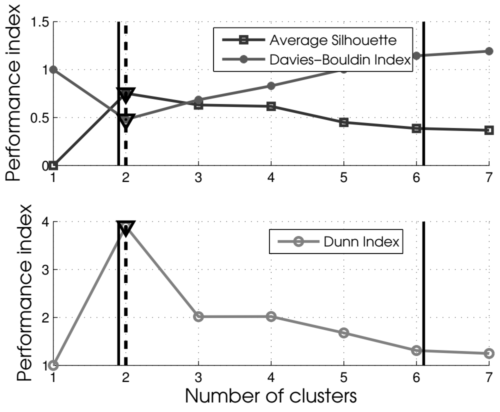
$N_c = 2$ or $N_c = 6$. For $N_c = 7$ the indices are computed. For $N_c = 1$ arbitrary values associated with a poor clustering outcome are assigned. All three indices indicate that the optimal number of clusters is 2.



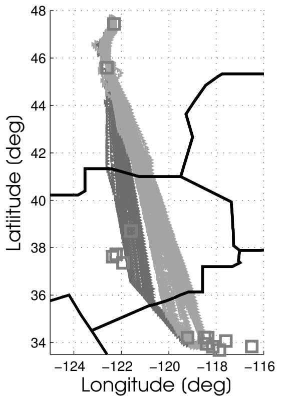
(a) Flight tracks, with outliers highlighted in dark gray.



(b) Dendrogram mapping distances between tracks.



(c) Performance indices.



(d) Fine clusters highlighted with different shades of gray.

Figure 2.13: Computation of the optimal number of fine clusters for the LAX-SEA coarse cluster.

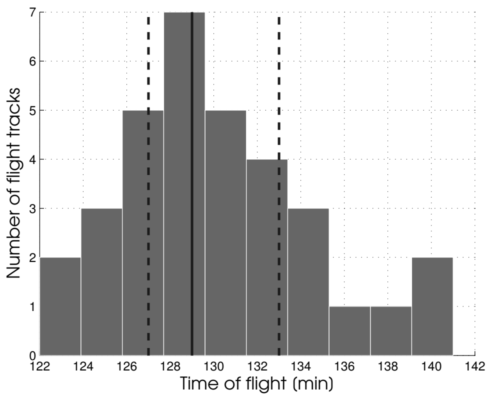
2.5 Aggregate Route Computation

Keeping in mind that a graph-oriented representation of the air traffic flow is our goal, we want to approximate each fine cluster with an aggregate route. Each fine cluster groups flight tracks that are similar spatially and dynamically. Nonetheless, discrepancies in the times of flight of the different tracks are present. We sort the times of flight of all tracks belonging to the fine cluster, compute the quartiles and select the median as the time of flight representative of the cluster. Similarly to what experienced in Sec. 2.2, with a large enough dataset the times of flight will be normally distributed. We then discard tracks belonging to the first and fourth quartiles, to eliminate tracks with a time of flight that is considered too different with respect to the median of the fine cluster. Note that these flight tracks still belong to the fine cluster, and should be considered for applications such as spatial distribution analysis of tracks.

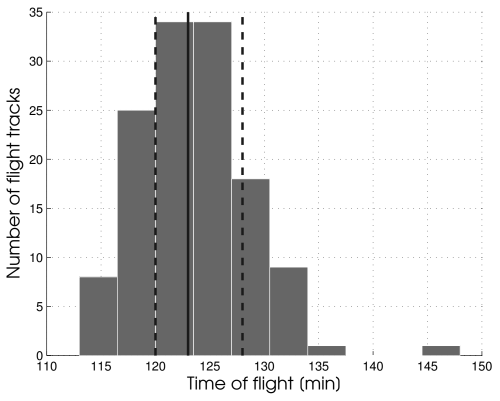
Second quartile tracks have fewer data points than the median. Sequences of data points where the local average ground speed is the highest are resampled so that one artificial data point is added. The procedure is iteratively repeated until consistency with the median value. Third quartile tracks have more data points than the median. The same process is applied, but sequences of data points where the local average ground speed is the lowest are resampled so that one data point is dropped. Given requirements on spatial and dynamical similarity, differences in times of flight (with respect to the median) no greater than 6 minutes (for a route with a time of flight of 150 minutes) were identified for the processed dataset.

When tracks are resampled, for each time entry there is a number of data points equivalent to the number of tracks belonging to the second and third quartiles combined. Quantities such as latitude, longitude, altitude, ground speed are averaged. For quantities such as flight level, Center and sector, the mode is selected. A node characterized by the averaged quantities is created for every time entry. The time-ordered sequence of nodes is the aggregate route.

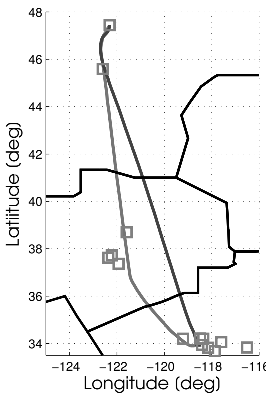
Figure 2.14 shows an example applied to flight tracks connecting the same airport pair described in Sec. 2.4. Figure 2.14(a) and Fig.2.14(b) show the times of flight distribution for the left and right fine clusters of Fig. 2.13(d). The left fine cluster has an aggregate time of flight of 129 minutes, compared to the 123 minutes of the right fine cluster. For the left fine cluster, flight tracks with a time of flight between 127 and 134 minutes are used to compute the aggregate route (20 out of 33 tracks). For the right fine cluster, flight tracks with a time of flight between 120 and 128 minutes are used to compute the aggregate route (90 out of 130 tracks). The two resulting aggregate routes are shown in Fig. 2.14(c). Figure 2.14(d) shows the distribution of the latitude-longitude centroids (for visual clarity, one centroid every ten is highlighted), and the resulting aggregate route, for the right fine cluster.



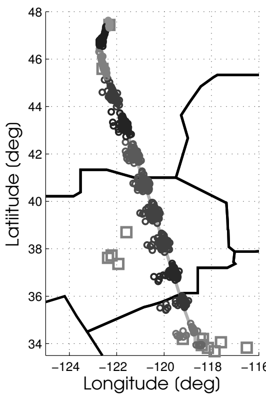
(a) Times of flight distribution for the left fine cluster.



(b) Times of flight distribution for the right fine cluster.



(c) Resulting aggregate routes.



(d) Latitude-Longitude centroids for the right aggregate route.

Figure 2.14: Computation of the aggregate routes for the LAX-SEA coarse cluster.

2.6 Airport Departure and Arrival Capacity Estimation

When in a graph, some nodes (sources) have only outflow, some nodes (sinks) have only inflow, and edges have flow restrictions, the graph becomes a network [78]. Our graph-oriented approximation of the air traffic flow is a network. (i) Airports inside D are both sources and sinks, since they release new flights and absorb landing flights, while boundary crossing nodes are either sources or sinks depending on the flight category. (ii) The state of each node has an upper bound that accounts for flow limitations, as Chap. 5 describes. For the n_a airports inside D, an estimation of the expected inflow and outflow can be computed via analysis of the same historical dataset used to build the ARM. In this section, we will describe how to compute such estimate.

The characterization of airport departure and arrival capacity has been introduced in [34], where a method for estimating practical airport capacities under different operational conditions is presented. This estimation is not an easy process, since airport capacity depends on many factors, such as the runway configuration, weather, and the aircraft type. Geometrically, the relationship between departures and arrivals can be described by a capacity curve that is generally called a Gilbo envelope. More recent work can be found in [9, 38]. Reference [9] interprets the Gilbo envelope as the intersection of a set of inequality constraints that define a feasible region for departures/arrivals allocation. The constants defining the different constraints are assumed to be known. Reference [38] shows a technique, based on [34], to compute Gilbo envelopes. Average values for hourly departure and arrival capacity are compared with the declared values.

In our approach, we look for a set of inequalities that bound departures and arrivals for each airport of interest. The inequalities fit well with the graph-oriented interpretation of the airflow because (i) from a network perspective, they bound the throughput along the network, and (ii) they are used as constraints in the IP problem as shown in Chap. 5. We want a simple, yet accurate description of the Gilbo envelope characterizing each airport. Our process has the following steps: (i) process the historical dataset and store hourly departures/arrivals pairs in a 2D Cartesian graph (x-axis for departures, y-axis for arrivals), (ii) eliminate outliers using DBSCAN and the Manhattan distance as metric (being departures and arrivals integer-valued), (iii) add the three auxiliary points [0 0], [0 $\max_a$], [$\max_d$ 0] (where $\max_d$ and $\max_a$ represent the maximum values for departures and arrivals, respectively) to the distribution and compute the convex hull, (iv) identify all the pairs that do not lie on one of the two axes, and compute the least-square line approximating those points (v) identify the Gilbo envelope as the union of a horizontal line passing through [0 $\max_a$], the line computed in (iv), and a vertical line passing through [$\max_d$ 0].

The three lines defining the Gilbo envelope can be interpreted as follows. The horizontal and vertical constraints define maximum operational capabilities of arrivals and departures due to limitations associated with runway availability and ATCs capabilities (regardless of the mutual interaction). This is also the reason why we discard outliers. An isolated departures/arrivals pair with a departure or/and arrival value that is much higher than the rest of the distribution, would lead to a capacity over-estimation. The sloped line identifies a high-demand region for both departures and arrivals, where maximum capacities cannot be reached simultaneously.

Figure 2.15 shows the procedure applied to PHX. Hourly departures/arrivals pairs are computed. Outliers are removed, the convex hull is approximated with the three segments. For this airport, $x \leq 42$, $y \leq 44$, $y + 1.67x \leq 89.53$, being x and y hourly values for departures and arrivals, respectively.

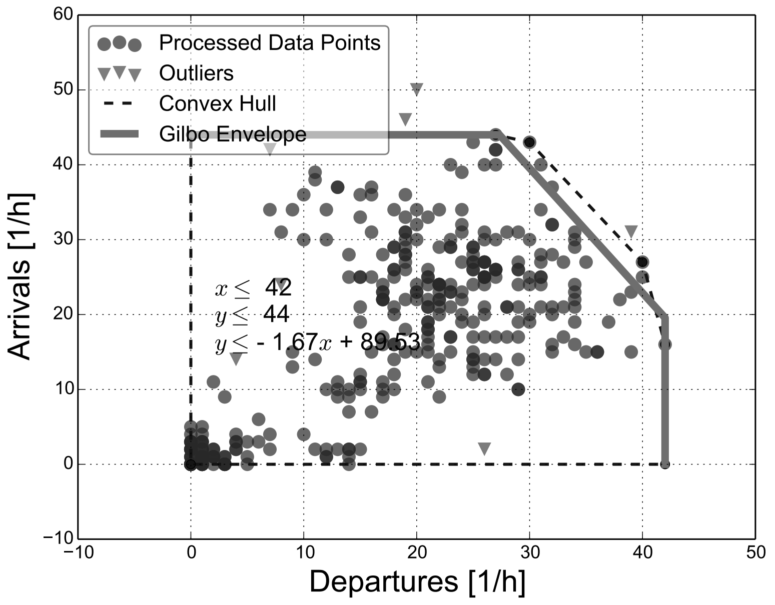


Figure 2.15: Gilbo envelope for PHX.

2.7 Application to the Six Western-most Centers of the NAS

The clustering algorithm is used to develop an aggregate route model for a planning domain that comprises the following 6 Centers: Los Angeles (ZLA), Albuquerque (ZAB), Seattle (ZSE), Oakland (ZOA), Salt Lake City (ZLC), and Denver (ZDV). 21 airports inside the planning domain are considered. They are (i) ZLA: Burbank Bob Hope airport (BUR), McCarran International airport (LAS), Los Angeles International airport (LAX), Long Beach airport (LGB), Ontario International airport (ONT), Oxnard airport (OXR), Palm Springs International airport (PSP), San Diego International airport (SAN), John Wayne airport (SNA), Van Nuys airport (VNY), (ii) ZAB: Albuquerque International Sunport airport (ABQ), Phoenix Sky Harbor International airport (PHX), Tucson International airport (TUS), (iii) ZSE: Portland International airport (PDX), Seattle-Tacoma International airport (SEA), (iv) ZOA: Oakland International airport (OAK), San Francisco International airport (SFO), Norman Y. Mineta San José International airport (SJC), Sacramento International airport (SMF), (v) ZLC: Salt Lake City International airport (SLC), (vi) ZDV: Denver International airport (DEN).

For each of the 21 airports considered, internal, exiting, and entering flight tracks are processed separately. The planning domain comprises the six western-most Centers of the continental NAS. As a consequence, overflights are scarce, and thus ignored. Flight trajectories for 14 days, from July 1st, 2014 to July 14th, 2014 are processed.

Considering internal flight tracks, 21 airports define 420 different O-D airport pairs. Out of the 420 different O-D airport pairs, 179 are not characterized by aircraft flow. This is due to two different factors. (i) Proximity of some of the airports considered, which results in absence of flight connections. This is particularly evident for the ZLA Center. Because of its location, LAS has flight connections with most of the other airports within the Center, while the other airports, apart from a couple of exceptions, have flight connections to LAS only. (ii) Scarcity in the number of flights from/to an airport. This is the case of OXR, where identified flight tracks were so scarce that no internal aggregate routes were identified.

For exiting and entering flight tracks, coarse clusters are first identified as described in Sec. 2.2. Then, for each coarse cluster the optimal number of clusters is computed similarly to internal flight tracks.

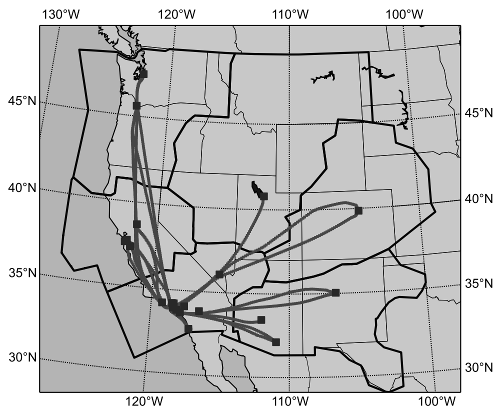
For internal flight tracks, 17,497 flight tracks are processed and, after outlier detection, 15,731 tracks are aggregated into 371 aggregate routes. For exiting flight tracks, 13,813 flight tracks are processed and, after outlier detection, 12,826 tracks are aggregated into 156 aggregate routes. For entering flight tracks, 15,897 flight tracks are processed and, after outlier detection, 14,495 tracks are aggregated into 184 aggregate routes.

Table 2.1 highlights the number of flight tracks processed for each airport. For all Centers apart from ZLA, the algorithm identified an internal, exiting and entering flow for each airport. Inside ZLA, no aggregate routes were identified for OXR. For three other airports, i.e., BUR, PSP, VNY, only internal aggregate routes were computed. BUR and PSP are de facto regional airports. PSP is labeled international because of a few connections with Canada, but the majority of the traffic flow is internal. On the other hand, VNY airport is one of the busiest general aviation (i.e., all civil aviation operations other than scheduled air services) airports in the world and mainly handles local business jet traffic. For VNY airport, the processed flight tracks showed less predictable patterns (the relative percentage of outliers identified was the highest among all airports), and only 2 internal aggregate routes were identified.

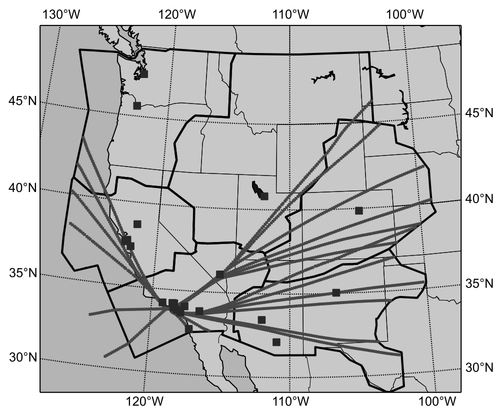
For airports with exiting and entering aggregate routes, the flow is mainly West-to-East for exiting aggregate routes, and East-to-West for entering aggregate routes. This is due to the location of D. Covering the western-most portion of the NAS, the outflow is directed towards airports in the remaining portion of the NAS, Europe, Africa, or the Middle East (and vice-versa for the inflow). One exception is LAX, which also has a western outflow and eastern inflow.

Figure 2.16 shows the internal, exiting, entering aggregate routes and the Gilbo envelope for LAX. Focusing on the exiting flow (Fig. 2.16(b)), six aggregate routes headed West are identified, and can be divided into two flows. The northern flow, with four routes, connects LAX to major Asian destinations, such as Narita International airport, Haneda airport, Hong Kong International airport, Beijing Capital International airport and Shanghai Pudong International airport. The southern flow, with two exiting aggregate routes, connects LAX to to Australia, with Sydney airport and Melbourne airport as main destinations. The same considerations can be extended to entering aggregate routes (Fig. 2.16(c)). The southern flow connects LAX to Mexican and South-American destinations. The eastern flow, that accounts for the majority of aggregate routes, connects LAX to the ASPM-77 airports not contained in D, and to major European, African and some Asian destinations.

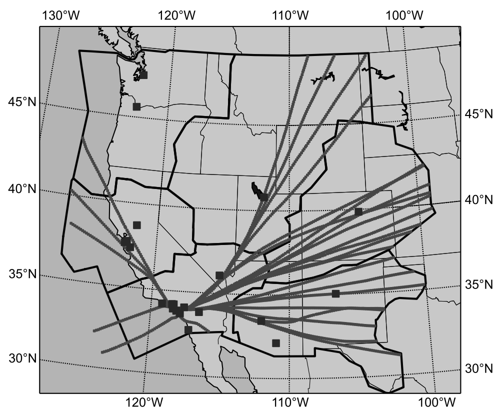
Considering all the airports in D, the airflow is thus approximated with (i) a set of 711 aggregate routes, divided into 371 internal, 156 exiting, and 184 entering routes (ii) 20 Gilbo envelopes that describe hourly departures/arrivals for each airport (OXR has been discarded for lack of aggregate routes).



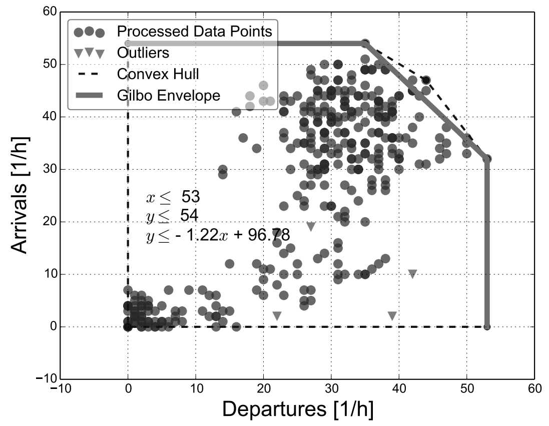
(a) Internal aggregate routes.



(b) Exiting aggregate routes.



(c) Entering aggregate routes.



(d) Gilbo envelope.

Figure 2.16: Internal, exiting, entering aggregate routes and Gilbo envelope with LAX as reference airport.

Table 2.1: Comparison between flight tracks before the outlier detection (IFT), after the outlier detection (FT), and the resulting number of aggregate routes (AR) for each airport within the planning domain.

	Internal			Exiting			Entering		
	IFT	FT	AR	IFT	FT	AR	IFT	FT	AR
BUR	423	376	11	38	12	×	38	14	×
LAS	1613	1468	34	1695	1515	12	1763	1612	21
LAX	2083	1930	21	2782	2627	20	3375	3197	27
LGB	244	207	10	77	65	3	66	33	1
ONT	379	338	13	98	87	5	136	82	4
OXR	19	×	×	35	17	×	2	×	×
PSP	118	97	10	10	×	×	12	7	×
SAN	1089	1005	20	640	595	10	809	661	14
SNA	601	516	19	240	202	8	319	236	8
VNY	78	18	2	34	12	×	41	29	×
ABQ	318	252	12	365	343	5	391	364	4
PHX	1728	1549	27	1424	1305	15	1892	1757	9
TUS	187	135	8	137	121	5	166	139	4
PDX	1064	973	26	436	384	9	517	424	13
SEA	1728	1565	25	998	916	8	988	858	18
OAK	861	801	17	157	119	4	223	169	6
SFO	1490	1382	24	1507	1423	18	2258	2173	22
SJC	599	540	14	130	120	4	202	182	6
SMF	547	509	18	135	122	6	195	180	8
SLC	1186	1073	33	791	729	14	841	796	13
DEN	1142	997	27	2084	1956	10	1663	1582	6
Total	17497	15731	371	13813	12826	156	15897	14495	184

Chapter 3

Airborne Rerouting Design

In Chap. 2, a routine to translate a historical dataset into a network flow model that well approximates the dataset has been presented. By populating the dataset with aircraft trajectories flown on clear weather days and discarding outliers, the resulting network flow model accounts for the well-traveled routes. For use on a day when convective weather is predicted, additional routes may be required to plan the flow with the additional capacity constraints imposed by the weather.

As example, consider that the clustering process identified a single aggregate route for a specific O-D airport pair, as shown in Fig. 3.1. If such route is blocked by a convective weather front, aircraft will be held on the ground as long as necessary to avoid the front. For a static convective weather front, aircraft will be issued a ground stop, since no feasible departure time exists (see Appendix B).

Our modeling approach has two parts: we start with a model for clear weather days and, for a particular day with convective weather, we add routes as appropriate. In this chapter a routine is presented that, given scheduled departures, identifies those scheduled departures that might incur prolonged ground holding. The original routes for such departures are

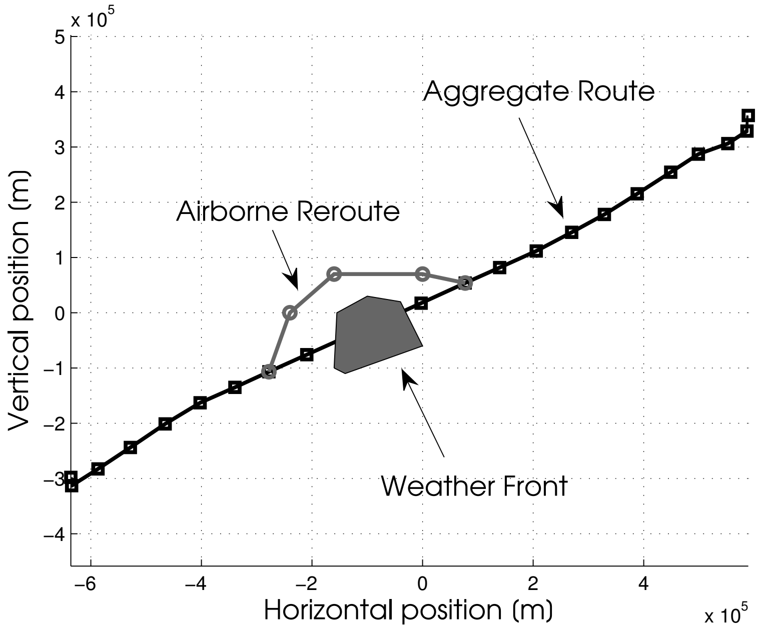


Figure 3.1: Graphical representation of the airborne rerouting design process.

modified with airborne reroutes that avoid the reduced-capacity regions, thus providing additional options for the strategic planning.

In the chapter, we show in Sec. 3.1 how an aggregate route is paired with a search space, i.e., a set of additional nodes and edges that define operationally feasible rerouting options. Then, in Sec. 3.2 we describe the weather data we process and use when designing airborne reroutes. Last, in Sec. 3.3 we describe how a shortest path problem is solved that outputs the best airborne rerouting option for those aggregate routes impeded by reduced-capacity regions. That option will be added to the network flow model to increase its capabilities.

3.1 Design of Operationally Feasible Rerouting Options

For each aggregate route, a search space where rerouting options are searched needs to be identified. This is a region around the original route (defined by some physical parameters as shown in the following), where airborne reroutes will be designed and computed. One driver, when designing an airborne rerouting path, is that it should not deviate too much from the planned route. The choice of the physical parameters for the search space has been designed accounting for this need. Otherwise, options such as pre-departure rerouting or contingency routes from the Playbook[1] might be preferred.

References [74, 75, 76] propose a method for generating operationally acceptable reroutes, where the search area is an ellipse containing the allowable set of nodes and edges. The semi-major axis is the great circle arc connecting the route deviation point (fix along the original route where reroute can begin) and the route rejoin point (fix along the original route where the reroute reconnects to the original route), plus two search buffers (one for each point) to increase the search options. A similar reasoning is applied to select the semi-minor axis [75]. Within this search area, optimal sequences of waypoints avoiding weather fronts are determined by a k-shortest algorithm [30]. One driver in the selection of the sequence of waypoints is the flow factor (i.e., how often the edge connecting two waypoints has been historically used), to ensure reroutes are operationally acceptable.

In our case, we use a similar strategy, which is modified to be consistent with the aggregate model for clear days presented in Chap. 2. In this section we show how the search space is generated, and how nodes and edges within the search space are computed. The set of original nodes and edges defining the original aggregate route, and the set of additional nodes and edges within the search space define together an augmented graph. In the augmented

[1] `https://www.fly.faa.gov/PLAYBOOK/pbindex.html`

graph, aircraft can leave and rejoin the initial route during the cruise phase via an airborne rerouting maneuver. The adjacency properties of the augmented graph are first regulated by a position-reachability requirement, that guarantees a motion towards the destination point and is compatible with speed limitations. Then, another operational constraint addressing heading is introduced.

In Sec. 3.1.1, the framework where the problem is treated for each aggregate route is presented. Then, Sec. 3.1.2 defines the search space. Section 3.1.3 describes how the set of additional nodes and edges is generated. Finally, Sec. 3.1.4 defines how adjacency properties between the nodes are computed.

3.1.1 Conversion of an Aggregate Route using a Lambert Conformal Conic Projection

In our context, where we are looking for "local" deviations around a nominal path, we use a framework based on the Lambert conformal conic projection. We project each aggregate route onto a 2D plane (i) that contains the region where airborne rerouting is designed, and (ii) where possible paths that avoid convective weather regions are searched. Note that, when using the projection, we slightly lose accuracy in terms of distances and angles, but we define an easier framework in which to pose our airborne rerouting problem.

Given a sequence of latitude-longitude pairs that define an aggregate route, we initially translate that sequence into a sequence of horizontal and vertical positions (x and y respectively) using the Lambert conformal conic projection transformation

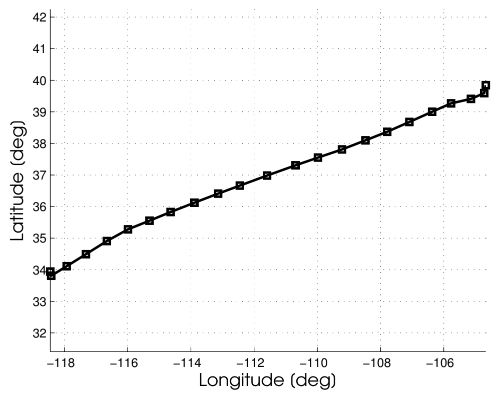
$$\begin{aligned} x &= \rho \sin[n(\lambda - \lambda_0)] \\ y &= \rho_0 - \rho \cos[n(\lambda - \lambda_0)] \end{aligned} \tag{3.1}$$

where

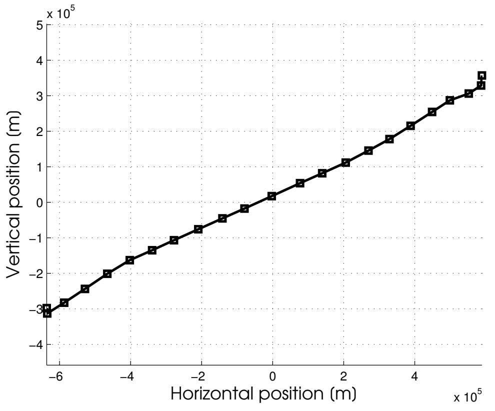
$$\begin{aligned} n &= \frac{\ln(\cos\phi_1 \sec\phi_2)}{\ln[\tan(\frac{1}{4}\pi + \frac{1}{2}\phi_2)\cot(\frac{1}{4}\pi + \frac{1}{2}\phi_1)]} \\ \rho &= F \cot^n\left(\frac{1}{4}\pi + \frac{1}{2}\phi_2\right) \\ \rho_0 &= F \cot^n\left(\frac{1}{4}\pi + \frac{1}{2}\phi_0\right) \\ F &= \frac{\cos\phi_1 \tan^n(\frac{1}{4}\pi + \frac{1}{2}\phi_1)}{n} \end{aligned} \tag{3.2}$$

λ is the longitude, while ϕ is the latitude. The four parameters for the transformation are λ_0, ϕ_0 (reference longitude and latitude), ϕ_1 and ϕ_2 (standard parallels). For λ_0 we choose the mean between the minimum and maximum longitude of the aggregate route, while for ϕ_0 we choose the mean between the minimum and maximum latitude of the aggregate route. In this way, we roughly center the origin of the Lambert projection in the center of the patch defined by the minimum and maximum longitudes and latitudes characterizing the aggregate route. For the two standard parallels, we define an offset $\eta = 0.5\ deg$ and select $\phi_1 = \min(\phi) - \eta$ and $\phi_2 = \max(\phi) + \eta$.

Using Eq. 3.1 as provided, Lambert coordinates refer to a sphere with unitary radius. We use a multiplying factor to project points onto a surface with an altitude of $35,000\ ft$ above Earth's surface. Figure 3.2 shows an aggregate route from LAX to DEN in the original spherical coordinates, and in the Lambert conformal conic projection coordinates.



(a) Spherical coordinates.



(b) Lambert conformal conic projection coordinates.

Figure 3.2: Aggregate route from LAX to DEN in spherical coordinates and in the Lambert conformal conic projection coordinates.

3.1.2 Definition of the Search Space

Similarly to [74, 75, 76], we define a search space where to restrict the design of the airborne reroute. Our search space lies on the Lambert plane, and is identified by two geometrical dimensions, namely a longitudinal extension and a lateral deviation. The longitudinal extension is the equivalent of the major axis in [74, 75, 76]. It identifies the first node of the aggregate route where aircraft are allowed to leave the route, and the last node where aircraft can rejoin the route. The lateral deviation is the equivalent of the minor axis in [74, 75, 76]. It identifies the lateral extension of the search space. As a consequence, it poses an upper limit to the lateral deviation an aircraft is allowed when performing an airborne rerouting maneuver.

To identify the longitudinal extension of the search space, we study the altitude profile of the aggregate route and identify the cruise phase. We identify the cruise phase as the set of nodes between the first node characterized by a percentile change in altitude (with respect to the previous node) smaller than 15% and the last node with a percentile change in altitude smaller than 15%. As [53] suggests, airborne rerouting is a maneuver that is mainly carried out during the cruise phase of a flight, thus we want to remove take off and landing from the search space. Reference [49] uses a similar strategy, designing reroutes around weather polygons as horizontal maneuvers. For the same route shown in Fig. 3.2, the altitude profile is shown in Fig. 3.3. The cruise phase is identified by the set of nodes within the dashed lines. Note that the definition of cruise phase is not standardized. References [26, 54], as example, define the cruise phase as the set of data points above 25,000 ft. The two definitions would provide the same output, if applied to the altitude profile shown in Fig. 3.3, but the output might be different in other circumstances.

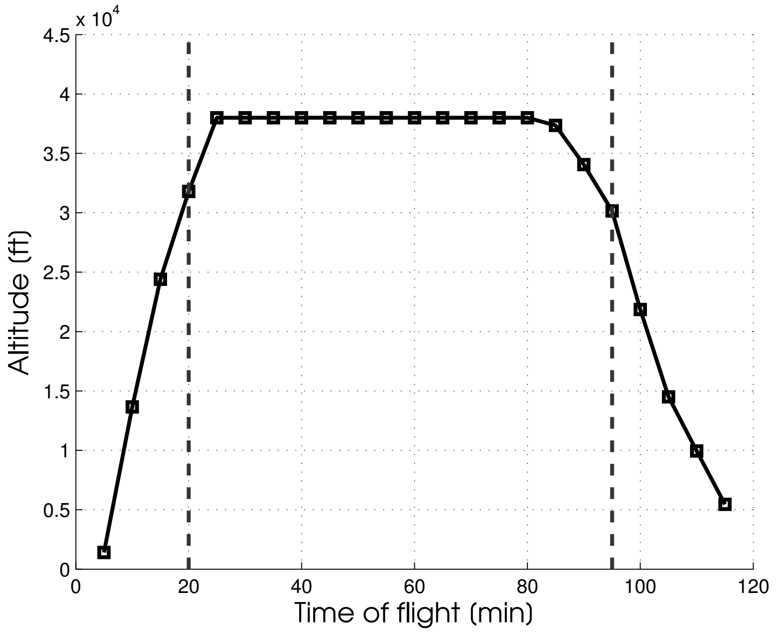


Figure 3.3: Altitude profile for the aggregate route shown in Fig. 3.2.

The longitudinal extension of the search space is the line between the first and last node of the cruise phase (labeled respectively c_{in} and c_{fin}), whose distance is d_c. For the transversal extension of the search space, we consider the two half-planes above and below the longitudinal line defined before. For each half plane, we identify the node in that half plane with the maximum distance with respect to the longitudinal line, if such a node exists. We also define an additional buffer ϵd_c, with $\epsilon < 1$ (we use $\epsilon = 0.1$). In each half-plane, the transversal extension of the search space is the summation of the buffer ϵd_c and the maximum distance previously computed. Naming d_{up} and d_{down} the distances for the upper and lower half-planes, respectively, the transversal extension of the search space is $2\epsilon d_c + d_{up} + d_{down}$. If the route completely belongs to one of the two half-planes only the associated distance will appear in the previous expression. In this way, we have a transversal extension for the search space that is not symmetrical with respect to the longitudinal line. If the route mainly develops in one of the two half-planes, the search space will also mainly develop in that half-plane. The search box is positioned onto the aggregate route as follows. Two transversal segments are placed perpendicularly to segment d_c, one passing through c_{in} and one passing through c_{fin}. The extension of the segments in the upper and lower half-plane follows the rationale mentioned above. Two longitudinal segments with a length of d_c complete the box (see Fig. 3.5 for an example of search space).

3.1.3 Additional Nodes Computation

We now describe how to generate a set of auxiliary nodes within the search space. For each node of the aggregate route inside the search box, we define a set of directions the aircraft might follow, instead of moving onto the next node of the original route. Given the heading of the current node, a conic region is defined that encompasses a range of feasible heading angles. This approach is consistent with the one described in [43] that is based on a similar strategy. Given the current heading angle ψ and a maximum heading angle increase/decrease $\Delta\psi$, the feasible range for the new heading angle will span from $\psi - \Delta\psi$ to $\psi + \Delta\psi$. In practice, the aircraft performs a banked turn to change its current heading, and then follows the direction dictated by the new heading as long as it remains inside the search box.

In this work, the goal is not to design maneuvers such as a banked turn with precision. In analogy to [67], our aim is to generate a coarse rerouting alternative that fits the strategic planning framework. At the same time, as done in [43] and [7], we want to design reroutes to be operationally acceptable. To pursue this goal, we first introduce the banked turn equations that will be used to compute a reasonable range for $\Delta\psi$

$$R_c = \frac{V^2}{g \tan\theta} \tag{3.3}$$

$$\omega = \frac{g \tan\theta}{V} \tag{3.4}$$

$$n = \frac{1}{\cos\theta} \tag{3.5}$$

where R_c is the curvature radius when performing the maneuver, V is the current ground speed, θ is the bank angle, ω the angular velocity and n the load factor. In practice, we want to limit n, which implies limiting the bank angle θ. Although in emergency scenarios, maneuvers with $\theta \geq 50$ *deg* might be necessary, our goal is to design safe reroutes around convective weather polygons. For commercial aircraft, 30 *deg* define a upper bound for a banked turn, but we want to further limit the banked turn in this context. As a consequence, we will tune our parameters to avoid scenarios where $\theta \geq 10$ *deg*.

Considering a time-step $\Delta t = 5\ min$ for the model, a change in heading of $\Delta\psi = 45\ deg$ would imply an angular velocity of $\frac{\pi}{4\Delta t} = 0.00262\ rad/s$. As example, with a speed of $380\ kts = 195\ m/s$, the resulting bank angle (using Eq. 3.4) is $\theta = 3\ deg$, which is consistent with our limitations. Considering ranges on speed and bank angle, a $\Delta\psi = 45\ deg$ is chosen in the design of the feasible directions.

Within the $2\Delta\psi$ heading angle range, a set of n_d different branches can be specified. The higher the number, the denser the resulting graph will be. More rerouting options will be available, but at the expense of an increased computational cost. Along each selected branch, new nodes are generated moving forward along the current direction using a spatial step of $V_m\Delta t$, where $V_m \in [V_{min} V_{max}]$ with $V_{min} = \eta_{min} V_c$ and $V_{max} = \eta_{max} V_c$. V_c is the average speed of the cruise phase, while η_{min} and η_{max} are selected to be respectively 0.6 and 1. Each new point is generated by randomly selecting a speed within the provided interval. Without the same mathematical rigor, this approach mimics the core idea behind a Rapidly-exploring Random Tree [44, 45, 46, 47]. The idea behind the choice of η_{min} and η_{max} is that, when performing an airborne rerouting maneuver, the aircraft will either keep the planned cruise speed or slow down. Figure 3.4 shows an example with $\Delta\psi = 30\ deg$ and $n_d = 14$.

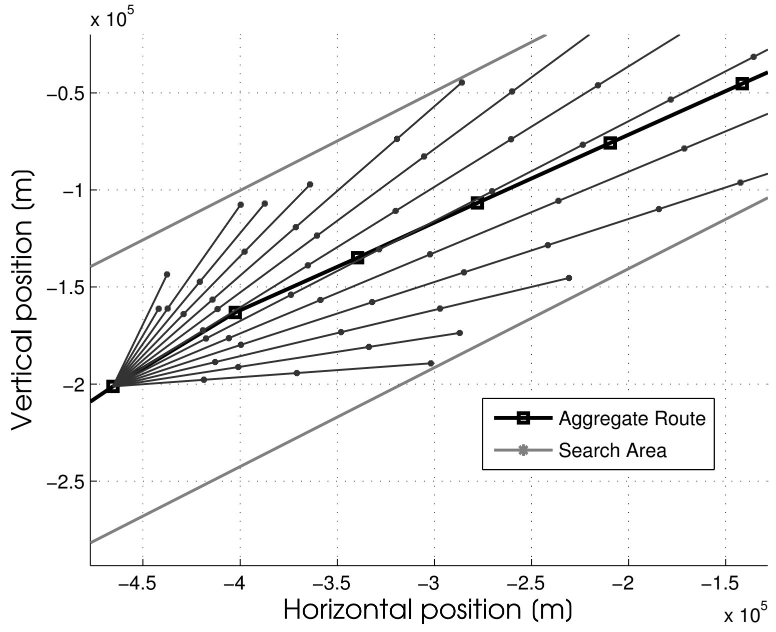


Figure 3.4: Set of feasible directions from a node of the original aggregate route.

For each node of the aggregate route inside the search box, a similar process to the one shown in Fig. 3.4 is repeated. Figure 3.5 shows the full set of rerouting nodes for the same aggregate route introduced before. When generating the rerouting nodes, an additional conic constraint is posed. For the last node of the cruise phase (i.e., the last node where the rerouting branch can rejoin the original aggregate route), the same conic constraint limiting the range of feasible directions is imposed. This is highlighted by the lack of nodes in the two right corners of the search box. In fact, in those regions rejoining the original aggregate route would violate our heading-feasibility requirement. This conic constraint can also be found in [43].

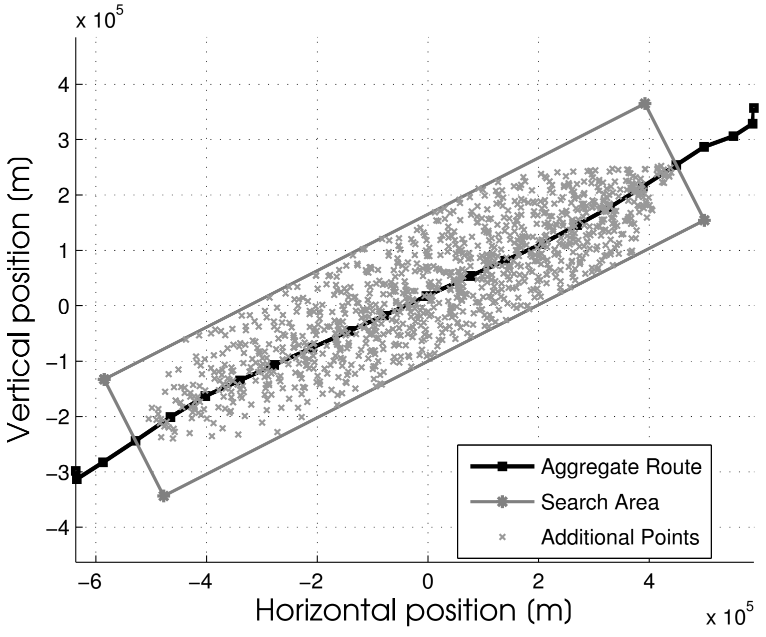


Figure 3.5: Geometry of the search area, and additional nodes that define the augmented graph. The two conic constraints at c_{in} and c_{fin} can be identified.

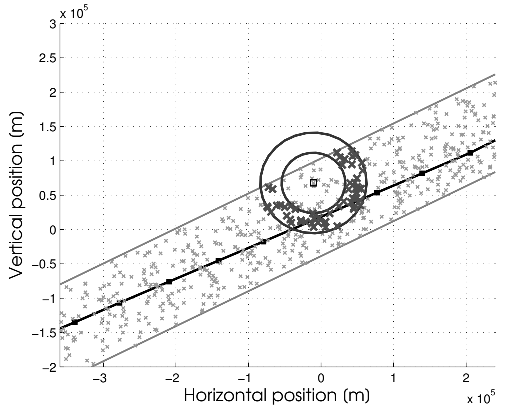
3.1.4 Computation of the Adjacency Properties of the Augmented Graph

Once the set of auxiliary nodes described in Sec. 3.1.3 is computed, the next step is to characterize the adjacency properties of the augmented graph. I.e., we want to determine which nodes are reachable from the each node of the augmented graph. One reachable node is the downstream node along the original aggregate route (if the current node belongs to the original route), or the downstream node along the branch the current node belongs to (see Fig. 3.4). Generally, some other nodes will be reachable, as we describe in the following. Consider an aggregate route with N_r nodes, and N_a auxiliary nodes inside the search box, so that we have a graph with $N_r + N_a$ nodes. In order to solve the rerouting problem, we need to map the connections (i.e., the adjacency properties) between the $N_r + N_a$ nodes. In the original aggregate route, the adjacency properties are such that each node is only reachable from the previous node of the route. When introducing the auxiliary nodes, we want to guarantee feasibility ensuring two requirements. For each node within the search box, the set of nodes that are reachable from the current node satisfies (i) a position-reachability requirement (ii) a heading-reachability requirement, similarly to [11, 43]. While requirement (i) does not depend on the actual sequence of nodes of the modified path, requirement (ii) does.

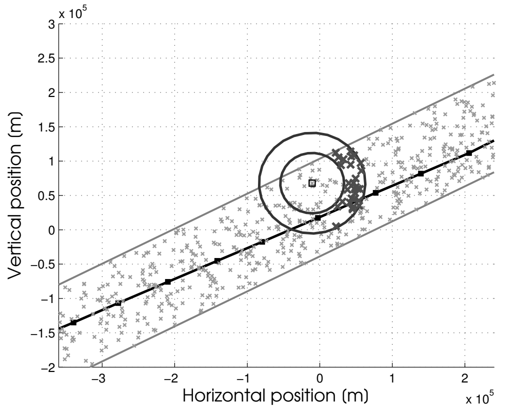
To satisfy requirement (i), for each node an annulus centered in the node is created. The inner radius is $V_{min}\Delta t$, and the outer radius is $V_{max}\Delta t$. All the nodes that are inside such annulus are position-reachable, since they are within the defined range of speeds and, as a consequence, of distances. An example of such concept is shown in Fig 3.6(a). To discard nodes that are position-reachable from the current node, but that define a path that is operationally unlikely, the search space is divided into two planes, where the dividing line is perpendicular to d_c and passes through the current node. Among the nodes initially labeled as position-reachable, only the ones belonging to the half-plane containing c_{fin} are kept. This procedure is motivated by the fact that backtracking is generally avoided when flying (unless a holding pattern is carried out, which is not the goal of this routine). Figure 3.6(b) shows how the initial position-reachable set of points of Fig. 3.6(a) is reduced to become the actual position-reachable set. From a graph perspective, an edge will connect the current node to each of the position-reachable nodes identified.

To satisfy requirement (ii), a more involved procedure is necessary. In fact, to ensure that a position-reachable node is also heading-reachable from the current node, the incoming edge to the current node must be known, i.e., heading-reachability depends on the path (intended as sequence of nodes) the aircraft is following. In practice, given a sequence of three nodes N_1, N_2, N_3 along the path, the heading change between the edge connecting N_1 and N_2 and the edge connecting N_2 and N_3 should be less or equal than the threshold $\Delta\psi$. How this condition is actually verified is described in Sec. 3.3.

In Fig. 3.7, the heading-reachability requirement is visually described. Node 448 is position-



(a) Before the half-plane constraint limitation.



(b) After the half-plane constraint limitation.

Figure 3.6: Position-reachable nodes for a given node in the search area.

reachable from both node 133 and node 192. while node 283 is position-reachable from node 448. The sequence of nodes 133-448-283 is also heading-reachable, since the heading change between the two edges is less than the threshold $\Delta\psi$. The sequence of nodes 192-448-283 is not heading-reachable, since the heading change between the two edges is more than the threshold $\Delta\psi$.

3.2 Convective Weather Data

Section 3.1 described a routine that, given an aggregate route, computes a set of auxiliary nodes and edges that form an augmented graph. The overall goal is to move along the augmented graph, from its origin node (first node of the aggregate route) to its destination node (last node of the aggregate route) in the most efficient way. Two major aspects must be addressed at this point, (i) how do the adjacency properties of the augmented graph change due to weather-related constraints, and (ii) how "efficiency" is evaluated when moving along the graph. In this section, we introduce the type of weather information which will be processed and used, while how to "efficiently" move along the graph is described in Sec. 3.3.

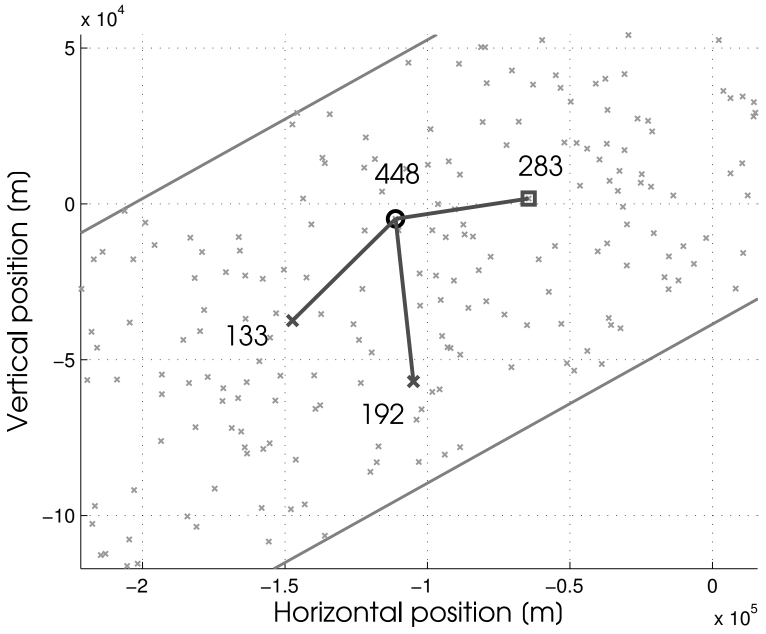


Figure 3.7: Example of two sequences of nodes, one that is not heading-reachable and one that is heading-reachable.

3.2.1 Convective Weather Avoidance Model

References [17, 26, 54] describe a model, called Convective Weather Avoidance Model (CWAM), that quantifies the effect of convective weather on filed flight plans. CWAM is based on the Corridor Integrated Weather System (CIWS) model [32, 40], which uses Vertically Integrated Liquid (VIL) data [6] and echo top data [68]. VIL is an estimate of the total mass of precipitation in the clouds, while an echo top is the radar indicated top of an area of precipitation. CIWS combines the two inputs to compute a 2-hour NAS-wide convection forecast, updated every 5 minutes and with a 5-minute forecast time-step.

CWAM processes CIWS, and outputs a set of non-convex polygons, with a look-ahead time of 2 hours and 5-minute forecast time-step (compatible with CIWS). For each forecast, polygons are provided for every flight level from 250 to 450, with a step of 10. Given a forecast time and flight level, three sets of polygons are provided, referring to an avoidance percentage of 60, 70, 80% respectively. For a polygon with a 60% percentage, 60% of the pilots would avoid such convective weather polygon, if it impedes the filed trajectory.

Each convective weather cell generated by CIWS is surrounded by 3 distinct polygons, corresponding to the three thresholds. The 60% polygon contains the 70% polygon, which contains the 80% polygon. Figure 3.8 shows an example of CWAM polygons for the three thresholds. The CIWS weather forecast is not reproduced, because CIWS data were not available for this research. Readers are referred to [55] for examples of CIWS polygons.

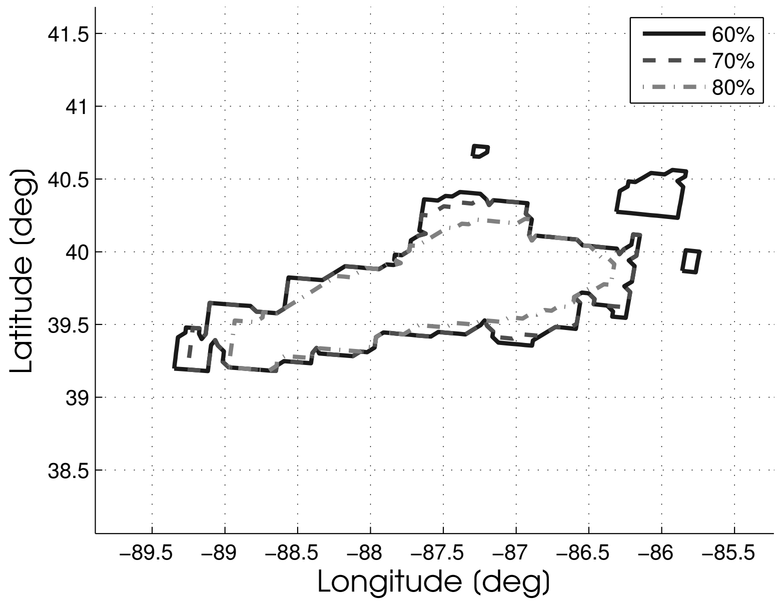


Figure 3.8: Example of CWAM polygons for the three deviation thresholds 60, 70, 80%.

Note that the forecast look-ahead time of 2 hours makes CWAM polygons makes this product most useful for planning short haul flights. Using a planning horizon that exceeds this limit (which is generally the case for strategic planning), would involve the selection of a weather product more appropriate for strategic planning of long haul flights. At the same time, the accuracy level of weather forecasts that go beyond two hours generally decays considerably.

Other weather data and models are available that cover a longer horizon, and that would fit longer planning horizons. In practice, CWAM polygons are generally used to design airborne reroutes (especially at a tactical level). In this work, we will use historical CWAM data that span time horizons greater than 2 hours, acknowledging that, given the planning horizon we target and the look-ahead time of CWAM files, the product is not compatible with our needs. The method we propose can be applied to any weather forecast product described via polygons. The choice of the most appropriate weather product to use is beyond the scope of this work.

3.2.2 Modeling Assumptions

The first modeling assumption, when processing CWAM data, is that a specific flight level will be considered. This simplification is justified by the fact that the vertical variability of the polygons is small when considering a limited range of flight levels (flight levels associated with the cruise phase, in this case). The flight level selected is FL380, being the most common flight level as Fig. 3.9 shows. Note that, since flight levels are computed assuming an International standard sea-level pressure, FL380 does not necessarily imply a local altitude of 38,000 feet. Additionally, only 60% polygons will be considered, to allow an additional safe-margin around convective weather regions.

The second modeling assumption, is that CWAM polygons are simplified by (i) clustering polygons using a distance threshold, and (ii) approximating each cluster with its convex hull [64, 66]. Although methods to compute non-convex hulls exist [2], in a strategic framework, the design of a reroute that exploits the non-convex portions of CWAM polygons is highly unlikely. This level of precision is left to tactical planning [55].

To approximate CWAM polygons, they are first converted into a x-y coordinate frame as described in Sec. 3.1.1. Then, the centroid of each polygon is computed, distances between centroids are computed and hierarchical clustering as described in Chap. 2 is used. A threshold value of 120 km has been selected as a trade-off between size reduction and accurate approximation of the initial distribution of polygons. Figure 3.10 shows how the

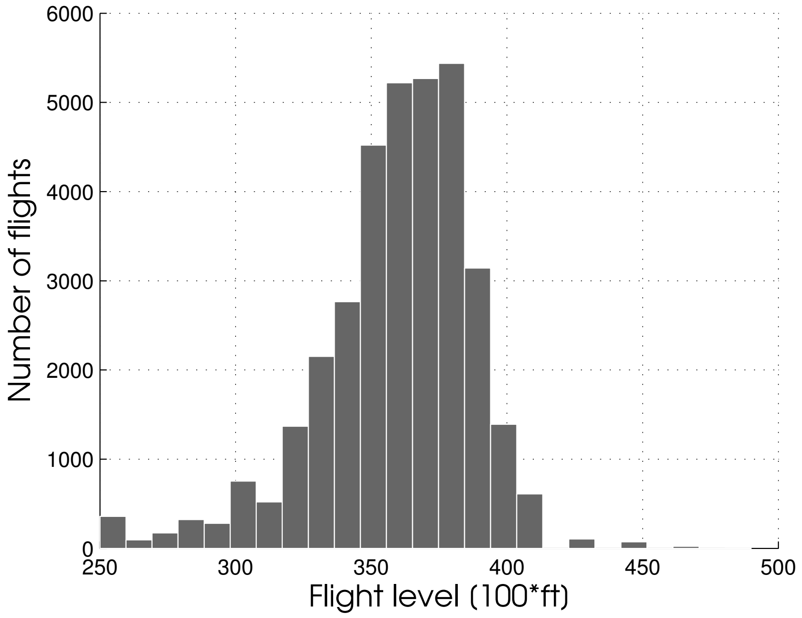
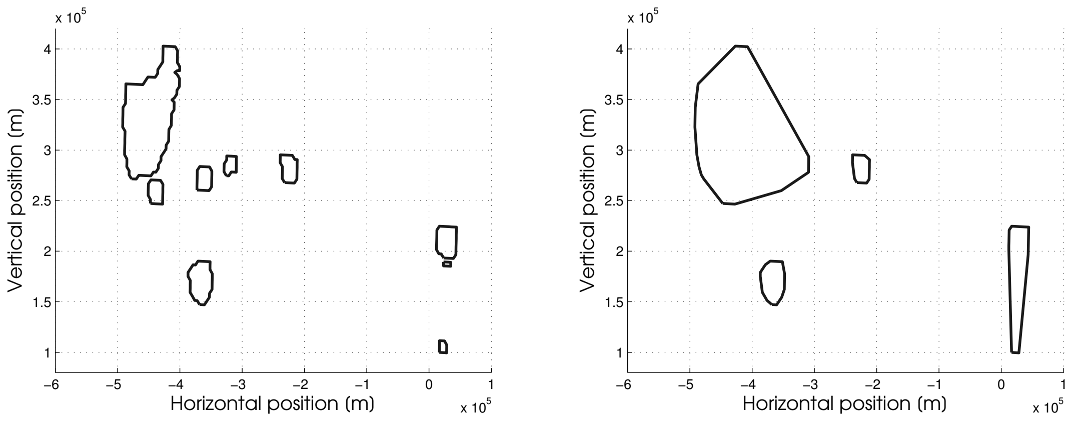


Figure 3.9: Mode of the flight level for the historical dataset of flights described in Chap. 2.

procedure clusters and approximates an initial set of CWAM polygons.



(a) Original non-convex polygons.

(b) Clustered convex polygons.

Figure 3.10: Original CWAM polygons and convex hull approximation.

The process is repeated for any time instance of interest, according to the time-step of the model and remembering that polygons are available with a forecast time of 5 minutes. Thus, if the time-step selected for the model is $\Delta t = 5\ min$, a match between the discrete model and weather forecast already exists, with no need to interpolate data.

3.3 Airborne Rerouting as a Shortest Path Problem

In this section, we will pose the problem of finding a path in the augmented graph that avoids weather polygons as a shortest path problem.

3.3.1 Edge Cost Computation

In related works [67], the airborne rerouting problem is solved using a risk-hedged approach. This means that the solution is a trade-off between the length of the path, and the risk that characterizes the path itself. Generally, the best solution minimizes the total length of the path and either avoids completely or barely enters convective weather polygons with a high probability.

In our case, we use a slightly different approach due to the characteristics of our model and the weather information we processed. Given the discrete nature of the ARM, rather than the physical length of each edge, a natural cost for each edge is the time-step of the model Δt. The main idea is to find a path that minimizes the overall time of flight (intended as a multiple of Δt), while satisfying a set of constraints. These constraints are imposed as soft constraints in the algorithm. The constraints are (i) avoid convective weather polygons, (ii) minimize the portion of path spent outside the original aggregate route, and (iii) impose a path that satisfies the heading-reachability requirement.

The cost of each edge is defined as

$$C(ij,k) = \Delta t(1 + \eta_1 + \eta_2) + \Delta t \eta_3 \tag{3.6}$$

where $C(ij,k)$ defines the cost of edge (i,j) at time t_k. Note that we are solving a problem where obstacles (i.e., convective weather fronts) are not static, i.e., time-variant. This is in contrast with [67, 75], where weather fronts appear to be static.

In Eq. 3.6, η_1, η_2 and η_3 are binary variables. $\eta_1 \in \{0, 1000\}$, depending whether the edge is intersected by a convective weather polygon or not. $\eta_2 \in \{0, 5\}$, and it is 0 if both nodes i and j belong to the original aggregate route, and 5 otherwise. η_1 and η_2 are computed before the shortest path problem is solved, since they only depend on the geometry of the graph. On the other hand, η_3 cannot be assigned a priori, but its computation is embedded in the shortest path algorithm. When moving from the current node to each of the position-reachable nodes, the initial heading angle is known since the path to the current node is known. For each position-reachable node, the heading from the current node to the position-reachable node is compared to the initial heading. If the difference is less than $\Delta\psi$, then $\eta_3 = 0$, otherwise $\eta_3 = 1000$. If we consider Fig. 3.7 again, for edge (448,283), $\eta_3 = 0$ if the node before 448 along the path is 133, $\eta_3 = 1000$ if the node before 448 along the path is 192. Section 3.3.4 will describe more into details how η_3 is computed within the algorithm.

Note that the condition is enforced only for edges that do not belong to the original aggregate route. In fact, for edges of the original route, the condition is generally not met in proximity to airports, where sudden changes in the heading are carried out for take off and landing maneuvers. Without this relaxation, a departure along the original aggregate route that is not affected by convective weather could erroneously be discarded.

Values for η_1, η_2 and η_3 are not fixed, and can be modified according to some specific requirements. As an example, η_2 could even be removed from the cost, since in real scenarios there is no actual penalization when leaving the planned route, if the maneuver is accepted by an ATC. This also leads to an assumption of our model. When designing an airborne rerouting maneuver, we imply such maneuver will be accepted by the ATC in charge. This approach is also used by [67] and [55], where rerouting is not necessarily bounded to follow a specific sequence of waypoints. References [74, 75, 76] are based on a different approach that is waypoint-based and favors edges connecting waypoints characterized by a high traffic volume according to historical data.

Considering η_1 and η_3, their values are considerably higher than η_2 since, if activated, they correspond to scenarios to be avoided. In practice, the activation of η_1, which implies that the path encounters a weather polygon somewhere, might still lead to a flyable route. In fact, (i) we are using approximated polygons that only a percentage of pilots would avoid, and (ii) the path might only slightly intersect the weather front, thus generating a scenario that most pilots would still consider safe. This is a level of accuracy that our model is not able to catch, while it is the main focus of purely tactical works such as [55].

3.3.2 Ground Holding vs. Airborne Rerouting Cost Analysis

A factor to consider when designing an airborne reroute, is how the delay it introduces compares to an equivalent delay on the ground. In fact, there exists a threshold α (adimensional number) such that if the ratio between the ground holding delay and the airborne delay is smaller than the threshold, ground holding is preferred. Literature [8] agrees that $\alpha > 1$, which means that ground holding is preferred when the two options introduce the same delay. How to estimate α is, by itself, a research topic of great interest, since it involves all the direct and indirect costs associated with an aircraft.

For our work, we are looking for a reasonable estimate to justify if the design of an airborne rerouting path is advisable. If convective weather fronts are moving such that a small ground holding would still lead to a clear route, airborne rerouting might not even be necessary.

References [20, 21] provide an exhaustive study that maps costs on the ground and airborne for a set of commercial aircraft under different scenarios. The work is further extended in [31], where three types of delay are considered. They are

- Tactical Delay Without Network Effect (TDWONE), i.e., the delay introduced by a maneuver without considering the reactionary delay caused by other aircraft.
- Tactical Delay With Network Effect (TDWNE), i.e., the delay introduced by a maneuver considering the reactionary delay caused by other aircraft.
- Strategic Delay (SD), i.e., a buffer that is already allocated into nominal schedules in anticipation of delays.

The three types of delay account for costs deriving from fuel, maintenance, crew, passenger handling, airport charges etc. The same cost, computed for a different type of delay, is generally characterized by similar values (expressed in €/min). As example, crew cost are 7.2, 8.6, 8.4 for TDWONE, TDWNE, SD respectively. Differences can be more evident when the same cost, for the same type of delay, refers to a ground or airborne delay. Fuel costs, as example, are 0.1/19.8, 0.2/19.8, 1.3/24 (ground/airborne) for TDWONE, TDWNE, SD respectively. If the aircraft is held on the ground, the extra amount of fuel necessary is very limited. On the other hand, an airborne delay generates a substantial fuel-related cost. Maintenance costs double as well when considering an airborne delay instead of a ground delay. Some costs do not change at all, like passenger compensation, while airport charges apply only to ground delays. Table 3.1 summarizes the overall cost for the three different delay types.

Table 3.1: Overall cost (€/min) for the three delay types TDWONE, TDWNE, SD, divided into a ground and airborne component. (Data from [31]).

	TDWONE		TDWNE		SD	
	Ground	Airborne	Ground	Airborne	Ground	Airborne
Overall cost	53.2	72.9	90.8	110.5	16.7	51.0

The delay type we consider is TDWONE. In fact, our goal is to compare, for a given scheduled departure that is blocked by weather, what is the control form (ground holding or airborne rerouting) that is most efficient. Thus, we want to compare the two controls by weighting the delay introduced by a maneuver without considering the reactionary delay caused by other aircraft, i.e., we want to consider TDWONE as reference. As a consequence, a value of $\alpha = \dfrac{72.9}{53.2} = 1.37$ will be used.

3.3.3 Shortest Path Algorithm Structure

Given a departure from an aggregate route scheduled at t_k, where $k \in [1, 2, \cdots, N_t]$ being N_t the number of discrete time-steps in the planning horizon, our primary goal is to understand if the effect of convective weather is such that the scheduled departure will be delayed consistently. If so, an airborne rerouting maneuver is searched that (i) avoids the convective weather that affects the original route, and (ii) introduces a smaller delay with respect to the ground holding solution.

The process can be summarized with the flowchart depicted in Fig. 3.11. Given an aggregate route with N_r nodes and a scheduled departure at t_k, the shortest path algorithm is first applied to the original route only. If the route is clear, the cost of the path is $(N_r - 1)\Delta t$, i.e., the scheduled time of flight. Otherwise, the same problem is solved for some later departure times t_{k+1}, t_{k+2} etc, and the smallest departure time (if any) such that the cost is $(N_r - 1)\Delta t$ is selected. A heuristic that can be applied is that, if there exists a departure time such that (i) the associated ground holding is "small" enough, and (ii) the aggregate route is now clear, then ground holding is still preferred to airborne rerouting. As example, considering $\Delta t = 5\ min$, a delay of $2\Delta t = 10\ min$ that results in a clear route might be small enough to avoid airborne rerouting. For scheduled departures with a prolonged ground holding (which could even result in a ground stop), an airborne rerouting path is searched.

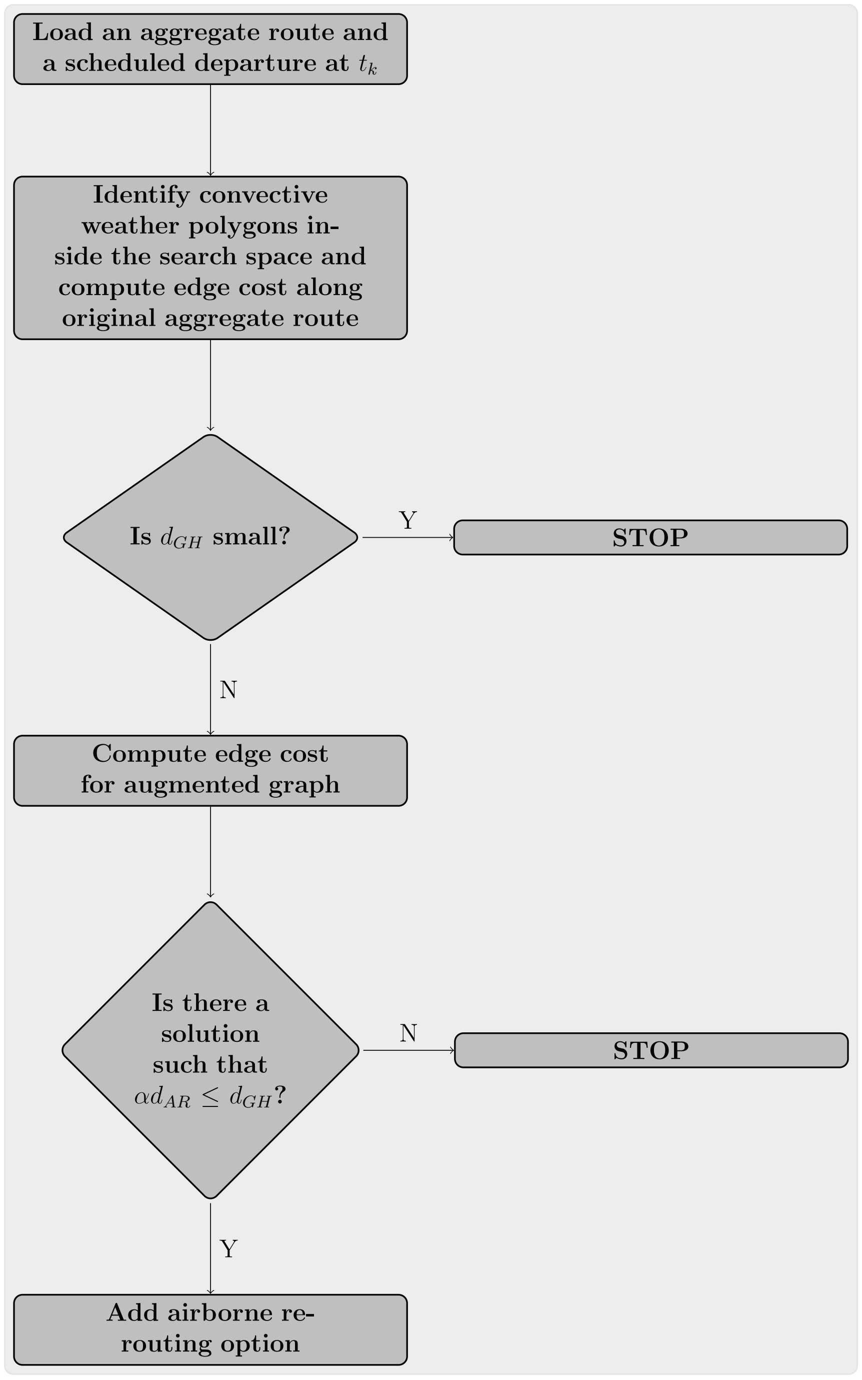

Load an aggregate route and a scheduled departure at t_k
Identify convective weather polygons inside the search space and compute edge cost along original aggregate route
Is d_{GH} small?
Y
STOP
N
Compute edge cost for augmented graph
Is there a solution such that $\alpha d_{AR} \leq d_{GH}$?
N
STOP
Y
Add airborne rerouting option

Figure 3.11: Flowchart describing how the airborne rerouting design problem is addressed.

This procedure has two main objectives. (i) Identify which aggregate routes (and scheduled departures) are not affected by convective weather, and (ii) for aggregate routes that are affected by convective weather, obtain an estimate of the ground holding that would result in an unimpeded path. In particular, for objective (ii) there might not even be a solution with a cost $(N_r - 1)\Delta t$. This is the case if a static or quasi-static convective weather polygon is impeding an aggregate route.

If an airborne rerouting deviation is recommended, the shortest path algorithm is run again, this time applied to the augmented graph. The delay introduced is compared with the ground holding delay that corresponds to an unimpeded path (if existing). Labeling d_{GH} the ground holding delay, and d_{AR} the airborne rerouting delay, if the inequality $\alpha d_{AR} \leq d_{GH}$ is satisfied, then the airborne rerouting option is added to the network flow model. For the airborne rerouting path, the delay introduced (as a multiple of Δt) is easily determined as the difference between the number of nodes of the airborne path and the number of nodes of the original aggregate route.

3.3.4 Shortest Path Algorithm Implementation

For the computation of the shortest path, the A* algorithm [35, 83] is used. This is a variation of the classic Dijkstra algorithm [27], where a heuristic cost is added to solve more efficiently the shortest path problem. With respect to the original algorithm, that generally addresses a time-invariant scenario, two main changes are applied. They account for (i) an edge cost which is time-variant due to the spatial evolution of convective weather polygons, and (ii) the heading-reachability requirement. Both changes are easily applicable given the nature of the algorithm, which stores for each node the best path (and associated cost) from origin to the current node. Given the current active node [27], the length of the path (intended as number of nodes) is used to select the correct time instance for the edge cost computation (i.e., the appropriate k in Eq. 3.6). For each node belonging to the position-reachable set, the active node and the previous node belonging to the stored path are used to verify if the heading condition is satisfied. Labeling $\vec{N}_1$, $\vec{N}_2$, $\vec{N}_3$ the node preceding the active node, the active node and the current position-reachable node, the heading condition is satisfied if

$$\cos^{-1}\left(\frac{\langle\vec{N}_2-\vec{N}_1,\vec{N}_3-\vec{N}_2\rangle}{|\vec{N}_2-\vec{N}_1||\vec{N}_3-\vec{N}_2|}\right)\leq\Delta\psi \tag{3.7}$$

Additionally, a proper heuristic function $h(u)$ needs to be chosen when solving the problem with the A* algorithm. A requirement of the heuristic function, is that it must satisfy the triangular inequality [83]. Given a node u and a node v reachable from node u via the edge (u,v), the triangular inequality

$$h(u)\leq e(u,v)+h(v) \tag{3.8}$$

where $e(u,v)$ is the cost of the edge connecting nodes u and v, must hold. We show that a proper heuristic function is

$$h(u)=\frac{d(u,c_{fin})}{V} \tag{3.9}$$

where $d(u,c_{fin})$ is the distance between the considered node and the last cruise phase node of the original aggregate route, and V is the average cruise speed of the aggregate route. Considering a generic node u, given the framework of the rerouting model, every position-reachable node v (i) has a distance range from u in the interval $[\eta_{min} V\Delta t, \eta_{max} V\Delta t]$, and (ii) belongs to the half-plane containing u and c_{fin}. We can use the law of cosines to express $d(v, c_{fin})$ as

$$d(v, c_{fin})^2 = (\eta V\Delta t)^2 + d(u, c_{fin})^2 - 2\eta V\Delta t d(u, c_{fin}) \cos\theta \tag{3.10}$$

where θ is the angle between $d(u, c_{fin})$ and $e(u, v)$. If inequality 3.8 is satisfied when the right hand side reaches its minimum value possible, then it will hold for any position-reachable node from node u. In Eq. 3.10, $d(v, c_{fin})^2$ is minimized if $\theta = 0$, so that $d(v, c_{fin})^2 = (\eta V\Delta t)^2 + d(u, c_{fin})^2 - 2\eta V\Delta t d(u, c_{fin})$, which implies $d(v, c_{fin}) = d(u, c_{fin}) - \eta V\Delta t$. $d(v, c_{fin})$ is further minimized if $\eta = \eta_{max} = 1$, so that $d(v, c_{fin}) \geq d(u, c_{fin}) - V\Delta t$. Practically, $d(v, c_{fin})$ is minimized if the aircraft moves along the line connecting u and c_{fin} at the maximum speed allowed. At the same time, every edge in the graph has a cost which is no smaller than the time-step of the model Δt. Thus, $e(u, v) \geq \Delta t$.

Proof. Let u, v be two nodes of the graph s.t. v is position-reachable from u. So,

$$\begin{aligned}
h(u) &\leq e(u, v) + h(v) \\
\frac{d(u, c_{fin})}{V} &\leq \Delta t + \frac{d(u, c_{fin}) - V\Delta t}{V} \\
\frac{d(u, c_{fin})}{V} &\leq \Delta t + \frac{d(u, c_{fin})}{V} - \Delta t \\
\frac{d(u, c_{fin})}{V} &\leq \frac{d(u, c_{fin})}{V}
\end{aligned}$$

□

Since the inequality holds when both $e(u, v)$ and $h(v)$ are minimized, the triangular inequality holds for any position-reachable node v from node u, and h is a valid heuristic function for the A* algorithm.

The goal of the algorithm is to find the shortest path (intended as the path with the minimum cost) between nodes 1 and N_r (origin and destination of the original aggregate route), such that convective weather polygons and heading changes exceeding a threshold are avoided.

We show the application of the procedure to the aggregate route from LAX to DEN that has been used throughout the chapter, assuming a departure at 0:00 UTC on 07/07/2011. The aggregate route and search space have already been introduced in Fig. 3.5. CWAM polygons are approximated as described in Sec. 3.2.

The ground holding solution is solved first, and results in terms of cost are shown in Table 3.2. Note that the cost we are considering does not account for the heuristic component $h(u)$. The aggregate route has 23 nodes from origin to destination, which means the cost of an unimpeded flight is $22\Delta t = 6600\ s$, which is the estimated time of flight for the aggregate route.

Table 3.2: Evolution of the path cost (without the heuristic component $h(u)$) of the example presented, for different departure times.

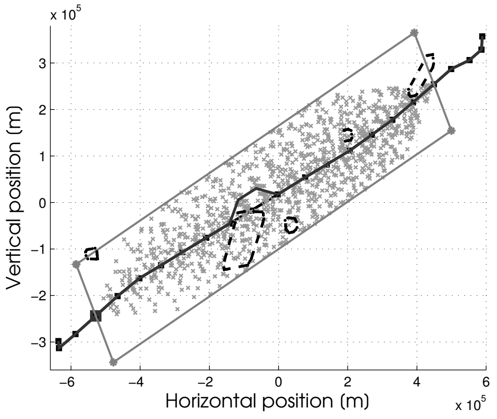
Departure time [UTC]	Cost [s]
0:00	606000
0:05	606000
0:10	306300
0:15	6600

For both the scheduled departure time 0:00 UTC and 0:05 UTC, two edges are impeded along the path. For a departure at 0:10 UTC, one edge is impeded, while a departure at 0:15 UTC would find an unimpeded path. Thus, a ground holding delay of 15 minutes (i.e., $3\Delta t$), is enough not to intersect any convective weather polygon.

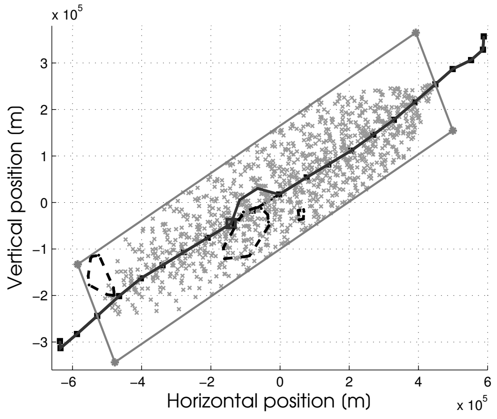
We now solve the same shortest path algorithm applied to the augmented graph. Given a departure at time 0:00, we obtain a solution with a sequence of 24 nodes (of which 2 do not belong to the original route) and a cost of 10500. The cost is the summation of

21 unimpeded edges of the original aggregate route, and of 3 edges that do not belong to the original route, i.e., $21\Delta t + 3\eta_2\Delta t = 6300 + 4500 = 10500$. We can design an airborne rerouting maneuver without pushing back the scheduled departure time, and that introduces an airborne rerouting delay of 5 minutes (i.e., Δt). Considering Fig. 3.11, we need to verify if $\alpha d_{AR} \leq d_{GH}$. In this case, $1.37\Delta t \leq 3\Delta t$ is verified, thus the airborne rerouting option if preferred. For the airborne rerouting solution, Fig. 3.12 shows for different time instances the position of the aircraft and the CWAM polygons within the search space.

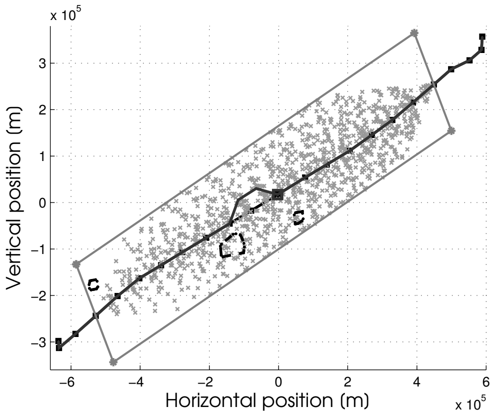
For all scheduled departures with CWAM polygons entering the associated search space, the routine presented in this chapter is carried out. All airborne reroutes paths generated and accepted following the rationale of Fig. 3.11 are added to the ARM baseline that was described in Chap. 2.



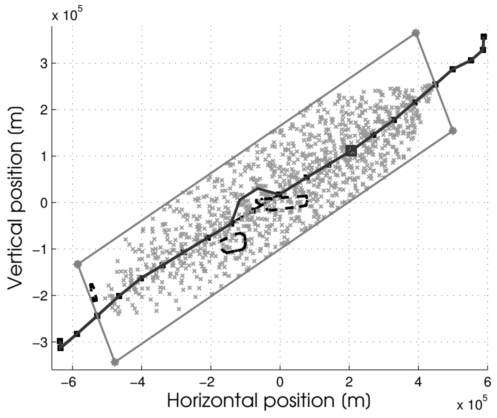
(a) 0:15 UTC.



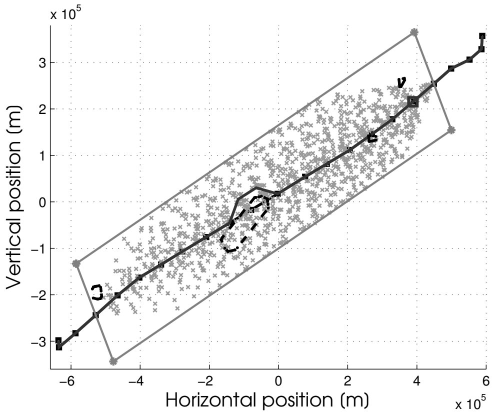
(b) 0:45 UTC.



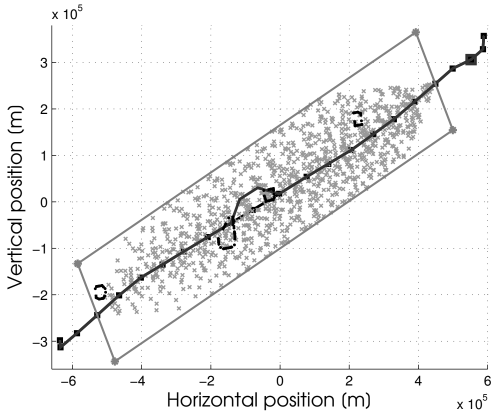
(c) 1:00 UTC.



(d) 1:20 UTC.



(e) 1:30 UTC.



(f) 1:45 UTC.

Figure 3.12: Temporal evolution of the airborne rerouting solution computed.

Chapter 4

Aggregate Route Model as a Discrete Linear Time-Invariant System

In Chap. 2 and Chap. 3, a procedure has been described to compute a network flow representation of the air traffic flow. Each of the $3n_a + 1$ network components is represented with a directed graph $\mathcal{G}_{nc} = (\mathcal{N}_{nc}, \mathcal{E}_{nc})$ [78]. Overall, the graph-oriented approximation of the air traffic flow is a network because of the presence of source and sink nodes, and because of flow limitations. In Sec. 4.1, we describe the dynamics of an uncontrolled network component. Then, in Sec. 4.2, we describe (i) the controls used to modify the scheduled air traffic flow, i.e., ground holding. pre-departure rerouting, and airborne rerouting, and (ii) the matrix form of the controlled dynamics of a network component. We also show the effect of controls on the system with two simple examples. Finally, Sec. 4.3 shows how the dynamics of the network components are assembled together.

4.1 Uncontrolled Traffic Flow Dynamics for a Network Component

The dynamics for each network component are uncoupled from those of the other network components. Because the formulation for the dynamics of each network component is similar, here we only develop the dynamics for a single network component in detail, and then we describe how the different blocks are assembled into the complete network dynamics in Sec. 4.3.

Consider a network component with N_r aggregate routes and total number of nodes $N_n = \sum_{i=1}^{N_r} N_i$, where N_1, N_2, $\cdots$, N_{N_r} are the numbers of nodes on the individual routes. The uncontrolled dynamics take the form of a discrete linear time-invariant system expressed as

$$\begin{aligned}\mathcal{X}_0 &= \mathcal{X}(t_0) \\ \mathcal{X}(t_{k+1}) &= \mathcal{A}\mathcal{X}(t_k) + \mathcal{C}\mathbf{b}(t_{k+1}), \;\; k = 0, 1, \cdots, N_t - 1\end{aligned} \tag{4.1}$$

where $\mathcal{X}(t_k) = [x_1(t_k)\ x_2(t_k)\ \cdots\ x_{N_n}(t_k)]^T \in \mathbb{Z}^{N_n}$ is the state at time t_k. Each state variable defines the number of aircraft in the associated node. Recalling that origins can be airports inside the planning domain or entry points on the boundary of the domain, the scheduled departures at t_{k+1} are represented by the vector $\mathbf{b}(t_{k+1}) \in \mathbb{Z}^{N_r}$. Let $\mathcal{C} \in \mathbb{Z}^{N_n \times N_r}$ denote the constant, integer-valued matrix that maps the scheduled departures at t_{k+1} into the state vector at t_{k+1}. That is, $\mathcal{C}$ places new departing airplanes at the first node (i.e., origin node) of an aggregate route. $\mathcal{A} \in \mathbb{Z}^{N_n \times N_n}$ is the constant, block diagonal, integer-valued system dynamics matrix. It propagates the aircraft, at each node, one node forward on the same route each time-step. Because routes are unidirectional, each diagonal block consists of ones on the subdiagonal and zeros elsewhere, namely

$$\mathcal{A} = \begin{bmatrix} \mathcal{A}_1 & 0 & 0 & \cdots & 0 \\ 0 & \mathcal{A}_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \mathcal{A}_{N_r} \end{bmatrix}, \text{ where } \quad \mathcal{A}_i = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{4.2}$$

is a $N_i \times N_i$ matrix. , Each aggregate route is thus an acyclic rooted tree [1, 24, 42], where (i) there is no aircraft exchange with other aggregate routes, (ii) aircraft cannot occupy the same node more than one time-step, and (iii) aircraft move from the current node to the next one. Given N_t time-steps and an initial state vector $\mathcal{X}_0$, using Eq. 4.1 recursively, the component state vectors for times $t_1, t_2, \cdots, t_{N_t}$ can be expressed as

$$\underbrace{\begin{bmatrix} \mathcal{X}(t_1) \\ \mathcal{X}(t_2) \\ \vdots \\ \mathcal{X}(t_{N_t-1}) \\ \mathcal{X}(t_{N_t}) \end{bmatrix}}_{\mathcal{X}_{nc}} = \underbrace{\begin{bmatrix} \mathcal{A} & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & \mathcal{A}^{N_t-1} & 0 \\ 0 & \cdots & 0 & \mathcal{A}^{N_t} \end{bmatrix}}_{\mathcal{A}_{nc}} \underbrace{\begin{bmatrix} \mathcal{X}_0 \\ \mathcal{X}_0 \\ \vdots \\ \mathcal{X}_0 \\ \mathcal{X}_0 \end{bmatrix}}_{\mathcal{X}_{0nc}} + \underbrace{\begin{bmatrix} \mathcal{C} & \cdots & 0 & 0 \\ \mathcal{A}\mathcal{C} & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ \mathcal{A}^{N_t-2}\mathcal{C} & \cdots & \mathcal{C} & 0 \\ \mathcal{A}^{N_t-1}\mathcal{C} & \cdots & \mathcal{A}\mathcal{C} & \mathcal{C} \end{bmatrix}}_{\mathcal{C}_{nc}} \underbrace{\begin{bmatrix} \mathbf{b}(t_1) \\ \mathbf{b}(t_2) \\ \vdots \\ \mathbf{b}(t_{N_t-1}) \\ \mathbf{b}(t_{N_t}) \end{bmatrix}}_{\mathbf{b}_{nc}} \tag{4.3}$$

Figure 4.1 shows an example of network component with $N_r = 2$ and $N_n = 6$. Route A is characterized by the ordered sequence of nodes 1, 2 and 3. Route B is characterized by the

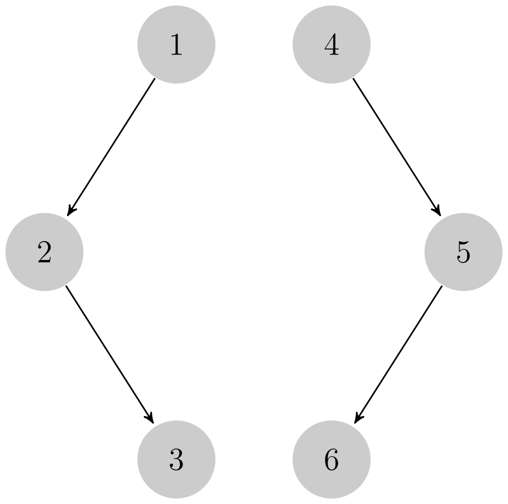


Figure 4.1: Example of network component with two aggregate routes.

ordered sequence of nodes 4, 5 and 6. For this network component, $\mathcal{A}$ and $\mathcal{C}$ are

$$\mathcal{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathcal{C} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.4}$$

$\mathcal{A}$ is block diagonal as expected, while the unitary elements $(1,1)$ and $(4,2)$ of $\mathcal{C}$ place departures in the origin nodes of the respective aggregate routes.

4.2 Traffic Flow Controls for a Network Component

Section 4.1 provides a general formulation that describes how dynamics of the network are modeled and propagated forward in time as a linear function of the initial state vector and the scheduled departures vector. Here, we describe what controls are considered, and their effect on the dynamics. Sections 4.2.1 and 4.2.2 then provide examples to show how the matrices defined in Sec. 4.1 look like. Three controls are considered in our framework. They are (i) ground holding, (ii) pre-departure rerouting, and (iii) airborne rerouting.

Each aggregate route is characterized by ground holding. In practice, some aircraft that are scheduled to take off at a certain time-step, might be held on the ground until the following time-step. From a graph perspective, ground holding introduces a cycle at the origin node of each aggregate route. This also implies that, if we neglect the initial state vector, there always exists a feasible solution to the IP problem, which is a ground stop for all scheduled departures, as Appendix B shows.

If aircraft can be exchanged between two aggregate routes, pre-departure rerouting can be exploited as a control. Unless differently specified, we will consider pre-departure rerouting as bi-directional. From a graph perspective, pre-departure rerouting couples different aggregate routes, which can now exchange aircraft.

Note that ground holding and pre-departure rerouting are defined once the ARM as described in Chap. 2 is computed. All aggregate routes of internal or exiting network components can be controlled with ground holding. For internal aggregate routes, if more than one aggregate route exists connecting the same O-D airport pair, pre-departure rerouting can be exploited. For exiting aggregate routes, the proximity of the boundary crossing nodes is used as a driver to detect if pre-departure rerouting is applicable. For entering aggregate routes and overflights aggregate routes, a control that is equivalent to ground holding, from a graph perspective, can still be applied to origin nodes. In this case, origin nodes do not define airports, but boundary crossing nodes with aircraft in enroute phase. The effect of this control is thus to place aircraft on a holding pattern. Such controls should be weighted differently, since they are applied to origin nodes that are formally airborne, as Chap. 5 describes.

If aircraft are allowed to leave the original aggregate route from a node that is not the origin node and then rejoin the aggregate route, airborne rerouting is the control action applied. From a graph perspective, the effect is similar to a pre-departure rerouting. Some aircraft can be moved from the node of the original aggregate route to the first node of the airborne branch at the current time. Airborne rerouting has three distinctive features. (i) It creates additional nodes that were not part of the original network, thus increasing the size of the network. (ii) It creates merging nodes in correspondence of nodes where a reroute branch rejoins the original aggregate route. From a graph perspective, the adjacency matrix $\mathcal{A}$ will now show non-zero elements out of the block diagonal structure (refer to Sec. 4.2.2 for an example). (iii) Airborne rerouting is not bi-directional, as aircraft can leave the original route and be moved on the reroute branch, but not vice-versa.

We introduce controls in Eq. 4.12 as follows

$$\underbrace{\begin{bmatrix} \mathcal{X}(t_1) \\ \mathcal{X}(t_2) \\ \vdots \\ \mathcal{X}(t_{N_t}) \end{bmatrix}}_{\mathcal{X}_{nc}} = \underbrace{\begin{bmatrix} \mathcal{A} & 0 & \cdots & 0 \\ 0 & \mathcal{A}^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \mathcal{A}^{N_t} \end{bmatrix}}_{\mathcal{A}_{nc}} \underbrace{\begin{bmatrix} \mathcal{X}_0 \\ \mathcal{X}_0 \\ \vdots \\ \mathcal{X}_0 \end{bmatrix}}_{\mathcal{X}_{0nc}} + \underbrace{\begin{bmatrix} \mathcal{C} & 0 & \cdots & 0 \\ \mathcal{A}\mathcal{C} & \mathcal{C} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \mathcal{A}^{N_t-1}\mathcal{C} & \mathcal{A}^{N_t-2}\mathcal{C} & \cdots & \mathcal{C} \end{bmatrix}}_{\mathcal{C}_{nc}} \underbrace{\begin{bmatrix} \mathbf{b}(t_1) \\ \mathbf{b}(t_2) \\ \vdots \\ \mathbf{b}(t_{N_t}) \end{bmatrix}}_{\mathbf{b}_{nc}}$$

$$+ \underbrace{\begin{bmatrix} \mathcal{D}_0 & 0 & \cdots & 0 & 0 \\ \mathcal{D}_1 & \mathcal{D}_0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathcal{D}_{N_t-2} & \mathcal{D}_{N_t-3} & \cdots & \mathcal{D}_0 & 0 \\ \mathcal{D}_{N_t-1} & \mathcal{D}_{N_t-2} & \cdots & 0 & \mathcal{D}_0 \end{bmatrix}}_{\mathcal{B}_{nc}} \underbrace{\begin{bmatrix} \mathbf{u}(t_1) \\ \mathbf{u}(t_2) \\ \vdots \\ \mathbf{u}(t_{N_t}) \end{bmatrix}}_{\mathbf{u}_{nc}} \tag{4.5}$$

where $\mathcal{B}_{nc}$ is a lower triangular block-matrix and each block $\mathcal{D}_{i-j}$ maps the effect of $\mathbf{u}_j = \mathbf{u}(t_j)$ (control vector with N_c elements at time t_j) on the state at time t_i with $t_i \geq t_j$. Each control variable is associated with a nominal path (i.e., the sequence of nodes an aircraft would follow if the control were not active), and with an alternate path (i.e., the sequence of nodes an aircraft would follow if the control is active). In the case of a ground holding, the route is not changed, but the first node is visited twice before proceeding to the second node, which in the network sense is a different path. The value of each element of $\mathbf{u}_j$ is the number of aircraft that should follow the alternate path instead of the nominal one. Consequently every component of the control vector must be non-negative. $\mathcal{B}_{nc}$ is lower triangular because controls cannot affect past states. Each block element $\mathcal{D}_{i-j} \in \mathbb{Z}^{N_n \times N_c}$ is a sparse matrix, with at most a '-1' and a '$+1$' in each column. The -1 location corresponds to the node of the nominal path that can be reached in $i-j$ steps from the node where the control is applied. The $+1$ location corresponds to the node of the alternate path that can be reached in $i-j$ steps from the node where control is carried out.

An explicit formula to calculate $\mathcal{D}_0, \cdots, \mathcal{D}_{N_t-1}$ may be given in terms of the result of control actions on origin nodes. Define two matrices $\mathcal{D}_a, \mathcal{D}_b \in \mathbb{Z}^{N_n \times N_c}$. For both of these matrices, rows that do not correspond to origin nodes of an aggregate route or an airborne rerouting node are uniformly 0. For a row in $\mathcal{D}_a$ that corresponds to an origin node or an airborne rerouting node, the entries are $+1$ in columns that correspond to reroutes *to* that node, and -1 in columns that correspond to reroutes *from* that node or to ground holding at that node. For a row in $\mathcal{D}_b$ that corresponds to an origin node, the only nonzero values are $+1$ in columns that correspond to ground holding at that node.

$\mathcal{D}_0, \cdots, \mathcal{D}_{N_t-1}$ may be generated from $\mathcal{D}_a, \mathcal{D}_b$ and the dynamics matrix $\mathcal{A}$ as $\mathcal{D}_0 = \mathcal{D}_a, \mathcal{D}_1 = \mathcal{A}\mathcal{D}_a + \mathcal{D}_b, ..., \mathcal{D}_{N_t-1} = \mathcal{A}^{N_t-1}\mathcal{D}_a + \mathcal{A}^{N_t-2}\mathcal{D}_b$. This means that $\mathcal{B}_{nc}$ may be expressed in a form similar to that of $\mathcal{C}_{nc}$ above as follows

$$\mathcal{B}_{nc} = \begin{bmatrix} \mathcal{D}_a & 0 & \cdots & 0 & 0 \\ \mathcal{A}\mathcal{D}_a + \mathcal{D}_b & \mathcal{D}_a & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathcal{A}^{N_t-2}\mathcal{D}_a + \mathcal{A}^{N_t-3}\mathcal{D}_b & \mathcal{A}^{N_t-3}\mathcal{D}_a + \mathcal{A}^{N_t-4}\mathcal{D}_b & \cdots & \mathcal{D}_a & 0 \\ \mathcal{A}^{N_t-1}\mathcal{D}_a + \mathcal{A}^{N_t-2}\mathcal{D}_b & \mathcal{A}^{N_t-2}\mathcal{D}_a + \mathcal{A}^{N_t-3}\mathcal{D}_b & \cdots & \mathcal{A}\mathcal{D}_a + \mathcal{D}_b & \mathcal{D}_a \end{bmatrix} \tag{4.6}$$

4.2.1 Network Component Example with Ground Holding and Pre-Departure Rerouting

A network component with two aggregate routes that can exchange aircraft is modeled. The network component is shown in Fig. 4.2.

Route A is characterized by the ordered sequence of nodes 1, 2, and 3, while route B is characterized by the ordered sequence of nodes 4, 5, and 6. For this network component

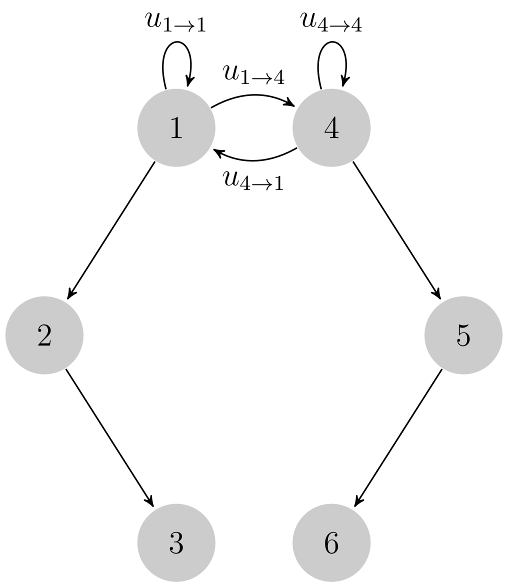


Figure 4.2: Example of network component controlled with ground holding and pre-departure rerouting.

there are four controls per time-step, denoted $u_{1\to1}$, $u_{4\to4}$, $u_{1\to4}$ and $u_{4\to1}$, representing, respectively, ground holding applied to route A, ground holding applied to route B, pre-departure rerouting from route A to route B and pre-departure rerouting from route B to route A. As example, if a flight from route A scheduled to depart at t_1 is ground held, then $u_{1\to1}(t_1) = 1$. If a flight is rerouted from route A to route B, then $u_{1\to4}(t_1) = 1$.

For this network component, matrices $\mathcal{A}$ and $\mathcal{C}$ are the same already shown for the equivalent uncontrolled version in Eq. 4.4, and are not repeated here. Matrices $\mathcal{D}_a$ and $\mathcal{D}_b$ are

$$\mathcal{D}_a = \begin{bmatrix} -1 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathcal{D}_b = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.7}$$

Assuming a null initial state, the state for the two origin nodes x_1 and x_4 for time-steps t_1 and t_2 is

$$\begin{aligned} x_1(t_1) &= b_1(t_1) - u_{1\to1}(t_1) - u_{1\to4}(t_1) + u_{4\to1}(t_1) \\ x_4(t_1) &= b_4(t_1) - u_{4\to4}(t_1) - u_{4\to1}(t_1) + u_{1\to4}(t_1) \\ x_1(t_2) &= b_1(t_2) + u_{1\to1}(t_1) - u_{1\to1}(t_2) - u_{1\to4}(t_2) + u_{4\to1}(t_2) \\ x_4(t_2) &= b_4(t_2) + u_{4\to4}(t_1) - u_{4\to4}(t_2) - u_{4\to1}(t_2) + u_{1\to4}(t_2) \end{aligned} \tag{4.8}$$

and identifies controlled departures from the origin nodes, as opposed to scheduled departures $b_1(t_1)$, $b_4(t_1)$, $b_1(t_2)$, $b_4(t_2)$, respectively. The cumulative effect of ground holding is only evident for controlled departures at time t_2, where maximum aircraft available are now $b_1(t_2) + u_{1\to1}(t_1)$ for route A, and $b_4(t_2) + u_{4\to4}(t_1)$ for route B.

4.2.2 Network Component Example with Ground Holding, Pre-Departure Rerouting and Airborne Rerouting

A network component with two aggregate routes that can exchange aircraft is modeled. In addition, one of the two routes offers an airborne rerouting alternative. The network component is shown in Fig. 4.3.

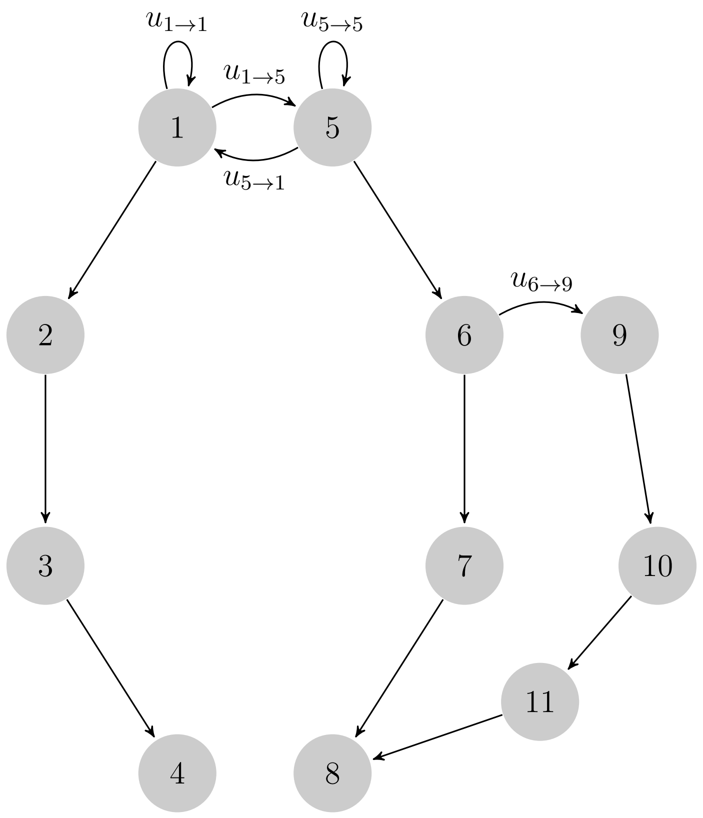


Figure 4.3: Example of network component controlled with ground holding, pre-departure rerouting and airborne rerouting.

Route A is characterized by the order sequence of nodes 1, 2, 3 and 4. Route B is characterized by the ordered sequence of nodes 5, 6, 7, 8. In addition, airborne rerouting is possible from node 6. From this node, aircraft can move from node 6 at t_k to node 10 at t_{k+1}. Node 9 is an auxiliary node, where aircraft rerouted from node 6 at t_k are "instantaneously" moved. The airborne rerouting route is thus characterized by the order sequence of nodes 9, 10, 11 and 8. Node 8 is now a merging node, that receives flow both from node 7 and node 11.

Network component matrices are slightly modified with respect to the case without airborne rerouting presented in Sec. 4.2.1. For each newly introduced airborne branch, some additional nodes are added to the original network, which will increase in size. Additionally, given the presence of some merging nodes (i.e., nodes of the original network where airborne branches reconnect to the original network), some elements of the adjacency matrix outside the block diagonal blocks will not be zero any longer. For the network shown in Fig. 4.3, matrices $\mathcal{A}$ and $\mathcal{C}$ are shown in Eq. 4.9. In $\mathcal{A}$, the three diagonal blocks mapping the node sequences 1-2-3-4, 5-6-7-8, 9-10-11 are identifiable. In addition, element $(8, 11)$ is now non-zero, since the airborne branch and route B merge in node 8. In $\mathcal{C}$, the only non-zero elements are elements $(1, 1)$ and $(5, 2)$, that represent the two origin nodes.

Matrices $\mathcal{D}_a$ and $\mathcal{D}_b$ are also similar with respect to the ones introduced in Sec. 4.2.1, and are shown in Eq. 4.10. Apart from the increment in the overall number of rows to account for the extra nodes, the structure of the first four columns of $\mathcal{D}_a$ is unchanged. This is because the four controls $u_{1\to1}$, $u_{5\to5}$, $u_{1\to5}$ and $u_{5\to1}$ affect the network in the same way the equivalent controls affected the network component shown in Sec. 4.2.1. In this case, a fifth column is added to account for airborne rerouting from node 6 to node 9 $u_{6\to9}$. Note that the structure of the control is identical to a pre-departure rerouting. Aircraft are exchanged "instantaneously" between node 6 and node 9, which can be regarded as an artificial origin node.

For this network component, we focus our attention on the state of nodes 6, 8 and 9 at time-steps t_1, t_2, t_3 and t_4, as shown in Eq. 4.11. We also consider the presence of an initial state vector. For node 6, the state is the algebraic summation of the state transmitted from the origin node 5 and of the state of node 9, i.e., aircraft that are airborne rerouted. The state of node 9 at time t_k is simply $u_{6\to9}(t_k)$: this guarantees conservation of flow entering and exiting node 6.

For node 8, i.e., the merging node, state at t_{k+1} is the summation of the states of nodes 7 and 11 at t_k. At t_1 and t_2, controls do not affect the state. At t_3, the potential effect of airborne rerouting is evident due to the negative term $-u_{6\to 9}(t_1)$. At t_4, the aforementioned negative term appears as positive (due to the different lengths of paths 6-7-8 and 9-10-11-8), in addition to the new negative term $-u_{6\to 9}(t_2)$. In addition, at t_4 controls at origin nodes have propagated until node 8, as controls $u_{5\to 5}(t_1)$, $u_{1\to 5}(t_1)$, $u_{5\to 1}(t_1)$ in the expression of $x_8(t_4)$ confirm.

$$\mathcal{A} = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0 \end{bmatrix}, \quad \mathcal{C} = \begin{bmatrix} 1&0 \\ 0&0 \\ 0&0 \\ 0&0 \\ 0&1 \\ 0&0 \\ 0&0 \\ 0&0 \\ 0&0 \\ 0&0 \\ 0&0 \end{bmatrix} \tag{4.9}$$

$$\mathcal{D}_a = \begin{bmatrix} -1 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathcal{D}_b = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.10}$$

$$
\begin{aligned}
x_6(t_1) &= x_5(t_0) - u_{6\to 9}(t_1) \\
x_8(t_1) &= x_7(t_0) + x_{11}(t_0) \\
x_9(t_1) &= u_{6\to 9}(t_1) \\
x_6(t_2) &= b_5(t_1) - u_{5\to 5}(t_1) + u_{1\to 5}(t_1) - u_{5\to 1}(t_1) - u_{6\to 9}(t_2) \\
x_8(t_2) &= x_6(t_0) + x_{10}(t_0) \\
x_9(t_2) &= u_{6\to 9}(t_2) \\
x_6(t_3) &= b_5(t_2) + u_{5\to 5}(t_1) - u_{5\to 5}(t_2) + u_{1\to 5}(t_2) - u_{5\to 1}(t_2) - u_{6\to 9}(t_3) \\
x_8(t_3) &= x_5(t_0) + x_9(t_0) - u_{6\to 9}(t_1) \\
x_9(t_3) &= u_{6\to 9}(t_3) \\
x_6(t_4) &= b_5(t_3) + u_{5\to 5}(t_2) - u_{5\to 5}(t_3) + u_{1\to 5}(t_3) - u_{5\to 1}(t_3) - u_{6\to 9}(t_4) \\
x_8(t_4) &= b_5(t_1) - u_{5\to 5}(t_1) + u_{1\to 5}(t_1) - u_{5\to 1}(t_1) + u_{6\to 9}(t_1) - u_{6\to 9}(t_2) \\
x_9(t_4) &= u_{6\to 9}(t_4)
\end{aligned}
\tag{4.11}
$$

4.3 Controlled Network Dynamics

The extension of Eq. 4.5 to multiple network components is straightforward. Given the assumptions of our model, the dynamics of different network components are uncoupled (coupling will be present in some constraints, as described in Chap. 5). As a consequence, full states $\mathcal{X}_{nc}$ of different network components can be stacked vertically. For the right hand side of the matrix equation, vectors $\mathcal{X}_{0nc}$, $\mathbf{b}_{nc}$, $\mathbf{u}_{nc}$ will be stacked vertically as well, while matrices $\mathcal{A}_{nc}$, $\mathcal{C}_{nc}$, $\mathcal{B}_{nc}$ will be assembled in block diagonal fashion. The states of the overall controlled network are described by Eq. 4.12

$$\underbrace{\begin{bmatrix} \mathcal{X}_{nc,1} \\ \mathcal{X}_{nc,2} \\ \vdots \\ \mathcal{X}_{nc,3n_a+1} \end{bmatrix}}_{\mathcal{X}_n} = \underbrace{\begin{bmatrix} \mathcal{A}_{nc,1} & \cdots & 0 \\ 0 & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & \mathcal{A}_{nc,3n_a+1} \end{bmatrix}}_{\mathcal{A}_n} \underbrace{\begin{bmatrix} \mathcal{X}_{0nc,1} \\ \mathcal{X}_{0nc,2} \\ \vdots \\ \mathcal{X}_{0nc,3n_a+1} \end{bmatrix}}_{\mathcal{X}_{0n}} + \underbrace{\begin{bmatrix} \mathcal{C}_{nc,1} & \cdots & 0 \\ 0 & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & \mathcal{C}_{nc,3n_a+1} \end{bmatrix}}_{\mathcal{C}_n} \underbrace{\begin{bmatrix} \mathbf{b}_{nc,1} \\ \mathbf{b}_{nc,2} \\ \vdots \\ \mathbf{b}_{nc,3n_a+1} \end{bmatrix}}_{\mathbf{b}_n}$$

$$+ \underbrace{\begin{bmatrix} \mathcal{B}_{nc,1} & \cdots & 0 \\ 0 & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & \mathcal{B}_{nc,3n_a+1} \end{bmatrix}}_{\mathcal{B}_n} \underbrace{\begin{bmatrix} \mathbf{u}_{nc,1} \\ \mathbf{u}_{nc,2} \\ \vdots \\ \mathbf{u}_{nc,3n_a+1} \end{bmatrix}}_{\mathbf{u}_n} \tag{4.12}$$

or, in more compact form,

$$\mathcal{X}_n = \mathcal{A}_n \mathcal{X}_{0n} + \mathcal{C}_n \mathbf{b}_n + \mathcal{B}_n \mathbf{u}_n \tag{4.13}$$

Chapter 5

Strategic Air Traffic Planning as an Integer Programming Problem

In this chapter, strategic air traffic planning is posed as an IP problem, which is a variation of the more common LP problem [50, 61, 69, 71]. Given the ARM and a set of scheduled departures, we want to assess what modifications are necessary to generate an aircraft flow that minimizes a weighted summation of delays and satisfies a set of constraints. The ARM described in Chap. 2 and Chap. 3 is characterized by linear time-invariant dynamics that are described in Chap. 4. With a proper manipulation of the dynamics, linear constraints can be posed that guarantee (i) no more aircraft than available are controlled, and (ii) the capacity constraints driven by operational limitations of ATCs and airports are satisfied. This is described in Sec. 5.1, Sec. 5.2, and Sec. 5.3.

A cost function is also introduced in Sec. 5.4. It is a weighted linear combination of the delay introduced by the control variables of the ARM, i.e., ground holding, pre-departure rerouting, and airborne rerouting. The problem we treat is integer because control variables are integer, since they are the number of aircraft controlled by a specific control. In Sec. 5.5, the solution method and some properties of the solution are presented and discussed.

5.1 Control Constraints and Capacity Constraints

In the IP framework, two types of constraints are modeled. The first set of constraints is labeled *control* constraints. For each aggregate route we need to guarantee that the number of flights delayed or rerouted at each time is not larger than the number of flights available to depart.

The second set of constraints is labeled *capacity* constraints. Goal of this set of constraints is to guarantee that the aircraft traffic flow satisfies capacity limitations imposed by ATCs and airports. As example, each sector in the NAS can accommodate a limited number of aircraft simultaneously. At the same time, departures and arrivals from/to each airport are constrained due to the limited number of runways and operational capabilities of ATCs.

As described in Chap. 4, dynamics of each network component are uncoupled. Control constraints are imposed on single nodes in the network, and thus preserve uncoupling between different network components. On the other hand, capacity constraints couple controls of different network components. In fact, aggregate routes of different network component share the same airspace and the same airports. Section 5.2 describes how control constraints are imposed, while Sec. 5.3 describes how the different capacity constraints are formulated. In both cases, we need to express constraints in the form $Ax \leq b$, which is the format required by the IP formulation.

5.2 Control Constraints

Control constraints are directly applied to every node characterized by a control. This means that a constraint will be imposed on every origin node, and on every node where airborne rerouting is allowed. Figure 5.1 shows the same example already introduced in Sec. 4.2.2, and will be used here to describe how control constraints are formulated. Figure 5.1 represents a network component with two aggregate routes that can exchange aircraft. In addition, the second aggregate route is characterized by an airborne rerouting branch that can be exploited. Overall, there are five controls per time-step, denoted $u_{1\to1}$, $u_{5\to5}$, $u_{1\to5}$, $u_{5\to1}$ and $u_{6\to9}$ representing, respectively, ground holding applied to route A, ground holding applied to route B, pre-departure rerouting from route A to route B, pre-departure rerouting from route B to route A and airborne rerouting applied to route B.

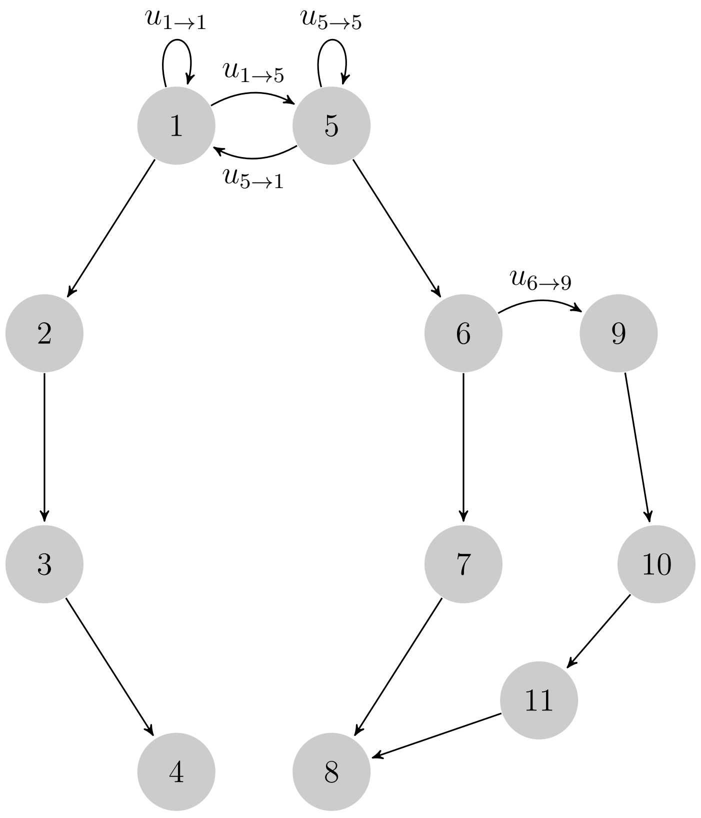


Figure 5.1: Example of network component used as reference to describe control constraints.

We will focus on origin nodes first. Considering Fig. 5.1 again, we can express the state in origin nodes 1 and 5 for time-steps t_1 and t_2 as follows

$$\begin{aligned}
x_1(t_1) &= b_1(t_1) - u_{1\to 1}(t_1) - u_{1\to 4}(t_1) + u_{4\to 1}(t_1) \\
x_5(t_1) &= b_5(t_1) - u_{5\to 5}(t_1) - u_{5\to 1}(t_1) + u_{1\to 5}(t_1) \\
x_1(t_2) &= b_1(t_2) + u_{1\to 1}(t_1) - u_{1\to 1}(t_2) - u_{1\to 5}(t_2) + u_{5\to 1}(t_2) \\
x_5(t_2) &= b_5(t_2) + u_{5\to 5}(t_1) - u_{5\to 5}(t_2) - u_{5\to 1}(t_2) + u_{1\to 5}(t_2)
\end{aligned} \tag{5.1}$$

which can be more generally extended using the following notation

$$x_1(t_k) = \begin{cases} b_1(t_k) - u_{1\to 1}(t_k) - u_{1\to 5}(t_k) + u_{5\to 1}(t_k) & \text{if k=1} \\ b_1(t_k) + u_{1\to 1}(t_{k-1}) - u_{1\to 1}(t_k) - u_{1\to 5}(t_k) + u_{5\to 1}(t_k) & \text{otherwise} \end{cases}$$

$$x_5(t_k) = \begin{cases} b_5(t_k) - u_{5\to 5}(t_k) - u_{5\to 1}(t_k) + u_{1\to 5}(t_k) & \text{if k=1} \\ b_5(t_k) + u_{5\to 5}(t_{k-1}) - u_{5\to 5}(t_k) - u_{5\to 1}(t_k) + u_{1\to 5}(t_k) & \text{otherwise} \end{cases}$$

For t_1, scheduled departures in each origin node can be ground held or pre-departure rerouted towards the other aggregate route. Each aggregate route can also receive a positive contribution deriving from the pre-departure rerouting flow coming from the other route. For all later time-steps, there is also the positive contribution of flights that were pushed back from the previous time-step. From a graph perspective, this is due to the cycle we introduced in the origin node.

To impose control constraints, we impose that the algebraic summation of scheduled departures from the current origin node and controls affecting the same origin node must be non-negative. For the two origin nodes mentioned above, control constraints are imposed as follows

$$\begin{cases} u_{1\to1}(t_k) + u_{1\to5}(t_k) \leq b_1(t_k) & \text{if k=1} \\ -u_{1\to1}(t_{k-1}) + u_{1\to1}(t_k) + u_{1\to5}(t_k) \leq b_1(t_k) & \text{otherwise} \end{cases}$$

$$\begin{cases} u_{5\to5}(t_k) + u_{5\to1}(t_k) \leq b_5(t_k) & \text{if k=1} \\ -u_{5\to5}(t_{k-1}) + u_{5\to5}(t_k) + u_{5\to1}(t_k) \leq b_5(t_k) & \text{otherwise} \end{cases}$$

For each aggregate route we need to guarantee that the number of flights delayed or rerouted at each time is not larger than the number of flights available to depart; this could be the number of originally scheduled departures, or more if flights have been delayed at previous times. For the origin node of each controllable aggregate route, the summation of the scheduled departures and the control actions modifying those departures must be non-negative. Note that this approach is slightly different than requiring states in nodes 1 and 5 to be non-negative. With our approach, scheduled departures only account for departures that were originally scheduled to use the aggregate route the current origin node belongs to. The imposition of the non-negativeness of states in origin nodes, would consider the additional positive contribution of aircraft that are rerouted from other aggregate routes. As an example, for node 1 this approach would translate into constraints $u_{1\to1}(t_k)+u_{1\to5}(t_k)-u_{5\to1}(t_k) \leq b_1(t_k)$ for $k = 1$, and $-u_{1\to1}(t_{k-1})+u_{1\to1}(t_k)+u_{1\to5}(t_k)-u_{5\to1}(t_k) \leq b_1(t_k)$ otherwise. Our approach is based on the assumption that, for each aggregate route, decisions on the number of aircraft to control should be based on the available number of aircraft scheduled to depart along that route. The addition of aircraft from other routes defines the overall controlled number of departures per each time-step. As example, the state of node 1 at t_2 defines the controlled departures along the first aggregate route at that time-step, and also contains term $u_{5\to1}(t_2)$. This term is not considered when imposing control constraints on node 1, but is accounted for when considering capacity constraints (e.g., departure capacity constraints).

Note that our approach guarantees also the non-negativeness of states at origin nodes. In fact, controls that are not explicitly accounted for, are all non-negative. Thus, they can only increase a quantity that is constrained to be non-negative already.

We now focus on airborne rerouting control constraints. Unlike origin nodes, airborne rerouting nodes are not characterized by a cycle. They only receive aircraft from a node of the original aggregate route. Referring to Fig. 5.1 again, we can express the state of the airborne rerouting node on the original route x_6 and the state of the first node of the airborne rerouting branch x_9 as follows

$$\begin{aligned}
x_6(t_1) &= x_5(t_0) - u_{6\to9}(t_1)\\
x_9(t_1) &= u_{6\to9}(t_1)\\
x_6(t_2) &= b_5(t_1) - u_{5\to5}(t_1) - u_{5\to1}(t_1) + u_{1\to5}(t_1) - u_{6\to9}(t_2)\\
x_9(t_2) &= u_{6\to9}(t_2)
\end{aligned} \tag{5.2}$$

For each airborne rerouting node there is an upper bound on the number of aircraft that can be rerouted. The upper bound is the state of the upstream node at the previous time-step. This translates into the imposition that the state in the airborne rerouting node on the original aggregate route (x_6 in our case) must be non-negative. Referring to Fig. 5.1

$$\begin{aligned} &u_{6\to9}(t_1) \le x_5(t_0) \\ &u_{6\to9}(t_2) \le b_5(t_1) - u_{5\to5}(t_1) - u_{5\to1}(t_1) + u_{1\to5}(t_1) \end{aligned} \tag{5.3}$$

For each network component, recalling Eq. 4.5, control constraints for origin nodes are expressed in matrix form as

$$M_{co}^{nc}\mathbf{u}_{nc} \le \mathbf{b}_{nc} \tag{5.4}$$

where M_{co}^{nc} is a $(N_r N_t, N_c N_t)$ matrix (being N_r the number of aggregate routes and N_c the number of controls) that maps controls. For each origin node, columns associated with controls from that node are characterized by a +1, the column associated with ground control from the previous time-step is characterized by a -1. Note that the latter case is active only from t_2 on.

Control constraints for airborne rerouting nodes are written in matrix form, recalling again Eq. 4.5, as

$$-M_{ca}^{nc}\mathcal{B}_{nc}\mathbf{u}_{nc} \le M_{ca}^{nc}(\mathcal{A}_{nc}\mathcal{X}_{0_{nc}} + \mathcal{C}_{nc}\mathbf{b}_{nc}) \tag{5.5}$$

where M_{ca}^{nc} is a $(N_{ar}N_t, N_c N_t)$ matrix (being N_{ar} the number of airborne rerouting nodes of the network component). Each row of the matrix is filled with zeros, apart from a single unitary element that corresponds to an airborne rerouting node at a specific time-step. Eq. 5.5 satisfies the required format for the IP problem.

When extending Eq. 5.4 and Eq. 5.5 to the full network, matrices M_{co}^{nc} and M_{ca}^{nc} are stacked in block diagonal fashion, while vectors $\mathbf{b}_{nc}$ are vertically stacked. For the full network we will have

$$\underbrace{M_{co}^{n}}_{A_{co}}\mathbf{u}_n \leq \underbrace{\mathbf{b}_n}_{b_{co}} \tag{5.6}$$

for origin nodes and

$$\underbrace{-M_{ca}^{n}\mathcal{B}_n}_{A_{ca}}\mathbf{u}_n \leq \underbrace{M_{ca}^{n}(\mathcal{A}_n\mathcal{X}_{0_n} + \mathcal{C}_n\mathbf{b}_n)}_{b_{ca}} \tag{5.7}$$

for airborne rerouting nodes.

5.3 Capacity Constraints

Capacity constraints are constraints that regulate the aircraft flow to avoid congestion both in the enroute phase and for proximity operations (i.e., take off and landing). There are constraints on the following:

1. Sector capacity: An upper bound on the number of aircraft within a sector.
2. Airport capacity: Due to the availability of runways, as well as control tower operational capabilities, departure and arrival rates for each airport have upper bounds.
3. Flow capacity: Due to the physical dimension of cells mapping the nodes of the ARM,

an upper bound on the state of nodes is posed. Also, in-plane separation between aircraft (by definition, a non-linear constraint) can be treated in an approximated way as a flow capacity constraint.

4. Convective weather blockage: An upper bound (possibly 0) on node capacity in regions with convective weather.

These are all capacity constraints, possibly time-varying, on the number of aircraft within a set of nodes that belong to a sector, airport, or portion of a route. They can be defined by multiplying the system dynamics in Eq. 4.12 by a matrix that sums the state values for the relevant nodes at each time-step, and then requiring that the resulting sums are less than or equal to the bounds.

5.3.1 Sector Capacity

The NAS is subdivided into sectors, i.e., regions of the airspace characterized by a specific boundary, each controlled by an ATC. Due to operational capabilities, each sector can handle a maximum number of aircraft simultaneously. The NAS is divided into two types of sectors: low and high. Low sectors are characterized by a lower maximum flight level, and define regions of the NAS closer to airports, where departure and arrival maneuvers need to be coordinated at a tactical level. High sectors are characterized by a higher maximum flight level, and define the enroute phase of the aircraft flow. For strategic purposes, we will focus on high sectors when assessing sector capacity constraints.

For each high sector in D and for each time-step of the planning horizon, the summation of the state variables inside the sector must not exceed the maximum capacity of such sector. Considering N_s high sectors, a (N_s, N_n) matrix m_s is defined, where elements of the i-row are 1 if the corresponding node is inside the i-th sector, and zero otherwise. Basically, each row maps the state into the sector count. We then build matrix M_s^{nc} by stacking N_t times matrix m_s in block diagonal fashion. M_s^{nc} of different network components are then horizontally stacked to obtain the network sector capacity matrix M_s. Recalling Eq. 4.12, we can write

$$\underbrace{M_s \mathcal{B}_n}_{A_s^n} \mathbf{u}_n \leq \underbrace{\eta_s - M_s(\mathcal{A}_n \mathcal{X}_{0_n} + \mathcal{C}_n \mathbf{b}_n)}_{b_s^n} \tag{5.8}$$

where η_s is a $(N_s N_t, 1)$ vector containing the upper bounds for each sector at all time-steps.

5.3.2 Departure and Arrival Capacity

For each airport, departures and arrivals are subject to constraints that depend on the number and geometry of the runways available. Generally, take off and landing operations must follow standardized procedures that assign some specific runways to departures, and some other runways to arrivals, unless ATCs authorize otherwise. As example, consider the LAX airport runway configuration, as shown in Fig. 5.2. This airport is one of the few airports in the world with four parallel runways, labeled 6L/24R, 6R/24L, 7L/25R, 7R/25R going from North to South. For each runway, the number represents the heading divide by ten (e.g., 6 means 60 *deg*, being 0 *deg* the North direction and 90 *deg* the East direction), while the addition of a letter is necessary in case of parallel runways. As example, runways 6L and 6R are parallel, with runway 6L being the left-most runway for an aircraft approaching LAX from the West. Each runway is also characterized by the "complementary" label that identifies an approach from the other direction. 180 *deg* are added to the heading, and left and right are switched in case of parallel runways. Thus, runways 6L and 6R become 24R and 24L, respectively, for an aircraft approaching from East. Also, note that although all four runways are characterized by the same heading (i.e., 70 *deg*), the heading of the two northern runways is fictitiously changed to 60 *deg* to avoid confusion. This is a common procedure anytime more than three parallel runways are present.

According to [48], westerly operations is the normal traffic pattern used at LAX during the daytime (6:30 AM to midnight) throughout the year. Aircraft approach the airport from the East and depart the airport to the West due to the prevailing westerly wind. Two other operational schemes, namely over-ocean and easterly operations, are implemented during the more noise-sensitive night-time (midnight to 6:30 AM) and when weather conditions (rainstorms or Santa Ana winds) require to reverse the traffic flow of the airport, respectively. If westerly operations are implemented, departures are operated via inboard runways (6R and 7L), while arrivals are operated via outboard runways (6L and 7R). Reference [48] reports that, in 2013, the runway utilization for the north complex (runways 6L and 6R) was 97%-3% for inboard-outboard departures, and 8%-92% for inboard-outboard arrivals. The runway utilization for the south complex (runways 7L and 7R) was 91%-9% for inboard-outboard departures, and 6%-94% for inboard-outboard arrivals.

For the majority of the traffic flow, two runways allocate departures, and two allocate arrivals, even though the reported percentages show that small variations are possible. Variations depend on the operational scheme adopted, or on unpredicted contingencies affecting the airport. As example, given a higher number of passenger gates and air cargo facilities on the south complex, runways 7L and 7R generally experience a greater number of operations. This also implies that, in high traffic conditions, some aircraft leaving from the south complex gates might depart from the outboard runway 7R (generally used for arrivals), in order to avoid the unnecessary crossing of active runways. Lastly, variations might depend on aircraft-specific requirements. As example, both the Boeing 747-800 and the Airbus A380 are not allowed to take off from runway 7L due to insufficient spacing between the runway and taxiway B due to an expanded wingspan. As a consequence, they are required to take off from runway 7R, which is generally dedicated to arrivals.

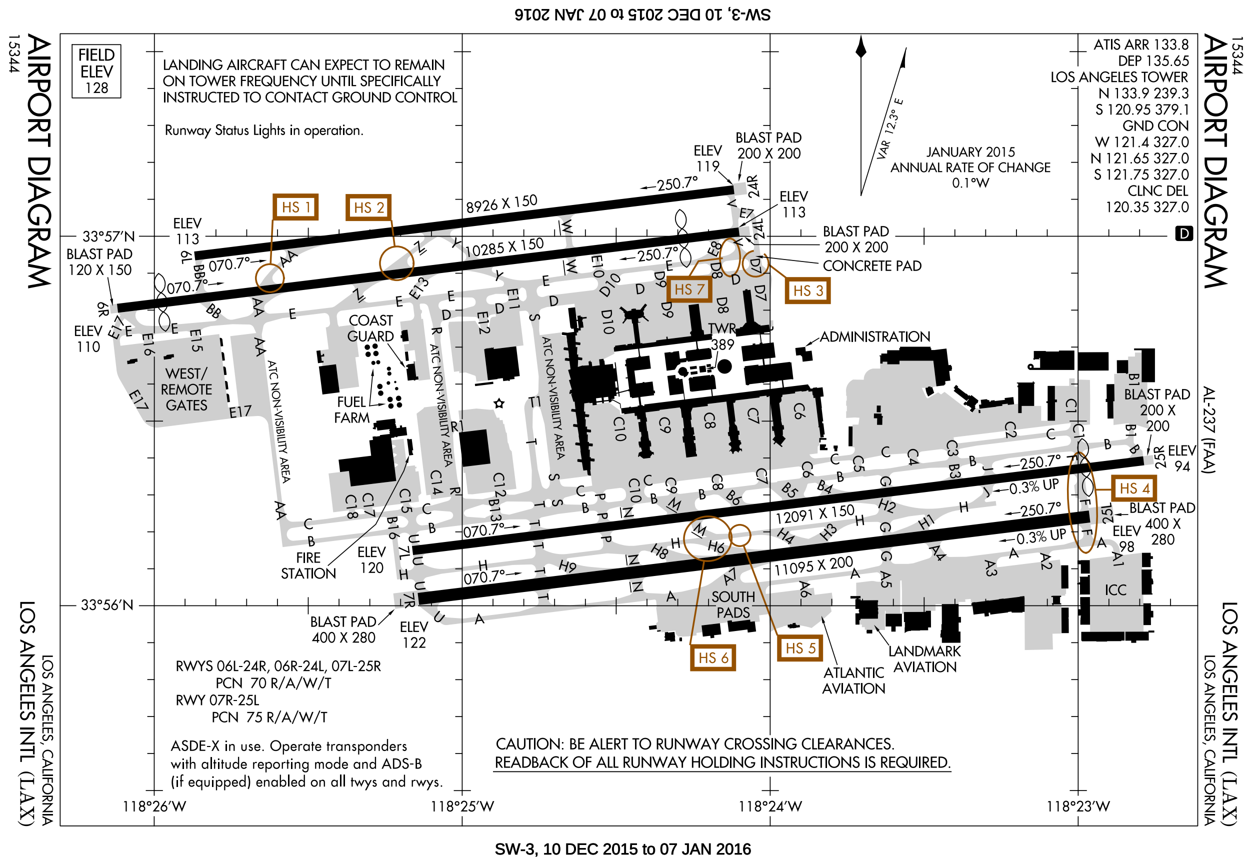

AIRPORT DIAGRAM
LOS ANGELES INTL (LAX)
LOS ANGELES, CALIFORNIA
SW-3, 10 DEC 2015 to 07 JAN 2016
FIELD ELEV 128
LANDING AIRCRAFT CAN EXPECT TO REMAIN ON TOWER FREQUENCY UNTIL SPECIFICALLY INSTRUCTED TO CONTACT GROUND CONTROL
Runway Status Lights in operation.
ATIS ARR 133.8
DEP 135.65
LOS ANGELES TOWER
N 133.9 239.3
S 120.95 379.1
GND CON
W 121.4 327.0
N 121.65 327.0
S 121.75 327.0
CLNC DEL
120.35 327.0
JANUARY 2015 ANNUAL RATE OF CHANGE 0.1°W
HS 1
HS 2
HS 3
HS 4
HS 5
HS 6
HS 7
RWYS 06L-24R, 06R-24L, 07L-25R PCN 70 R/A/W/T
RWY 07R-25L PCN 75 R/A/W/T
ASDE-X in use. Operate transponders with altitude reporting mode and ADS-B (if equipped) enabled on all twys and rwys.
CAUTION: BE ALERT TO RUNWAY CROSSING CLEARANCES. READBACK OF ALL RUNWAY HOLDING INSTRUCTIONS IS REQUIRED.
AL-237 (FAA)
118°26′W
118°25′W
118°24′W
118°23′W
33°57′N
33°56′N

Figure 5.2: Runway diagram for LAX.

The LAX example is particularly significant to understand the complexity of surface operations that encompass gate leaving, taxiing and actual take off [36, 37]. The same complexity can be found in arrival procedures [15] to time landings and ensure safety. In our strategic planning, we do not have the ambition to treat departures and arrivals with such a degree of accuracy, but we acknowledge that limitations exist on departures and arrivals. As a consequence, we use a data-driven approach to compute (i) upper bounds on departures and arrivals per hour, and (ii) upper bounds on departures and arrivals per time-step.

Both constraints are relevant in a strategic framework. In fact, as already introduced in Chap. 2, constraints (i) represent the Gilbo envelope [34] for a given airport, and define the maximum hourly load the airport can handle in terms of departures and arrivals. In Fig. 5.3, the Gilbo envelope for LAX is shown: the airport cannot handle more than 53 departures per hour and 54 arrivals per hour. The two upper bounds cannot be reached simultaneously due to the presence of the sloped constraint. At the same time, it is important to provide an estimate of the capability of each airport per time-step. Otherwise, a solution might exists that is contained in the Gilbo envelope, but that forces aircraft to take off and/or land at a rate that the airport cannot sustain.

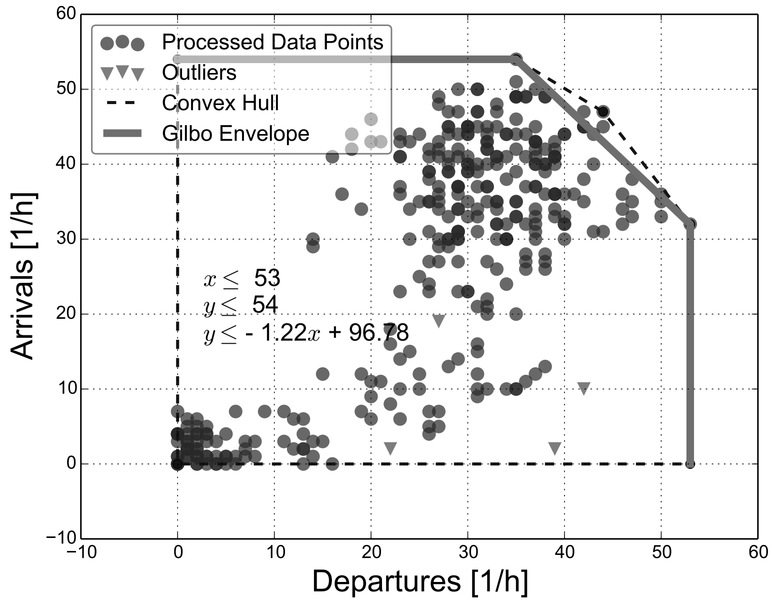
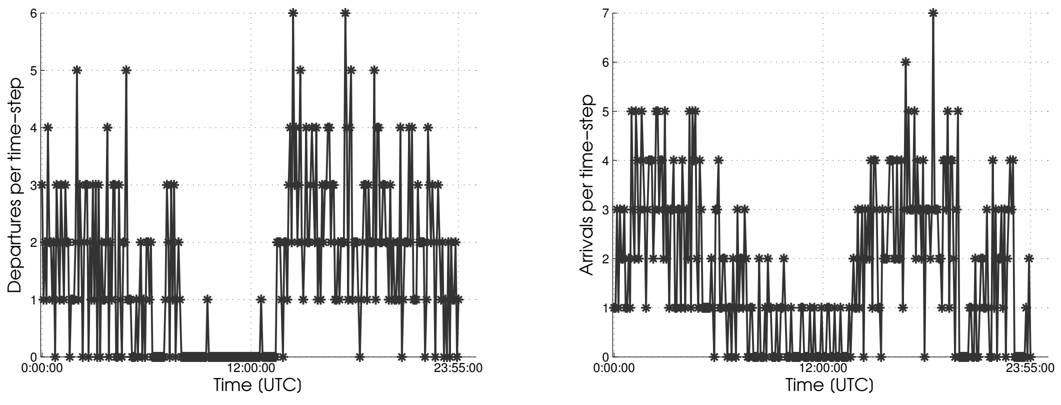


Figure 5.3: Gilbo envelope for LAX.

For constraints (ii), we process the historical dataset introduced in Chap. 2. For all the airports of interest, departures and arrivals are grouped into bins whose time interval is the time-step of the model Δt. As example, if $\Delta t = 5\ min$, departures and arrivals for each airports will be divided into time intervals as follows 0:00:00-0:04:59 UTC, 0:05:00-0:09:59 UTC etc. For both departures and arrivals, the maximum value among the different bins will be selected as the upper bound. In general, the maximum number of departures and arrivals per time-step is a time-dependent quantity. Airports will reach maximum operational capabilities during peak hours, releasing and absorbing a higher number of aircraft. Thus, this constraint could be easily turned into a time-dependent constrain. At the same time, we are mostly interested in an upper bound that represents a maximum operational capability.

Even though the overall number of hourly departures and arrivals decreases consistently for non-peak hours (see Fig. 5.3), we are interested in the maximum capability for each airport. Figure 5.4 shows the approach applied to LAX.



(a) Departures. (b) Arrivals.

Figure 5.4: Departures (07/01/2014) and arrivals (07/03/2014) per time-step ($\Delta t = 5\ min$) for LAX.

In both Fig. 5.4(a) (departures for 07/01/2014) and Fig. 5.4(b) (arrivals for 07/03/2014), operations drop significantly between 8:00:00 and 13:00:00 UTC, which correspond to night-time in the pacific coast. The upper bound for both departures and arrivals is around 18:00:00 UTC, which corresponds to late morning in the pacific coast. The bounds are 6 departures/Δt and 7 arrivals/Δt. If the airport releases and absorbs aircraft at the maximum capacity for a full hour, the Gilbo envelope constraints is not satisfied. Hence, the necessity to impose both sets of constraints at the same time.

We now proceed to describe how constraints (i) and (ii) are posed in matrix form. We will start with the constraint set (i), and then show how constraint set (ii) is a slight variation of (i).

We start with the constraint on maximum departures. Given a planning horizon that encompasses N_h hours, we have N_h constraints for each network component characterized by departures. We define N_n the number of nodes of the current network component, and $N_{\Delta t}$

the number of time-steps contained in one hour (as example, if $\Delta t = 5\,\mathrm{min}$, $N_{\Delta t} = 12$). We define a $(1, N_n N_{\Delta t})$ vector m_d^{nc}. This vector is obtained by horizontally stacking $N_{\Delta t}$ times a $(1, N_n)$ vector with 1 in columns that map origin nodes, and 0 otherwise. We build a $(N_h, N_n N_t)$ matrix M_d, which is a block diagonal matrix where vector m_d^{nc} is repeated N_h times. We also define a $(N_h, 1)$ vector η_d^{nc}, where the maximum allowed hourly departure value, as obtained from the Gilbo envelope, is repeated N_h times. Using the same strategy shown in Eq. 5.8, we pre-multiply the state vector obtaining

$$M_d^{nc}(\mathcal{A}_{nc}\mathcal{X}_{0_{nc}} + \mathcal{C}_{nc}\mathbf{b}_{nc} + \mathcal{B}_{nc}\mathbf{u}_{nc}) \leq \eta_d^{nc} \tag{5.9}$$

that is rewritten as

$$\underbrace{M_d^{nc}\mathcal{B}_{nc}}_{A_d^{nc}}\mathbf{u}_{nc} \leq \underbrace{\eta_d^{nc} - M_d^{nc}(\mathcal{A}_{nc}\mathcal{X}_{0_{nc}} + \mathcal{C}_{nc}\mathbf{b}_{nc})}_{b_d^{nc}} \tag{5.10}$$

The structure of matrix M_d does not change if we consider an internal or exiting network component, while departure constraints are not defined for an entering network component.

For maximum arrivals, the structure of the matrix equation has the same structure as Eq. 5.9 for every entering network component, since the associated airport behaves as a sink from a network perspective. m_d^{nc} is replaced by m_a^{nc}, with 1 in columns that map destination nodes, and 0 otherwise. The block diagonal matrix M_a is then assembled, and Eq. 5.11

$$\underbrace{M_a^{nc}\mathcal{B}_{nc}}_{A_a^{nc}}\mathbf{u}_{nc} \leq \underbrace{\eta_d^{nc} - M_a^{nc}(\mathcal{A}_{nc}\mathcal{X}_{0_{nc}} + \mathcal{C}_{nc}\mathbf{b}_{nc})}_{b_a^{nc}} \tag{5.11}$$

is obtained.

While exiting network components do not contribute to maximum arrivals constraints, the structure of the matrix is more involved when considering internal network components. In fact, since for each internal network component the airport is a source, there will be no contribution to arrivals from the internal network component of that airport. On the other hand, all the other internal network components will contribute to arrivals to the current airport, as long as aggregate routes connecting the two airports are present. If we consider the states of all internal network components $\mathcal{X}_{int}$ stacked vertically

$$\mathcal{X}_{int} = \begin{bmatrix} \mathcal{X}_{int}^1 \\ \mathcal{X}_{int}^2 \\ \vdots \\ \mathcal{X}_{int}^{n_a} \end{bmatrix} \tag{5.12}$$

we can map hourly arrivals to each airport using the following (n_a, n_a) block-matrix

$$\underbrace{\begin{bmatrix} 0 & M^a_{1,2} & M^a_{1,3} & \cdots & M^a_{1,n_a} \\ M^a_{2,1} & 0 & M^a_{2,3} & \cdots & M^a_{2,n_a} \\ M^a_{3,1} & M^a_{3,2} & 0 & \cdots & M^a_{3,n_a} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ M^a_{n_a,1} & M^a_{n_a,2} & \cdots & \cdots & 0 \end{bmatrix}}_{M^{int}_a} \tag{5.13}$$

where each block $M^a_{i,j}$ is a $(N_h, N^j_n N_t)$ block diagonal matrix. In gray are highlighted blocks that might contain non-zero entries. Diagonal blocks are identically zero, since each internal network component behaves only as a source and does not contribute to its own arrivals, as mentioned before. If we pre-multiply vector 5.12 by matrix 5.13, we obtain a $(n_a N_h, 1)$ vector where hourly arrivals for each of the n_a airports are vertically stacked in sequence. Each element of this vector should be less or equal to the maximum arrival capacity of the associated airport stored in the $(n_a N_h, 1)$ vector η_a as follows

$$M^{int}_a \mathcal{X}_{int} \leq \eta_a \tag{5.14}$$

When considering all internal network components, the different M_d defined in Eq. 5.9 will have a matrix structure that is similar to Eq. 5.13

$$\underbrace{\begin{bmatrix} M_d^{1,1} & 0 & 0 & \cdots & 0 \\ 0 & M_d^{2,2} & 0 & \cdots & 0 \\ 0 & 0 & M_d^{3,3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & M_d^{n_a,n_a} \end{bmatrix}}_{M_d^{int}} \tag{5.15}$$

In this case, the only non-zero blocks are the diagonal blocks, and the equivalent form of Eq. 5.14 is

$$M_d^{int}\mathcal{X}_{int} \leq \eta_d \tag{5.16}$$

where η_d is obtained vertically stacking the η_d^{nc} vectors that defines the maximum allowed departures from the n_a airports inside D.

When considering the sloped constraint that couples departures and arrivals, we use both M_d^{int} and M_a^{int}. Given n_a Gilbo envelopes as shown in Fig. 4.2, each airport is characterized by a y-intercept q_G and a slope m_G. We define M_G as a (n_aN_h, n_aN_h) diagonal matrix, where diagonal elements $(i-1)N_h+1$, $(i-1)N_h+2$, $(i-1)N_h+3$, iN_h are the m_G of the i-th airport considered. Q_G is a $(n_aN_h, 1)$ vector where elements $(i-1)N_h+1$, $(i-1)N_h+2$, $(i-1)N_h+3$, iN_h are the q_G of the i-th airport considered. We then define the matrix inequality

$$M_a^{int}\mathcal{X}_{int} - M_G M_d^{int}\mathcal{X}_{int} - Q_G \leq 0 \tag{5.17}$$

that defines a set of $n_a N_h$ constraints.

The extension of maximum departures, maximum arrivals, and sloped constraints to the full dynamics is straightforward. The overall number of constraints does not change. The only change is that matrices M_d^{int} and M_a^{int} need to be horizontally padded with similar matrices that map origin nodes for exiting network components, and destination nodes for entering network components. For exiting network components, M_d^{exit} has the same block diagonal structure as M_d^{int}. The number of rows is unchanged, while the number of columns depends on the size of all exiting network components. M_d^{ent} is a zero matrix that is necessary for matrix multiplication purposes. With a similar reasoning, M_a^{exit} is identically zero. M_a^{ent} shows a different structure with respect to M_a^{int} instead. In fact, now each airport behaves as a sink for the associated entering network component. As a consequence, M_a^{ent} has a block diagonal structure. For both departures and arrivals, overflights have no contribution.

For departures, the full M_d matrix has the following format

$$\left[\begin{array}{cccc|cccc|cc|cc}
M_{d,int}^{1,1} & 0 & \cdots & 0 & M_{d,exit}^{1,1} & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
0 & M_{d,int}^{2,2} & \cdots & 0 & 0 & M_{d,exit}^{2,2} & \cdots & 0 & 0 & 0 & & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & M_{d,int}^{n_a,n_a} & 0 & 0 & \cdots & M_{d,exit}^{n_a,n_a} & 0 & 0 & 0 & 0
\end{array}\right] \tag{5.18}$$

where the only non-zero blocks are the internal and exiting ones. For arrivals, the full M_a matrix has the following format

$$\left[\begin{array}{cccc|cc|cccc|cc} 0 & M_{a,int}^{1,2} & \cdots & M_{a,int}^{1,n_a} & 0 & 0 & M_{a,ent}^{1,1} & 0 & \cdots & 0 & 0 & 0 \\ M_{a,int}^{2,1} & 0 & \cdots & M_{a,int}^{2,n_a} & 0 & 0 & 0 & M_{a,ent}^{2,2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ M_{a,int}^{n_a,1} & M_{a,int}^{n_a,2} & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & M_{a,ent}^{n_a,n_a} & 0 & 0 \end{array}\right] \tag{5.19}$$

where the only non-zero blocks are the internal and entering ones.

Recalling Eq. 4.12, maximum departure capacity constraints can we written as

$$\underbrace{M_d\mathcal{B}_n}_{A_d^n}\mathbf{u}_n \leq \underbrace{\eta_d - M_d(\mathcal{A}_n\mathcal{X}_{0_n} + \mathcal{C}_n\mathbf{b}_n)}_{b_d^n} \tag{5.20}$$

maximum arrival constraints can be written as

$$\underbrace{M_a\mathcal{B}_n}_{A_a^n}\mathbf{u}_n \leq \underbrace{\eta_a - M_a(\mathcal{A}_n\mathcal{X}_{0_n} + \mathcal{C}_n\mathbf{b}_n)}_{b_a^n} \tag{5.21}$$

while the sloped constraint has the form

$$\underbrace{(M_a\mathcal{B}_n - M_GM_d\mathcal{B}_n)}_{A_{da}^n}\mathbf{u}_n \leq \underbrace{Q_G - M_a(\mathcal{A}_n\mathcal{X}_{0_n} + \mathcal{C}_n\mathbf{b}_n) + M_GM_d(\mathcal{A}_n\mathcal{X}_{0_n} + \mathcal{C}_n\mathbf{b}_n)}_{b_{da}^n} \tag{5.22}$$

Departure and arrival constraints of type (ii), i.e., per time-step and not referred to Gilbo envelopes, have a similar structure with respect to Eq. 5.20 and Eq. 5.21. The only difference is in the structure of each block of matrices M_d and M_a. In fact, in Eq. 5.18 and Eq. 5.19 each matrix block has N_h rows and is characterized by a block diagonal structure where each block diagonal element is the same vector mapping departure/arrival nodes horizontally repeated $N_{\Delta t}$ times. When considering departures and arrivals per time-step instead, each matrix block has an increased number of rows that is equal to $N_{\Delta t} N_h = N_t$, where each block diagonal element is simply the vector mapping departure/arrival nodes. We can define matrices $M_{d\Delta t}$ and $M_{a\Delta t}$, that are assembled as described above. They are "taller" than M_d and M_a since one constraint per time-step (and not per hour) must be imposed. The structure of the inequality constraints remains unchanged. For maximum departures per time-step we have

$$\underbrace{M_{d\Delta t}\mathcal{B}_n}_{A^n_{d\Delta t}}\mathbf{u}_n \leq \underbrace{\eta_{d\Delta t} - M_{d\Delta t}(\mathcal{A}_n\mathcal{X}_{0_n} + \mathcal{C}_n\mathbf{b}_n)}_{b^n_{d\Delta t}} \tag{5.23}$$

while for maximum arrivals for time-step the inequality is

$$\underbrace{M_{a\Delta t}\mathcal{B}_n}_{A^n_{a\Delta t}}\mathbf{u}_n \leq \underbrace{\eta_{a\Delta t} - M_{a\Delta t}(\mathcal{A}_n\mathcal{X}_{0_n} + \mathcal{C}_n\mathbf{b}_n)}_{b^n_{a\Delta t}} \tag{5.24}$$

In Eq. 5.23 and Eq. 5.24, $\eta_{d\Delta t}$ and $\eta_{a\Delta t}$ are $(N_t n_a, 1)$ vectors, where n_a vectors containing N_t values of maximum departures and arrivals per time-step for the current airport are vertically stacked.

5.3.3 Cell Capacity Constraints

Another capacity constraint is the maximum capacity of each cell associated to a node. As described in Chap. 2, we compute a network flow model that is described with a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. The graph $\mathcal{G}$ is the mathematical abstraction we use to define the dynamics of the ARM. The discrete linear time-invariant dynamics are a natural consequence of this abstraction. Each node is a dimensionless element of the graph but, in reality, each node is also associated with a 2D cell. This is the physical element where the state characterizing the node, i.e., aircraft, is placed. We now proceed with the description of our geometric interpretation of cells.

Each cell is centered in the associated node. The shape of a cell is defined by two parameters, i.e., a transversal and a longitudinal extension. The transversal extension is chosen to be 10 NM (5 NM for each side with respect to the node), which represents the lateral separation threshold according to Federal Aviation Administration (FAA) requirements. The longitudinal extension is determined as follows. The midpoints between the current node and the previous/following nodes are computed. In each point, a 10 NM long segment is drawn perpendicularly to the edge, and from each of the two endpoints of the segment a line parallel to the current edge is drawn. The left and right lines will intersect in two points that, together with the four vertices of the two segments, define the geometry of the cell. Each cell is thus a six sided polygon. When aggregate routes turn, our cells can describe the heading change. For trajectories that have a quasi-constant heading, the six sided polygon degenerates into a four sided polygon (rectangle). For origin and destination nodes, where sudden changes in the heading are part of take off and landing procedures, this geometric interpretation might fail to properly identify the six sided cell. We decide not to define cell for such nodes. This modeling choice is also motivated by the fact that the two constraints where cells are actually relevant (i.e., flow capacity and separation), are mainly imposed in the enroute phase. Figure 5.5 shows nodes and associated cells for an aggregate route from

LAX to DEN.

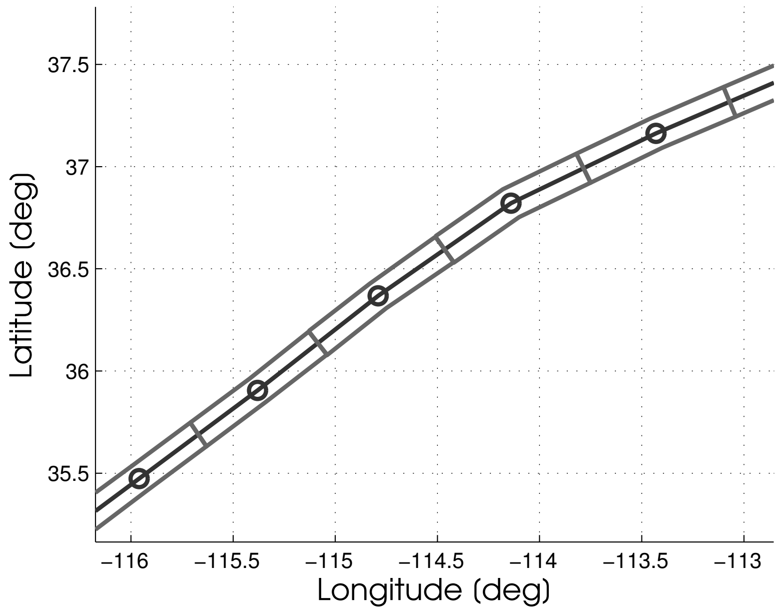


Figure 5.5: Aggregate route (zoomed) with nodes (circles) and associated cells from LAX to DEN.

When assessing maximum capacity for a single cell, the flight level of each cell is fixed and has been computed as shown in Chap. 2. As a consequence, only the in-plane spacing will be considered. In Sec. 5.3.4 it will be shown how the out-of-plane spacing issue is tackled. When it comes to longitudinal separation of aircraft flying the same route, a rule of thumb by FAA suggests that a 10 minute separation between aircraft should be guaranteed. In practice, this requirement is over-conservative and can be replaced with a longitudinal separation of roughly 20 NM, if both aircraft are equipped with a distance measuring equipment[1]. Figure 5.6 shows an example in this sense. Two recorded trajectories on 07/11/2014, from LAX to SFO are highlighted. The simultaneous recorded position of the two aircraft at 21:46:25 UTC is highlighted. Both aircraft occupy FL300, and their distance at that time is easily computable given the two latitude-longitude pairs, and is equal to 44.5 km, or 24 NM, which is consistent with the separation limit introduced above.

[1]http://tfmlearning.faa.gov/Publications/ATpubs/ATC/atc0604.html

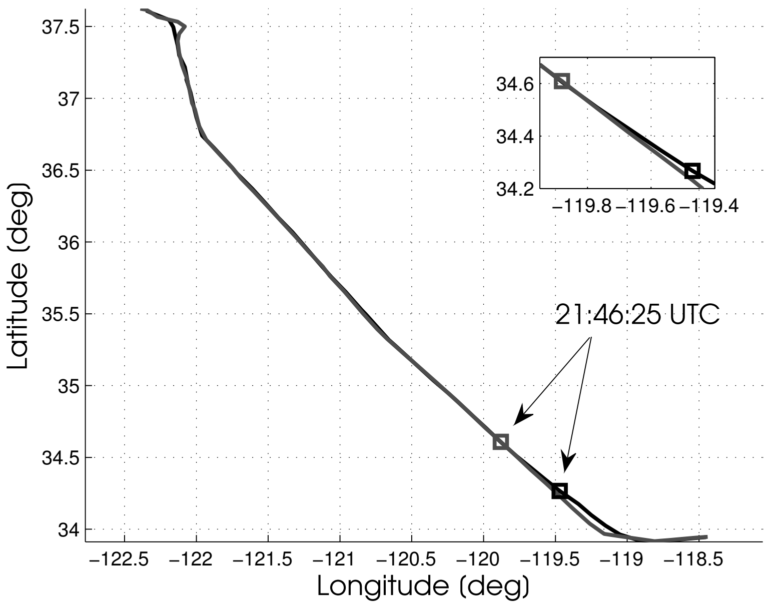


Figure 5.6: Two recorded flight trajectories from LAX to SFO.

On average, cells in our model have a longitudinal extension of 70 km when $\Delta t = 5$ min. Considering some margin at the two longitudinal extremes of each cell, to account for spacing of aircraft belonging to consecutive cells, a maximum cell capacity η_{cc} of 2 satisfies the requirement. The state of any node in the graph, at any given time-step, cannot be greater than η_{cc}.

To understand where this constraint is enforced, we switch back to the graph-oriented approximation of the traffic flow. Once aircraft are injected into the graph via origin nodes, there is only a scenario where the state along an aggregate route can increase. That is, in merging nodes. Hence, for each network component cell capacity constraints are posed for each origin node and for merging nodes associated to airborne rerouting branches.

For each network component, a $(N_r + N_{ar}, N_n)$ matrix m_f^{nc} is computed, where the i-th row has a 1 in correspondence of the origin of an aggregate route or in correspondence of a merging node. Then, matrix M_f^{nc} is built with N_t block diagonal elements m_f^{nc}. Pre-multiplying the dynamics of the network component by M_f^{nc}, a $(N_r + N_{ar}, 1)$ vector with states of all nodes of interest is isolated. Each element of this vector should be less or equal to the maximum allowed cell capacity η_{cc}. To extend the concept to all network components, we only need to assemble matrix M_f, which is a block diagonal matrix where each block defines a specific M_f^{nc}. Overall, cell capacity constraints can be written in matrix form as

$$\underbrace{M_f \mathcal{B}_n}_{A_f^n} \mathbf{u}_n \leq \underbrace{\eta_{cc} - M_f(\mathcal{A}_n \mathcal{X}_{0_n} + \mathcal{C}_n \mathbf{b}_n)}_{b_f^n} \tag{5.25}$$

5.3.4 Separation Constraints

In addition to in-plane separation among aircraft flying the same course, separation among aircraft flying different routes in a fundamental issue in ATM. In general, the horizontal separation between aircraft must be at least 5 NM, while vertical separation should be at least 1,000 ft. Each aircraft is thus surrounded by a “hockey puck” with a 5 NM radius and a 2,000 ft width at any given time. In our strategic planning framework, we do not have the capabilities to enforce distance constraints with such a degree of precision. Distance is a non-linear constraint, which cannot be treated explicitly in a linear framework like ours. At the same time, we want to provide some reasonable bounds, so that separation between aircraft can be handled with safety at a tactical level.

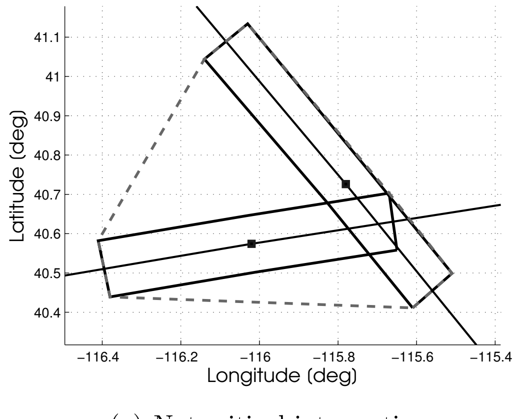
Our approach has two steps. First, (i) we divide cells according to their flight level, ranging from FL300 to FL400 (which are the flight levels that mainly contribute to the enroute phase of the traffic flow). In this way, only longitudinal separation needs to be accounted for. Then, (ii) for each flight level, intersections between cells belonging to different aggregate routes are identified. For each intersection, the convex hull is computed and converted into an area, and an estimate for the admissible number of aircraft is obtained.

In our strategic framework, we are interested in avoiding scenarios with a high density of aircraft in a small portion of the airspace, that would result in an operational burden for ATCs. Thus, we use a simple approach that aims at limiting the number of aircraft in each convex hull identified, using some basic assumptions. Consider a cell with a longitudinal extension l_1 of roughly 70 km (average size of cells for a ground speed of 450 kts and $\Delta t = 5$ min) and a lateral extension l_2 of 10 NM $\sim$ 20 km, which is the diameter of the "hockey puck" that defines the safe zone of each aircraft. A reasonable estimate of the area of such cell is $A_c = l_1 l_2 = 1400 km^2$. The in-plane non-admissible area around each aircraft is thus $A_{na} = \pi 10^2 = \pi 100$ km^2. Considering that the maximum state of each node is $\eta_{cc} = 2$, the fraction of the cell occupied by the 2 aircraft is $\frac{\eta_{cc} A_{na}}{A_c} = 0.45$. Let us label this ratio ϵ_a. This is only an estimate which should help us avoiding scenarios with regions with a high density of aircraft. Now, given the area A_{ch} of a convex hull that encompasses n_c cells, we can estimate the number of aircraft n_{sep} that can be allocated inside using Eq. 5.26

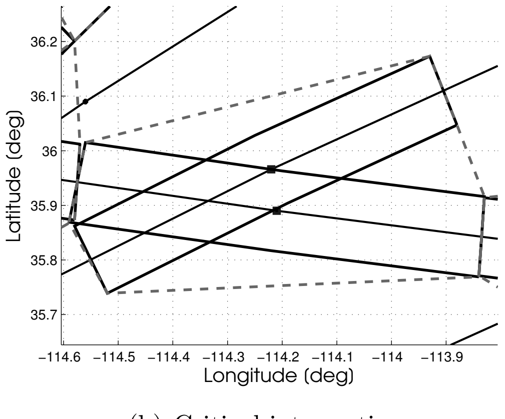
$$n_{sep} = \left\lfloor \frac{A_{ch} \epsilon_a}{A_{na}} \right\rfloor \tag{5.26}$$

Now, if $n_{sep} < \eta_{cc} n_c$, a constraint is posed that limits the summation of the states to be less or equal to n_{sep}.

By using the convex hull, for each group of cells the overall admissible area is overestimated. As a consequence, the admissible number of aircraft within the region might be overestimated as well. Although methods exist to compute the non-convex hull of a set of polygons [2], we consider the convex hull approximation reasonable for our strategic approach for the following reason. The scenario where the convex hull area over-estimation is maximized, is when we have two cells that are perpendicular. This is also the case where the overlapping between the different cells is minimal, thus reducing the chances of separation issues. Although this is still a scenario that ATCs need to monitor, it is less severe than when different routes intersect with a smaller intersection angle, as shown in [53] as well. This is also a case where the convex hull provides a better approximation of the overall area.



(a) Not-critical intersection.



(b) Critical intersection.

Figure 5.7: Examples of intersection between cells that is not labeled (left) and is labeled (right) as critical.

Figure 5.7 shows an example that sheds some light in this sense. In Fig. 5.7(a), the convex hull over-estimates considerably the overall area, and $n_{sep} = 4$, which means that the intersection is not considered critical for separation issues. If we consider the intersection angle and the resulting overlapping area, the non-criticality of the intersection seems reasonable. In Fig. 5.7(b), the convex hull provides a better approximation of the overall area, and $n_{sep} = 3$, since the overlapping is now more consistent and results in a critical intersection.

In matrix form, this inequality constraint has the same structure that characterizes most of the other constraints. We define a matrix M_{sep} that has $n_{sep}N_t$ rows, where n_{sep} is the overall number of critical intersections as described above. In each row, the only non-zero elements are the elements mapping the states of nodes whose cells define the current critical intersection at a specific time-step. We also define a $(n_{sep}N_t, 1)$ vector η_{sep}, where each element defines the maximum number of aircraft the associated critical intersection can accommodate. We can thus write

$$\underbrace{M_{sep}\mathcal{B}_n}_{A^n_{sep}} \mathbf{u}_n \leq \underbrace{\eta_{sep} - M_{sep}(\mathcal{A}_n\mathcal{X}_{0_n} + \mathcal{C}_n b_n)}_{b^n_{sep}} \tag{5.27}$$

5.3.5 Weather-Related Constraints

All the capacity constraints presented so far, are generally time-invariant. They define a baseline that well characterizes the capabilities of airports and sectors inside D. As described in Chap. 2, our model is a benchmark that well describes routes between O-D pairs under "nominal" conditions. In Chap. 3, we show how to modify aggregate routes in presence of weather that would impend the nominal traffic flow. Here, we treat again weather in the IP framework. More specifically, we use the same weather information used to design airborne rerouting branches to generate a set of hard constraints in the form of no-fly zones.

No-fly zones are defined as zones of the airspace that cannot be entered. Some regions of the airspace are military controlled, and thus commercial flights cannot enter such regions (i.e., time-invariant no-fly zones). Since the network we built is based on a historical dataset, it is very unlikely that some of the resulting aggregate routes cross military no-fly zones. On the other hand, some regions of the airspace can become no-fly zones because of convective weather (i.e., time-variant no-fly zones). As shown in Chap. 3, convective weather fronts are not necessarily no-fly zones. First of all, a proper definition of the geometrical shape and severity of convective weather fronts is, by itself, a research topic in the ATM framework [54]. Then, human factors come into play. As [54] points out, the same weather front might be considered safe or unsafe depending on the experience of the pilot, on the advisories received by ATCs, and on many other factors. In our strategic framework, we assume that all convective weather fronts as processed in Chap. 3 are no-fly zones in the IP formulation.

More refined solutions with a finer level of accuracy are left to tactical planners [55].

Unlike the other constraints, this constraint is time-dependent, and the overall number of constraints cannot be determined with precision a priori. Let us assume that the current network component has n^{nc}_{nfz} no-fly zones constraints. n^{nc}_{nfz} is obtained as the summation over all time-steps of all nodes inside any of the detected no-fly zones. A $(n^{nc}_{nfz}, N_n N_t)$ matrix M^{nc}_{nfz} is computed, where the i-th row has a 1 in correspondence of the state that needs to be constrained, and 0 otherwise. Matrix M_{nfz} is obtained stacking in block diagonal fashion the different M^{nc}_{nfz}. The full set of no-fly zones constraints is

$$\underbrace{M_{nfz}\mathcal{B}_n}_{A^n_{nfz}} \mathbf{u}_n \leq \underbrace{\vec{0} - M_{nfz}(\mathcal{A}_n \mathcal{X}_{0_n} + \mathcal{C}_n \mathbf{b}_n)}_{b^n_{nfz}} \tag{5.28}$$

Note that, from a general perspective, Eq. 5.28 should be a set of equality constraints, since we are imposing states in those nodes to be zero. At the same time, given that states are constrained to be non-negative thanks to the combined effect of the other constraints, we can still pose this set of constraints as a set of inequality constraints.

5.4 Cost Function

The cost function to be minimized is a weighted linear combination of control actions. The weights are chosen such that every control action: i) has a positive cost, ii) is penalized less if done later in the planning horizon, and iii) is penalized proportionally to the delay it introduces relative to other control options. The first requirement is satisfied by having all of the weights be positive, ensuring a lower bound of zero. This lower bound is attained if and only if the proposed flight plan for the domain satisfies all of the problem constraints without any control actions.

All the potential control actions are applied at origin nodes (airports, or boundary nodes of D), or at airborne rerouting nodes. We address internal and exiting network components first, and then address entering and overflight network components. For internal and exiting network components, origin nodes correspond to airports. We describe in detail the weighting factors for the controls associated with a particular aggregate route of a generic O-D pair; the remaining weights are specified in the same manner. Denote the number of aggregate routes for the O-D pair by N_r. If $N_r = 1$, there is a single control, ground holding; if $N_r > 1$, there are multiple controls, ground holding plus $N_r - 1$ reroute options. Let d_i denote the delay, given as the additional number of time-steps resulting from the control action u_i, where i indexes the controls for our particular route. The cost function weight for origin node control u_i (ground holding or pre-departure rerouting) at time t_k is defined as

$$w_i(t_k) = d_i\left(1 - \epsilon(k-1)\right) \qquad i = 1, 2, \cdots, N_r \qquad k = 1, 2, \cdots, N_t \tag{5.29}$$

where d_i is the number of time-steps of delay caused by the i-th control action. For ground holding one time-step, $d_i = 1$. For pre-departure rerouting, d_i is equal to the difference between the numbers of time-steps of the alternate route and the scheduled route, unless the alternate route is faster than the scheduled route. If the alternate route is faster, d_i is set to unity to ensure that the control action has a positive cost, and thus only used when necessary to ensure the constraints are satisfied.

The factor $-\epsilon(k-1)$ penalizes control actions earlier in the planning horizon more, the rationale being that there will be subsequent replanning with newer information, thus it is prudent not to react sooner than necessary. ϵ can be tuned to achieve a desired decreasing weight, for the same control action, as time increases. As example, if we want to weigh the same control action at $k = N_t$ with weight η_f instead of 1, we will set $\epsilon = \dfrac{1-\eta_f}{N_t - 1}$.

This factor has the additional effect that if a sequence of ground holding controls and a pre-departure reroute introduce the same delay, ground holding will be preferred. This is because each ground hold delay is multiplied by a factor that decreases with time thanks to ϵ.

For airborne rerouting, the cost function weight for airborne rerouting node control u_i at time t_k is defined as

$$w_i(t_k) = \alpha d_i \left(1 - \epsilon(k-1-p)\right) \qquad i = 1, 2, \cdots, N_r \qquad k = 1, 2, \cdots, N_t \tag{5.30}$$

where α has been defined in Chap. 3 and p is the number of time-steps between the origin node and the airborne rerouting node. We introduce p because an airborne rerouting control at t_k should be compared with a ground holding that starts at t_{k-p}. In Eq. 5.30, we set equal to zero values of $(k-1-p)$ that are negative.

One could argue that, if a pre-departure rerouting introduces a longer route, α should be used in the associated weight as well, since the extra-time is spent while flying. We decided not to apply this additional penalization for the following reason. While an airborne rerouting control is a contingent action (and generally implies an extra fuel consumption), a pre-departure control moves aircraft to another well-traveled route that has been flown consistently according to our historical dataset. Thus, we associate the airborne reroute penalization mainly to operational reasons.

For entering and overflight network components, all controls are formally airborne. Origin nodes are crossing boundary points where new aircraft enter D. Controls here are modeled as holding patterns. On the other hand, airborne rerouting controls are the same as the ones introduced above. For both, the cost function weight for airborne rerouting node control u_i at time t_k is defined as

$$w_i(t_k) = \alpha d(i)\left(1 - \epsilon(k-1)\right) \quad i = 1, 2, \cdots, N_r \quad k = 1, 2, \cdots, N_t \tag{5.31}$$

where p has been dropped since here there's no ground holding for comparison.

The weights for all the controls are arranged into a single vector $\mathbf{w}$ compatible with $\mathbf{u}_n$ so that the cost function may be written as $J = \mathbf{w}^T \mathbf{u}_n$

5.5 Properties of the Solution

The strategic traffic flow planning problem is stated as the following IP

$$\min_{\mathbf{u}_n} J = \mathbf{w}^T \mathbf{u}_n \text{ s.t. } \begin{cases} M_{co}^n \mathbf{u}_n \leq \mathbf{b}_n \\ -M_{ca}^n \mathcal{B}_n \mathbf{u}_n \leq M_{ca}^n (\mathcal{A}_n \mathcal{X}_{0_n} + \mathcal{C}_n \mathbf{b}_n) \\ M_s \mathcal{B}_n \mathbf{u}_n \leq \eta_s - M_s (\mathcal{A}_n \mathcal{X}_{0_n} + \mathcal{C}_n \mathbf{b}_n) \\ M_d^{nc} \mathcal{B}_{nc} \mathbf{u}_{nc} \leq \eta_d^{nc} - M_d^{nc} (\mathcal{A}_{nc} \mathcal{X}_{0_{nc}} + \mathcal{C}_{nc} \mathbf{b}_{nc}) \\ M_a \mathcal{B}_n \mathbf{u}_n \leq \eta_a - M_a (\mathcal{A}_n \mathcal{X}_{0_n} + \mathcal{C}_n \mathbf{b}_n) \\ (M_a \mathcal{B}_n - M_G M_d \mathcal{B}_n) \mathbf{u}_n \leq Q_G - M_a (\mathcal{A}_n \mathcal{X}_{0_n} + \mathcal{C}_n \mathbf{b}_n) + M_G M_d (\mathcal{A}_n \mathcal{X}_{0_n} + \mathcal{C}_n \mathbf{b}_n) \\ M_{d\Delta t} \mathcal{B}_n \mathbf{u}_n \leq \eta_{d\Delta t} - M_{d\Delta t} (\mathcal{A}_n \mathcal{X}_{0_n} + \mathcal{C}_n \mathbf{b}_n) \\ M_{a\Delta t} \mathcal{B}_n \mathbf{u}_n \leq \eta_{a\Delta t} - M_{a\Delta t} (\mathcal{A}_n \mathcal{X}_{0_n} + \mathcal{C}_n \mathbf{b}_n) \\ M_f \mathcal{B}_n \mathbf{u}_n \leq \eta_{cc} - M_f (\mathcal{A}_n \mathcal{X}_{0_n} + \mathcal{C}_n \mathbf{b}_n) \\ M_{sep} \mathcal{B}_n \mathbf{u}_n \leq \eta_{sep} - M_{sep} (\mathcal{A}_n \mathcal{X}_{0_n} + \mathcal{C}_n b_n) \\ M_{nfz} \mathcal{B}_n \mathbf{u}_n \leq \vec{0} - M_{nfz} (\mathcal{A}_n \mathcal{X}_{0_n} + \mathcal{C}_n \mathbf{b}_n) \\ \mathbf{u}_n \geq \vec{0} \end{cases} \tag{5.32}$$

In general, a LP problem is guaranteed to have a minimizing solution when the constraints define a feasible set bounded in the direction of the negative gradient of the cost function. The inequality constraints and lower bounds on controls specify a convex polytope over which the objective function is to be optimized. The solution set can be empty, have a single vertex or multiple solutions corresponding to points on an entire edge of the polytope. As Appendix B shows, with no airborne aircraft at t_0 there always exists a trivial solution to the optimization problem.

Since the controls are integer numbers of aircraft, we require an integer-valued solution to the optimization problem. Some authors have used a LP solver, in particular the simplex method. If the inequality matrix A is unimodular, the simplex method will produce an integer-valued solution to this LP relaxation [73, 81]. Unimodularity can be achieved by approximating routes in sectors as straight lines and by approximating sector boundaries with convex polygons [73, 81]. In the interest of model resolution, we have not made these approximations; thus unimodularity is not guaranteed for our model. We used an IP solver for the tests presented in the next chapter. It guarantees an integer solution with a computational time that fits with the strategic framework. A branch-and-bound method is used to tackle the IP framework, where each sub-problem is solved via simplex or dual-simplex method.

Lastly, the optimal value of the cost function J will be zero if and only if the air traffic flow is feasible with $\mathbf{u}_n = \vec{0}$. This fact suggests that the ARM and planner can be used to evaluate proposed flight plans. If $J = 0$ then the proposed plans are accepted without modification. Otherwise there is a positive minimum cost signifying a constraint violation with the scheduled flight plans; and the optimal control vector defines a cost-minimizing feasible solution.

We envision that the planning results would be used to inform human decision makers and would lead to Traffic Management Initiatives (TMIs). Some current TMIs are Ground Delay Programs, Miles-In-Trail, and Airspace Flow Programs. Our planning results could be translated into these or a different set of TMIs. Given the aggregate nature of the model, the strategic planner determines how many aircraft should be delayed on the ground and how many should be rerouted, but does not specify which aircraft. This specification is left to the airport controllers who will have more information about individual aircraft on which to base decisions.

Chapter 6

Tests of the Strategic Planning Method

The ARM is tested in three scenarios. The first two tests were also presented in [12], and are thus based on a previous version of the clustering algorithm presented in Chap. 2. The purpose of the first test is to assess the accuracy of the ARM with respect to the original historical dataset and introduce and justify the need for replanning. The second test is focused on strategic planning in presence of convective weather with ground controls (i.e., ground holding and pre-departure rerouting). The results motivate the adjustments to address weather-related constraints. The third test addresses the scalability to a bigger planning domain that comprises the six western-most Centers of the NAS. This test employs the complete version of the ARM, where ground holding, pre-departure rerouting and airborne rerouting are all exploited as control actions to optimize the air traffic flow.

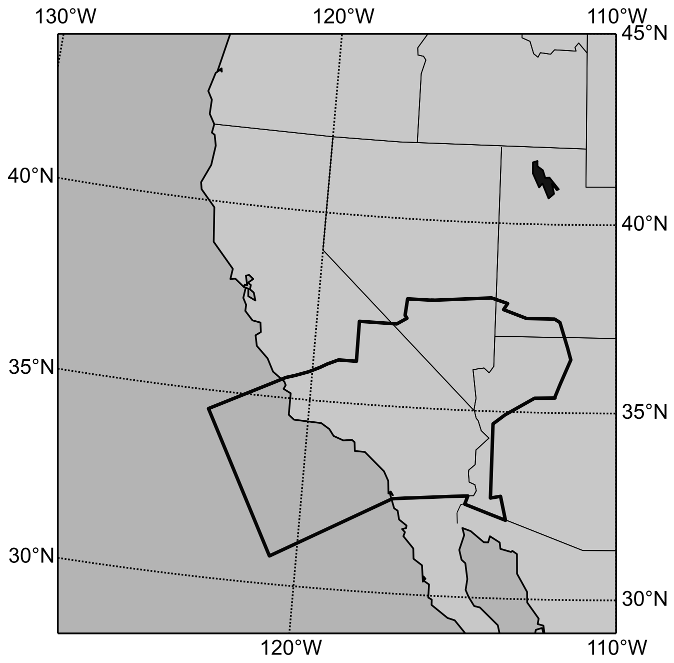


Figure 6.1: Planning domain in Test 1.

6.1 Test 1: ZLA Air Traffic Simulation and Management

In this test, an ARM is constructed and its accuracy in simulating traffic flow is assessed by comparison with the FACET Simulation; then ARM-based strategic planning is used to adjust the flow to eliminate sector capacity violations. The planning domain is the ZLA Center, as shown in Fig. 6.1.

The eight commercial airports considered are Burbank Bob Hope (BUR), Las Vegas McCarran International (LAS), Los Angeles International (LAX), Long Beach (LGB), Ontario International (ONT), San Diego International (SAN), Santa Barbara Municipal (SBA) and Orange County John Wayne (SNA). Using flight trajectory data from October 15th, 2012, from 0:00 to 4:00 UTC from FACET Playback Mode, an ARM with 2587 nodes is built using the algorithm described in [12]. The 25 network components are generated independently and then assembled to obtain the complete network. Table 6.1 shows the number of trajectories for each network component and the overall number of routes for each flight category. In the clustering, a threshold distance of 30 km is used for internal flights (due to the proximity of airports inside the domain) and 60 km is used otherwise. The speed difference threshold is 20 kts. The number of aggregate routes representing internal flights and overflights is half the number of original trajectories, and there is an 80% reduction for entering and exiting flights. The dimension reduction due to aggregation depends on the resolution levels in the clustering similarity measures and on how many trajectories in the dataset are similar. For the present case, a small dataset for a short time period was used; there is not as much opportunity for dimension reduction as there would be with a longer time period when a greater number of similar trajectories would likely be present. To visualize aggregation, we zoom in on the region of the planning domain around LAS and show the before and after views. Figure 6.2 depicts the flight trajectory data; whereas Fig. 6.3 shows the resulting aggregate routes after the trajectory data is processed.

With reference to Eq. 4.3, the smallest and largest network components are characterized by matrices with the dimensions (using the notation (# rows, # columns)) $\mathcal{A}_{nc}(720, 720)$, $\mathcal{C}_{nb}(720, 60)$, $\mathcal{B}_{nc}(720, 60)$ and $\mathcal{A}_{nc}(15300, 15300)$, $\mathcal{C}_{nc}(15300, 1080)$ and $\mathcal{B}_{nc}(15300, 1080)$ respectively. The full network, according to Eq. 4.12, has dimensions $\mathcal{A}_{n}(155220, 155220)$, $\mathcal{C}_{n}(155220, 15360)$ and $\mathcal{B}_{n}(155220, 5580)$.

The accuracy of the ARM for simulation is assessed by comparison with the FACET Simulation Mode. Both simulations are initialized with the same data from FACET for the timespan stated in the previous paragraph. For the high sectors in ZLA, the sector counts from each of the simulations are averaged over all the discrete times within the planning horizon. Figure 6.4, comparing the average sector counts, indicates the accuracy of the ARM. The average difference in aircraft count ranges from 0.1 to 0.8. Figure 6.5 shows the

Table 6.1: All trajectories considered in the 4-hour timespan divided by airport and route category, and the final number of routes for the four different route categories in Test 1.

Airport	Internal	Exiting	Entering	Overflights
BUR	2	19	28	-
LAS	33	109	133	-
LAX	16	140	213	-
LGB	3	10	22	-
ONT	4	14	16	-
SAN	12	36	57	-
SBA	3	12	5	-
SNA	7	34	51	-
Total	80	374	525	134
Routes	35	87	110	60

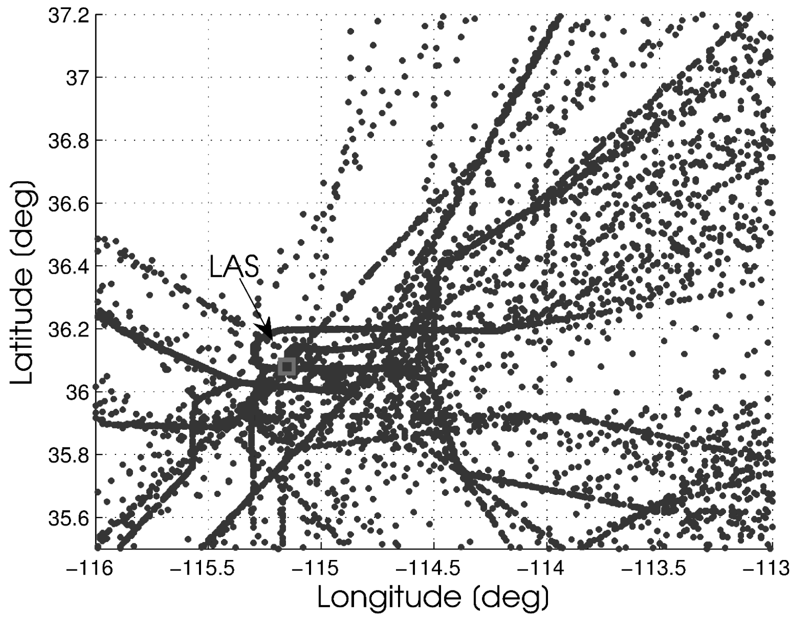


Figure 6.2: LAS region of planning domain and ground tracks for trajectory data used to build the ARM in Test 1.

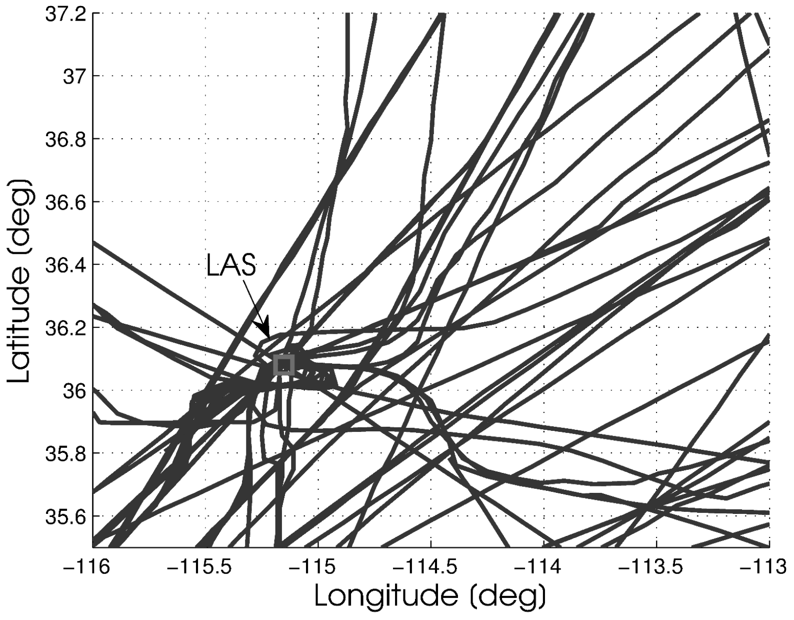


Figure 6.3: Resulting aggregate routes in LAS region in Test 1.

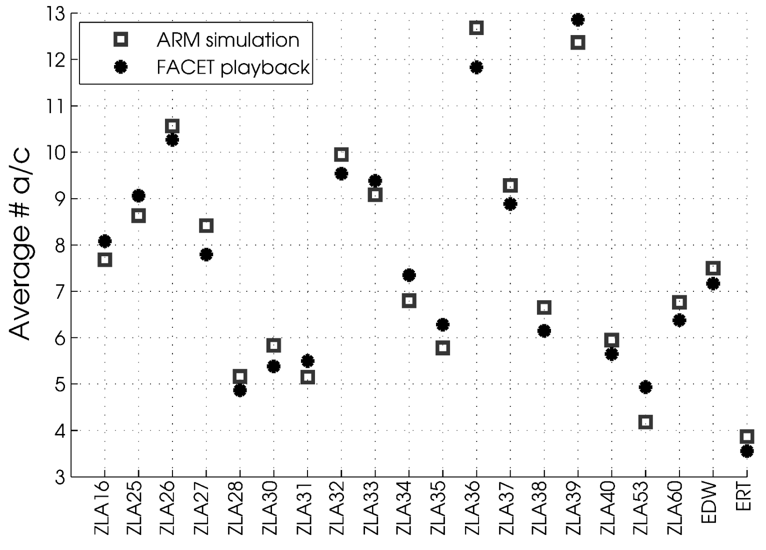


Figure 6.4: Comparison between the ARM and FACET simulations in terms of average sector counts in Test 1.

sector counts versus time for ZLA26 from the ARM and FACET simulations closely match.

Given that the simulated sector counts violate the capacity limits, we proceed to use the ARM-based strategic planner to introduce ground holds and generate a plan that provides a feasible solution minimizing the total delay. Table 6.2 provides the key values for the test. The minimum overall delay for the ZLA center is 3684 min, for an average delay of 3.3 min per aircraft and a worst case delay of 12 min. The planning problem is set up in Matlab® and solved with Gurobi® in 93 s using an Intel i5 processor with 8GB of RAM. Examining sector ZLA26 more closely, Fig. 6.6 shows that the adjusted traffic flow avoids exceeding the capacity limit of 15 aircraft.

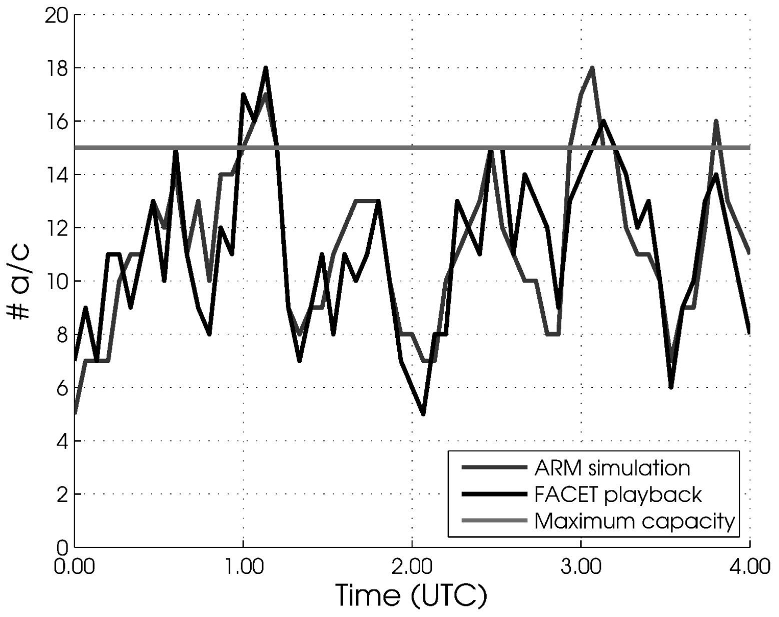


Figure 6.5: Comparing aircraft count in sector ZLA26 for the ARM simulation of the scheduled flight plans and for the optimally planned flow in Test 1.

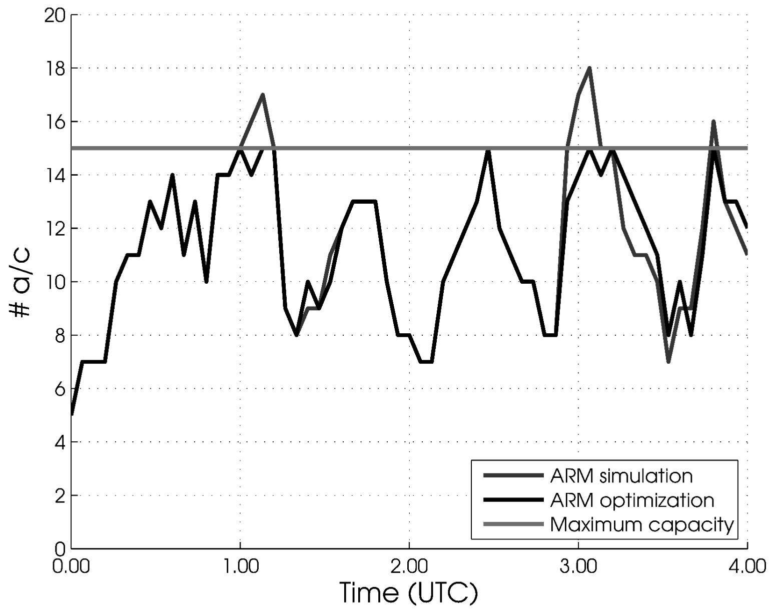


Figure 6.6: Aircraft count in ZLA26 using the ARM without and with optimization in Test 1.

Table 6.2: Key strategic planning quantities in Test 1

# nodes of the ARM	2,587
# aircraft	1,113
# scheduled departures	378
# airports	8
Δt	4 min
Planning Horizon	4 h
Aggregate Routes	292
Constraints in the IP	12,540
Controls in the IP	5,580

6.2 Test 2: Managing Internal Flights in ZLA in Convective Weather

In this test, ground holding and pre-departure rerouting are used to manage air traffic when convective weather is blocking portions of the airspace. Only internal flights are considered. We use the same planning domain as in Test 1, but add the airports Van Nuys (VNY) and Palm Springs (PSP) and drop SBA, for a total of 9 airports. The ARM is constructed from the FACET dataset of recorded trajectories from July 1st to July 14th, 2014, each day between 9 AM and 9 PM Central Time. The ARM-based strategic planner is then used to manage the flow on July 15th, 2014, between 19:00 and 23:00 UTC using initialization data from FACET. Because the planning horizon differs from the timespan for the ARM construction, each departure is not already associated with an aggregate route. For the O-D pair of a particular departure, if there is only a single route, the association is straightforward. Otherwise, the trajectory will be associated with the closest aggregate route based on the Fréchet distance. We note that when the planning horizon is in the future, the flight plan associated with each scheduled departure would be used instead of a recorded trajectory.

Figure 6.7 shows the airports and aggregate routes in the planning domain. The distance and the speed difference thresholds are 30 km and 20 kts, respectively. Table 6.3 gives, for each origin airport, the number of flight trajectories clustered and the resulting number of aggregate routes, showing that the level of aggregation is substantial. With only internal flights in the model, LAS has the most departure routes with at least one aggregate route to each of the other airports. The other airports all have at least one aggregate route to LAS, and in some cases, routes to additional destinations (e.g., LAX to PSP and SAN). The smallest network component is for ONT (18 nodes), where $\mathcal{A}_{nc}(1080, 1080)$, $\mathcal{C}_{nc}(1080, 120)$ and $\mathcal{B}_{nc}(1080, 120)$. The largest network component is for LAS (272 nodes), where $\mathcal{A}_{nc}(32640, 32640)$, $\mathcal{C}_{nc}(32640, 1560)$ and $\mathcal{B}_{nc}(32640, 2760)$. For the overall network

(739 nodes) we have $\mathcal{A}_{nc}(88680, 88680)$, $\mathcal{C}_{nc}(88680, 5160)$ and $\mathcal{B}_{nc}(88680, 15720)$.

Table 6.3: Numbers of trajectories and aggregate routes for each origin airport in Test 2.

Airport	Trajectories	Aggregate Routes
BUR	120	1
LAS	747	13
LAX	493	7
LGB	70	1
ONT	63	1
PSP	70	7
SAN	280	4
SNA	174	7
VNY	104	2
Total	2121	43

Figure 6.7 also shows two weather fronts that create no-fly zones. The first front blocks air traffic from LAS to PSP and SAN from 19:00 to 22:10 UTC. In both cases, a longer alternate route is available. The second front blocks air traffic from BUR, LAX, LGB, SNA and VNY to LAS from 19:00 to 21:00 UTC. In order to demonstrate the airport capacity constraints, we impose a departure/arrival rate of 1 aircraft per time-step for each airport except LAS, for which we choose a value of 2. In addition, we impose for each time-step an arrival rate of 1 aircraft per time-step for flights to LAS from BUR, LAX, LGB, SNA and VNY. The constraint bounds were chosen by analyzing Fig. 6.7. All aggregate routes connecting those airports with LAS merge near LAS; the arrival rate constraint avoids the simultaneous arrival of two aircraft to the same fix.

The problem is set up in Matlab® and solved with Gurobi® in 22 s using an Intel i5 processor with 8GB of RAM. The key quantities for Test 2 are reported in Table 6.4. The results are shown in Table 6.5 and Fig. 6.8. Table 6.5 illustrates how the 3 departures from LAS to SAN are modified to accommodate the weather front. The first 2 departures, scheduled at 20:28 and 21:18 UTC, are rerouted to the alternate route connecting LAS and SAN since the delay caused by ground holding until the weather front disappears is greater

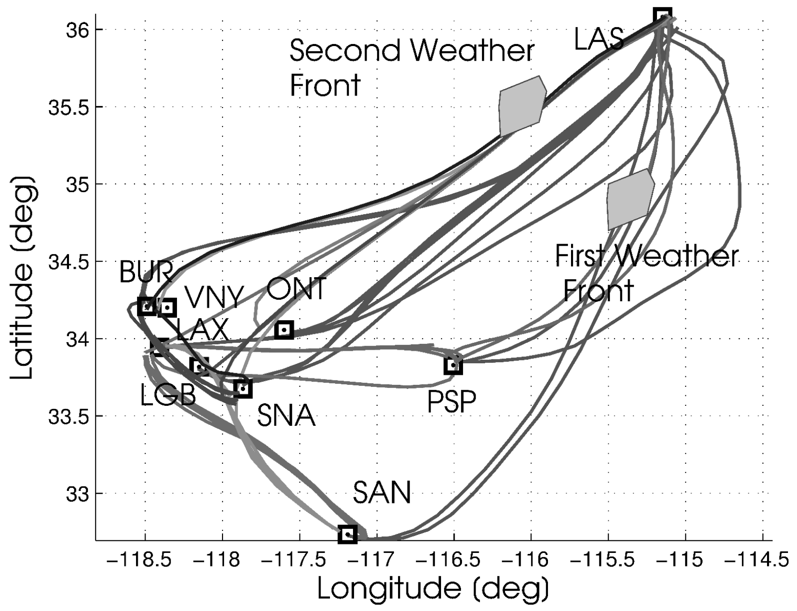


Figure 6.7: Planning domain, airports, and aggregate routes in Test 2.

than the extra 6 minutes required to fly the alternate route. On the other hand, the third departure is scheduled for 21:56 UTC and by ground holding for 6 minutes, the aircraft can follow the scheduled route without entering the weather front. As stated in Chap. 5, the cost function favors the scheduled route unless an alternate route saves time.

Table 6.4: Key strategic planning quantities in Test 2.

# nodes of the ARM	739
# aircraft	58
# scheduled departures	50
# airports	9
Δt	2 min
Planning Horizon	4 h
Aggregate Routes	43
Constraints in the IP	12,300
Controls in the IP	15,720

Figure 6.8 shows how scheduled departures from BUR, LAX, LGB, SNA, and VNY to LAS are modified to satisfy constraints mainly due to the second weather front and the arrival rate in LAS. The location of the front blocks departures from LAX and SNA until 20:40

Table 6.5: Scheduled infeasible departures and revised departures from the strategic planner in Test 2.

	Scheduled departures		Optimized departures	
	Nominal route	Alternate route	Nominal route	Alternate route
20:28	1	0	0	1
21:18	1	0	0	1
21:56	1	0	0	0
22:02	0	0	1	0

UTC, and departures from BUR, LGB, and VNY until 20:42 UTC. Because the routes from BUR, LGB, SNA, and VNY have the same time of flight, while the route from LAX is 2 time-steps longer, no simultaneous departures from any of the first four airports can be scheduled. Further, a departure from any of those 4 airports cannot be scheduled 2 time-steps after a departure from LAX.

Figure 6.8 shows how all departures before 20:40 UTC are pushed back due to the weather front. At some times, two departures are scheduled. One of those is always from LAX, since it will land four minutes later than the other, without violating the arrival constraint. For similar reasons, no departure from BUR, LGB, SNA, or VNY is scheduled two time-steps after a departure from LAX. It is also interesting how coupling via the constraints can affect departures that would not be rescheduled otherwise. As an example, the only departure from BUR was originally scheduled at 20:56 UTC, which is a departure time not affected by the weather front. However, to accommodate the arrival of delayed flights from LAX and VNY at LAS, the departure of the BUR flight is delayed by 8 min to avoid exceeding the arrival capacity at LAS.

The overall delay is 594 min, with an average delay of 11.9 min per aircraft and a worst case delay of 90 min. Although a 90 min delay may seem high, the strategic planner has considered the options that exist and determined the feasible solution that minimizes the cost function, i.e., the weighted sum of delays.

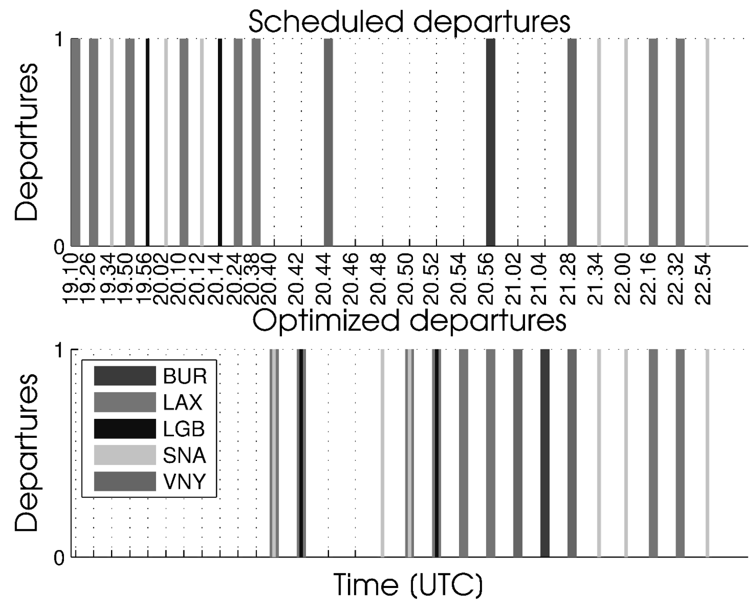


Figure 6.8: Scheduled and optimized departures from BUR, LAX, LGB, SNA, and VNY to LAS in Test 2.

6.3 Test 3: Air Traffic Flow Optimization for the Six Western-Most Centers of the NAS

In this test, the strategic planning problem is formulated and solved for the six western-most Centers of the NAS, i.e., ZLA, ZAB, ZSE, ZOA, ZLC, ZDV. The aggregate routes for this planning domain have been presented in Chap. 2. We analyze the dimension of the model and provide some insights regarding the sparsity of the dynamics matrices. We describe how the initial state vector and the scheduled departures vector are assembled. Then, we describe how the impact of severe weather is accounted for, and how rerouting paths are added to avoid severe weather fronts that impede the scheduled routes. We conclude with a critical analysis of the solution.

In Table 6.6, the number of aggregate routes, the number of nodes, and the number of controls of each network component is shown. The values refer to the nominal scenario given by the algorithm presented in Chap. 2, where no airborne rerouting has been considered yet.

Table 6.6: Number of routes N_r, number of nodes N_n, and number of controls N_c for each network component in Test 3.

	Internal			Exiting			Entering		
	N_r	N_n	N_c	N_r	N_n	N_c	N_r	N_n	N_c
BUR	11	117	15	×	×	×	×	×	×
LAS	34	406	72	12	234	20	21	458	21
LAX	21	337	37	20	393	36	27	664	27
LGB	10	160	14	3	76	3	1	25	1
ONT	13	192	19	5	109	7	4	90	4
OXR	×	×	×	×	×	×	×	×	×
PSP	10	114	20	×	×	×	×	×	×
SAN	20	359	38	10	235	10	14	350	14
SNA	19	238	55	8	195	12	8	200	8
VNY	2	20	2	×	×	×	×	×	×
ABQ	12	223	20	5	38	11	4	36	4
PHX	27	431	18	15	249	55	9	165	9
TUS	8	119	14	5	66	17	4	59	4
PDX	26	523	28	9	220	15	13	370	13
SEA	25	624	18	8	188	14	18	503	18
OAK	17	270	8	4	110	4	6	191	6
SFO	24	366	16	18	398	10	22	470	22
SJC	14	213	6	4	121	6	6	182	6
SMF	18	262	10	6	161	8	8	231	8
SLC	33	561	30	14	232	36	13	246	13
DEN	27	618	20	10	88	30	6	70	6

Originally, 21 airports inside D are considered (i.e., $n_a = 21$). Excluding overflights, a total number of $3n_a = 63$ network components defines the overall aggregate model. As already introduced in Chap. 2, OXR is discarded because its contribution to the air traffic flow is extremely scarce. In addition, BUR, PSP, and VNY only have an internal network component. In total, 54 network components are present.

The aggregate route model is characterized by a time-step $\Delta t = 5\ min$, and the planning horizon is 4 hours, which implies $N_t = 48$. Recalling Eq. 4.12, the dimensions of the different matrices for the full dynamics are as follows. $\mathcal{X}_n = (654528, 1)$, $\mathcal{A}_n = (654528, 654528)$,

$\mathcal{X}_{0n} = (654528, 1)$, $\mathcal{C}_n = (654528, 34128)$, $\mathbf{b}_n = (34128, 1)$, $\mathcal{B}_n = (654528, 56496)$, $\mathbf{u}_n = (56496, 1)$. Note that sparsity properties are fundamental in this context to avoid memory usage issues. As example, for $\mathcal{A}_n$ there are 143547 non-zero entries out of an overall number of entries that is 4.2841^{11}. The sparsity of $\mathcal{A}_n$, defined as percentile ratio between number of non-zero entries and overall entries, is 0.000034%. Storing the non-zero elements in a sparse double-precision matrix, we still generate a 7.5 Mb matrix. Figure 6.9 shows the sparsity of matrix $\mathcal{A}_n$, where the block diagonal structure of the non-zero elements is evident.

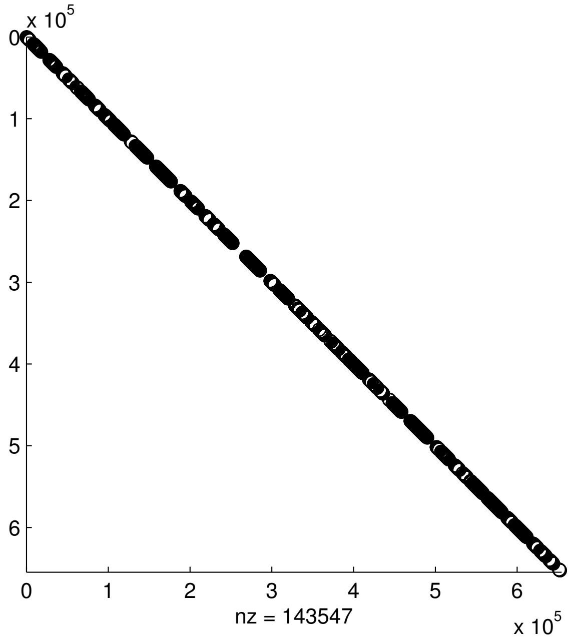


Figure 6.9: Sparsity of matrix $\mathcal{A}_n$ in Test 3.

We now describe how the initial state vector $\mathcal{X}_{0n}$ and the scheduled departure vector $\mathbf{b}_n$ are populated. In a real application, $\mathcal{X}_{0n}$ will contain the predicted position of all aircraft inside D at t_0, and $\mathbf{b}_n$ will contain departures as scheduled by airports for internal and exiting network components, and predicted position of flights entering D via boundary crossing points for entering network components.

In this work, we use historical data to populate the vectors for the test presented. For a given day in the historical dataset, we select an initial time t_0 and a number of hours N_h, and define the time interval $[t_0 \quad t_0 + N_h]$ as the planning horizon of interest. For this test, we select the planning horizon $[15:00 \quad 19:00]$ UTC (i.e., $t_0 = 15:00$ and $N_h = 4$) on July 2nd, 2014. In the planning domain D, this planning horizon ranges from the late morning to the early afternoon, which corresponds to a high inflow and outflow for the airports of interest. Given a network component and the planning horizon, all flights compatible with the network component such that

1. their first recorded time is less than t_0, and their last recorded time is less than $t_0 + N_h$
2. their first recorded time is $\in (t_0 \quad t_0 + N_h]$

are stored. The first category collects all flights that are airborne at t_0, and that will populate the initial state vector $\mathcal{X}_{0nc}$. The second category collects all scheduled departures that will populate the scheduled departure vector $\mathbf{b}_{nc}$.

For each flight of the first category, we initially identify the aggregate route it belongs to. We compute the Fréchet distance between the recorded flight and all the potential aggregate routes, and assign the flight to the aggregate route with the lowest distance. Then, we identify the closest recorded time entry with respect to t_0, and compute the haversine distance between the latitude-longitude pair of that entry and all the nodes of the aggregate route. The flight is assigned to the closest node of the aggregate route. For the network component describing the air traffic flow from LAX and exiting the planning domain D, Fig. 6.10 shows the initial state vector assignment procedure.

For each flight of the second category, the procedure is the same, i.e., the flight is assigned to the aggregate route with the lowest Fréchet distance. The departure time is then assigned to the closest discrete time of the model $t_i = t_0 + i\Delta t$, $i = 1, 2, \cdots, N_t$. In this case, the node assignment is trivial, since departures are assigned to origin nodes.

The overall number of flights in the initial state vector $\mathcal{X}_{0n}$ and in the scheduled departure

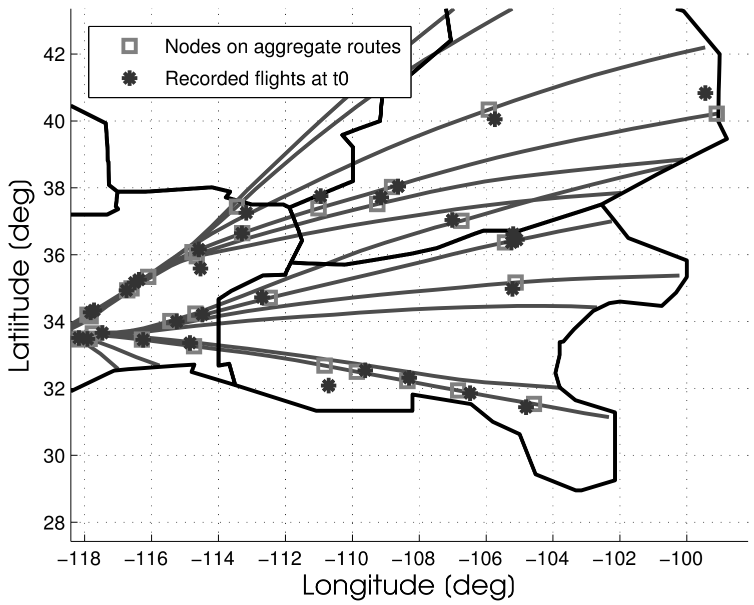


Figure 6.10: Initial state vector assignment example in Test 3. Airborne aircraft (stars) are assigned to the closest node available (squares).

vector $\mathbf{b}_n$ is shown in Table 6.7

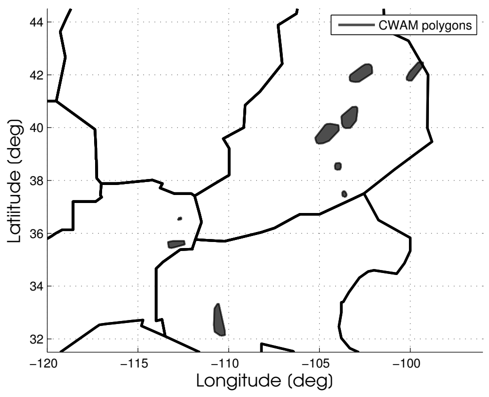
Overall, 466 flights are airborne at t_0, while 2,120 flights are potentially released into the system as part of the scheduled departure vector $\mathbf{b}_n$.

For Test 3, we use real historical weather data to construct weather-related constraints. We process CWAM data from July 7th, 2011, from 0:00 to 4:00 UTC, with a 5-minute time-step. For each time instance, we use the available nowcast. As explained in Chap. 3, in a real-time application with a 4-hour time window, there would be no CWAM availability for the full planning horizon (each CWAM file spans a 2-hour interval from the nowcast, with a 5-minute increment). This work, and this test in particular, assumes the availability of a weather product to process for the planning horizon of interest.

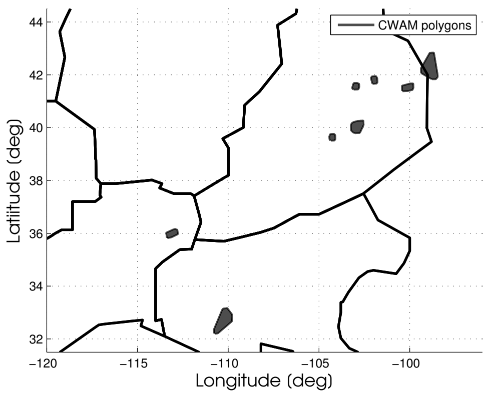
The 60% polygons for FL380 are shown, for some time-steps within the planning horizon, in Fig. 6.11. Severe weather mainly affects the eastern boundary of the planning domain D.

Table 6.7: Elements of $\mathcal{X}_{0n}$ and $\mathbf{b}_n$ for the different network components in Test 3.

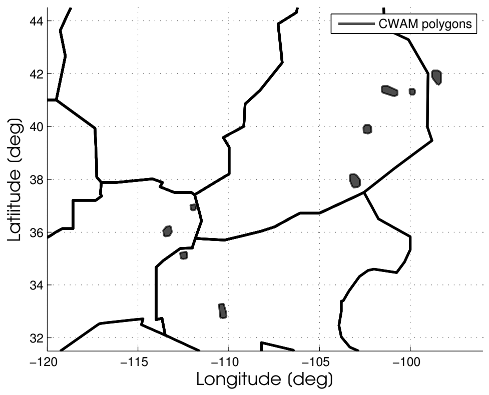
	$\mathcal{X}_{0n}$			$\mathbf{b}_n$		
	Internal	Exiting	Entering	Internal	Exiting	Entering
BUR	6	$\times$	$\times$	11	$\times$	$\times$
LAS	8	11	19	68	90	77
LAX	14	31	39	66	123	102
LGB	3	0	2	3	10	7
ONT	4	3	0	5	12	7
OXR	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
PSP	2	$\times$	$\times$	5	$\times$	$\times$
SAN	9	18	3	42	45	45
SNA	5	3	2	8	45	16
VNY	0	$\times$	$\times$	6	$\times$	$\times$
ABQ	3	1	0	5	16	16
PHX	12	8	21	53	102	66
TUS	0	4	0	8	5	10
PDX	16	12	1	29	28	29
SEA	31	17	10	52	66	65
OAK	3	4	1	28	25	20
SFO	11	13	32	46	71	82
SJC	4	3	1	23	18	18
SMF	4	4	0	23	8	22
SLC	8	9	13	38	72	37
DEN	19	9	10	45	119	72



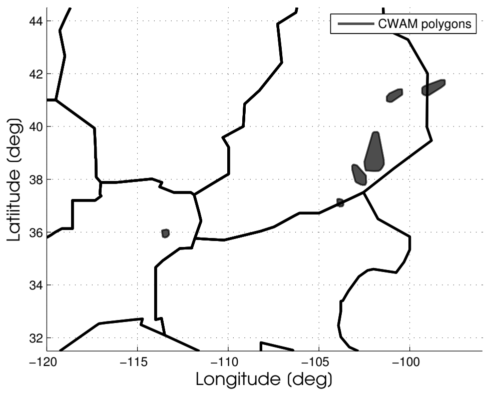
(a) CWAM polygons at t_1.



(b) CWAM polygons at t_3.



(c) CWAM polygons at t_5.

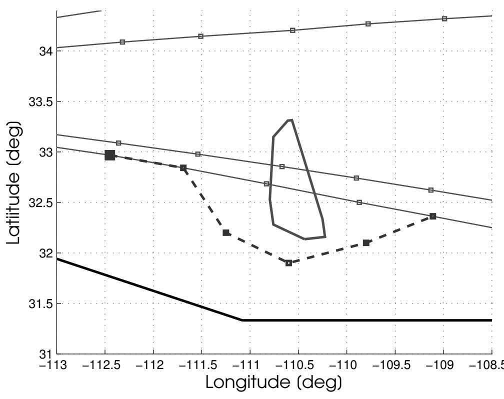


(d) CWAM polygons at t_7.

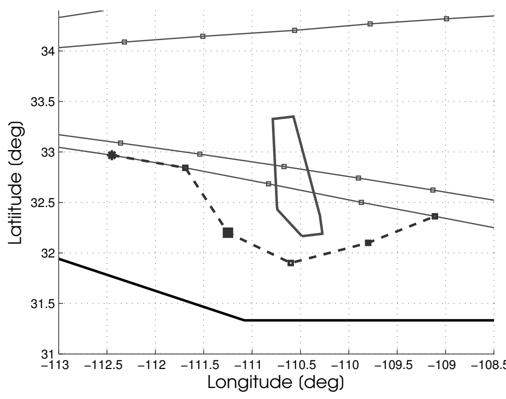
Figure 6.11: CWAM polygons for different time instances used in Test 3.

With the routine presented in Chap. 3, it is checked if airborne flights at t_0 (i.e., elements of the initial state vector) and scheduled departures need to be airborne rerouted. Note that this procedure is especially critical for elements of the initial state vector. In fact, in the worst case scenario scheduled departures can be held on the ground indefinitely via ground stop (see Appendix B). For 12 elements of the initial state vector and 27 scheduled departures along aggregate routes, an airborne rerouting path is computed and added to the associated network component. Weather-related constraints are also computed at this stage by determining, for each time instance, the nodes of the aggregate model that lie inside any of the CWAM polygons.

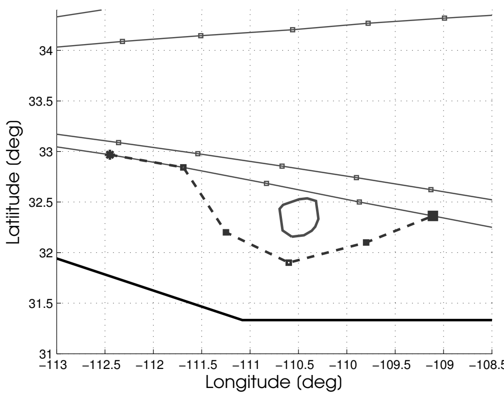
Overall, 163 new nodes are added to the aggregate model. 1,872 control constraints, 1,872 flow constraints and 2,943 weather-related constraints are added to the IP problem. Figure 6.12 shows an example of airborne reroute branch with 3 additional nodes.



(a) Position of aircraft at t_0.



(b) Position of aircraft at t_2.



(c) Position of aircraft at t_5.

Figure 6.12: Example of airborne rerouting path that avoids a convective weather front in Test 3.

The IP problem is posed as described in Chap 5. In Table 6.8, the number of constraints (divided by type) and optimization variables is summarized. Table 6.9 shows all the key quantities for Test 3 instead. The sparsity of the inequality matrix is also shown in Fig. 6.13. Constraints are vertically stacked with the same order they appear in Table 6.8. It is straightforward to identify constraints that do not couple different network components by their diagonally dominated sparsity (control constraints and flow constraints). On the other hand, sector capacity, departure/arrival capacity, weather-related and separation constraints couple controls of different network components, as evident by the two denser regions of the inequality matrix.

Table 6.8: Constraints and optimization variables in Test 3.

Constraints	
Control	36,000
Sector capacity	5,760
Departure capacity (per hour)	80
Arrival capacity (per hour)	80
Departure/Arrival capacity (per hour)	80
Departure capacity (per time-step)	960
Arrival capacity (per time-step)	960
Flow capacity	36,000
Weather-related	2,943
Separation	6,504
Total	89,367
Variables	
Total	58,368

Table 6.9: Key strategic planning quantities in Test 3.

# nodes of the ARM	13,636
# aircraft	2,586
# scheduled departures	2,120
# airports	20
Δt	5 min
Planning Horizon	4 h
Aggregate Routes	711
Constraints in the IP	89,367
Controls in the IP	58,368

The computation of airborne rerouting paths and the assembly of all the matrices required for the IP problem is carried out in Matlab®. The IP problem is solved with Gurobi® via Matlab®interface. The IP problem is solved using using an Intel i5 processor with 8GB RAM. Table 6.10 shows a breakdown of the computational time for the different steps required to solve the problem.

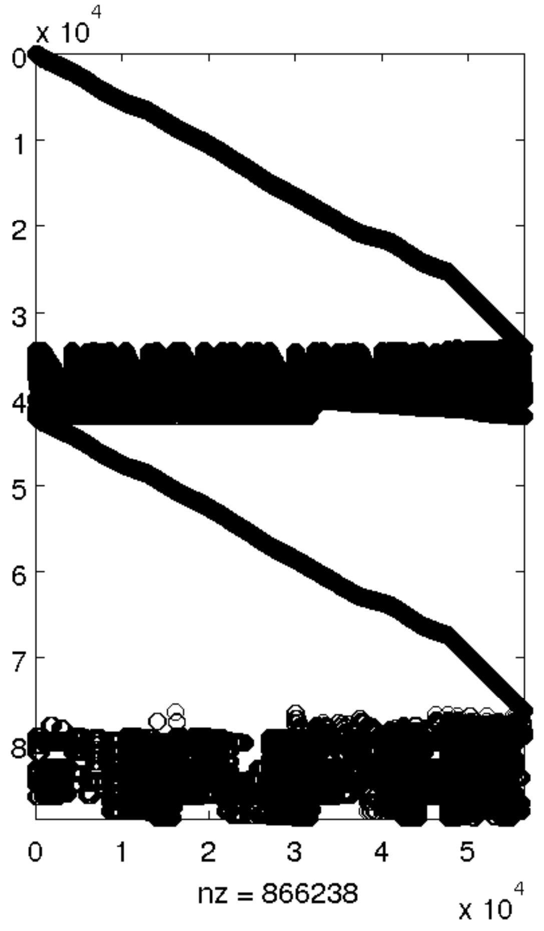


Figure 6.13: Sparsity of the inequality matrix in Test 3.

The IP problem is solved using a branch and bound [41] method[1], and is stopped when the gap optimality is less than 0.3%. The integral solution is obtained using a combination of different cutting planes, e.g., Gomory, MIR, Flow cover, Zero half.

A solution is obtained that minimizes the weighted summation of controls while satisfying the constraints. Optimized departures can be easily retrieved plugging $\mathbf{u}_n$ in Eq. 4.12 and isolating origin nodes for all aggregate routes. Figures 6.14 and 6.15 show optimized departures from LAX compared to scheduled departures. Figure 6.14 compares departures for each time-step $t_i \;\; i = 1, 2, \cdots, 48$, while Fig. 6.15 shows the cumulative summation of departures until the current time-step.

[1]http://www.gurobi.com/resources/getting-started/mip-basics

Table 6.10: Computational time for the different steps required to solve Test 3.

Step	Computational time [s]
Airborne rerouting paths computation	632
Assembly of matrices for IP problem	2
Solution of IP problem	963

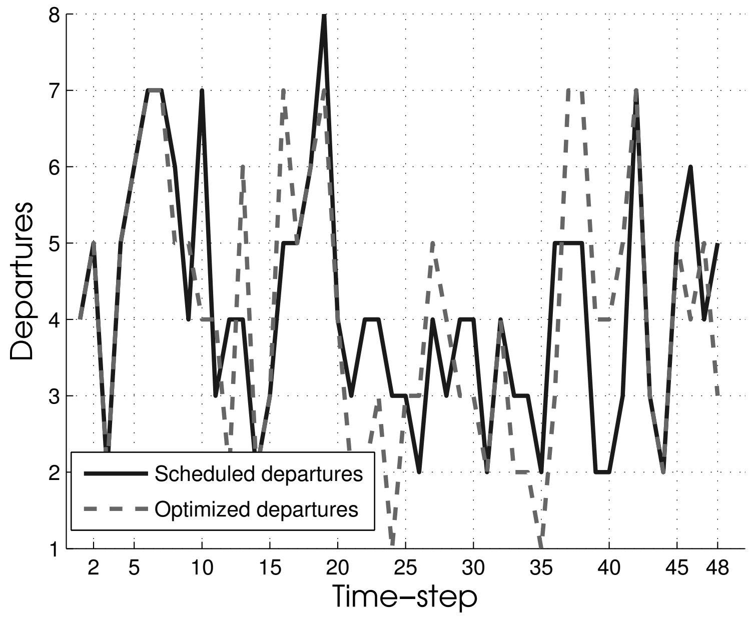


Figure 6.14: Comparison of scheduled and optimized departures (per time-step) for LAX in Test 3.

Out of the 199 scheduled departures, 195 flights actually take off. The 4 ground stops all refer to take offs scheduled for t_{48}, which is the last time-step of the planning horizon. For the remaining 195 departures, 92 flights experience no delay, 53 are delayed of 1 time-step unit Δt (5 min), 40 of two units $2\Delta t$ (10 min), and 10 of three units $3\Delta t$ (15 min). Considering only flights affected by a delay, the average delay per flight is 7.9 min. In Table 6.11, results for all the airports in the planning domain are shown.

Figure 6.16 shows the Gilbo envelope for LAS. The black cross represents the departures/arrivals count for the first hour, which does not satisfy the sloped constraint. The optimized solution, shown with a black cirlce, falls just below the slope, thus satisfying the constraint and

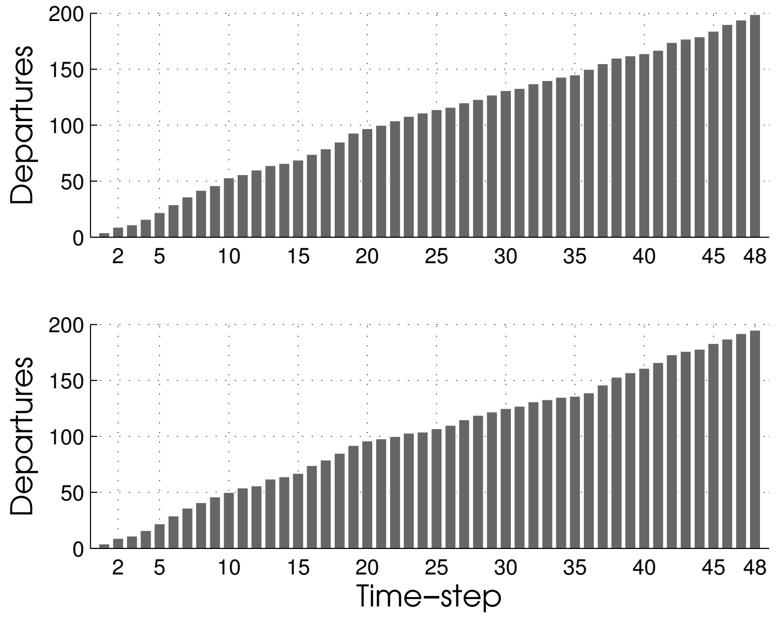


Figure 6.15: Comparison of scheduled and optimized departures (cumulative) for LAX in Test 3.

ensuring the summation of hourly departures and arrivals is maximized.

Figure 6.17 shows another constraint that the scheduled departures plan does not satisfy, and that the optimized departures plan satisfies instead. Figure 6.17(a) shows the sector count for sector ZDV40 as a function of the time-step Δt. It can be appreciated how sector capacity is violated multiple times by the scheduled air traffic flow, while the optimized air traffic flow saturates without exceeding the upper bound. Figure 6.17(b) shows the distribution of states inside ZDV40 for the first time instance (i.e., t_5) when sector capacity is violated for the uncontrolled case. Figure 6.17(c) shows the distribution of states for the same time instance in the controlled case instead. Two aircraft are delayed in the optimized schedule, so that the sector count reaches the upper bound without violating the constraint. The position that the delayed aircraft would have occupied without control (Fig. 6.17(b)), is shown with gray circles in Fig. 6.17(c).

Table 6.11: Scheduled departures, optimized departures, and average delay in Test 3.

Airport	Scheduled dep.	Optimized dep.	Flights delayed	Average delay [min]
BUR	11	11	3	15.0
LAS	158	139	66	10.2
LAX	199	195	107	7.9
LGB	13	13	2	15.0
ONT	17	17	3	6.7
PSP	5	5	3	10.0
SAN	87	82	22	5.6
SNA	53	53	17	5.3
VNY	6	6	3	8.3
ABQ	21	21	5	6.0
PHX	155	149	55	6.95
TUS	13	13	2	5.0
PDX	57	56	10	5.6
SEA	118	108	35	8.2
OAK	53	51	11	5.6
SFO	117	109	34	5.4
SJC	41	41	5	6.0
SMF	31	31	6	5.9
SLC	110	108	33	6.0
DEN	164	159	23	5.6

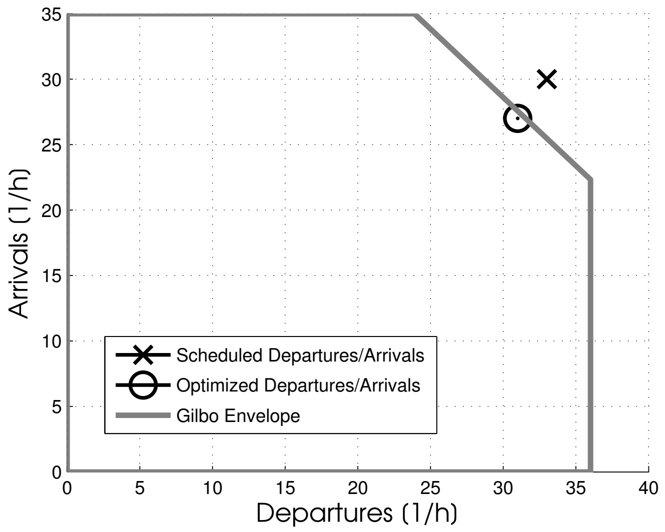
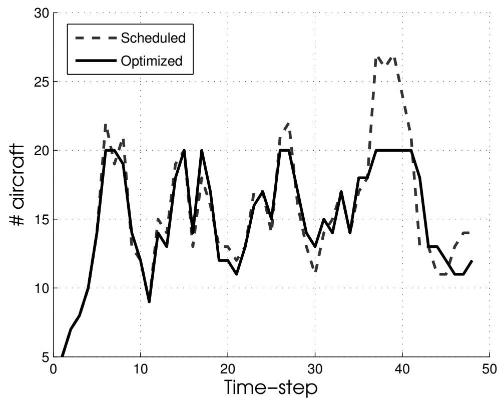
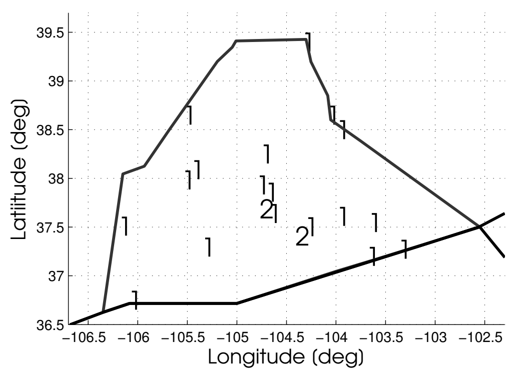


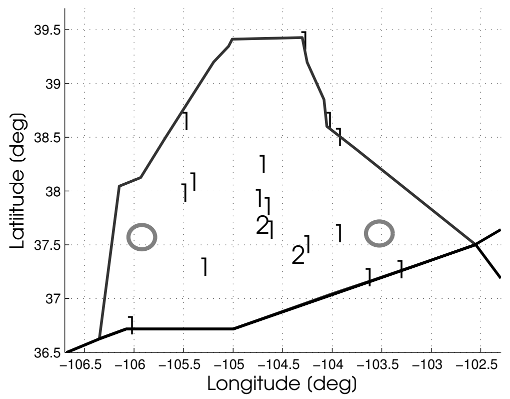
Figure 6.16: Gilbo envelope for LAS, and scheduled and optimized departures/arrivals for the first hour in Test 3.



(a) Sector occupancy during the whole planning horizon.



(b) Scheduled air traffic flow at t_5.



(c) Optimized air traffic flow at t_5.

Figure 6.17: ZDV40 sector occupancy in Test 3.

Chapter 7

Conclusions

A strategic planning approach for air traffic flow management has been developed. The approach is based on an Eulerian aggregate route model. The flow of aircraft is accounted for by the evolution of number of aircraft in discrete cells along the aggregate routes. It has been demonstrated that the model can accurately account for the number of aircraft in sectors by comparison with a high-fidelity Lagrangian model that tracks individual aircraft. With this model, whose dynamics are linear and time-invariant, strategic planning is posed as a constrained integer programming problem, in which schedule delays are minimized subject to capacity constraints. Strategic planning has been demonstrated for several scenarios to illustrate the implementation and the results, and was shown to be effective.

There were several new contributions in the construction of the Eulerian aggregate route model. For clustering historical flight data, the Fréchet distance between flight tracks was used as the accurate measure of spatial similarity. To reduce the computational burden, an initial coarse clustering step was introduced, grouping tracks with the same origin-destination and the same flight category (internal, entering, exiting, overflights). The fine clustering step includes the identification and elimination of outlier flight tracks. The base aggregate route model only includes well-traveled routes for clear weather days. Prior to application to a planning horizon with predicted delays due to weather, appropriate airborne reroute options are added to the model. Both the clustering and airborne reroute features were demonstrated to be effective.

The final, most challenging, strategic planning demonstration was for the six western-most Centers of the 22-Center national airspace. The planning time horizon was four hours long, and there was weather predicted that caused significant delays to the scheduled flights. Airborne reroute options were computed and added to the route model and it was demonstrated that the predicted delays could be significantly reduced. The example also demonstrated the computational feasibility of the approach. Assessing the viability of the approach for the entire national airspace remains for future work.

Bibliography

[1] Addario-Berry, L., Havet, F., Sales, C. L., Reed, B., and Thomass, S. Oriented Trees in Digraphs. *Discrete Mathematics 313*, 8 (2013), 967–974.

[2] Akkiraju, N., Edelsbrunner, H., and Facello, M. Alpha Shapes: Definition and Software. In *Proceedings of the 1st International Computational Geometry Software Workshop* (1995), pp. 63–66.

[3] Almeida, J., Barbosa, L., Pais, A., and Formosinho, S. Improving Hierarchical Cluster Analysis: A New Method with Outlier Detection and Automatic Clustering. *Chemometrics and Intelligent Laboratory Systems 87* (2007), 208–217.

[4] Alt, H., and Godau, M. Computing the Fréchet Distance Between Two Polygonal Curves. *International Journal of Computational Geometry & Applications 5*, 1–2 (1995), 75–91.

[5] Alt, H., Knauer, C., and Wenk, C. Matching Polygonal Curves with Respect to the Fréchet Distance. In *18th Annual Symposium on Theoretical Aspects of Computer Science* (2001), STACS.

[6] Amburn, S. A., and Wolf, P. L. VIL Density as a Hail Indicator. *Weather and Forecasting 12* (1997), 473–478.

[7] Andersson Granberg, T., Polishchuk, T., Polishchuk, V., and Schmidt, C. Automatic Design of Aircraft Arrival Routes with Limited Turning Angle. In *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems* (2016), ATMOS.

[8] Bertsimas, D., Lulli, G., and Odoni, A. An Integer Optimization Approach to Large-Scale Air Traffic Flow Management. *Operations Research 59*, 1 (2011), 211–227.

[9] Bertsimas, D., and Patterson, S. S. The Air Traffic Flow Management Problem with Enroute Capacities. *Operations Research 46* (1998), 406–422.

[10] Bilimoria, K., Shridar, B., Chatterij, G., Sheth, K., and Grabbe, S. FACET: Future ATM Concepts Evaluation Tool. In *3rd USA/Europe Air Traffic Management R&D Seminar* (2000).

[11] Bokadia, S., and Valasek, J. Severe Weather Avoidance using Informed Heuristic Search. In *AIAA Guidance, Navigation, and Control Conference and Exhibit* (2001).

[12] Bombelli, A., Soler, L., Trumbauer, E., and Mease, K. D. Strategic Air Traffic Planning with Fréchet Distance Aggregation and Rerouting. In *AIAA Aviation 2015 Conference* (2015).

[13] Bombelli, A., Soler, L., Trumbauer, E., and Mease, K. D. Strategic Air Traffic Planning with Fréchet Distance Aggregation and Rerouting. *Journal of Guidance, Control, and Dynamics 40*, 5 (2017), 1117–1129.

[14] Bombelli, A., Torné, A. S., Trumbauer, E., and Mease, K. D. Automated Route Clustering for Air Traffic Modeling. In *AIAA Aviation 2017 Conference* (2017).

[15] Bosson, C. S., Xue, M., and Zelinski, S. J. Optimizing Integrated Terminal Airspace Operations Under Uncertainty. In *Digital Avionics Systems Conference (DASC), 2014 IEEE/AIAA 33rd* (2014), IEEE.

[16] Cao, Y., and Sun, D. Link Transmission Model for Air Traffic Flow Management. *Journal of Guidance, Control, and Dynamics 34*, 5 (2011), 1342–1351.

[17] Chan, W. N., Refai, M., and DeLaura, R. An Approach to Verify a Model for Translating Convective Weather Information to Air Traffic Management Impact. In *AIAA Aviation 2007 Conference* (2007).

[18] Chen, D., Driemel, A., Guibas, L. J., Nguyen, A., and Wenk, C. Approximate Map Matching with Respect to the Fréchet Distance. In *Proceedings of the 13th Workshop on Algorithm Engineering & Experiments* (2011), ALENEX11.

[19] Clemons, E., DeLaura, R., Glina, Y., Jordan, R., Proschitsky, A., Reynolds, T. G., Avery, J., Balakrishnan, H., Brooks, C., Murca, M. C. R., et al. Multi-Scale Data Mining for Air Transportation System Diagnostics. In *AIAA Aviation 2016 Conference* (2016).

[20] Cook, A., and Tanner, G. European Airline Delay Cost Reference Values. Tech. rep., University of Westminster, 2015.

[21] Cook, A. J., Tanner, G., and Anderson, S. Evaluating the True Cost to Airlines of One Minute of Airborne or Ground Delay. Tech. rep., Eurocontrol, 2004.

[22] Cormen, T., Leiserson, C. E., Rivest, R., and Stein, C. *Introduction to Algorithms*, 2nd ed. McGraw-Hill Higher Education, 2001.

[23] Daganzo, C. F. The Cell Transmission Model: a Dynamic Representation of Highway Traffic Consistent with the Hydrodynamic Theory. *Transportation Research Part B: Methodological 28*, 4 (1994), 269–287.

[24] Dasgupta, S. Learning Polytrees. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* (San Francisco, CA, USA, 1999), UAI'99, Morgan Kaufmann Publishers Inc., pp. 134–141.

[25] Davies, D. L., and Donald, W. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence 1* (1979), 224–227.

[26] Delaura, R., and Evans, J. An Exploratory Study of Modeling Enroute Pilot Convective Storm Flight Deviation Behavior. In *Proceedings of the 12th Conference on Aviation, Range, and Aerospace Meteorology* (2006).

[27] Dijkstra, E. W. A Note on Two Problems in Connexion with Graphs. *Numer. Math. 1*, 1 (Dec. 1959), 269–271.

[28] Douglas, D., and Peucker, T. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. *The Canadian Cartographer 10* (1973), 112–122.

[29] Dunn, J. C. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal Of Cybernetics 3*, 3 (1973).

[30] Eppstein, D. Finding the k Shortest Paths. *SIAM Journal on Computing 28*, 2 (1998), 652–673.

[31] EUROCONTROL. Standard Inputs for EUROCONTROL Cost-Benefit Analyses. Tech. rep., EUROCONTROL, 2015.

[32] Evans, J. E., and Ducot, E. R. Corridor Integrated Weather System. *Lincoln Laboratory Journal 16*, 1 (2006), 59–80.

[33] Ferreira, L., and Hitchcock, D. B. A Comparison of Hierarchical Methods for Clustering Functional Data. *Communications in Statistics-Simulation and Computation 38*, 9 (2009), 1925–1949.

[34] Gilbo, E. P. Airport Capacity: Representation, Estimation, Optimization. *IEEE Transactions on Control Systems Technology 1*, 3 (1993), 144–154.

[35] Hart, P. E., Nilsson, N. J., and Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics 4*, 2 (July 1968), 100–107.

[36] Hayashi, M., Hoang, T., Jung, Y. C., Gupta, G., Malik, W. A., and Dulchinos, V. L. Usability Evaluation of the Spot and Runway Departure Advisor (SARDA) Concept in a Dallas/Fort Worth Airport Tower Simulation. In *Tenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2013)* (2013).

[37] Hoang, T., Jung, Y. C., Holbrook, J. B., and Malik, W. A. Tower Controllers Assessment of the Spot and Runway Departure Advisor (SARDA) Concept. In *Ninth USA/Europe Air Traffic Management Research and Development Seminar (ATM2011)* (2011).

[38] Kellner, S. Airport Capacity Benchmarking by Density Plots. In *German Aviation Research Society (GARS) Seminar* (2009).

[39] Kleinberg, J., and Tardos, E. *Algorithm Design.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[40] Klingle-Wilson, D., and Evans, D. J. Description of the Corridor Integrated Weather System (CIWS) Weather Products. Tech. rep., MIT Lincoln Laboratory, 2005.

[41] Klotz, E., and Newman, A. M. Practical Guidelines for Solving Difficult Mixed Integer Linear Programs. *Surveys in Operations Research and Management Science 18*, 1 (2013), 18–32.

[42] Knuth, D. E. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[43] Krozel, J., Lee, C., and Mitchell, J. S. B. Turn-Constrained Route Planning for Avoiding Hazardous Weather. *Air Traffic Control Quarterly 14*, 2 (2006), 159–182.

[44] Kuffner, J. J., and LaValle, S. M. RRT-Connect: An Efficient Approach to Single-Query Path Planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on* (2000), vol. 2, IEEE, pp. 995–1001.

[45] Lavalle, S. M. Rapidly-Exploring Random Trees: A New Tool for Path Planning. Tech. rep., Iowa State University, 1998.

[46] LaValle, S. M. *Planning Algorithms.* Cambridge university press, 2006.

[47] LaValle, S. M., and Kuffner Jr, J. J. Rapidly-Exploring Random Trees: Progress and Prospects, 2000.

[48] LAWA Los Angeles World Airports. Los Angeles International Airport - Preferential Runway Use Policy. Tech. rep., Environmental Services Division Noise Management Section, 2014.

[49] Love, J. F., Chan, W. N., and Lee, C. H. Analysis of Automated Aircraft Conflict Resolution and Weather Avoidance. In *AIAA Aviation 2009 Conference* (2009).

[50] Luenberger, D. G., and Ye, Y. *Linear and Nonlinear Programming.* Springer Publishing Company, Incorporated, 2015.

[51] Maheshwari, A., and Yi, J. On Computing Frèchet Distance of Two Paths on a Convex Polyhedron. In *Proceedings of the 21st European Workshop on Computational Geometry* (2005), EWCG.

[52] Martin, E., et al. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *2nd International Conference on Knowledge Discovery and Data Mining* (1996), KDD-96.

[53] Marzuoli, A., Gariel, M., Vela, P. A., and Feron, E. Data-Based Modeling and Optimization of En Route Traffic. *Journal of Guidance, Control, and Dynamics 37*, 6 (2014), 1930–1945.

[54] Matthews, M. P., and DeLaura, R. Assessment and Interpretation of En Route Weather Avoidance Fields from the Convective Weather Avoidance Model. In *AIAA Aviation 2010 Conference* (2010).

[55] McNally, D., Sheth, K., Gong, C., Love, J., Lee, C. H., Sahlman, S., and Cheng, J. H. Dynamic Weather Routes: a Weather Avoidance System for Near-Term Trajectory-Based Operations. In *28th International Congress of the Aeronautical Sciences* (2012).

[56] Menon, P. K., Sweriduk, G. D., and Bilimoria, K. D. New Approach for Modeling, Analysis, and Control of Air Traffic Flow. *Journal of Guidance, Control, and Dynamics 27*, 5 (2004), 737–744.

[57] Menon, P. K., Sweriduk, G. D., Lam, T., Cheng, V. H. L., and Bilimoria, K. D. Air Traffic Flow Modeling, Analysis and Control. In *AIAA Conference on Guidance, Navigation, and Control* (2003).

[58] Murca, M. C. R., DeLaura, R., Hansman, R., Jordan, R., Reynolds, T., and Balakrishnan, H. Trajectory Clustering and Classification for Characterization of Air Traffic Flows. In *AIAA Aviation 2016 Conference* (2016).

[59] Newell, G. F. A Simplified Theory of Kinematic Waves in Highway Traffic, Part I: General Theory. *Transportation Research Part B: Methodological 27*, 4 (1993), 281–287.

[60] Odoni, A. R. *The Flow Management Problem in Air Traffic Control.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1987, pp. 269–288.

[61] Papadimitriou, C. H., and Steiglitz, K. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[62] Ramer, U. An Iterative Procedure for the Polygonal Approximation of Plane Curves. *Computer Graphics and Image Processing 1* (1972), 244–256.

[63] Rehm, F. Clustering of Flight Tracks. In *AIAA InfoTech @Aerospace* (2010).

[64] Rios, J., Jehlen, R., and Shu, Z. A Spatial Database for Reroute Planning. In *31st Digital Avionics Systems Conference* (2012), DASC.

[65] Rousseeuw, P. J. Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Computational and Applied Mathematics 20* (2009), 53–65.

[66] Rubnich, M., and DeLaura, R. An Algorithm to Identify Robust Convective Weather Avoidance Polygons in En Route Airspace. In *AIAA Aviation 2010 Conference* (2010).

[67] Sadovsky, A. V., and Bilimoria, K. D. Risk-Hedged Approach for Re-routing Air Traffic Under Weather Uncertainty. In *AIAA Aviation 2016 Conference* (2016).

[68] Saunders, P., and Ronne, F. A Comparison Between the Height of Cumulus Clouds and the Height of Radar Echoes Received from Them. *Journal of Applied Metereology 1* (1962), 296–302.

[69] Solow, D. *Linear Programming; an Introduction to Finite Improvements Algorithms.* Dover Publications, Inc., Mineola, NY, USA, 2014.

[70] Sridhar, B., Soni, T., Sheth, K., and Chatterji, G. Aggregate Flow Model for Air-Traffic Management. *Journal of Guidance, Control, and Dynamics 29*, 4 (2006), 992–997.

[71] Sultan, A. *Linear Programming; an Introduction with Applications (Second Edition).* Department of Mathematics, Queens College of CUNY, Flushing, NY, USA, 2016.

[72] Sun, D., and Bayen, A. M. Multicommodity Eulerian-Lagrangian Large-Capacity Cell Transmission Model for En Route Traffic. *Journal of Guidance, Control, and Dynamics 31*, 3 (2008), 616–628.

[73] Sun, D., Clinet, A., and Bayen, A. M. A Dual Decomposition Method for Sector Capacity Constrained Traffic Flow Optimization. *Transportation Research Part B 45* (2011), 880–902.

[74] Taylor, C., and Wanke, C. Dynamic Generation of Operationally Acceptable Reroutes. In *AIAA Aviation 2009 Conference* (2009).

[75] Taylor, C., and Wanke, C. Generating Operationally-Acceptable Reroutes Using Simulated Annealing. In *AIAA Aviation 2010 Conference* (2010).

[76] Taylor, C., and Wanke, C. Dynamically Generating Operationally-Acceptable Reroutes Using Simulated Annealing. *Air Traffic Control Quarterly 20*, 1 (2012), 97–121.

[77] Tibshirani, R., Walther, G., and Hastie, T. Estimating the Number of Clusters in a Data Set via the Gap Statistic. *Journal of Royal Statistical Society 63*, 3 (2001), 411–423.

[78] Trudeau, R. J. *Introduction to Graph Theory.* Dover Publications, Inc., 2013.

[79] Van Brummelen, G. *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry.* Princeton University Press, 2012.

[80] Vranas, P., Bertsimas, D., and Odoni, A. R. The Multi-Airport Ground-Holding Problem in Air Traffic Control. *Operations Research 42* (1994), 249–261.

[81] Wei, P., Cao, Y., and Sun, D. Total Unimodularity and Decomposition Method for Large-Scale Air Traffic Cell Transmission Model. *Transportation Research Part B 53* (2013), 1–16.

[82] Xue, M. Airspace Sector Redesign Based on Voronoi Diagrams. *Journal of Aerospace Computing, Information, and Communication 6* (2009), 624–634.

[83] Zeng, W., and Church, R. L. Finding Shortest Paths on Real Road Networks: The Case for A*. *International Journal of Geographical Information Science 23*, 4 (Apr. 2009), 531–543.

[84] Zhang, W., Kargampour, M., Sun, D., and Tomlin, C. J. A Hierarchical Flight Planning Framework for Air Traffic Management. *Proceedings of the IEEE 100*, 1 (2011), 179–194.

Appendices

A List of Acronyms

Acronym	Full Name
ARM	Aggregate Route Model
ASPM	Aviation System Performance Metric
ATC	Air Traffic Controller
ATM	Air Traffic Management
CIWS	Corridor Integrated Weather System
CWAM	Convective Weather Avoidance Model
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
FAA	Federal Aviation Administration
FACET	Future ATM Concept Evaluation Tool
IP	Integer Programming
LP	Linear Programming
LTV	Linear Time-Varying
NAS	National Airspace System
O-D	Origin-Destination
RDP	Ramer-Douglas-Peucker

SD	Strategic Delay
TDWNE	Tactical Delay With Network Effect
TDWONE	Tactical Delay Without Network Effect
TFM	Traffic Flow Management
TFMP	Traffic Flow Management Problem
TMI	Traffic Management Initiative
TRX	FACET Track File
UTC	Coordinated Universal Time
VIL	Vertically Integrated Liquid

Airport	Full name
ABQ	Albuquerque International Sunport airport
BUR	Burbank Bob Hope airport
DEN	Denver International airport
LAS	Las Vegas McCarran International airport
LAX	Los Angeles International airport
LGB	Long Beach airport
OAK	Oakland International airport
ONT	Ontario International airport
OXR	Oxnard airport
PDX	Portland International airport
PHX	Phoenix Sky Harbor International airport
PSP	Palm Springs International airport
SAN	San Diego International airport
SBA	Santa Barbara Municipal airport
SEA	Seattle-Tacoma International airport
SFO	San Francisco International airport
SJC	Norman Y. Mineta San José International airport

SLC	Salt Lake City International airport
SMF	Sacramento International airport
SNA	John Wayne airport
TUS	Tucson International airport
VNY	Van Nuys airport

Center	Full name
ZAB	Albuquerque Center
ZDV	Denver Center
ZLA	Los Angeles Center
ZLC	Salt Lake City Center
ZOA	Oakland Center
ZSE	Seattle Center

B Ground Stop as Feasible Solution of the IP Problem

In this appendix we show how, under the assumption of a zero initial state vector, the IP problem is always solvable, with the worst case scenario solution being a ground stop for all scheduled departures. We show the result for a single aggregate route of a network component. The concept is easily applicable to all aggregate routes of the ARM.

Consider a planning domain with N_t time-steps $t_1, t_2, \cdots, t_{N_t}$. Scheduled departures are $\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_{N_t}$. Optimized departures along aggregate route k at time t_1 are

$$\mathbf{b}_{t_1} - g(t_1) - \sum_{j=1}^{N_r} p_{k\to j}(t_1) \tag{B.1}$$

while for every following time-step t_i they are

$$\mathbf{b}_{t_i} + g(t_{i-1}) - g(t_i) - \sum_{j=1}^{N_r} p_{k\to j}(t_i) \tag{B.2}$$

Considering no departures, all pre-departure controls will be identically zero (i.e., $p_{k\to j} = 0\,\forall\, j$). Additionally, the control inequality constraints

$$\begin{aligned} g(t_1) &\leq \mathbf{b}_{t_1} \\ g(t_2) &\leq \mathbf{b}_{t_2} + g(t_1) \\ &\dots \\ g(t_{N_t}) &\leq \mathbf{b}_{t_{N_t}} + g(t_{N_t-1}) \end{aligned}$$

become equalities since departures are not allowed. If we sum all the left hand sides and right hand sides we obtain

$$g(t_{N_t}) = \sum_{i=1}^{N_t} \mathbf{b}_i \tag{B.3}$$

The cumulative summation of the optimized departures is $\mathbf{b}_1 - g(t_1) + \mathbf{b}_2 + g(t_1) - g(t_2) +$

$\cdots + \mathbf{b}_{N_t} + g(t_{N_t-1}) - g(t_{N_t})$, or equivalently

$$\sum_{i=1}^{N_t} \mathbf{b}_i - g(t_{N_t}) \tag{B.4}$$

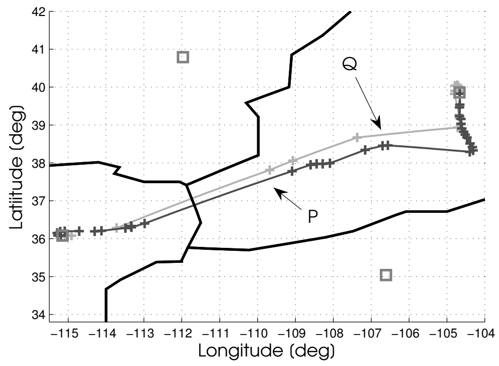
which is identically zero using Eq. B.3. Thus, neglecting the initial state vector, the system can always react to the set of constraints by imposing a ground stop to all flights that exploits cycles in origin nodes of the network. It is clearly a scenario which is not desirable from an operational perspective, but analytically the problem is still solvable.

C Overlapping Index Algorithm

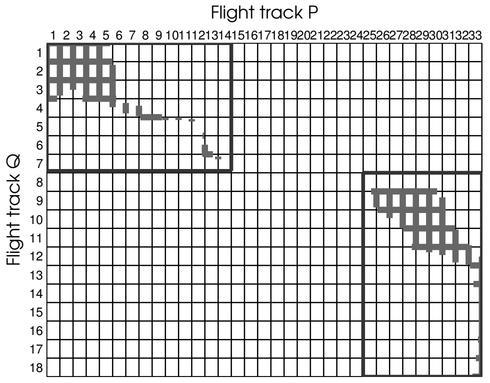
In this appendix we provide some insights on an index, defined overlapping index, that is based on an application of the Fréchet distance. Instead of computing the Fréchet distance between two great circle arc sequences $\mathcal{P}$ and $\mathcal{Q}$, goal of the index is to identify common regions between $\mathcal{P}$ and $\mathcal{Q}$ characterized by a "local" Fréchet distance that can be either (i) a fraction of the Fréchet distance characterizing the two curves, or (ii) a fixed distance value. For each of the two tracks, the associated overlapping index is the ratio between the length of the track characterized by such "local" Fréchet distance, and the overall length of the track (i.e., the overlapping index ranges between zero and one).

As example, given two flight tracks characterized by a crossing point, it could be interesting to identify the common region where two aircraft, one on each track, can incur in loss of separation (in this case, a fixed distance can be specified). Another application addresses the categorization of outlier tracks computed as described in Sec. 2.3.3. By computing the overlapping index between an outlier and its nearest neighbor (i.e., the track with the smallest Fréchet distance $\mathcal{F}$ with respect to the outlier), some insights on why the outlier was labeled as such can be extrapolated. This second application is what will be primarily addressed here.

The first step in the computation of the overlapping index, is to detect for each outlier the nearest-neighbor ground track. Then, a value for α ($0 \leq \alpha \leq 1$) is chosen, and a free space diagram is generated using $\alpha\mathcal{F}$ as distance (as mentioned before, α can be chosen such that either the desired distance is a certain percentage of $\mathcal{F}$ or a fixed distance, as long as the distance is smaller than $\mathcal{F}$). Figure C.1(a) shows $\mathcal{P}$, one of the three outliers described in Sec. 2.3.3, and the nearest-neighbor $\mathcal{Q}$. They are characterized by a Fréchet distance $\mathcal{F} = 76.7\ km$. Figure C.1(b) shows the free space computed using $\alpha = 0.4$, i.e., using a distance of $30.7\ km$.



(a) Two ground tracks $\mathcal{P}$ and $\mathcal{Q}$.



(b) Free space diagram computed using $\alpha\mathcal{F}$ ($\alpha = 0.4$) as distance.

Figure C.1: Example of how, given two ground tracks $\mathcal{P}$ and $\mathcal{Q}$, their Fréchet distance $\mathcal{F}$, and α, the free space diagram associated with $\alpha\mathcal{F}$ is generated and overlapping regions are identified.

Since a fraction of the Fréchet distance is used, a monotone path from the upper left corner to the lower right corner does not exist in Fig. C.1(b). On the other hand, some regions characterized by a monotone path exist. Each of these regions is characterized by a "local" Fréchet distance $\alpha\mathcal{F}$. In Fig. C.1(b) the two regions where a monotone path exists are highlighted with a gray box. The first region starts from the origin of both tracks, while the second ends in correspondence to the endpoints of the two tracks. This is consistent with the fact that the tracks share the same O-D airport pair.

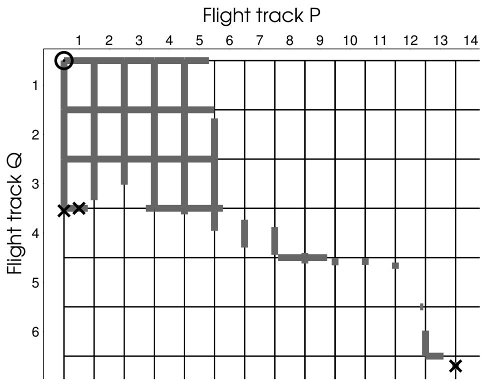
The problem to be solved is dual with respect to the computation of the Fréchet distance. For the Fréchet distance, the origin and destination nodes of the directed graph are fixed, and the adjacency properties of the directed graph are modified according to the candidate distance $\mathcal{F}_m$. For the overlapping index, the adjacency properties of the directed graph are fixed, since the distance $\alpha\mathcal{F}$ is fixed. What needs to be determined is (i) the regions with a monotone path, and (ii) the origin and destination for each region. Within each region (like the two shown in Fig. C.1(b)), multiple monotone paths can be identified. The policy that is adopted is, among the monotone paths identified, to select the one that maximizes the sum of the paths along the two tracks. If a region spans more that one unitary square vertically and/or horizontally, the great circle arcs associated with internal squares will be accounted for completely. Instead, for the initial/final points on the two tracks, a more careful analysis of the initial/final unitary square's free space is necessary. The procedure can be summarized as follows

1. Given $\mathcal{F}$ and α, compute the free space diagram using $\alpha\mathcal{F}$ as distance. Identify if at least a region with a monotone path exists.

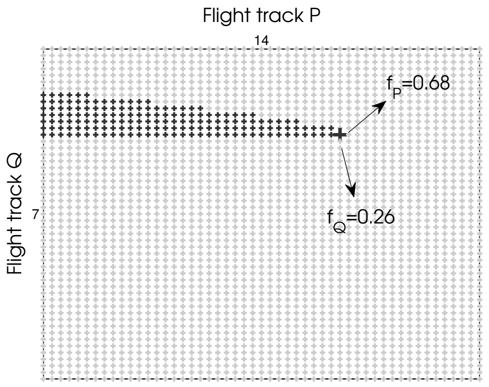
2. For each region, determine all the nodes of the original directed graph that are possible origins (no other nodes upstream) and destinations (no other nodes downstream). For each O-D node combination, determine if a path exists.

3. For each path, compute the overall length along the two ground tracks. To determine the initial/final points along the two tracks, focus on the unitary square upstream/downstream. Define a gridded discretization, compute all the pairs of points whose distance is less or equal to $\alpha\mathcal{F}$, and choose the pair that maximizes the sum of

the two great circle arcs (see Fig. C.2).



(a) Overlapping region, with origin and destination nodes highlighted with circles and crosses, respectively.



(b) Focus on the unitary square associated with the 14th and 7th great circle arcs of $\mathcal{P}$ and $\mathcal{Q}$, respectively.

Figure C.2: Focus on the first overlapping region identified in the free space shown in Fig. C.1.

The application of the procedure is analyzed for the upper left region shown in Fig. C.1(b)). Inside the region, a potential origin and three potential destinations are identified. They are highlighted with a circle and a cross, respectively, in Fig. C.2(a) The first two destination nodes are associated with the same final unitary square, since they are located on the top and left edge of the same unitary square in position $(4, 1)$. The third destination node is located on the left side of the unitary square in position $(7, 14)$, and it is the destination node that maximizes the length of the overlapping region. The free space of the unitary square is shown in Fig. C.2(b) in dark gray. The point that maximizes the sum of the two great circle arcs fractions is highlighted. The overlapping region ranges from the origin to the 26% of the 7th great circle arc of $\mathcal{Q}$, and from the origin to the 68% of the 14th great circle arc of $\mathcal{P}$.

If the process is repeated for the second block shown in Fig. C.1(b), the overlapping index suggests that the common region such that the local Fréchet distance is $\alpha\mathcal{F}$ covers more than 60% of both curves. This is evident if Fig. C.1(a) is analyzed. For most of the route, the matching between $\mathcal{P}$ and $\mathcal{Q}$ is very accurate. Track $\mathcal{P}$ is labeled as outlier because it performs a consistent path stretch before the landing procedure in DEN. $\mathcal{P}$ is not a different route with regard to $\mathcal{Q}$, but a tactical maneuver in proximity of the destination airport was enough to turn $\mathcal{P}$ into an outlier from a clustering perspective.

Our main takeaway can be summarized as follows. For low values of the overlapping index, the outlier is most likely a single track showing little similarity with respect to its nearest-neighbor. For higher values of the overlapping index (like the 60% characterizing the example presented), the outlier is most likely affected by a localized tactical maneuver. Although in portions of the track the overlapping with the nearest-neighbor is high, the presence of the tactical maneuver is enough to label the track as an outlier.