COMPUTATIONAL CONTROL OF NETWORKS OF DYNAMICAL SYSTEMS: APPLICATION TO THE NATIONAL AIRSPACE SYSTEM

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF
AERONAUTICS AND ASTRONAUTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Alexandre M. Bayen
December 2003

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Sanjay Lall
(Aeronautics and Astronautics)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

George Meyer
(NASA Ames)

Approved for the University Committee on Graduate Studies.

Abstract

The research presented in this dissertation is motivated by the need for efficient analysis, automation, and optimization tools for the *National Airspace System* (NAS) and more generally for large scale networks of dynamical systems.

A new modeling framework based on hybrid system theory is developed, which captures congestion propagation into the *Air Traffic Control* (ATC) system. This model is validated against *Enhanced Traffic Management System* (ETMS) data and used for analyzing low level actuation of the human Air Traffic Controller. This model enables us to quantify the capacity limit of the airspace in terms of geometry and traffic patterns, as well as the speed of propagation of congestion in the system. Once this setting is in place, maneuver assignment problems are posed as optimization programs, some of which can be reduced to *Mixed Integer Linear Programs* (MILPs). Problem specific algorithms are designed to show that certain MILPs can be solved exactly in polynomial time. These algorithms are shown to run faster than CPLEX (the leading commercial software to solve MILPs) when implemented on the same platforms. For other problems, approximation algorithms are designed, with guaranteed bounds on running time and performance. An architecture is proposed for the implementation of this method using a live ETMS data feed.

Flow control problems in the NAS are modeled using an Eulerian framework. A *Partial Differential Equation* (PDE) model of high altitude traffic is derived, using a modified *Lighthill-Whitham-Richards* (LWR) PDE. High altitude traffic is modeled as a network of LWR PDEs linked through their boundary conditions. The model is validated against ETMS data. A new adjoint-based method is developed for controlling ATC network flow management problems and successfully applied to realistic scenarios for the airspace between Chicago and the east coast. Accurate numerical analysis schemes are used and run very fast on this set of coupled one dimensional problems. The resulting simulations provide high level ATC control strategy (i.e. NAS-wide) in the form of flow patterns and routing policies to apply to streams of aircraft going through the system.

Finally, tactical control problems at the level of the dynamics of individual aircraft are studied in order to meet safety specifications. The problem of proving safety of conflict avoidance protocols is posed in the Hamilton-Jacobi framework, and linked to existing mathematical results. A proof of safety is derived for conflict avoidance. It is tested on real ATC scenarios for en route traffic and shows an excellent match with recorded Air Traffic Controller's actions.

Preface

This dissertation presents three years of research in the Hybrid Systems Laboratory within the Department of Aeronautics and Astronautics at Stanford University under the supervision of Professor Claire Tomlin. I arrived into the world of control theory September 2000 with a background in fluid mechanics. My first look at control theory was influenced by my knowledge of partial differential equations. Within the lab, Ian Mitchell, my elder brother in the group, introduced me to a project in which I started to work on the Hamilton-Jacobi equation, a partial differential equation which shared some similarities with hyperbolic conservation laws I was familiar with from the past. Our first publication enabled me to attend my first control conference and initiated a collaboration which led to several other articles. At the same time, I began to work on my own research described in this thesis.

Most of all, I would like to acknowledge the guidance of my academic advisor Professor Claire Tomlin. Claire has been a mentor in the etymological sense of the term. Our frequent meetings and fruitful discussions, thanks to her immense availability (25 hours a day and 8 days a week) have been beneficial in identifying relevant engineering problems and learning more about hybrid systems, her field of expertise. Claire suggested new interesting directions, brought me to conferences which sparked new ideas, provided insightful comments and organized collaborations with the right experts. The combination of her support, enthusiasm and energy have been an inspiration for me which I shall remember in the future for my own research group.

Unlike other systems, which have been studied for years, there are very few models of the National Airspace System available in the literature, and most of them focus on specific applications. In order to palliate this lack of material, Claire initiated collaborations with

the Oakland Air Route Traffic Control Center and with NASA Ames, which have lasted for three years. My work has greatly benefited from these ties.

At the Oakland Center, which I visited frequently during these three years, I had the opportunity to interact with the Air Traffic Controllers and the FAA operational personnel. The model of the National Airspace System, which I subsequently developed and derive in Chapter 3, partially builds on long hours spent next to the Air Traffic Controllers, while they were operating the system, as well as debriefings afterwards. This interaction enabled me to understand the underlying logic of their decisions and the behavior of the system under heavy workload.

At NASA Ames, I have had the privilege of being hosted one day a week by Dr. George Meyer within the Automation Concepts branch. Dr. Meyer's influence on my work has been significant. In addition, working with him has been a great pleasure. He is the person who initially suggested building this model, in order to understand the propagation of disturbances into the system. The legend says he devised this plan of research after he was forced to spend one night sleeping on the floor of the Chicago airport, because of a storm in South Carolina. At NASA, I was able to interact with numerous researchers from various backgrounds, bringing me up to speed on most of the issues relevant to Air Traffic Control. After a few months, the second floor of Building 210 at Ames felt like a second academic home to me. I would like to thank Dr. Gano Chatterji for being supportive and making numerous suggestions. Besides being my ETMS data feed, Dr. Shon Grabbe spent a lot of time with me in my initial investigation of this data. Many researchers have impacted my work, and provided useful material: Dr. Karl Bilimoria, Dr. Heinz Erzberger, Dr. Banavar Sridhar, Forrest West and others.

Most fields within engineering, control in particular, are multidisciplinary. This is reflected in my dissertation, which makes contributions to the areas of partial differential equations, optimal control, and combinatorial optimization, all applied to practical problems. At Stanford, I have benefited from outstanding teaching in these respective fields; Professors Juan Alonso, Stephen Boyd, Parviz Moin, Bill Reynolds and Claire Tomlin each helped in their own way to move me forward. I would like to thank Professors Antony Jameson, Ilan Kroo, Sanjay Lall, Doron Levy and Tai-Ping Liu for welcoming me into their offices and giving me the opportunity to have my questions answered. The proximity of UC Berkeley enabled me to interact with several researchers on the other side of the bay. I am grateful to Professors Laurent El Ghaoui, Craig Evans, Alexander Kurzhanski, Shankar Sastry and Pravin Varaiya for valuable discussions.

Thanks are also due to Professor Yinyu Ye, who opened the door of combinatorial optimization for me. He supervised most of the work on combinatorial optimization presented in this dissertation. I am extremely grateful to Professors Jean-Pierre Aubin and Patrick Saint-Pierre for their warm welcome at Paris Dauphine University, and for setting up an ongoing collaboration with Stanford and Berkeley, which not only bridged continents but also the fields of viability theory and partial differential equations. The research work done with my M.S. advisor, Professor Sanjiva Lele and his vision of hydrodynamic instabilities shaped my view of engineering. I wish to express my gratitude to Professor Eric Féron (MIT), Professor Shankar Sastry (UC Berkeley) and Professor George Springer (Stanford) for their support. I want to thank Professors Tom Bewley (UCSD), Jonathan How (MIT), John Lygeros (U. of Patras) and George Pappas (U. Penn) for useful scientific discussions.

Some of this work was done jointly with other students. Working with Ian on reachability analysis has been extremely beneficial for me, as it has enabled us to share our respective knowledge of Hamilton-Jacobi equations, viability theory, level set methods, fast marching methods, and numerical analysis. When I have some questions about unknown material on these subjects, I still call him in sunny Vancouver where he now lives. Chapter 8 uses a formalism developed jointly with him; the results obtained in Chapter 9 use his code. It has been a delight to work on approximation algorithms with Jiawei Zhang, the 1.52 hero of facility location. The work on the simulator is joint with Pascal Grieder, an ETHZ-trained William Tell of practical implementation, who coded the simulator in c++. The work on Eulerian networks and adjoint-based optimization is joint with Robin Raffard, whose energy contributed to the success of this project. I thank Eva Crück for our joint work on viability theory, Meeko Oishi for our collaborations on interface analysis and Shriram Santhanam for his contribution to the reachability project.

Tracing my roots back to my pre-Stanford life, I want to thank Professors at Ecole Polytechnique for their outstanding teaching and for triggering an interest in research leading to a Ph.D. Memories of Professors Patrick Huerre, Keith Moffatt, Bernard Larrouturou, Patrick Le Tallec, Jean Salençon and Laurent Jacquin are still vivid in my mind.

My many friends have been a wonderful community during my Stanford years, providing support, camaraderie and entertainment: Keith Amonlirdviman, Nicolas Antoine, Hamsa

Balakrishnan, Omar Besbes, Eric Blaise, Daniel Bodony, Francis Carr, Chris Clark, Nathalie Collin-Sisteron, Gonzalo Feijoo, Ronojoy Ghosh, Inseok Hwang, Gökhan Ìnalhan, Jung-Soon Jang, Kerstin Kling, Jerry Lynch, Antoine Manens, Ted Manning, Matt McMullen, Ian Mitchell, Meeko Oishi, Robin Raffard, Tom Schouwenaars, Faye Steiner, Dušan Stipanović and Rodney Teo.

While at Stanford I have been supported by a graduate fellowship from the Délégation Générale pour l'Armement (France). The research projects in which I was involved were supported by NASA under Grant NCC 2-5422, by ONR under MURI contract N00014-02-1-0720 and by DARPA under the Software Enabled Control Program (AFRL contract F33615-99-C-3014). I want to thank Sally Gressens, Sherann Ellsworth and Dana Parga for helping me sort out the administrative surprises which prevent a Stanford graduate student's life from being monotonous.

To end on a more personal note, my girlfriend Zoë Abrams has given me ongoing love and support. Zoë also has been a great interlocutor for combinatorial optimization, since she is currently doing her Ph.D. in this field. It is no wonder that many of our discussions often end up on Air Traffic Control and approximation algorithms, our respective areas of interest.

My studies at Stanford have been a motivation for numerous trips to the Bay Area for my family. I want to thank my sister Eléonore and brother Térence for being so close to me throughout these years. I thank my parents who have devoted their energy to my education, of which the formal part ends with this Ph.D. Their constant encouragements have been a great support during my graduate student life at Stanford.

Contents

List of Tables

List of Figures

Part I

Algorithms for network control of Air Traffic Management systems

Chapter 1

Introduction

The recent growth in embedded control system applications has been triggered by the advent of numerous factors and technologies: increase of computer power, advances in monitoring and communications, progress in miniaturization. While many kinds of automated systems have greatly benefited from these technologies (for example, cars, avionic systems, single robots, unmanned air vehicles), a similar technology for networks of these systems is still lacking. Sensor networks, groups of mobile communication devices, swarms of robots, and the *Air Traffic Control* (ATC) system are just a few examples of networks of automated systems for which a clear paradigm does not yet exist.

The difficulty of controlling ATC systems is manifold: it is by nature distributed; it demands high levels of efficiency, performance, reliability, and safety; a portion of the control needs to be centralized (planning), the remainder needs to be decentralized (local flow control); and it is extremely sensitive to disturbances such as congestion and weather.

Some of the systems mentioned above — the ATC system in particular — even suffer from the lack of an appropriate model. The difficulty of modeling the ATC system arises from the number of different elements interacting in the system. Humans-in-the-loop are a major difficulty in generating a mathematical model with predictive capability. ATC is a multiple agent system with conflicting goals: Air Traffic Controllers are supposed to ensure fairness among selfish users (airlines), who try to use the system to their advantage. Not all scales of the system are necessarily relevant for all models (for example flow management does not require a precise description of aircraft longitudinal dynamics). Also, the amount of available information to be processed is often greater than treatment allows.

1.1 The need for application specific air traffic flow models

As will be seen in the next chapter, the enormous size of the *National Airspace System* (NAS) in the US forces any model or control strategy to ignore some of its components, which are irrelevant for the application of interest. A few numbers illustrate this fact. More than 5000 aircraft (military, commercial and general aviation) can be present in the US airspace at the same time. A single Air Traffic Controller can communicate with more than 15 aircraft at a given time, and there are about 17000 Air Traffic Controllers in the NAS. The update rate of the monitoring system is 12 seconds for high altitude traffic, sometimes up to 3 seconds near airports. This data is displayed to Controllers and then collected and assembled into a single data file by a central ATC facility. The volume of this data file is thus extremely large. Furthermore, some of the data is not useful for every Air Traffic Controller of the NAS: for example, Air Traffic Controllers in charge of planning traffic at a high level do not need 3 second updates. Reciprocally, an Air Traffic Controller in charge of arrivals into San Francisco needs this update rate, but does not need to know about airspace congestion in Philadelphia. To give an idea of the size of the data, when this work was started, this data was processed and redistributed at the rate of one update every 3 minutes, limited by bandwidth (vs. every 3 or 12 seconds). Even with the slow update rate of 3 minutes, the accumulated data was already on the order of 700 MB a day.

Prevention of airborne collisions, security and robustness of the control with respect to disturbances, congestion avoidance in busy areas, optimization of airline profit, avoidance maneuvers for parallel landing, free flight — each of these is a problem of interest to ATC or airlines. In order to treat these problems, different kinds of models are needed, appropriate for an efficient mathematical treatment of the issues involved.

When small numbers of aircraft are involved in close range, and the problem to be solved is collision avoidance, a fairly precise knowledge of each aircraft's possible motion is needed. When a larger number of aircraft are considered, such as for flow scheduling near an airport, it might be sufficient to know the set of speeds and maneuvers achievable by each aircraft in order to perform the appropriate time adjustments. Finally, for problems involving very large numbers of aircraft, such as in trying to balance the NAS-wide overload of the system, a relevant approach might be to discard the individual identity and local behavior of the aircraft and focus on their density (i.e. number per surface unit). These three levels of detail will be addressed in this work, with relevant illustrative examples.

1.2 Computational control of networks of dynamical systems

The three main goals of this work are thus: (*i*) to come up with efficient and reliable control paradigms for regulating air traffic flow in the vicinity of large airports; (*ii*) to create a control framework for air traffic mass balancing in the NAS; (*iii*) to provide a mathematical framework for proving safety of collision avoidance protocols. The specifications on the control are driven by the necessity of online implementation (therefore high speed of computation is required), and the need for a guaranteed performance (for example, the amount of separation between aircraft in space or time *should be* greater than a fixed threshold). We use the word *computational* to refer to both speed and performance guarantees, which we will address in developing our algorithms.

When we need to describe aircraft individually, we use the framework of dynamical systems to model their behavior, because we are interested in their trajectories, which are well represented by solutions of *ordinary differential equations* (ODEs). The dynamical systems are networked to the ground through a communication link. The network connection between the respective dynamical systems is mathematically not formalized as precisely as it would be in a wireless communication or sensor network for example; in fact one of the difficulties is precisely to model this linkage, especially when the human is part of it.

Most of the descriptions of air traffic flow which model aircraft as dynamical systems are by essence *Lagrangian.* This means that the description of the aircraft is made through its trajectory. This is a natural model for numerous applications of interest: collision avoidance, scheduling, multiple aircraft path planning. Within the Lagrangian approach, we will use *hybrid systems* as a modeling framework. Hybrid systems are a very powerful modeling tool for describing systems exhibiting distinct continuous and discrete behaviors. This is useful for ATC: Air Traffic Controllers tend to think about aircraft as being in a mode (climb, descent, cruise, turn, in a holding pattern, etc). These modes constitute a set of discrete states, in which the continuous evolution of the system can be described by a dynamical system. The discrete evolution of the system corresponds to mode switches (for example from cruise to turn to cruise). The framework of hybrid systems is thus very suitable for ATC, which is a very procedural environment, in which it is easy to identify these discrete modes and switches within the structure of the existing control; in fact the

Air Traffic Controllers' phraseology includes these modes and switches, as will appear in the later chapters.

However, modeling a physical system as a hybrid system or a network of hybrid systems does not provide a mathematical solution to the problem of controlling it. Within the field of hybrid systems, several classes of problems have been identified, and there are only few of them for which we know theoretically how to design a controller. There are even fewer for which computationally efficient algorithms exist. Therefore, unlike in optimization, there do not exist standard toolboxes or software that one can use, as we commonly do when we identify a problem as a linear program, a quadratic program or a convex program. This fact advocates for several possible research approaches to solve hybrid systems controller synthesis problems. One approach consists of deriving general methods for classes of systems; the benefit is that the method applies to any system in the class; the drawback is that the applicability of the method is limited to specific systems. Another approach consists of deriving problem specific algorithms; the advantage is that these algorithms are usually much faster than general algorithms, and apply to "harder" scenarios; the drawback is that they are difficult to generalize to different problems. The work presented here uses both approaches. We will develop problem specific algorithms based on dynamic programming, LP relaxation and matching algorithms to solve combinatorial optimization problems specific to ATC. We will also develop a general methodology to compute reachable sets of low dimension systems and apply it to the specific problems of interest.

As appears in the title of this dissertation, the main application of this work is ATC. We have not found prior to this work good mathematical models which enable a systematic description of the airspace, the aircraft motion and Air Traffic Controller action in terms of ATC procedures. The first contribution of this work is thus a mathematical model of the airspace, which was first described in [18, 19, 17, 25]. This model enables the analysis of airspace congestion propagation, and is validated against air traffic data. The control part of the problem consists of using this model to derive optimal ATC policies. The word *optimal* in the ATC context can have multiple meanings: maximal throughput into an airport, minimal sum of all flight times, minimum legal separation between the aircraft upon landing, etc. The second contribution of this work is thus to pose specific ATC scenarios as hybrid system controller synthesis problems, and reduce them to known optimization programs [27]. For the reasons mentioned above, problem specific combinatorial optimization algorithms were developed to solve these programs, with appropriate guarantees on performance and speed [29, 28].

As mentioned above, if one is interested in controlling large scale effects of the system (for example balancing aircraft density in the NAS), the identity of the aircraft is not crucial: only their concentration in certain regions of airspace is useful. Inspired by highway congestion models, a new *Eulerian* airspace model was developed [22], specifically tailored to balance densities in the NAS. Eulerian means that instead of being represented by their trajectories, aircraft are described by their concentration (density) in given control volumes. The resulting *partial differential equation* (PDE) was used to build a network model, for which a new adjoint-based constrained optimization method was build to control it [21]. The novelty of this method is that it applies to PDE governed networks and incorporates nonlinear constraints on the state and the control. It is quite general, was successfully extended to highway congestion control [23], and applies to several other PDE driven network problems.

Finally, the second part of this dissertation tackles vehicle-vehicle interaction problems. It is usually assumed that once high level optimal policies are determined, there will be an appropriate control action to separate the aircraft, i.e. to prevent them from coming too close to each other. In fact, this task is currently performed manually by Air Traffic Controllers. Several automation methodologies have been proposed to automate this task. A key requirement of this automation is proof of safety. A mathematical proof was developed in [116] to address this. An intermediate result in coming up with the proof [116] was to construct an analytical solution [26] of the problem, in order to understand difficulties or errors published in previous literature on the subject. Finally, efficient numerical methods presented in [116] were used with real data for conflict detection in [24]. This work is to our best knowledge the first to apply differential game theory to real Air Traffic Control problems.

The general framework developed in [116, 24] was also used for other applications, which are not presented in this dissertation. Extensions of this framework to hybrid systems were presented in [150], applications to flap deflection for landing aircraft was published in [117, 20, 148], applications to takeoff go-around were investigated in [126, 125], applications to en route flight will appear in [8]. Finally, an alternate numerical technique for fast computations of safety sets appeared in [16], and is outlined at the end of this dissertation.

1.3 Organization of this work

The network models and corresponding control methodologies are summarized in Part I. Part II contains the vehicle-vehicle interaction algorithms and their applications.

After a short presentation of the organization of the NAS in Chapter 2, Chapter 3 derives a *Lagrangian* model of the NAS, based on trajectories of the aircraft, analyzes delay propagation in the system, and provides a model of the human sector Air Traffic Controller. Chapter 4 casts the problem of optimal maneuver assignment in converging traffic into a hybrid system controller synthesis format, and subsequently reduces it to a mixed integer linear program. This program is solved in Chapter 5 with combinatorial optimization algorithms which we derived. An Eulerian model of the NAS is derived in Chapter 6, in order to provide an appropriate mathematical framework for balancing density in the NAS. A control algorithm is developed in Chapter 7, which relies on adjoint-based constrained control for PDEs.

The mathematical framework for proving safety of conflict avoidance protocols is developed in Chapter 8, and applied to real ATC scenarios in Chapter 9.

Additional results are summarized in the different Appendices. Appendix A outlines a fast algorithm for overapproximating safety zones in ATC, and applies it to converging traffic in the Oakland Center. Appendix B provides additional proofs required for the combinatorial optimization algorithms developed in Chapter 5.

Chapter 2

Architecture for Automated Air Traffic Management

This chapter presents a short overview of the current organization of Air Traffic Management (ATM) in the US. In particular, it presents the structure of the airspace, which will be used in subsequent chapters for analysis, modeling and control. This introductory chapter is based on the book [123] by Nolan, and was published as a part of a case study book chapter on ATM [25]. Section 2.1 briefly summarizes the history of ATM in the US, and explains how the current system was created. Section 2.2 explains how the airspace is physically organized and how the different portions are managed. Section 2.3 summarizes the current tools and technologies used for monitoring the system. Finally, Section 2.4 gives an overview of the procedures which will be used to model the system.

2.1 A short history of Air Traffic Control

Air Traffic Control in the United States began in the late 1920s, pioneered by airport employees using red and green flags, and lights to signal their instructions to pilots. The Air Commerce Act of May 20, 1926, was the first step of the Federal government towards regulation of civil aviation. This legislation was pushed by the leaders of the aviation industry, who were convinced that the airplane could not reach its full commercial potential without Federal action to define, improve and maintain safety standards. The Air Commerce Act defined several tasks, including issuing and enforcing air traffic rules, licensing pilots, certifying aircraft, establishing airways, and operating and maintaining aids to air navigation. The first city to have a radio-equipped control tower was Cleveland (1930). The first three centers for *Air Traffic Control* (ATC) were established by an airline consortium, encouraged by the Federal government, between 1935 and 1936. Maps, blackboards, and mental calculations were the first tools used by early Air Traffic Controllers to ensure the safe separation of aircraft traveling between cities along designated routes.

In 1938, the Civil Aeronautics Act transferred the Federal civil aviation responsibilities from the Department of Commerce to a new independent agency, the Civil Aeronautics Authority. The legislation also expanded the government's role by giving the Authority the power to regulate airline fares and to determine the routes that airlines would serve. The Authority was split in 1940, giving birth to the *Civil Aeronautics Administration* (CAA) and the *Civil Aeronautics Board* (CAB) placed under the Department of Commerce. The CAA was responsible for ATC, airman and aircraft certification, safety enforcement, and airway development. The CAB's task was safety rulemaking, accident investigation, and economic regulation of the airlines.

The increasing airspace congestion triggered by the growing traffic in the 1940s, the introduction of jet airliners and a series of midair collisions motivated passage of the Federal Aviation Act of 1958, which generated a new agency: Federal Aviation Agency. Even though a special committee had already recommended the use of radar in 1947, it was not until the late 1950s that a civilian radar system was installed by the CAA. The Federal Aviation Agency was given sole responsibility to develop and maintain a common civil-military system of air navigation and ATC. The Act also transferred safety rulemaking from the CAB to the Federal Aviation Agency. On April 1, 1967, the Federal Aviation Agency became one of several organizations within the *Department of Transportation* (DOT) and became the *Federal Aviation Administration* (FAA).

In the mid-1970s, the FAA achieved a semi-automated ATC system based on a combination of radar and computer technology. By automating certain functionalities of ATC, the system allowed Air Traffic Controllers to concentrate more efficiently on the vital task of aircraft separation, which is still not automated today. The Air Traffic Controller graphical display encompassed technology able to visualize numerous information about aircraft (identity, altitude, and ground speed of aircraft carrying radar beacons), while controlling the airspace. Despite its effectiveness, this system was not able to keep up with the growth of traffic and increasing congestion. The NAS Plan, created in January 1982, by FAA aimed at finding solutions to the congestion problem, defined more advanced systems for en route and Terminal ATC, modernized flight service stations, and improved in ground-to-air surveillance and communication. Several other levels of automation were introduced until the events of September 11, 2001. Shortly after, Congress created the *Transportation Security Administration* (TSA), whose principal responsibility is civil aviation security.

Despite the negative impact of September 11 and the concurrent economic recession, which made passenger demand fall initially by more than 20%, airspace congestion is a continued problem. Recent studies [143] have shown that the overall impact of September 11 on air traffic congestion was just a two year delay in previous estimates. In 40 years, the delays have increased by 50% in the United States. From 1995 to 1999, the average delay (over all US flights) grew from 42 minutes to 50 minutes. Flight cancellations increased by 68% between 1995 and 1999. Annual traffic growth is still 2.3% and airlines have increased their flight times on 80% of all busy routes, up to 27 minutes. This means that the published time flight (i.e. available on the passengers' tickets) was increased by 27 minutes, which include forecasted delays as well as a "safety buffer" (it is better for an airline to declare a flight early if this forecast was too pessimistic). Such a list of alarming numbers could be extended almost endlessly. It vehemently speaks for optimization of the current ATC system, in order to satisfy the ever-increasing amount of traffic. Automation of certain ATC tasks, which this work is focused on, is aimed at contributing to the optimization of performance of the system, while maintaining safety guarantees.

2.2 Overview of the current airspace structure

The *National Airspace System* (NAS) is divided into different *classes*, which correspond to regions under different regulations and use. An exhaustive classification of airspace is

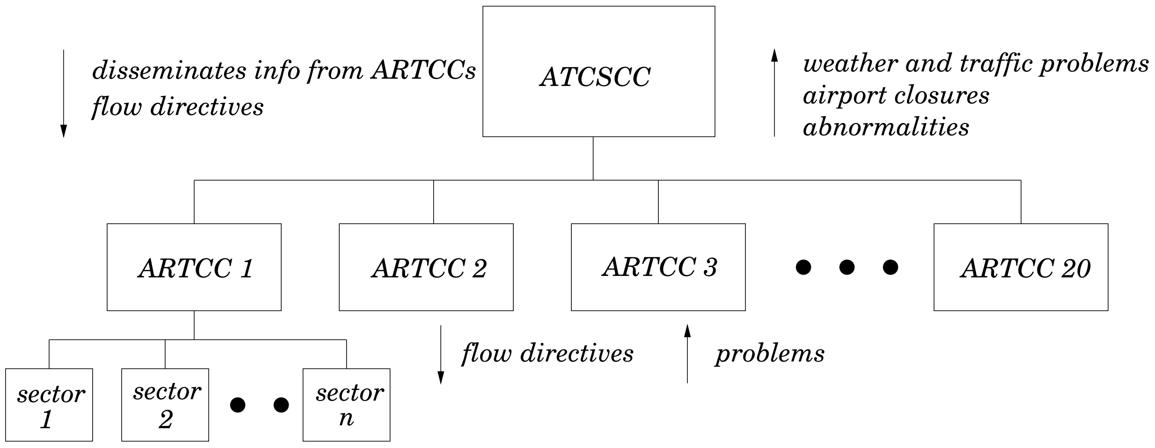


Figure 2.1: Control hierarchy in the current structure of National Airspace System.

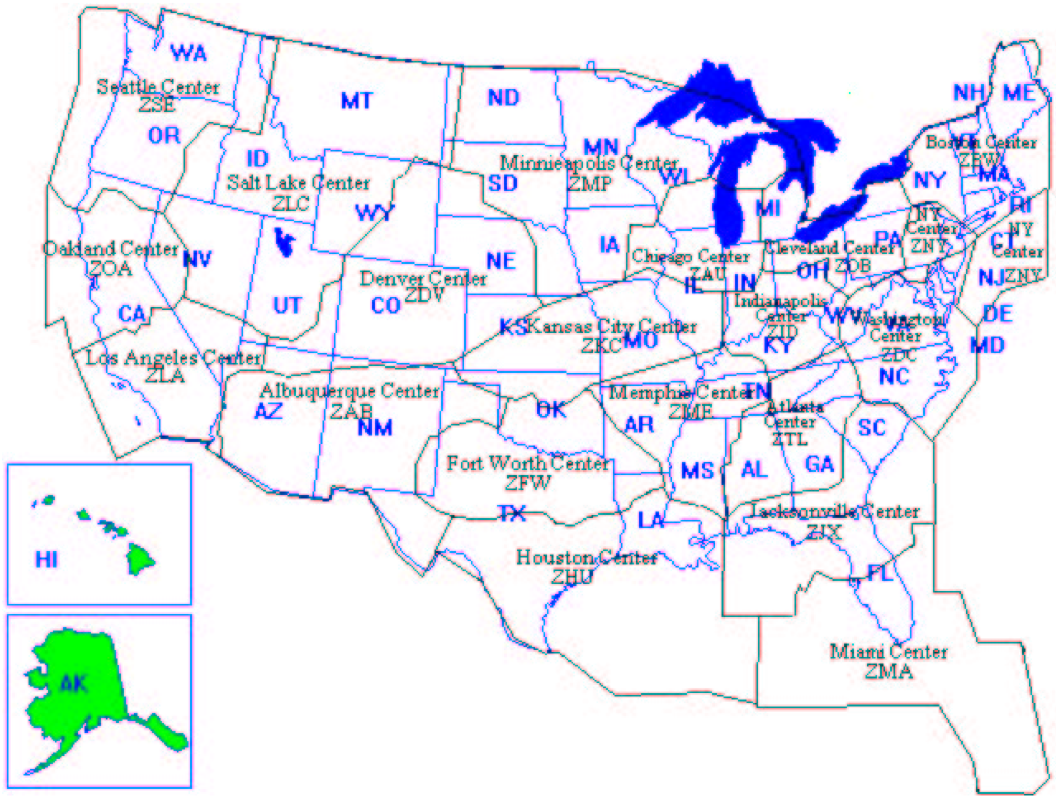


Figure 2.2: Map of the 22 ARTCCs in the U.S. (map courtesy of `http://www.seaartcc.org`).

available in [123], and is only briefly summarized here.

Class A airspace exists from 18,000 to 60,000 feet. All operations in this airspace must be under *instrument flight rules* (IFR) (pilots must be rated to fly according to the rules governing the procedures for conducting instrument flight) and are subject to air traffic control clearances and instructions. *Class B airspace* surrounds “busy” airports in the US. Each Class B area is individually tailored and consists of a ground surface area and two or more surrounding layers (see Figure 2.3) (most Class B airspace would look like an inverted wedding cake if viewed in profile). Again, pilots must receive an ATC clearance to enter class B airspace. *Class C airspace* generally surrounds “smaller” airports with an operating control tower, a radar approach control facility, and a certain number of IFR operations. The area encompassed by this airspace is delimited by two circles with the inner circle extending 5 nautical miles from the airport starting at the surface and extending up to 4000 feet above airport elevation. The outer circle extends to 10 nautical miles from the airport and consists of a shelf from 1200 feet to 4000 feet above airport elevation. The rest of civilian airspace is divided in further categories (*Classes D, E, G*), not relevant for the description in this chapter. Airspace also includes *special use airspace*, which encompasses prohibited areas, restricted areas, warning areas and military operations areas, which will not be detailed here.

The NAS is a large scale, layered, dynamic system: its control authority is currently organized hierarchically with a single *Air Traffic Control System Command Center* (ATCSCC), in Herndon VA, supervising the overall traffic flow. This is supported by 22 (20 in the continental US or CONUS) *Air Route Traffic Control Centers* (ARTCCs, or simply, Centers) organized by geographical region up to 60,000 feet [123, 94]. Each Center is sub-divided into about 20 sectors, with at least one Air Traffic Controller responsible for each sector. Each sector Air Traffic Controller may talk to 25-30 aircraft at a given time (the maximum allowed number of aircraft per sector depends on the sector itself). The Air Traffic Controller is in charge of preventing *losses of separation* (LOS) between aircraft, keeping them separated by more than 5 nautical miles horizontally, and 2000 feet vertically. In general, the Air Traffic Controller has access to the aircraft’s flight plan and may revise the altitude and provide temporary heading assignments, amend the route, speed, or profile, in order to attempt to optimize the flow and to keep aircraft *separated*, as well as to provide weather reports and winds. An illustration of the current control structure is presented in Figure 2.1.

There are about $17,000$ Air Traffic Controllers in the NAS infrastructure, each controlling a zone with rough diameter from 20 to 200 miles [123]. There are about $19,000$ landing facilities, with about 400 of these major airports with ATC towers. The acceptance rate of each airport is usually 1 aircraft/minute per runway in normal operations (if the runway is used for both take off and landing); this capacity is doubled if the runway is used for landing only.

Within the Center airspace, the regions away from the airports, in which the aircraft usually fly at high altitude belong to the *en route airspace*, which is controlled by the ARTCC. Air traffic is delivered to the en route airspace from the *Terminal Radar Approach Control* (TRACON) facilities, in charge of the airspace around the airports (for departure and arrival). The US contains more than 150 TRACON facilities, which generally control this airspace up to 15,000 feet. TRACONs may serve more than one airport: for the San Francisco Bay Area TRACON includes the San Francisco, Oakland, and San Jose airports along with smaller airports at Moffett Field, San Carlos, and Fremont. The regions of airspace directly around an airport as well as the runways and ground operations at the airport are controlled by the *Air Traffic Control Towers*.

2.3 Navigation and surveillance

ATC surveillance is performed through the use of radar. A primary radar processes the reflection of the waves sent to the aircraft and reflected by their bodies, which provide the position of the corresponding aircraft. A secondary radar triggers a transmitter in the aircraft to automatically emit an identification signal. The range of radars depend on their use. For the en route airspace, the *Air Route Surveillance Radar* (ARSR) is used; in the TRACON, the *Automated Radar Terminal System* (ARTS) is used, which has a shorter range. The accuracy as well as their update rates depend on their use. The update frequency can vary from 3 to 12 seconds. Each ATC facility is equipped with their own computer system to process the information collected by the radar systems, and incorporate additional

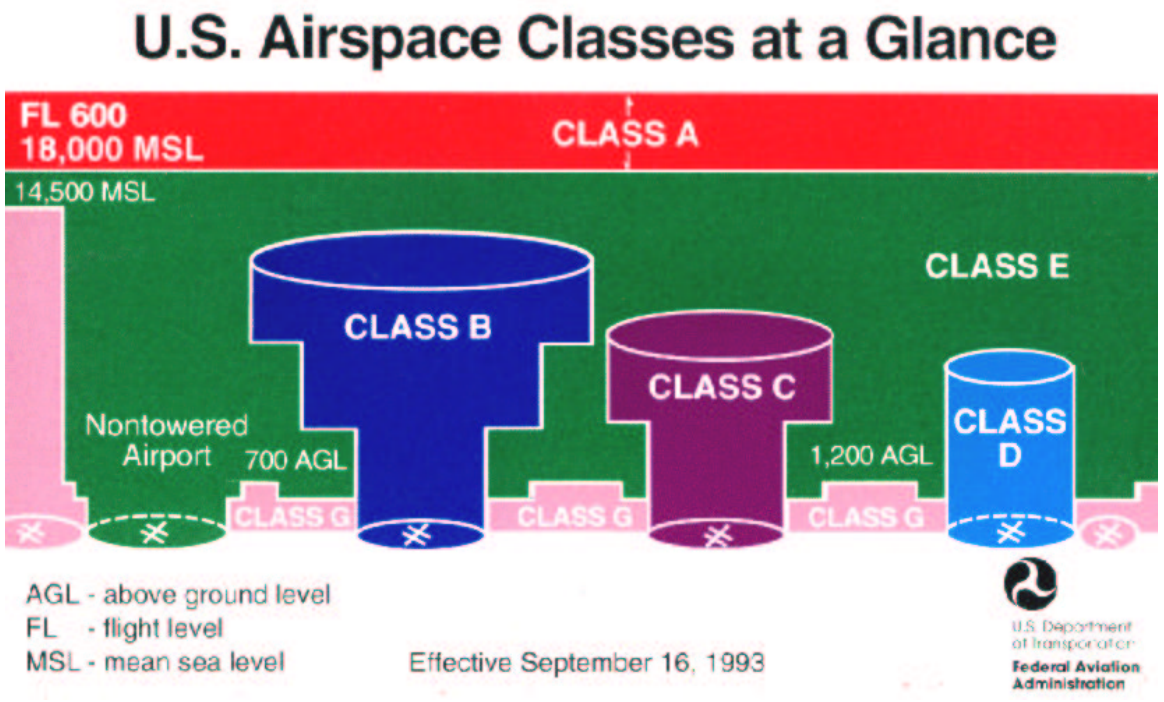


Figure 2.3: Airspace Classes (courtesy of U.S. Department of Transportation).

information from the other ARTCCs or the ATCSCC. Human Air Traffic Controllers use a visual interface showing the traffic situation in their sector (or area of control). This interface includes some forecast functionality (for example anticipated trajectories of aircraft, filed flight plans), as well as graphical tools to estimate or forecast aircraft separation (for example speed vector displays or circles centered around the aircraft). The main task of sector Air Traffic Controllers is to maintain the FAA standards for aircraft separation, which are 5 nautical miles horizontal separation, 1000 feet (2000 feet above 29,000 feet) vertical separation in the Center airspace, and 3 nautical miles horizontal separation, 1000 feet vertical separation in the TRACON.

ATC currently directs air traffic along predefined routes called *victor airways* (low altitude $< 18,000$ feet) and jetways (high altitude), which are "freeways in the sky". Often, the aircraft might request to fly directly from *waypoint* to *waypoint*, which are a system of *beacons* (*non-directional beacons* (NDBs), *very high frequency omni-range receivers* (VORs), and *distance measuring equipment* (DME)). These beacons help the pilots or autopilots navigate through the system and are used by the *inertial navigation systems* (INS) on board each aircraft.

Other systems are also available for navigation and surveillance, and are currently in the process of being certified. They are therefore used in conjunction with other system, in a redundant manner.

The *Global Positioning System* (GPS) is a radio navigation system that allows land, sea, and airborne users to determine their exact location, velocity, and time 24 hours a day, in all weather conditions, anywhere in the world. Its *Wide Area* and *Local Area Augmentation Systems* (WAAS and LAAS) is a form of *differential GPS* (DGPS) giving enhanced position accuracy developed primarily for aeronautical applications but usable by other users. One advantage of the GPS and derivative technologies is its accuracy: unlike INS systems, which are known to decrease over time due to sensor drift rates, the accuracy is uniform from aircraft to aircraft [77].

The communication protocol *Automatic Dependent Surveillance* (ADS) enable aircraft to transmit over digital satellite communication their GPS position information, velocity, as well as information about their intended trajectory, to the ground ATC. ADS-B (for broadcast) is a protocol for broadcasting this information to neighboring aircraft. One advantage of this system is to provide very accurate information to other aircraft, including trajectory prediction, without the use of the radar system. This could provide very helpful in two situations: oceanic flight, as well as in the vicinity of large airports. To the current day, very few airlines have equipped their aircraft with ADS-B, as the system becomes useful only if all users are equipped with it. A good example of its use is by *United Parcel Service* (UPS), which uses ADS-B for night landings in its hub airport. In this case, a large portion of the fleet has to land in a short period of time at the same location, which makes this tool useful. A potential use of ADS-B in the same context would be for reducing aircraft separation.

2.4 Communication and procedures

All IFR pilots must file a *flight plan* at least 30 minutes before pushing back from the gate. The pilot reviews the weather along the intended route, maps the route and files the plan. The flight plan includes: flight number (which includes the airline identification), the aircraft type, the intended airspeed and cruising altitude, the route of flight (departure airport, Centers that will be crossed, and destination airport). It also includes additional information, called *waypoints, navaids*, or *fixes*, which will be used by the aircraft to navigate through sectors of airspace. The flight plan also contains the *arrival*, which is a set of closely spaced waypoints, navaids or fixes leading to an airport. An example of arrivals into the Oakland airport (in California) is shown in Figure 2.4, with corresponding infrastructure. The pilot transmits the desired flight plan information to ATC, where a Air Traffic Controller called a *flight data person* reviews the weather and flight plan information and enters the flight plan into the FAA main, or "host" computer. The computer generates a set of flight progress *strips* that are sent electronically from sector Air Traffic Controller to sector Air Traffic Controller across the flight plan; these strips, and flight plans, may be updated by each Air Traffic Controller throughout the flight. The flight progress strip contains all of the necessary data for tracking the aircraft.

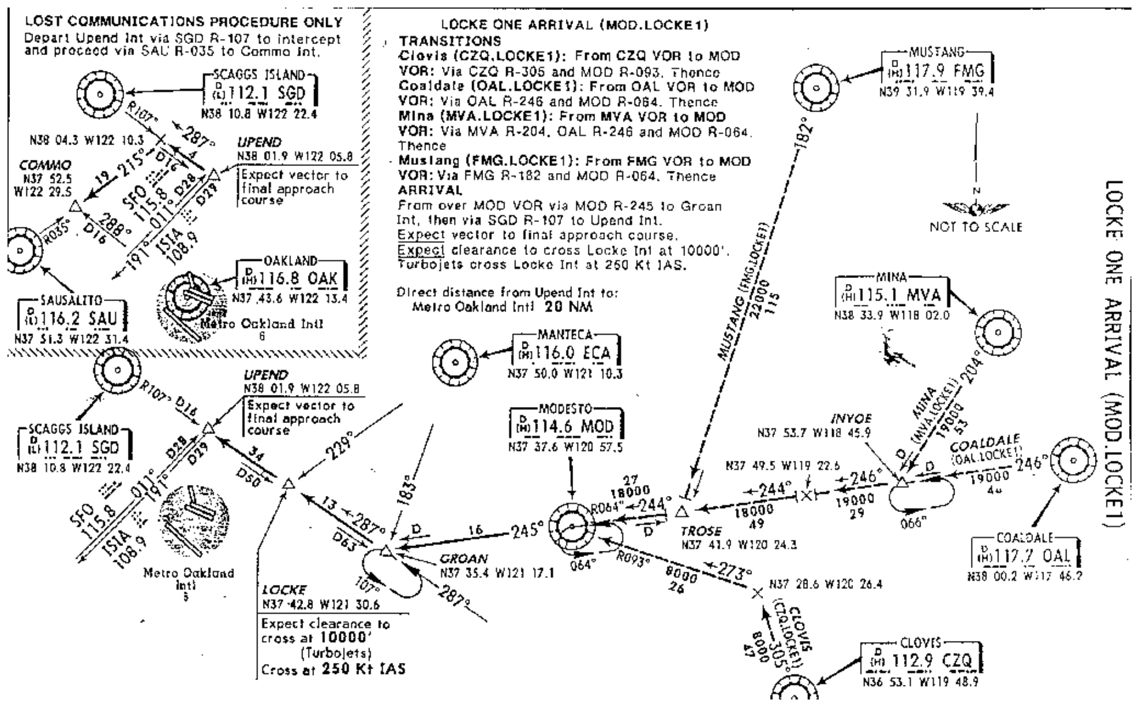

LOST COMMUNICATIONS PROCEDURE ONLY
Depart Upend Int via SGD R-107 to intercept and proceed via SAU R-035 to Commo Int.
LOCKE ONE ARRIVAL (MOD.LOCKE1)
TRANSITIONS
Clovis (CZQ.LOCKE1): From CZQ VOR to MOD VOR: Via CZQ R-305 and MOD R-093. Thence
Coaldale (OAL.LOCKE1): From OAL VOR to MOD VOR: Via OAL R-246 and MOD R-064. Thence
Mina (MVA.LOCKE1): From MVA VOR to MOD VOR: Via MVA R-204, OAL R-246 and MOD R-064. Thence
Mustang (FMG.LOCKE1): From FMG VOR to MOD VOR: Via FMG R-182 and MOD R-064. Thence
ARRIVAL
From over MOD VOR via MOD R-245 to Groan Int, then via SGD R-107 to Upend Int.
Expect vector to final approach course.
Expect clearance to cross Locke Int at 10000'.
Turbojets cross Locke Int at 250 Kt IAS.
Direct distance from Upend Int to: Metro Oakland Intl 20 NM
SCAGGS ISLAND 112.1 SGD N38 10.8 W122 22.4
COMMO N37 52.5 W122 29.5
UPEND N38 01.9 W122 05.8
Expect vector to final approach course
OAKLAND 116.8 OAK N37 43.6 W122 13.4
SAUSALITO 116.2 SAU N37 51.3 W122 31.4
Metro Oakland Intl
MANTECA 116.0 ECA N37 50.0 W121 10.3
MODESTO 114.6 MOD N37 37.6 W120 57.5
MUSTANG 117.9 FMG N39 31.9 W119 39.4
NOT TO SCALE
MINA 115.1 MVA N38 33.9 W118 02.0
INYOE N37 53.7 W118 45.9
COALDALE (OAL.LOCKE1)
COALDALE 117.7 OAL N38 00.2 W117 46.2
TROSE N37 41.9 W120 24.3
GROAN N37 35.4 W121 17.1
LOCKE N37 42.8 W121 30.6
Expect clearance to cross at 10000' (Turbojets) Cross at 250 Kt IAS
CLOVIS 112.9 CZQ N36 53.1 W119 48.9
LOCKE ONE ARRIVAL (MOD.LOCKE1)

Figure 2.4: Example of arrivals into the Oakland (OAK) airport : LOCKE 1 (MOD.LOCKE1). Aircraft enter this airspace through the waypoints: MUSTANG, MINA, COALDALE, CLOVIS, whose acronyms are FMG, MVA, OAL, CZQ. Note the tracks for holding patterns (shown as loops at various merge points). Source: [93].

After the pilot has filed the flight plan, ATC may modify the flight plan according to constraints of the NAS and other aircraft (information which is available to each Air Traffic Controller from conversations with the ARTCC and ATCSCC Air Traffic Controllers), and issues a clearance to the pilot. After take-off, the control of the aircraft is passed through the Tower, TRACON, and possibly several Center facilities until the destination TRACON is reached.

Each sector Air Traffic Controller may talk to 25-30 aircraft at a given time [123]. When an aircraft crosses the boundary from one sector to the next, there is a "hand-off" in which the communication is transferred from one Air Traffic Controller to the next. Potential conflicts must be resolved before hand-off occurs. The Air Traffic Controller directs the aircraft according to a set of simple *control directives*, voiced sequentially. One of the most important, and time consuming, Air Traffic Controller tasks is to prevent LOS between aircraft.

Radio communications are a critical link in the ATC system. The most important aspect in pilot / Air Traffic Controller communications is mutual understanding of the command and response. Therefore, pilots acknowledge each radio communication with ATC by using the appropriate aircraft call sign; contacts are kept as brief as possible. For example, a contact procedure is codified as follows: name of facility being called, full aircraft identification as filed in the flight plan, and eventually, the request or type of message to follow. Each procedure is codified in a similar way. Each sector is handled by one key Air Traffic Controller; each Air Traffic Controller has his own radio frequency over which the communication with pilots in his sector takes place. As a flight progresses from one sector to another, the pilot is requested to change to the appropriate frequency. The *International Civil Aviation Organization* (ICAO) phonetic alphabet is used by FAA personnel when communications conditions are such that the information cannot be readily received without their use. The grammar and phraseology used in the current system is available in [123] and has been the focus of recent studies [83]. In general, the commands given to the aircraft by ATC are very precise and can be easily categorized in a discrete set of functions, parameterized by real numbers indicating speed, heading, or other flight variables. This very procedural command environment facilitates the task of modeling human ATC action, communication and aircraft behavior, as will be shown in this dissertation. A sample command given by a human ATC to an aircraft might be: *"achieve flight level 290, turn to a heading of 130, reduce airspeed to 120 knots . . . "*. In addition, the procedures differ from TRACON to Center control: in the TRACON, the Air Traffic Controller is responsible for taking the aircraft from *climb* → *en route* (the control actions must meet impromptu flow restrictions, hand-off to en route control, clear to join filed route); in the Center, the Air Traffic Controller may revise the altitude and provide temporary heading assignments, amend the route, speed, profile, and provide weather reports and winds.

In this way, the control is *distributed*, since it is applied locally in each sector. There is loose *coupling* within the ATC hierarchy: the ATCSCC Air Traffic Controllers talk to the ARTCC Air Traffic Controllers several times a day to provide updates and receive feedback about the flow control in each Center/sector. In the case of bad weather, airport closures, or other large disturbances, this feedback tightens, and the directives and updates among the levels of the hierarchy become more frequent.

Chapter 3

A Lagrangian model of sector-based air traffic flow

This chapter derives a *Lagrangian* model of air traffic flow. Lagrangian refers to the framework used: we describe the evolution of traffic in terms of the trajectories of each individual aircraft. This approach contrasts with the Eulerian approach presented in Chapter 6 which is control volume-based (i.e. the observer records variations of traffic in a given portion of airspace). We will describe the trajectories of aircraft, and decompose these trajectories into successive maneuvers, which we model as modes of a hybrid system. We will explain how the different modes in the model relate to the commands which the human ATC gives to the aircraft.

The goal of building this model is threefold. (*i*) We will use it to model congestion propagation and derive metrics useful for ATC. (*ii*) We will use it to derive a model of the human sector Air Traffic Controller. (*iii*) In the next two chapters, we will use it as a mathematical framework for designing appropriate control policies to achieve certain ATC tasks.

As will be explained, this model was derived from observations made at the Oakland Center. Despite the fact that the model presented here matches our observations, a validation against real data is necessary to assess its accuracy and limitations. Such a validation will also be presented in this chapter.

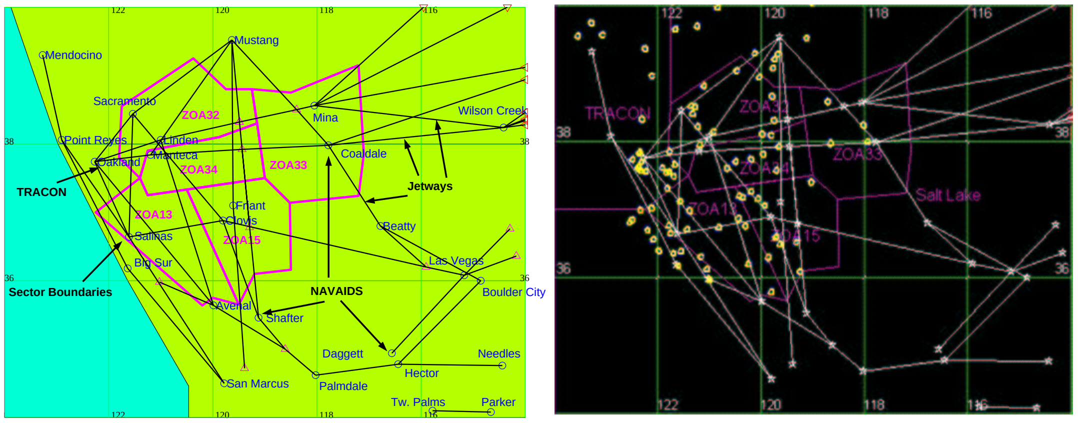


Figure 3.1: **Left:** ATC sectors modeled for this study: 32, 33, 34, 13 and 15 within the Oakland ARTCC (labeled above as ZOA32, etc.). Most crucial jetways and waypoints for a San Francisco approach are shown here. The data modeled comes from FACET [39] as well as JEPPESEN high altitude en route charts [93]. **Right:** Visual display of the simulator used in [17]. Traffic in the Oakland ARTCC (aircraft not in this Center have been filtered out). This plot has been generated using ETMS data.

This chapter is organized as follows. Section 3.1 gives a short introduction to *hybrid systems*, which is a very powerful framework for modeling single maneuvers in this problem (a more mathematical formalization of hybrid systems will be given in Chapter 4). This section also describes how the hybrid system model shown here can be used to construct aircraft trajectories made up of successive maneuvers. Once this setting is in place, an analytical model of airspace congestion is derived in Section 3.1.2. Because it is impossible to use the real ATM system as a testbed to assess the accuracy of our model, a simulator is designed, implemented and validated (Section 3.2) and used as a replication of the real system. It is then used to validate the accuracy of our analytical model in Section 3.3.

3.1 Sector-based air traffic flow modeling and analysis

The structure of the NAS is complex, for it involves a multitude of interacting agents and technologies: aircraft monitoring, flow management, communication, and human-in-the-loop. For the present work, in which we are interested in predicting delay, we extract and model only the features which are important for this purpose. We model a portion of the Oakland ARTCC, which contains five *sectors*. These sectors surround the Oakland

TRACON, which controls the aircraft on their approaches into San Francisco, San Jose and Oakland airports. The TRACON is the final destination of the traffic that we consider in this chapter.

We model a sector by a portion of airspace containing aircraft under the local control of the responsible Air Traffic Controller (Figure 3.1). The interior of this domain is the controlled area (in which the local Air Traffic Controller can actuate the flow). Within each sector, navigation infrastructure, including jetways, navigation aids, and *waypoints*, is used to help the flow follow desired patterns; we therefore include it in the model and use it, even if it is observed that more than 40% of the aircraft may fly off the jetway at any given time. Our model allows for aircraft to fly at different altitudes, but not to climb or descend. Altitude changes are not crucial for the effects we want to identify: the type of sector overloads we are interested in mainly results from aircraft acceptance rates at destination airports.

3.1.1 Aircraft behavior

We use a *hybrid model* for each aircraft. A hybrid model describes the evolution of a system by a set of *discrete modes*, each associated with a *continuous dynamical system* and *discrete switches* which enable the system to jump from one mode to another instantaneously. In mathematical terms, we will describe the motion of aircraft i by:

$$\dot{\vec{x}}_i = \frac{d\vec{x}_i}{dt} = \vec{v}_{\text{current heading}} \tag{3.1}$$

where $\vec{v}_{\text{current heading}} \in \mathbb{R}^2$ is a constant velocity vector held by the aircraft until the next heading or speed change. $\vec{x}_i \in \mathbb{R}^2$ is the planar position of aircraft i. Integration of equation (3.1) over time produces a continuous piecewise affine trajectory. We prefer such a model over a continuous dynamical model for two reasons. First, the time scale of a "change in aircraft behavior" (for example a turn or slow down) is on the order of 30 seconds, whereas the time scale of a straight line portion of the flight is usually much longer, sometimes half an hour or more, thus we ignore the dynamics of such maneuvers and focus on their effects only (the set of resulting straight lines). Second, the update rate of ATC monitoring is in general not more than 30 seconds, which makes the details of these maneuvers inaccessible to the ATC. This approximation is widely accepted in the literature [68, 130, 37, 107, 122, 33].

We augment this set of concatenated portions of trajectories with a specific maneuver, called *holding pattern*, in which each aircraft flies circular trajectories (as shown in Figure 3.3). The kinematic equations related to this maneuver can be written as a dynamical system similar to (3.1) very easily. Monitoring ATC shows that a finite set of maneuvers, which depends on local parameters, is used. Combinations of these maneuvers result in a conflict-free flight environment in which the constraints of the air traffic flow are met. The maneuvers shown in Figure 3.2 are modeled (and consist of changing the right hand side of (3.1) according to certain rules which we now make explicit). The validity of models similar to this has been confirmed by statistical studies realized in [83].

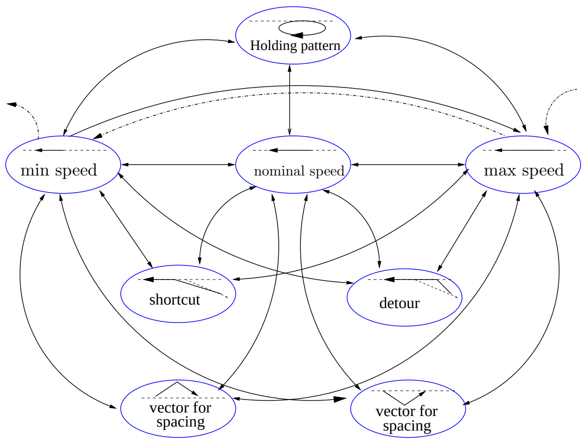


Figure 3.2: Hybrid automaton representing the action of one Air Traffic Controller on a single aircraft. Each of the eight modes represents one possible state of the aircraft. The arrows joining these states are the mode switches, initiated by the Air Traffic Controller. The dash-dotted transitions are used for the analytical solution. The complete set of arrows is used for the simulation.

1. *Speed change:* ATC may decelerate or accelerate the aircraft along its flight plan:

$$\vec{v}_{\text{modified speed}} := \lambda \cdot \vec{v}_{\text{current heading}} \tag{3.2}$$

   where $\lambda \in \mathbb{R}^+$ defines the magnitude of the velocity change. Our model will allow a finite set of speeds (which means λ has a finite number of acceptable values). This encodes the fact that the ATC generally has a finite set of possibilities in the choice of speeds, because the aircraft flies at its optimal speed per altitude and ATC will speed up or slow down the aircraft by not more than 10% of the current value.

2. *Vector-for-spacing (VFS):* This maneuver consists of a deviation of the aircraft from its original flight plan for a short time (part 1 of the maneuver), and a second deviation, bringing it back to its original flight plan (part 2 of the maneuver). This stretches the path that the aircraft must follow, and therefore generates a delay. The length of this maneuver depends on the geometry of the sector. Calling R_ψ the rotation matrix by angle ψ, we have:

$$\begin{aligned} \vec{v}_{\text{part 1}} &:= R_\psi \cdot \vec{v}_{\text{current heading}} && \text{(First half of the maneuver)} \\ \vec{v}_{\text{part 2}} &:= R_{-2\psi} \cdot \vec{v}_{\text{part 1}} && \text{(Second half of the maneuver)} \end{aligned} \tag{3.3}$$

   This maneuver is illustrated in Figure 3.3.

3. *Shortcut / Detour:* In certain situations, the ATC will have the aircraft "cut" between two jetways, a maneuver which could either shorten or lengthen the flight plan. The decision to command such a maneuver is often dictated by conflict resolution, but could also be used to shorten the overall flight time if sector occupancy allows it (sometimes called "direct-to" by pilots):

$$\vec{v}_{\text{shortcut}} := R_\psi \cdot \vec{v}_{\text{current heading}} \quad \text{for the duration of the maneuver} \tag{3.4}$$

   until the next ATC action is taken. Here again ψ is the angle by which ATC turns the aircraft to achieve the shortcut.

4. *Holding pattern:* In some extreme conditions, holding patterns are used to maintain an aircraft in a given region of space before eventually letting it follow its original flight plan. This is modeled by assigning the aircraft to a predefined zone and keeping it there while preventing other aircraft from entering that zone. This maneuver is illustrated in Figure 3.3.

3.1.2 Lagrangian analysis of delay propagation in the NAS

A large proportion of air traffic jams, i.e. portions of airspace saturated by aircraft, is generated by restrictions imposed at destination airports, usually themselves driven by weather or airport congestion. These restrictions are imposed as either miles-in-trail or minutes-in-trail, representing the distance (or time) required between aircraft in a flow

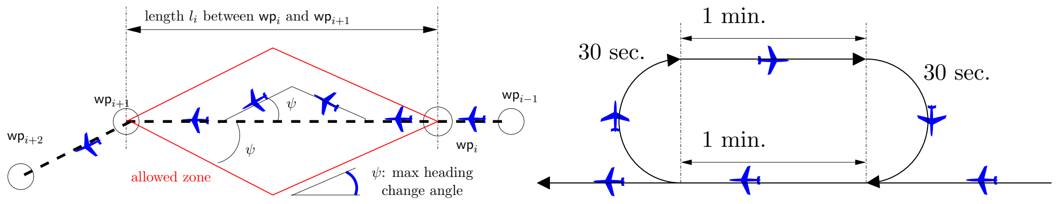


Figure 3.3: Two examples of maneuvers corresponding to modes of the automaton in Figure 3.2. **Left:** Deviation ψ from flight plan using a *Vector For Spacing* (VFS) available in a given sector. More precise models are also available in [67]. **Right:** *Holding Pattern* (HP). The prescribed "time to lose" is given to the aircraft in minutes: one minute in each straight portion and 30 seconds in each half circle. For this scenario, $T_{\mathsf{hp}} = 1\,\text{min}\,. + 30\text{ sec.} + 1\,\text{min}\,. + 30\text{ sec.} = 3\,\text{min}.$

incoming to the TRACON, and are referred to as *metering constraints*. Figure 3.4 illustrates the topology of the flows incoming to San Francisco (SFO) which are often subject to this type of constraint. These constraints tend to propagate backwards from the airport into the network, and result in miles-in-trail constraints imposed at the entry points of each sector. For example, in the case of Figure 3.4, typically, the backpropagation of these metering conditions is as follows: TRACON $\rightarrow$ sector 34 $\rightarrow$ sector 33 $\rightarrow$ Salt Lake Center $\cdots$ and similarly for the two other flows.

In the current system, these restrictions are imposed empirically. We would like to understand *(i)* how the traffic jams propagate; and *(ii)* what the optimal control policy should be under these restrictions, in order to ensure maximal throughput into the TRACON.

Shock wave propagation

We present a simple Lagrangian model of merging flows [18, 19, 17]. This model of merging flows predicts the backpropagation of a traffic jam from a destination airport into the system. It is Lagrangian, because it models the trajectories of all agents in the considered space, rather than averaged quantities in a control volume, such as the number of aircraft in a given sector. This model can be applied to the merging flows of the type shown in Figure 3.4. We will use it here to derive the *dynamic capacity* of a sector, i.e. the number of aircraft that can be actuated inside the sector (given inflow and outflow conditions), before saturation is reached. We solve the following problem:

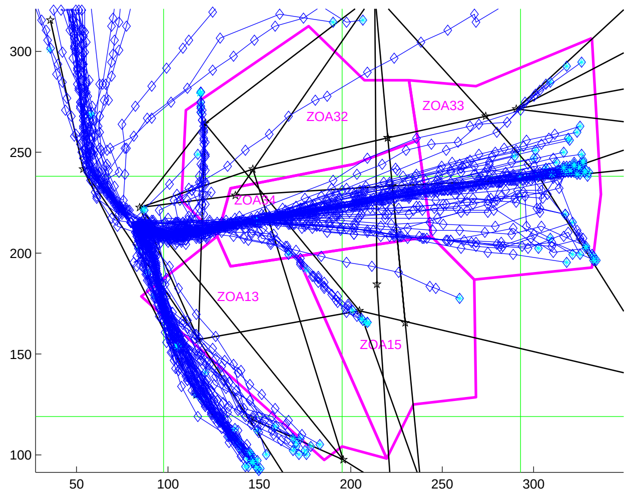


Figure 3.4: Overlay of trajectories merging into San Francisco (11 hours of traffic). The data modeled comes from ETMS and FACET [39]. The Lagrangian approach presented in this chapter models each of the trajectories included in this plot.

Problem 1. *Given a metering constraint of ΔT_{out}, compute a controller policy which will force groups of aircraft to* exactly *satisfy the metering constraint at the sector exit point (each aircraft is separated by exactly ΔT_{out}) while maintaining separation at all times.*

For this, we introduce the initial position a_i^0 of aircraft i and the location x_{ex} where the metering condition is imposed, $i \in [1, N]$ where N represents the total number of aircraft. We consider the following problem: all aircraft are initially at maximal speed $v_{\max}$, and in order to enforce metering, ATC slows down aircraft i to its minimum speed $v_{\min}$ (see Figure 3.5), at a location x_i^{switch} and time t_i^{switch} which we will determine (see Figure 3.6). This scenario is represented as a dash-dot line in Figure 3.2. We impose the condition that aircraft i (where $i \in \{1, \cdots, N\}$) crosses the metering point x_{ex} at exactly $t_{\text{block}}+(i-1)\Delta T_{\text{out}}$ where t_{block} is the time when the metering condition is invoked. This leads to the following

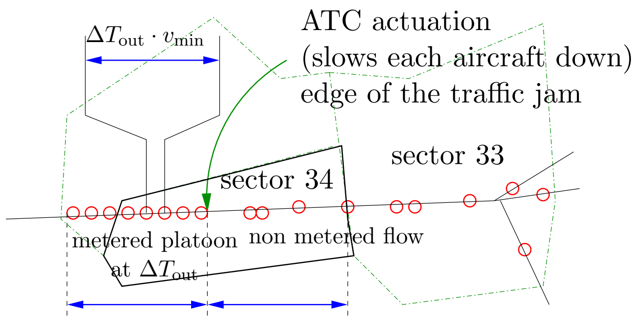

$\Delta T_{\text{out}} \cdot v_{\min}$
ATC actuation
(slows each aircraft down)
edge of the traffic jam
sector 33
sector 34
metered platoon
at ΔT_{out}
non metered flow

Figure 3.5: ATC control action on the merging flow. The traffic jam extends from SFO to the edge of the platoon metered by $\Delta T_{\text{out}} \cdot v_{\min}$ (one aircraft every ΔT_{out} minutes). Once the actuation point (edge of the traffic jam) has moved upstream (into sector 33), sector 34 is called saturated.

kinematic equations of the aircraft under this actuation:

$$\begin{aligned} x_i(t) &= a_i^0 + v_{\max} t \quad & t \in [0, t_i^{\text{switch}}] \\ x_i(t) &= b_i + v_{\min} t \quad & t \in [t_i^{\text{switch}}, t_{\text{block}} + (i-1)\Delta T_{\text{out}}] \end{aligned}$$

The assumption of continuity of $x_i(t)$ enables us to solve for b_i, from which we deduce the switching time as: $t_i^{\text{switch}} = (b_i - a_i^0)/(v_{\max} - v_{\min})$. Under the following feasibility conditions:

$$\begin{aligned} & a_i^0 \geq x_{\text{ex}} - v_{\max}(t_{\text{block}} - (i-1)\Delta T_{\text{out}}) \\ \text{and} \quad & a_i^0 \leq x_{\text{ex}} - v_{\min}(t_{\text{block}} - (i-1)\Delta T_{\text{out}}) \end{aligned} \tag{3.5}$$

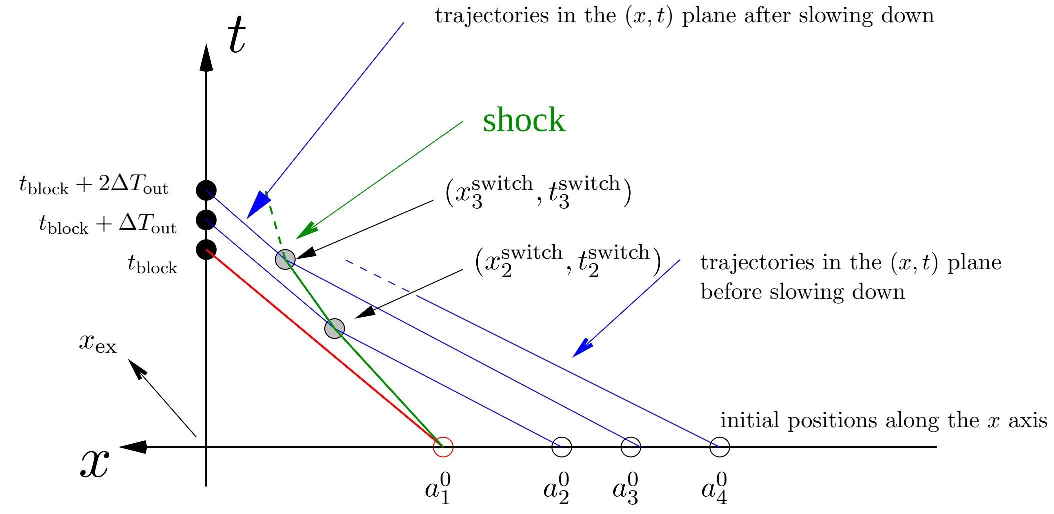
the propagation speed of the traffic jam may be computed analytically, by solving for the location of the edge of the traffic jam in space and time:

$$\begin{aligned} t_i^{\text{switch}} &= \tfrac{x_{\text{ex}} - v_{\min} t_{\text{block}} - (i-1)\Delta L - a_i^0}{v_{\max} - v_{\min}} \\ x_i^{\text{switch}} &= a_i^0 + \tfrac{v_{\max}[x_{\text{ex}} - v_{\min} t_{\text{block}} - (i-1)\Delta L - a_i^0]}{v_{\max} - v_{\min}} \end{aligned} \tag{3.6}$$

where $\Delta L := v_{\min} \Delta T_{\text{out}}$ is the metered spacing at the outflow of the sector. It follows directly from (3.6) that the traffic jam will not grow if the two following conditions are met:

$$\begin{aligned} t_i^{\text{switch}} < t_{i+1}^{\text{switch}} \quad &\Leftrightarrow \quad \Delta L < a_i^0 - a_{i+1}^0 \\ x_{i+1}^{\text{switch}} < x_i^{\text{switch}} \quad &\Leftrightarrow \quad \left(\tfrac{v_{\min}}{\Delta L}\right) < \left(\tfrac{v_{\max}}{a_i^0 - a_{i+1}^0}\right) \end{aligned} \tag{3.7}$$

Condition (3.6) above is a sufficient condition for the length of the traffic jam to decay (which can be observed by inspection of the slope of the shock wave displayed in Figures 3.6 and 3.7) and it is a local property of the problem: it only depends on $a_i^0 - a_{i+1}^0$, not on all



trajectories in the (x,t) plane after slowing down
t
shock
$t_{\text{block}} + 2\Delta T_{\text{out}}$
$t_{\text{block}} + \Delta T_{\text{out}}$
t_{block}
$(x_3^{\text{switch}}, t_3^{\text{switch}})$
$(x_2^{\text{switch}}, t_2^{\text{switch}})$
trajectories in the (x,t) plane before slowing down
x_{ex}
initial positions along the x axis
x
a_1^0
a_2^0
a_3^0
a_4^0

Figure 3.6: Shock wave construction. The aircraft trajectories are represented in the (x,t) plane. They originate at $t = t_{\text{block}}$ from the horizontal axis (white circle on each trajectory). After some amount of time, the aircraft is switched to speed v_{min} at location $(x_i^{\text{switch}}, t_i^{\text{switch}})$ (shaded circle on each trajectory). Ultimately they reach x_{ex}, the entrance of TRACON (black circle).

aircraft considered here. The second equation in (3.6) can be linked to a one-dimensional discretized steady state Lighthill-Whitham-Richard equation, which appears naturally in highway congestion problems [121], as will be discussed in the next chapter. This fact links this Lagrangian approach, based on aircraft trajectory analysis, with Eulerian approaches such as [110], which are based on conservation equations; it relates local properties of the flow (local monotonicity of a variable, here the x-location of the wavefront) to global quantities (here the trajectories of the aircraft). The construction of this result is illustrated in Figure 3.6 for the methodology. A numerical example is provided in Figure 3.7.

The condition that each aircraft reaches x_{ex} exactly at the time prescribed is restrictive, what is important is the flow rate but not the actual crossing times. Therefore it would be of greater use to pose the problem as follows:

Problem 2. *Given $\{a_i^0\}_{i\in\{1,N\}}$, compute the switching policy which delivers at most one aircraft every ΔT_{out} sec. at the location x_{ex} while maintaining separation, and minimizes the arrival time of aircraft N.*

This cannot be solved as easily analytically because the arrival times of all aircraft are now variables and need to be computed while satisfying constraints. However, this problem may

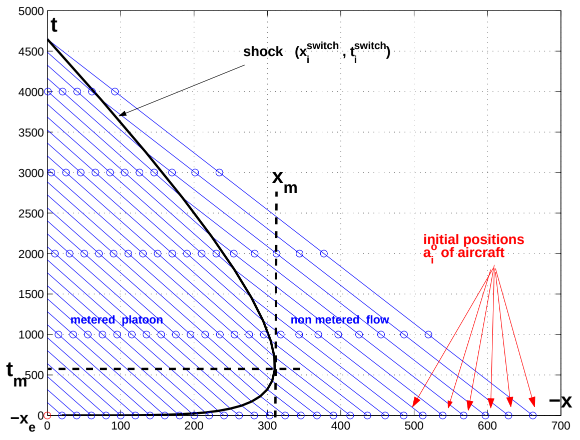


Figure 3.7: Example of switching curve (shock) for a vanishing traffic jam, for the specific case $t_{\text{block}} = 0$. x denotes the distance to the metering point (SFO). The lines are the trajectories of the aircraft in the (x, t) space. The positions of aircraft are represented every 1000 sec. as dots. Once they have passed through the shock, they are metered (at one aircraft every $v_{\min}\Delta T_{\text{out}}$). The point (x_m, t_m) is the furthest reachable point by this traffic jam.

be posed as a linear program: minimize the arrival time of aircraft N, while separating the aircraft by more than ΔT_{out} at x_{ex}, with at most one switch between the initial position $a_i^0 \le x_{\text{ex}}$ and the exit x_{ex} of the considered airspace:

$$\textbf{Minimize} \quad [0, \cdots 0, -1]\vec{b}$$

Subject to

$$\begin{bmatrix} -1 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & -1 & 1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & -1 & 1 & 0 \\ 0 & \cdots & \cdots & 0 & -1 & 1 \end{bmatrix} \vec{b} \succeq v_{\min} \begin{bmatrix} \Delta T_{\text{out}} \\ \vdots \\ \vdots \\ \vdots \\ \Delta T_{\text{out}} \end{bmatrix}$$

$$\vec{a} \preceq \vec{b} \preceq \frac{v_{\min}}{v_{\max}} \overrightarrow{a} + \left(1 - \frac{v_{\min}}{v_{\max}}\right) x_{\text{ex}} [1, \cdots, 1]^T$$

where $\vec{a} = [a_1^0, \cdots, a_N^0]^T$ and $\vec{b} = [b_1, \cdots, b_N]^T$. In the previous formula, $\succeq$ means componentwise inequalities. Note that the right hand side of the first matrix inequality can be changed to $[\Delta T_1, \cdots, \Delta T_N]^T$ in order to account for time-varying boundary conditions. The advantages of this formulation are that one includes the possibility to optimize an objective function, which is in the present case the arrival time of the last aircraft in the platoon, and that one is able to deal with time varying acceptance rates at the airport. Condition (3.7) for shock monotonicity derived previously is still valid locally for any solution derived with the linear program above.

Sector overload predictions

Using the analysis of the previous section, it is fairly easy to predict the dynamic capacity of a sector. Consider the worst case scenario: an incoming flow of aircraft, each at $v_{\max}$, separated in time by ΔT_{in} chosen to violate the second condition in (3.7). This will create a traffic jam originating at SFO, which "piles up" and progressively fills sector 34. Calling l the length of sector 34 along the main jetway, we use equations (3.6) to compute the maximal number N_{limit} of aircraft going through this sector before the edge of the traffic jam moves to sector 33 (*dynamic capacity*), and the time T_{limit} required for this to happen:

$$N_{\text{limit}} = \frac{l(v_{\max} - v_{\min})}{v_{\max} v_{\min} (\Delta T_{\text{out}} - \Delta T_{\text{in}})} \tag{3.8}$$

$$T_{\text{limit}} = \frac{l}{v_{\max} v_{\min}} \frac{v_{\max} \Delta T_{\text{in}} - v_{\min} \Delta T_{\text{out}}}{\Delta T_{\text{out}} - \Delta T_{\text{in}}} \tag{3.9}$$

Several comments can be made regarding the two previous results: *(i)* as $v_{\min} - v_{\max} \to 0$, $N_{\text{limit}} \to 0$: no aircraft can be handled in the sector additional to the aircraft already there, because no actuation is possible (it is not possible to "make the aircraft lose time" in this sector); *(ii)* as $\Delta T_{\text{out}} - \Delta T_{\text{in}} \to 0$, $N_{\text{limit}} \to \infty$ and $T_{\text{limit}} \to \infty$: if the incoming flow is such that it is almost metered as imposed at the exit of the sector, the number of aircraft required to saturate this airspace becomes large and the time it takes to saturate this sector grows accordingly.

The construction of the *switching curve* or *shock* $(x_i^{\text{switch}}, t_i^{\text{switch}})$ described previously, can be used to compute the maximal extent of a traffic jam, or the portion of jetway affected by a traffic jam. Using (3.6), one can trace the shock location in the (x, t) plane (Figure 3.7). The edge of the traffic jam, called x_m, obtained at t_m gives the worst situation obtained from the initial configuration a_i^0 of the aircraft: at t_m, the portion of space with high density of aircraft is of maximal size. In the case of Figure 3.7, we see that the traffic jam does not propagate more than 300 nm upstream from the destination of the aircraft, called

x_{ex}. Therefore no metering conditions should be applied upstream from that point. In the current system, such information is not available to the ATC, thus leading to extra buffers taken by the Air Traffic Controllers, which in turn leads to non optimal operating conditions as well as backpropagation of "virtual overload", in fact a set of conservative precautions.

3.2 Simulator design and validation

This section and the next attempt to validate the predictive results derived above, which rely on analysis of ATM procedures. The validation is a two stage process, since it is not possible to experiment on the real ATM system directly. We therefore create a simulator, which mimics human Air Traffic Controller behavior. Section 3.2.1 presents the design and implementation of the simulator. Section 3.2.2 describes the validation of the code against real data, in order to show good agreement with observations in the real system. Finally, Section 3.3 shows how analytical predictions (as presented in Section 3.1.2) can be validated against the simulator.

3.2.1 Simulator design

Since the real system is not available as a testbed, we have designed an abstraction of it which reproduces its behavior adequately. We have created a simulator which we have validated against real data (the format of this data is called ETMS data and will be explained later), which mimics true ATC behavior and against which we validate our predictions. This simulator is based on empirical studies that we realized at the Oakland ARTCC, its core is based on observed behavior. Figure 3.2 (all transitions enabled) summarizes the behavior model observed at the ARTCC. The switching logic behind the transitions is the object of this section, and is implemented in the form of a cost function, described below.

Overall Program Flow

The overall program flow of the simulator is shown in Figure 3.8. The input to the code is a set of aircraft filed flight plans (Figure 3.8, middle column), that can either be user generated or taken from ETMS data (as in FACET). As in the true system, these flight plans are not

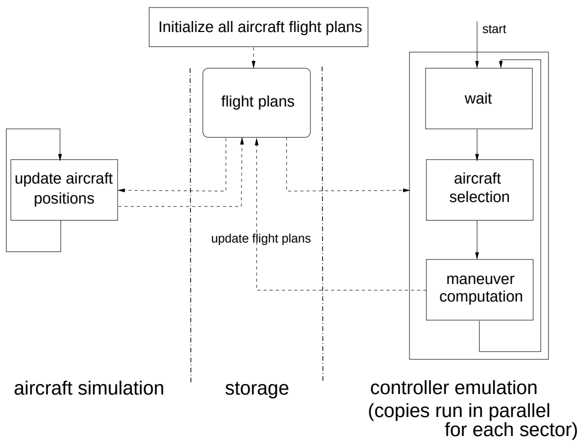

Initialize all aircraft flight plans
start
flight plans
wait
update aircraft positions
aircraft selection
update flight plans
maneuver computation
aircraft simulation
storage
controller emulation (copies run in parallel for each sector)

Figure 3.8: Program flow of the simulator.

conflict-free and usually do not satisfy metering conditions imposed on the network. Once the program is initialized, aircraft motion simulation follows these flight plans (Figure 3.8, left column). As time is advanced, conflict as well as metering constraints are dealt with on a sector by sector basis (with sector-wide look ahead, Figure 3.8, right column), according to the full automaton shown in Figure 3.2. The flight plans are updated accordingly.

Key Data Structures

Aircraft dynamic equations (3.1) produce a set of segments; the knowledge of the points connecting the segments and of the aircraft velocity is thus enough to define an aircraft trajectory. This trajectory is thus implemented as a *linked list* of *points* $[x, y, z]$, with a prescribed velocity between the points. The linked list is modified by the simulated controller in the program. The output for each aircraft is the updated linked list. The sectors are implemented as sets of accessible linked lists. They also contain additional data such as metering conditions (number of aircraft through a given boundary per time unit).

Controller Emulation

ATC behavior is modeled by three levels of priority:

- *Priority 1: No loss of separation.* The prevalent requirement for ATC is to ensure that any aircraft pair is always separated by more than 5 nautical miles.

- *Priority 2: Metering conditions.* The Air Traffic Controller needs to ensure that the outflow from his sector is an acceptable inflow for the next sector (or TRACON). Metering conditions can be of various nature: admittance rate or separation at downstream junctions.

- *Priority 3: Best possible throughput.* ATC will try if possible to give direct routes to aircraft in order to minimize their flight times.

These priorities may be modeled using the cost function J:

$$J = \text{cost}_{\text{LOS}} + \text{cost}_{\text{BC breach}} + \text{cost}_{\text{delay}} + \text{cost}_{\text{aircraft actuation}} + \text{cost}_{\text{maneuver}} + \text{cost}_{\text{min dist}} \tag{3.10}$$

Each term of the cost is a weighted function:

$$J = \sum_{i=1}^{N} \frac{n_{\text{LOS}}^i \cdot w_{\text{LOS}}}{\Delta T_{\text{LOS}}^i} + \sum_{i=2}^{N} (T_{\text{breach}}^i)^2 \cdot w_{\text{breach}} \sum_{i=1}^{N} (TOA_{\text{pred}}^i - TOA_{\text{real}}^i) \cdot w_{\text{delay}} + \\ N_{\text{moved}} \cdot w_{\text{single move}} + \sum_{i=1}^{N} J_{\text{maneuver}}^i + \sum_{i=1}^{N_{\max}} f(d_{\min}^i) \cdot w_{\text{dist}}$$

1. *Loss of separation* (LOS) *cost:* n_{LOS}^i is the number of losses of separation involving aircraft i in the current sector with its current flight plan. ΔT_{LOS}^i is the time until the first loss of separation for aircraft i.

2. *Boundary condition (BC) breach cost:* T_{breach}^i is the time by which an aircraft violates the ΔT time separation constraint from its predecessor (set to zero if the two aircraft are separated by more than ΔT).

3. *Delay cost:* $TOA_{\text{pred}}^i - TOA_{\text{real}}^i$ accounts for the difference between predicted and actual *time of arrival* (TOA) at the last waypoint of the flight. Positive delays are penalized; earlier arrivals are favored. TOA_{pred}^i and TOA_{real}^i are computed by integration of the flight plans for each aircraft.

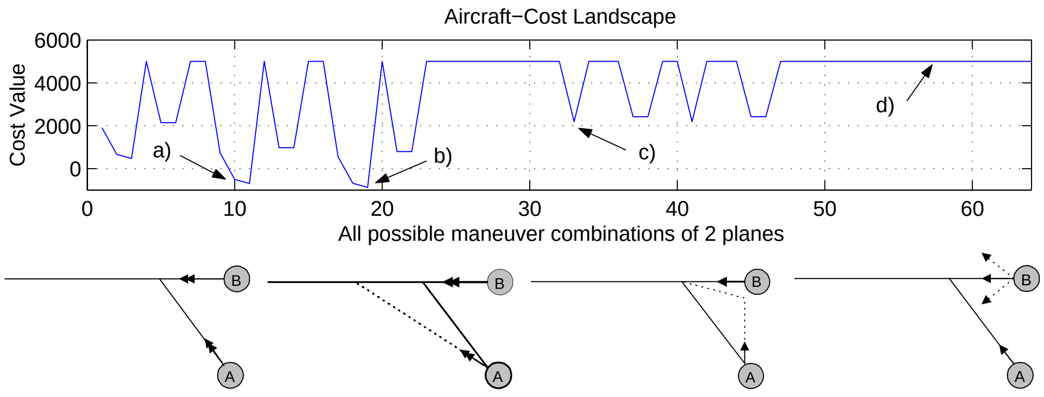


Figure 3.9: Top: cost values for all possible maneuver combinations in a two-aircraft intersection scenario, where the eight maneuvers of Figure 3.2 are enabled (thus generating $8^2 = 64$ possible values of J). Four out of 64 examples are extracted and illustrated (four lower pictures). (a) Both aircraft maintain same speed; (b) Aircraft A takes a shortcut maintaining aircraft B at max speed; (c) A makes a VFS at low speed; (d) A does nothing, B is not able to prevent the loss of separation. In this case, the simulated controller would choose solution (b) since the lowest cost is associated with that maneuver.

4. *Aircraft actuation cost:* N_{moved} accounts for the number of flight plan modifications chosen in the current solution. Large N_{moved} are penalized (the solution chosen by the ATC is often the simplest).

5. *Maneuver cost:* J^i_{maneuver} accounts for the cost of the maneuver selected for aircraft i. Not all maneuvers are of equal preference and therefore have different costs. It is easier for an Air Traffic Controller to prescribe a speed change than a VFS or a shortcut. A holding pattern is the least preferred option, for it requires constant monitoring of the aircraft. This reflects in the weight choice:

$$J^i_{\text{speed change}} < J^i_{\text{shortcut}} \sim J^i_{\text{VFS}} \ll J^i_{\text{holding pattern}} \tag{3.11}$$

6. *Minimal distance cost:* $f(d^i_{\min})$ penalizes aircraft distributions in which aircraft are closely spaced (but do not lose separation) against more sparse distributions. Here, $\text{dist}_{\max} = 7\text{nm}$.

$$\begin{aligned} f(d^i_{\min}) &= \tfrac{1}{d^i_{\min}} \cdot w_{\text{dist}} && \text{if } d^i_{\min} < \text{dist}_{\max} \\ f(d^i_{\min}) &= 0 && \text{otherwise.} \end{aligned}$$

In order to reflect the three levels of priority of the human ATC stated earlier, the weights shown in the cost function J are: $w_{\text{LOS}} \sim 10^{300} \gg w_{\text{breach}} \sim 10^4 \gg$ other weights ~ 10. Minimizing J will thus first deal with losses of separation, then metering conditions, and finally optimization of the flow. An example of a cost landscape[*] for a given topology of two aircraft is illustrated in Figure 3.9. Among all maneuvers used for flow optimization, some hierarchy is also prescribed, reflected in the relative values of $J^i_{\text{speed change}}$, J^i_{shortcut}, J^i_{VFS}, and $J^i_{\text{holding pattern}}$. These adjustments are important, but are technical and are not detailed here.

In order to reduce the computational time, we define as N_{choice} the maximum number of aircraft considered by the simulated controller in each time iteration. We restrict N_{choice} to be generally less than the actual number of aircraft per sector. This term is a trade-off between run-time and control-quality and in our simulations it was set in the range of 4 to 8, depending on the targeted goals. Aircraft are selected according to the following rule: aircraft involved in LOS are selected first, then aircraft breaching boundary conditions, and finally remaining aircraft until the selection list has reached N_{choice} aircraft, or until there are no more aircraft to select. In practice, $4 \leq N_{\text{choice}} \leq 8$, where $N_{\text{choice}} = 8$ enables more complicated situations but makes the code run more slowly. The set of all maneuver combinations for the N_{choice} aircraft is called the maneuver set.

At each iteration of the controller emulation loop, an exhaustive recursive search on the maneuver set is run in order to find a set of N_{choice} maneuvers which minimizes J. The computational complexity of finding the optimal J for N_{choice} aircraft subject to n_{maneuver} possible discrete maneuvers is $O((n_{\text{maneuver}})^{N_{\text{choice}}})$. We can reduce this cost to $O((n_{\text{maneuver}} - 2)^{N_{\text{choice}}})$: (i) the cost of the current maneuver has already been computed at the previous step and thus does not need to be recomputed; (ii) two maneuvers are mutually exclusive (shortcuts), therefore only one needs to be called. Including the cost of checking for conflicts, the total cost of a time iteration becomes:

$$O\left(N_{\max}^2 \cdot (n_{\text{maneuver}} - 2)^{N_{\text{choice}}}\right)$$

where $N_{\max}$ represents the total number of aircraft in the sector. Due to both the discretization of time, and the restriction of the search space to a manageable number of aircraft,

[*]In Figure 3.9, the cost J has been truncated at $5 \cdot 10^3$ for readability. Thus a LOS cannot be visually differentiated from a breach in this plot, though it can in our data. Thus, too large breaches as well as LOS do not appear on this plot.

our search is not guaranteed to find the global optimum. However, we believe that the search does provide a reasonable approximation of the Air Traffic Controller's behavior. By adjusting the two key control parameters, the number of selected aircraft N_{choice} and time between controller activation ΔT_{act}, a transparent trade-off between run-time and control quality can be made.

3.2.2 Code validation against ETMS data

The controller logic presented in the previous section is the result of numerous observations we made in the Oakland ARTCC, monitoring the work of Air Traffic Controllers. The fact that one can classify ATC action into a set of preferred directives was experimentally validated for a different airspace [83]. However, even if the automaton of Figure 3.2 and the cost function of the previous section implemented in the simulator are consistent with our observations, there is no a priori guarantee that these would produce the same effects on the system as a human Air Traffic Controller. For this reason, we would like to assess how well our cost function describes the decision making of a human Air Traffic Controller. We therefore need to validate our code against recorded aircraft trajectories. We use ETMS data provided by NASA Ames.

The ETMS database contains all flight plan information for flights in the NAS. Data are collected from the entire population of flights in the NAS with filed flight plans. ETMS data is sent from the Volpe National Transportation System Center to registered participants via the Aircraft Situation Display to Industry electronic file server. The Federal Aviation Administration (FAA) uses these data to monitor the effectiveness of its National Route Program, in which the user community is offered flexible, cost-effective routing options as an alternative to published ATC preferred routes. A subset of the data used is shown in Figure 3.10.

We first explain the data extraction process which enables us to convert ETMS data to a readable format for our simulator. We then present the three types of validations realized.

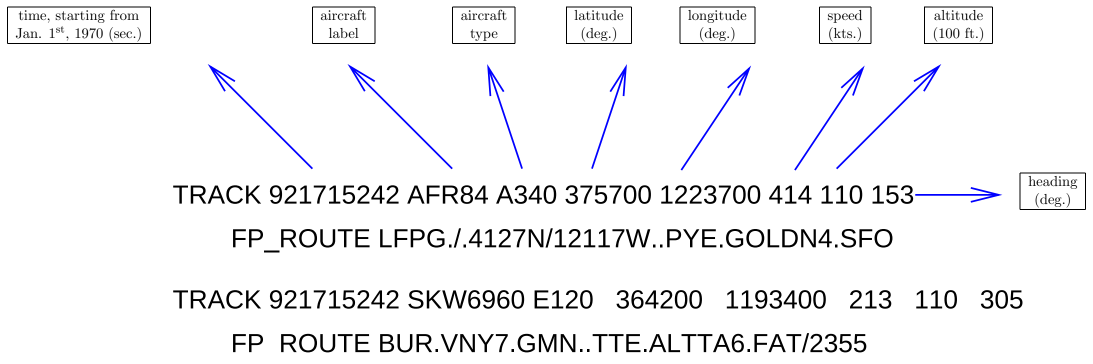

time, starting from Jan. 1st, 1970 (sec.)
aircraft label
aircraft type
latitude (deg.)
longitude (deg.)
speed (kts.)
altitude (100 ft.)
heading (deg.)
TRACK 921715242 AFR84 A340 375700 1223700 414 110 153
FP_ROUTE LFPG./.4127N/12117W..PYE.GOLDN4.SFO
TRACK 921715242 SKW6960 E120 364200 1193400 213 110 305
FP ROUTE BUR.VNY7.GMN..TTE.ALTTA6.FAT/2355

Figure 3.10: Format of the ETMS data subset used for the model validation. The data is available approximately every 3 min. We only use a subset of the data (two lines per airborne aircraft per time tag, whereas in the full data set, ETMS also contains lists of jetways and centers). The first line gives the flight information for one airborne aircraft in the NAS: time, flight number, aircraft type, latitude, longitude, speed, altitude and heading. The second line is the filed flight plan and can be used to determine the intent of the aircraft. Each of the acronyms in this line corresponds to a navaid, a fix, a jetway or an arrival (for full details and definitions, we use the same database as in [17, 19]).

Data extraction

We can extract two types of information from the ETMS data: the actual flown aircraft trajectories, and the filed flight plans for each aircraft (eventually updated if modifications are made during the flight). The position is given in latitude / longitude and in terms of navaids, fixes, and jetways which we represent in cartesian coordinates, an approximation valid for the portion of airspace of interest to us. The filed flight plan is given in terms of navaids, fixes, jetways, which we also convert into cartesian coordinates using a public database (`http://www.airnav.com`). Future versions of our simulator might use recently developed ETMS analysis tools such as [103] to perform these tasks automatically.

This study is limited to sector control, and we did not implement the *Traffic Management Unit* (TMU) action. TMU operates at the Center level and makes strategic flow scheduling decisions, which go beyond the range of a single sector Air Traffic Controller. We therefore need to validate our simulator at a scale at which TMU actuation is already incorporated in the flight plans (typically one or two sectors). Since our interest focuses on sectors 32, 33, 34, 15, and 13, the actual flight plans are truncated, and we keep only points in that sector. The filed flight plans are truncated similarly. Their estimated time of entrance in the sector is set to the actual time of entrance of the closest actual recorded point. The entrance point in the computational domain is taken to be the closest conflict-free point to the intersection

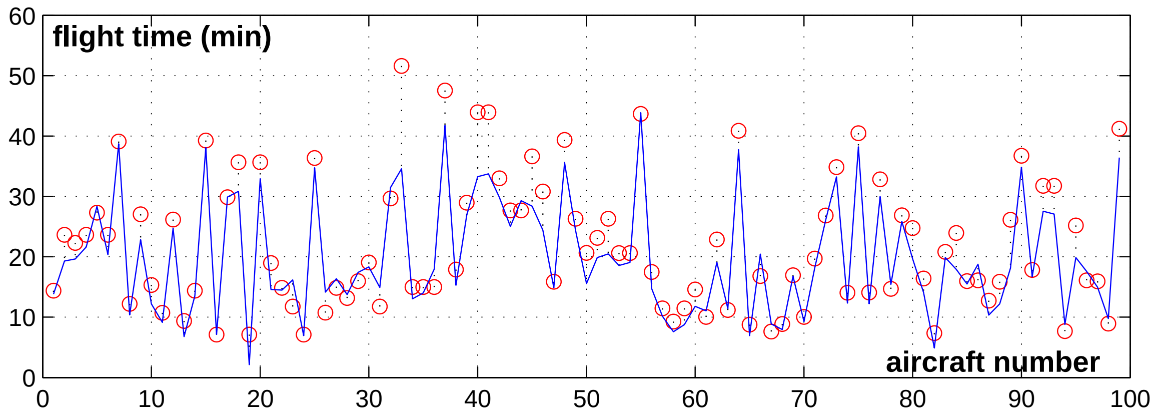


Figure 3.11: Flight time comparisons for the first 100 aircraft going through sector 33 in the ETMS data set we used. The dots are the flight times for the ETMS recorded points. The solid curve is the result of our simulations.

of the flight plan and the boundary of the corresponding sector. The altitude assigned to the filed flight plans is set to the average altitude of the actual trajectory in that sector.

Validation

Comparison of flight times. We select the first 100 aircraft of the ETMS data set which are flying above 33000ft for more than 6 minutes. Their recorded trajectories are extracted as sequence of waypoints which are used as initialized flight plans for our simulations. We compare the flight time in the simulation to the real one. The experiment is run for the following set of speeds: $M \in \{0.6, 0.7, 0.8\}$ (M is the Mach Number), which matches our observations in the data for this altitude. In the run, the controller is activated every $\Delta T_{\text{act}} = 10$sec. The results are shown in Figure 3.11. Two main conclusions can be made: *(i)* the simulator is able to recreate the flow without major modification, and eventually resolves apparent conflicts in the data – these conflicts can be due to inaccuracy of the measurements or transmission (two aircraft separated by less than 5 nm at the same altitude), or due to problems of interpolation when speed changes in time; *(ii)* the time comparison (Figure 3.11) shows relatively good matching. The flight times provided by the simulator are usually shorter than in reality because by default the simulator will always try to maximize the throughput in the sector. The mean deviation is 120 sec. for flight plans with an average duration of 1300 sec. (9.2% error).

Verification of conflict resolution. We select aircraft flying through sector 33 in a time frame of 10 hours (a total set of 314 aircraft) and simulate these flights. The filed flight plans are not conflict-free. We would like to show that the simulator is able to provide a conflict-free environment. For this run, the activation time of the controller is $\Delta T_{\text{act}} = 20\text{sec.}$[†] The set of speeds allowed is $M \in \{0.55,\ 0.75,\ 0.89\}$ (since we are considering the full range of altitude, we need to consider the full range of speed). The simulator is able to provide a conflict free environment. During the simulation, it has to actuate 50 different aircraft. The number of resolved conflicts can be assumed to be on the same order, since a single intervention will usually resolve not more than one conflict.

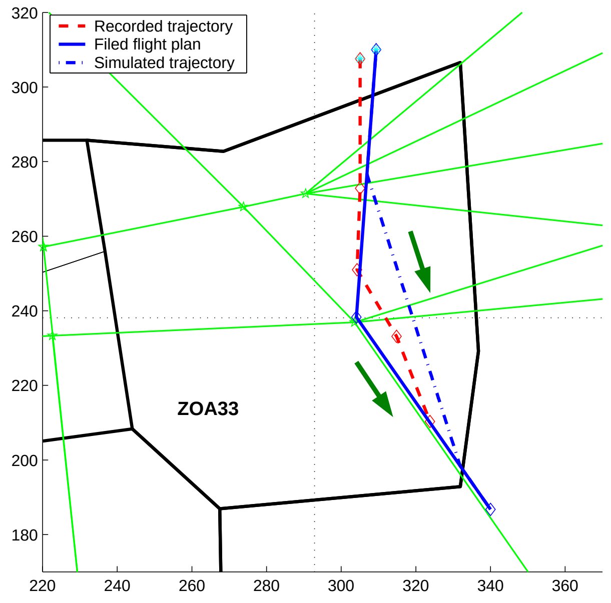


Figure 3.12: An example of maneuver caused by conflict resolution, reproduced by the simulator. The recorded data (dashed) exhibits a shortcut from the filed flight plan (solid). The simulated trajectory (dashed-dotted) is a shortcut of the same type.

Validation of maneuver assignments. The core of the simulator is the model of the human

[†]$\Delta T_{\text{act}} = 10\text{sec.}$ or $\Delta T_{\text{act}} = 20\text{sec.}$ is on the order of the maximal actuation rate of a human Air Traffic Controller. We choose $\Delta T_{\text{act}} = 20\text{sec.}$ in this particular case because of the duration of the computation (10 hours of real time are simulated).

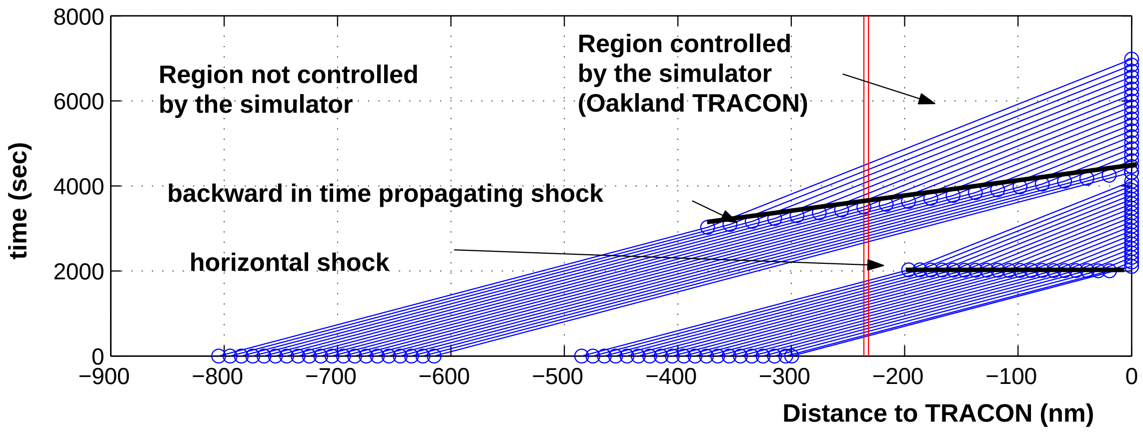


Figure 3.13: Shocks generated by two successive platoons. The first shock is steady in time (it only propagates backward in space). It corresponds to a piling up process on a highway where all vehicles slow down at the same time. The second shock propagates backward in space and time (which is much harder to handle in practice, because actuation must be performed upstream first).

Air Traffic Controller by a decision procedure based on the cost function described previously. The validation so far has shown the correlation of flow patterns generated by our code and these observed in reality. We would also like to assess the validity of our decision procedure. For this, we identify conflict resolution maneuvers, which are typically obtained by identifying deviations from the filed flight plan. For these maneuvers, we generate the following input data: all aircraft are assigned their actual flight plan (recorded trajectories), and the aircraft for which the maneuver is identified is assigned its filed flight plan (a set of way points). We thus put the simulator in the same situation as the human Air Traffic Controller, in which it has to make the decision that was actually taken. For sector 33, we were able to identify 20 distinct maneuvers out of 300 examined flight plans. The simulator reproduced 16 of them.[‡] An example is shown in Figure 3.12.

[‡]Small-scale maneuvers are less likely to be executed correctly by the simulator because the probability of selecting the respective aircraft at exactly the 'right' time is small, which explains the small discrepancy between the results. Also, even if the maneuver is executed correctly by the simulator, the resulting flight plan will look different from the ETMS data, since the simulator is restricted to a single angle of deviation ($\theta = 22.5°$).

3.3 Validation of the analytical predictions

In this section, we assess how accurate our analytical predictions are with respect to the real system, by comparing them with simulations. We present an example of two back-propagating shocks, solved with an extension of the method explained in Section 3.1. Two platoons, each at 10 miles-in-trail, are respectively subjected to 15 miles-in-trail and 20 miles-in-trail outflow boundary conditions (the boundary conditions for platoon 2 start after all aircraft of platoon 1 have reached the TRACON at time $t = 4300$). The speeds are $M \in \{0.59,\ 0.75,\ 0.89\}$. The aircraft flows for this run are shown in Figure 3.15. The results and interpretations are shown in Figures 3.13 and 3.14. Two shocks appear successively. From Figure 3.13, we can see that within the second platoon, the first twelve aircraft need to be actuated within the Oakland Center, whereas the last eight need to be actuated upstream (Salt Lake Center). Since in general, no knowledge of the required boundary conditions is propagated upstream, we can predict that in the real system, the last eight aircraft would not be actuated until they enter the Oakland Center and that no solution to this metering problem would be found without putting the aircraft on hold.

We verify this by simulating this flow. In Figure 3.14, we see that for the last eight aircraft, the first activation time in the simulator is higher than the predicted (upper plot): the simulated controller is not able to actuate the aircraft on time, because they are not in its airspace. We can verify on the middle plot that these aircraft are breaching the boundary conditions (by about one minute each), which can also be seen in the lower plot: their delays (the inverse of the cumulated breaches) become negative. This is an illustration of distributed and decentralized control: the actuation occurs in different sectors, and the only communication between the sectors is through the metering conditions. Obviously, the lack of centralized actuation (here communication and strategic TMU planning) disables efficient flow scheduling.

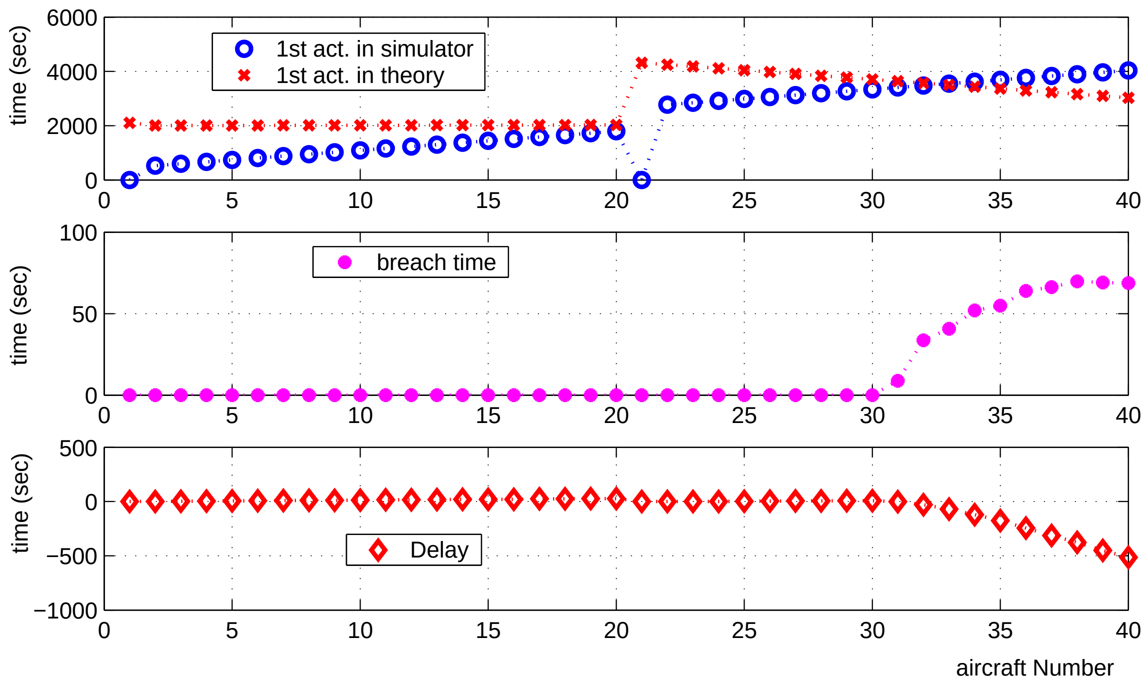


Figure 3.14: First actuation times of the aircraft (upper plot), simulated and predicted; breaches in metering conditions (middle plot), simulated; delays (lower plot), simulated, for the case of the two platoons of Figure 3.13.

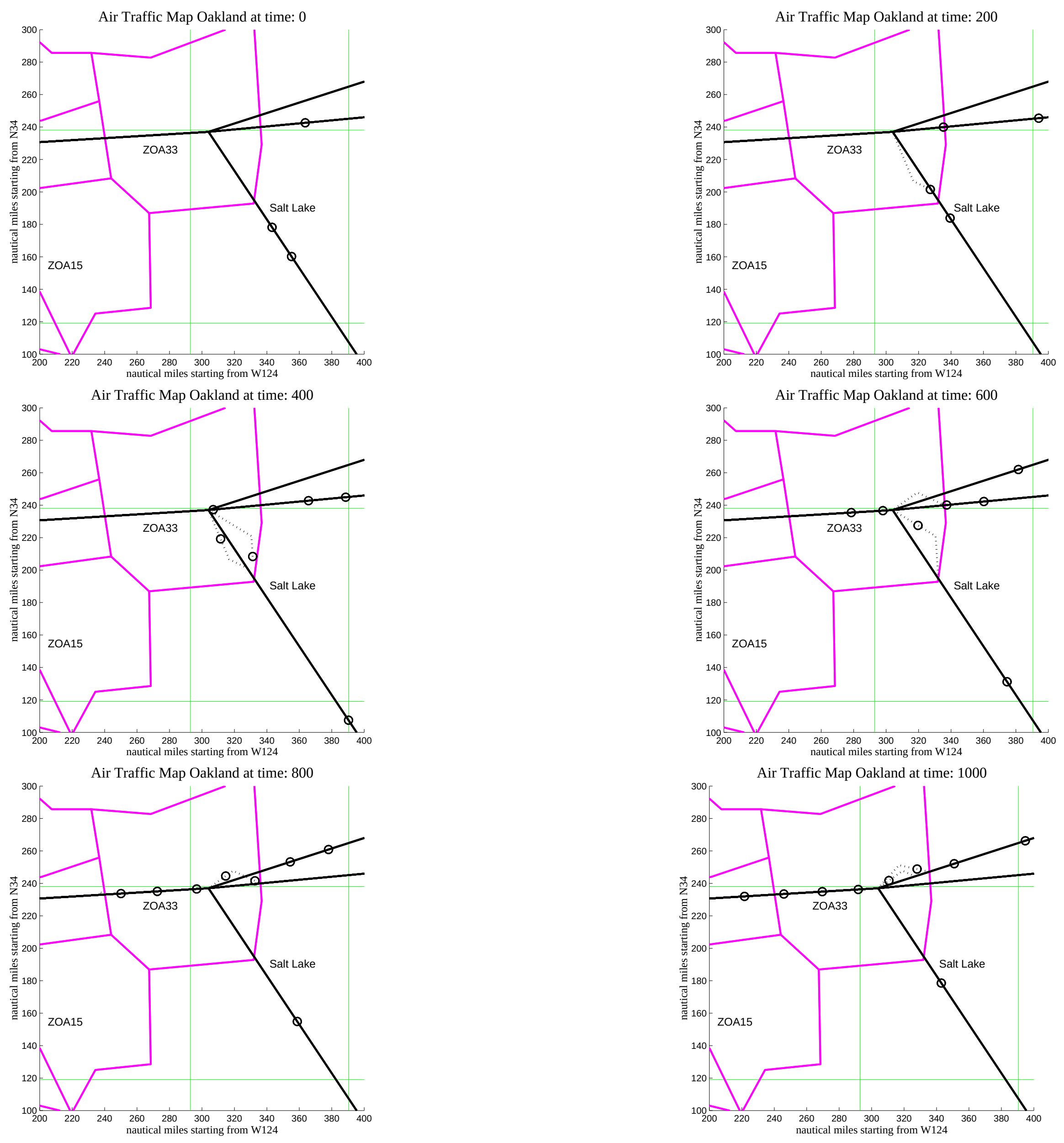

Air Traffic Map Oakland at time: 0
Air Traffic Map Oakland at time: 200
Air Traffic Map Oakland at time: 400
Air Traffic Map Oakland at time: 600
Air Traffic Map Oakland at time: 800
Air Traffic Map Oakland at time: 1000
nautical miles starting from N34
nautical miles starting from W124
ZOA33
Salt Lake
ZOA15

Figure 3.15: Sector 33: traffic flow for the merging traffic simulation of Figures 3.13 and 3.14. The radius around the aircraft is 2.5 nm. The solid lines represent the aircrafts' flight plan. The dotted lines correspond to maneuvers assigned by the simulator. The simulator makes extensive use of the vector-for-spacing to meter the aircraft.

Chapter 4

Mixed Integer Linear Program formulation of task planning

The architecture of ATM, as well as a Lagrangian model of the NAS, were presented in the two previous chapters. The model enables a better understanding of congestion propagation into the system, and was used for modeling human Air Traffic Controllers in charge of sector-based traffic flow. As we have explained, the actions that the Air Traffic Controllers take are mainly based on experience, and therefore often not optimal (for example, in the sense of trying to maximize the throughput into an airport given constraints). Furthermore, when traffic becomes too heavy, Air Traffic Controllers use *playbooks*, which are procedures that have been established over time, based on Air Traffic Controller experience. These procedures are guaranteed to be safe (i.e. in agreement with FAA regulations), but are known not to be optimal, because they add buffers between aircraft, in order to ensure that these regulations are met. In order to achieve the goal of optimizing the flow, one approach would be to enhance the Air Traffic Controller's performance by helping his decision making. Ideally, we would like a methodology which outputs a set of directives directly understandable by a human in the form of a tool which would sit next to the ATC monitoring screen and would provide him with optimal strategies to apply. The emphasis of this chapter is thus on task planning: how does one generate maneuver assignments for a set of vehicles, such that scheduling constraints are met (for example, a proper aircraft delivery rate at an airport), and furthermore how should these maneuvers assignments be provided? Finally, how to output them to the Air Traffic Controller in a language which he understands? This chapter uses the framework of hybrid systems to provide a method which synthesizes ATC understandable commands to achieve optimal solutions of air traffic problems. The method relies on an optimization problem, which will be analyzed and solved in the next chapter.

Section 4.1 presents the general problem of task planning for incoming traffic in arrival airspace. It highlights the mathematical challenges of this problem, inherent to non-convexity, which can be cast in optimization programs in which integer variables coexist with real variables. Section 4.2 uses the Lagrangian setting developed in Chapter 3, and casts the scheduling problem into a multiple hybrid system controller synthesis problem. Hybrid system theory is an excellent modeling framework for this type of problem; we use it in Section 4.3 to reduce the controller synthesis problem to a *mixed integer linear program* (MILP). In Section 4.4, we present a possible implementation architecture for the algorithm and identify the computational bottleneck due to the explosion of the combinatorial state space in the MILP. This difficulty will be solved in the next chapter.

4.1 Sequencing-routing on arrival routes

In centers which have relatively high traffic density close to terminal areas, there exist prescribed routes corresponding to different approaches into airport runways, called *arrivals*. These arrivals are the final portion of the aircraft *flight plans*. A flight plan is a set of *waypoints* (reference points defined precisely in the airspace), which the aircraft are expected to follow. Even though in low traffic density regions, aircraft might fly off these flight plans to benefit from faster routes (because of winds, for example), when this airspace becomes congested, aircraft will follow arrivals for up to 200 nautical miles (nm) from the destination airport. Arrivals aid the Air Traffic Controller in the problem of flow *metering*, or delivering a prescribed number of aircraft per unit time to the airport runway, as the routes can be viewed as tracks which the aircraft follow closely with minor deviations until they reach the arrival airport. In the current system, the Air Traffic Controllers build a *mental model* of this airspace: they know how much time an aircraft takes to fly from one point to another, and how much time an aircraft can lose using minor deviations of their flight plans in order to delay the arrival. An Air Traffic Controller can thus regulate the flow by adjusting the flight plans of individual aircraft, according to procedures (called *playbooks*) which have been established over time to meet the acceptance rates at airports. In this chapter, we study the feasibility of automatically generating these flight plan adjustments, in order to meter flow in real time, even when the system is operating at maximal capacity. Figure 4.1 illustrates the potential use of our algorithm in the current Air Traffic System.

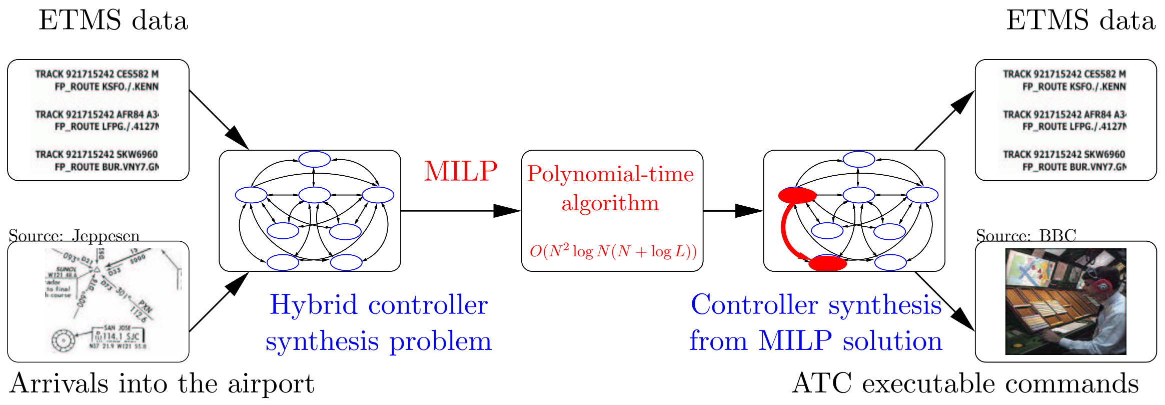


Figure 4.1: Block diagram representation of the embedding of the algorithm of this chapter within the algorithm presented in [27]. The input of the algorithm of [27] is the set of flight parameters of the aircraft (from ETMS data [39]), as well as the known structure of the arrivals. The resulting hybrid systems controller synthesis problem is posed as a MILP, which is solved by the algorithm presented in this chapter. The output, which is an execution of the hybrid system [27], is translated in ETMS data for simulation or set of ATC commands for advisory purposes.

We present and demonstrate the implementation of an algorithm which takes as input current positions of aircraft in the airspace (available from radar data through a monitoring system), produces a schedule which attempts to optimize a user-defined cost, and generates a set of commands which are contained within the command set of Air Traffic Controllers, and are thus directly understandable by pilots. These commands are represented as discrete transitions of hybrid automata (which we will define later in this document).

The main challenge in the design of this algorithm is the non-convexity of the problem, as it precludes the use of efficient convex optimization techniques with guarantees of global optimality. We employ a technique which has been used by Richards, How, Feron, Schouwenaars and coauthors [136, 135, 137, 134] to address the inherent non-convexity in conflict avoidance problems. The problem can be transformed by recognizing that non-convex constraints such as $u \in U$ where U is not convex, can sometimes be rewritten as $(u \in V) \vee (u \in W)$ where V and W are convex. More generally, non-convex constraints may be written as the disjunction of several convex constraints. If in addition V and W can be expressed as $V = \{u \in \mathbb{R}^n | \mathbf{A}u \leq b\}$ and $W = \{u \in \mathbb{R}^n | \mathbf{C}u \leq d\}$, with $\mathbf{A} \in \mathbb{R}^{p \times n}$, $\mathbf{C} \in \mathbb{R}^{q \times n}$, $b \in \mathbb{R}^p$ and

$d \in \mathbb{R}^q$ (p and q are integers), and the objective function is linear in u, the problem can be posed as a (MILP), where the logical OR ($\vee$) is encoded as an integer variable.

Transposed in the framework of hybrid systems, this idea reveals itself to be very useful. A hybrid system H_i is a system with $K_i \in \mathbb{N}$ different modes $\{q_i^k\}_{k\in\{1,\cdots,K_i\}}$. Each of these modes has a given dynamic $\dot{x}_i = f_{q_i^k}(x_i)$. We want the system to accomplish certain tasks, characterized by an initial state $x_i = x_i^{\text{initial}}$ and a final state $x_i = x_i^{\text{final}}$ (and any trajectory between these two points, defined as a sequence of discrete modes whose trajectories obey the corresponding $\dot{x}_i = f_{q_i^k}(x_i)$). There may be many possible trajectories between x_i^{initial} and x_i^{final}; for each trajectory, the elapsed time may be different, as it depends on the mode *occupancy time*, or the time spent in each mode. The *completion time* is thus a function of the trajectory; we call S_i the set of all possible completion times to go from $x_i = x_i^{\text{initial}}$ to $x_i = x_i^{\text{final}}$ using the set of modes q_i^k:

$$S_i = \left\{ t \,\middle|\, \begin{array}{l} \exists L \in \mathbb{N}, \exists \{\theta_l\}_{l\in\{1,\cdots,L\}} \in \mathbb{R}^L,\ \theta_L = t \\ \exists \mu(\cdot) : \{1,\cdots,L-1\} \rightarrow \{1,\cdots,K_i\} \end{array} \text{ s.t. } x_i^{\text{final}} = x_i^{\text{initial}} + \sum_{l=1}^{L-1} \int_{\theta_l}^{\theta_{l+1}} f_{q_i^{\mu(l)}}(\xi) d\xi \right\} \tag{4.1}$$

In the previous formula, the function μ gives the mode $q_i^{\mu(l)}$ of the system between θ_l and θ_{l+1}. S_i is in general not a single interval of $\mathbb{R}$: we will assume that it can be represented as a finite closed union of intervals, possibly unbounded in $\mathbb{R}$.[*]

We combine multiple automata H_i, where $i \in \{1,\cdots,N\}$, with prescribed initial and final conditions for each. Assume we can compute the set S_i for each H_i.[†] Call $\Theta \in \mathbb{R}^N$ a generic element of the set of possible completion times for this family of automata. $\Theta := \{t_i\}_{i\in\{1,\cdots,N\}} \in \prod_{i=1}^N S_i$. We are interested in computing a discrete control law which achieves a user-fixed linear cost in Θ, and where linear combinations of components of Θ belong to non-convex sets:

$$\begin{array}{ll} \textbf{Minimize:} & c^T \cdot \Theta \\ \textbf{Subject to:} & \Theta \in \bigcup_{p=1}^P \{u | \mathbf{A}_p u \leq b_p\} \\ & w_r^T \cdot \Theta \in \bigcup_{s=1}^S \{v | \mathbf{C}_{s,r} v \leq d_{s,r}\} \quad r \in \{1,\cdots,R\} \end{array} \tag{4.2}$$

[*]We therefore exclude the pathological cases such as $S_i = \bigcup_{i=1}^\infty \left(\frac{1}{2^{n+1}}, \frac{1}{2^n}\right)$, $S_i = \bigcup_{i=1}^\infty \left[\frac{1}{2^n}, \frac{1}{2^n} + \frac{1}{2^{n+2}}\right]$, $S_i = \bigcup_{i=1}^\infty \{e^{-n}\}$, $S_i = \mathbb{Q}$, etc. which might be of mathematical interest, but are not really useful for the physical problems we are interested in studying.

[†]Note that the computation of S_i does not necessary require the knowledge of occupancy time in each of the modes, i.e. the ability to integrate. It only requires the ability to produce tight bounds on the set of occupancy times for each modes, which is much easier. This is what we do here.

where $c \in \mathbb{R}^N$, $P \in \mathbb{N}$, $u \in \mathbb{R}^N$, $\mathbf{A}_p \in \mathbb{R}^{n_p \times N}$ ($n_p \in \mathbb{N}$, given by the constraints of the physical problem), $b_p \in \mathbb{R}^{n_p}$, $w_r \in \mathbb{R}^N$, $S \in \mathbb{N}$, $R \in \mathbb{N}$, $v \in \mathbb{R}$, $\mathbf{C}_{s,r} \in \mathbb{R}^{n_{s,r} \times 1}$, $d_{s,r} \in \mathbb{R}^{n_{s,r}}$ ($n_{s,r} \in \mathbb{N}$, given by the constraints of the physical problem). This problem in turn can be formulated as a MILP. We can retrieve from its solution the switching history (mode sequences and switching times of the corresponding hybrid automaton) which achieves it. This chapter presents an instantiation of (4.2) useful for ATC.

Our formulation bears some similarities with the formulation of Bertsimas and Stock [33, 34], however the physical problem is different. They deal with the global NAS, and optimize a cost including delays on the ground as well as airborne delays. We deal with local airborne flights which we have to adjust in their last flight portion, do not have the notion of sectors but of flight plan alteration. Panayiotou and Cassandras [131] also posed the single airport problem as we do, and treat it as a queue problem but solve it using ground hold policies. Mishra and Pappas [113] use an optical network to solve spatial conflicts; we use similar concepts, except that our conflicts are in time (and not in space).

Mixed integer linear programming [34] is a powerful mathematical formulation to extend linear programming to problems with both continuous and integer variables. It appears naturally in various fields where these two types of variables coexist, for example processing or chemical engineering [86, 72]. It enables inclusion of computational logic [155] into optimization problems, such as the program of equation (4.2), which provides an excellent tool for multi vehicle or conflict avoidance problems [136, 135, 137, 134] and discrete time hybrid systems [31, 30]. The present work enables solving continuous time hybrid systems problems under the assumptions on the continuous dynamics stated above. Of course, when the dynamics is too general, or the formulation above is not available to solve the problems of interest, more general frameworks have to be invoked, for example mixed integer nonlinear programming [124].

The rest of this chapter is organized as follows. Section 4.2 defines the mathematical model used for the aircraft, which is based on the hybrid automata models derived in Chapter 3. It explains the commands used by the ATC, and gives a model of airspace capacity (in terms of aircraft storage capability), which leads to a mathematical model for the physical constraints of the system. Section 4.3 shows how to pose the problem as a MILP and how to use the solution of the MILP to synthesize a set of executions of these hybrid automata which are ATC commands. The method is summarized in the form of an algorithm, whose implementation is demonstrated in Section 4.4. Several examples of simulations are given and analyzed, for up to 60 aircraft. Experiments have suggested that the running time grows exponentially with the number of aircraft. This difficulty will be addressed in the next chapter.

4.2 Physical model

This section presents the mathematical model we use to describe the airspace as well as aircraft motion. It is essentially used to derive a mathematical expression of the physical constraints of the problem. It represents a specific instantiation of the general model derived in Section 3.1 of the previous chapter.

4.2.1 Airspace model

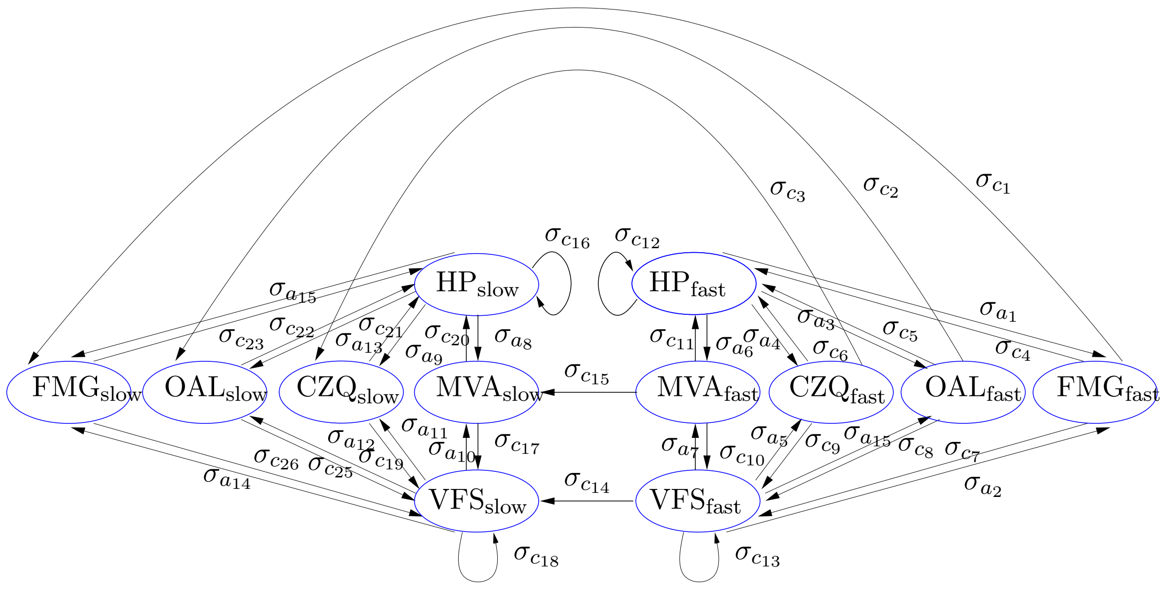
Arrival routes pass through different sectors before reaching the final descent in the TRACON. The geometry of these sectors, as well as the flows of aircraft going through them, determine the maximum possible deviations available to the aircraft. In other words, it is possible to look at the geometry of these sectors and find the maximal additional flight plan length available to ATC in order to delay the flight. This is used when speed changes are not available or not sufficient to delay a flight by a certain amount of time. The most common way to lengthen the flight plan of an aircraft is Vector For Spacing (VFS), or Path Stretching, as defined in Chapter 3, Section 3.1.1 (see also [18, 19]). Dugail et al. [67] give a method relating this additional length to the geometry of the sector. In the present study, we use the following approximation for this additional length: $l_{\text{additional}} = \sum_i l_i \left(\frac{1}{\cos\psi} - 1\right)$, where $l_i = d(\mathsf{wp}_i, \mathsf{wp}_{i+1})$ represents the great circle distance of flight plan between waypoint wp_i and waypoint wp_{i+1} (see Figure 3.3), and ψ is the maximal available rotation (heading change) angle. The VFS translates into a delay of $t_{\text{additional}} = \sum_i \frac{l_i}{v} \left(\frac{1}{\cos\psi} - 1\right)$, where v is the speed of the aircraft.

The second way for ATC to delay flights with respect to their current schedule is to prescribe Holding Patterns (HP), as defined in Chapter 3, Section 3.1.1. The LOCKE 1 arrival shown in Figure 2.4 has three possible holding patterns in the airspace, respectively at INYOE, MODESTO and GROAN. These holding patterns generate a prescribed delay for an aircraft (generally 3 minutes): one minute for each straight portion of the loop, and two times 30 seconds for the two half circles. When such an option is available to a given flight plan, our model will take it into account by adding $t_{\text{additional}} = pT_{\mathsf{hp}} = p \cdot 3$ minutes to any feasible arrival time at the destination airport, where $p \in \mathbb{N}$ is the number of loops done by the aircraft before recovering its original course. T_{hp} is in general given by the ATC to the pilot in minutes (see Figure 3.3). Note that we model the HP for the sake of completeness. However, it should be clear that one of the goals of this study is to alleviate the use of them, since they are in general the least preferred option chosen by the ATC.

4.2.2 Aircraft dynamics

In Chapter 3, we have defined and performed a validation of the model of the NAS we developed. Our model is based on hybrid automata [3, 42, 149], in which we assume that an aircraft is well modeled using a finite set of modes Q_i with a dynamical system $\dot{x}_i = f_{q_i^k}(x_i)$ associated to each mode. Figure 3.2 represents the model of the behavior of a single aircraft. This model is very general, once the aircraft in on an arrival (for example the arrival shown in Figure 2.4), the set of possible transitions for the automaton is constrained. For example: aircraft flying on an arrival are usually only decelerating; the locations of HPs and VFSs are predefined. This enables us to rewrite the automaton of Figure 3.2, as shown in Figure 4.2. The automaton of Figure 4.2 is an instantiation of the automaton of Figure 3.2, in which some of the transition have been disabled, and some of the modes have been duplicated (corresponding to fast and slow speed for example). The set of transitions is thus easier to compute: for example in the case of fast and slow, the only enabled transition is from fast to slow. The number of modes is thus higher, but the discrete part of controller synthesis (mode switches) is easier to compute. We now make the automaton explicit. It is defined as a *hybrid automaton* $H_i = (Q_i, \mathbb{X}_i, \Sigma_i, \text{Init}_i, f_{q_i}, \text{Dom}_i, R_i)$.

- $Q_i \cup \mathbb{X}_i = \{\ \text{OAL}_{\text{slow}},\ \text{MVA}_{\text{slow}},\ \text{CZQ}_{\text{slow}},\ \text{FMG}_{\text{slow}},\ \text{OAL}_{\text{fast}},\ \text{MVA}_{\text{fast}},\ \text{FMG}_{\text{fast}},\ \text{CZQ}_{\text{fast}},\ \text{HP}_{\text{fast}},\ \text{VFS}_{\text{fast}},\ \text{HP}_{\text{slow}},\ \text{VFS}_{\text{slow}}\ \} \cup \mathbb{R}^2$. The first eight modes correspond to the dynamics of the aircraft incoming into the arrival from any entry waypoint (acronym), at a speed (fast or slow). The rest of the modes are holding patterns and



HP_{slow}
HP_{fast}
FMG_{slow}
OAL_{slow}
CZQ_{slow}
MVA_{slow}
MVA_{fast}
CZQ_{fast}
OAL_{fast}
FMG_{fast}
VFS_{slow}
VFS_{fast}
σ_{c_1} σ_{c_2} σ_{c_3} σ_{c_4} σ_{c_5} σ_{c_6} σ_{c_7} σ_{c_8} σ_{c_9} $\sigma_{c_{10}}$ $\sigma_{c_{11}}$ $\sigma_{c_{12}}$ $\sigma_{c_{13}}$ $\sigma_{c_{14}}$ $\sigma_{c_{15}}$ $\sigma_{c_{16}}$ $\sigma_{c_{17}}$ $\sigma_{c_{18}}$ $\sigma_{c_{19}}$ $\sigma_{c_{20}}$ $\sigma_{c_{21}}$ $\sigma_{c_{22}}$ $\sigma_{c_{23}}$ $\sigma_{c_{25}}$ $\sigma_{c_{26}}$
σ_{a_1} σ_{a_2} σ_{a_3} σ_{a_4} σ_{a_5} σ_{a_6} σ_{a_7} σ_{a_8} σ_{a_9} $\sigma_{a_{10}}$ $\sigma_{a_{11}}$ $\sigma_{a_{12}}$ $\sigma_{a_{13}}$ $\sigma_{a_{14}}$ $\sigma_{a_{15}}$

Figure 4.2: Hybrid automaton for each aircraft for the arrival shown in Figure 2.4. The transition between modes is given either by airspace use (σ_{a_p} switches, $p \in \mathbb{N}$, see for example Figure 2.4) and Air Traffic Controller commands (σ_{c_p} switches $p \in \mathbb{N}$, which we are interested in synthesizing). The duplication of modes fast / slow is done to ensure that the aircraft can only decelerate through the execution of the hybrid automaton, as for a normal arrival. This model is a subset of our former model [18, 19], adapted to the specific scenario of arrivals.

vector for spacing, respectively, at fast and slow speed. $\mathbb{X}_i = \mathbb{R}^2$ where $x_i \in \mathbb{X}_i$ represents the lateral position of the aircraft. Here, we solve a routing scheduling problem at fixed altitude, as in [18, 19, 67, 131, 33] (assuming that it is possible to separate aircraft vertically if they conflict horizontally).

- Σ_i, the set of discrete inputs, which indexes the switches from one mode to another. Figure 3.2 shows the two different types of elements in Σ_i. The first set (σ_{c_p}, where p is an integer) is generated by ATC: for example "slow down on MOD.LOCKE1 from MVA" ($\sigma_{c_{15}}$), "hold at slow speed on MOD.LOCKE1 from OAL" ($\sigma_{c_{22}}$). The second set (σ_{a_p}, where p is an integer) is generated by the airspace: for example at the end of a slow HP on MOD.LOCKE1 from CZQ, retrieve original course (σ_{a_9}).

- $\text{Init}_i \subseteq Q_i \times \mathbb{X}_i$: each aircraft is initially in a given mode $q_i \in Q_i$ at a given location $x_i \in \mathbb{X}_i$, for example, at maximum speed 50 nm east from OAL.

- $f_{q_i} : Q_i \times \mathbb{X}_i \to T\mathbb{X}_i$ is a vector field $f_\cdot(\cdot)$, where $q_i \in Q_i$ represents the current mode. Note that q_i is denoted as a subscript for brevity (for a given $q_i \in Q_i$ and $x_i \in \mathbb{X}_i$, we write $\dot{x}_i = f_{q_i}(x_i)$). For $q_i \in \{ \text{HP}_{\text{fast}}, \text{VFS}_{\text{fast}}, \text{HP}_{\text{slow}}, \text{VFS}_{\text{slow}} \}$, the dynamics $f_{q_i}(x_i)$ can be derived very easily from Figure 3.3. For the other modes, they are a set of successive speed vectors. For example, for FMG_{fast}, $f_{q_i}(x_i)$ is defined as (see Figure 2.4):

$$f_{q_i}(x_i) = \begin{cases} v_1 & \text{if } \mathbf{A}_1 x_i \le b_1 \quad \text{aircraft above the line OAL-GROAN} \\ v_2 & \text{if } \mathbf{A}_2 x_i \le b_2 \quad \text{aircraft east of the line GROAN-UPEND} \\ v_3 & \text{if } \mathbf{A}_3 x_i \le b_3 \quad \text{aircraft east of the line UPEND-SFO(D29)} \\ v_4 & \text{if } \mathbf{A}_4 x_i \le b_4 \quad \text{final approach to OAK (not shown in Figure 2.4)} \end{cases} \tag{4.3}$$

  The matrices $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, and $\mathbf{A}_4$ and the corresponding vectors b_1, b_2, b_3, b_4 encode the geometrical constraints of the problem: $v_i \in \mathbb{R}^2$ have all the same magnitude (corresponding to the fast mode), but point towards different destination waypoints.

- $\text{Dom}_i \subseteq Q_i \times \mathbb{X}_i \times \Sigma_i$ defines for each discrete mode $q_i \in Q_i$ the subset of $\mathbb{X}_i \times \Sigma_i$ for which continuous evolution is allowed. Dom can be expressed as $\text{VFS}_{\text{fast}} \times P_{\text{VFS}_{\text{fast}}} \cup \text{HP}_{\text{fast}} \times P_{\text{HP}_{\text{fast}}}$, etc. , where $P_{\text{VFS}_{\text{fast}}}$ and $P_{\text{HP}_{\text{fast}}}$, etc. are subsets of $\mathbb{R}^2$ where these maneuvers are allowed (see for example the allowed VFS zone in Figure 3.3 for the VFS, and the three HP available in LOCKE 1 in Figure 2.4 for the HP).

- $R_i : Q_i \times \mathbb{X}_i \times \Sigma_i \to 2^{Q_i \times \mathbb{X}_i}$ is a reset relation which maps a state to the sets of its possible successors. For example, if the aircraft has to be put on hold by ATC from its current flight plan at location x_i at low speed arriving from OAL, $(\text{HP}_{\text{slow}}, x_i) \in R_i(\text{OAL}_{\text{slow}}, x_i, \sigma_c)$, where σ_c is the discrete controller action which triggers this mode switch.

If one ignores for a moment the continuous part of the problem and views H_i as a finite state machine, the grammar (see, for example, [101]) accepted by H_i is easy to compute. Experimental and statistical studies of a similar grammar have been realized for another airspace at MIT ICAT by Histon and Hansman [83]. Their study and analysis of this grammar rely on a careful classification of its instances. In the present work, we are interested in synthesizing a set of "timed sentences" accepted by this grammar, which represent the ATC commands delivered to the aircraft (i.e. which we can use to generate a hybrid automaton execution). For this, we can use the definition of *hybrid time trajectory* $\tau(i)$ of aircraft i (see for example [149]) $\tau(i) = \{[\tau_l, \tau_l']\}_{l=0}^{L_i}$ (where $L_i \in \mathbb{N}$) which times the execution of H_i. The intervals $[\tau_l, \tau_l']$ represent occupancy time of a mode. The mode switches are given by $(q_i(\tau_{l+1}), x_i(\tau_{l+1})) \in R_i(q_i(\tau_l'), x_i(\tau_l'), \sigma(\tau_l'))$. The timed grammar that we are interested in synthesizing is therefore of the following form for each aircraft: $([\tau_0, \tau_0'], \sigma_0, [\tau_1, \tau_1'], \sigma_1, [\tau_2, \tau_2'], \sigma_2, \cdots)$, where in addition to the σ_p (and modes) of a usual grammar, we have the corresponding time intervals $[\tau_p, \tau_p']$. Note that equation (4.1) directly follows from this grammar.

4.3 Hybrid controller synthesis

We can now restate in mathematical terms the problem described in the introduction.

Problem 3. *Generate a procedure which takes N airborne aircraft (with the following information: current position x_i, requested entrance waypoint into the arrival airspace, range of achievable speeds), and provides:*

- *An optimal arrival schedule with route (i.e. arrival) assignment for each aircraft; where optimal means either the result of minimizing the sum $\sum_{i=1}^{N} t_i$ of arrival times $\{t_i\}_{i\in\{1,\cdots,N\}}$ of all aircraft, or the result of maximizing the minimal separation of two successive aircraft $\min_{i<j,(i,j)\in\{1,\cdots,N\}^2} |t_i - t_j|$ at the destination airport without putting any aircraft on hold.*
- *The set of mode switches $\{\sigma_k\}_{k\in\{1,\cdots,K_i\}}$ to achieve this schedule with corresponding hybrid time trajectory for each aircraft i: $([\tau_0, \tau_0'], \sigma_0, [\tau_1, \tau_1'], \sigma_1, [\tau_2, \tau_2'], \sigma_2, \cdots)$.*

We now present a resolution algorithm for the problem above, for a set of aircraft (indexed by i), given a set of possible arrivals (indexed by j).

We now go over all steps of the algorithm and explain what they achieve.

Scheduling / routing resolution algorithm

Input: Position, flight plan of the aircraft.
Output: Route and maneuver assignment for each aircraft.

```
      for all aircraft i=1:N
 1      $\mathsf{wp}_i$ := entrance wp into airspace
        for all arrivals j starting from $\mathsf{wp}_i$
 2        Compute shortest distance $\tilde{s}_{ij}$ to destination
 3        Compute buffered distance $\tilde{b}_{ij}$ to destination
 4        $p_{ij}$ := number of holding patterns available for aircraft $i$ in arrival $j$
 5        Feasible $j$-arrival times $\mathsf{Feas}_{ij} := \bigcup_{k=1}^{p_{ij}}[\frac{\tilde{s}_{ij}}{\text{max speed } i} + kT_{\mathsf{hp}}, \frac{\tilde{b}_{ij}}{\text{min speed } i} + kT_{\mathsf{hp}}]$
        end
 6      Compute the set $S_i$ of possible arrival times of aircraft $i$: $S_i = \bigcup_j \mathsf{Feas}_{ij}$
 7      Represent $S_i$ as a disjoint union: $S_i := \bigcup_{k=1}^{n_i}[a_{i,k}, b_{i,k}]$, where $b_{i,k} < a_{i,k+1}$
 8      if MILP($\Delta T$,$\{S_i\}$) is not feasible
          Return not feasible
        else
 9        For every $t_i$, identify arrival route and number of holding patterns
10        Compute switching time sequence $\{\tau_1^i, \cdots, \tau_N^i, t^i\}$
        end
      end
```

1. At time t, get the requested entrance waypoint wp_i of aircraft i into the arrival area, as well as predicted arrival time $t_i^{\text{predicted}}$ at this way point.

2. For the current arrival j, compute the shortest distance $\tilde{s}_{ij}$ from wp_i to the destination airport: $\tilde{s}_{ij} = \sum d(\mathsf{wp}_k, \mathsf{wp}_{k+1})$ for all waypoints wp_k between the entrance waypoint wp_i and the destination airport in arrival j. The computation of $d(\mathsf{wp}_k, \mathsf{wp}_{k+1})$, defined in section 4.2.1 and also shown in Figure 3.3, is performed by calculating the great circle distance between wp_k and wp_{k+1}. Note that the positions x_i are therefore given as (latitude, longitude).

3. For the current arrival j, compute the buffered distance $\tilde{s}_{ij}$ from wp_i to the destination airport: $\tilde{b}_{ij} = \sum_{\text{buffered}} d(\mathsf{wp}_k, \mathsf{wp}_{k+1}) \left(\frac{1}{\cos\psi} - 1\right)$. The sum only ranges over the set of segments for which it is possible to do VFS.

4. p_{ij} is the number of allowed HP for aircraft i in arrival j (it accounts for example for the remaining fuel or congestion of the airspaces). This number could thus range

from zero (if the HP locations are totally saturated) to a large number, limited by the fuel autonomy of the aircraft (for a duration of sometimes one hour or more, in the current system). Again, one of the applications of this study is precisely to compute the best available spacing of the aircraft without HP, in order to alleviate their use.

5. $\mathsf{Feas}_{ij} = \bigcup_{k=1}^{p_{ij}} [\frac{\tilde{s}_{ij}}{\mathsf{max\ speed}i} + kT_{\mathsf{hp}}, \frac{\tilde{b}_{ij}}{\mathsf{min\ speed}i} + kT_{\mathsf{hp}}]$ is the union of all possible arrival times at the destination when using arrival j. Feas_{ij} represents the set of possible arrival times for aircraft i using arrival j with the allowed number of HP in that arrival. It is in general not convex.

6. $S_i = \bigcup_j \mathsf{Feas}_{ij}$ represents the set of all possible arrival times for aircraft i using any of the available arrival routes. It is generally not convex either.

7. The set $S_i \subset \mathbb{R}$ is a union of intervals, possibly overlapping. This set can be rewritten in polynomial time as $S_i := \bigcup_{k=1}^{n_i} [a_{i,k}, b_{i,k}]$, where $b_{i,k} < a_{i,k+1}$, eliminating any redundancy in the expression of the constraints. This last expression is fundamental for the next step, which consists of writing the constraints in terms of decision variables.

8. We want to minimize the sum of arrival of all aircraft, such that the aircraft arrive in their achievable time intervals and are separated by at least ΔT:

$$\begin{array}{lll} \textbf{Minimize:} & \sum_{i=1}^{N} t_i & \\ \textbf{Subject to:} & t_i \in S_i = \bigcup_{k=1}^{n_i} [a_{i,k}, b_{i,k}] & \forall i \in \{1, \cdots, N\} \\ & |t_i - t_j| \geq \Delta T & \forall (i,j) \in \{1, \cdots, N\}^2 \text{ s.t. } i < j \end{array} \tag{4.4}$$

We now transform this problem into a MILP. Let us pick two constants C and D such that $D \geq \max_{i=1}^{N} \max\{a_{i,n_i} - a_{i,1}, b_{i,n_i} - b_{i,1}\}$, and $C \geq 2(\max_{j=1}^{N} b_{j,n_j} - \min_{i=1}^{N} a_{i,1})$, and introduce decision variables c_{ij} and d_{ij}. We rewrite the non-convex optimization problem (4.4) as a MILP, which we denote MILP$(\Delta T, \{S_i\})$:

$$\begin{array}{lll}
\textbf{Minimize:} & \sum_{i=1}^{N} t_i & \\
\textbf{Subject to:} & t_i \geq a_{i,1} & \forall i \in \{1, \cdots, N\} \\
 & t_i \leq b_{i,n_i} & \forall i \in \{1, \cdots, N\} \\
 & t_i \geq a_{i,k+1} - D d_{ik} & \forall i \in \{1, \cdots, N\}, \forall k \in \{1, \cdots, n_i - 1\} \\
 & t_i \leq b_{i,k} + D(1 - d_{ik}) & \forall i \in \{1, \cdots, N\}, \forall k \in \{1, \cdots, n_i - 1\} \\
 & d_{ik} \in \{0, 1\} & \forall i \in \{1, \cdots, N\}, \forall k \in \{1, \cdots, n_i - 1\} \\
 & t_i - t_j \geq \Delta T - C c_{ij} & \forall (i,j) \in \{1, \cdots, N\}^2, \text{ s.t. } i > j \\
 & t_i - t_j \leq -\Delta T + C(1 - c_{ij}) & \forall (i,j) \in \{1, \cdots, N\}^2, \text{ s.t. } i > j \\
 & c_{ij} \in \{0, 1\} & \forall (i,j) \in \{1, \cdots, N\}^2, \text{ s.t. } i > j
\end{array} \tag{4.5}$$

The handiness of MILP formulations for non-convex optimization problems such as (4.4) has been extensively explored and used successfully by Richards, How, Feron, Schouwenaars, and coauthors in [136, 135, 137, 134]. They use it mainly to perform collision and obstacle avoidance under discretized linear dynamics. In this work, we take advantage of the same formulation to express non-convex constraints of two types. The first type is "time collision avoidance": $|t_i - t_j| \geq \Delta T$, which enables metering of the flow. The second type is non-convex feasibility constraints $t_i \in \bigcup_{k=1}^{n_i} [a_{i,k}, b_{i,k}]$, which encapsulate the airspace structure and the aircraft capacities.

9. For every t_i, $i \in \{1, \cdots, N\}$, $\exists! \nu \in \{1, \cdots, n_i\}$, such that $t_i \in [a_{i,\nu}, b_{i,\nu}]$. For this $[a_{i,\nu}, b_{i,\nu}]$, we can reconstruct at least one route (i.e. choice of arrival and waypoint sequence $(\mathsf{wp}_1, \cdots, \mathsf{wp}_{n(\nu, a_{i,\nu}, b_{i,\nu})})$) and retrieve the number p_{ij} of HP which achieves this t_i. The arrival route j, and the number p_{ij} of holding patterns can be retrieved from steps 2-3-4, by labeling the feasible arrival time intervals (i.e. by storing for each arrival interval portion, the arrival routes which achieve it, as well as the number of corresponding HP and VFS). Sometimes, more than one solution is available (if for example the achievable arrival time intervals using two different arrivals overlap).

10. Construction of the timed grammar $([\tau_0, \tau_0'], \sigma_0, [\tau_1, \tau_1'], \sigma_1, [\tau_2, \tau_2'], \sigma_2, \cdots)$. We restrict, $k \in \mathbb{N}$, $\tau_k' - \tau_k > 0$ (the system has to stay in each mode for a nonzero amount of time).[‡] We thus have $\tau_k' = \tau_{k+1}$ for all relevant indices k. Therefore, we only need

[‡]Instantaneous switches with zero occupancy times are relevant for other physical systems modeled as hybrid automata, but are not included in the current problem.

to compute the τ_k. Two possibilities exist:

- If $t_i - t_i^{\text{predicted}} - p_{ij}T_{\mathsf{hp}} \in \left[\frac{\tilde{s}_{ij}}{\overline{v}_i}, \frac{\tilde{s}_{ij}}{\underline{v}_i}\right]$, there is no need for use of VFS. The inclusion above means that a simple solution where aircraft i switches from upper speed $\overline{v}_i$ to lower speed $\underline{v}_i$ between wp_1 and $\mathsf{wp}_{n(\nu,a_\nu^i,b_\nu^i)}$ works.[§] The aircraft trajectory should include a portion at upper speed of duration

$$T_{\text{fast}} = \frac{1}{\overline{v}_i - \underline{v}_i}\left[\tilde{s}_{ij} - \underline{v}_i(t_i - t_i^{\text{predicted}} - p_{ij}T_{\mathsf{hp}})\right]$$

  eventually interrupted by k of the p_{ij} holding patterns (for a cumulated duration of kT_{hp}), and a portion at lower speed of duration

$$T_{\text{slow}} = \frac{1}{\overline{v}_i - \underline{v}_i}\left[-\tilde{s}_{ij} + \overline{v}_i(t_i - t_i^{\text{predicted}} - p_{ij}T_{\mathsf{hp}})\right]$$

  eventually interrupted by the rest of the holding patterns (for a cumulated duration of $(p_{ij} - k)T_{\mathsf{hp}}$). Note again that the current speed does not matter for the holding patterns, since they are prescribed in "time to lose" (lose 3 minutes at $\underline{v}_i$ is the same as lose 3 minutes at $\overline{v}_i$). The hybrid trajectory of aircraft i is then computed according to the diagram in Figure 4.3: the interval $[t_i^{\text{predicted}}, t_i]$ is divided in two: $[t_i^{\text{predicted}}, t_i^{\text{slow down}}]$ and $[t_i^{\text{slow down}}, t_i]$, such that $t_i^{\text{slow down}} - t_i^{\text{predicted}} = T_{\text{fast}} + kT_{\mathsf{hp}}$ and $t_i - t_i^{\text{slow down}} = T_{\text{slow}} + (p_{ij} - k)T_{\mathsf{hp}}$. Here, $k \leq p_{ij}$ is the number of HP realized at high speed and $p_{ij} - k$ is the number of HP realized at low speed.

- If $t_i - t_i^{\text{predicted}} - p_{ij}T_{\mathsf{hp}} \in \left[\frac{\tilde{s}_{ij}}{\underline{v}_i}, \frac{\tilde{b}_{ij}}{\underline{v}_i}\right]$, flying at minimal speed $\underline{v}_i$ for the complete arrival is not sufficient. The aircraft has to switch from upper speed $\overline{v}_i$ to lower speed $\underline{v}_i$ upon entrance in the arrival area, and will then have to use VFS for a portion of the path, given by:

$$T_{\text{VFS slow}} = \frac{t_i - t_i^{\text{predicted}}}{\tilde{b}_{ij} - \tilde{s}_{ij}}\left[-\tilde{s}_{ij} + \underline{v}_i(t_i - t_i^{\text{predicted}} - p_{ij}T_{\mathsf{hp}})\right]$$

[§]Of course, for other reasons such as weather or traffic, the ATC might decide to achieve the same arrival time t_i by a VFS instead of a speed change.

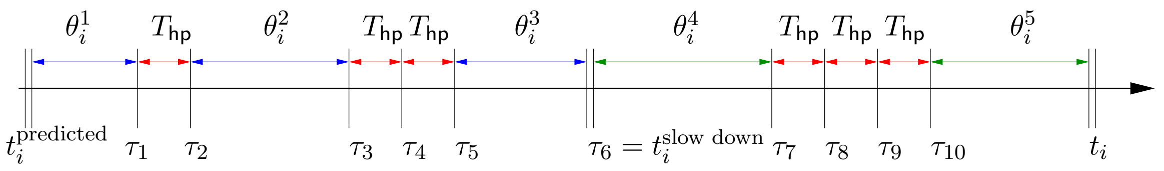

θ_i^1
T_{hp}
θ_i^2
T_{hp} T_{hp}
θ_i^3
θ_i^4
T_{hp} T_{hp} T_{hp}
θ_i^5
$t_i^{\text{predicted}}$
τ_1 τ_2 τ_3 τ_4 τ_5
$\tau_6 = t_i^{\text{slow down}}$
τ_7 τ_8 τ_9 τ_{10}
t_i

Figure 4.3: Example of hybrid trajectory $\{t_i^{\text{predicted}}, \tau_1, \tau_2, \cdots \tau_9, \tau_{10}, t_i\}$ of the hybrid automaton for aircraft i. τ_1, τ_3, τ_4, τ_7, τ_8, τ_9 are determined by geometry (i.e. the switch from F to HP occurs upon entry on the holding pattern zone: see Figure 2.4). θ_i^1, θ_i^2, θ_i^3 are determined by the τ_i and the equation of $T_{\text{fast}} = \theta_i^1 + \theta_i^2 + \theta_i^3$ (see definition of T_{fast} in step 10 of the algorithm). Here $p_{ij} = 3$ since there are three holding patterns. This enables the computation of τ_6, the switching time from upper speed $\overline{v}_i$ to lower speed $\underline{v}_i$. For τ_i, $i \geq 7$, the same construction applies. For the case of VFS, the construction is identical, except that the $\underline{v}_i$ portions and the VFS portions might alternate depending on spatial availability of VFS along certain routes.

and be at low speed for the remaining portion

$$T_{\text{slow}} = \frac{t_i - t_i^{\text{predicted}}}{\tilde{b}_{ij} - \tilde{s}_{ij}} \left[\tilde{b}_{ij} - \underline{v}_i (t_i - t_i^{\text{predicted}} - p_{ij} T_{\mathsf{hp}}) \right]$$

As in the previous case, both portions can be interrupted by HPs, if $p_{ij} > 0$. The hybrid trajectory can then again be computed using the diagram shown in Figure 4.3.

4.4 Implementation and simulations

We now describe a possible implementation of the algorithm derived in the previous section. This implementation will be used in the next chapter for numerical tests and experiments on running time.

4.4.1 Implementation using ETMS data

For this implementation, the user interface of the code is written in MATLAB, which reads ETMS data and translates it into AMPL code (steps 1-7 of the algorithm). The MILP is coded and solved in CPLEX (step 8), interfaced by the modeling language AMPL (see for example [73] for a complete description of the AMPL / CPLEX language). The results of the optimization are read in MATLAB and transformed into ATC command lists and ETMS format data (steps 9-10), readable directly by FACET, a software developed by

NASA Ames [39]. MATLAB is run under UNIX on a SOLARIS 8 workstation (1GB of RAM). The CPU runtimes presented in this section and the next chapter are obtained with this platform.

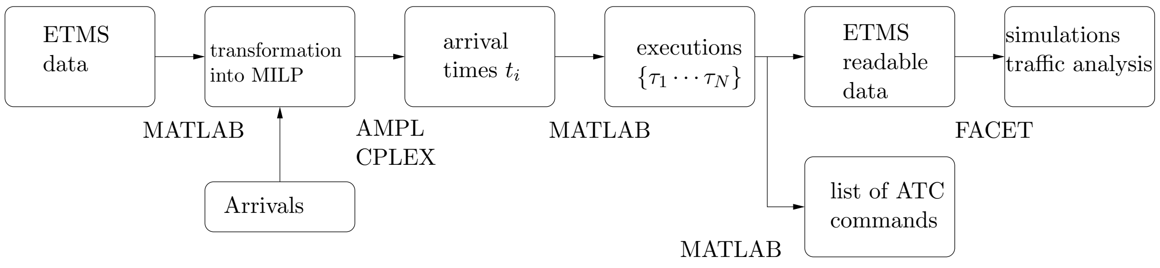


Figure 4.4: Block diagram of the implementation of the algorithm.

Even though this implementation works in real time for a reasonable number of aircraft in most of the realistic cases (flow almost metered, i.e. such that few changes in order of arrival and flight plan modifications are needed to achieve an acceptable spacing between the aircraft), it is important to show the limits of this approach as well, i.e. identify a threshold number of aircraft above which the CPU time required by the method becomes too large to be realizable online. We expect the method to work well in cases where heuristics are available to rule out large number of possible integer solutions. For example when $a_{i,1} > b_{j,n_j}$ holds for a large number of aircraft, it is likely that the resulting partial order (aircraft i has to arrive after aircraft j) will help CPLEX to solve the problem faster. Figure 4.5, left shows an example for 10 aircraft, in which such heuristic is not obvious.

In the next chapter, we will show that in the case where partial order is not available as a decision heuristic, the computational time can become extremely large. This phenomenon is sometimes referred to as exponential computational time or explosion of the combinatorial state space.

4.4.2 Simulation example: robustness of merging flows

We show a possible use of this algorithm to characterize flow stability of merging traffic. A very common phenomenon in congested areas (as the Oakland center for example) is delay of short flights because of incoming transoceanic flights. In general, in the presence of flights incoming from all destinations (Europe, Asia, South America), ATC tends to delay

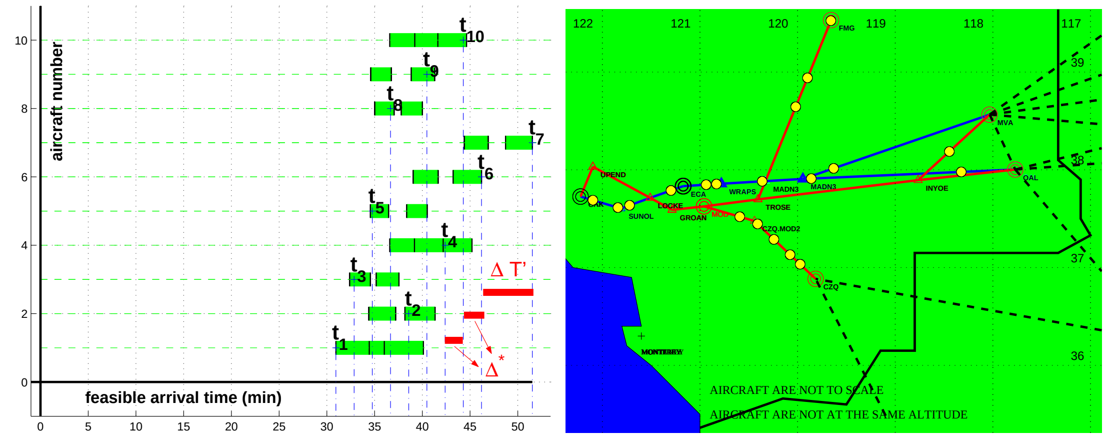


Figure 4.5: **Left:** Example of solution of the MILP (4.6) for 10 aircraft, with two arrivals to the Oakland airport available (LOCKE1 and MADWIN3, shown in the right Figure), and without holding patterns. For aircraft i, the two horizontal segments represent the feasible arrival times: $[a_{i,1}, b_{i,1}]$ and $[a_{i,2}, b_{i,2}]$. In some cases, they overlap ($i = 1, 4, 10$) or have empty intersection ($i = 2, 3, \cdots$). The largest Δ solving (4.6) is shown and called Δ^*. In some cases, spacing between two aircraft can be greater than Δ^* (for example aircraft 6 and 7). This would not happen for the solution of the MILP (4.5) where the goal is to minimize the makespan. **Right:** GUI interface of our implementation: arrivals LOCKE1 and MADWIN3 to the Oakland airport.

or cancel short flights (for example from L.A., Seattle, Vancouver) if the capacity of the arrival airports is limited, because the international flights are already airborne and might not have as much freedom in delays (for fuel reasons). We would like to show that our algorithm enables quantification of the influence of additional traffic into merging traffic. We ask the following question:

Problem 4. *Given a sequence of m incoming aircraft into a single airport, by how much does an additional set of n aircraft reduce the available spacing of the original sequence ?*

In other words, we want to compute the Δ^* resulting from an additional n aircraft when solving the following MILP:

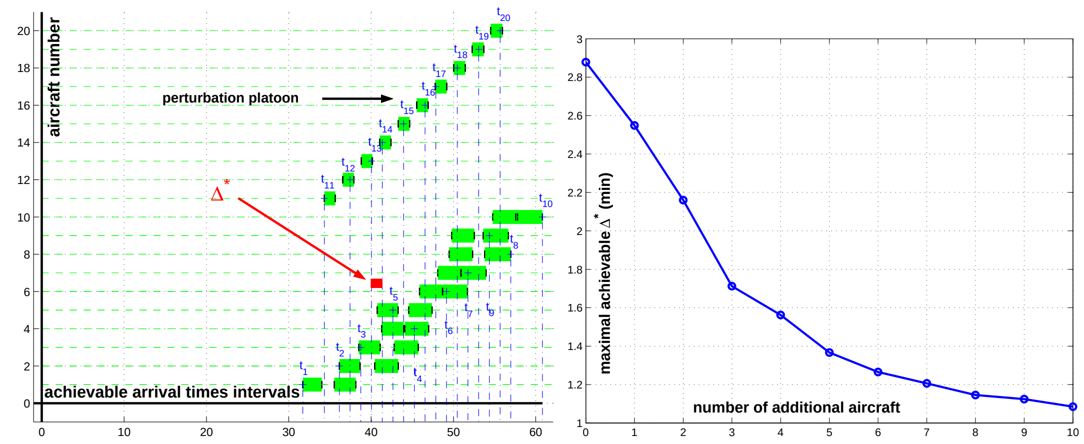


Figure 4.6: **Left:** Feasible arrival times intervals for the perturbed flow. Aircraft 1 to 10 are incoming into LOCKE 1 and MADWIN 3 through OAL, MVA and FMG. Aircraft 11 to k (where $k \in \{11, \cdots, 20\}$ are incoming into LOCKE 1 through CZQ. their maneuvering availability is very small (short intervals). The Δ^* for this run is shown as well. **Right:** Variation of Δ^* with k. As expected, the more the flow is perturbed (large k), the smaller Δ^* becomes.

$$\begin{array}{lll}
\textbf{Maximize:} & \Delta & \\
\textbf{Subject to:} & t_i \geq a_{i,1} & \forall i \in \{1, \cdots, N\} \\
& t_i \leq b_{i,n_i} & \forall i \in \{1, \cdots, N\} \\
& t_i \geq a_{i,k+1} - D d_{ik} & \forall i \in \{1, \cdots, N\}, \forall k \in \{1, \cdots, n_i - 1\} \\
& t_i \leq b_{i,k} + D(1 - d_{ik}) & \forall i \in \{1, \cdots, N\}, \forall k \in \{1, \cdots, n_i - 1\} \\
& d_{ik} \in \{0, 1\} & \forall i \in \{1, \cdots, N\}, \forall k \in \{1, \cdots, n_i - 1\} \\
& t_i - t_j \geq \Delta - C c_{ij} & \forall (i,j) \in \{1, \cdots, N\}^2, \text{ s.t. } i > j \\
& t_i - t_j \leq -\Delta + C(1 - c_{ij}) & \forall (i,j) \in \{1, \cdots, N\}^2, \text{ s.t. } i > j \\
& c_{ij} \in \{0, 1\} & \forall (i,j) \in \{1, \cdots, N\}^2, \text{ s.t. } i > j
\end{array} \tag{4.6}$$

We generate the following scenario (see Figure 4.5 right for the map and Figure 4.6 left for the schedule): m aircraft are scheduled to come into Oakland through the two arrival LOCKE 1 and MADWIN 3, entering through the waypoints OAL, MVA and FMG. An additional n aircraft are fed into LOCKE 1 through CZQ, with very little available maneuvering freedom (short single intervals $[a_{i,1}, b_{i,1}]$). Figure 4.6 shows a numerical example of achievable schedule for this scenario, with $m = n = 10$. Figure 4.6 (right) shows the averaged results of 300 simulations of perturbed merging traffic. We perturb $m = 10$ aircraft by n aircraft, where

n increases from 1 to 10. We compute the maximal available spacing Δ^* for the set of $m+n$ aircraft. We see that the additional 10 aircraft reduce the spacing by a factor almost three. In this case, we thus see that holding patterns will be necessary, since in general, one aircraft a minute is too high a frequency for a single track airport like Oakland (because of the vortex wake hazard). More generally, this algorithm enables the prediction of a feasible spacing (and the set of instructions to achieve it), and enables the ATC to take appropriate decisions given the spacing available at the destination airport.

Chapter 5

Combinatorial optimization algorithms for task planning

Chapter 4 ended with a formulation of ATC maneuver assignment problems as a *mixed integer linear program* (MILP). Preliminary implementations in the architecture presented in Chapter 4 have suggested that the combinatorial explosion of the state space makes this problem intractable for more than 20 aircraft in the general case, without a proper algorithmic treatment [27]. The goal of this chapter is to show evidence of this fact and to develop proper algorithms to circumvent this difficulty.

This chapter derives algorithms to solve subproblems of the general routing sequencing problem posed in Chapter 4. We believe that the general form of the problem derived in Chapter 4 is NP-complete, i.e. that under the common assumption $P \neq NP$ (see for example [76, 153]), it is not possible to derive a deterministic algorithm which solves this problem in polynomial time. In subcases which we investigate in this chapter, we will construct algorithms, which either (i) solve a subproblem exactly in polynomial time (in cases which we identify and for which it is possible); (ii) are approximation algorithms, i.e. they approximate the solution in polynomial time, within worst case guaranteed bounds which we compute.

Section 5.1 investigates the case in which the number of arrival intervals is one for each aircraft. This corresponds to the physical situation in which only one arrival is available to each aircraft, and ATC does not want to put the aircraft on a holding pattern. Several questions can be posed for this scenario: (i) Is a given separation at the destination airport feasible? (ii) What is the largest possible separation at the airport which ATC can guarantee given the constraints? Question (i) turns out to be a simple polynomial time solvable scheduling problem; question (ii) can be reduced to a polynomial time solvable problem via a bisection algorithm for which we derive an explicit stopping criterion. We implement our algorithm and show numerical evidence of its superiority over the leading commercial software to solve MILPs.

Section 5.2 investigates the case in which only one arrival is open to each aircraft, but they can fly a large number of holding patterns (which can be bounded). Since aircraft can be held for as long as needed, feasibility is not a problem anymore, but sum of the delays, or arrival times of the last aircraft (makespan) are a problem. We derive approximation algorithms which are polynomial time, and are proved to converge to a value with guaranteed worst case bounds from the optimum.

5.1 A polynomial time algorithm for a one interval MILP

5.1.1 Problem formulation

We consider N aircraft converging to a single airport. Aircraft are indexed by $i \in \{1, \cdots, N\}$. As a result of ATC actuation, the set of arrival times for aircraft i is a union of intervals called $S_i := \bigcup_{k=1}^{n_i} [a_{i,k}, b_{i,k}]$. For each i, n_i is the number of intervals of time in which aircraft i can arrive. These intervals encode a set of maneuvers which are relatively easy to implement for ATC, as was shown in the previous chapter. A key problem for an Air Traffic Controller is to estimate the largest time spacing achievable with this set of maneuvers, in order to determine if the use of large maneuvers is necessary (significant flight plan changes or large numbers of holding patterns at the direct vicinity of airports):

Problem 5. *What is the maximal separation we can guarantee at the airport, without putting the aircraft on a holding pattern?*

Note that the solution to this problem also provides the safety buffer – or margin, which is the difference between the maximal spacing that ATC can guarantee and the spacing required by the airport.

As was derived in the previous chapter, the general problem translates into the following optimization problem: given $\{S_i\}_{i\in\{1,\cdots,N\}}$, we want to find a N-tuplet $(t_1,\cdots,t_N)\in\prod_{i=1}^{N}S_i$ of feasible arrival times for the N aircraft maximizing the minimum time separation Δ between any two aircraft:

$$\begin{array}{lll} \textbf{Maximize:} & \Delta & \\ \textbf{Subject to:} & t_i \in \bigcup_{k=1}^{n_i}[a_{i,k},b_{i,k}] & \forall i\in\{1,\cdots,N\} \\ & |t_i-t_j|\geq\Delta & \forall(i,j)\in\{1,\cdots,N\}^2 \end{array} \tag{5.1}$$

For the present Section, in which we only allow one arrival and prohibit holding patterns, we focus on the one interval case ($n_i=1$ for all i), and show that this problem is polynomial-time solvable. As was shown in the previous chapter, for the one interval case, the problem can be rewritten in the following form:

$$\begin{array}{lll} \textbf{Maximize:} & \Delta & \\ \textbf{Subject to:} & t_i\geq a_{i,1} & \forall i\in\{1,\cdots,N\} \\ & t_i\leq b_{i,1} & \forall i\in\{1,\cdots,N\} \\ & t_i-t_j\geq\Delta-Cc_{ij} & \forall(i,j)\in\{1,\cdots,N\}^2,\ \text{s.t. } i>j \\ & t_i-t_j\leq-\Delta+C(1-c_{ij}) & \forall(i,j)\in\{1,\cdots,N\}^2,\ \text{s.t. } i>j \\ & c_{ij}\in\{0,1\} & \forall(i,j)\in\{1,\cdots,N\}^2,\ \text{s.t. } i>j \end{array} \tag{5.2}$$

5.1.2 Closed form solution for fixed arrival order

In order to solve (5.2), we need to identify which constraints are active in the optimal solution: in the present case, they determine the optimum of the problem analytically. For this, we fix the order of arrival at an arbitrary setting, which enables us to find the active constraints. The result obtained for this arbitrary order will be used for solving the general problem (5.2). Theorem 1 is thus a building block of the full algorithm, which is presented in Section 5.1.4. Lemma 1 below is used to prove Theorem 1.

Thus, assume for this section only that the order of arrival of the aircraft is known a priori. Without loss of generality, assume that the aircraft have been labeled in this order (aircraft 1 arrives first, aircraft 2 arrives second, $\cdots$). Then, problem (5.2) reduces to

$$\begin{array}{lll} \textbf{Maximize:} & \Delta & \\ \textbf{Subject to:} & t_i - t_{i-1} \geq \Delta & i \in \{2, \cdots, N\} \\ & t_i \geq a_{i,1} & i \in \{1, \cdots, N\} \\ & t_i \leq b_{i,1} & i \in \{1, \cdots, N\} \end{array} \tag{5.3}$$

Lemma 1. *The optimal solution Δ of (5.3) satisfies $\Delta \leq m$, where*

$$m = \min_{(i,j) \in \{1, \cdots, N\}^2,\ i<j} \left(\frac{b_{j,1} - a_{i,1}}{j - i} \right) \tag{5.4}$$

Proof — Call $(k, l) = \arg\min_{(i,j) \in \{1, \cdots, N\}^2,\ i<j} \left(\frac{b_{j,1} - a_{i,1}}{j-i} \right)$, i.e. one pair of aircraft such that $\left(\frac{b_{l,1} - a_{k,1}}{l-k} \right) = m$ and $k < l$. Suppose that $\Delta > m$. Then $(l-k)m < (l-k)\Delta \leq b_{l,1} - a_{k,1} = (l-k)m$. The first inequality is by assumption. The second by definition of Δ: the best spacing between aircraft k and l is bounded by the mean. The last equality is by definition of m. This inequality cannot be true. Therefore $\Delta \leq m$. □

Theorem 1. *It is possible to construct a sequence $(t_1, \cdots, t_N)$ such that $(m, t_1, \cdots, t_N)$ satisfies (5.3), which proves that*

$$\Delta = \min_{(i,j) \in \{1, \cdots, N\}^2,\ i<j} \left(\frac{b_{j,1} - a_{i,1}}{j - i} \right) \tag{5.5}$$

Proof — We prove this theorem by induction on N. For $N = 2$, the solution is trivial. For $N = 3$, we see that the following solution achieves the optimal:

$$\begin{aligned}
t_1 &= a_{1,1} \\
t_2 &= (b_{3,1} + a_{1,1})/2 && \text{if } (b_{3,1} + a_{1,1})/2 \in [a_{2,1}, b_{2,1}] \\
&= a_{2,1} && \text{if } (b_{3,1} + a_{1,1})/2 < a_{2,1} \\
&= b_{2,1} && \text{if } (b_{3,1} + a_{1,1})/2 > b_{2,1} \\
t_3 &= b_{3,1}
\end{aligned} \tag{5.6}$$

Suppose now that we have proved the property up to N, i.e. that (5.5) is the solution of (5.3) for N aircraft. We now prove that (5.5) is the solution of (5.3) for $N+1$ aircraft. Call $m(k,l)$ the following quantity, with k and l both in $\{1, \cdots, N+1\}$, and $k < l$:

$$m(k,l) := \min_{(i,j)\in\{k,\cdots,l\}^2,\ i<j} \left(\frac{b_{j,1} - a_{i,1}}{j-i}\right) \tag{5.7}$$

We obviously have $m(1, N+1) = \min\left\{m(1,N), \min_{i\in\{1,\cdots,N\}}\left(\frac{b_{N+1,1}-a_{i,1}}{N+1-i}\right)\right\}$. We now investigate which of these terms achieves the min for $m(1, N+1)$.

<u>Case 1:</u> $m(1, N+1) = m(1, N)$. We thus have $m(1,N) \leq \frac{b_{N+1,1}-a_{i,1}}{N+1-i}$ for all $i \leq N$ (i.e. the arrival interval of aircraft $N+1$ does not decrease the overall minimum). Take $t_{N+1} = b_{N+1,1}$. Now call $b'_{N,1} = \min\{b_{N+1,1} - m(1,N), b_{N,1}\}$.

- If $b'_{N,1} = b_{N,1}$, we have successfully constructed $(t_1, \cdots, t_{N+1})$: t_{N+1} satisfies $t_{N+1} - t_N \geq m(1, N+1) = m(1,N)$, and for all $(i,j) \in \{1, \cdots, N\}^2$ such that $i < j$, $t_j - t_i \geq m(1,N) = m(1,N+1)$.

- If $b'_{N,1} = b_{N+1,1} - m(1,N)$. Let us rename every other $b_{i,1}$ as $b'_{i,1}$, for $i < N$, for notational ease. Now define $\underline{m} = \min_{i<N}\left(\frac{b'_{N,1}-a_{i,1}}{N-i}\right)$. Now

$$\underline{m} = \min_{i<N}\left(\frac{b'_{N,1} - a_{i,1}}{N-i}\right) = \min_{i<N}\left(\frac{b_{N+1,1} - m(1,N) - a_{i,1}}{N-i}\right) \tag{5.8}$$

  Thus

$$\underline{m} = \min_{i<N}\left(\frac{b_{N+1,1} - a_{i,1}}{N+1-i}\frac{N+1-i}{N-i} - \frac{m(1,N)}{N-i}\right) \geq m(1,N)\min_{i<N}\left(\frac{N+1-i}{N-i} - \frac{1}{N-i}\right) \tag{5.9}$$

which proves $\underline{m} \geq m(1,N)$, and therefore the change in $b_{N,1}$ into $b'_{N,1} = b_{N+1,1} - m(1,N)$ does not decrease (5.7). We now have successfully constructed $(t_1, \cdots, t_{N+1})$: $t_{N+1} - t_N \geq m(1,N+1) = m(1,N)$, and for the N-tuplet $(t_1, \cdots, t_N)$, we just proved that $\underline{m} \geq m(1,N)$ is achievable.

Case 2: $m(1,N+1) = \min_{i<N+1} \left(\frac{b_{N+1,1} - a_{i,1}}{N+1-i} \right) < m(1,N)$. Obviously, adding a new aircraft has restricted the available spacing of the others: $\exists l \in \{1, \cdots, N\}$ such that $m(l,N+1) = m(1,N+1) < m(1,N)$.

- If $l = 1$, $m(1,N+1) = \frac{b_{N+1,1} - a_{1,1}}{N}$. The evenly spaced solution is the optimal solution. Let us show it is feasible, i.e. $\forall j \in \{2, \cdots, N\}$, $a_{1,1} + \frac{j-1}{N}(b_{N+1,1} - a_{1,1}) \in [a_{j,1}, b_{j,1}]$. If this were not the case, then, without loss of generality[*], we would have $a_{1,1} + \frac{j_0-1}{N}(b_{N+1,1} - a_{1,1}) > b_1^{j_0}$ for a particular j_0, which would mean $\frac{b_{j_0,1} - a_{1,1}}{j_0 - 1} < \frac{b_{N+1,1} - a_{1,1}}{N}$, which is by assumption wrong.

- If $l \geq 2$, we can use the symmetry of the problem to flip it (the $a_{i,1}$ and $b_{i,1}$ are inverted)[†], and we are now in case 1.

In case 1 and case 2, we were able to construct a solution with spacing $m(1,N)$ for every N. By Lemma 1, $m(1,N)$ is the best possible Δ which we can achieve, and equation (5.5) follows. $\square$

5.1.3 Transformation of the feasibility problem into a scheduling problem

The previous section solved for the maximal available spacing in the case of fixed order of arrival. In the current ATC system, if one interprets $a_{i,1}$ as the nominal time of arrival of aircraft i, and $(a_{i,1}, b_{i,1}]$ as the set of possible arrival times with delays, equation (5.5) predicts the best available spacing without altering the order of arrival (first come first served policy). Despite the fact that it is used almost all the time in practice, this policy is of course not optimal.[‡] We therefore now allow to change the order of arrival of the aircraft.

[*]Since the problem is symmetric.

[†]We treat first the subset $\{l \cdots N+1\}$ of aircraft (which we can do using the induction assumption, since $l > 1$, and thus $(N+1-l < N)$). We then complete with the set $\{1 \cdots, l-1\}$, but this time we will be in case 1 because the argmin of (5.5) is obtained in the set $\{l \cdots N+1\}$.

[‡]Take for example $a_{1,1} = 00:00:00$, $a_{2,1} = 00:00:30$, $b_{1,1} = 00:03:00$, $b_{2,1} = 00:01:30$, $n_1 = n_2 = 1$.

We show that the feasibility problem of (5.2) can be reduced to a single processor scheduling problem with release times and deadlines [76], known to be solvable in polynomial time [12, 49, 50, 48, 51]. The two following problems are equivalent:

Problem 6. *Let Δ be a given positive number (separation). Given a set of N intervals $[a_{i,1}, b_{i,1}]$, $i \in \{1, \cdots, N\}$, find a set $\{t_i\}_{i\in\{1,\cdots,N\}}$ such that $\forall i \in \{1, \cdots, N\}$, $t_i \in [a_{i,1}, b_{i,1}]$ and $\forall (i,j) \in \{1, \cdots, N\}^2$, s.t. $i > j$, $|t_i - t_j| \geq \Delta$.*

Problem 7. *Let D be a given positive number (processing time). Let $I = \{1, \cdots, N\}$ be a set of jobs. Let each job have a release time r_i and a deadline d_i. Find an ordering $\{t_i\}_{i\in I}$ such that $t_i \geq r_i$ (job i cannot start before the release time r_i), $t_i + D \leq d_i$, (job i has to be processed before the deadline d_i), $t_i < t_j \Longrightarrow t_j \geq t_i + D$ (two jobs cannot be processed simultaneously).*

Solving Problem 7 for a given set of $[r_i, d_i]$ and D, is thus equivalent to finding a feasible solution to our original problem, with $a_{i,1} = r_i$ and $b_{i,1} + D = d_i$. Problem 7 is a known problem in scheduling, for which polynomial algorithms have been developed [12, 49, 50, 48, 51]. We will use a slight modification of Carlier's algorithm [49] to solve the feasibility problem (Problem 7). This algorithm is presented in Appendix B.

5.1.4 Algorithm

We call Carlier the algorithm presented in Appendix B. Carlier solves Problem 7 above. We will use the following notation: Carlier(r_i, d_i, D) represents the ordering obtained with release times r_i, deadlines d_i and processing time D if such a schedule exists.

Theorem 2. *Assume that $n_i = 1$ for all $i \in \{1, \cdots, N\}$, $a_{i,1} \in \mathbb{N}$, $b_{i,1} \in \mathbb{N}$, $D \in \mathbb{N}$. The following bisection algorithm converges to the exact solution of (5.2) in a finite number of steps. The time complexity of the algorithm is $O(N^2 \log(N)(N + \log L))$ where $L = \max b_{i,1} - \min a_{i,1}$.*

Bisection algorithm solving (5.2)

Input: One feasible arrival time interval per aircraft.
Output: Maximal spacing available at airport, with corresponding arrival times.

```
1  $\Delta := \frac{1}{N-1}\left(\max_{i\in\{1,\cdots,N\}} b_{i,n_i} - \min_{i\in\{1,\cdots,N\}} a_{i,1}\right)$
2  if Carlier$(a_{i,1}, b_{i,1}+\Delta, \Delta)$ is feasible,  return $\Delta$
3  while Carlier$(a_{i,1}, b_{i,1}+\Delta, \Delta)$ is not feasible, $\Delta := \Delta/2$
4  $\underline{\Delta} := \Delta$,  $\overline{\Delta} := 2\cdot\Delta$
   while $\left((\overline{\Delta}-\underline{\Delta}) \geq \frac{1}{N-1}\right)$
     if Carlier$\left(a_{i,1}, b_{i,1}+\frac{\underline{\Delta}+\overline{\Delta}}{2}, \frac{\underline{\Delta}+\overline{\Delta}}{2}\right)$ is feasible
       $\underline{\Delta} := \frac{\underline{\Delta}+\overline{\Delta}}{2}$
     else
       $\overline{\Delta} := \frac{\underline{\Delta}+\overline{\Delta}}{2}$
5  return $\max\left\{\Delta | \Delta \in [\underline{\Delta},\overline{\Delta}] \cap \bigcup_{k=1}^{N-1}\left(\frac{\mathbb{N}}{k}\right) \wedge \text{Carlier}(a_{i,1}, b_{i,1}+\Delta,\Delta) \text{ feasible}\right\}$
```

Proof — **Convergence** If $\Delta = \frac{1}{N-1}[b_{N,n_N} - a_{1,1}]$ is feasible, the algorithm stops at step 2. This is the ideal case (the largest achievable spacing is feasible).

Otherwise, let us call Δ^* the optimum of the problem. Let us call $\nu(\cdot)$ the order of arrival of the aircraft (aircraft i arrives in position $\nu(i)$). By Theorem 1, we have:

$$\Delta^* = \min_{(i,j)\in\{1,\cdots,N\}^2,\ i<j} \left(\frac{b_{\nu(j),1} - a_{\nu(i),1}}{\nu(j)-\nu(i)}\right) \tag{5.10}$$

Because of the limited possible values of the difference $\nu(j)-\nu(i)$, and because of the fact that the $a_{i,1}$, and $b_{i,1}$ are integers, equation (5.10) implies:

$$\Delta^* \in \bigcup_{k=1}^{N-1} \frac{\mathbb{N}}{k} \tag{5.11}$$

where $\bigcup_{k=1}^{N-1} \frac{\mathbb{N}}{k} = \{\frac{p}{q} \mid q\in \{1,\cdots,N-1\},\ p\in\mathbb{N}\} \subset \mathbb{Q}$. In step 3, the algorithm will divide Δ by 2 until it becomes feasible. In step 4, starting from the last value of step 3, it will approach Δ^* by bisection until it comes within $\frac{1}{N-1}$ of Δ^*. Then we know that $\Delta^* \in [\underline{\Delta},\overline{\Delta}]$.

Because of the previous remark on Δ^*, $\Delta^* \in [\underline{\Delta}, \overline{\Delta}] \cap \bigcup_{k=1}^{N-1} \left(\frac{\mathbb{N}}{k}\right)$. But this set contains at most $N-1$ elements since $\overline{\Delta} - \underline{\Delta} \leq \frac{1}{N-1}$, which are easy to enumerate (step 5). Δ^* is the largest of them.

<u>**Complexity**</u> The number of calls to Carlier is 1 for step 2, and $N-1$ for step 5. The worst case for step 3 is if Δ has to be divided by 2 p times until

$$\frac{1}{N-1} \geq \frac{\max b_{i,1} - \min a_{i,1}}{N-1} \frac{1}{2^p}$$

and the same applies for step 4, which gives the following bound for the number of calls to Carlier: [§]

$$N + \left\lceil \frac{2}{\log 2} \log\left(\max b_{i,1} - \min a_{i,1}\right) \right\rceil \tag{5.12}$$

The number of calls of the procedure Carlier is thus $O(N + \log L)$, where $L = \max b_{i,1} - \min a_{i,1}$. The complexity of the modified Carlier's algorithm is $O(N^2 \log(N))$ (see Appendix 1). The complexity of the algorithm is thus $O(N^2 \log(N)(N + \log L))$. □

Corollary 1. *Theorem 2 also holds for $a_{i,j} \in \mathbb{Q}$ and $b_{i,j} \in \mathbb{Q}$.*

Proof — Call $a_{i,j} = \frac{p_i^j}{q_i^j}$ and $b_{i,j} = \frac{r_i^j}{s_i^j}$, and call L the least common denominator of the q_i^j and s_i^j (note that $L = \prod_{i=1}^N \prod_{j=1}^{n_i} q_i^j s_i^j$ works as well). Then problem (4.5) is equivalent to the following problem:

$$\begin{array}{ll} \textbf{Maximize:} & L\delta \\ \textbf{Subject to:} & L\theta_i \in \bigcup_{j=1}^{n_i} [L\frac{p_i^j}{q_i^j}, L\frac{r_i^j}{s_i^j}] \\ & |L\theta_i - L\theta_j| \geq L\delta \qquad \forall (i,j) \in \{1, \cdots, N\}^2, \text{ s.t. } i > j \end{array} \tag{5.13}$$

In the previous optimization program, $L\frac{p_i^j}{q_i^j} \in \mathbb{N}$ and $L\frac{r_i^j}{s_i^j} \in \mathbb{N}$. Therefore, Theorem 2 gives us a solution $(\Delta^*, t_1, \cdots, t_N) = (L\delta^*, L\theta_1, \cdots, L\theta_N)$. Therefore, $(\delta^*, \theta_1, \cdots, \theta_N)$ solves the problem in $\mathbb{Q}$. □

[§]Combining the length of the two intervals might give a slightly better bound than (5.12) but does not change the complexity.

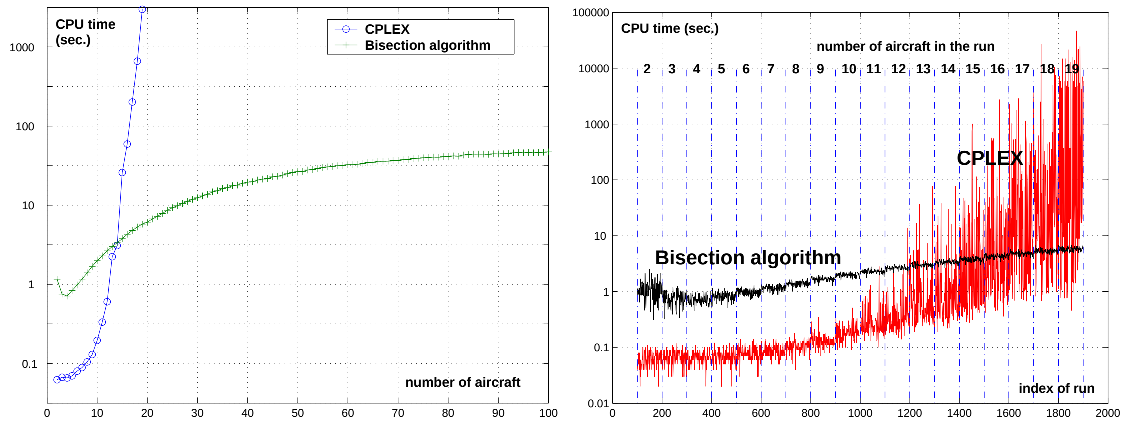


Figure 5.1: **Left:** CPU time used by our algorithm and by CPLEX to solve the same problems. The curve is an average over 100 runs for each value of N (1800 runs for CPLEX, 9900 runs for our algorithm). **Right:** Comparison of the results for the first 1800 simulations realized. As can be seen, even for $N \geq 14$, a significant number of CPLEX computations are faster than our algorithm by at least one order of magnitude.

5.1.5 Numerical performance of the algorithm

The running time of the algorithm is theoretically bounded above by $O(N^2 \log(N)(N + \log L))$, where $O(\cdot)$ means there is a constant in front of $N^2 \log(N)(N + \log L)$ and it is independent of N and L. To verify this time bound we have simulated a set of 1800 cases and compare the measured CPU time used by our algorithm and that using CPLEX to solve the same problem, to demonstrate the efficiency of our algorithm ($N \in \{2, \cdots, 19\}$). We have run 8100 additional simulations ($N \in \{20, \cdots, 100\}$) to assess the performance of our algorithm in a range of N where CPLEX cannot handle the computation in real time.

We simulate situations in which N aircraft are within one hour of the destination airport. For each value of N, we randomly generate intervals $[a_{i,1}, b_{i,1}]$ within one hour of Oakland, with various width $b_{i,1} - a_{i,1}$ ranging from 30 sec. (almost no possibility to adjust the arrival time of the aircraft) to 15 min. We measure the CPU time needed by our method to compute the largest available Δ^* between the aircraft. We run 100 simulations for each N.

Our algorithm is implemented in MATLAB; the CPLEX solution is coded in AMPL interfaced with MATLAB, so that both methods can run from the same application on the same platform.[¶] The results of these runs are shown in Figure 5.1. This figure suggests a few comments. The average result is not representative of the relative performance of the two algorithms. Looking at Figure 5.1 (left) would suggest that CPLEX's performance is much worse than our algorithm for $N \geq 14$. Figure 5.1 (right) shows that the average is "polluted" by abnormal runs: in fact for the set of simulations shown here, CPLEX is faster than our algorithm in 85% of the cases, but among the 15% remaining, the CPU time used by CPLEX exceeds the CPU time used by our algorithm by several orders of magnitude, which increases the average significantly.

We are also aware that there might be more efficient ways to encode the MILP (4.5) in CPLEX or even the one interval version (5.2). For example, another way to encode (4.5) is:

$$\begin{array}{lll}
\textbf{Maximize:} & \Delta & \\
\textbf{Subject to:} & t_i \geq a_{i,n_i} + \sum_{j=1}^{n_i-1} x_{ij}(a_{i,j} - a_{i,j+1}) & \forall i \in \{1, \cdots, N\} \\
& t_i \leq b_{i,n_i} + \sum_{j=1}^{n_i-1} x_{ij}(b_{i,j} - b_{i,j+1}) & \forall i \in \{1, \cdots, N\} \\
& x_{ij} \leq x_{i\ j+1} & \forall i \in \{1, \cdots, N\}\ \forall j \in \{1, \cdots, n_i - 2\} \\
& x_{ij} \in \{0, 1\} & \forall i \in \{1, \cdots, N\}\ \forall j \in \{1, \cdots, n_i - 1\}
\end{array} \tag{5.14}$$

This formulation (suggested to us by Francis Carr) has been observed by other researchers to be more efficient than (5.2). Regardless of the respective benefits of different MILP formulations of the original problem (5.1), the goal of our algorithm is to provide a guarantee on running time for a given task, which CPLEX does not provide, as is illustrated by Figure 5.1.

5.2 An approximation algorithm for a periodic MILP

5.2.1 Problem and related work

We now consider the case in which each aircraft can only use one arrival, but can do a possibly large number of holding patterns before being released back into the arrival.

[¶]We use a SOLARIS 8 workstation under UNIX (2GB of RAM). This is not important as long as the two methods run on the same machine. Comment: The MATLAB implementation is quite unfavorable to our algorithm. MATLAB is notoriously slow for conditional operation such as"if","else"... Note that the CPLEX/AMPL solution procedure is coded separately, and MATLAB is used for it only an interface.

This problem can be formulated as a single machine scheduling problem as follows. We are given N jobs denoted by $J_1, J_2, \cdots, J_N$. Each job J_j is characterized by two nonnegative numbers r_j and d_j. We are given two nonnegative numbers Δ (processing time), and T (holding time). In a feasible schedule (i) each job J_j is assigned a starting time τ_j, such that $r_j + l \cdot T \leq \tau_j \leq d_j + l \cdot T - \Delta$, for some integer $l \geq 0$ which represents the number of holding patterns used; (ii) if $j \neq j'$ then $|\tau_j - \tau_{j'}| \geq \Delta$. Our goal is to find a feasible schedule such that one of the two following objectives are minimized: the makespan $C_{\max} = \max_{1 \leq j \leq n} \tau_j + \Delta$ (the time at which all jobs are finished) and the sum of the starting times of all jobs $\sum_{j=1}^{n} \tau_j$.

Note that this problem is not an *airline scheduling problem*, but an *Air Traffic scheduling problem*, which is why the relevant cost functions are either sum of arrival times (which represents the overall minimization of delays), or makespan (which represents the fastest way to land a given number of aircraft). Another relevant cost function might be the weighted sum of arrival times, which might help to penalize some aircraft less than others (for example if they suffered ground delay prior to departure). Finally, other cost functions might be considered for airline scheduling problems, but they are beyond the scope of this work.

To the best of our knowledge, this scheduling problem has not been studied before in approximation algorithm literature.

In [27], the authors posed the general aircraft scheduling problem with holding time. They used an integer programming approach to solve the problem, using CPLEX, and showed the limit of this approach (the computational time grows exponentially with the number of aircraft, making this approach unattractive for online implementations). In Section 5.1, we present a polynomial time algorithm for solving the single interval case: the objective is to maximize $\delta := \min_{i<j} |\tau_j - \tau_i|$ without putting the aircraft on hold. In the general case, $\delta < \Delta$, which forces ATC to put aircraft on hold: the present algorithm has been developed to address this situation.

Our problem generalizes those studied by Dantzig and Fulkerson [65], and Gertsbakh and Stern [78] where each job's starting time can only be in a single interval $[r_j, d_j - \Delta]$. To find a feasible schedule for the given jobs, one can schedule them on parallel identical machines. Here, the goal is to minimize the number of machines needed. One can easily see that this problem is equivalent to our problem when the objective is to minimize the number of periods we have to use and when all $d_j \in [0, T]$. The problem can be solved to optimality if $r_j = d_j - \Delta$ for for j. If $r_j < d_j - \Delta$, the complexity of the problem is still open.

This work can also be related to the job interval selection problem (JISP) [56, 69]. If the maximum number of intervals required for feasibility of our problem is known a priori, our problem might be reduced to the JISP. However, the number of intervals of the corresponding JISP is $O(L)$ where $L = O(\max_j d_j - \min_j r_j)$, which is an undesirable feature. Given the similarity of our problem with the JISP, we believe that our problem is NP-complete. Another closely related scheduling problem is the one studied by Carlier [49] and Baptiste [12] where the job's starting time is restricted to a single interval $[r_j, d_j - \Delta]$ and the objective is to maximize the number of jobs which can be scheduled. This problem can be solved in polynomial time [12]. Their results imply that for our problem, if all the jobs can be scheduled in the first available interval for each aircraft, the problem is polynomial time solvable.

Notice that when the processing time for each job is the same, minimizing the sum of starting times of jobs is equivalent to minimizing the sum of completion times of jobs. Designing approximation algorithms for various scheduling problems with the objective minimizing the sum of completion times has received considerable attention during the last decades, see for example an excellent survey by Chen, Potts and Woeginger [54]. However, because of the structure of our problem, it seems that none of the algorithms presented above can be applied to give an approximation for our problem. In most of the models, only release times of jobs are present, but here, each job has a deadline, although the job can be scheduled in a prescribed later range if this deadline cannot be met.

5.2.2 Results and techniques

We present an approximation algorithm which approximates the sum of completion times with factor 5. We extend it to approximate makespan with factor 3.

Our first attempt at this problem is based on a simple observation that the starting time of the jobs can be restricted to a polynomial bounded set. Therefore, the problem can be formulated as a constrained assignment problem: assign the jobs to those possible starting points, with the constraints that at most one job can be assigned to any interval of length Δ. As a standard approach, we solve the linear programming (LP) relaxation of this constrained assignment problem. The key in our algorithm is the rounding of the LP solution x, which is fractional in general, into an integer solution for the original problem. Here we observe that there is a way to modify the fractional solution x and obtain another fractional solution $\bar{x}$ (which still satisfies the constraints of the LP) such that the corresponding costs (both makespan and the sum of the starting times) are increased by a factor of at most 3 if all $r_j \geq T$. Then we construct an instance of the (unconstrained) assignment problem such that $\bar{x}$ is a feasible solution of the assignment problem and any integer solution of the new instance will automatically satisfy the constraints of the original problem. Using the fact that we can find an integer solution for the (unconstrained) assignment in polynomial time, whose cost is not worse than that of $\bar{x}$, we find an approximation solution for the original problem with guaranteed error factor if all $r_j \geq T$. However, bounding the ratio of the algorithm if there are jobs with r_j less than T requires an initial step to schedule these jobs.

Motivated by the results of Baptiste [12] that for the single interval case, there exists a polynomial time algorithm that can schedule the maximum number of jobs, we treat the jobs with $r_j < T$ separately. We thus develop a greedy approach: we schedule the maximum number of jobs in the first period $[0, T]$, and then apply the above mentioned LP-based algorithm to the rest of the jobs. Two issues need to be addressed. (i) After we schedule the jobs in the first interval, how do we bound the optimal cost of the rest of the jobs? Indeed, when we apply the LP-based algorithm to the rest of the jobs, we are comparing their cost in this schedule with their cost if they were scheduled by the optimum solution. Our proof provides a bound on the ratio of these two costs. (ii) Scheduling the maximum number of jobs in the first period does not necessarily imply that the sum of starting times of this set of jobs is minimized. Here, we generalize the algorithm and result of Baptiste [12] and apply it to the set of aircraft which can arrive before T: we derive a polynomial-time algorithm for the single interval problem, which schedules (feasibly) the maximum number of jobs; furthermore, we show that among all feasible schedules that schedule the same number of jobs, the algorithm produces a solution that has the minimum sum of starting times. The algorithm is based on dynamic programming.

Combining the above algorithms, we obtain an approximation algorithm which approximates the optimal solution of minimizing the sum of starting times with factor 5. For makespan, we modify both parts of the algorithm and obtain a ratio 3.

The rest of this Chapter is organized as follows. Section 5.2.3 describes the optimization program formulation of the physical problem. Section 5.2.4 shows a trivial 2 approximation algorithm for the case in which $T \leq \Delta$, which works when the objective function is sum of arrival times or makespan. Section 5.2.5 provides a high level description of the approximation algorithm for the sum of arrival times in the case $T > \Delta$. The part of the algorithm which schedules the first set of jobs before T (dynamic programming) is explained in detail in Section 5.2.6. Section 5.1.4 presents the part of the algorithm which schedules the rest of the jobs after T (LP rounding). Section 5.2.8 explains how to modify the algorithm to provide a 3-approximation algorithm for makespan. The proof of the algorithm of Section 5.2.6 is presented in Appendix B.

5.2.3 Optimization program

We call a_i the earliest arrival time of aircraft i, and b_i the latest arrival time of aircraft i without holding pattern.[‖] In the absence of holding pattern, the problem of scheduling aircraft i in $[a_i, b_i]$ for all $i \leq N$ such that two aircraft are separated by at least Δ is equivalent to scheduling N jobs on a single machine, with processing times Δ, release times $r_i = a_i$, deadlines $d_i = b_i + \Delta$. We call T the time of a holding pattern, and let $[a_j, b_j] + T\mathbb{N} := \bigcup_{k=0}^{\infty}[a_j + kT, b_j + kT]$. The problem can be formulated as follows:

$$\begin{array}{lll} \textbf{min:} & \sum_{j\in\{1,\cdots,N\}} \tau_j & \\ \textbf{s.t.:} & \tau_j \in [a_j, b_j] + T\mathbb{N} & \forall j \in \{1,\cdots,N\} \\ & |\tau_j - \tau_j'| \geq \Delta & \forall (j,j') \in \{1,\cdots,N\}^2, \text{ such that } j \neq j' \end{array} \tag{5.15}$$

5.2.4 A trivial algorithm for the $T \leq \Delta$ case

In this case, the following first come first served algorithm reads:

Theorem 3. *The first come first served algorithm is a 2 approximation algorithm for makespan and sum of arrival times.*

[‖]We use the notation a_i and b_i rather than $a_{i,1}$ and $b_{i,1}$ for simplicity, since all other intervals in (5.15) are obtained from $[a_i, b_i]$ by translation by an integer multiple of T.

First come first served approximation 2 algorithm

Input: Arrival time intervals for each aircraft, time to hold T.
Output: Approximation 2 schedule.

1) Sort the aircraft by earliest possible time of arrival: without loss of generality, we can write $a_1 \leq a_2 \leq \cdots \leq a_{N-1} \leq a_N$.

2) Construct the following schedule:

$$\begin{aligned} \tau_1 = &\quad a_1 \\ \tau_j = &\quad \min\{x \mid x \in [\tau_{j-1} + \Delta, +\infty) \cap \\ &\quad ([a_j, b_j] + T\mathbb{N})\} \quad \forall j \in \{2, \cdots, N\} \end{aligned}$$

Proof of Theorem 3: Sum of arrival times: Divide the N aircraft into K subsets: $S_1 = \{N_1, \cdots, N_2 - 1\}$, $S_2 = \{N_2, \cdots, N_3 - 1\}$, $\cdots$, $S_K = \{N_K, \cdots, N\}$, where

$$\begin{aligned} &N_1 = 1 \\ &N_k = \min\{p | a_{N_{k-1}} + (p - N_{k-1})(\Delta + T) < a_p\} \end{aligned}$$

For all $k \in \{1, \cdots, K-1\}$, for all $j \in S_k$, the algorithm provides $\tau_j \leq a_{N_k} + (j - N_k)(\Delta + T)$. Indeed, if it were not true, we would have a $j \in S_k$ such that $\tau_j > a_{N_k} + (j - N_k)(\Delta + T)$, which would contradict the definition of N_{k+1}. We call c the cost of this algorithm:

$$\begin{aligned} c &\leq \textstyle\sum_{k=1}^{K} \sum_{j \in S_k} \tau_j \\ &\leq \textstyle\sum_{k=1}^{K} \sum_{j \in S_k} (a_{N_k} + (j - N_k)(\Delta + T)) \\ &\leq \textstyle\sum_{k=1}^{K} \sum_{j \in S_k} \frac{(a_{N_k} + (j-N_k)(\Delta+T))(a_{N_k} + (j-N_k)\Delta)}{a_{N_k} + (j-N_k)\Delta} \end{aligned}$$

Since we know that OPT $\geq \sum_{k=1}^{K} \sum_{j \in S_k} (a_{N_k} + (j - N_k)\Delta)$, we get the following approximation ratio β for the sum of arrival times:

$$\begin{aligned} \beta &= \max_{k=1}^{K} \max_{j \in S_k} \left\{ \frac{a_{N_k} + (j-N_k)(\Delta+T)}{a_{N_k} + (j-N_k)\Delta} \right\} \\ &= \max_{k=1}^{K} \max_{j \in S_k} \left\{ 1 + \frac{(j-N_k)T}{a_{N_k} + (j-N_k)\Delta} \right\} \\ &\leq 1 + \frac{T}{\Delta} \leq 2 \end{aligned} \tag{5.16}$$

Makespan: We know that $\tau_N \leq a_{N_K} + (N - N_K)(\Delta + T)$. Since the jobs have been ordered such that $a_1 \leq a_2 \leq \cdots \leq a_{N-1} \leq a_N$, the arrival time τ_N^* of aircraft N in the optimum schedule has to satisfy $\tau_N^* \geq a_{N_K} + (N - N_K)\Delta$. The approximation ratio α for makespan thus satisfies:

$$\begin{aligned}\alpha &\leq \tfrac{a_{N_K}+(N-N_K)(\Delta+T)+\Delta}{a_{N_K}+(N-N_K)\Delta+\Delta}\\ &\leq 1+\tfrac{(N-N_K)T}{a_{N_K}+(N-N_K)\Delta+\Delta} \leq 2\end{aligned}$$

□

5.2.5 The case $T > \Delta$

Before presenting our algorithm, we introduce some notation and initial steps.

Data preprocessing

Input: Arrival time intervals for each aircraft, time to hold T.
Output: Reduced constrained set containing the optimum of the optimization program.

(i) Sort the aircraft by earliest possible time of arrival: without loss of generality, we can write $a_1 \leq a_2 \leq \cdots \leq a_{N-1} \leq a_N$.

(ii) Divide the N aircraft into K subsets:
$S_1 = \{N_1, \cdots, N_2 - 1\}$,
$S_2 = \{N_2, \cdots, N_3 - 1\}, \cdots$
$S_K = \{N_K, \cdots, N\}$
where $N_1 = 1$, and N_k is given by $N_k = \min\{p | a_{N_{k-1}} + (p - N_{k-1})(\Delta + T) < a_p\}$.

(iii) Let $\sigma_k = \left\{\bigcup_{l\in S_k}\{a_l + \Delta\mathbb{N} + T\mathbb{N}\}\right\} \cap [a_{N_k}, a_{N_k} + (N_k - N_{k-1})(\Delta + T)]$ for each k. Index the elements t_i of $\Sigma = \bigcup_{k=1}^{K} \sigma_k$ by $i \in \{1, \cdots, |\Sigma|\}$ in increasing order.

Step *(i)* is used for notational convenience. Step *(ii)* enables us to construct a grid of polynomial size: it separates aircraft into groups for which we have at least a feasible solution (which we can construct using the trivial algorithm of Section 5.2.4). In step *(iii)*, the K groups are gridded (σ_k) and assembled in a single grid Σ of size $O\left(\frac{T}{\Delta}N^2\right)$. The set $\bigcup_{l\in S_k}\{a_l+\Delta\mathbb{N}+T\mathbb{N}\}$ is the set of earliest arrival times plus an integer number of T (holding patterns) and/or Δ (time to process one aircraft). It follows from Proposition 1 that the optimum schedule lies on this grid (the optimum is sometimes referred to as *left shifted*).

In the Main algorithm shown above, $I(i)$ represents the set of aircraft for which the arrival times are less than Δ time units after t_i, and thus not allowed if t_i is chosen by an integer solution of (5.17). $G(j)$ represents the set of i such that t_i is available for aircraft j.

Main Algorithm

1. If $a_1 < T$, call F the set of i such that $[a_i, b_i] \cap \{t | t \leq T\}$ is not empty. Call $[a_i, b'_i] = [a_i, b_i] \cap \{t | t \leq T\}$. Schedule the maximum number of aircraft of F according to the algorithm of Section 5.2.6. If this number is equal to N, stop.

2. Solve the relaxed LP (5.17) for the remaining aircraft. If $a_1 \geq T$, solve the relaxed LP (5.17) directly.

$$\begin{array}{lll} \textbf{Minimize:} & \sum_j \sum_{i \in G(j)} t_i x_{ij} & \\ \textbf{Subject to:} & \sum_{i \in G(j)} x_{ij} = 1 & \forall j \in \{1, \cdots, N\} \\ & x_{ij} \geq 0 & \forall j \in \{1, \cdots, N\}, \ \forall i \in G(j) \\ & \sum_{i' \in I(i)} \sum_j x_{i'j} \leq 1 & \forall i \in \{1, \cdots, |\Sigma|\} \end{array} \tag{5.17}$$

where $\forall i \in \{1, \cdots, |\Sigma|\}$, $I(i) = \{i' \leq |\Sigma| \mid t_{i'} \geq t_i \wedge t_{i'} - t_i < \Delta\}$, and $\forall j \in \{1, \cdots, N\}$, $G(j) = \Sigma \cap \{[a_j, b_j] + T\mathbb{N}\}$.

3. Modify the x_{ij} using the procedure $\overline{x}_{ij} = \mathsf{TransformLPsol}(x_{ij})$.

4. Compute the integer solution $\tilde{x}_{ij} = \mathsf{Matching}(\overline{x}_{ij})$ of the matching problem constructed by the Matching procedure. The result is a feasible schedule for the remaining aircraft.

We call n the number of aircraft scheduled before T by the algorithm of Step 1 of the Main Algorithm, and $m = N - n$ the number of aircraft scheduled after T. In the rest of this Chapter, as well as for the proofs, we use the notation of Table 5.1:

Lemma 2. *Consider the $|F|$ aircraft of F (Step 1 of the Main Algorithm). Step 1 schedules the largest number of them in $\bigcup_{i=1}^{|F|} [a_i, b'_i]$, with minimum sum of arrival times. The complexity of Step 1 of the algorithm is $O(|F|^9)$.*

Lemma 3. *The sum of starting times for the $m = N - n$ other aircraft scheduled after T is bounded above by $5 \cdot OPT(m, last)$. The complexity of scheduling these m aircraft is dominated by solving the LP (5.17).*

Theorem 4. *The Main Algorithm has a 5-approximation ratio for the sum of arrival times. Its complexity is $O(N^9)$, and is dominated by Step 1 of the Main Algorithm.*

Proof of Theorem 4: Let c_n be the cost of the n jobs scheduled before T by Step 1. Let c_m be the cost of the m jobs scheduled after T by Step 4. By Lemma 2, we have

$C(n)$	cost of scheduling n jobs before T with Step 1 of the Main Algorithm (cost means sum of arrival times)
$C(LP, m)$	min. cost of relaxed LP (5.17) for the m jobs
$C(IP, m)$	min. cost of a feasible integer solution to (5.17) for the m jobs
$C(\overline{x}, m)$	cost of the fractional solution (for the m jobs) $\overline{x}_{ij}$ of (5.17) produced by Step 3 of the Main algorithm
$C(\tilde{x}, m)$	cost of the fractional solution (for the m jobs) $\tilde{x}_{ij}$ produced by Step 4 of the Main Algorithm
$\mathrm{OPT}(m)$	cost of the m jobs when scheduled by the optimal solution OPT
$\mathrm{OPT}(n, \mathrm{first})$	cost of the first n jobs when they are scheduled by OPT
$\mathrm{OPT}(m, \mathrm{last})$	cost of the last m jobs when they are scheduled by OPT

Table 5.1: Notation for the main algorithm

$c_n \leq \mathrm{OPT}(n, \mathrm{first})$. By Lemma 3, we have $c_m \leq 5 \cdot \mathrm{OPT}(m, \mathrm{last})$. Now combining the two bounds for c_n and c_m, we get:

$$\begin{aligned} c_n + c_m &\leq \mathrm{OPT}(n, \mathrm{first}) + 5 \cdot \mathrm{OPT}(m, \mathrm{last}) \\ &\leq 5 \cdot \mathrm{OPT} \end{aligned}$$

The complexity of Step 1 of the algorithm is $O(|F|^9)$. The worst case is when $|F| = N$ (all jobs can be scheduled in F), therefore the complexity of Step 1 is $O(N^9)$. The complexity of the other steps is determined by solving the linear program (5.17), so the overall complexity is $O(N^9)$. □

5.2.6 Dynamic programming algorithm for step 1 of the main algorithm

We call $|F| = f \in \mathbb{N}$. We are given a set of time intervals $[r_i, d_i]$, $i \in \{1, \cdots, f\}$, and assume that the d_i have been sorted in chronological order: $d_1 \leq d_2 \leq \cdots \leq d_f$. A schedule $\{t_i\}_{i \in \{1, \cdots, l\}}$ is said to be admissible if for all i in $\{1, \cdots, l\}$, $t_i \geq r_i$ (jobs start after their release time), $t_i + \Delta \leq d_i$ (jobs are finished before their deadline), and $\forall i \neq j$, $|t_i - t_j| \geq \Delta$ (two jobs cannot be processed simultaneously). For l given ($l \leq f$), we want to compute $\min \sum_{p=1}^{l} t_{i_p}$, the minimum of the sum of starting times of l of the f jobs.

Definition 1.

- *We call* $\Theta = \{t \mid \exists i \in \{1, \cdots, f\},\ \exists l \in \{0, \cdots, f\},\ \text{such that } t = r_i + l\Delta\}$.

• *We call $U_k(s,e) = \{J_i \mid i \leq k$ and $s \leq r_i < e\}$ the set of jobs of index less than k released between s and e.*

• *For a given l, we call $C_k(s,e,l)$ the minimum sum of starting times among the set of admissible schedules containing l jobs in $U_k(s,e)$, which satisfy*

(i) *for all i, $t_i \geq s + \Delta$ (job i starts after $s + \Delta$)*

(ii) *for all i, $t_i + \Delta \leq e$ (job i is processed before e)*

(iii) *for all i, $t_i \in \Theta$ (starting times are in Θ).*

We pose $C_k(s,e,l) = +\infty$ for all $l > k$.

Proposition 1. *(Carlier [49], Baptiste [12]) There exists an optimal schedule such that $t_i \in \Theta$ for any i scheduled.*

The proof of Proposition 1 is an extension of the work of Carlier and Baptiste [49, 12].

Proposition 2. *$C_k(s,e,l)$ can be computed recursively by the following formula:*

$$C_k(s,e,l) = C_{k-1}(s,e,l)$$

if $r_k \notin [s,e)$ for the k^{th} job J_k, and

$$C_k(s,e,l) = \min \begin{cases} C_{k-1}(s,e,l), \\ \min\limits_{\substack{s' \in \Theta \\ 1 \leq q \leq l-1 \\ \max(r_k, s+\Delta) \leq s' \leq \min(d_k, e) - \Delta}} \left(C_{k-1}(s,s',l-q) + s' + C_{k-1}(s',e,q-1)\right) \end{cases}$$

otherwise.

The proof of Proposition 2 can be found in the Appendix.

Proof of Lemma 2: The aircraft scheduling problem can be transformed into a job scheduling problem by letting $r_i = a_i$ and $d_i = b'_i + \Delta$, for $i \in F$, as defined in the Main Algorithm. We call $f = |F|$. For notational convenience, we can relabel these jobs from 1 to f. Note that f can range from 1 to N. We can now apply Proposition 2. For all k, s, e, l, such that

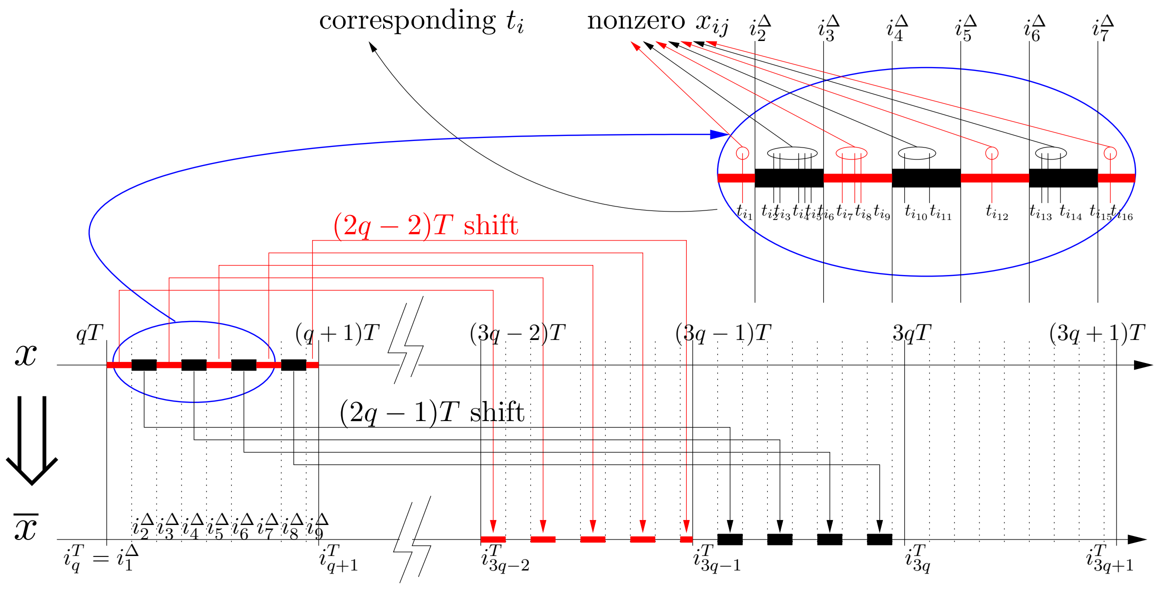


Figure 5.2: Illustration of the procedure TransformLPsol. This procedure transforms the feasible fractional solution $x = x_{ij}$ of (5.17) into another feasible fractional solution $\overline{x} = \overline{x}_{ij}$ with idle time. Half of the fractional x_{ij} in the interval $[qT,(q+1)T)$ is shifted to $[(3q-2)T,(3q-1)T)$ (shift by $(2q-2)T$), and the other half to $[(3q-1)T,3qT)$ (shift by $(2q-1)T$). The interval $[3qT,(3q+1)T)$ is empty for now; this is a waste of space, but this enables us to satisfy the constraints of (5.17) for the part of the fractional solution between i_Q^Δ (i_9 on the figure) and i_{q+1}^T.

$$\begin{aligned}
&k \le f\\
&s \ge \min_{i=1}^{f} r_i - \Delta\\
&e \le \max_{i=1}^{f} d_i\\
&l \le f
\end{aligned}$$

one can compute recursively $C_k(s,e,l)$, and find l_0, the largest l such that $C_f(\min_{i=1}^{f} r_i - \Delta, \max_{i=1}^{f} d_i, l_0)$ is finite.

k and l range from 1 to f, e and s are on the grid Θ of Proposition 2, thus they are in a set of size $O(f^2)$. The size of the dynamic program table to build is thus $O(f^6)$. In building the table, we take the minimum over a set of size $O(f^3)$, since it is indexed by $s' \in \Theta$ and $q \in \{1,\cdots,l-1\}$. Thus, the total cost of the algorithm is $O(f^9)$. $\square$

5.2.7 Linear programming based algorithm for steps 2-3-4 of the main algorithm

This section summarizes steps 2-3-4 of the Main Algorithm. We first solve the relaxed LP (5.17). This LP is a constrained assignment problem: aircraft j are assigned arrival times t_i; the last constraint means that any integer solution of this problem can only assign one t_i every Δ (by definition of the interval $I(i)$). In the relaxed version, it means that the x_{ij} (fractional assignments) sum to at most 1 over a period of length Δ.

Procedure TransformLPsol

Input: fractional solution x_{ij} of (5.17).
Output: $\overline{x}_{ij}$, another feasible fractional solution of (5.17).

Perform the following assignment:

- $\overline{x}_{ij} = x_{i'j}$ where $t_i = t_{i'} + (2q-2)T$, if $\exists q \geq 0$, $\exists l \geq 0$, such that $l < \lfloor \frac{T}{\Delta} \rfloor$ is even, and $t_{i'} \in [qT + l\Delta, qT + (l+1)\Delta)$ or such that $l = \lfloor \frac{T}{\Delta} \rfloor$ is even, and $t_i = t_{i'} + (2q-2)T$ for $t_{i'} \in [qT + l\Delta, (q+1)T)$.
- $\overline{x}_{ij} = x_{i'j}$ where $t_i = t_{i'} + (2q-1)T$, if $\exists q \geq 0$, $\exists l \geq 0$, such that $l < \lfloor \frac{T}{\Delta} \rfloor$ is odd, and $t_{i'} \in [qT + l\Delta, qT + (l+1)\Delta)$ or such that $l = \lfloor \frac{T}{\Delta} \rfloor$ is odd, and $t_i = t_{i'} + (2q-1)T$ for $t_{i'} \in [qT + l\Delta, (q+1)T)$.

The two last items correspond to the last (incomplete) chunk of $[qT, (q+1)T)$.

The fractional solution x is transformed to another fractional solution $\overline{x}$ by the procedure TransformLPsol (illustrated in Figure 5.2). The fractional solution is decomposed into sets of t_i such that $t_i \in [qT, (q+1)T)$, where $q \in \mathbb{N}$. Each of these sets is decomposed into chunks of length Δ. Figure 5.2 shows one of these intervals (eight chunks of length Δ, and the last chunk of length less than Δ). The x_{ij} in each of these chunks is shifted by $(2q-2)T$ or $(2q-1)T$ depending on parity, in order to insert idle time of length Δ between the corresponding chunks (see arrows in Figure 5.2). Because of the last chunk (interval $\{i_9^\Delta, \cdots, i_{q+1}^\Delta\}$ in Figure 5.2), it is necessary to insert a period of idle time of length at least Δ after the highest shifted x_{ij} (the black chunk coming from $\{i_8^\Delta, \cdots, i_9^\Delta\}$ in Figure 5.2). This is done by adding a full interval $[3qT, (3q+1)T)$ of idle time after $3qT$.[**] The result is another fractional solution $\overline{x}$ which satisfies (5.17) and has idle time periods of length Δ

[**]This is not optimal, but enables us to construct $\overline{x}$ systematically. This might be improved in the future.

alternating with non idle time periods of length Δ. This means sets in which $\bar{x}_{ij} = 0$ for t_i ranging in an interval of length Δ alternate with sets of the same length with nonzero $\bar{x}_{ij}$.

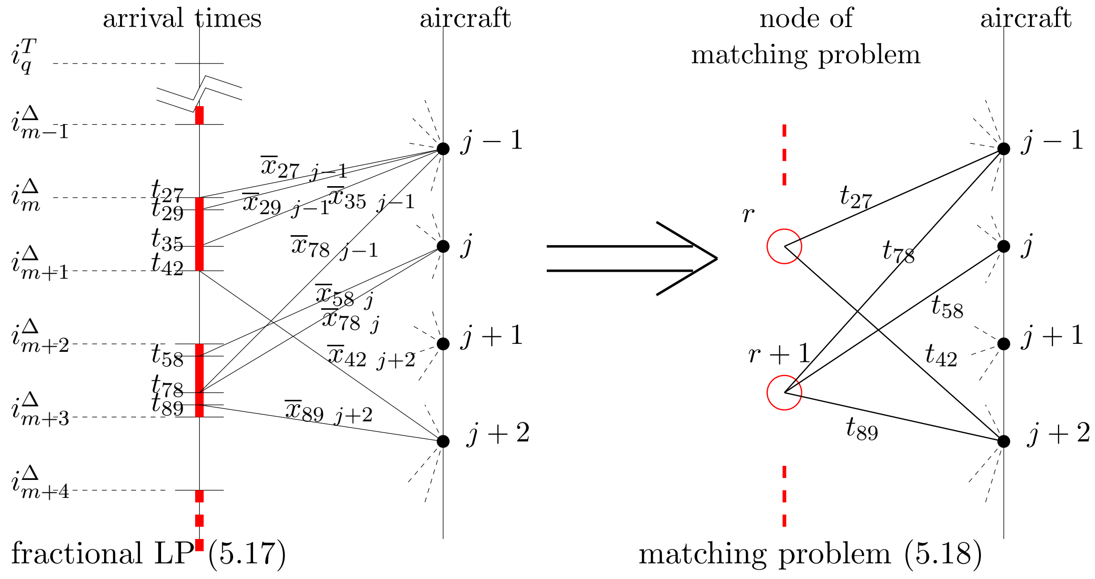


Figure 5.3: Illustration of the procedure Matching. This procedure transforms the feasible fractional solution $\overline{x}_{ij}$ into a weighted matching problem. In the fractional LP solution, only the nonzero $\overline{x}_{ij}$ are represented (solid if they are in the range of the figure, dashed if they connect aircraft with arrival times outside the figure). On the right plot, the weights are the arrival times from the fractional LP solution: for example, $\Theta_{rj-1} = t_{27}$.

The procedure Matching takes the new fractional solution $\overline{x}$, and constructs a feasible instance of a weighted assignment problem. An illustration is given in Figure 5.3. Every chunk $\{i_m^\Delta, \cdots, i_{m+1}^\Delta - 1\}$ with nonzero $\overline{x}_{ij}$ is reduced to a single node indexed by r. Any aircraft j which has one (or more) nonzero $\overline{x}_{ij}$ with $i \in \{i_m^\Delta, \cdots, i_{m+1}^\Delta - 1\}$ is now linked to r (Figure 5.3 right). The weight on the corresponding link is the smallest t_i with nonzero $\overline{x}_{ij}$ in $\{i_m^\Delta, \cdots, i_{m+1}^\Delta - 1\}$. For example in Figure 5.3 right, the weight on the link $r \to (j-1)$ is t_{27} because in Figure 5.3 left, node $j-1$ was such that x_{27}, x_{29} and x_{35} are nonzero. The corresponding weighted assignment problem is (5.18). A fractional feasible solution $\tilde{x}$ is obtained from $\overline{x}$ by adding all $\overline{x}_{ij}$ emanating from the same chunk towards aircraft j. We know that (5.18) has an integer optimal, which is therefore less or equal to our fractional feasible solution. We are now able to prove Lemma 3 (notations are defined in Section 5.2.5).

Procedure Matching

Input: fractional solution $\overline{x}_{ij}$ of (5.17).
Output: integer solution of a matching problem solving (5.17).

1. Construct the following set of Θ_{qj}:

```
$q = 1$, $r = 1$
while $\exists t_i > qT$ such that $\exists j \in \{1, \cdots, N\}$ such that $\overline{x}_{ij} \neq 0$
  $i_1^{\Delta} = i_q^T$,  $m = 1$
  while $i$ such that $t_i = t_{i_m^{\Delta}} + \Delta$ is less than $i_{q+1}^T$
    $i_{m+1}^{\Delta} = i$ such that $t_i = t_{i_m^{\Delta}} + \Delta$,  $r = r + 1$
    for $j = 1$ to $N$
      if $\exists i \in \{i_m^{\Delta}, \cdots, i_{m+1}^{\Delta} - 1\}$ s.t. $\overline{x}_{ij} \neq 0$
        $\Theta_{rj} = \min\{t_i \mid i \in \{i_m^{\Delta}, \cdots, i_{m+1}^{\Delta} - 1\}, \overline{x}_{ij} \neq 0\}$ end if
    end for
  end while
  for $j = 1$ to $N$
    if $\exists i \in \{i_m^{\Delta}, \cdots, i_{q+1}^T - 1\}$ s.t. $\overline{x}_{ij} \neq 0$
      $r = r + 1$, $\Theta_{rj} = \min\{t_i \mid i \in \{i_m^{\Delta}, \cdots, i_{q+1}^T - 1\}, \overline{x}_{ij} \neq 0\}$ end if
  end for
  $q = q + 1$
end while
for all $j$, call $H(j)$ the set of $r$ for which $\Theta_{rj}$ has been assigned.
```

2. Solve for the integral solution $\tilde{x}_{ij}$ of the following weighted matching problem:

$$
\begin{array}{lll}
\textbf{Minimize:} & \sum_j \sum_{q \in H(j)} \Theta_{qj} \tilde{x}_{qj} & \\
\textbf{Subject to:} & \sum_{q \in H(j)} \tilde{x}_{qj} = 1 & \forall j \in \{1, \cdots, N\} \\
& 0 \leq \tilde{x}_{qj} \leq 1 & \forall j \in \{1, \cdots, N\},\ \forall q \in H(j) \\
& \sum_j \tilde{x}_{qj} \leq 1 & \forall q \in \bigcup_{j=1}^N H(j)
\end{array}
\tag{5.18}
$$

Proof of Lemma 3: Let us call c_m the cost of the m jobs scheduled after T. For each of these jobs, we compute an upper bound of the ratio by which the cost is increased by the procedure TransformLPsol for each nonzero x_{ij}. The minimum cost $C(IP, m)$ of scheduling the m jobs after T is less than the cost of scheduling these m jobs within OPT and shifting them by T. Therefore:

$$C(LP, m) \leq C(IP, m) \leq \mathrm{OPT}(m) + mT$$

Let us define B_q as the sum of fractional x_{ij} in interval $[qT, (q+1)T)$:

$$B_q = \sum_{i:qT \leq t_i < (q+1)T} \sum_j x_{ij}.$$

Let us call r the largest q such that $B_q \neq 0$. By construction of the LP (5.17), we have

$$\sum_{q=1}^{r} B_i = m \qquad \text{and} \qquad \sum_{q=1}^{r} B_q \cdot qT \leq C(LP, m)$$

Let us define $C(B_q)$ as the cost of the fractional LP solution of (5.17) corresponding to the interval $[qT, (q+1)T)$. As shown in Figure 5.2, the amount by which x_{ij} is shifted is either $(2q-2)T$ or $(2q-1)T$, and therefore the increase in cost of the x_{ij} in interval $[qT, (q+1)T)$ due to the procedure TransformLPsol is at most $B_q \cdot (2q-1)T$. Thus, we have the following upper bound on the cost c_m of the m jobs scheduled after T by our algorithm.

$$\begin{array}{rcl}
C(\overline{x}, m) & \leq & \sum_{q=1}^{r} (C(B_q) + B_q \cdot (2q-1)T) \\
& = & C(LP, m) + \sum_{q=1}^{r} B_q \cdot (2q-1)T \\
& \leq & \mathrm{OPT}(m) + mT + \sum_{q=1}^{r} B_q \cdot (2q-1)T \\
& = & \mathrm{OPT}(m) + 2\sum_{q=1}^{r} qB_qT + mT - \sum_{q=1}^{r} B_qT \\
& = & \mathrm{OPT}(m) + 2\sum_{q=1}^{r} qB_qT \\
& \leq & \mathrm{OPT}(m) + 2C(LP, m) \\
& \leq & \mathrm{OPT}(m) + 2\,(\mathrm{OPT}(m) + mT) \\
& \leq & 3 \cdot \mathrm{OPT}(m) + 2mT \\
& \leq & 5 \cdot \mathrm{OPT}(m, \mathrm{last})
\end{array}$$

c_m is the optimal solution of the weighted matching problem (5.18) obtained by the procedure Matching. We know from [2] that there exists at least one integral solution which achieves c_m, which we can find in polynomial time. For this solution, we thus have $c_m \leq C(\overline{x}, m)$ $\square$

5.2.8 Modified algorithm for makespan

Theorem 5. *The following modification of the* Main Algorithm *is a 3-approximation algorithm for makespan.*

Proof of Theorem 5: If Carlier's algorithm schedules the N aircraft before T, it follows from [49] that this schedule provides the optimum makespan. If $n < N$, Carlier's algorithm implies that it is not possible to schedule N aircraft in $[0, T]$. Therefore the optimum makespan $C^*_{\max}$ satisfies $C^*_{\max} > T$. Applying Steps 2-3-4 of the algorithm directly provides a feasible solution $\overline{C}_{\max}$. A similar proof as for Lemma 3 provides the approximation ratio 3 for makespan. $\square$

Modified algorithm for makespan

Input: Same as for the Main Algorithm
Output: Arrival schedule, with approximation ratio 3 for makespan.

1) In Step 1, replace the algorithm of Section 5.2.6 by Carlier's algorithm [49]. If this algorithm schedules N aircraft, Stop. Else, apply Step 2 directly to the N aircraft.

2) In step 2, given $C_{\max}$, let us replace (5.17) with the following feasibility problem:

$$\begin{array}{ll} \sum_{i \in G(j)} x_{ij} = 1 & \forall j \in \{1, \cdots, N\} \\ x_{ij} \geq 0 & \forall j \in \{1, \cdots, N\},\ \forall i \in G(j) \\ \sum_{i' \in I(i)} \sum_j x_{i'j} \leq 1 & \forall i \in \{1, \cdots, |\Sigma|\} \end{array} \qquad (5.19)$$

where $\forall i \in \{1, \cdots, |\Sigma|\}$, $I(i) = \{i' \leq |\Sigma| \mid t_{i'} \geq t_i \ \wedge \ t_{i'} - t_i < \Delta\}$, and $\forall j \in \{1, \cdots, N\}$, $G(j) = \Sigma \cap \{[a_j, b_j] + T\mathbb{N}\} \cap [0, C_{\max}]$. If there exists a fractional solution to the set (5.19) of constraints with corresponding $C_{\max}$-dependent $G(j)$, it represents a fractional schedule of makespan less than $C_{\max}$, which provides a lower bound on the makespan of the original problem. We can compute the smallest possible $C_{\max}$ solving (5.19) to ϵ using bisection. (In fact, we can compute it in $O(\log |\Sigma|)$ bisection steps, since the total number of grid points is $|\Sigma|$.)

3,4) Steps 3 and 4 are identical with those in the Main Algorithm. The complexity of the algorithm is dominated by the solution of the LPs (5.19) and (5.18).

Chapter 6

An Eulerian model of network air traffic flow

In Chapter 3, we derived a Lagrangian model of sector-based traffic flow, which relies on hybrid automata, in order to model the aircraft's behavior. This model enabled the analysis of congestion propagation into the system. It also modeled human Air Traffic Controller actuation on the flow, as a cost function trying to minimize a cost corresponding to a set of hierarchical priorities of the Air Traffic Controller. In Chapter 4, we used this model to pose the problem of optimally routing and sequencing aircraft into an arrival airport. This problem was first posed as a hybrid system controller synthesis problem, and reduced to a *Mixed Integer Linear Program* (MILP). Finally, in Chapter 5, we derived combinatorial optimization algorithms to solve the MILP, with guaranteed bounds on optimality and speed.

As was explained in Chapters 2 and 3, while the sector Air Traffic Controllers deal mainly with traffic separation, the *Traffic Flow Management* (TFM) at the Center level tries to optimize the flow. At the highest control level (ATCSCC, see Figure 2.1 in Chapter 2), the control authority is interested in controlling mass balances in the NAS, i.e. controlling the system at a higher level (multi-Center). At this level of control, the individual trajectories of the individual aircraft might not be crucial; what is more important is their density (or concentration) in the system. This entails preventing the density of aircraft from becoming too large in certain regions of airspace, and operating efficient reroutes when the weather does not allow traffic to cross a given region of airspace. These tasks are currently not optimized, not even automated. Rather, they are prescribed by playbooks (as defined in Chapter 4).

The goal of this Chapter and the next is to derive a model and a mathematical method to achieve the same effects, in an automated way, and to create an optimization strategy capable of automatically generating more efficient control strategies for these tasks. At this level of description of the system, one is interested in deriving "flow patterns", that is, coming up with ways to route streams of aircraft by generating the corresponding aircraft velocities. The individual identity of the aircraft is thus not important, since the objective of such tasks is to come up with a more efficient use of the airspace, rather than optimizing local trajectories of aircraft. Ideally, one would like to automatically generate Air Traffic Control - friendly procedures of the following kind: "all aircraft on airway 148 at 33,000 ft, fly at 450 kts for the next hour and then accelerate by 25 kts for the next half hour".

This suggests following an *Eulerian* approach advocated by Menon et al. [110, 111] and dividing the airspace into line elements corresponding to portions of airways, on which we can describe the density of aircraft as a function of time and of the coordinate along the line. Such an approach focuses on the conservation of aircraft on the line elements. A traditional way to describe the evolution of the density along these portions of lines is to use a *partial differential equation* (PDE), written in conservation law form. This PDE appears naturally in highway traffic and is called the Lighthill-Whitham-Richards (LWR) PDE [102, 138, 121]. In this work, we will derive a modified version of the LWR PDE specifically applicable to the *Air Traffic Control* (ATC) problem of interest.

The primary goal of this chapter is to show that despite the information loss inherent in any Eulerian model, the aircraft count (which is a crucial ATC metric defined in this chapter) is predicted accurately. In Chapter 7, our goal is to show that fast numerical analysis tools can be applied efficiently to this problem for simulations purposes, and that adjoint based methods advocated by Bewley [35, 52, 36] can be adapted for this real-time network control problem. The main difference between ours and previous work using LWR models of air traffic [110, 111] or highway traffic [62, 120, 154] is that we generate an optimization technique using the continuous PDE directly (instead of its discretization), which enables the use of fast numerical techniques specifically developed to treat first order hyperbolic

PDEs with discontinuities in their solutions. Furthermore, the optimization methodology enables the treatment of constraints in the control and the state.

This Chapter is organized as follows. Since the air traffic flow problem is significantly different from the highway problem, we will first rederive the LWR PDE for the case of interest in Section 6.1 (which shows a modified form of the standard LWR equation), and generalize it to a network. Then, we determine an analytical solution for the case of time-invariant velocity control, which, in Chapter 7, will be used for numerical validation purposes. In Section 6.2, we explain how to identify the numerical values of the parameters for the airspace of interest, using Enhanced Traffic Management System (ETMS) data. Finally, in Section 6.3, we validate the model against real data.

6.1 A new Eulerian network model of airspace

6.1.1 A modified LWR model of air traffic

In describing the air traffic system, like the road system, one has to first look at aircraft (or cars) present in the system and estimate a density of vehicles. Therefore, given a portion of airspace (airway or sector), one needs to introduce the *aircraft count* [39] defined as the number of aircraft in that region. Let us consider a portion airway of length L, described by a coordinate $x \in [0, L]$. The number of aircraft in the segment $[0, x]$ at time t is called $n(x,t)$. Thus, $n(L,t)$ represents the aircraft count on the portion of airway $[0, L]$. Assuming a static mean velocity profile v defined on $[0, L]$, $v(x) > 0$ represents the mean velocity of aircraft at location x, and the motion of an aircraft is described by the dynamical system $\dot{x} = v(x)$.

Introducing $K(x) = \int_0^x \frac{ds}{v(s)}$, it is fairly easy to see that if an aircraft were at location x_0 at time t_0, it would be at x at time $t = t_0 + K(x) - K(x_0)$. Because of the sign of v, K is invertible, and therefore x_0 is related to x, t and t_0 by $x_0 = K^{-1}(K(x) - (t - t_0))$.

Consider a point x and $x + h > x$. The number of aircraft between x and $x + h$ at t can be related to the number of aircraft at t_0 at locations $x_0 = K^{-1}(K(x) - (t - t_0))$ and $x_h = K^{-1}(K(x + h) - (t - t_0))$ (conservation of aircraft): $n(x + h, t) - n(x, t) =$

$n(K^{-1}(K(x+h)-(t-t_0)),t_0) - n(K^{-1}(K(x)-(t-t_0)),t_0)$. In other words, assuming that there is no inflow at 0,

$$n(x,t) = n(K^{-1}(K(x)-(t-t_0)),t_0)$$

Some simple algebra (two successive applications of the chain rule) shows that the space derivative and the time derivative of n are related by:

$$\frac{\partial n(x,t)}{\partial t} + v(x)\frac{\partial n(x,t)}{\partial x} = 0$$

We recognize this as a first order linear hyperbolic PDE, and can now enunciate the following proposition:

Proposition 3. *Let $v(\cdot) : [0,L] \to \mathbb{R}^+$ be a $PC_1([0,L])$ function (i.e. piecewise differentiable) with a finite number of discontinuities at $\{x_k\}_{k\in\{0,\cdots,K\}}$ on $[0,L]$ such that $\exists m > 0$, $m \le v(x)$ for all $x \in [0,l]$. Let $q^{in} \in C_0([0,T])$ and $n_0 \in C_0([0,L])$. Then the following PDE*

$$\begin{cases} \frac{\partial n(x,t)}{\partial t} + v(x)\frac{\partial n(x,t)}{\partial x} = q^{in}(t) & \text{in } [0,L]\times(0,T] \\ n(x,0) = n_0(x) & \text{in } [0,L]\times\{0\} \\ n(0,t) = 0 & \text{in } \{0\}\times(0,T] \end{cases} \tag{6.1}$$

admits a unique continuous (weak) solution, given by:

$$\begin{array}{ll} n(x,t) = n_0\left(K^{-1}(K(x)-t)\right) + \int_0^t q^{in}(u)du & \text{if } t \le \int_0^x \frac{du}{v(u)} \\ n(x,t) = \int_{t-K(x)+K(0)}^t q^{in}(u)du & \text{if } t \ge \int_0^x \frac{du}{v(u)} \end{array} \tag{6.2}$$

where $K(x) = \int_0^x \frac{du}{v(u)}$, and K^{-1} is its inverse.

Proof — Existence: K is well defined because $v(x) \ge m$ for all $x \in [0,l]$. Its inverse exists because K is (strictly) increasing. It is easy to check that (6.2) satisfies (6.1) almost everywhere, and that it is continuous. This solution has been constructed using a technique analogous to the algorithm of [26] (also shown in Chapter 8, based on the method of characteristics).

Uniqueness: Let us call n_1 and n_2 two continuous weak solutions of (6.1). Call $\delta := n_1 - n_2$. δ satisfies: $\frac{\partial\delta}{\partial t} + v(x)\frac{\partial\delta}{\partial x} = 0$ a.e. in $[0,L]\times(0,T]$, $\delta(x,0) = n_0(x)$ in $[0,L]\times\{0\}$ and

$\delta(0,t) = 0$ in $\{0\} \times (0,T]$. Multiplying this PDE by δ and integrating from $x_0 = 0$ to the first discontinuity x_1 of $v(\cdot)$ gives:

$$\int_{x_0}^{x_1} \delta(u,t)\frac{\partial \delta}{\partial t}(u,t)du + \int_{x_0}^{x_1} v(u)\delta(u,t)\frac{\partial \delta}{\partial x}(u,t)du = 0$$

from which we deduce

$$\frac{1}{2}\frac{d}{dt}\int_{x_0}^{x_1} \delta(u,t)^2 du + \int_{x_0}^{x_1} v(u)\delta(u,t)\frac{\partial \delta}{\partial x}(u,t)du = 0$$

Integrating by parts gives

$$\frac{1}{2}\frac{d}{dt}\int_{x_0}^{x_1} \delta(u,t)^2 du \leq \int_{x_0}^{x_1} v'(u)\frac{1}{2}\delta(u,t)^2 du - \left[v(u)\frac{1}{2}\delta(u,t)^2\right]_{x_0}^{x_1} \leq \int_{x_0}^{x_1} v'(u)\frac{1}{2}\delta(u,t)^2 du$$

since $\delta(x_0,t) = 0$ and $v(x_1) > 0$. Using the fact that $v(\cdot) \in C_1([x_0,x_1])$, $\exists M > 0$, $|v'(x)| \leq M$ for all $x \in [x_0,x_1]$, from which we deduce

$$\int_{x_0}^{x_1} v'(u)\frac{1}{2}\delta(u,t)^2 du \leq M\int_{x_0}^{x_1} \frac{1}{2}\delta(u,t)^2 du$$

we can write

$$\frac{1}{2}\frac{d}{dt}\int_{x_0}^{x_1} \delta(u,t)^2 du \leq M\int_{x_0}^{x_1} \frac{1}{2}\delta(u,t)^2 du$$

Using Gronwall's lemma, this implies $\delta(x,t) = 0$ almost everywhere in $[x_0,x_1]$. By continuity, $n_1(x,t) = n_2(x,t)$ everywhere in $[x_0,x_1]$, and therefore at x_1. The same proof applies to $[x_1,x_2]$ since $n_1(x_1,t) = n_2(x_1,t)$ for all t. By induction on x_k, they are equal everywhere in $[x_0,x_k]$ and therefore in $[0,L]$. $\square$

In equation (6.1), q^{in} represents the inflow at the entrance of the link (i.e. at $x = 0$). In highway traffic flow analysis, n is sometimes referred to as cumulative flow. It can be related to the vehicle density through the integral relation

$$n(x,t) = \int_0^x \rho(u,t)du \tag{6.3}$$

where $\rho(x,t)$ is the vehicle density. It can be checked that the vehicle density satisfies the following PDE:

$$\begin{cases} \frac{\partial \rho(x,t)}{\partial t} + \frac{\partial}{\partial x}(\rho(x,t)v(x)) = 0 \\ \rho(0,t)v(0) = q^{\text{in}}(t) \\ \rho(x,0) = \rho_0(x) \end{cases} \tag{6.4}$$

Equation (6.4) can be related to equation (6.1) by a simple integration of ρ along $[0,x]$. Equation (6.4) is a mass conservation equation, written in conservation law form. This equation is very closely related to the original LWR PDE [102, 138, 121].

The LWR PDE, originally developed for highways, in fact reads $\frac{\partial \rho(x,t)}{\partial t} + \frac{\partial}{\partial x}(q(\rho(x,t))) = 0$, where $q(\cdot)$ is a flux function depending on ρ, which relates the car density on the highway to the flux. In practice, $q(\cdot)$ is empirically determined, and several models of $q(\cdot)$ are currently used [5, 62, 55]. Computation of the numerical value of the parameters associated with these flux functions is a difficult task, which can for example be achieved with Kalman filtering techniques [154]. In the present case, the flux function $q(\cdot)$ is replaced by a mean velocity $v(\cdot)$ multiplied by the density. In the next chapter, $v(x)$ will also depend on t and will be the control input of the system.

It is also possible to rewrite the first equation in (6.4) as

$$\frac{\partial(\rho(x,t)v(x))}{\partial t} + v(x)\frac{\partial}{\partial x}(\rho(x,t)v(x)) = 0$$

which provides the following corollary:

Corollary 2. *The corresponding solution for ρ is given by:*

$$\rho(x,t) = \begin{cases} \rho(K^{-1}(K(x)-t),0)\frac{v(K^{-1}(K(x)-t))}{v(x)} & \textit{if } t \leq K(x) \\ \frac{q^{in}(t-K(x))}{v(x)} & \textit{otherwise} \end{cases} \tag{6.5}$$

Note that a more convenient way to write the solution for $t \leq K(x)$ is $\rho(x,t) = \rho(x_0(x,t),0)\frac{v(x_0)}{v(x)}$ where $x_0 = K^{-1}(K(x)-t)$ is the origin of the characteristic curve of the system in the (x,t) plane, going through x at t.

The interpretation of the corollary is the following: the quantity ρv is conserved along the characteristic curves $t - t_0 = K(x) - K(x_0)$.

At this stage, ρ is defined by $\rho = \frac{\partial n}{\partial x}$ and satisfies (6.4). However, unlike for highway traffic, the density ρ might not be the best way to characterize the flow situation at a given time: if the number of aircraft in the system is small, ρ will be a set of spikes, which is intractable numerically. Therefore, a more tractable quantity to work with would be $\frac{\delta n}{\delta x}$, where δn represents the number of aircraft contained in a finite interval of length δx. This quantity does not a priori satisfy the PDE (6.4).

It is meaningful to introduce an additional "density-like" quantity called r, which satisfies the PDE and for which we can suggest a physical interpretation.

$$r(x,t) = \frac{1}{2t_{\text{ref}}v(x)}[n(K^{-1}(K(x)-(t-t_{\text{ref}}))) - n(K^{-1}(K(x)-(t+t_{\text{ref}})))] \tag{6.6}$$

where t_{ref} is a reference time. $r(x,t)v(x)$ represents the number of aircraft included into a window of $2t_{\text{ref}}$ time units of location x and can be referred as "time density". This way of accounting for density is meaningful for Air Traffic Control, since it incorporates a time scale t_{ref} into the density computation and thus provides access to the time separation between aircraft. It is easy to show that r itself satisfies the same PDE as ρ for any value of t_{ref}:

$$\frac{\partial r(x,t)}{\partial t} + \frac{\partial (r(x,t)v(x))}{\partial x} = 0$$

One can also show that in the limit $t_{\text{ref}} \to 0$, r and ρ are the same:

$$\begin{aligned}\lim_{t_{\text{ref}}\to 0}\left[\frac{n(K^{-1}(K(x)-(t-t_{\text{ref}}))) - n(K^{-1}(K(x)-(t+t_{\text{ref}})))}{2t_{\text{ref}}v(x)}\right] &= \lim_{t_{\text{ref}}\to 0}\frac{n(x,-t_{\text{ref}}) - n(x,t_{\text{ref}})}{2t_{\text{ref}}v(x)}\\ &= \frac{1}{v(x)}\left(-\frac{\partial n(x,t)}{\partial t}\right)\\ &= \frac{1}{v(x)}v(x)\frac{\partial n(x,t)}{\partial x}\\ &= \rho(x)\end{aligned}$$

At this stage, we have three quantities: ρ, $\frac{\delta n}{\delta x}$ and r. The meaning of ρ as we know it in fluid mechanics assumes a large number of particles (i.e. aircraft) per unit volume. The threshold number is sometimes defined by the Knüdsen number. In the present case, the number of aircraft we consider will almost certainly be below this number, meaning that the fluid approximation is questionable. This means that instead of using $\rho = \frac{\partial n}{\partial x}$, we will

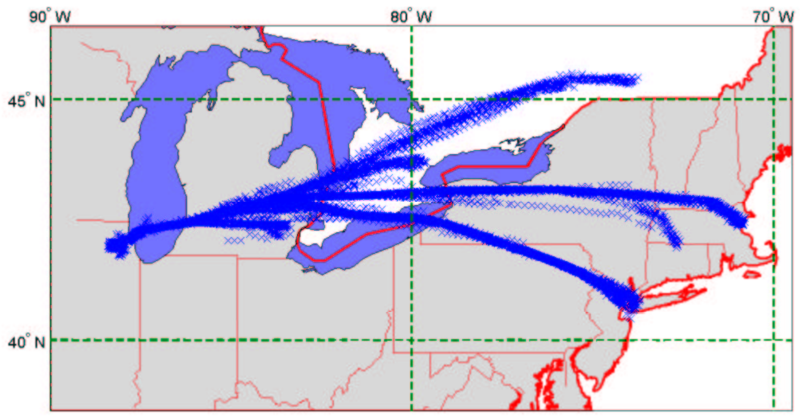


Figure 6.1: Tracks of flights incoming into the Chicago ORD airport from the east coast and Canada. The upper stream comes from Canada, the lower respectively from New York and Boston. Additional streams merge into the network (Detroit and Hartford Bradley). The data comes from ETMS. A diagram of the network is shown in Figure 6.2.

use $\rho \sim \frac{\delta n}{\delta x}$ in the PDE: we will justify this approximation with appropriate validations. In particular, we will have to make a choice of a numerical parameter called $L_{\text{ref}} := \delta x/2$. This will be done in Section 6.2. We then will validate the model against real data to show its accuracy and predictive capabilities (Section 6.3).

6.1.2 Network model

The model of the previous section describes traffic on a single portion of airway or line element. As was done earlier for highways [63], this model can be generalized to airway networks, i.e. sets of interconnected airways. Figure 6.1 shows aircraft tracks for inbound traffic into Chicago (ORD) from the east coast.

We now derive a framework to describe unidirectional air traffic in a network such as the network shown in Figure 6.1. We describe the topology of the network by a unidirectional graph (E, V), in which E is the set of edges or links, and V the set of vertices. For simplicity of notation, we will index the links by $i \in \{1, \cdots, N\}$, rather than by the indices of the two corresponding vertices. For all $i \in \{1, \cdots, N\}$, we call $\mathcal{U}(i)$ the set of upstream links merging into link i, and $\mathcal{M}$ the set of links for which the upstream links are only

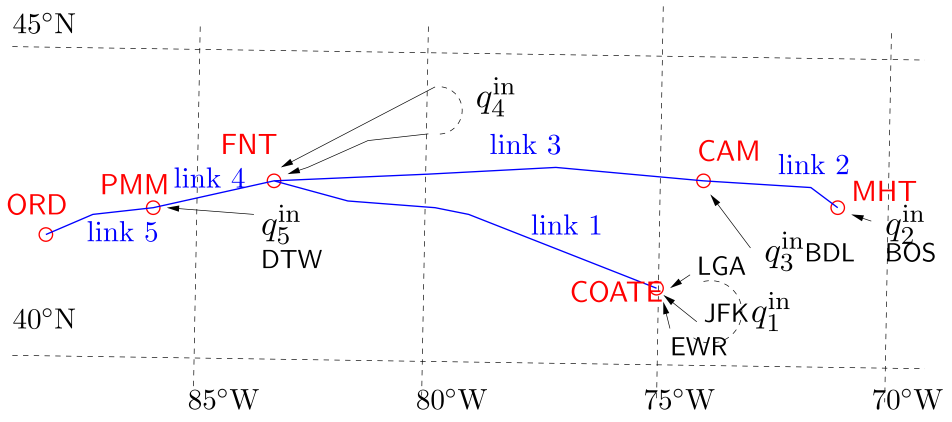


Figure 6.2: Network model for the tracks shown in Figure 6.1 with waypoints labeled. The model includes five links, merging into Chicago (ORD). The corresponding inflow terms correspond to a single airport as in Boston (Boston BOS) or Detroit (DTW), or to a set of airports, as in New York (Newark EWR, John F. Kennedy JFK, La Guardia LGA).

merging. The number of links merging into a single link is not limited; it is possible to have $|\mathcal{U}(i)| > 2$.

If there is a divergence at the end of a link i, we assume for simplicity that there are only two emanating links from the corresponding vertex. We index by i_l and i_r the two emanating links (left and right), and call β_i the portion of the flow going from i to i_l, and $1 - \beta_i$ the proportion of the flow going from i to i_r. We call $\mathcal{D}$ the set of links with a divergence at the end of it. The β_i are not known a priori and have to be determined. These coefficients might depend on t as well, and therefore a dependence $\beta_i(t)$ is included in the model.

We call $\mathcal{S}$ the set of sources in the network, and $\mathcal{T}$ a sink of the network, at which we might want to perform optimization. We index all variables of the previous section by i: the aircraft density on link i is ρ_i, the coordinate is x_i, the main velocity profile is v_i, etc. Note that we are NOT using Einstein's notation. The notation is summarized in Table 6.1. The governing PDE system thus reads:

N	number of links
$\mathcal{S}$	set of source links
$\mathcal{M}$	set of links into which other links merge
$\mathcal{D}$	set of links ending in a fork
$\mathcal{U}(i)$	set of links merging into link i (if $i \in \mathcal{M}$)
i_l, i_r	indices of the two links of a fork if link $i \in \mathcal{D}$
L_i	length of link i
x_i	arclength on link i: $x_i \in [0, L_i]$
$\rho_i(x_i, t)$	aircraft density on link i
$\rho_i^\circ(x_i)$	initial aircraft density on link i
$v_i(x_i)$	nominal velocity profile on link i: $v_i(\cdot) : [0, L_i] \to \mathbb{R}^+$
$q_i^{\text{in}}(t)$	inflow at $x_i = 0$ for link i (if applicable)
$\beta_i(t)$	portion of ρ_i which flows into link i_l (if applicable)

Table 6.1: Notation for the network problem.

$$\left\{\begin{array}{ll} \frac{\partial \rho_i(x_i,t)}{\partial t} + \frac{\partial}{\partial x_i}\left(\rho_i(x_i,t)v_i(x_i)\right) = 0 & \forall i \in \{1, \cdots, N\} \\ \rho_i(x,0) = \rho_i^\circ(x) & \forall i \in \{1, \cdots, N\} \\ \rho_i(0,t)v_i(0,t) = \sum\limits_{j \in \mathcal{U}(i)} \rho_j(L_j,t)v_j(L_j,t) + q_i^{\text{in}}(t) & \forall i \in \mathcal{M} \\ \left\{\begin{array}{l} \rho_{i_l}(0,t)v_{i_l}(0,t) = \beta_i(t)\,\rho_i(L_i,t)v_i(L_i,t) \\ \rho_{i_r}(0,t)v_{i_r}(0,t) = (1-\beta_i(t))\rho_i(L_i,t)v_i(L_i,t) \end{array}\right. & \forall i \in \mathcal{D} \\ \rho_i(0,t)v_i(0,t) = q_i^{\text{in}}(t) & \forall i \in \mathcal{S} \end{array}\right. \tag{6.7}$$

In the previous system, the PDE (first equation) describes the evolution of ρ_i on each link. The second equation is the initial condition (i.e. the initial density of aircraft on each link). The third equation expresses the conservation of aircraft at the merging points. The fourth and fifth equation express the conservation of aircraft at the divergence points. The last equation expresses the boundary conditions (inflow at the sources of the network). The sinks of the system are free boundary conditions, and therefore do not appear in the previous system.

For the example of Figures 6.1 and 6.2, $N = 5$, the source links are $\mathcal{S} = \{1, 2\}$, the merging points are $\mathcal{U}(5) = \{4\}$, $\mathcal{U}(4) = \{1, 3\}$, $\mathcal{U}(3) = \{2\}$, the sink is at the end of link 5, each link has an inflow term q_i^{in}. For the link 1, three airports are grouped into a single cumulative inflow q_1^{in} (airports LGA, JFK, EWR). For this particular example, there is no diverging point: $\mathcal{D} = \emptyset$. Diverging points will be added in the presence of control strategies, for which some of the flows have to be split among two airways (see next chapter).

Assuming one can solve (6.7), it is possible to use the solution to compute (and optimize) certain metrics useful for ATC. For example, one quantity of interest is aircraft count, which is the number of aircraft in a given sector. If all links of a given sector are indexed by $i \in \text{Sec}$, the aircraft count of the sector is obtained by $\sum_{i \in \text{Sec}} \int_0^{L_i} \rho_i(x_i, t) dx_i$.

6.2 Application to air traffic flow

In this section, we first explain how to identify the mean velocity profiles from real Air traffic data. We then explain how to choose the numerical value of parameters of the model. In the next section, these choices will be validated against real data.

6.2.1 System identification: main velocity profiles

We first explain how we identify the mean velocity profiles $v_i(x_i)$ on each link. We use *Enhanced Traffic Management System* (ETMS) data, which we can obtain at NASA Ames. ETMS database contains all flight plan information for flights in the National Airspace System (NAS). Data are collected from the entire population of flights in the NAS with filed flight plans. ETMS data is sent from the Volpe National Transportation System Center (VNTSC) to registered participants via the *Aircraft Situation Display to Industry* (ASDI) electronic file server. The Federal Aviation Administration (FAA) uses these data to monitor the effectiveness of its National Route Program, in which the user community is offered flexible, cost-effective routing options as an alternative to published ATC preferred routes.

From ETMS data, we can obtain useful flight information at a 1 minute rate: position of each aircraft in the NAS, altitude, velocity, flight plan (i.e. set of airways and waypoints). From this data, we are able to identify the routes in which traffic is concentrated. Note that in recent work, Menon et al. [110, 111] focused on creating a tool which performs similar tasks automatically at a NAS-wide level, using FACET [39], a tool developed by NASA

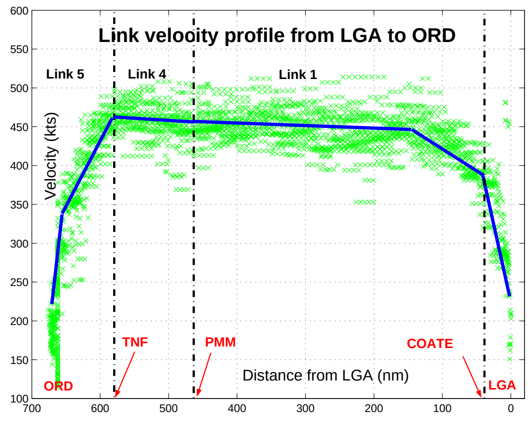


Figure 6.3: Example of velocity profile used for the junction New-York (LGA) to Chicago (ORD). The horizontal coordinate is the distance from the destination Airport (ORD) in nautical miles. The corresponding links are shown as well as the location of the airspace fixes between the links. The vertical coordinate is the velocity tracks (knots) from the ETMS data set. The curve is a piecewise affine fit obtained using least squares. Each of the tracks corresponds to one aircraft at a given time at a given location, over a period of 24 hours. The total number of aircraft used for this plot is 42.

Ames. The details of this tool are not available to us, and we developed our own method to identify the main links used by aircraft in the NAS.

We analyze 24 hours of ETMS data and select all aircraft using the links of the network shown in Figures 6.1 and 6.2. We identify all aircraft which use each of the links, and record all tracks and corresponding speeds between takeoff and landing. For each of the links shown in Figure 6.1, we identify the mean velocity profiles as piecewise affine function, using a least squares fit. The total number of aircraft used is 220. The result for the flight New York – Chicago is displayed in Figure 6.3 and Table 6.2. As can be seen, once the en route altitude is reached, the curve fits are almost flat, which means that the aircraft are en route at a high altitude cruise speed.

It can also be seen from Figure 6.3 that the data is relatively broadly spread (standard deviation 19.6 kts). This suggests deriving more precise models, which are multilayer: dividing the link in sublayers corresponding to altitudes has the benefit of being more precise, as aircraft tend to have a Mach number – and therefore speed which is a function of altitude. This will be considered for future work.

Link	L_i	range (nm)	$v_i(x_i)$ (kts)
1	470.6	$[0, 40]$	$289.7 + 2.4x_1$
		$[40, 145]$	$385.7 + 0.53(x_1 - 40)$
		$[145, 470.6]$	$441.3 + 0.046(x_1 - 145)$
2	137.7	$[0, 30]$	$320.7 + 0.51x_2$
		$[30, 90]$	$344.2 + 1.62(x_2 - 30)$
		$[90, 137.7]$	$441.38 + 0.045(x_2 - 90)$
3	417.1	$[0, 417.1]$	$443.5 + 0.006x_3$
4	112.8	$[0, 112.8]$	$445.8 + 0.2x_4$
5	98.9	$[0, 85]$	$468.3 - 1.8x_5$
		$[85, 98.9]$	$315.3 - 3.6(x_5 - 85)$

Table 6.2: Velocity profiles and characteristics along the links of the network shown in Figure 6.1.

6.2.2 Identification of the initial and boundary conditions

Once the mean velocity profiles are computed, we identify the initial density of aircraft and the inflow (boundary conditions) in the network.

The initial position of the aircraft is easy to extract from the ETMS data: at the prescribed time, all airborne aircraft which are on the relevant links are selected.

- For any selected aircraft a at location x_i^a on link i, the classical density $\rho_i(x_i, 0) = \rho_i^\circ(x_i)$ is taken to be a "box" around x_i^a, of length $2L_{\text{ref}}$. Calling χ_I the characteristic function of an interval I (i.e. equal to 0 outside of I and 1 inside), $\rho_i(x_i, 0)$ is $\frac{1}{2L_{\text{ref}}}\chi_{[x_i^a - L_{\text{ref}}, x_i^a + L_{\text{ref}}]}(x_i)$. Taking all aircraft initially airborne on link i, the density is:

$$\rho_i^\circ(x_i) = \sum_{a \text{ in link } i} \frac{1}{2L_{\text{ref}}}\chi_{[x_i^a - L_{\text{ref}}, x_i^a + L_{\text{ref}}]}(x_i)$$

- Similarly, the density-like function r is computed using the knowledge of the mean velocity profile along link i, called $v_i(x_i)$, and the parameter t_{ref}:

$$r_i^\circ(x_i) = \sum_{a \text{ in link } i} \frac{1}{2t_{\text{ref}}v_i(x_i^a)}\chi_{[x_i^a - t_{\text{ref}}v_i(x_i^a), x_i^a + t_{\text{ref}}v_i(x_i^a)]}(x_i)$$

The inflows (boundary conditions) can also be extracted from ETMS data: each time an aircraft takes off, it will appear in the ETMS data as soon as it is airborne. The ETMS data also shows the filed flight plan, which we select when it intends to use the links of interest to us. $q^{\text{in}}(t)$ is computed the following way. At any instant when the data shows a new aircraft on one of the source links $\mathcal{S}$, the track is in general passed the entrance point of that link (because of the sampling rate of 3 minutes). Calling x_i^a the position of this aircraft on link i at the first time it appears, we compute the time t_a at which it crossed the location $x_i = 0$ (using the knowledge of the mean velocity profile on the link). We then use one of the two definitions above to compute $q^{\text{in}}(t)$ corresponding to either ρ or r.

In this chapter, there is no diverging link, therefore there is no need to identify the β_i. In Chapter 7, the β_i will be a control parameter, which we will have to determine. Note that for networks with diverging links, determining the $\beta_i(t)$ is a very difficult task, as it depends on the final destination of the aircraft, and therefore cannot be modeled in a totally Eulerian framework. This difficulty was already pointed out by Menon [110, 111] and is also known for highway traffic [63]. For control problems, since β_i is a control parameter, this difficulty partially disappears, since we have the freedom to change it to achieve the desired effects, while maintaining the proper outflow at the sink(s).

6.2.3 Identifying the numerical parameters

As explained in Section 6.2.2, we have two ways of describing the density of aircraft in the network, in terms of a density function ρ and a "density-like" function r, which respectively account for spatial and temporal distribution of aircraft. The function ρ depends on the numerical parameter L_{ref}, which we need to adjust. The value of this parameter is crucial for predicting aircraft count: Figure 6.4 shows how errors can occur in translating density functions into aircraft count. We want to determine the choice of parameters leading to the smallest error in aircraft count prediction.

We first run the following set of experiments. For the link New York – Chicago, we run a set of simulations involving N_{aircraft} aircraft, where N_{aircraft} successively takes all values between 1 and 50. We vary L_{ref} between 0 and 120 nm. For each value of N_{aircraft} and L_{ref}, we run 400 experiments. Each experiment corresponds to a uniformly distributed random density of N_{aircraft} aircraft along link 1 in $[0, 400]$ (see Figure 6.2). The simulation starts at a time t_0 with the density ρ_i° computed as in the previous section, and computes the solution of the LWR PDE until the time $t_0 + \Delta T$. For the experiments, ΔT was chosen equal to one

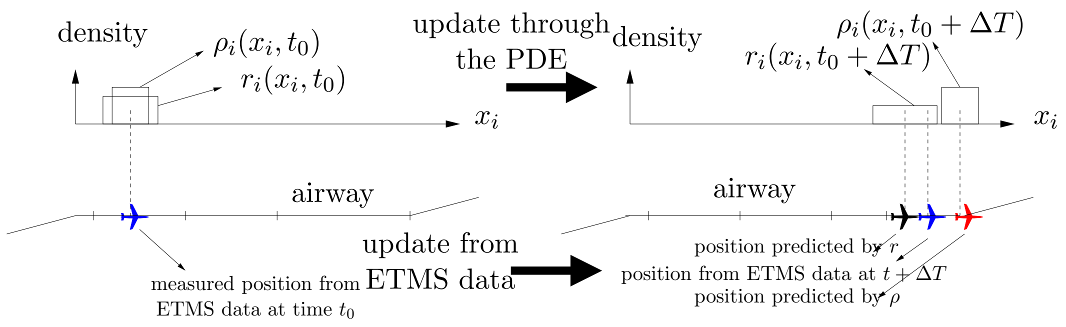

density
$\rho_i(x_i, t_0)$
$r_i(x_i, t_0)$
x_i
update through the PDE
density
$\rho_i(x_i, t_0 + \Delta T)$
$r_i(x_i, t_0 + \Delta T)$
x_i
airway
airway
update from ETMS data
measured position from ETMS data at time t_0
position predicted by r
position from ETMS data at $t + \Delta T$
position predicted by ρ

Figure 6.4: Different predictions obtained by the use of ρ and r for aircraft density. Above: density propagation through the PDE system (6.7); below: position update form ETMS data and from the result of the PDE.

hour (note that the duration of the total flight is on the order of two and a half hours). This solution is compared with the solution obtained by propagating the Lagrangian trajectories of each of the aircraft independently from t_0 to $t_0+\Delta T$ and computing the resulting density. In mathematical terms, we compare the two following quantities

- $\rho_i(\cdot, t_0 + \Delta T)$ computed by the LWR PDE (6.7)
- $\tilde{\rho}_i(\cdot, t_0 + \Delta T) := \sum_{a \text{ in link } i} \frac{1}{2L_{\text{ref}}} \chi_{[x_i^a(t_0+\Delta T)-L_{\text{ref}}, x_i^a(t_0+\Delta T)+L_{\text{ref}}]}(\cdot)$ where $x_i^a(t_0 + \Delta T)$ is the position of aircraft a at time $t_0 + \Delta T$.

In order to characterize the best choice of numerical parameters, we compute the following two quantities:

- The relative density error, defined by

$$\frac{\sum\limits_{i=1,4,5} \int_0^{L_i} |\rho_i(x_i, t_0 + \Delta T) - \tilde{\rho}_i(x_i, t_0 + \Delta T)|\, dx_i}{\sum\limits_{i=1,4,5} \int_0^{L_i} \rho_i(x_i, t_0 + \Delta T) dx_i}$$

  This quantity represents the error in density prediction due to the propagation of ρ by the PDE.

- The absolute aircraft count error, defined by

$$\sum_{i=1,4,5} \quad \sum_{\text{sublinks of } i} \left| \int_0^{L_i} \rho_i(x_i, t_0 + \Delta T) dx_i - \sharp(a | a \in \text{link } i) \right|$$

where $\sharp$ means number. This quantity is the sum of count error for all sublinks of links 1,4, and 5. Typically, a link is divided into sublinks which correspond to different airspace sectors. For example, if link 1 goes through 8 sectors, we divide it in 8 sublinks and are interested in the aircraft counts on these sublinks. This error thus estimates the difference between the number of aircraft predicted by the PDE and the number obtained by a Lagrangian propagation of aircraft, where the error is the sum of all errors on the sublinks.

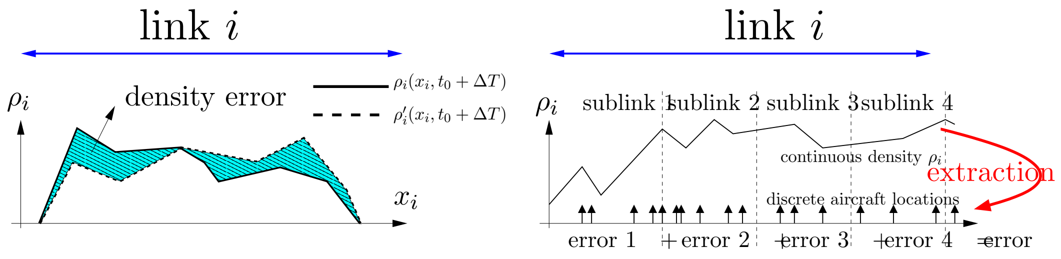


Figure 6.5: **Left:** Illustration of the computation of the relative density error depicted in Figure 6.6. The difference between the two density curves (shaded area) is divided by the area below the ρ_i curve. **Right:** Illustration of the computation of the error in aircraft count. The link is divided into sublinks (which can correspond to sectors). For each of these sublinks, we compare the number of aircraft predicted by the method (depicted by arrows, which are computed from the density) with the number of aircraft obtained by a Lagrangian propagation of the trajectories. The error is the sum of errors for all sublinks, i.e. the sum of the errors in sector counts.

The computation of both quantities is illustrated in Figure 6.5. The relative density error and absolute aircraft count error are averaged (over the 400 runs) and plotted for the range of n and L_{ref} considered. The result is shown in Figure 6.6. The left plot shows the relative density error. As expected, the error decreases when the number of aircraft increases and L_{ref} increases (typically in fluid mechanics, the Knüdsen number defines the number of particles per volume above which the fluid approximation becomes valid). The right plot shows the absolute aircraft count error, averaged over 400 simulations. For this plot, each of the links 1, 4 and 5 have been divided in sublinks (20 total), of about 50 nm length. This is a worst case scenario, i.e. the number of relevant sectors for a flight of this length is never higher. One can see that for $L_{\text{ref}} \leq 60$ and $N_{\text{aircraft}} \geq 25$, the average aircraft count error is always extremely small.

The best choice for L_{ref} is thus obtained at the intersection of the lowest level sets of both plots of Figure 6.6, i.e. for a range of $L_{\text{ref}} \in [20, 60]$ and $N_{\text{aircraft}} \geq 20$.

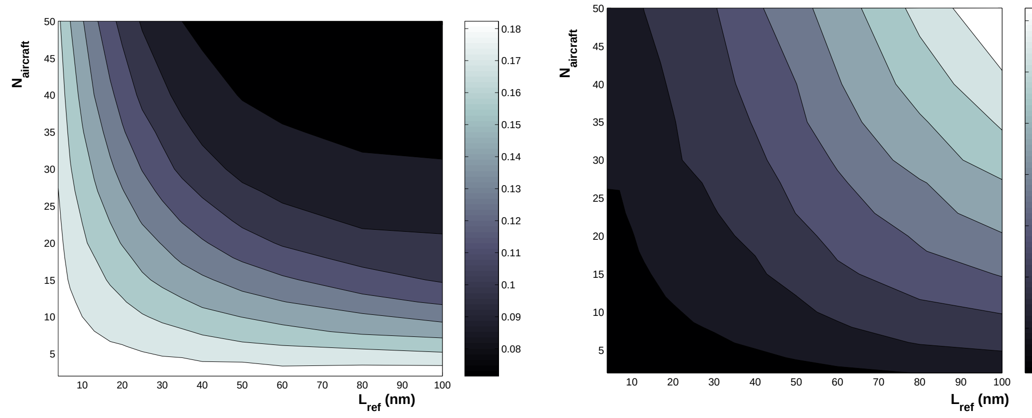


Figure 6.6: **Left:** Relative density error between the density predicted by the Eulerian PDE propagation of the density. **Right:** Absolute aircraft count error for the junction New York – Chicago.

6.3 Validation of the model

In the previous section, we have shown that the use of the modified LWR PDE either with r (with any t_{ref}) or $\rho \sim \frac{\delta n}{\delta x}$ (with an appropriate choice of L_{ref}) enables accurate aircraft count predictions. In this section, we validate the model against real data: in particular, we had made the assumption that $v_i(x_i, t) = v_i(x_i)$. As we know, this will not be the case in general. Therefore we run simulations and assess if the model is still in agreement with real data over two separate experiments.

Static validation

In the first experiment, we use the static velocity profiles $v_i(\cdot)$ determined in the previous sections for the validation of the method. We use a 6 hour ETMS data set. From this data set, we extract the position of the aircraft, at the initial time, construct the corresponding initial aircraft density, and propagate it through the PDE system. At any given time, we compare the aircraft count predicted by our method and the aircraft count provided by the ETMS data (which is exact, since it provides the position of each aircraft). We compute the error in aircraft count for a set of ten sublinks for the network shown in Figure 6.2. The result is shown in Figure 6.7 (left). The window width L_{ref} was taken equal to 15 nm. One can see on the left plot that the total error (for all airborne aircraft in this airspace) is relatively low (the maximal error is 7 aircraft). In fact, the results are much better than

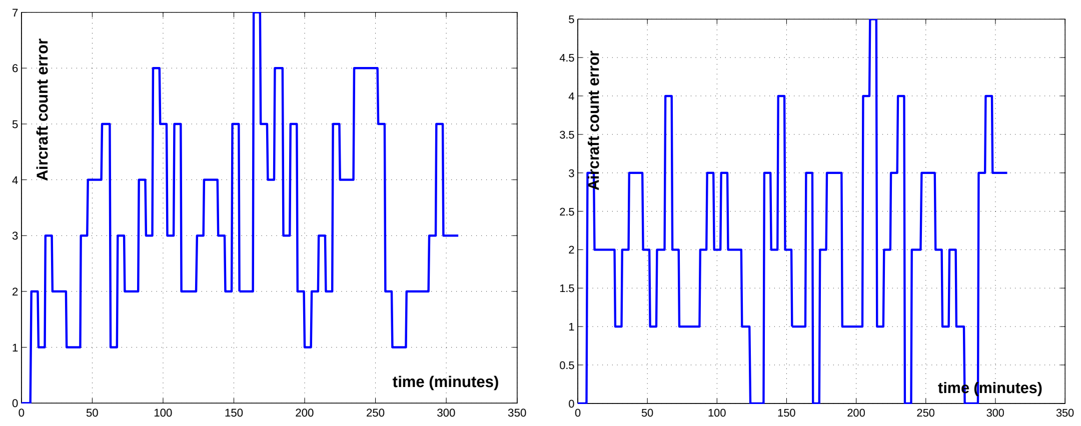


Figure 6.7: **Left:** Error in aircraft count for the static validation over a five hour period. **Right:** Error in aircraft count for the dynamic validation over a five hour period.

they seem: most of the errors come from the fact that the aircraft distribution is such that there is always at least one or two aircraft close to a sublink boundary, which will thus be counted in the wrong sublink. In fact, this is not really a problem, as it is more an artifact of the computation rather than a true error (Figure 6.8 shows that the density unambiguously shows where the aircraft is). Furthermore, some of the errors in aircraft count are due to errors present in the ETMS data (some have clearly erroneous data; this fact has also been reported in [53]).

Dynamic validation

We extend the validation to a case in which the velocity profiles are time dependent, i.e. $v_i(x_i, t)$. The details of the identification of these profiles are technical and are not explained here. The comparison is the same as in the static case. Note that our analytical solution does not handle dynamic velocities. This difficulty is alleviated by using a numerical scheme presented in Chapter 7. The results are shown in Figure 6.7 and are more accurate than the static results, as expected. The same remarks apply, and the results are again affected by the quality of ETMS data and the inclusion of the computation artifact. The only weakness of this validation is that the simulation is run using data from the same day as the data used in identification. A way to improve this would be to perform the velocity identification with data of a given day over a 24 hour period, and validate it over the next 24 hour period, using the fact that there is periodicity in the traffic for normal days [84]. This was not done here due to lack of available data. An animation (.avi movie file) corresponding to the snapshots of Figure 6.7 is available at [157].

In both cases, the validation is very encouraging and shows strong predictive capability for our model. The model was also tested successfully using data from the western states (Oakland Center with traffic incoming into Bay Area airports), though for brevity these results are not included here. Finally, in Chapter 7, we will use the model for control: in that case, one of the control variables is speed, which means that we will have direct access to $v_i(x_i, t)$ (since we compute it). This alleviates difficulties of mean velocity profile identification shown before.

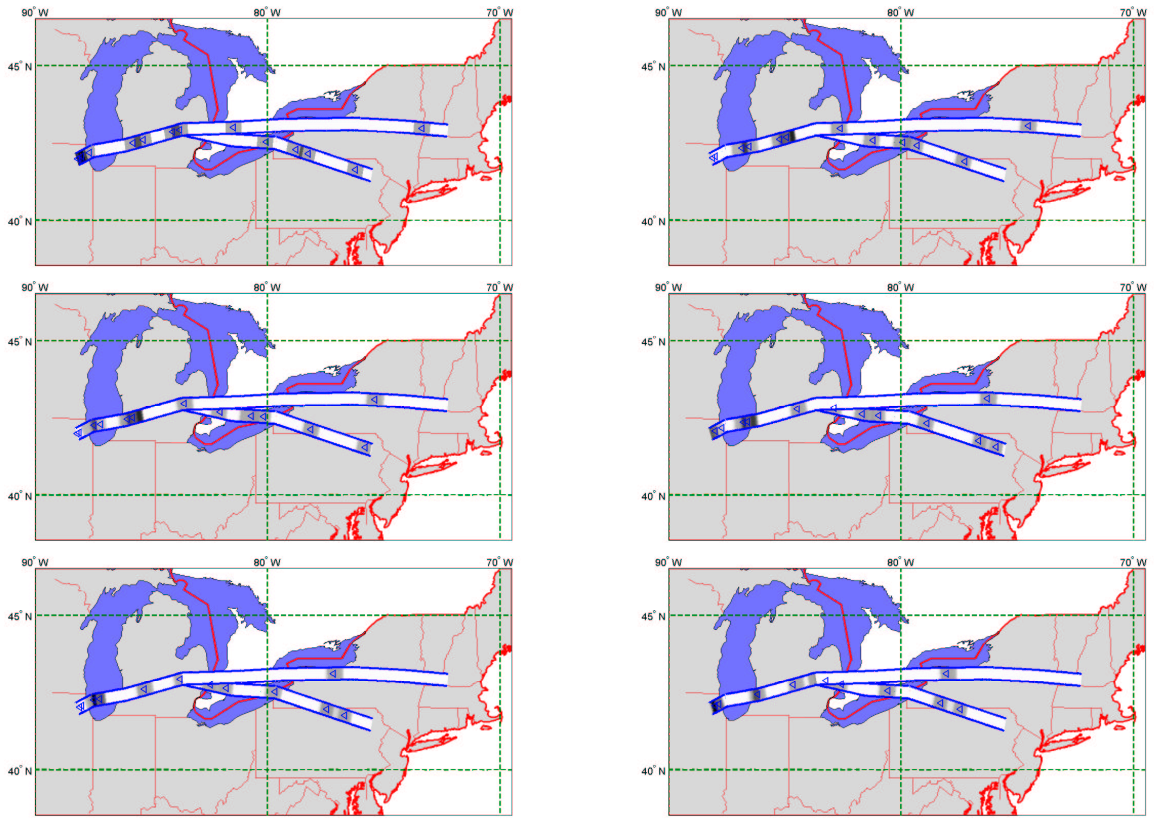


Figure 6.8: Display of the traffic situation for the static validation. The density of the links is depicted by the color. The colored rectangles shown in this plot represent the density. The colorscale is: white for zero density; black for highest density. The actual aircraft positions are superimposed (triangles). Traffic is shown at t_0 (top left), $t_0 + 8\,\text{min}$ (top right), $t_0 + 16\,\text{min}$ (middle left), etc. As can be seen, the peaks of density corresponds to the actual positions of the aircraft.

6.4 Concluding modeling remarks

In this Chapter, we have derived an Eulerian model of the airspace based on a modified LWR partial differential equation. The network structure of the airspace was modeled as a set of coupled LWR PDEs. Given initial positions of aircraft and airports inflows, this system of PDEs enables the prediction of the aircraft density at further times. An analytical solution was derived for a single link in the case in which the mean velocity profiles of aircraft along airways do not vary with time (just with space). It can be used for multiple links as well. ETMS data was used to identify the numerical parameters associated with this model. The data is also used to validate the model, i.e. to demonstrate good predictive capability of this method. Future work might incorporate a recently developed tool [111], which automatically identifies network links and the corresponding velocity profiles.

Note that in addition to the ATC system presented in this chapter and [110, 22, 21], numerous physical systems are best modeled by PDEs, which in turns make their control challenging, as PDE control is still an unexplored field. The highway system may be modeled by the LWR PDE, which inspired the ATC work [102, 138, 121]. The governing PDE is a first order hyperbolic conservation law (in fact very similar to Burger's equation), which makes the concept of entropy solution directly applicable [70] to solution selection. An analogous problem to the ATC problem shown here, or the highway network problem, is the irrigation channel problem, which is also a network problem [108, 104] governed by a conservation law with diffusive term (Saint-Venant equation). Several other similar PDE driven systems are available in [132], with corresponding control strategies (telegraphist equation, Burgers equation, wave equation). In general, as soon as one expands to more complicated PDEs or systems of PDEs, the application fields become more numerous, with several examples in fluid flow control [35, 79], biology [4, 95], process engineering [40, 132] thermal manufacturing [66], solid mechanics [156].

In the next chapter, we will show how efficient numerical schemes can be used to perform numerical simulations of the system of LWR PDEs, even in the case in which the velocity profiles are time dependent. Finally, we will show that adjoint-based techniques enable the control of this model, through velocity and routing assignments. The outputs of this technique could thus be used by Air Traffic Controllers as high level policies for speed assignments and routing in congested airspace.

Chapter 7

Adjoint-based constrained control of Eulerian networks

In Chapter 6, we have described how to model the National Airspace System (NAS) using an Eulerian framework, inspired by the work of Menon et al. [110, 111]. The result of this chapter was a model of airspace as a network of interconnected links, on which the aircraft density is governed by a set of first order hyperbolic *partial differential equations* (PDEs), coupled through boundary conditions. Despite the inherent questions regarding the continuous approximation of aircraft density, which we addressed, we validated this model and showed that it predicts accurate aircraft counts.

In this chapter, we first show how to apply standard numerical analysis tools to perform accurate numerical simulations of this system of PDEs, when we do not have explicit analytical solutions available to us (which is the case in general). The major difficulty which we will show is that the solutions we construct are by essence discontinuous and have kinks, a very undesirable property for numerical solutions of PDEs. We also show that the use of linear numerical schemes to approximate the solution of the PDE perform very poorly, which unfortunately precludes the use of standard linear optimization programs to control the system.

We therefore have to rely on flow control techniques [87, 88, 89, 92, 99, 35, 52, 36], which are directly applicable to PDE-driven systems. We use an adjoint-based method, which enables us to compute the gradient of a cost function algebraically using the adjoint problem. We have to adapt the adjoint method to the case in which the system is described by a set of PDEs coupled through the boundary conditions, in presence of constraints. Unfortunately, this is a nonlinear control problem which does not provide proofs of convergence to a *global* optimal. However, this method, as well as other flow control approaches [11, 1], has been shown to work extremely well in practice in fluid mechanics. In addition, the dimension of our PDE is one, enabling online implementations, as solving a set of one dimensional PDEs may be done extremely quickly.

Controlling transportation networks in general is extremely challenging and numerically difficult [63, 120, 80]. In the present case, the control consists in speed assignments and routing policies (i.e. determining optimal routes for the aircraft). As shown in the previous chapter, we use an *Eulerian* framework for this problem, despite the known difficulties inherent to PDE control [87, 88, 89, 92, 99, 35, 11, 1]. However, there are a few benefits of the above outlined approach over *Lagrangian* methods, which incorporate all trajectories of all aircraft:

- Most of the Lagrangian methods will end up posing a control problem as an integer optimization program. Integer optimization programs are in general intractable because they are NP-complete [34], and therefore do not provide guarantees of convergence to a global optimum. In addition, the solution provided by these methods often takes advantage of actuating single aircraft individually [33], which precludes the derivation of global policies, which we are interested in for this chapter. Finally, this framework scales very well with the number of aircraft (the higher the number of aircraft is the better the more accurate the model becomes, without further computational complexity).

- The method presented below is very general and can be very easily adapted to specific classes of controllers, which are "Air Traffic Controller friendly", i.e. it is possible to use this method to derive a control law in a required format, which is compatible with aircraft capabilities.

- This method can be applied to highway traffic with minor modifications [23] and, we believe, can be extended to other problems such as networks of irrigation channels [104].

This chapter is organized as follows. In Section 7.1, we show the relative benefits of several numerical schemes, and run tests against an analytical solution derived previously. We select the *Jameson-Schmidt-Turkel* (JST) scheme for the rest of this study. In Section 7.2, we derive the adjoint system to our problem, and show how to use it to determine the required velocity profiles along the links as well as the routing policy to apply in order to control the system. Finally, in Section 7.3, we show how to apply this method to a busy portion of airspace: the area enclosed by Chicago, New-York, Boston and the east coast of Canada.

7.1 Numerical solutions

The Eulerian PDE model which we presented before (equation (6.7)) is summarized as follows.

$$\begin{cases} \mathcal{N}_i(\rho_i) := \frac{\partial \rho_i(x_i,t)}{\partial t} + \frac{\partial}{\partial x_i}\left(\rho_i(x_i,t)v_i(x_i,t)\right) = 0 & \forall i \in \{1,\cdots,N\} \\ \rho_i(x,0) = \rho_i^\circ(x) & \forall i \in \{1,\cdots,N\} \\ \rho_i(0,t)v_i(0,t) = \sum\limits_{j\in\mathcal{U}(i)} \rho_j(L_j,t)v_j(L_j,t) & \forall i \in \mathcal{M} \\ \begin{cases} \rho_{i_l}(0,t)v_{i_l}(0,t) = \beta_i\,\rho_i(L_i,t)v_i(L_i,t) \\ \rho_{i_r}(0,t)v_{i_r}(0,t) = (1-\beta_i)\rho_i(L_i,t)v_i(L_i,t) \end{cases} & \forall i \in \mathcal{D} \\ \rho_i(0,t)v_i(0,t) = q_i^{\text{in}}(t) & \forall i \in \mathcal{S} \end{cases} \tag{7.1}$$

where notations are summarized in Table 6.1. Please refer to the previous chapter for more precision about the different variables used. In (6.7), $\mathcal{N}_i(\cdot)$ represents the LWR operator.

7.1.1 Numerical schemes

Even for a single link i, it is in general not possible to solve the system (6.7) analytically (by hand). In the previous chapter, we show an analytical solution based on the method of characteristics presented in [26] and Chapter 8, which works in the case in which $v_i(x_i,t) = v_i(x_i)$, i.e. the nominal velocity does not depend on time. When the velocity depends on time, numerical integration is needed. We can benefit from numerous numerical schemes to do that, which we now summarize.

The solutions of the LWR PDE in the system (6.7) have very undesirable properties for numerical integrations: they are by construction discontinuous (see the construction of ρ_i in Chapter 6); they can develop kinks if the velocity profiles are discontinuous. Ad hoc numerical schemes of the original LWR PDE have been the focus of recent research [64] in order to address similar difficulties encountered in the original LWR PDE; they have proved extremely efficient in the case of highway traffic. We have chosen to use three different schemes to compare their respective benefits.

- The well known *Lax-Friedrichs scheme* [82], for which the update is simply given at every time step by

$$\rho_k^{n+1} = \frac{1}{2}[\rho_{k+1}^n + \rho_{k-1}^n] - \frac{1}{2\Delta x}\Delta T[v(x_{k+1})\rho_{k+1}^n - v(x_{k-1})\rho_{k-1}^n].$$

  This scheme is linear; we chose it motivated by the recent work of Menon et al. [110, 111], which makes explicit use of the linearity of their discretization scheme in the control synthesis for their problem.

- A *left-centered scheme*, inspired by the Daganzo scheme in light traffic: the time update at every time step is given by [64]:

$$\rho_k^{n+1} = \left(\rho_k^n(1 - \frac{v(x_k)\Delta T}{\Delta x}) + \rho_{k-1}^n(\frac{v(x_k)\Delta T}{\Delta x})\right)\left(1 - \frac{v(x_k)\Delta T}{\Delta x} + \frac{v(x_{k-1})\Delta T}{\Delta x}\right).$$

  Note that this scheme only provides a first order approximation of the PDE, in the case in which v is continuous. However, it has been shown to work well in practice.

- The *Jameson-Schmidt-Turkel* (JST) *scheme*. This scheme is nonlinear, and has very desirable properties for this work: it captures shocks (which are present in the solutions we compute, as will be seen), and when the PDE has an entropy solution, which is the case for highway traffic in the original LWR setting, it converges to the entropy solution of the problem. It combines second order accuracy with non-oscillatory properties, adds an anti-diffusive term to the first-order approximation of the flux. This makes the scheme accurate especially in discontinuous regions where traditional schemes diffuse. Moreover, the coefficients of this additional term are variable and chosen in order to ensure that the maxima cannot increase and the minima cannot decrease. This condition has been proved to prevent oscillations. The JST scheme finally produces a low level of diffusion in regions were the solution is smooth but prevents oscillations near discontinuities. Details of this scheme are available in [90, 91].

7.1.2 Numerical validation

Even if a numerical scheme is theoretically proved to converge to the analytical solution of a PDE, one usually does not know a priori the required gridsize to guarantee that the numerical solution is close to the analytical solution. Even if this type of validation is standard in numerical analysis [82], it seems to be absent from literature using these schemes for highway or air traffic problems [110, 111, 64]. We use the method developed in the previous chapter to compute the analytical solution of the following three problems, which we use as benchmark problems to assess the accuracy of our numerical method. We start with one link and present three validation scenarios.

- **Case 1:** *Continuous initial conditions and inflow, continuous velocity profile*

  Velocity profile:
  $$v(x) = \begin{cases} 2 & \text{if } x \in [0,1] \\ 3-x & \text{if } x \in [1,2] \end{cases} \tag{7.2}$$
  Initial conditions:
  $$\rho(x,0) = \begin{cases} \sin(2\pi x) & \text{for } x \in [0,\frac{1}{2}] \\ 0 & \text{for } x \in [\frac{1}{2},2] \end{cases} \tag{7.3}$$
  Inflow conditions:
  $$q(t) = \begin{cases} 0 & \text{for } t \leq \frac{1}{4} \\ \sin(2\pi(1-2t)) & \text{for } t \in [\frac{1}{4},\frac{1}{2}] \\ 0 & \text{for } t \geq \frac{1}{2} \end{cases} \tag{7.4}$$

- **Case 2:** *Discontinuous initial conditions and inflow, continuous velocity profile*

  Velocity profile:
  $$v(x) = \begin{cases} 2 & \text{if } x \in [0,1] \\ 3-x & \text{if } x \in [1,2] \end{cases} \tag{7.5}$$
  Initial conditions:
  $$\rho(x,0) = \begin{cases} 1 & \text{for } x \in [0,\frac{1}{2}] \\ 0 & \text{for } x \in [\frac{1}{2},2] \end{cases} \tag{7.6}$$
  Inflow conditions:
  $$q(t) = \begin{cases} 0 & \text{for } t \leq \frac{1}{4} \\ 1 & \text{for } t \in [\frac{1}{4},\frac{1}{2}] \\ 0 & \text{for } t \geq \frac{1}{2} \end{cases} \tag{7.7}$$

- <u>**Case 3:**</u> *Continuous initial conditions and inflow, discontinuous velocity profile*
  Initial conditions and inflow conditions: as in Case 1. Velocity profile:

$$v(x) = \begin{cases} 2 & \text{if } x \in [0,1] \\ 1 & \text{if } x \in [1,2] \end{cases} \tag{7.8}$$

Note that even if these scenarios seem academic, they are relevant for the real scenarios which will be simulated in the next section. In particular, in case 2, we have discontinuous densities: this can happen when aircraft are isolated in the network (locally, the density is a “box” centered around the aircraft). We need to capture this type of solution accurately, particularly the width and height of the boxes (which encode positions and number of aircraft), which the JST scheme does, because of its anti-diffusive term. In case 3, the velocity profile is discontinuous: this can model an area in which the aircraft are required to vector (see definition in Chapter 3): when this is the case, the projected velocity (along the direction to the destination waypoint) is discontinuous at the start of the vector.

The results of the computations are shown in Figure 7.1 for Case 1, Figure 7.2 for Case 2, Figure 7.3 for Case 3. As can be seen for the three cases, the Lax-Friedrichs scheme is very diffusive. Note that the behavior of the Lax-Friedrichs scheme is very representative of linear schemes to approximate a hyperbolic PDE. Consequently, we do not think that it is a good idea to use linear numerical schemes to approximate the solution of the PDE, even if it would have the advantage of making the constraints linear in the resulting optimization program, which would be used for control. The left centered scheme is less diffusive, but fails to capture the kinks of the solution. We also show a plot of the L_2 norm of the error as a function of the number of grid points. The result is shown in Figure 7.4. As can be seen, The Jameson-Schmidt-Turkel scheme gives the best results overall and will be used for the rest of this study.

This validation is to the best of our knowledge the first which was actually implemented to assess the accuracy of the approximation as a function of the discretization size. Note also that confusion often arises between scale cell size (defined previously by Daganzo [63], i.e. line element) and discretization size (whose mathematical definition is available in [82]). These quantities are completely unrelated, as cell size is a physical length which pertains to the application (for example a portion of highway or jetway), whereas the discretization length is an arbitrary small length chosen between gridpoints such that the discretization will approximate the continuous problem accurately. Typically, a cell should contain at least a dozen grid points (in fact at least a hundred in the present simulations).

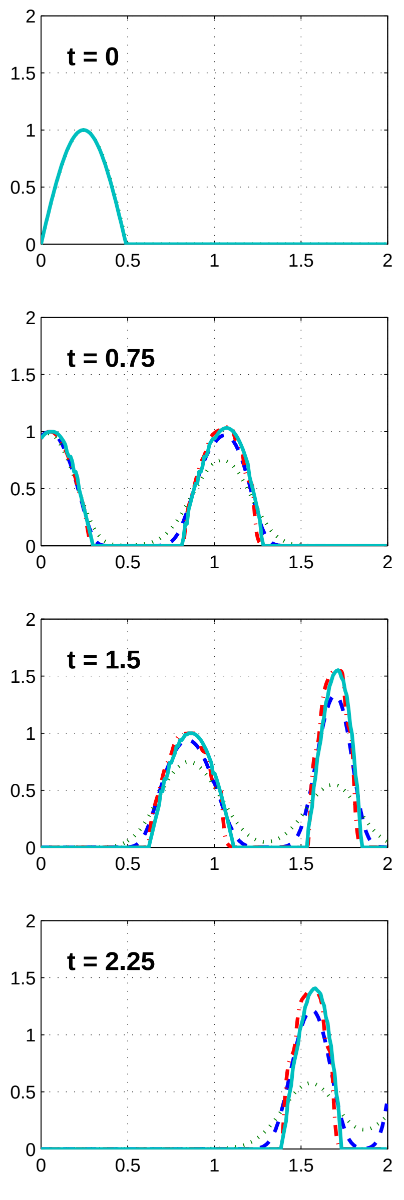


Figure 7.1: Numerical experiment for Case 1. Result of the LxF (·), Left Centered (- -), JST (—·) schemes and the exact solution (solid) for 200 grid-points. Horizontal axis: x; vertical axis: ρ.

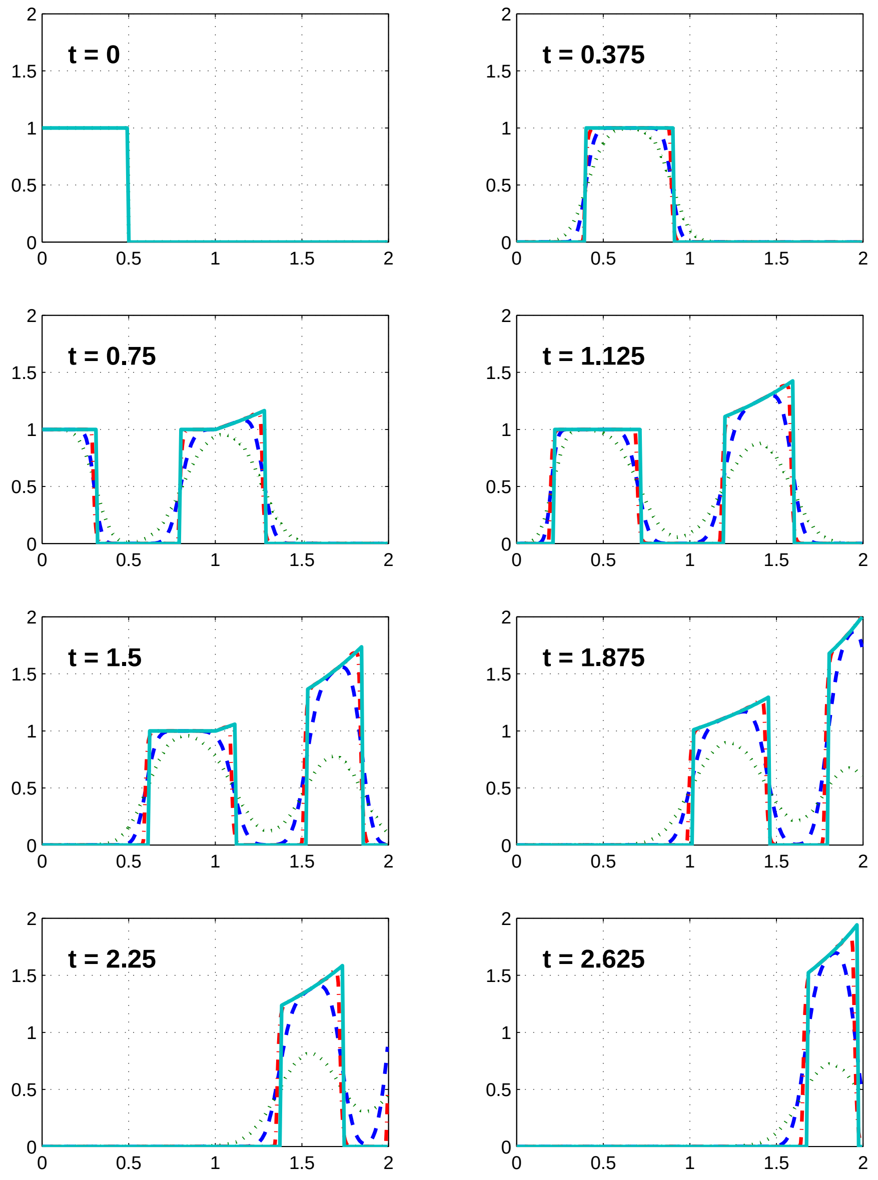


Figure 7.2: Numerical experiment for Case 2. Result of the LxF (·), Left Centered (- -), JST (—·) schemes and the exact solution (solid) for 200 grid-points. Horizontal axis: x; vertical axis: ρ.

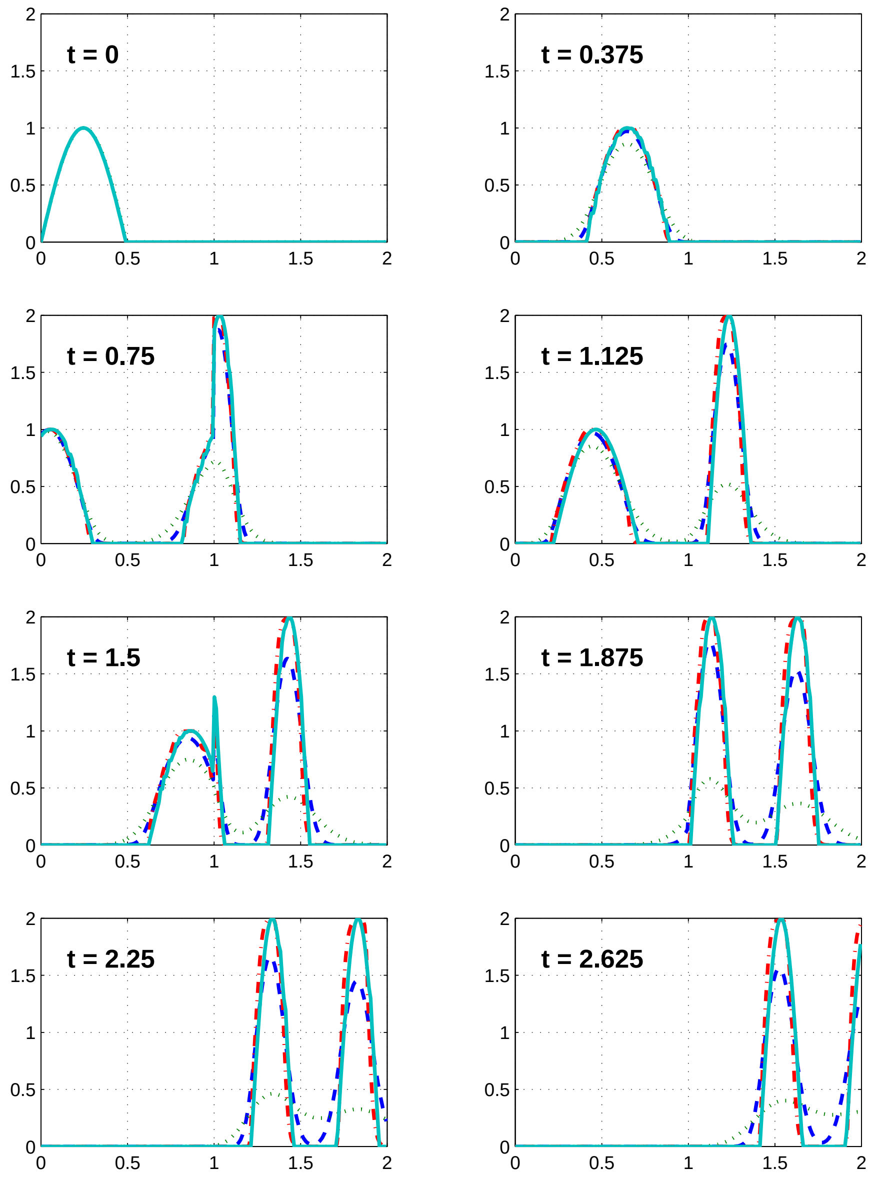


Figure 7.3: Numerical experiment for Case 3. Result of the LxF (·), Left Centered (- -), JST (—·) schemes and the exact solution (solid) for 200 grid-points. Horizontal axis: x; vertical axis: ρ.

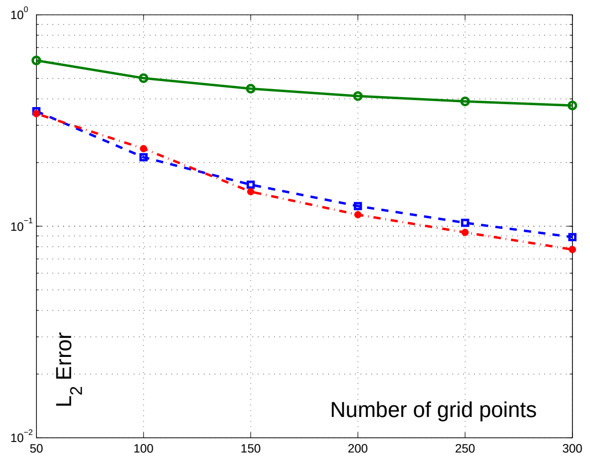


Figure 7.4: L_2 error due to the discretization method, as a function of the number of grid points for both schemes. The analytical solution presented in the previous chapter is used for this comparison. Lax-Friedrichs scheme (solid), Jameson-Schmidt-Turkel scheme (—·), left-centered scheme (- -).

7.2 Network control via adjoint methods

7.2.1 Control problem, derivation of the adjoint

Consider solving the following problem: maximize the throughput (i.e. flux of landing aircraft) at a destination airport, while maintaining the density of aircraft everywhere lower than a given threshold. Let us call $\rho_{\max,i}$ the maximal allowed density on link i, $v_{\max,i}(\cdot)$ and $v_{\min,i}(\cdot)$ the maximal and minimal achievable speeds on link i (which can depend on location). Using the notations of Section 7.1, the optimization problem thus reads:

$$\begin{array}{lll} \textbf{min:} & -\sum_{i:i\in\mathcal{F}}\int_0^T \rho_i(L_i,t)v_i(L_i,t)\,dt & \\ \textbf{s.t.:} & (6.7) & \\ & \rho_i(x_i,t)\le\rho_{\max,i} & \forall i\in\{1,\cdots,N\},\forall x_i\in[0,L_i],\forall t\in[0,T] \\ & v_{\min,i}(x_i)\le v_i(x_i)\le v_{\max,i}(x_i) & \forall i\in\{1,\cdots,N\},\forall x_i\in[0,L_i],\forall t\in[0,T] \\ & 0\le\beta_i(t)\le 1 & \forall i\in\mathcal{D},\forall t\in[0,T] \end{array} \tag{7.9}$$

where $\mathcal{F}$ is the set of links merging into the sink or airport (see Table 6.1 for notation). In the optimization program above, the control variables are β_i and v_i; ρ_i is the state. The difficulty posed by the constraints can be avoided in practice by using a classical optimization technique called *barrier* (see for example Boyd et al. [41]), in which the cost is augmented by a logarithmic term, which prohibits violation of the constraints.

$$\begin{array}{ll}\textbf{min:} & -\sum_{i:i\in\mathcal{F}}\int_0^T \rho_i(L_i,t)v_i(L_i,t)\,dt \\ & -\frac{1}{M}\sum_{i=1}^N\int_0^T\int_0^{L_i}\log\left((\rho_{\max}-\rho_i(x_i,t))(v_{\max}-v_i(x_i,t))(v_i(x_i,t)-v_{\min,i})\right)\,dx_i\,dt \\ & -\frac{1}{M}\sum_{i:i\in\mathcal{D}}\int_0^T\log(\beta_i(t)(1-\beta_i(t)))\,dt \\ \textbf{s.t.:} & (6.7)\end{array} \tag{7.10}$$

We call $H(v,\beta,\rho)$ the augmented cost function. When ρ, β and v are used without indices, it means that they are vectors, i.e. $v=[v_1,\cdots,v_N]$. Note that the two last constraints in the optimization program (7.9) have disappeared into the cost function. This constrained optimization problem is easier to solve in practice. It is asymptotically equivalent to the problem of interest when $M\rightarrow+\infty$.

We use an adjoint method to algebraically compute the gradient of the cost function. This method was extensively used by Jameson [87, 88, 89, 92, 99] and Bewley [35, 52, 36] in flow control. We now adapt the adjoint method to the case in which we have a set of PDEs coupled through the boundary conditions, and subject to constraints. The adjoint method computes the gradient of the cost function $H(v,\beta,\rho)$ when ρ is an implicit function of v and β via the dynamics (6.7). Let us denote $\mathcal{J}$ the cost function of the two variable v and β: $\mathcal{J}:(v,\beta)\rightarrow\mathcal{J}(v,\beta)=H(v,\beta,\rho)$ where ρ is the solution of the PDE system (6.7). Following Bewley [35], we compute the linearized (6.7), which we will use to compute the gradient of the cost function in the optimization program (7.10).

We denote by ' the linearized quantities around a nominal value denoted by $^-$: $\rho_i=\bar{\rho}_i+\rho'_i$. We call $\mathcal{N}'_i(\cdot)$ the linearized LWR operator, and $q_i=\rho_i v_i$. In order to abbreviate the notation, we will write $q'_i=\rho'_i\bar{v}_i+\bar{\rho}_i v'_i$ and $\bar{q}_i=\bar{\rho}_i\bar{v}_i$. We omit the time and space dependence when they are obvious. We are NOT using Einstein's notations. The linearized (6.7) reads:

$$\begin{cases}\mathcal{N}'_i(\rho'_i):=\mathcal{N}'_i\rho'_i=\frac{\partial\rho'_i}{\partial t}+\frac{\partial\rho'_i\bar{v}_i}{\partial x}+\frac{\partial\bar{\rho}_i v'_i}{\partial x}=0 & \forall i\in\{1,\cdots,N\} \\ \rho'_i(x,0)=0 & \forall i\in\{1,\cdots,N\} \\ q'_i(0,t)=\sum_{j\in\mathcal{U}(i)}q'_j(L_j,t) & \forall i\in\mathcal{M} \\ q'_{i_l}(0,t)=\beta'_i(t)\,\bar{q}_i(L_i,t)+\bar{\beta}_i(t)\,q'_i(L_i,t) & \forall i\in\mathcal{D} \\ q'_{i_r}(0,t)=-\beta'_i(t)\,\bar{q}_i(L_i,t)+(1-\bar{\beta}_i(t))q'_i(L_i,t) & \forall i\in\mathcal{D} \\ \rho'_i(0,t)\bar{v}_i(0)+\bar{\rho}_i(0,t)v'_i(0,t)=0 & \forall i\in\mathcal{S}\end{cases} \tag{7.11}$$

The cost function perturbation can be obtained by linearization of the objective function in (7.10):

$$\begin{array}{rcl} \mathcal{J}' & = & -\sum\limits_{i:i\in\mathcal{F}} \int_0^T \rho_i'(L_i,t)\bar{v}_i(L_i,t) + \bar{\rho}_i(L_i,t)v_i'(L_i,t) \\ & & +\frac{1}{M}\sum_{i=1}^N \int_0^T \int_0^{L_i} \left(\frac{\rho_i'(x_i,t)}{\rho_{\max,i}-\bar{\rho}_i(x_i,t)} + \frac{v_i'(x_i,t)}{v_{\max,i}-\bar{v}_i(x_i,t)} - \frac{v_i'(x_i,t)}{\bar{v}_i(x_i,t)-v_{\min,i}}\right) dx_i\,dt \\ & & +\frac{1}{M}\sum_{i:i\in\mathcal{D}} \int_0^T \frac{\beta_i'(t)}{1-\beta_i(t)} - \frac{\beta_i'(t)}{\beta_i(t)}\,dt \end{array} \tag{7.12}$$

An integration by parts leads to the following identity for any two functions ρ_i' and ρ_i^*.

$$\begin{array}{rcl} \int_0^T \int_0^{L_i} \rho_i^* \mathcal{N}_i' \rho_i'\,dx_i\,dt & = & \int_0^{L_i} [\rho_i^*\rho_i']_0^T\,dx_i - \int_0^T \int_0^{L_i} \frac{\partial \rho_i'}{\partial t}\rho_i^*\,dt\,dx_i + \\ & & \int_0^T [\rho_i^*\rho_i'\bar{v}_i]_0^{L_i}\,dt - \int_0^T \int_0^{L_i} \rho_i'\bar{v}_i \frac{\partial \rho_i^*}{\partial x_i}\,dt\,dx_i \end{array} \tag{7.13}$$

which can be rewritten using the standard inner product denoted $\langle\cdot|\cdot\rangle_i$ for the domain $[0,L_i]\times[0,T]$:

$$\langle \rho_i^* | \mathcal{N}_i' \rho_i' \rangle_i \;=\; \langle \mathcal{N}_i^* \rho_i^* | \rho_i' \rangle_i + b_i \tag{7.14}$$

with

$$\left\{ \begin{array}{l} \mathcal{N}_i{}^* = -\frac{\partial(.)}{\partial t} - \bar{v}_i \frac{\partial(.)}{\partial x_i} \\ b_i = \int_0^{L_i} [\rho_i^*\rho_i']_0^T\,dx_i + \int_0^T [\rho_i^*\rho_i'\bar{v}_i]_0^{L_i}\,dt \end{array} \right. \tag{7.15}$$

We will denote by $\langle\cdot,\cdot\rangle_{[0,T]}$ the standard inner product in $[0,T]$. $\mathcal{N}_i^*$ is called the adjoint operator of $\mathcal{N}_i'$. In order to express the gradient of $\mathcal{J}$ as a function of the v_i' and β_i' only, we choose an adjoint density field ρ_i^* that cancels all the terms containing ρ_i' in the cost function.

First, in order to eliminate the term $\frac{1}{M}\sum_{i=1}^N \int_0^T \int_0^{L_i} \frac{\rho_i'(x_i,t)}{\rho_{\max,i}-\bar{\rho}_i(x_i,t)} dx_i dt$, we choose ρ_i^* such that

$$\mathcal{N}_i{}^* \rho_i^* = \frac{1}{M(\rho_{\max,i} - \bar{\rho}_i)} \tag{7.16}$$

This is a first order linear hyperbolic PDE, which is well-posed if $\bar{\rho}_i$ is known and both the boundary conditions at one location and the initial conditions at one time are specified. This allows us to enforce two other conditions for ρ_i^* in order to cancel all the terms containing ρ_i'. We can choose:

$$\begin{cases} \rho_i^*(x_i,T)=0 & \forall i\in\{1,\cdots,N\}, \forall x_i\in[0,L_i] \\ \rho_i^*(L_i,t)=-1 & \forall i\in\mathcal{F},\ \forall t\in[0,T] \\ \rho_i^*(0,t)=\rho_j^*(L_j,t) & \forall i\in\mathcal{M},\ \forall j\in\mathcal{U}(i),\ \forall t\in[0,T] \\ \rho_i^*(L_i,t)=\bar{\beta}_i(t)\rho_{i_l}^*(0)+(1-\bar{\beta}_i(t))\rho_{i_r}^*(0) & \forall i\in\mathcal{D},\ \forall t\in[0,T] \end{cases} \tag{7.17}$$

where i_r and i_l have been defined in Table 6.1 in the previous chapter. These conditions have been chosen by necessity of the algebraic derivation, in order to cancel appropriate terms in the perturbation of the cost function, as will appear in the derivation below.

In addition, a physical interpretation of the boundary and terminal conditions of the adjoint can be given. The first condition is terminal and stipulates that the sensitivity of the solution to perturbations of the system at $t=T$ is zero. The second condition accounts for the sensitivity of the solution at the sink: by actuating directly there, one decreases the objective function. The third condition at every merging node says that the sensitivity is the same for all branches connected to the node, at that point. The last condition says the same at the diverging nodes, weighted by the mean $\bar{\beta}_i$ which represents the portion of flow choosing the respective links outgoing from the node.

We now show the algebra which led to the "good choice" (7.17).

$$\begin{aligned} \mathcal{J}' \ = \ & -\sum_{i:i\in F}\int_0^T q_i'(L_i,t)\,dt \\ & -\textstyle\sum_{i=1}^N\left(\int_0^T\int_0^{L_i}\rho_i^*\frac{\partial v_i'\bar{\rho}_i}{\partial x_i}\,dx_i\,dt+\int_0^L\underbrace{[\rho_i^*\rho_i']_0^T}_{0}\,dx_i+\int_0^T[\rho_i^*\rho_i'\bar{v}_i]_0^{L_i}\,dt\right) \\ & +\frac{1}{M}\sum_{i=1}^N\int_0^T\int_0^{L_i}\left(\frac{v_i'(x_i,t)}{v_{\max,i}-\bar{v}_i(x_i,t)}-\frac{v_i'(x_i,t)}{\bar{v}_i(x_i,t)-v_{\min,i}}\right)\,dx_i\,dt \\ & +\frac{1}{M}\sum_{i:i\in\mathcal{D}}\int_0^T\left(\frac{\beta_i'(t)(t)}{1-\bar{\beta}_i(t)}-\frac{\beta_i'(t)}{\bar{\beta}_i(t)}\right)dt \end{aligned}$$

The cancellation of the terms shown here is due to the initial condition on ρ_i' and the terminal conditions on ρ_i^*. Thus, an integration by parts leads to:

$$\begin{aligned}\mathcal{J}' = & -\sum_{i:i\in\mathcal{F}}\int_0^T q_i'(L_i,t)\\ &+\sum_{i=1}^{N}\Big(\int_0^T\int_0^{L_i}\tfrac{\partial\rho_i^*}{\partial x_i}v_i'\bar{\rho}_i\,dx_i\,dt-\int_0^T\left(\rho_i^*(L_i,t)q_i'(L_i,t)-\rho_i^*(0,t)q_i'(0,t)\right)\,dt\Big)\\ &+\tfrac{1}{M}\sum_{i=1}^{N}\int_0^T\int_0^{L_i}\left(\tfrac{v_i'(x_i,t)}{v_{\max,i}-\bar{v}_i(x_i,t)}-\tfrac{v_i'(x_i,t)}{\bar{v}_i(x_i,t)-v_{\min,i}}\right)dx_i\,dt\\ &+\tfrac{1}{M}\sum_{i:i\in\mathcal{D}}\int_0^T\left(\tfrac{\beta_i'(t)}{1-\beta_i(t)}-\tfrac{\beta_i'(t)}{\beta_i(t)}\right)dt\end{aligned}$$

Partitioning the terms in two, we have

$$\begin{aligned}\mathcal{J}' \;=\; & \left.\begin{aligned}&+\sum_{i=1}^{N}\int_0^T\int_0^{L_i}\tfrac{\partial\rho_i^*}{\partial x_i}\bar{\rho}_iv_i'\,dx_i\,dt\\ &+\tfrac{1}{M}\sum_{i=1}^{N}\int_0^T\int_0^{L_i}\left(\tfrac{v_i'(x_i,t)}{v_{\max,i}-\bar{v}_i(x_i,t)}-\tfrac{v_i'(x_i,t)}{\bar{v}_i(x_i,t)-v_{\min,i}}\right)dx_i\,dt\\ &+\tfrac{1}{M}\sum_{i:i\in\mathcal{D}}\int_0^T\left(\tfrac{\beta_i'(t)}{1-\bar{\beta}_i(t)}-\tfrac{\beta_i'(t)}{\bar{\beta}_i(t)}\right)dt\end{aligned}\right\}A\\ & \left.-\int_0^T\Big(\sum_{i:i\in\mathcal{F}}q_i'(L_i,t)+\sum_{i=1}^{N}\rho_i^*(L_i,t)q_i'(L_i,t)-\rho_i^*(0,t)q_i'(0,t)\Big)dt\quad\right\}B\end{aligned}$$

The term B can be written as follows:

$$\begin{aligned}B \;=\; & -\int_0^T\Bigg(\sum_{i:i\in\mathcal{F}}\underbrace{q_i'(L_i,t)+\rho_i^*(L_i,t)q_i'(L_i,t)}_{0}\\ &+\sum_{i:i\in\mathcal{D}}\rho_i^*(L_i,t)q_i'(L_i,t)-\rho_{i_l}^*(0,t)q_{i_l}'(0,t)-\rho_{i_r}^*(0,t)q_{i_r}'(0,t)\\ &-\sum_{i:i\in\mathcal{M}}\underbrace{\Big(\rho_i^*(0,t)q_i'(0,t)-\sum_{j:j\in\mathcal{U}(i)}\rho_j^*(L_j,t)q_j'(L_j,t)\Big)}_{0}-\sum_{i:i\in\mathcal{S}}\rho_i^*(0,t)\underbrace{q_i'(0,t)}_{0}\Bigg)dt\end{aligned}$$

The first cancellation in the term above is due to the choice $\rho_i^*(L_i,t)$ for the links merging into the sink. The second cancellation is due to the choice of ρ_i^* at the merging links and the conservation of mass of the perturbed flow at the merging points. Exploiting the flow boundary conditions at the diverging links, similar cancellations are obtained:

$$B=-\int_0^T\Big(\sum_{i:i\in\mathcal{D}}q_i'(L_i)\underbrace{(\rho^*(L_i,t)-\beta_i(t)\rho_{i_l}^*(0,t)-(1-\beta_i(t))\rho_{i_r}^*(0,t))}_{0}-\beta_i'\bar{q}_i(L_i,t)(\rho_{i_l}^*(0,t)-\rho_{i_r}^*(0,t))\Big)\,dt$$

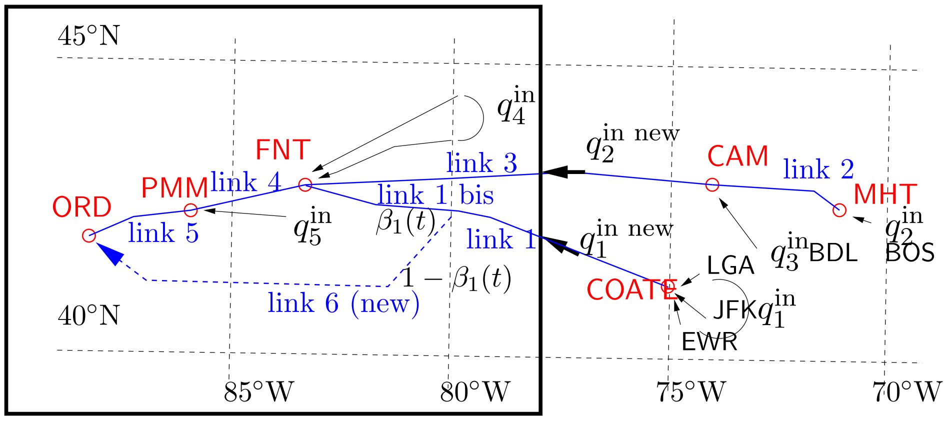


Figure 7.5: Network model shown in Chapter 6. We now add a divergence link in order to be able to show that we are able to control the density of aircraft by splitting the flow. For this simulation, we restrict ourselves to the box including ORD. The new additional link is shown with a dashed line. We call β_1 the portion of flow which stays on link 1 (called 1 bis) and $1-\beta_1$ the portion which goes into the new link (link 6).

The remaining terms can be rewritten in scalar product form:

$$\begin{aligned}\mathcal{J}' &= \sum_{i=1}^{N}\left\langle \bar{\rho}_i\frac{\partial\rho_i^*}{\partial x_i}+\frac{1}{M}\left(\frac{1}{v_{\max,i}-\bar{v}_i}-\frac{1}{\bar{v}_i-v_{\min,i}}\right)\middle|\, v_i'\right\rangle_i \\ &+ \sum_{i\in\mathcal{D}}\left\langle \bar{\rho}_i(L_i)\bar{v}_i(L_i)(\rho_{i_l}^*(0)-\rho_{i_r}^*(0))+\frac{1}{M}\left(\frac{1}{1-\bar{\beta}_i}-\frac{1}{\bar{\beta}_i}\right)\middle|\,\beta_i'\right\rangle_{[0,T]}\end{aligned} \tag{7.18}$$

where again $\langle\cdot,\cdot\rangle_i$ denotes the inner product for the domain $[0,L_i]\times[0,T]$ and $\langle\cdot,\cdot\rangle_{[0,T]}$ for the domain $[0,T]$.

7.2.2 Restriction to specific classes of control

The functions $v_i(\cdot,\cdot)$ and $\beta_i(\cdot)$ generated by this method might be ill-behaved and thus be inappropriate for practical Air Traffic Control applications. We can alleviate this difficulty by projecting the descent direction $\bar{\rho}_i\frac{\partial\rho_i^*}{\partial x_i}+\frac{1}{M}\left(\frac{1}{v_{\max,i}-\bar{v}_i}-\frac{1}{\bar{v}_i-v_{\min,i}}\right)$ into a vector space $\mathcal{E}$ of appropriate functions, for example the set of continuous functions with bounded derivative, or the set of continuous piecewise affine functions.

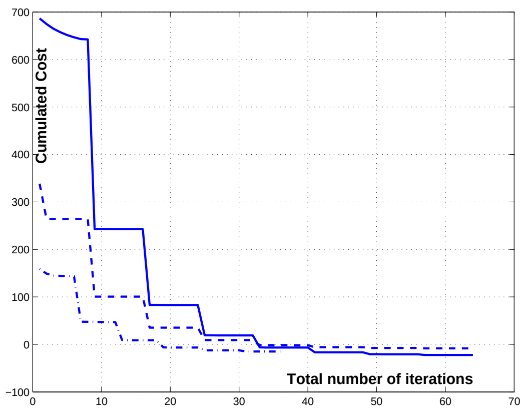


Figure 7.6: Decrease of the cost as a function of the iterations for the three scenarios. The increases in M are clearly visible (steps), while the gradient descent provided by (7.18) is more subtle. Congested traffic (solid); heavy traffic (- -); normal traffic (—·).

7.3 Application to controller design for the airspace

In this section, we demonstrate the effectiveness of the method by applying it to the air traffic model built in the previous chapter. Please refer to it for a description of airspace. Figure 7.5 shows the area which we will control (enclosed by a box). The inflows into the box are thus now $q_1^{\text{in new}}$ and $q_2^{\text{in new}}$ as shown in Figure 7.5. We want to impose the following constraint: for all links, the density should be below a threshold $\rho_{\max}$ which we impose. We allow the flow to be split into a new link (link 6), in order to aid satisfaction of the maximal density constraints. We call β_1 the corresponding split factor: β_1 is the fraction of the flow which stays on link 1 (called 1 bis); $1 - \beta_1$ is the fraction which is routed through link 6. This new link might use another arrival into the airport (it enters the arrival airspace from another direction).

We simulate the following three scenarios:

- **Scenario 1: normal traffic.** (Real data) We take ETMS data, from which we extract initial conditions and inflows, as explained in Chapter 6. We impose a restriction on the density and control the flow.
- **Scenario 2: heavy traffic.** (Modified real data) We take the same data as for the previous case, and add additional aircraft in order to overload even more the network.

- **Scenario 3: congested network.** We generate data with very high densities of aircraft. This situation does not use ETMS data; it is generated randomly.

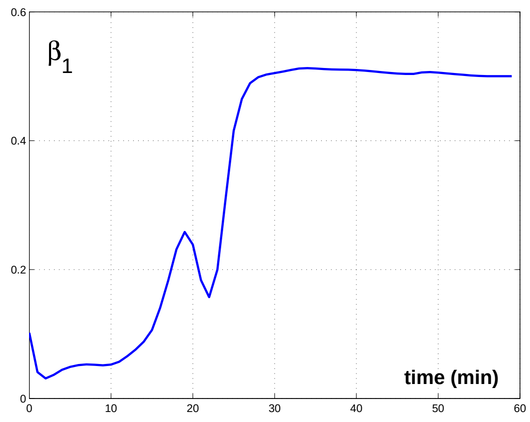


Figure 7.7: Evolution of the β_1 parameter as a function of time.

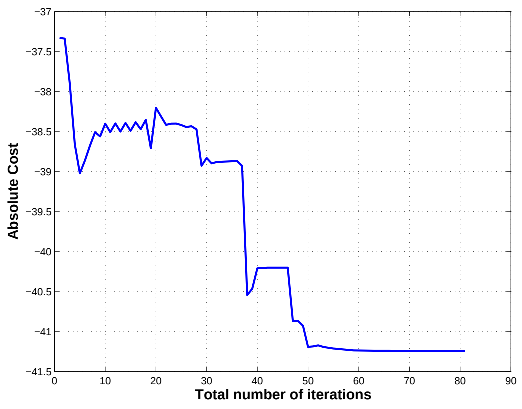


Figure 7.8: Decrease of the true cost as a function of the iterations for the three scenarios. The true cost is the cost $\mathcal{J}$ without the barrier terms. The method does not guarantee the monotonicity of the decrease but only the convergence.

Figure 7.6 shows the decrease in cost for the three scenarios as a function of the total number of iterations (i.e. iterations on M and gradient advances). As can be seen in this Figure, the more congested the situation is, the higher the cost is. The evolution of the cost with iterations exhibits two distinct behaviors, as often with barrier methods [41]: large jumps corresponding to the increases in M, and shallower decreases corresponding to the gradient advances. Convergence is clearly observed for the three scenarios. Figure 7.8 shows the true cost for the congested network experiment (true cost means $\mathcal{J}$ minus the barrier terms). As can be seen, this cost effectively decreases through the experiment, and eventually converges to a minimum.

We display some of the results for the third case. An animation (in form of an .avi movie file) corresponding to each of the three scenarios is available at [157]. An animation showing the evolution of density $\rho_i(\cdot,\cdot)$ and $v_i(\cdot,\cdot)$ as a function of time an optimization iterations is also available: it shows how the cost is decreased and the velocity functions modified with the increase of M. We now describe in detail the scenario corresponding to Case 3.

We run a one hour simulation. Figure 7.11 shows the aircraft density on all links at various instants, in the absence of control: the velocity is the mean velocity profile determined for each link in the previous chapter, and no aircraft is allowed into link 6 (i.e. $\beta_1 = 1$). The initial density is shown in the top left corner. The inflow into links 1 and 3 is such that at time $t = 27$, the density threshold (represented by the horizontal line on each subplot) is violated until time $t = 45$. At time $t = 55$, it is violated again, until the end of the experiments. Figure 7.10 shows the same experiment when link 6 is now opened to traffic, and velocity is enabled. As can be seen, about half of the traffic incoming into link 1 is rerouted into link 6, and the other half into link 1 bis. Figure 7.7 shows the variation of β_1 with time. As can be seen, around $t = 20$ min., there is a peak of about 25% of aircraft routed into link 6, which settles to 50% at $t = 30$. The routing control enables avoidance of violation of maximal density shown in subfigures 11 and 12 of Figure 7.11. The first violation (shown in subfigures 5–8 of Figure 7.11) is avoided by velocity changes.

The velocity profiles $v_i(x_i, t)$ are shown in Figure 7.9. Each of the subplots corresponds to one of the links. For links 5 and 6, one can clearly see the descent velocity profiles. Also, for link 6 (subfigure below), one can see a ridge. It corresponds to a set of aircraft which have to fly at high speed into the airport. One can also see similar ridges on the other subplots, which have the same interpretation. For any ridge, the Air Traffic Controller command could be to the corresponding set of aircraft: "fly direct at 420 kts direct into [the next waypoint]".

Note that in the absence of control, the first violation of the aircraft density threshold occurs 33 minutes after the beginning of the experiment, almost at the end of the network, which

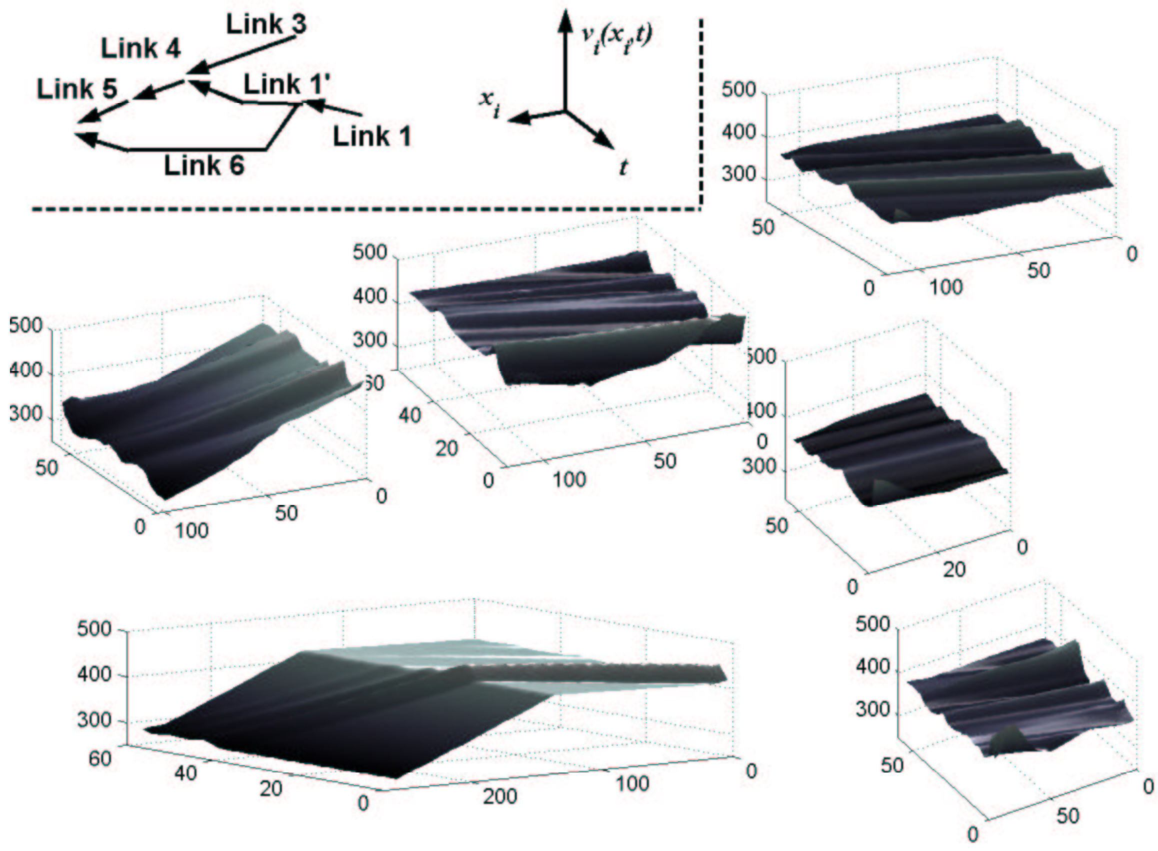


Figure 7.9: Evolution of the velocity fields as a function of time for the different links. Each of the plots corresponds to a link, (see top left corner). The axis of each subplot are: x_i (arclength along the link), t (time) and $v_i(x_i, t)$, the velocity distribution.

is not intuitive. This shows the efficiency of the method, which is capable of generating the right routing and speed assignments to prevent undesirable events from happening much later.

Finally, the simulations shown in Figure 7.7, 7.9, 7.10, 7.11 are also depicted on a US map in Figure 7.12 using the same density encoding as in Figure 7.12. One can see that before $t = 27$, all aircraft choose the direct route through link 5 to Chicago (it is shorter). After $t = 27$, the excessive amount of flow incoming into links 1 and 3 forces the flow to be split through links 1 bis and 6.

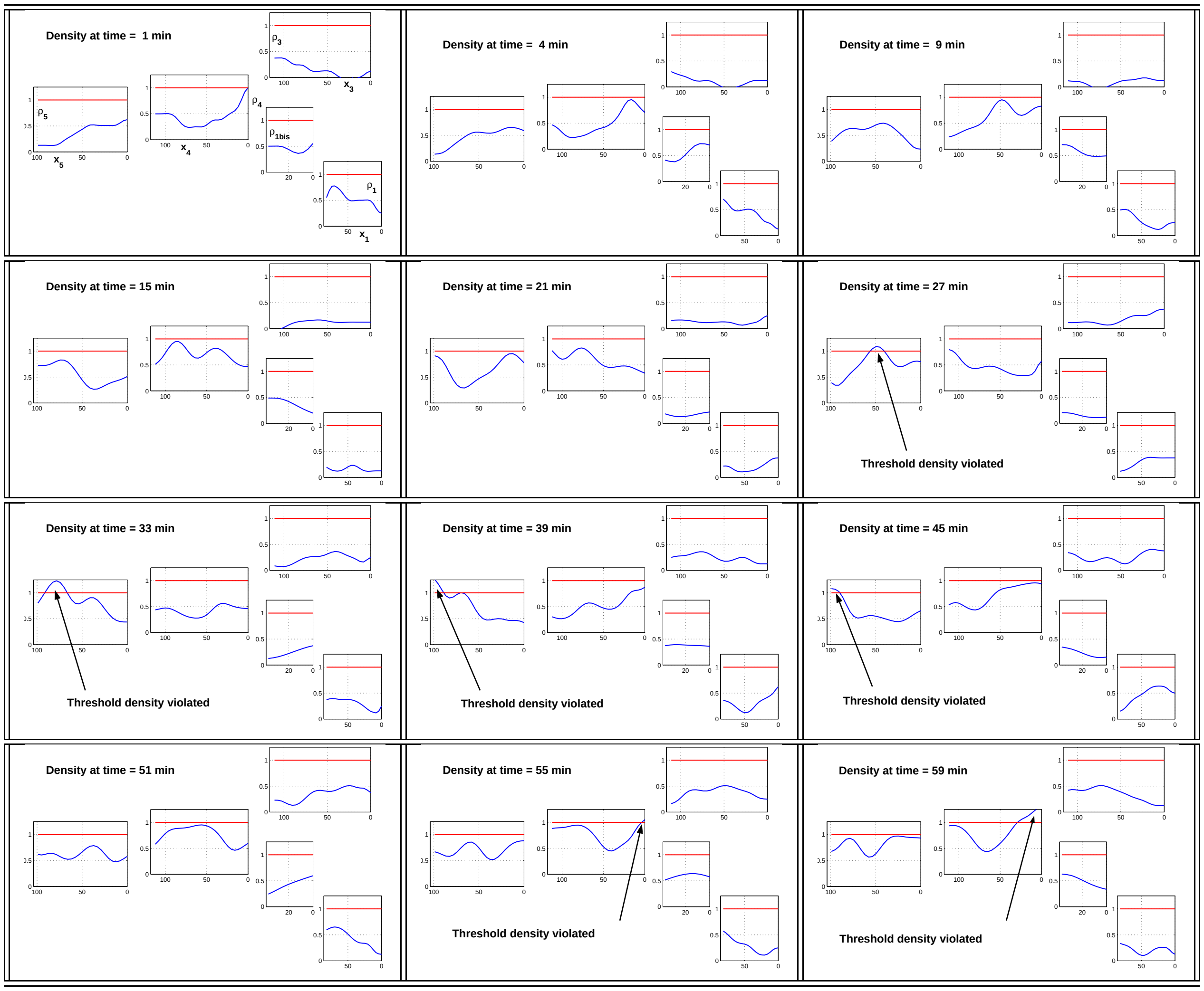


Figure 7.10: Evolution of the aircraft density on the different links in the absence of control. Each of the subplot shows the density distribution at a given time on the corresponding link as in Figure 7.9 (see legends on the upper left plot). The horizontal line represents the density threshold (all quantities are nondimensionalized by $\rho_{\max}$, so that the threshold density is 1).As can be seen, the density threshold is violated in link 5 at $t = 27$ until $t = 45$, and again at $t = 55$ until $t = 59$. This figure is also available in form of a .avi file at [157].

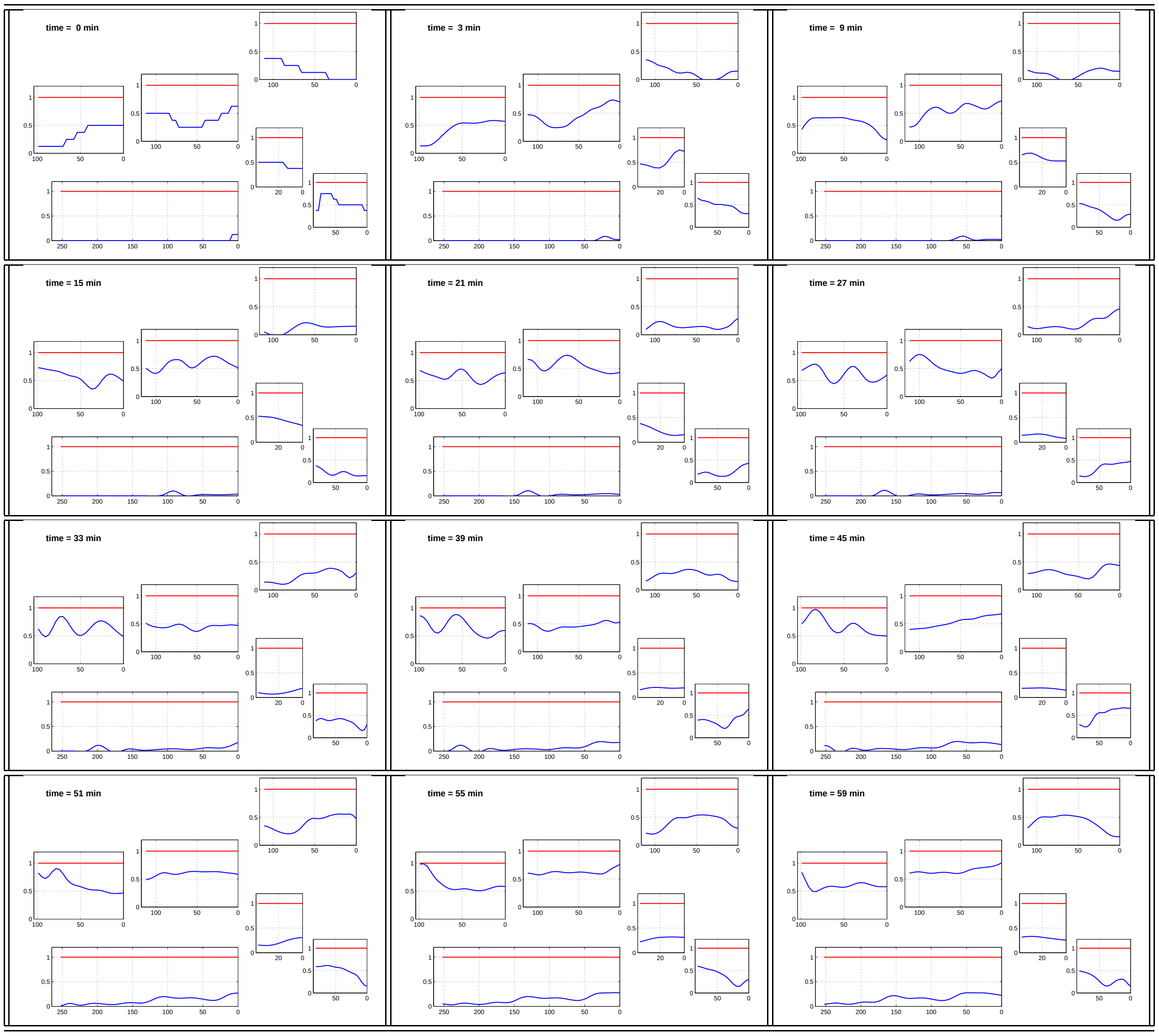


Figure 7.11: Evolution of the aircraft density with control applied. Note that link 6 is now open, and used. This prevents the second violation of density threshold observed in Figure 7.10 (from $t = 55$ to $t = 59$): some of the flow is directly routed from link 1 to link 6. The first violation seen in Figure 7.11 is avoided by speed changes. One can see that at $t = 45$ and $t = 55$, the maximal density is exactly $\rho_{\max}$. This figure is also available in form of a .avi file at [157].

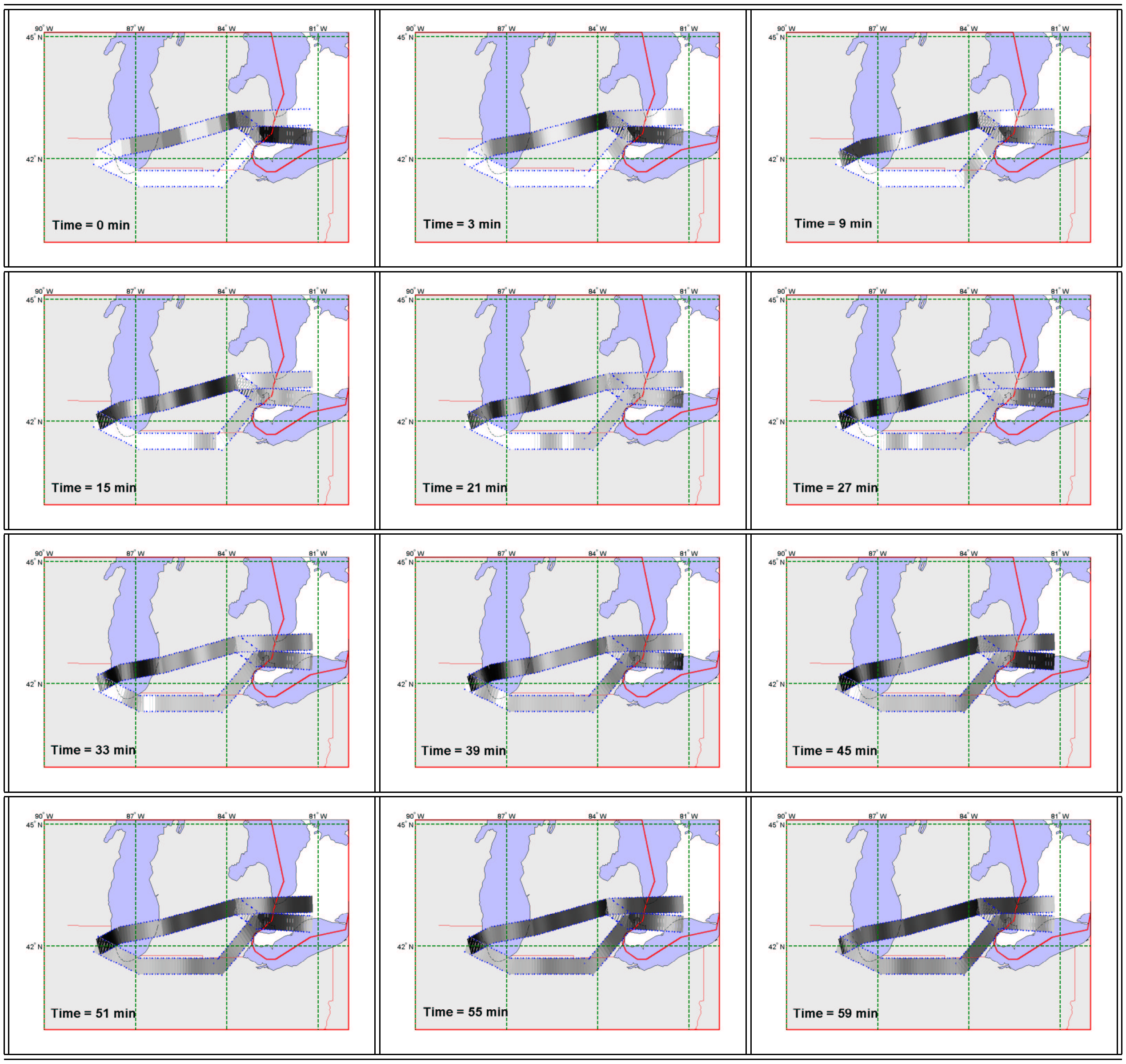


Figure 7.12: Aircraft density in the network around Chicago in presence of control and velocity assignment. The density of the links is depicted by the color. The colored rectangles shown in this plot represent the density. The colorscale is: white for zero density; black for highest density. As can be seen and was shown in Figures 7.11 and 7.10, a good portion of the flow is routed into link 6 starting at time $t = 51$.

Part II

Algorithms for vehicle-vehicle interaction

Chapter 8

Theoretical and computational frameworks for Hamilton-Jacobi formulations of reachability problems

As was mentioned throughout this dissertation, one of the most important and time consuming Air Traffic Controller tasks is to prevent *losses of separation* (LOS) between aircraft; for high altitude sectors (above 29,000 ft), this means that the Air Traffic Controller must keep each pair of aircraft in the sector separated by more than 5 nautical miles (nm) horizontally, and 1000 feet vertically. The terminology *protected zone* is used to represent the 5 nm radius, 2000 foot high cylinder around an aircraft, that another aircraft must not penetrate. For any pair of aircraft, the relative configuration or *state* (relative position and orientation) is referred to as *unsafe* if there is a *rational* process of actions which leads one aircraft to penetrate the protected zone of another.

The previous chapters dealt with models of the airspace and corresponding control policies at a high level, i.e. for a large number of aircraft. In the Lagrangian setting, the control consisted in determining an optimal maneuver assignment in order to achieve the optimum of a prescribed cost function. This assignment was posed as an optimization program, which we solved using combinatorial optimization. In the Eulerian setting, the goal was to balance aircraft density in the system, which was achieved using flow control techniques based on adjoint formulations.

Both approaches suppose that at the sector level, the Air Traffic Controller has the ability to separate the aircraft, given the constraints imposed by this "higher level control", and that the corresponding potential conflict avoidance resolution mechanisms would not affect the overall objective. Several methodologies have tried to incorporate conflict resolution into more general objectives, such as scheduling (see for example [133]). Others have tried to quantify the impact of conflict resolution on scheduling, for example [38, 119]. Solving large scale problems while guaranteeing safety (i.e. aircraft separation) is still an open problem, and it is not clear that it is actually solvable.

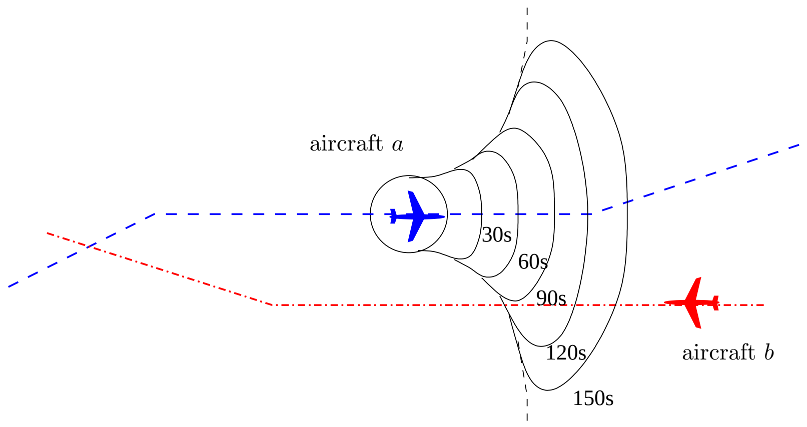


Figure 8.1: Sketch of the reachable set. Aircraft a is following the dashed flight plan; aircraft b is following the dash-dotted flight plan. The isocontours show all possible locations of aircraft a in the next 30, 60, 90, etc. seconds in relative dynamics of the two aircraft. This is called the reachable set and will be defined mathematically in this chapter.

However, in practice, aircraft have conflicting flight plans (see Figure 8.1). In the current system, the problem is solved by humans, using rules based on experience. Several methodologies have been proposed for automating this set of tasks (see [96] for an extensive survey). Given a protocol, i.e. a rule that aircraft follow when encountering a conflict, a proof of safety is a mathematical proof that no matter how both aircraft behave within a set of behaviors which we assume we can model, the separation between the two aircraft is always above the minimum required by FAA. Figure 8.1 illustrates this idea: the isolines represents all possible positions of the left aircraft in relative dynamics, for the times labeled on the isolines. These positions incorporate uncertainties (for example unexpected behavior of the right aircraft, or winds), within bounds which we know a priori. This set is called the *reachable set* and will be precisely defined in this chapter. Naively phrased, proving the safety of a conflict avoidance protocol between two aircraft thus consists in checking that the "uncertain" aircraft will not lie in the reachable set, provided proper action is taken (a more mathematical definition will be given in the next sections).

This chapter is organized as follows. Section 8.1 describes the reachability problem and defines a mathematical framework which enables the proof of safety of protocols for avoiding collisions, with use of the reachable set. Section 8.2 links this framework with other frameworks historically developed to solve the same problems (viability theory and optimal control), through an equivalence proof. This mathematical proof is crucial, since it links our formulation with known mathematical results, which enable the proof of safety. In Section 8.3, we show how to solve analytically a Mayer problem, which is the problem formulation of used in the framework of Section 8.1.

8.1 The reachability problem

8.1.1 Origins of the problem

This section investigates the definition and contraction of the *reachable set.* For a dynamical system, one can define a set of states (which is a subset of the state space) in which it is undesirable to be as a target. The reachable set can be informally defined as the set of states from which a disturbance has the possibility to drive the system into the target, no matter what action the control takes. The problem was formalized at the beginning of the 1950's and a memorandum was written by Isaacs [85], which became one of the founding stones for the field of *differential games.* In this work, numerous differential games are posed, and a solution method is proposed, which Isaacs called the *retrograde path equations method.* It is in fact an adapted version of the *method of characteristics*, which dates back to the early age of the theory of partial differential equations [70, 145]. The method of Isaacs enabled researchers to solve by hand numerous problems posed as differential games, at a time when computers were not available. Isaacs also notes the connection of the differential game problem with the static *Hamilton-Jacobi Equation* (HJE), but had "found no means as yet of capitalizing on this fact" [85]. Isaacs' formulation of the differential game problem inspired active research in the field of differential games, which lasted for decades and produced numerous analytical solutions to specific differential game problems, for example [97, 81, 98, 112, 129, 32].

Until the discovery of the viscosity solution of the HJE for control problems [59, 58] in 1983 and its generalization to differential game problems [71], followers of Isaacs had been computing analytical solutions of the Hamilton-Jacobi equations, which at best were weak solutions [70]. The concept of viscosity solution, which is a specific weak solution selected through a test functions criterion, enables the proof of existence and uniqueness of a correct solution to the differential game problem. This discovery opened the door to a new type of research, in which the emphasis was not anymore on analytically constructing weak solutions of the HJE, but viscosity solutions, and later, with the advent of fast computers, convergent approximation schemes to the viscosity solution.

In the late 1980's, research emanating from the set valued analysis [9, 10] community led to the emergence of *Viability Theory* [6], which is the first systematic attempt to classify the different problems from control theory and differential games as sets depending on information patterns and type of game or control problem. Viability theory provides numerous existence and uniqueness theorems for these problems, and characterizes conditional victory domains, guaranteed victory domains, reachable tubes, capture basins, viability kernels, etc. mathematically. The link between viability theory and viscosity solutions of the Hamilton-Jacobi equations was made several times [44, 45, 46, 74], and show the equivalence of both formulations under specific regularity assumptions.

In the mid 1990's, the development of computer power gave the research community the possibility to realize massive numerical computations for discretization of PDEs. Initially motivated by fields such as computational fluid dynamics [82] which itself dates back to the 1970's, researchers started developing very efficient methods to compute numerical solutions to the Hamilton-Jacobi equations. *Level Set Methods* and *Fast Marching Methods* [128, 142, 127], which find their roots in this research are very powerful tools which enable the computations of such solutions.

The original formulation of Isaacs (a static Hamilton-Jacobi equation sometimes referred to as Hamilton-Jacobi-Isaacs or HJI PDI), and its viability counterpart both share the same difficulty however: the value function which they attempt to compute is discontinuous, and can have infinite values. These features are very undesirable for numerical computations. This difficulty is well known and motivated numerous algorithms and methods to alleviate it. These methods either try to solve for this ill-behaved value function directly [139, 14], or try to transform the problem in a numerically more tractable problem [14]. In both cases, the difficulties posed by these numerical issues made it very hard to compute accurate solutions to the problem (even though mathematical proofs of convergence were given, practical cases were observed to converge very slowly).

The work presented in this chapter is part of a joint research effort which started in Berkeley in the mid 1990's and which to our best knowledge is the first to have successfully posed the problem of reachability in a Hamilton-Jacobi format enabling accurate computations. A time dependent Hamilton-Jacobi formulation of the reachability problem was proposed in [147, 149]. Initial implementations were realized later in [118, 117]. The actual mathematical proof of the validity of this method was recently completed in [116]. The main contribution of this work, which will be explained through this chapter, is to transform the problem of computing a ill-behaved value function into another one, which we can guarantee to be continuous.

8.1.2 Formal definition of the reachability problem

In this section we formally define the reachable set for a system, discuss a few of its properties, and formulate a terminal value HJI PDE whose solution describes it.

We model our system with the ordinary differential equation

$$\frac{dx}{dt} = \dot{x} = f(x, a, b), \tag{8.1}$$

where $x \in \mathbb{R}^n$ is our state, $a(\cdot)$ is the input for player I and $b(\cdot)$ is the input for player II.

Assumption 1. *The input signals are drawn from the following sets*

$$a(\cdot) \in \mathfrak{A}(t) \triangleq \{\phi : [t, 0] \to \mathcal{A} | \phi(\cdot) \text{ is measurable}\}$$
$$b(\cdot) \in \mathfrak{B}(t) \triangleq \{\phi : [t, 0] \to \mathcal{B} | \phi(\cdot) \text{ is measurable}\}$$

where $\mathcal{A} \subset \mathbb{R}^{n_a}$ and $\mathcal{B} \subset \mathbb{R}^{n_b}$ are compact and $t \in [-T, 0]$ for some $T > 0$. We will consider input signals which agree almost everywhere to be identical.

Assumption 2. *The flow field $f : \mathbb{R}^n \times \mathcal{A} \times \mathcal{B} \to \mathbb{R}^n$ is uniformly continuous, bounded, and Lipschitz continuous in x for fixed a and b. Consequently, given a fixed $a(\cdot) \in \mathfrak{A}(t)$, $b(\cdot) \in \mathfrak{B}(t)$ and initial point, there exists a unique trajectory solving (8.1).*

Solutions of (8.1) are trajectories of our system and will be denoted by

$$\xi_f(s; x, t, a(\cdot), b(\cdot)) : [t, 0] \to \mathbb{R}^n$$

where

$$\begin{aligned} \frac{d}{ds}\xi_f(s; x, t, a(\cdot), b(\cdot)) &= f(\xi_f(s; x, t, a(\cdot), b(\cdot)), a(s), b(s)) \text{ almost everywhere,} \\ \xi_f(t; x, t, a(\cdot), b(\cdot)) &= x. \end{aligned}$$

Note that we employ a semi-colon to distinguish between the argument s of ξ_f and the trajectory parameters t, x, $a(\cdot)$ and $b(\cdot)$.

Assumption 3. *The* target set *$\mathcal{G}_0 \subset \mathbb{R}^n$ for our reachability problem is closed and can be represented as the zero sublevel set of a bounded and Lipschitz continuous function $v_0 : \mathbb{R}^n \to \mathbb{R}$*

$$\mathcal{G}_0 = \{x \in \mathbb{R}^n | v_0(x) \le 0\}. \tag{8.2}$$

We assume that player I will try to steer the system away from the target with her input $a(\cdot)$, and player II will try to steer the system towards the target with her input $b(\cdot)$. For readers who prefer a more intuitive understanding of the inputs, consider that in our example the target set will represent the capture set in a pursuit-evasion game. Our control will then be player I and the adversarial disturbance will be player II.

In a differential game setting, it is important to address what information the players know about each other's decisions. To specify our information pattern, define first a *strategy* for the second player as a map $\gamma : \mathfrak{A}(t) \to \mathfrak{B}(t)$ which specifies an input signal for player II as a function of the input signal that player I chooses. We will allow player II to use only *nonanticipative strategies*; that is strategies

$$\gamma \in \Gamma(t) \triangleq \{\beta : \mathfrak{A}(t) \to \mathfrak{B}(t) | a(r) = \hat{a}(r) \forall r \in [t, s] \implies \beta[a](r) = \beta[\hat{a}](r) \forall r \in [t, s]\}.$$

It will turn out that this choice gives an advantage to player II over player I. A more detailed description of the information patterns is available in [45, 46].

Note that in our formulation of the problem, a trajectory starts at some initial time $t < 0$ and we would like to know if it has passed into or through the target set by time zero. We will sometimes want to discuss the length of time that a trajectory has had to evolve; we adopt the differential game notation $\tau = -t$ to denote this positive quantity. We use the free variables s and r to denote times in the range $[t, 0]$.

To solve the backwards reachability problem, we want to determine the *backwards reachable set* $\mathcal{G}(\tau)$ for $\tau \in [0, T]$. Remembering that $t = -\tau$, we define this set as

$$\mathcal{G}(\tau) \triangleq \{x \in \mathbb{R}^n | \exists \gamma \in \Gamma(t), \forall a(\cdot) \in \mathfrak{A}(t), \exists s \in [t, 0], \xi_f(s; x, t, a(\cdot), \gamma[a](\cdot)) \in \mathcal{G}_0\}. \tag{8.3}$$

Informally, $\mathcal{G}(\tau)$ is the set of states from which there exists strategies for player II that for all inputs of player I will generate trajectories which lead to the target set within time τ.

Theorem 6 (Modified HJE formulation of the reachability problem). *Let $v : \mathbb{R}^n \times [-T, 0] \to \mathbb{R}$ be the viscosity solution of the terminal value HJI PDE*

$$\begin{aligned} D_t v(x,t) + \min[0, H(x, D_x v(x,t))] &= 0, \qquad \textit{for } t \in [-T, 0], x \in \mathbb{R}^n; \\ v(x, 0) &= v_0(x), \quad \textit{for } x \in \mathbb{R}^n; \end{aligned} \tag{8.4}$$

where

$$H(x, p) = \max_{a \in \mathcal{A}} \min_{b \in \mathcal{B}} p^T f(x, a, b). \tag{8.5}$$

The zero sublevel set of v describes $\mathcal{G}(\tau)$

$$\mathcal{G}(\tau) = \{x \in \mathbb{R}^n | v(x,t) \leq 0\}.$$

The significance of this theorem is that we can harness well developed numerical schemes from the level set literature to compute accurate approximations of $v(x, t)$, and therefore accurate approximations of $\mathcal{G}(\tau)$, for even complicated nonlinear dynamics. The full proof of this theorem is presented in [116]. An alternate proof was derived at the same time and is available in [105, 106].

8.2 Link with viability and minimum time

In the following sections, we review a variety of alternative techniques for finding the backwards reachable set. In Section 8.2.3, we prove the equivalence between our formulation and other formulations from optimal control.

In the discussion below, we restrict ourselves to the optimal control case of a single input attempting to drive the system into the target set. The dynamics in this case are

$$\frac{dx}{dt} = \dot{x} = f(x, b), \tag{8.6}$$

because this restriction is equivalent to removing player I from the game in section 8.1. As a consequence, we no longer need worry about the strategy of player II and can define the reachable set in the slightly simpler form

$$\mathcal{G}(\tau) \triangleq \{x \in \mathbb{R}^n | \exists b(\cdot) \in \mathfrak{B}(t), \exists s \in [t, 0], \xi_f(s; x, t, b(\cdot)) \in \mathcal{G}_0\}. \tag{8.7}$$

8.2.1 Minimum time to reach

In this section we examine two formulations of the reachability problem which can theoretically compute the exact reachable set $\mathcal{G}(\tau)$. Analytically, their results are equivalent to one another and to those produced by our formulation (see section 8.2.3), but their practical implementations differ.

The first, and perhaps most basic among all reachability formulations draws on the *minimum time to reach* function, originally defined in optimal control [43, 144] which we will define as

$$\mathsf{t}(x, b(\cdot)) = \begin{cases} \min\{\tau | \xi(0; x, -\tau, b(\cdot)) \in \mathcal{G}_0\}, & \text{if } \{\tau | \xi(0; x, -\tau, b(\cdot)) \in \mathcal{G}_0\} \neq \emptyset; \\ +\infty & \text{otherwise.} \end{cases} \tag{8.8}$$

Note that $\mathsf{t}(x, b(\cdot)) \geq 0$, because our trajectories always start from some $t = -\tau \leq 0$. The *minimum time to reach* function is then

$$\mathsf{T}(x) = \inf_{b(\cdot) \in \mathfrak{B}(-\infty)} \mathsf{t}(x, b(\cdot)). \tag{8.9}$$

For more mathematical details on this formulation, see [13, 14] (the definitions (8.8) and (8.9) are respectively equivalent to the functions $t_x(b)$ and $T(x)$ discussed in [13, 14], although the statement of (8.8) differs slightly due to the shifted time domain in our formulation). Those references also discuss how to move these definitions and most of the results described below into the differential game setting.

Based on (8.8) and (8.9), it is easy to deduce the following fact.

Fact 1. *The reachable set $\mathcal{G}(\tau)$ is the τ sublevel set of the minimum time to reach function*

$$\mathcal{G}(\tau) = \{x \in \mathbb{R}^n | \mathsf{T}(x) \leq \tau\}. \tag{8.10}$$

Although there exists a static Hamilton-Jacobi equation whose solution is $\mathsf{T}(x)$, it involves infinite boundary conditions applied along a boundary which cannot always be determined a priori.

8.2.2 A Formulation from Viability Theory

An alternative approach to reachability is based on *viability theory* [6] and *set valued analysis* [10]. The first step is to transform our ordinary differential equation with one input (8.6) into an input free *differential inclusion* [9].

$$\frac{dx}{dt} = \dot{x} \in F(x) \triangleq \{f(x, b) \in \mathbb{R}^n | b \in \mathcal{B}\} \tag{8.11}$$

almost everywhere in t. The right hand side of this equation F is a *set valued map*—for any x, $F(x) \subseteq \mathbb{R}^n$. For reasons related to the existence of solutions to (8.11) [6], viability theory typically assumes that $F(x)$ is convex, compact and nonempty for any $x \in \mathbb{R}^n$ and F is uppersemicontinuous with linear growth in x; an F with these properties is called a *Marchaud map.* We make the following assumption in any subsequent discussion involving viability theory.

Assumption 4. *The set valued map F is a Marchaud map. If the flow field f satisfies Assumptions 1 and 2, then the only additional constraint we need to put on F to guarantee that it is Marchaud is that its value $F(x)$ is a convex set for any $x \in \mathbb{R}^n$.*

A solution of (8.11) is a trajectory $\zeta_F(\tau; x) : [0, \infty] \to \mathbb{R}^n$ such that

$$\begin{aligned} \frac{d}{d\tau}\zeta_F(\tau; x) &\in F(\zeta_F(\tau; x)) \text{ almost everywhere,} \\ \zeta_F(0; x) &= x. \end{aligned}$$

We use the symbol $\rightsquigarrow$ for set valued maps in the same way that $\to$ is used for regular functions. Define $\mathcal{S}_F : \mathbb{R}^n \rightsquigarrow C([0, +\infty], \mathbb{R}^n)$ so that $\mathcal{S}_F(x)$ is the set of absolutely continuous trajectories leading from a point $x \in \mathbb{R}^n$. Thus $\zeta_F(\cdot; x) \in \mathcal{S}_F(x)$.

With these definitions in place, we can define the τ finite horizon *capture basin* of the target set $\mathcal{G}_0$ under differential inclusion F as

$$\mathsf{Capt}_F(\mathcal{G}_0, \tau) \triangleq \{x \in \mathbb{R}^n | \exists \zeta_F(\cdot; x) \in \mathcal{S}_F(x), \exists \hat{\tau} \in [0, \tau], \zeta_F(\hat{\tau}; x) \in \mathcal{G}_0\}. \tag{8.12}$$

In words, the capture basin is the set of states from which emanates at least one trajectory leading to $\mathcal{G}_0$ in time τ or less. The equivalence of the capture basin and reachable set is stated in Theorem 7. The concept of capture basin can also be extended to the differential game setting in the form of leadership and discriminating kernels, see [45, 46] for more details.

Based on earlier work [75, 139] an algorithm has been developed to compute $\mathsf{Capt}_F(\mathcal{G}_0, \tau)$ directly [46]. Their scheme divides the state space into a fixed grid. Each point on the grid is labeled by a boolean variable which is true if that mesh point is within the capture basin—initially, only points in $\mathcal{G}_0$ will have a true value. Given a fixed timestep $\Delta\tau$, they compute a discrete approximation of F at each mesh point and hence determine for that point what other mesh points can be reached in a single timestep. They then iterate, setting the value of a mesh point to true if it can be reached from any mesh point that is already true (this procedure is the forward Euler scheme adapted to set valued differential inclusions). Any mesh point which is true after $\tau/\Delta\tau$ timesteps is in $\mathsf{Capt}_F(\mathcal{G}_0, \tau)$. The cost of the algorithm is doubly exponential in the dimension of the state space, because in general F must be discretized for each mesh point separately.

The approximation provided by this algorithm can be shown to converge to the true capture basin as the grid is refined [46], although explicit convergence rates are only given in specific cases [140, 47]. Unlike the methods based on PDEs, this algorithm can guarantee an overapproximation of the reachable set if the discretization of F is sufficiently overapproximated. Because of the boolean nature of the data on the grid, this algorithm has no problem treating systems whose time to reach function is discontinuous. Conversely, absolutely no subgrid resolution of the capture basin is possible because every grid point is either completely inside or completely outside of it. Adaptive mesh refinement in the neighborhood of the capture basin's boundary has been used to develop more accurate approximations, but the cost grows very quickly as the grid is made finer.

A slightly modified version of this algorithm can be used to compute underapproximations of the viscosity solution $\mathsf{T}(x)$ of the static Hamilton-Jacobi equation [46, 16].

8.2.3 Equivalence of the different formulations

The different formulations above generate the same reachable set:

Theorem 7 (Equivalence of Convergent Formulations). *The following sets are equivalent*

$$\mathcal{G}(\tau) = \mathsf{Capt}_F(\mathcal{G}_0, \tau) = \{x \in \mathbb{R}^n | \mathsf{T}(x) \leq \tau\} = \{x \in \mathbb{R}^n | v(x,t) \leq 0\}$$

where $\tau = -t > 0$.

We keep our proof brief in order to maintain the focus of this section on the comparison of various reachability algorithms.

Proof. Because the set of trajectories which solve (8.6) is the same as the set of trajectories which solve (8.11) [57, Section 0.4], the equivalence of $\mathcal{G}(\tau)$ and $\mathsf{Capt}_F(\mathcal{G}_0, \tau)$ is a trivial consequence of their definitions. In the viability literature, the τ sublevel set of $\mathsf{T}(x)$ is given as an alternative definition for the capture basin in time τ [7, Definition 2.7.4], so clearly the second and third sets are equivalent.

Theorem 6 provides the equivalence of $\mathcal{G}(\tau)$ and the zero sublevel set of $v(x, -\tau)$; the proof is developed in [116]. An alternative method of achieving the same equivalence result is to show that the τ sublevel set of $\mathsf{T}(x)$ and the zero sublevel set of $v(x, -\tau)$ are the same: we now prove that

$$\{x \in \mathbb{R}^n | \mathsf{T}(x) \leq \tau\} = \{x \in \mathbb{R}^n | v(x, -\tau) \leq 0\}, \tag{8.13}$$

and thereby draw some connections with previous results from viability theory.

We begin with some additional definitions from viability theory. The *epigraph* of some function $\psi : \mathbb{R}^n \to R^+$ is the set of all points above the graph of that function

$$\mathsf{Epi}(\psi) \triangleq \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^+ | \psi(x) \leq y\}. \tag{8.14}$$

The *viability kernel* of a differential inclusion Ψ for some set $\mathcal{M}$ is the set of all points from which a trajectory of Ψ begins that stays in $\mathcal{M}$ for all time

$$\mathsf{Viab}_\Psi\left(\mathcal{M}\right) \triangleq \{x \in \mathcal{M} | \exists \zeta_\Psi((\cdot); x) \in \mathcal{S}_\Psi(x), \forall s \in \mathbb{R}^+, \zeta_\Psi(s; x) \in \mathcal{M}\}. \tag{8.15}$$

The concepts of viability kernel and capture basin are related, although that relationship is not important for the discussion below. Let $\mathsf{ch}\left(\mathcal{M}\right)$ be the convex hull of $\mathcal{M}$, and then define

$$\tilde{F}(x) \triangleq \mathsf{ch}\left(F(x) \cup \{0\}\right). \tag{8.16}$$

Now define the two auxiliary set valued maps $\Phi, \Psi : \mathbb{R}^{n+1} \rightsquigarrow \mathbb{R}^{n+1}$

$$\Phi((x, y)) \triangleq \begin{cases} F(x) \times \{-1\}, & \text{if } x \notin \mathcal{G}_0; \\ \tilde{F}(x) \times [-1, 0] & \text{if } x \in \mathcal{G}_0; \end{cases} \tag{8.17}$$

$$\Psi((x, y)) \triangleq \begin{cases} \tilde{F}(x) \times \{-1\}, & \text{if } x \notin \mathcal{G}_0; \\ \tilde{F}(x) \times [-1, 0] & \text{if } x \in \mathcal{G}_0; \end{cases} \tag{8.18}$$

where $F : \mathbb{R}^n \rightsquigarrow \mathbb{R}^n$ is the Marchaud map (8.11), $\tilde{F} : \mathbb{R}^n \rightsquigarrow \mathbb{R}^n$ is the Marchaud map (8.16), $x \in \mathbb{R}^n$ and $y \in \mathbb{R}$. Because we use Φ and Ψ below as the right hand side of some differential inclusions, let us introduce the notation

$$\tilde{\zeta}_\Psi(\tau; (x, y)) = \begin{bmatrix} \zeta_\Psi(\tau; x), & \underline{\zeta}_\Psi(\tau; y) \end{bmatrix}$$

to break apart the two components $\zeta_\Psi(\tau; x) \in \mathbb{R}^n$ and $\underline{\zeta}_\Psi(\tau; y) \in \mathbb{R}$ of the trajectory $\tilde{\zeta}_\Psi(\tau; (x, y)) \in \mathbb{R}^{n+1}$ which solves

$$\begin{aligned} \frac{d}{d\tau} \tilde{\zeta}_\Psi(\tau; (x, y)) &\in \Psi(\tilde{\zeta}_\Psi(\tau; (x, y))) \text{ almost everywhere}, \\ \tilde{\zeta}_\Psi(0; (x, y)) &= (x, y). \end{aligned}$$

A similar definition applies for $\tilde{\zeta}_\Phi(\tau;(x,y))$ and its components. By the construction of Ψ,

$$\begin{aligned} &\forall \zeta_\Psi(\cdot;x), \exists \zeta_{\tilde{F}}(\cdot;x) \text{ such that } \zeta_\Psi(\tau;x) = \zeta_{\tilde{F}}(\tau;x), \forall \tau \geq 0; \\ &\forall \zeta_{\tilde{F}}(\cdot;x), \exists \zeta_\Psi(\cdot;x) \text{ such that } \zeta_{\tilde{F}}(\tau;x) = \zeta_\Psi(\tau;x), \forall \tau \geq 0. \end{aligned} \tag{8.19}$$

Finally, define the set $\mathcal{H} \subset \mathbb{R}^{n+1}$

$$\mathcal{H} \triangleq \mathbb{R}^n \times \mathbb{R}^+$$

With these definitions in place, we can demonstrate (8.13) by proving the four equations

$$\{(x,t) \in \mathbb{R}^n \times [T,0] | \mathsf{T}(x) \leq -t\} = \{(x,t) \in \mathbb{R}^n \times [T,0] | (x,-t) \in \mathsf{Epi}(\mathsf{T})\} \tag{8.20}$$

$$\mathsf{Epi}(\mathsf{T}) = \mathsf{Viab}_\Phi(\mathcal{H}) \tag{8.21}$$

$$\mathsf{Viab}_\Phi(\mathcal{H}) = \mathsf{Viab}_\Psi(\mathcal{H}) \tag{8.22}$$

$$\{(x,t) \in \mathbb{R}^n \times [T,0] | (x,-t) \in \mathsf{Viab}_\Psi(\mathcal{H})\} = \{(x,t) \in \mathbb{R}^n \times [T,0] | v(x,t) \leq 0\} \tag{8.23}$$

Proving (8.20) is a simple matter of substituting the definition (8.14) of the epigraph. Theorem 8 shows that (8.21) holds, while Lemma 5 shows that (8.22) holds. Finally, Lemma 6 demonstrates (8.23). This sequence of equations may seem like a lot of work to generate an alternative proof for part of Theorem 7, but we feel that it makes some worthwhile connections between viability theory and our formulation of reachability. In particular, we would like to draw attention to the equivalence of allowing the system's dynamics to be frozen everywhere or in only the target set. Lemma 5 shows that the viable set under Φ (which allows frozen dynamics via $\tilde{F}$ in the target set only) is the same as the viable set under Ψ (which allows frozen dynamics everywhere). We anticipate future payoffs from this equivalence and the fact that Ψ (and its associated ODE flow field $\tilde{f}$) modifies the system's dynamics more consistently throughout the state space. $\square$

The remainder of this section is devoted to the theorems and lemmas needed to prove (8.20)–(8.23). The following theorem of Cardaliaguet, Quincampoix and Saint-Pierre states that (8.21) holds.

Theorem 8 (Cardaliaguet - Quincampoix - Saint-Pierre). *The epigraph of* T *is the viability kernel of* Φ *in* $\mathcal{H}$

$$\mathsf{Epi}(\mathsf{T}) = \mathsf{Viab}_\Phi(\mathcal{H}).$$

Proof. See [46, Theorem 3.2] □

Before proceeding to show that (8.22) holds, we need an intermediate result.

Lemma 4. *A point is in the viability kernel of $\mathcal{H}$ under Ψ if and only if a trajectory beginning at that point reaches $\mathcal{G}_0$ in finite time.*

$$(x, y) \in \mathsf{Viab}_\Psi(\mathcal{H}) \iff \exists \zeta_{\tilde{F}}(\cdot; x) \in \mathcal{S}_{\tilde{F}}(x), \exists z \in [0, y], \zeta_{\tilde{F}}(z; x) \in \mathcal{G}_0.$$

Proof. **Case 1 ($\Longrightarrow$):** We prove the contrapositive. Assume that $\zeta_{\tilde{F}}(z; x) \notin \mathcal{G}_0$ for all $\zeta_{\tilde{F}}(\cdot; x) \in \mathcal{S}_{\tilde{F}}(x)$ and $z \in [0, y]$. By (8.19), for all $\tilde{\zeta}_\Psi(\cdot; (x, y)) \in \mathcal{S}_\Psi((x, y))$, $\zeta_\Psi(z; x) \notin \mathcal{G}_0$. By (8.18), the corresponding $\dot{\underline{\zeta}}_\Psi(z; y) \in \{-1\}$ and thus $\underline{\zeta}_\Psi(z; y) = y - z$. In particular, $\underline{\zeta}_\Psi(y; y) = 0$, which implies $\tilde{\zeta}_\Psi(y; (x, y)) \in \partial\mathcal{H}$. Since $\dot{\underline{\zeta}}_\Psi(y; y) \in \{-1\}$, we also have that $\dot{\tilde{\zeta}}_\Psi(y; (x, y))$ points out of $\mathcal{H}$, and by standard continuity arguments about the trajectory, $\tilde{\zeta}_\Psi(y + \epsilon; (x, y)) \notin \mathcal{H}$ for any $\epsilon > 0$. Since $\tilde{\zeta}_\Psi(\cdot; (x, y))$ was arbitrary, there is no trajectory starting at (x, y) that stays in $\mathcal{H}$ forever, and hence $(x, y) \notin \mathsf{Viab}_\Psi(\mathcal{H})$.

Case 2 ($\Longleftarrow$): Given that there exists $\zeta_{\tilde{F}}(\cdot; x) \in \mathcal{S}_{\tilde{F}}(x)$ and $z \in [0, y]$ such that $\zeta_{\tilde{F}}(z; x) \in \mathcal{G}_0$, by (8.19) there exists a component $\zeta_\Psi(\cdot; x)$ of some $\tilde{\zeta}_\Psi(\cdot; (x, y)) \in \mathcal{S}_\Psi((x, y))$ such that $\zeta_\Psi(z; x) \in \mathcal{G}_0$. The remaining component $\underline{\zeta}_\Psi(\cdot; y)$ of $\tilde{\zeta}_\Psi(\cdot; (x, y))$ is free, and we choose

$$\frac{d}{d\tau}\underline{\zeta}_\Psi(\tau; y) = \begin{cases} -1, & \text{for } \tau \in [0, z]; \\ 0, & \text{for } \tau \in [z, \infty]; \end{cases}$$

(which is legal by (8.18)) so that

$$\underline{\zeta}_\Psi(\tau; y) = \begin{cases} y - \tau, & \text{for } \tau \in [0, z]; \\ y - z, & \text{for } \tau \in [z, \infty]. \end{cases}$$

Because $y - z \geq 0$, $\underline{\zeta}_\Psi(\tau; y) \in \mathbb{R}^+$ for all $\tau \geq 0$, $\tilde{\zeta}_\Psi(\tau; (x, y)) \in \mathcal{H}$, and therefore $(x, y) \in \mathsf{Viab}_\Psi(\mathcal{H})$. □

A simple consequence of Lemma 4 is that

$$(x, y) \in \mathsf{Viab}_\Psi(\mathcal{H}) \iff \exists \zeta_{\tilde{F}}(\cdot; x) \in \mathcal{S}_{\tilde{F}}(x), \exists z \in [0, y], \forall \tau \in [z, y], \zeta_{\tilde{F}}(\tau; x) \in \mathcal{G}_0 \qquad (8.24)$$

We are now ready to show that (8.22) holds.

Lemma 5. *The viability kernels in $\mathcal{H}$ under Φ or Ψ satisfy*

$$\mathsf{Viab}_\Phi(\mathcal{H}) = \mathsf{Viab}_\Psi(\mathcal{H}).$$

Proof. We want to show inclusion in both directions. We prove the inclusion in one direction. The first inclusion is trivial, because for all $(x,y) \in \mathbb{R}^{n+1}$ we have $\Phi((x,y)) \subseteq \Psi((x,y))$ and thus

$$\mathsf{Viab}_\Phi(\mathcal{H}) \subseteq \mathsf{Viab}_\Psi(\mathcal{H}).$$

$\square$

We outline a sketch of the proof for the inclusion in the other direction. To show inclusion in the other direction, assume $(x,y) \in \mathsf{Viab}_\Psi(\mathcal{H})$. By (8.15), there exists $\tilde{\zeta}_\Psi(\cdot;(x,y)) \in \mathcal{S}_\Psi((x,y))$ such that $\tilde{\zeta}_\Psi(\tau;(x,y)) \in \mathcal{H}$ for all $\tau \geq 0$. Using this solution of Ψ, we will construct a solution of Φ that remains in $\mathcal{H}$ for all time, and thus show that $(x,y) \in \mathsf{Viab}_\Phi(\mathcal{H})$.

Because $F(x)$ is compact and convex for all $x \in \mathbb{R}^n$, for any $\eta_{\tilde{F}} \in \tilde{F}(x)$ there exists $\eta_F \in F(x)$ and $\underline{b} \in [0,1]$ such that $\eta_{\tilde{F}} = \underline{b}\eta_F$. Therefore, for all $\tau \in [0,y]$ there exists $\underline{b}(\tau) \in [0,1]$ and $\eta_F(\tau) \in F(\zeta_\Psi(\tau;x))$ such that $\dot{\zeta}_\Psi(\tau;x) = \underline{b}(\tau)\eta_F(\tau)$. Let $\zeta_{\eta_F}(\cdot;x)$ be the trajectory associated with the differential inclusion

$$\dot{\zeta}_{\eta_F}(\tau;x) \in \{\eta_F(\tau)\} \text{ for all } \tau \in [0,y].$$

Note that the set might be a single point for some values of τ. Note also that one still needs to show in which sense $\zeta_{\eta_F}(\cdot;x)$ is a solution to the differential inclusion above. Based on this definition, $\dot{\zeta}_\Psi(\tau;x) = \underline{b}(\tau)\dot{\zeta}_{\eta_F}(\tau;x)$. Let $\sigma : [0,y] \to [0,y]$ be the pseudo-time variable

$$\sigma(\tau) = \int_0^\tau \underline{b}(\lambda)\, d\lambda.$$

By (8.24), $\zeta_\Psi(y;x) \in \mathcal{G}_0$ because $(x,y) \in \mathsf{Viab}_\Psi(\mathcal{H})$. Also,

$$\begin{aligned}
\zeta_\Psi(y;x) &= \zeta_\Psi(0;x) + \int_0^y \dot{\zeta}_\Psi(\lambda;x)\, d\lambda,\\
&= x + \int_0^y \dot{\zeta}_{\eta_F}(\lambda;x)\underline{b}(\lambda)\, d\lambda,\\
&= x + \int_0^{\sigma(y)} \dot{\zeta}_{\eta_F}(\sigma^{-1}(\rho);x)\, d\rho,\\
&= x + \int_0^{\sigma(y)} \dot{\zeta}_F(\rho;x)\, d\rho,\\
&= \zeta_F(\sigma(y);x),
\end{aligned}$$

where we have introduced the trajectory $\zeta_F(\tau;x) = \zeta_{\eta_F}(\sigma^{-1}(\tau);x)$. A justification of this variable change in the integration is given in [116]. Thus, we can see that $\zeta_F(\sigma(y);x) \in \mathcal{G}_0$. In addition, we have* that

$$\dot{\zeta}_F(\tau;x) = \dot{\zeta}_{\eta_F}(\sigma^{-1}(\tau);x) = \eta_F(\sigma^{-1}(\tau)) \in F(\zeta_\Psi(\sigma^{-1}(\tau);x)), \quad \text{for } \tau \in [0,\sigma(y)].$$

Therefore, we can construct a trajectory of Φ whose first n components are

$$\zeta_\Phi(\tau;x) = \begin{cases} \zeta_F(\tau;x), & \text{for } \tau \in [0,\sigma(y)];\\ \zeta_F(\sigma(y);x), & \text{for } \tau \geq \sigma(y). \end{cases}$$

Because $\zeta_\Phi(\tau;x) = \zeta_F(\tau;x) \in \mathcal{G}_0$ for $\tau \geq \sigma(y)$, according to (8.17) we can choose the last component of this trajectory as

$$\underline{\zeta}_\Phi(\tau;y) = \begin{cases} y-\tau, & \text{for } \tau \in [0,\sigma(y)];\\ y-\sigma(y), & \text{for } \tau \geq \sigma(y). \end{cases}$$

Since $\underline{\zeta}_\Phi(\tau;y) \geq y - \sigma(y) \in \mathbb{R}^+$ for all $\tau \geq 0$, $\tilde{\zeta}_\Phi(\tau;(x,y)) \in \mathcal{H}$ for all time and $(x,y) \in \mathsf{Viab}_\Phi(\mathcal{H})$. Consequently

$$\mathsf{Viab}_\Psi(\mathcal{H}) \subseteq \mathsf{Viab}_\Phi(\mathcal{H}).$$

Our final lemma shows that (8.23) holds.

*Note that this is not a differential inclusion in the traditional setting $\dot{x} \in F(x)$ since the x is not the same in the time derivative operator and the F. It remains to be proved that the formula stated defines a solution to the desired differential inclusion.

Lemma 6. *The zero sublevel set of $v(x,t)$ is a particular subset of the viability kernel of Ψ in $\mathcal{H}$*

$$\{(x,t) \in \mathbb{R}^n \times [T,0] | (x,-t) \in \mathsf{Viab}_\Psi(\mathcal{H})\} = \{(x,t) \in \mathbb{R}^n \times [T,0] | v(x,t) \leq 0\} \tag{8.25}$$

Proof. Start with $(x,-t) \in \mathsf{Viab}_\Psi(\mathcal{H})$. By (8.24)

$$\exists \zeta_{\tilde{F}}(\cdot;x) \in \mathcal{S}_{\tilde{F}}(x), \zeta_{\tilde{F}}(-t;x) \in \mathcal{G}_0. \tag{8.26}$$

Now consider constructing a finite horizon optimal control problem with terminal payoff

$$V(x,\tau) = \inf_{\hat{\zeta}_{\tilde{F}}(\cdot;x) \in \mathcal{S}_{\tilde{F}}(x)} v_0(\hat{\zeta}_{\tilde{F}}(\tau;x)), \tag{8.27}$$

where v_0 defines our target set (8.2). It can be shown [13, 57] that $V(x,\tau)$ solves the Hamilton-Jacobi PDE

$$\begin{aligned} D_\tau V + \hat{H}(x, D_x V) &= 0, & &\text{for } \tau \in [0,\infty[,\ x \in \mathbb{R}^n; \\ V(x,0) &= v_0(x), & &\text{for } x \in \mathbb{R}^n; \end{aligned}$$

where

$$\hat{H}(x,p) = \max_{q \in \tilde{F}(x)} -p^T q, = \max_{q \in F(x)} [0, -p^T q].$$

With a change of variables $\tau = -t$, it can be shown that $v(x,t) = V(x,\tau)$ [71]. From (8.26) and (8.27) we can see

$$v(x,t) = V(x,\tau) = \inf_{\hat{\zeta}_{\tilde{F}}(\cdot;x) \in \mathcal{S}_{\tilde{F}}(x)} g(\hat{\zeta}_{\tilde{F}}(\tau;x)) \leq g(\zeta_{\tilde{F}}(\tau;x)) \leq 0.$$

The steps above can all be reversed to prove the converse direction of (8.25). □

8.3 Analytical closed form solution of the Mayer problem

As was noted in Section 8.1, some formulations of the reachability problem rely on a modified Hamilton-Jacobi equation, which is originally a formulation of the Mayer problem. In the early stage of this work, two unanswered research questions had struck the control community, which triggered the results presented in this section. We list these two questions, which provide the context of the present work.

(i) In trying to compute the reachable set (prior to actual implementation of level set methods on control problems [118, 117]), several authors [152] observed separation of characteristics, which they could not explain. *(ii)* In other works [151] using a new formulation of the reachability problem [118], authors were observing the disappearance of level sets in reachability computations and attributed this fact to numerical dissipation.

Question (i) has been resolved since by looking at an earlier work of Merz [112], available in a more modern version [114], which shows how to match characteristics correctly in order to compute the solution of the reachability problem using traditional differential games techniques. Question (ii) was resolved using the method presented in this section, subsequently published in [26].

The method presented below relies on the method of characteristics [70, 145], which was historically the first method used to solve (analytically) the Hamilton-Jacobi equation, at a time when computers were unavailable. The technique of Isaacs [85] is based on it, and was used intensely afterwards (see [100, 15] for a more modern treatment of the subject). A new approach linking the viscosity solution to the method of characteristics and the work of Isaacs was recently developed by Melikyan [109], but we were unable to use these results for this work.

8.3.1 Construction of C_0 solutions of a 1D Hamilton-Jacobi equation

In this section and the following, we explain how to construct continuous solution to the following one dimensional Hamilton-Jacobi equation:

$$\begin{aligned} D_t v(x,t) + H(x, D_x v(x,t)) = 0, \qquad & \text{for } t \in [-T, 0], x \in \mathbb{R}; \\ v(x,0) = v_0(x), \quad & \text{for } x \in \mathbb{R}^n; \end{aligned} \tag{8.28}$$

associated with the dynamics (8.1), where the Hamiltonian is given by:

$$H(x,p) = \max_{a \in \mathcal{A}} \min_{b \in \mathcal{B}} p^T f(x,a,b). \tag{8.29}$$

The problem (8.28)-(8.29) is sometimes referred to as the Mayer problem [74, 43, 144, 14, 13, 8]. Note that (8.4)-(8.5) is in fact a modified version of the Mayer problem. The intuitive interpretation of the Mayer problem is the following. Given a point x in the space and a terminal payoff function v_0, the value $v(x, T)$ gives the optimal payoff obtained by both players in T time units. Optimal means maximal for player a and minimal for player b.

Sections 8.3.2 and 8.3.3 respectively solve the cases in which a single shock or a single void appear in the solution. Section 8.3.4 generalizes this to a problem with an arbitrary number of shocks and voids. We do not provide any proof that the solution constructed below is the viscosity solution. We only use the method of characteristics to analytically explain numerical phenomena observed in practical implementations [118, 151].

Inspired by earlier work [147, 149], we consider a differential game with input $a \in \mathcal{A} = [-1, 1]$ and disturbance $b \in \mathcal{B} = [-1, 1]$, for a specific dynamical system of the following form:

$$\dot{x} = f(x, a, b) = g_0(x) + a g_1(x) + b g_2(x) \tag{8.30}$$

where $x \in \mathbb{R}$, $g_0, \ g_1, \ g_2 : \mathbb{R} \to \mathbb{R}$ are chosen so that the hypotheses of the previous section hold. In one dimension, the optimal input and worst disturbance can be computed explicitly and (8.28) can be written in the simplistic form:

$$D_t v(x,t) + (f(x) + (\ |g_1(x)| - |g_2(x)| \ ) \ \mathrm{sgn}(D_x v(x,t)) \, D_x v(x,t) = 0 \tag{8.31}$$

This formulation is somewhat simplistic, and does not generalize easily to multiple dimensions, though we will be able to adapt it in the next sections. We define the set valued map $F(x) = [\ f(x) - | \ |g_1(x)| - |g_2(x)| \ | \ , \ f(x) + | \ |g_1(x)| - |g_2(x)| \ | \ ]$, and notice that the sign of $D_x v(x,t)$ makes the Hamiltonian switch between the two extremal values of F. We show the construction in forward time for simplicity, but the examples will be constructed in backward time, as is required for the Mayer problem (8.28)-(8.29). We use the following notations and assumptions (which could be weakened to be more general if needed):

- We rewrite (8.31) as $D_t v(x,t) + \phi_\sigma(x) D_x v(x,t) = 0$ where we have introduced $\phi_\sigma(x) := f(x) + \sigma| \ |g_1(x)| - |g_2(x)| \ |$, with $\sigma = \pm 1$. σ is thus determined by the sign of the gradient $D_x v$ of v (see Figure 8.2). Let us assume that given σ, ϕ_σ is at least C_0, and $\forall x \in \mathbb{R}, \ \phi_\sigma(x) \neq 0$.

- Let $x_c \in \mathbb{R}$ be arbitrary for now. Call $\Phi_\sigma(x) = \int_{x_c}^{x} \frac{du}{\phi_\sigma(u)}$ and assume it is defined.
- Assume $\{ (x,t) \mid (x,t) \in \mathbb{R} \times \mathbb{R}^+,\ \exists x_0 \in \mathbb{R},\ \text{s.t. } x = \Phi_\sigma^{-1}(\Phi_\sigma(x_0) + t - t_0) \} = \mathbb{R} \times \mathbb{R}^+$. This statement guarantees that each family of characteristics $(x(t), t)$, spans the whole characteristic space $\mathbb{R}^+ \times \mathbb{R}$. This situation is sometimes referred to as *characteristic boundary conditions* [70, 145].
- Assume that $v(x,t) = v_0(\Phi_\sigma^{-1}(\Phi_\sigma(x) - (t - t_0)))$ is the C_1 solution of $v_t + \phi_\sigma(x) v_x = 0$ obtained with the classical method of characteristics for $\sigma = \pm 1$.

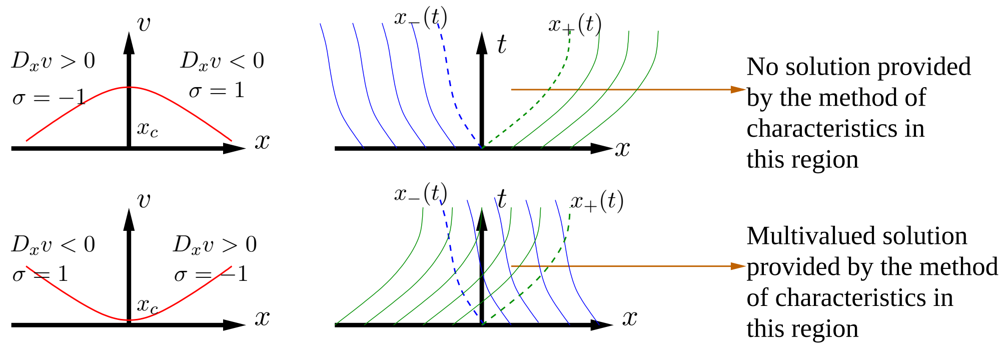


Figure 8.2: Problems usually encountered using characteristics: separation or void (top), collapse or shock (bottom)

Before proceeding further, we show an example, inspired by Clarke et al. [57], in which the classical method of characteristics provides the viscosity solution.

Example (classical method of characteristics): Consider solving the following problem, written in the terminology originally proposed by Tomlin et al. [147, 149, 150]:

$$D_t v(x,t) + \min_{d \in [-1,1]} d x D_x v(x,t) = 0 \tag{8.32}$$

with terminal condition $v(x,1) = v_0(x) = x$, and $t \in [-\infty, 1]$ (backwards time). This example is one of the few cases in which the classical method of characteristics works: one can carefully check that the characteristics span the $\mathbb{R} \times]-\infty, 1]$ in the (x,t) plane. The characteristics are given by:

$$\begin{cases} x > 0 & t = 1 - \log(x/x_0) \\ x < 0 & t = 1 + \log(x/x_0) \end{cases} \tag{8.33}$$

These two families of characteristic curves are separated by a vertical one ending at $(x,t) = (0,1)$. We now can compute the solution at each point of $\mathbb{R}\times]-\infty,1]$:

$$\begin{cases} x \geq 0 & v(x,t) = v_0(xe^{t-1}) \\ x \leq 0 & v(x,t) = v_0(xe^{-(t-1)}) \end{cases} \tag{8.34}$$

where $v_0(x) = x$ here but it could in fact have been any monotonically increasing function.

8.3.2 Single C_0, PC_1 shock construction procedure

We now show how to construct a C_0, PC_1 weak solution of (8.31): we find the domains of validity of the characteristics and explain how to construct the solution when they intersect. The present construction is in forward time. The same procedure applies in backward time with appropriate sign changes.

Assume here that v_0 is decreasing in $]-\infty, x_c]$, and increasing in $x \in [x_c, \infty[$ (i.e. $\sigma = +1$ for $x \in]-\infty, x_c]$ and $\sigma = -1$ for $x \in [x_c, \infty[$). v_0 has thus a local minimum at x_c. A shock appears instantaneously at x_c: call $x_\sigma(t) = \Phi_\sigma^{-1}(\Phi_\sigma(x_c) + (t - t_0))$ the respective characteristics emanating from x_c. Then, $x_-(t) \leq x_+(t)$ at least in a vicinity of (x_c, t_0), which means the characteristics cross and the solution is not uniquely defined.

Initial data matching: Since $v_0(x_c)$ is a local minimum of v_0 (in fact global) and v_0 is C_0, then $\exists (x_l, x_r) \in [-\infty, x_c] \times [x_c, \infty]$ and $\exists m :]x_l, x_c] \to [x_c, x_r[$ bijective, such that $\forall x \in]x_l, x_c]$, $v_0(x) = v_0(m(x))$. Consider the set $\{(x,t) | t \geq 0 \wedge x_-(t) \leq x \leq x_+(t)\}$. In this set, consider the curve $x = \mathcal{S}_c(t)$ defined by the solution of the following system:

$$s \in]x_l, x_c] \begin{cases} t - t_0 = \Phi_+(x) - \Phi_+(s) \\ t - t_0 = \Phi_-(x) - \Phi_-(m(s)) \end{cases} \tag{8.35}$$

We now define v which is by construction a C_0 a.e. solution of (8.31):

$$\begin{aligned} v(x,t) &= v_0(\Phi_+^{-1}(\Phi_+(x) - (t - t_0))) \text{ if } x \leq \mathcal{S}_c(t) \\ v(x,t) &= v_0(\Phi_-^{-1}(\Phi_-(x) - (t - t_0))) \text{ if } x \geq \mathcal{S}_c(t) \end{aligned} \tag{8.36}$$

Thus, we have matched the initial data: we defined regions in which the two families of characteristics are valid, such that the overall $v(x,t)$ is C_0 because each intersecting pair of characteristics originate from points with same initial value $v(x,t_0) = v_0(x)$. The curve $(\mathcal{S}_c(t),t)$ is a specific case of a singular surface called dispersal surface by Isaacs [85].

Construction procedure for a single shock:

Input: Initial conditions of the HJE PDE (8.28).
Output: C_0, PC_1 solution of (8.28) for the single shock case.

```
1  Find $x_c := \text{argmin}(v_0(x))$ (maximum if backwards in time)
2  Generate the bijection $m : ]x_l, x_c] \rightarrow [x_c, x_r[$ s.t. $\forall x \in ]x_l, x_c],\ v_0(x) = v_0(m(x))$
3  Solve equation (8.35) and construct $x = \mathcal{S}_c(t)$
4  Define $v(x,t)$ according to (8.36)
```

Example (C_0, PC_1 shock): Consider the following HJE (8.28), written in the same terminology as previously [147, 149, 150]:

$$D_t v(x,t) + \min_{d \in [-1,1]} d D_x v(x,t) = 0 \tag{8.37}$$

where time is backwards, with the following terminal conditions at $T = 1$:

$$\begin{cases} v_0(x,1) = \frac{1}{1+x^2} \text{ if } x \in \mathbb{R}_- \\ v_0(x,1) = \frac{1}{1+(2x)^2} \text{ if } x \in \mathbb{R}_+ \end{cases} \tag{8.38}$$

Here we trivially have $x_c = 0$, $m : x_0 \rightarrow -x_0/2$, $x_l = -\infty$, $x_r = \infty$: the shock can be solved analytically: $t = 1 + 3x$, ($\mathcal{S}_c(t) = \frac{1}{3}(t-1)$). The C_0, PC_1 solution constructed by our method reads:

$$\begin{array}{ll} v(x,t) = v_0(x+t-1) & \text{for } x \leq \frac{t-1}{3} \\ v(x,t) = v_0(x-t+1) & \text{for } x \geq \frac{t-1}{3} \end{array} \tag{8.39}$$

In this case, the solution (8.39) is the only continuous solution that can be constructed using the method of characteristics, to a set of zero measure. Note though that unless proved, nothing in this case guarantees that this solution is the viscosity solution of the HJE PDE (8.28). Several steps could be undertaken to achieve such a proof, that go beyond the work of this dissertation. (*i*) Prove it directly with test functions, using Evans' definition [70, Chap. 10, p. 542]. (*ii*) Prove it using the equivalence with the Frankowska solutions and epiderivatives [74, 8]. (*iii*) Prove it using a minimax criterion, such as the criterion proposed by Clarke et al. [57, Chap. 7, p. 225]. None of these definitions is in fact appealing, and checking that an analytical solution is the viscosity solution of a particular HJE is in general a difficult process. Simple examples are available in [14].

8.3.3 Single C_1 void construction procedure

When v_0 is C_1 and characteristics separate, we are able to construct a C_1 solution, which is then the classical (therefore viscosity and unique) solution of HJE.

Plateau construction: We now assume that v_0 is C_1, is increasing in $x \in]-\infty, x_c]$ and decreasing in $[x_c, \infty[$. (i.e. $\sigma = -1$ for $x \in]-\infty, x_c]$ and $\sigma = +1$ for $x \in [x_c, \infty[$). A void appears instantaneously at x_c: call $x_\sigma(t) = \Phi_\sigma^{-1}(\Phi_\sigma(x_c) + (t - t_0))$ the respective characteristics emanating from x_c. Then, $x_-(t) \leq x_+(t)$ in a neighborhood of (x_c, t_0): a void appears (the characteristics separate). v_0 is C_1, so it follows that $v_0'(x_c) = 0$.

Plateau construction for a single void:

Input: Initial conditions of the HJE PDE (8.28).
Output: C_0, PC_1 solution of (8.28) for the single void case.

```
1  $v(x,t) = v_0(\Phi_-^{-1}(\Phi_-(x) - (t - t_0)))$ if $x \leq x_-(t)$
2  $v(x,t) = v_0(x_c)$ if $x_-(t) \leq x \leq x_+(t)$
3  $v(x,t) = v_0(\Phi_+^{-1}(\Phi_+(x) - (t - t_0)))$ if $x_+(t) \leq x$
```

This solution solves (8.28) almost everywhere and is C_1 by assumption in the interior of the three domains above. The continuity of the derivatives on the boundary of the void (characteristics emanating from x_c) is given by:

$$(D_x v(x,t), D_t v(x,t)) = v_0'(x_c)\phi_\sigma(x_c)\left(-1, \frac{1}{\phi_\sigma(x)}\right) = (0,0)$$

(this equality is only true on the boundary of the void). The solution is thus C_1 on $\mathbb{R} \times \mathbb{R}^+$. This procedure will also work when v_0 is C_0, PC_1, but will then only provide a C_0, PC_1 function (see example below).

Example (C_0,PC_1 solution): We use our procedure to solve the following HJE, inspired by an example of Clarke et al. [57]:

$$D_t v(x,t) + \min_{b \in [-1,1]} b D_x v(x,t) = 0 \tag{8.40}$$

where $b \in [-1, 1]$ and $t \in]-\infty, 1]$ (time is backwards). The initial condition is $v_0(x) = |x|$. The solution constructed by our method is:

$$v(x,t) = v_0(\max(0, |x| + t - 1)) = |\max(0, |x| + t - 1)| = \max(0, |x| + t - 1)$$

Note that $v(x,t) = |x| + t - 1 \ \forall (t,x) \in]-\infty, 1] \times \mathbb{R}$, obtained by filling the whole space with characteristics, solves (8.28) almost everywhere, but is not the viscosity solution. It is straightforward to show that it is not the viscosity solution.

If instead, we had chosen $v_0(x) = x^2/(1 + x^2) \in C_1(\mathbb{R}, \mathbb{R})$, the corresponding solution constructed by our method would be

$$\begin{aligned} v(x,t) &= v_0(x + t - 1) && \text{for } x \le (t-1) \\ v(x,t) &= v_0(x - t + 1) && \text{for } x \ge -(t-1) \\ v(x,t) &= 0 && \text{for } (t-1) \le x \le -(t-1) \end{aligned}$$

Note that $v(x,t)$ is C_1 and the classical (therefore viscosity) solution.

8.3.4 General initial data

If $v_0 \in C_0(\mathbb{R}, \mathbb{R})$ is arbitrary, we can generalize the previous results and construct a C_0, PC_1 solution:

Construction procedure for general initial data

Input: Initial conditions of the HJE PDE (8.28).
Output: C_0, PC_1 solution of (8.28) for the general case.

```
1  Compute locations $x_c^i$ of extrema of $v_0(x)$;
2  Identify the $\bar{x}_c^i$ corresponding to shocks: $\phi_+$ for $x \le \bar{x}_c^i$, $\phi_-$ for $x \ge \bar{x}_c^i$
   and the $\underline{x}_c^i$ corresponding to voids: $\phi_-$ for $x \le \underline{x}_c^i$, $\phi_+$ for $x \ge \underline{x}_c^i$.
   We get: ... $\le \bar{x}_c^{i-1} \le \underline{x}_c^i \le \bar{x}_c^{i+1} \le \underline{x}_c^{i+2} \le$ ...;
3  Generate the matchings $m_i$ of the $\bar{x}_c^i$;
4  Construct the shocks, fill the $(x,t)$ space with characteristics, and void space.
5  $v(x,t) = v_0(\Phi_\sigma^{-1}(\Phi_\sigma(x) - (t - t_0)))$ in the domains filled with characteristics,
   $v(x,t) = v_0(\underline{x}_c^i)$ in the voids.
```

Example (general initial data): We now apply our technique to solve a problem with initial data generating multiple shocks and voids. Consider the following dynamical system $\dot{x} = (-|x+1| + |x-1|)b$ (all quantities in $\mathbb{R}$), for which we solve the following HJE:

$$D_t v(x,t) + \min_{b \in [-1,1]} (|1-x| - |1+x|) b D_x v(x,t) = 0 \tag{8.41}$$

with $t \in \mathbb{R}_-$ (backwards time), and the terminal condition $v(x,0) = v_0(x)$ given by:

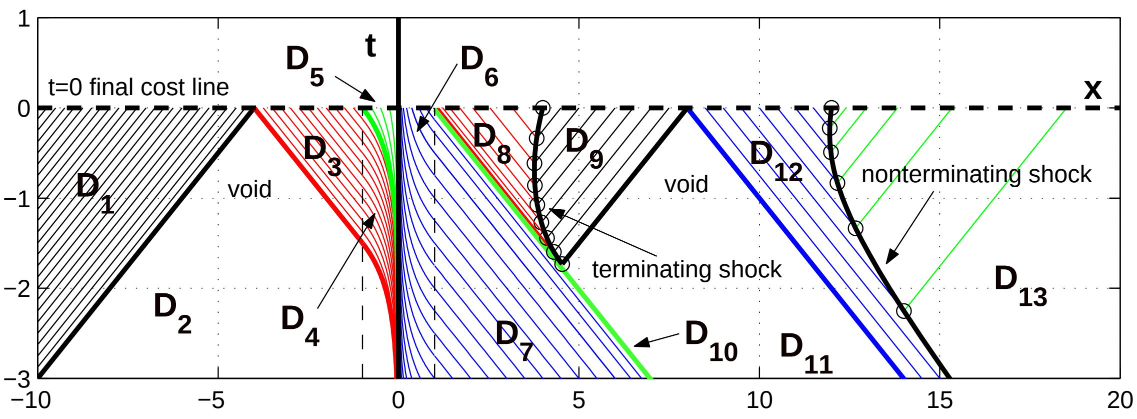


Figure 8.3: Characteristics pattern for (8.41), domains given by Table 8.1. Two voids appear at $x = -4$ and $x = 8$, two shocks appear at $x = 4$ and $x = 12$.

$$v_0(x) = \begin{cases} \frac{x}{1+\left(\frac{x}{4}\right)^2} & \forall x \in]-\infty, 4] \\ 1 + \left(\frac{x-8}{4}\right)^2 \left(2 - \left(\frac{x-8}{4}\right)^2\right) & \forall x \in [4, 12] \\ 1 + \frac{2(x-11)}{1+(x-11)^2} & \forall x \in [12, \infty] \end{cases} \tag{8.42}$$

Two shocks appear in the (x,t) plane for $t = 0$ at local maxima of $v_0(t,x)$ ($x = 4$ and 12). Two voids appear for $t = 0$ at local minima ($x = -4$ and 8), see Figure 8.3.

The equations of the shocks can be obtained in parametric form with appropriate matchings:

$$m_1(s) = \frac{4}{\phi(s)}[2 - \sqrt{4 - \phi(s)^2}]$$
$$m_2(s) = 11 + \frac{1}{\psi(s)}[1 + \sqrt{1 - \psi(s)^2}]$$

where

$$\phi(s) = 1 + (\tfrac{s-8}{4})^2(2 - (\tfrac{s-8}{4})^2) \quad \forall s \in [4,8]$$
$$\psi(s) = (\tfrac{s-8}{4})^2(2 - (\tfrac{s-8}{4})^2) \quad \forall s \in [8,12]$$

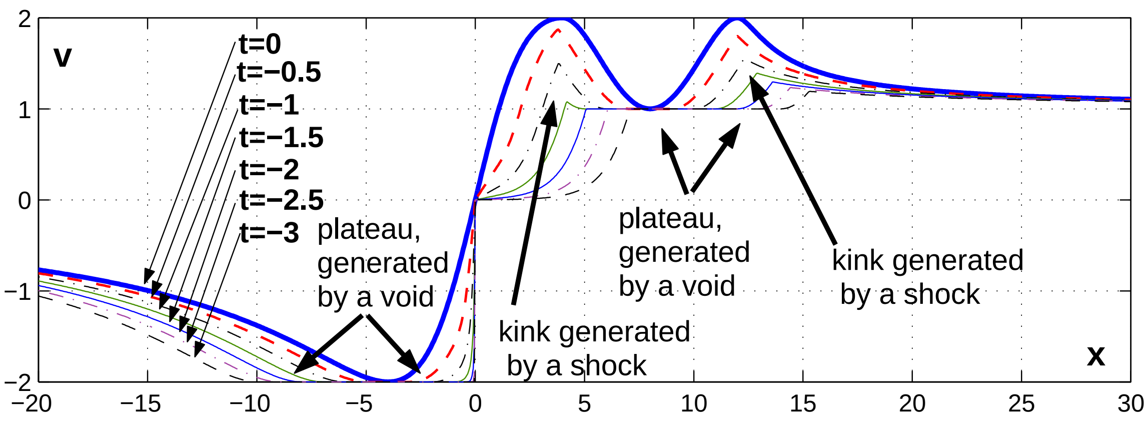


Figure 8.4: Analytical solution $v(x,t)$ for equation (8.41) for various times. The effect shocks (edges) and voids (plateaux) are clearly visible.

The equations of the two shocks are given by solving the matching problems for m_1 and m_2:

$$\begin{aligned} (t(s), x(s)) &= (\tfrac{1}{4}(m_1(s) - s), \tfrac{1}{2}(m_1(s) + s)) \\ (t(s), x(s)) &= (\tfrac{1}{4}(s - m_2(s)), \tfrac{1}{2}(m_2(s) + s)) \end{aligned} \tag{8.43}$$

For each of them we compute the shock $t \to \mathcal{S}_c(t)$, given by the solution of (8.43).

We now define the different subdomains D_i of $\mathbb{R}\times]-\infty, 0]$ which are displayed in Figure 8.3, mathematically described in Table 8.1. The corresponding solution is depicted in Figure 8.4, and summarized in Table 8.2.

In table 8.1, $x = X_1(t)$ and $x = X_2(t)$ are the explicit solutions of (8.43). The analytical solution obtained with this method is summarized in Table 8.2.

8.3.5 Application to two dimensional examples

The technique developed in the previous sections is extremely difficult to generalize to dimensions greater than three, because of the geometrical difficulties involved in constructing the matching surfaces (shocks), or voids in a three dimensional space. This method however was successfully applied to a few two dimensional examples (see for example [26]): the "tax pollution example" (from Saint-Pierre [139]), the "one chance collision game" (from Isaacs [85]) and the "proportional navigation problem" (from Leitmann [81, 98]). We now show

D_1	$t \geq \frac{1}{2}(x+4)$
D_2	$\{t \leq \frac{1}{2}(x+4) \ \wedge \ t \leq -\frac{1}{2}(x+4)\} \wedge$ $x \leq -1\} \cup \{t \leq \frac{1}{2}\log(-x) - \frac{3}{2} \ \wedge -1 \leq x \leq 0\}$
D_3	$t \geq -\frac{1}{2}(x+4) \ \wedge \ x \leq -1$
D_4	$t \geq \frac{\log(-x)}{2} - \frac{3}{2} \ \wedge \ t \leq \frac{\log(-x)}{2} \ \wedge -1 \leq x \leq 0$
D_5	$t \geq \frac{1}{2}\log(-x) \ \wedge \ x \leq -1$
D_6	$0 \leq x \leq 1$
D_7	$x \geq 1 \ \wedge \ t \leq -\frac{1}{2}(x-1)$
D_8	$t \geq -\frac{1}{2}(x - 4(2-\sqrt{3})) \ \wedge \ x \leq X_1(t)$
D_9	$t \geq \frac{1}{2}(x-8) \ \wedge \ t \geq -\frac{1}{2}(x-8) \ \wedge \ x(t) \geq X_1(t)$
D_{10}	$-\frac{x-1}{2} \leq t \leq -\frac{1}{2}(x-4(2-\sqrt{3}))$
D_{11}	$t \leq \frac{x-8}{2} \ \wedge \ t \leq -\frac{x-8}{2} \ \wedge \ t \geq -\frac{x-8+4\sqrt{3}}{2}$
D_{12}	$x \leq X_2(t) \ \wedge \ t \geq -\frac{x-8}{2}$
D_{13}	$x \geq X_2(t)$

Table 8.1: Definition of the different domains of validity of the analytical solution obtained by the method of characteristics. The different domains are shown in Figure 8.3, as well as the characteristics.

D_i	Eqn. of characteristics	$v(x,t)$
D_1	$\frac{1}{2}(x-x_0)$	$v_0(x-2t)$
D_2	void	$v_0(-4)$
D_3	$-\frac{1}{2}(x-x_0)$	$v_0(x+2t)$
D_4	$\frac{\log(-x)}{2} + \frac{x_0+1}{2}$	$v_0(2t - \log(-x) - 1)$
D_5	$\frac{1}{2}\log(\frac{x}{x_0})$	$v_0(xe^{-2t})$
D_6	$-\frac{1}{2}\log(\frac{x}{x_0})$	$v_0(xe^{2t})$
D_7	$-\frac{x-1}{2} + \frac{\log x_0}{2}$	$v_0(e^{2t+x-1})$
D_8	$-\frac{x-x_0}{2}$	$v_0(2t+x)$
D_9	$\frac{x-x_0}{2}$	$v_0(x-2t)$
D_{10}	$-\frac{x-x_0}{2}$	$v_0(2t+x)$
D_{11}	void	$v_0(8)$
D_{12}	$-\frac{x-x_0}{2}$	$v_0(x+2t)$
D_{13}	$\frac{x-x_0}{2}$	$v_0(x-2t)$

Table 8.2: Analytical solution to the Mayer Problem (8.28) obtained with the method developed above. The name of the different domains refer to Table 8.1, and is depicted in Figure 8.4. The characteristics are shown in Figure 8.3.

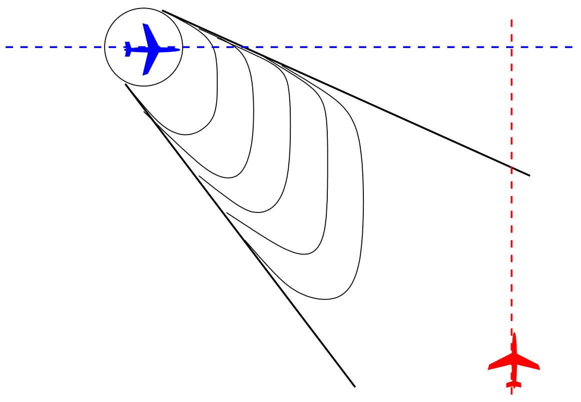


Figure 8.5: Sketch of the reachable set for the 2D aircraft problem, in the case $\psi_r = \pi/2$, corresponding to Figure 8.1.

an application to the "2D aircraft problem" (from Tomlin [147], which is a specific case of the "3D aircraft problem shown in the next chapter). We consider the following dynamics:

$$\begin{cases} \dot{x}_r = -a + b\cos\psi_r \\ \dot{y}_r = b\sin\psi_r \end{cases} \tag{8.44}$$

where $x = [x_r, y_r]$, $a \in \mathcal{A} = [2,4]$ and $b \in \mathcal{B} = [1,5]$. This system represents the relative dynamics of two aircraft respectively cruising at speeds a and b, with a relative coordinates x_r and y_r, and relative heading ψ_r, which is for now fixed.[†] The previous method can be used to solve analytically the following HJE associated to the Mayer problem:

$$\begin{aligned} D_t v(x,t) + H(x, D_x v(x,t)) = 0, \qquad & \text{for } t \in [-T,0], x \in \mathbb{R}^n; \\ v(x,0) = v_0(x), \quad & \text{for } x \in \mathbb{R}^n; \end{aligned} \tag{8.45}$$

where

$$H(x,p) = \max_{a\in[2,4]} \min_{b\in[1,5]} p^T f(x,a,b). \tag{8.46}$$

with $v_0(x) = v_0((x_r, y_r)) = x_r^2 - y_r^2 - 25$. This choice of v_0, introduced by Tomlin [147] corresponds to a safety disk of radius 5 nautical miles around one aircraft. Applying the formulation (8.45) to the dynamics (8.44) thus solves the following payoff problem: "compute the set of relative positions of two aircraft from which a disturbance aircraft (aircraft

[†]In the next chapter, ψ_r will be allowed to vary. An illustration of the relative dynamics is given in the next Chapter in Figure 9.1.

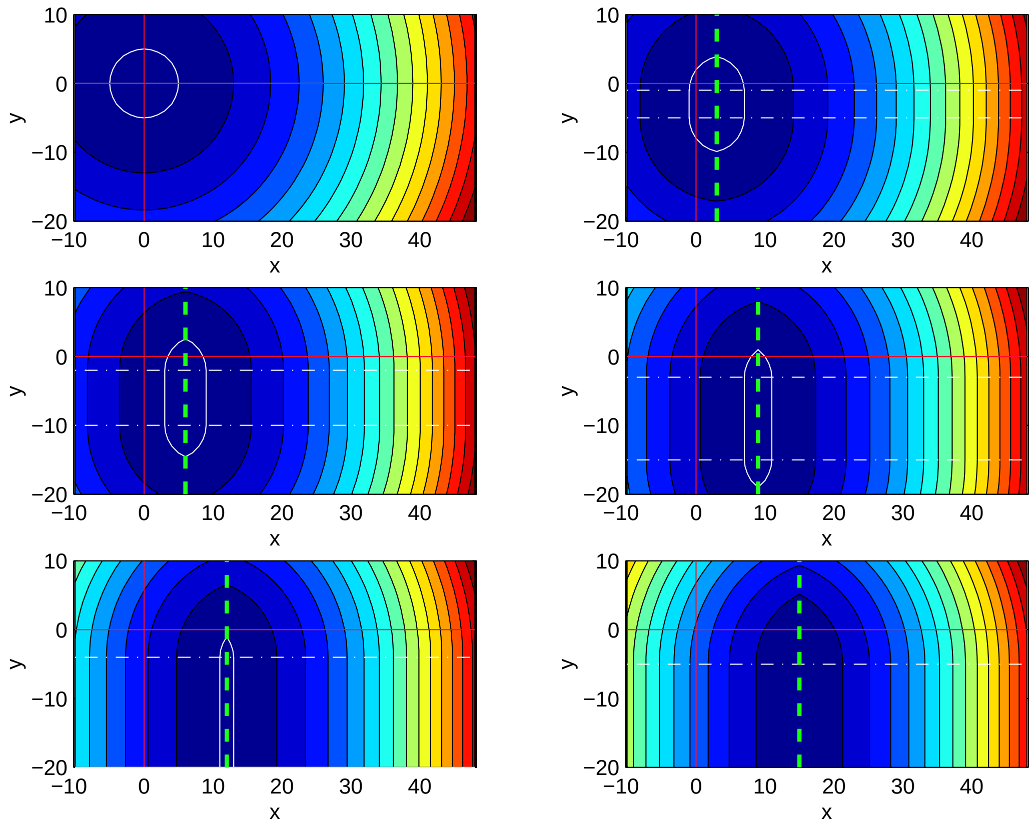


Figure 8.6: Analytical solution to the equation (8.45). At $t = 0$, $v(x_r, y_r, 0) = v_0(x_r^2, y_r^2) = x_r^2 + y_r^2 - 25$: the zero sublevel sets correspond to the safety disk in which aircraft b shall not penetrate. The solution is shown for $t = 0,\ -1,\ -2,\ -3,\ -4,\ -5$. As can be seen for $t = -5$, the subzero level sets have disappeared completely.

b) can end up in a safety disk of 5 nautical miles around the other aircraft (aircraft a), no matter what a does".

The method of the previous sections can be applied to this case. We show the results for $\psi_r = \frac{\pi}{2}$ in Figure 8.6. We see that a shock and a void appear. The shock propagates along the x_r-axis at speed 3: the equation of the shock (which is a line here) is: $x_r = -3(t - t_0)$ in the (x_r, y_r) plane. It originates at the $x_r = 0$ axis. There is a void propagating along the y_r-axis, whose bounds have the following motion: $y_r = t - t_0$ and $y_r = 5(t - t_0)$. The void originates from the $y_r = 0$ axis. The void and the shock interact. In fact, the shock goes through the void and prolongates the discontinuity of the gradient through it.

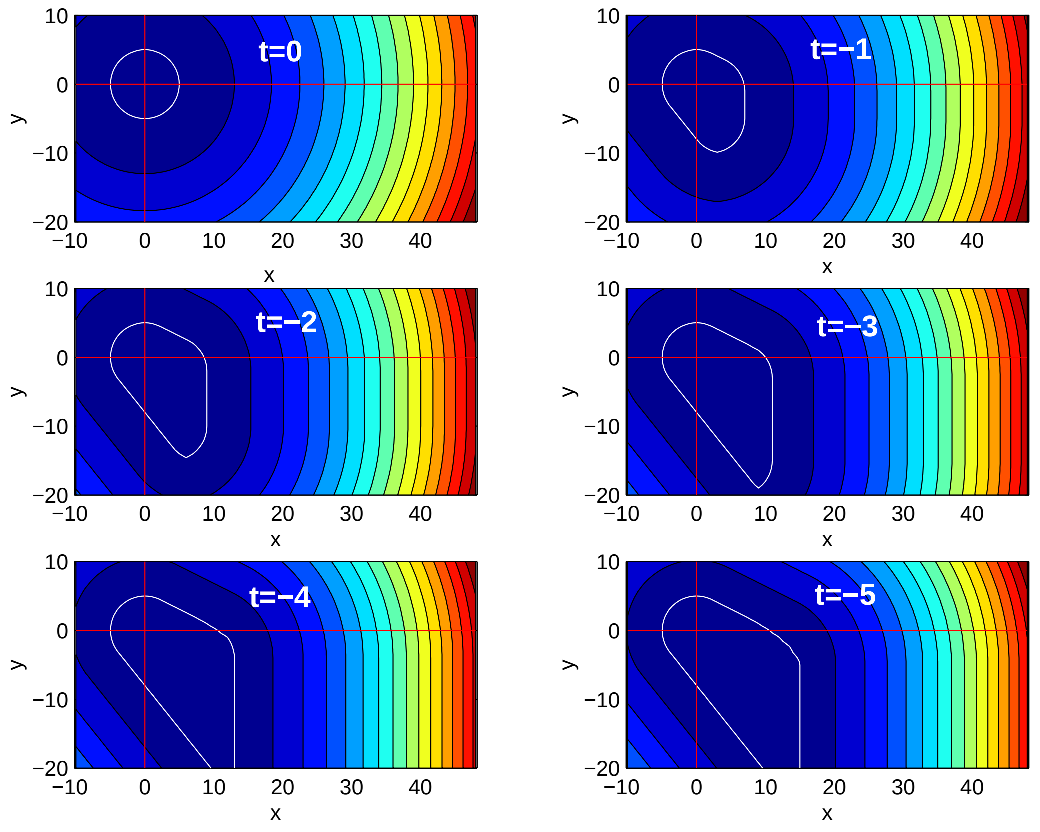


Figure 8.7: Minimum over time of the two dimensional solution of the HJE (8.45). As can be seen, the zero sublevel sets stop growing after $t = 5$.

Finally, we get the following solution for v:

$$\begin{array}{ll}
v(x_r, y_r, t) = v_0(x_r + 4(t - t_0), y_r - (t - t_0)) & \text{if } y_r \geq t - t_0 \wedge x_r \leq -3(t - t_0) \\
v(x_r, y_r, t) = v_0(x_r + 4(t - t_0), 0) & \text{if } y_r \in [5(t - t_0), t - t_0] \wedge x_r \leq -3(t - t_0) \\
v(x_r, y_r, t) = v_0(x_r + 4(t - t_0), y_r - 5(t - t_0)) & \text{if } y_r \leq 5(t - t_0) \wedge x_r \leq -3(t - t_0) \\
v(x_r, y_r, t) = v_0(x_r + 2(t - t_0), y_r - (t - t_0)) & \text{if } y_r \geq t - t_0 \wedge x_r \geq -3(t - t_0) \\
v(x_r, y_r, t) = v_0(x_r + 2(t - t_0), 0) & \text{if } y_r \in [5(t - t_0), t - t_0] \wedge x_r \geq -3(t - t_0) \\
v(x_r, y_r, t) = v_0(x_r + 2(t - t_0), y_r - 5(t - t_0)) & \text{if } y_r \leq 5(t - t_0) \wedge x_r \geq -3(t - t_0).
\end{array}$$

In [151], the authors introduced a new formulation of the reachability problem, consisting in taking the minimum over time of the solution of the Mayer problem. The result of this technique applied to the formula above is represented in Figure 8.7. Since the zero sublevel sets have disappeared from the solution for $t > 5$ (as can be seen in Figure 8.6), the zero

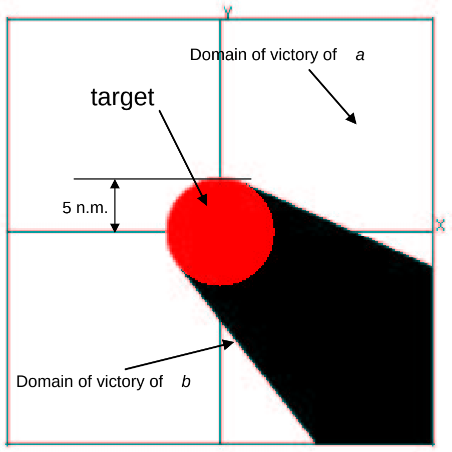


Figure 8.8: Viability solution to the 2D aircraft problem. In black: reachable set for (8.45), in the same situation as for Figures 8.6 and 8.7. Computation realized on a 1024 $\times$ 1024 grid (domain of computation is $[-20, 20] \times [-20, 20]$), using [139, 45].

sublevel sets obtained by keeping the minimum of the solution over time are frozen as well, as can be seen in Figure 8.7. This fact had been observed in numerical experiments in [151], and explained by the numerical dissipation introduced by the numerical techniques used to solve the problem. The result above, illustrated by Figures 8.6 and 8.7, shows that this feature is present in the analytical solution itself.

The correct solution to the problem above was subsequently computed and proved in [116]. This result was confirmed later using results from viability theory: Figure 8.8 shows the result of the computation for the same problem, using the techniques developed in [139, 45].

Chapter 9

Application of reachability analysis to Air Traffic Control

The previous chapter investigated the problem of proving safety of conflict avoidance protocols, in order to contribute to the automation of sector Air Traffic Controller tasks. It provided a mathematical framework which can be used to prove safety of a pair of aircraft from which the behavior of one aircraft is uncertain. The mathematical result was the construction of the *reachable set*. This set represents the set of relative positions outside of which the uncertain aircraft is unable to create a loss of separation (i.e. a situation in which the two aircraft are closer than the minimum distance allowed by FAA).

The previous chapter dealt with the mathematical aspects of this proof. This chapter will now apply this framework to real ATC scenarios. We will apply the same methodology for conflict detection, and run a set of experiments on ETMS data to show that this methodology can be applied successfully to real systems.

This chapter is organized as follows. Section 9.1 states the practical problem to be solved and links it with the framework of the previous chapter. Section 9.2 defines the reachable set in the context of collision avoidance and displays the mathematical problem which will be solved numerically. Section 9.3 shows the methodology used to apply the method to real data (ETMS data). The choice of the relevant numerical parameters is explained in Section 9.4 and 9.5. Finally, Section 9.6 presents the application of the method on real scenarios.

9.1 Problem statement

We will first cast the physical problem of conflict detection and resolution into the framework of the previous chapter (in particular Theorem 6). The action of one of the aircraft (aircraft b) is assumed to be uncertain but bounded within known bounds, and is treated as an opposing player to the control action of the first aircraft (aircraft a). From these assumptions, using the framework of differential game theory, two regions may be calculated and used to determine LOS threats[*] . These regions are: *(i)* the set of states from which aircraft b can cause a LOS of separation with aircraft a, regardless of the control action of aircraft a; *(ii)* the set of states from which aircraft a can find a safe escape maneuver to prevent conflict, for all possible actions of aircraft b. This scenario could model the situation in which one aircraft blunders, due to navigation error, human error, or communication loss, yet ATC maintains communication and control of the second aircraft. Variations of this framework could be used to treat other scenarios, such as the case in which communication is lost with both aircraft, yet still only one blunders, as well as the true worst case, in which communication is lost and the aircraft appear to coordinate to cause a collision. This third case is extreme, yet it could be used to assess the true worst case: a tragic recent example is the midair crash in July 2002 above Überlingen, caused by communication of erroneous maneuvers.

In this chapter, our goal is to evaluate this tool as a possible online ATC advisory for assessing the LOS *alert level* in high altitude traffic. By alert level, we mean a metric which can be used practically to indicate when ATC should modify the aircraft's trajectory. As a testbed for this work, we use a set of ETMS data[†] for high altitude traffic in several sectors of the Oakland Center airspace. The ETMS data file used for this work is from several years ago. This work is to the best of our knowledge the first application of Isaacs' *Game of two identical vehicles* to ATC with real data (in the original work of Isaacs [85] and Merz [112], specific cases are investigated, with dummy numerical parameters).

Using the classification of Kuchar and Yang [96], the *state propagation* of our method is *worst case*, the *state dimension* is two (even if our system enables three dimensional conclusions, as will appear), the definition of *detection* is one of the points of this work, and the *conflict*

[*] *loss of separation* (LOS) has been defined in Chapter 2.

[†] *Enhanced Traffic Management System* data, see Chapter 3, Section 3.2.2 for a more precise definition.

resolution is *optimized*. We attempt to answer some of the questions raised by Kuchar [96]. In particular, we show that a worst case approach does not always provide false alarms and might be a very appropriate metric for short term conflicts, and we also discuss the issue of the time horizon.

Finally, using the ETMS data, we demonstrate a good agreement between the decisions advised by our tool, and the human ATC decision currently observed in the ARTCC (and observable in ETMS data through comparison of actual trajectory with original flight plan). Thus, we believe that the tool presented in this chapter could provide a useful advisory to ATC.

In the current ARTCC monitoring system, human Air Traffic Controllers have a visual display showing all airborne aircraft in their airspace (sector). Numerous information can be superimposed on the symbol corresponding to each aircraft, such as: heading vector, speed, altitude, filed flight plan, flight plan based anticipated positions of the aircraft in the current sector. Based on this information, the Air Traffic Controller makes a decision regarding which flight plans to alter to avoid potential LOS. The Air Traffic Controller decision is based on anticipation of the positions of each pair of potentially conflicting aircraft. The set of commands applied to the aircraft to achieve certain goals (spacing, conflict avoidance) has been well studied [83, 19, 17]. The method presented in this chapter aims at making this anticipation systematic and providing a guarantee on the relative separation of the two aircraft, through the generation of an automated tool to aid the decision-making of the Air Traffic Controller:

Problem 8. *Consider a pair of aircraft (a, b) in the en route airspace, each aircraft having known bounds on its control actions. Compute the set of relative positions and orientations of a and b from which for all possible control action of aircraft a (resp. b), there exists an uncertainty in the control input of aircraft b (resp a) rendering the pair susceptible to a LOS.*

This problem captures, for example, the situation in which one aircraft blunders, and ATC maintains communication with the second aircraft. In addition, two other problems are investigated, which are relevant for situations with communication loss. First, given an aircraft a following its flight plan and an aircraft b, what is the set of relative configurations of a and b from which there exists an uncertainty in the input of b which could cause a LOS? Second, given two aircraft a and b, what is the set of relative positions of a and b from which the two aircraft can collaboratively (unwillingly) create a LOS?

9.2 Reachability of the collision avoidance problem

The collision avoidance scenario used for each pair of aircraft is modeled using the *Game Of Two Identical Vehicles*, introduced in Isaacs [85]. This differential game was solved analytically by Merz for a particular choice of numerical parameters [112, 114] and has been a focus of research since [129]. The results of the previous chapter and [116] provides a mathematical method to solve the problem (and other differential games) numerically in the general case, based on the initial formulation of Tomlin et al. [149]. Our work uses this formulation, extended to the case of non-identical vehicles, having different capabilities, which we now summarize.

The two aircraft system is modeled with a commonly used, very simple kinematic system. The state of each vehicle is represented by a location in the $x-y$ plane and a heading ψ relative to the x-axis. The evolution of these states is governed by the vehicle's forward velocity v and rotational velocity ω. For example, for vehicle a, indexing v and ω by a,

$$\frac{d}{dt}\begin{bmatrix} x_a \\ y_a \\ \psi_a \end{bmatrix} = \begin{bmatrix} v_a \cos\psi_a \\ v_a \sin\psi_a \\ \omega_a \end{bmatrix} \tag{9.1}$$

and similarly for vehicle b. The linear velocity v of each aircraft is assumed fixed. The angular velocity (turn rate) ω is allowed to vary, and is considered here the input of the aircraft. Even if in practice the aircraft could also change their velocities, most of the conflicts of the type investigated here are solved horizontally by modifying the heading of one of the vehicles, which makes the assumption of constant velocity valid. Conflict resolutions involving speed changes are also observed in practice (mostly in converging traffic [17]), but for these cases, conflict resolution is very tightly linked with scheduling [27] which is a fundamentally different problem and uses other mathematical resolution techniques. The two vehicles have the same kinematics (9.1) with different values of v and ranges of ω. Using differential game terminology [85], we denote the evader by the

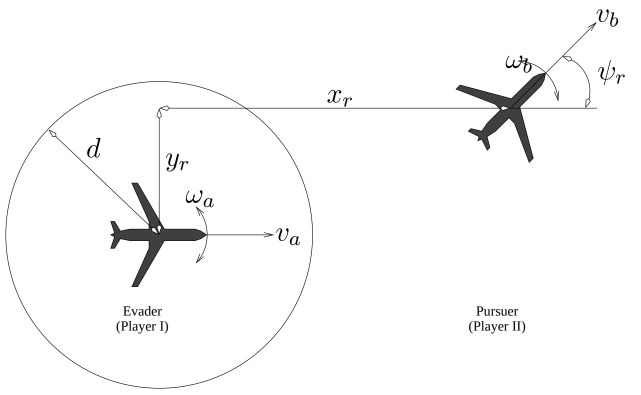


Figure 9.1: Relative coordinate system. Origin is located at the center of the evader. Notations are the same as in Section 8.1 and 8.3

subscript a and the pursuer by the subscript b. The turn rate of each aircraft is limited by the performance of the aircraft. We denote $\mathcal{A} = [\underline{\omega}_a, \overline{\omega}_a]$ the range of possible turn rates of aircraft a and similarly $\mathcal{B} = [\underline{\omega}_b, \overline{\omega}_b]$ for aircraft b. The values of $\underline{\omega}_a$, $\overline{\omega}_a$, $\underline{\omega}_b$ and $\overline{\omega}_b$ will be derived later.

We say that a LOS has occurred if the two vehicles come within distance d of one another. Our goal is to determine the set of states from which the pursuer can cause a LOS to occur. Translating into reachability terms, we define $\mathcal{G}_0$ as the set of all states at which the two vehicles are within d units of one another. Denote by $\mathcal{G}(\tau)$ the set in which the pursuer can cause a LOS in the next τ time units despite the best efforts of the evader. As was explained before, this set is variously referred to as the *reachable set* [116], *victory domain* [85], or *discriminating kernel* [46]. We can simplify the two-aircraft system to three dimensions by working in relative coordinates. We will denote the state vector as $x \in \mathbb{R}^3$.[‡] We fix the evader at the origin, facing along the positive x_r axis (see Figure 9.1 and Table 9.1). Then, the pursuer's relative location and heading are described by the following dynamical system:

$$\dot{x} = \begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\psi}_r \end{bmatrix} = \begin{bmatrix} v_b \cos\psi_r - v_a + \omega_a y_r \\ v_b \sin\psi_r - \omega_a x_r \\ \omega_b - \omega_a \end{bmatrix} = f(x, \omega_a, \omega_b) \tag{9.2}$$

[‡]$x = [x_r, y_r, \psi_r] \in \mathbb{R}^3$ shall not be confused with x_r, the relative horizontal coordinate of the two vehicles, shown in Figure 9.1. x is the state of the system, in agreement with notations used in Sections 8.1 and 8.3.

var.	meaning
x_r	rel. position in direction of evader's flight
y_r	rel. position perp. to direction of evader
ψ_r	rel. heading ($0 \leq \psi_r < 2\pi$)
x	state vector ($x = [x_r, y_r, \psi_r]^T$)
ω_a	angular vel. and input of evader ($\omega_a \in \mathcal{A}$)
ω_b	angular vel. and input of pursuer ($\omega_b \in \mathcal{B}$)
v_a	speed of evader
v_b	speed of pursuer
d	minimum safe separation distance ($d = 5$nm)
$\mathcal{G}_0$	LOS set $\{(x_r, y_r, \psi_r)\|x_r^2 + y_r^2 \leq d^2$

Table 9.1: Variables for the two vehicles game.

where v_a and ω_a are the velocity and turn rate of the evader, v_b and ω_b, of the pursuer. The meaning of the variables above is explained in Table 9.1.

Since a collision can occur at any relative heading, the target set $\mathcal{G}_0$ depends only on x_r and y_r and includes any state within distance d of the planar origin:

$$\mathcal{G}_0 = \{x \in \mathbb{R}^3 | x_r^2 + y_r^2 \leq d^2\} \tag{9.3}$$

which can be converted into a signed distance function

$$v_0(x) = \sqrt{x_r^2 + y_r^2} - d \tag{9.4}$$

The set $\mathcal{G}_0$ is given by $\mathcal{G}_0 = \{x \in \mathbb{R}^3 | v_0(x) \leq 0\}$. We proved in the previous chapter that the set $\mathcal{G}(\tau)$ is given by $\mathcal{G}(\tau) = \{x \in \mathbb{R}^3 | v(x, -\tau) \leq 0\}$, where $v(\cdot, \cdot)$ is the viscosity solution [59, 58] of the following modified Hamilton-Jacobi-Isaacs partial differential equation (HJI PDE):

$$D_t v(x,t) + \min[0, H(x, D_x v(x,t))] = 0 \tag{9.5}$$

for $t \in \mathbb{R}^-$, with terminal conditions

$$v(x, 0) = v_0(x) \tag{9.6}$$

In (9.5), $D_x v$ represents the gradient of v, and H is the Hamiltonian of the system. For the problem defined at the beginning of this section, in which one aircraft blunders, H is defined as:

$$H(x,p) = \max_{\omega_a \in \mathcal{A}} \min_{\omega_b \in \mathcal{B}} h(x,p,\omega_a,\omega_b) \tag{9.7}$$

with h given by: $h(x,p,\omega_a,\omega_b) = p^T f(x,\omega_a,\omega_b)$:

$$\begin{aligned} h(x,p,\omega_a,\omega_b) = \;& -p_1 v_a + p_1 v_b \cos\psi_r + p_2 v_b \sin\psi_r \\ & +\omega_a(p_1 y_r - p_2 x_r - p_3) + \omega_b p_3 \end{aligned} \tag{9.8}$$

In the previous formula, the costate is $p := [p_1, p_2, p_3]$. The optimal input ω_a^* and worst disturbance ω_b^* achieving H in (9.7) can be easily computed from (9.8):

$$\begin{aligned} \omega_a^* &= (\underline{\omega}_a + \overline{\omega}_a)/2 + \operatorname{sgn}(p_1 y_r - p_2 x_r - p_3)(\overline{\omega}_a - \underline{\omega}_a)/2 \\ \omega_b^* &= (\underline{\omega}_b + \overline{\omega}_b)/2 - \operatorname{sgn}(p_3)(\overline{\omega}_b - \underline{\omega}_b)/2 \end{aligned}$$

Equation (9.5) has to be solved from time $t = 0$ backwards to time $t = -\tau \leq 0$. The

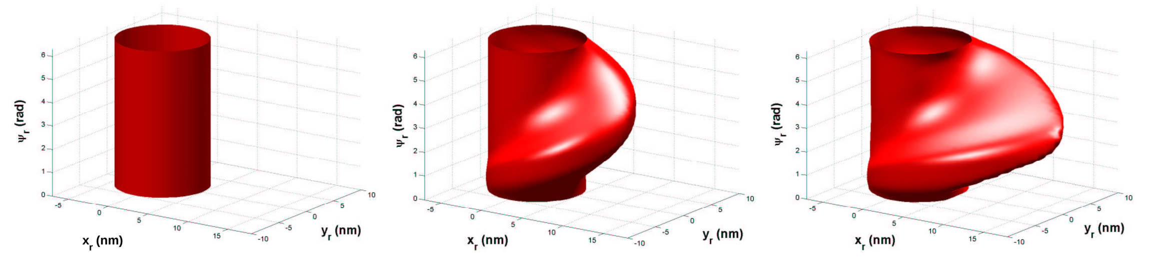


Figure 9.2: **Left:** Set $\mathcal{G}_0$ in the $x - y - \psi$ space. **Center:** Set $\mathcal{G}(\tau)$ for $\tau = 30$sec. **Right:** Set $\mathcal{G}(\tau)$ for $\tau = 120$sec. (the set has converged). For this case, both aircraft are cruising at 500kts. $\mathcal{A} = \mathcal{B} = [-1.3, 1.3]$ $^\circ\text{sec}^{-1}$.

set $\mathcal{G}(\tau)$ is shown in Figure 9.2 center and right for $\tau = 30$sec. and $\tau = 120$sec. The sets $\mathcal{G}(\tau)$ can be interpreted the following way. If the relative positions of two aircraft are such that $(x_r, y_r, \psi_r) \in \mathcal{G}(\tau)$, then the pursuer can cause a LOS in the next τ time units. If $(x_r, y_r, \psi_r) \notin \mathcal{G}(\tau)$, then no matter what the pursuer does, the evader can always avoid a LOS. As can be seen in Figure 9.2 center and right, the growth of $\mathcal{G}(\tau)$ is not isotropic: Figure 9.3 shows slices of the set $\mathcal{G}(\tau)$ for various τ, obtained at $\psi_r = 90°$. The slices extend in the $(1, -1)$ direction (to the bottom right on the plot): the conflict is most difficult to avoid when the relative position of the two aircraft points in this direction (the two aircraft are heading towards the same point). The set $\mathcal{G}(\tau)$ converges (i.e. stops growing) after $\tau = 150$sec. This fact, which is proved analytically for $\mathcal{A} = \mathcal{B}$ [112, 114], will have practical significance to our implementation, in which we will demonstrate numerical convergence of the set even for non identical vehicles. A possible heuristic interpretation of this fact is that a can "always" escape by turning, even if b is faster. Similar observations have been reported by Saint-Pierre for an analogous acoustic capture problem [46, 32, 115]. Note that the shapes of the two dimensional slices of $\mathcal{G}(\tau)$ change as the heading changes.

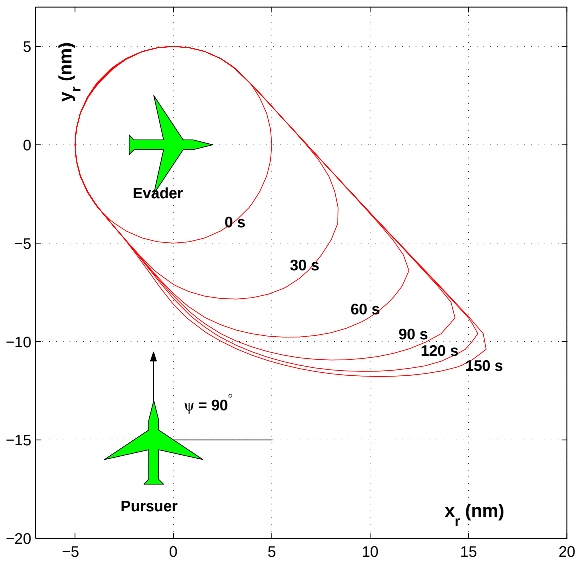


Figure 9.3: Slice of $\mathcal{G}(\tau)$ from Figure 9.2 right, for $\psi_r = 90°$, for $\tau = 30,\ 60,\ 90,\ 120,\ 150$ sec. The set extends in the direction $(1, -1)$: the conflict is most difficult to avoid when the pursuer is initially to the bottom right of the evader on this plot (the aircraft are heading towards the same point). Numerical convergence to a fixed point is observed for $\tau \geq 150$sec.: $\mathcal{G}(\tau)$ stops growing.

9.3 Extraction of parameters from ETMS data

In the following sections, we use the method summarized above. We perform reachability computations (i.e. computing the set $\mathcal{G}(\tau)$ from the set $\mathcal{G}_0$) for given choices of v_a, v_b, and ranges $\mathcal{A}$ and $\mathcal{B}$ of input turning rates ω_a and ω_b. We now explain how to choose these values from ETMS data (a description of ETMS data is given in Chapter 3, Section 3.2.2).

Let us consider any pair of aircraft described in this ETMS set, and arbitrarily assign one as the evader (aircraft a) and one as the pursuer (aircraft b). From ETMS data, we can extract the velocity v_a of the evader aircraft; from this, we determine the range $\mathcal{A}$ of available turning rates ω_a as follows. A first order approximation of the dynamics of aircraft

Alt. (100ft)	310	350
num. of measur.	2097	2515
Min. v (kts)	414	410
Max. v (kts)	514	480
Mean v	454	441
Std. dev.	16	12
Min. ω (deg·s^{-1})	2.12	2.27
Max ω (deg·s^{-1})	2.63	2.66
Mean ω (deg·s^{-1})	2.41	2.47
Std. dev.	0.079	0.067

Table 9.2: Speed and extremal turning rate distribution for two altitudes of traffic (31,000ft and 35,000ft), over a 24 hour period in the Oakland ARTCC. The number of measurements used for the computation of these statistics is reported in the first row. Each aircraft can lead to several measurements (if an aircraft appears more than once at the corresponding altitude).

a provides the following equations for lift L, weight $W = mg$, bank angle α, mass m and turning radius R_a:

$$L \cos \alpha = W \quad \text{and} \quad L \sin \alpha = m v_a^2 / R_a \tag{9.9}$$

from which we can compute R_a:

$$R_a = v_a^2 \ / \ g \tan \alpha \tag{9.10}$$

The previous result provides a lower bound on R_a, given by $\underline{R}_a = \frac{v_a^2}{g \tan \alpha_{\max}}$, where $\alpha_{\max}$ is the maximal bank angle, which for this study is $\alpha_{\max} = 45°$. The turning radius can be related to the maximal turning rate by: $\overline{\omega}_a = \frac{v_a}{\underline{R}_a}$, and symmetrically, $\underline{\omega}_a = -\frac{v_a}{\underline{R}_a}$, which provides $\mathcal{A} = \left[-\frac{v_a}{\underline{R}_a}, \frac{v_a}{\underline{R}_a}\right]$ and similarly for $\mathcal{B}$. Using this relationship, the parameters of the relative kinematics (9.2) of the two aircraft are completely defined. In order to solve the corresponding HJI PDE (9.5), we need to prescribe ϕ_0: $\mathcal{G}_0$ is chosen to be a cylinder of radius $d = 5$nm according to FAA regulations. In this chapter, we use one 24 hour subset of ETMS data in several sectors of the Oakland airspace.

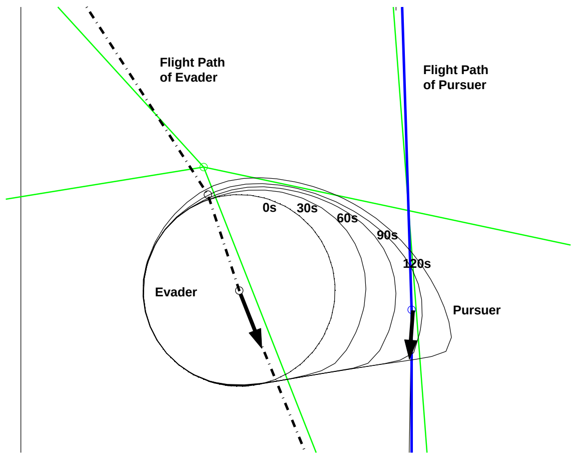


Figure 9.4: Reachable set for the situation in which aircraft a (dash dotted flight plan) and aircraft b (solid flight plan) have intersecting flight plans. $\mathcal{G}(\tau)$ is shown for $\tau = 30,\ 60,\ 90,\ 120$sec. Aircraft b is in $\mathcal{G}(\tau = 90)\backslash\mathcal{G}(\tau = 60)$, which means that aircraft b is a potential threat to aircraft a in the next 90sec, but aircraft a is safe for at least 60sec. The conflict would not have been detected if the time horizon of the computation had been 60sec. For the rest of the computations, we will take a time horizon of 150sec., in order to take into account points at the extremity of the reachable set.

9.4 Choice of the time horizon, computational time

We need to determine the time horizon τ for the computation of $\mathcal{G}(\tau)$. This choice is determined by the time scale of the physical problem of interest: a conflict avoidance maneuver is on the order of several minutes; the anticipation ability of a human ATC is also on the order of several minutes. Therefore, we choose a time horizon of $\tau = 3\,\text{min}$. The distance flown at 500kts. in 3min. is 25nm., which enables one to capture potential threats with aircraft 50nm from each other. Even if 50nm. might seem excessive, it is necessary to choose at least 3min., as is illustrated by the growth of the reachable set in Figure 9.3. One can see in this figure that the shape of $\mathcal{G}(\tau)$ changes significantly in the interval $[0, 3\,\text{min}]$.

In practice, numerical convergence was observed for $\tau \leq 3\,\text{min}$ for all the examples treated here (see for example Figures 9.3 and 9.4). This confirms the relevance of our choice: numerical convergence for $\tau \leq 3\,\text{min}$ implies that the duration of a conflict avoidance maneuver (obtained when a chooses ω_a^* to be its input during the maneuver) is of that order.

The computation of $\mathcal{G}(\tau)$ cannot yet be performed online. It takes approximately 5 minutes to compute $\mathcal{G}(3\,\text{min})$ (on a standard laptop) for a case similar to Figure 9.2. It is therefore

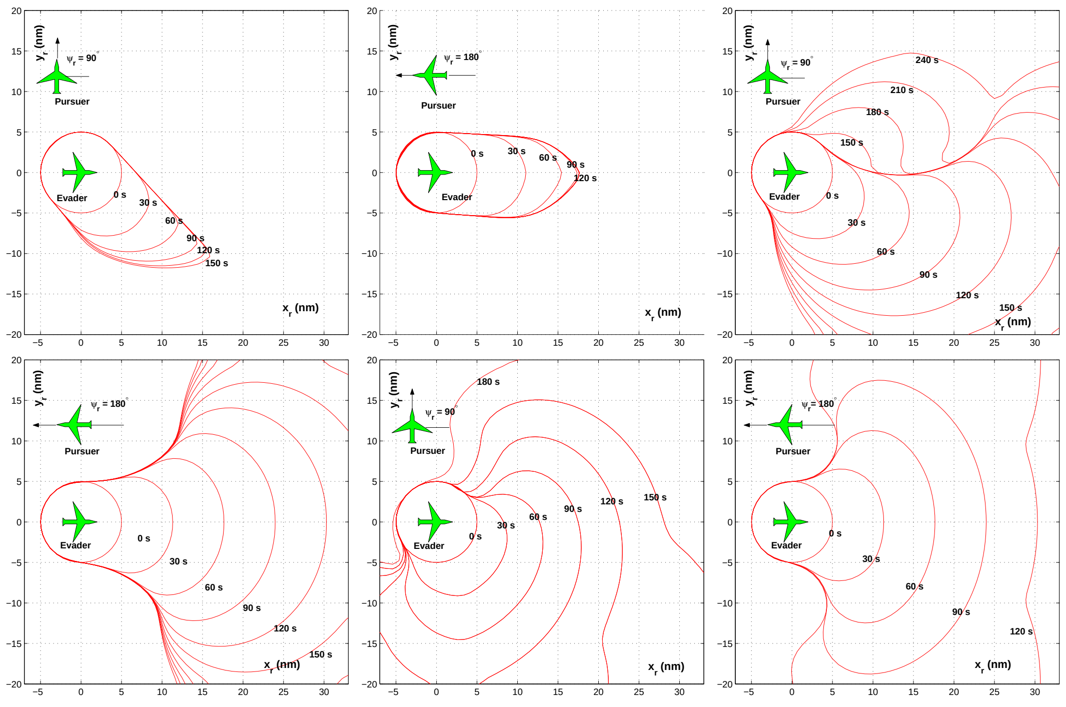


Figure 9.5: Growth of the reachable set $\mathcal{G}(\tau)$ for various values of τ, for a given initial relative heading $\psi_r = 90^\circ$ (left), and $\psi_r = 180^\circ$ (right) of the two aircraft. **First column:** DG setting. $\mathcal{G}(\tau)$ is bounded, and extends in the $(1, -1)$ – resp. $(1, 0)$ direction. For points in the $(1, -1)$ – resp. $(1, 0)$ direction outside the reachable set, the evader can avoid the LOS by going around the pursuer on both sides. **Second column:** BB setting. For the $\psi = 90^\circ$ case, $\mathcal{G}(\tau)$ extends first in the $(1, -1)$ direction: the pursuer can cause a *direct* collision by heading towards the evader if it is below and to the right of the evader on this plot. For $t = 150s$, $\mathcal{G}(\tau)$ grows for $y_r > 0$. The corresponding (x_r, y_r) are positions for which the pursuer has to turn right by 90° and intercept the evader (*indirect collision*). For the $\psi = 180^\circ$, the situation is entirely symmetrical. **Third column:** CC setting. Both aircraft collaborate to a LOS, therefore, $\mathcal{G}(\tau)$ grows much faster, and for the same values of τ, all initial positions in $\mathcal{G}(\tau)$ correspond to a direct collision (both aircraft are heading at each other). For the three cases above, the velocity of the aircraft is 500kts (and therefore, $\mathcal{A} = \mathcal{B} = [-1.3, 1.3]\ ^\circ\text{sec}^{-1}$).

impossible for an online implementation of our method to perform this computation in real time for a given pair of $\mathcal{A}$ and $\mathcal{B}$. However, it is possible to create a library of $\mathcal{G}(3\,\mathrm{min})$ for different $\mathcal{A}$ and $\mathcal{B}$. Since this method is a horizontal conflict detection method — and therefore applies to aircraft within 2000 ft of each other, we can use the fact that the range of v_a, v_b is relatively small (and therefore the same is true for the ranges of possible $\mathcal{A}$ and $\mathcal{B}$). Table 9.2 justifies the fact that velocities at a given altitude have a relatively small range. The range in turning rates is accordingly small.

In computing the statistics of Table 9.2, we accounted for the fact that the speeds might change over time. Therefore, the average quantities in Table 9.2 are computed using one entry for each occurrence of an aircraft in the 24 hour ETMS data set which we use. For example, if an aircraft appears 12 times at altitude 35,000ft before it changes altitude, we will account for the 12 speed records. We used a database of $\mathcal{G}(3\,\mathrm{min})$ for our implementation and believe that a similar database could be used for an online implementation.

9.5 Choice of the strategy

The differential game formulation (9.5),(9.6) enables the investigation of multiple scenarios, which are all relevant for ATC. Figure 9.5 shows three possible applications of this methodology:

1. *Communication loss with a blunderer.* This situation is labeled DG (differential game). It is modeled by the differential game setting developed previously. Blunders can happen in ATC because of navigation or human errors. This scenario models the situation in which one aircraft blunders and ATC communicates with the other aircraft (and prescribes a corresponding conflict resolution maneuver). This setting thus encompasses the worst case blunderer, heading directly towards the other aircraft.

2. *Communication loss with both aircraft in presence of a blunderer and a "blind aircraft".* This situation is labeled BB (blunderer + blind). ATC loses communication with both aircraft. In general, when there is a communication loss, aircraft are supposed to follow their original paths until communication is reestablished. The present scenario models a situation in which one aircraft follows its original path while the other blunders (because of navigation or human error, for example). The worst case for this scenario is thus: the pursuer tries to cause LOS directly, while the evader blindly stays on its course. This is modeled by setting $\mathcal{A} = \{0\}$ (in other words, the evader has no input, i.e. no turning ability).

3. *Collaborative collision strategy.* This situation is labeled CC (collaborative collision). Because of misunderstanding in communication with ATC or failure to follow procedures, both aircraft might collaborate (unwillingly) to a LOS. The worst case scenario for collaborative collision is thus when both aircraft head directly at each other from their initial position. It can be modeled by replacing the Hamiltonian (9.7) by

$$H(x,p) = \min_{\omega_a \in \mathcal{A}} \min_{\omega_b \in \mathcal{B}} h(z, p, \omega_a, \omega_b)$$

In the previous formula, the two aircraft contribute to the LOS, as can be seen in the min-min instead of the max-min.

Conflict avoidance maneuvers typically fall into category 1 (the no blunder case is encompassed by the worst blunder case). However, a case such as the midair crash above Überlingen (Germany) between a DHL Boeing 757-200 (DHX 611) and a Bashkirian Airlines Tupolev 154 (BTC 2937) on July 1st, 2002, was caused by the simultaneous action of the two pilots. The DHX Boeing followed a command issued by TCAS (onboard), while the BTC Tupolev followed the (erroneous) orders of ATC, leading to a collaborative collision rather than conflict avoidance. It therefore could fall in categories 2 or 3, depending on how the communication records of the accident are interpreted.

The three scenarios are of equal interest for ATC, and will obviously lead to different reachable sets. It is intuitive to see that the following inclusion holds: $\mathcal{G}_{\text{scenario 1}} \subseteq \mathcal{G}_{\text{scenario 2}} \subseteq \mathcal{G}_{\text{scenario 3}}$. Figure 9.5 illustrates the growth of $\mathcal{G}(\tau)$ with τ for the three scenarios, for two cases where the relative headings of the pursuer and the evader are respectively $\psi_r = 90°$ and 180°. The fast growth of $\mathcal{G}(\tau)$ for scenarios 2 and 3 can clearly be seen from these figures.

For the rest of the study, we will focus on scenario 1: it is assumed that ATC can communicate correctly with at least one aircraft. We use this scenario to create a metric for conflict detection.

Conflict detection alert method

```
Input: ETMS data feed.
Output: Potential LOS threats.

1   At time $t$, select all aircraft pairs with a relative vertical separation of within 2000ft.
    This can be done manually for specific aircraft or exhaustively (automatically)
2   For a given pair, extract $v_a$, $v_b$, compute $\mathcal{A}$, $\mathcal{B}$ using (9.9) and (9.10).
    This is done using the ETMS data extraction procedure of the previous section
3   Compute $\mathcal{G}(\tau)$ for a desired time horizon $\tau$.
3'  Relabel aircraft $a$ and $b$. Compute new $\mathcal{G}(\tau)$.
4   Compute $x_r$, $y_r$, $\psi_r$ from the ETMS data at $t$.
    For $\mathcal{G}(\tau)$ of 3 and 3', check if $(x_r, y_r, \psi_r) \in \mathcal{G}(\tau)$.
5   If $(x_r, y_r, \psi_r) \in \mathcal{G}(\tau)$ for either 3 or 3', a potential LOS is detected with $\tau$ time units.
6   Go back to 2 until all pairs have been tested.
7   Wait until next data update and return to 1.
```

9.6 Alert levels in ETMS data

9.6.1 Conflict detection methodology

We apply the previous method to high altitude traffic in ATC, using prerecorded ETMS data. Since our goal is to develop a technique which can serve as an advisory tool to ATC, the method is designed to work in real time with information provided at a given rate. For the present case, the update rate in the ETMS data is 3 min., but using precomputed sets, our method works as well with higher rates, such as 15 sec., which is on the order of the current monitoring display at a sector level in the ARTCC. Our methodology is as follows:

We now describe the steps of the method above.

1. The aircraft selection can be done manually or automatically. A manual selection would enable ATC to directly pick out an aircraft pair and test for a LOS threat. It is also very easy to automate the procedure, to do an exhaustive test of all aircraft pairs and display the threats only. Technically all pairs of aircraft should be selected iteratively. It is easy however to introduce heuristics based on separation and heading, in order to discard a large portion of them (for example if two aircraft are separated by more than 50 nm and are flying opposite directions). This enables one to reduce the computational time required by the method.

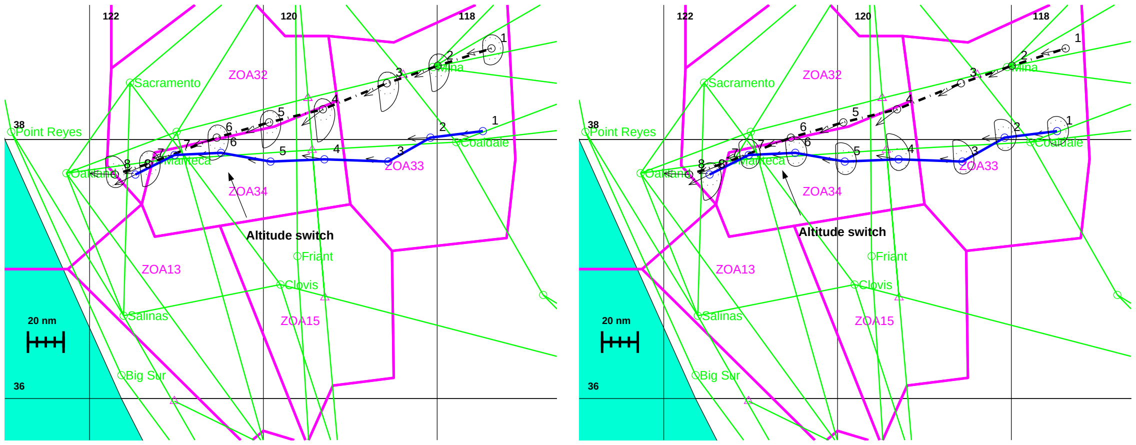


Figure 9.6: Aircraft 1 (solid), arriving from Dallas/Fort Worth Airport to Oakland (OAK); aircraft 2 (dash-dotted), arriving from Indianapolis Airport to OAK. Both aircraft are initially supposed to follow the Madwin 3 arrival (from Coaldale and Mina respectively), entering the Oakland ARTCC (positions labeled 1) at 35,000ft. At the position labeled 6, both aircraft are in $\mathcal{G}(\tau)$ w.r.t. the other aircraft, for $\tau = 3\,\text{min}$. An altitude change - while descending into OAK - is observed shortly after (aircraft 1 descends to 14,000ft, while aircraft 2 descends to 24,000ft). This action of ATC keeps the aircraft separated.

2. The velocities are directly read from the ETMS data. $\mathcal{A}$ and $\mathcal{B}$ are computed using the procedure of the previous section.

3. The computation of $\mathcal{G}(\tau)$ is performed by solving the HJE PDI, as described in the previous section. It is realized for an arbitrary choice of a and b, and accounts for a possible blunder of aircraft b. At step 3', the labels a and b are inverted, so that a possible blunder of aircraft a is also taken into account.

4. The relative position of the two aircraft can be computed directly from the latitude and longitude of each aircraft. We ignore earth curvature for conflicts, since the size of the reachable sets for our computations rarely exceeds 50nm. The heading difference ψ_r can be directly computed from the ETMS data as well (Figure 3.10). Using ψ_r, one can slice $\mathcal{G}(\tau)$ at ψ_r (Figure 9.2). This provides a 2D set. Plotting (x_r, y_r) on top of the slice of $\mathcal{G}(\tau)$ shows if the aircraft b is inside $\mathcal{G}(\tau)$ or not, and is a potential threat or not. All examples which follow will be displayed in this format for readability. In practice, the test $(x_r, y_r, \psi_r) \in \mathcal{G}(\tau)$ can be automated using level set methods[116]. It is instantaneous and consists of evaluating ϕ numerically from a grid using an interpolation subroutine.

5. If $(x_r, y_r, \psi_r) \in \mathcal{G}(\tau)$, there is a potential LOS within τ time units. Our method also

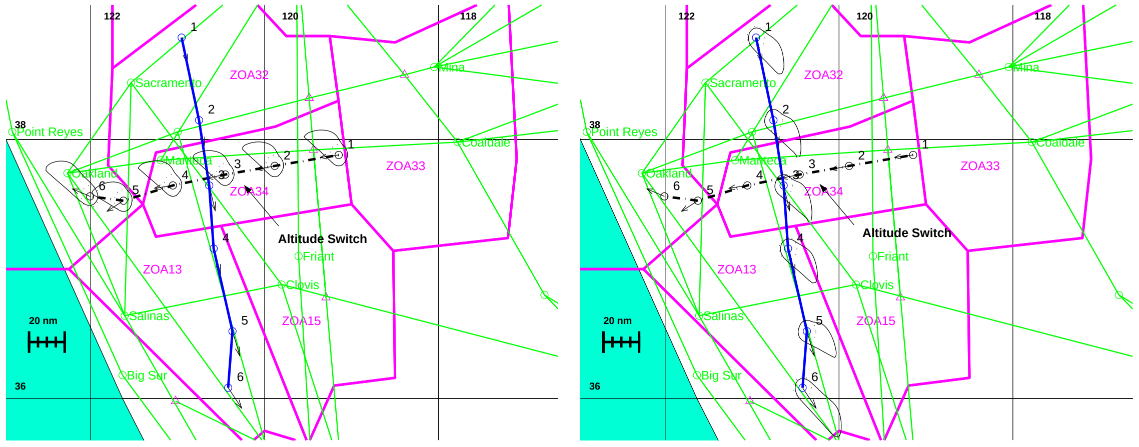


Figure 9.7: Aircraft 1 (dash dotted), arriving from Philadelphia Airport to San Francisco Airport (SFO); aircraft 2 (solid), en route from Ted Stevens Anchorage Airport to Los Angeles Airport. Aircraft 1 is using the Modesto 2 arrival to SFO. Interpolation of the positions of both aircraft between the labels 2 and 3 shows that aircraft 2 is in the $\mathcal{G}(\tau)$ for aircraft 1. ATC avoids the conflict by commanding aircraft 1 to descend to 24,000ft (which initiates the descent into SFO).

allows one to check if $(x_r, y_r, \psi_r) \in \mathcal{G}(\tau)\backslash\mathcal{G}(\tau')$ where $\tau' \leq \tau$. In this case, if τ' is large enough to climb or descend to the next floor, a LOS can be avoided by altitude change.

6. All pairs of aircraft have to be tested for potential threats.

7. The update rate of the ETMS data is on the order of 3 min., as appears in the examples presented in this chapter.

9.6.2 Applications and validation of the results

Several *metrics* have been defined in the past to help the decision making process in ATC [96]. The specific metric used here is *time to minimum separation*. It is easy to adapt our mathematical formulation to other metrics such as *predicted minimum separation* or *estimated time to closest point of approach* [96], using an equivalence theorem [116]. The goal of the rest of the chapter is to show that this metric is appropriate for short term LOS detection.

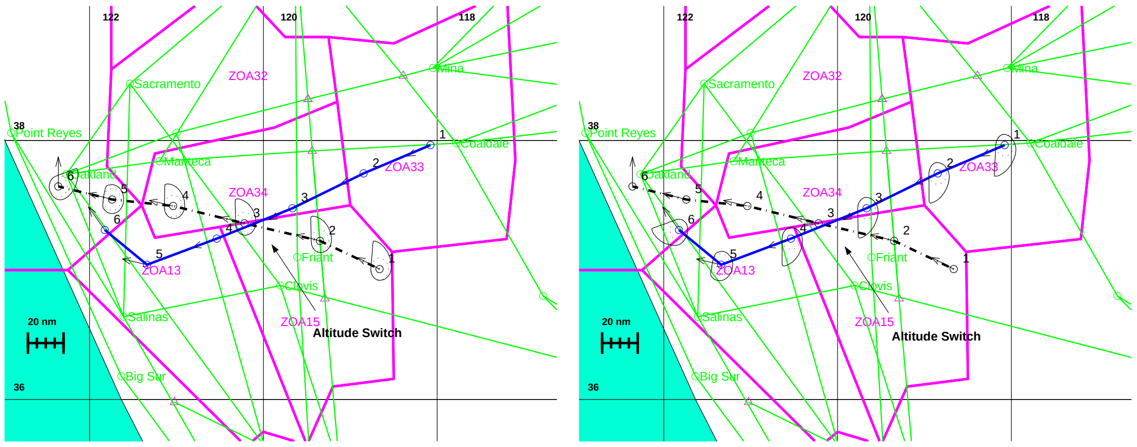


Figure 9.8: Aircraft 1 (solid), arriving from Chicago O'Hare Airport to San Jose Airport (SJC); aircraft 2 (dash-dotted), arriving from Phoenix Sky Harbor Airport to San Francisco Airport (SFO). Aircraft 1 is on El Nido arrival at 35,000ft entering through Coaldale; aircraft 2 is on Modesto 2 arrival, entering through Clovis. Interpolation between positions 2 and 3 shows that aircraft 2 is in the $\mathcal{G}(\tau)$ of aircraft 1. The action of ATC coincides with their respective descents: aircraft 1 is commanded to descend to 24,000ft, while aircraft 2 is commanded to descend to 11,000ft (starting from position 2 and 3 respectively).

Our method provides LOS prediction for aircraft pairs, and the corresponding horizontal maneuvers to apply to one of the aircraft in order to avoid the LOS. It is not always realistic however to apply the optimal heading changes provided by ω_a^* directly to a real aircraft: even though it has already been implemented in practice for UAVs [146], the technical difficulties for communicating ω_a^* from the ground ATC to an aircraft are still significant, and enhanced by the ill-behaved nature of ω_a^* [116]. However, our goal is not an implementation of the controller provided by our method, but the use of the reachable set as a safety set for LOS alert; the maneuver assignment is left to ATC.

In order to assess the value of our metric, we run it on an ETMS data sample, and rate its success in the following way. We classify the scenarios encountered by the method in four categories. Each pair of aircraft treated can be in one of the following cases:

- *Detected conflicts.* A conflict is detected when the method presented in the previous section assesses a threat, and the ETMS data unambiguously shows an actual conflict resolution by the ATC. We consider the following conflict resolution protocols unambiguous: (1) altitude change prior to LOS, (2) significant heading change before LOS. Case (1) is easily detected from the altitude tag in the ETMS data. Case (2) is harder and requires comparison with filed flight plan or current heading.

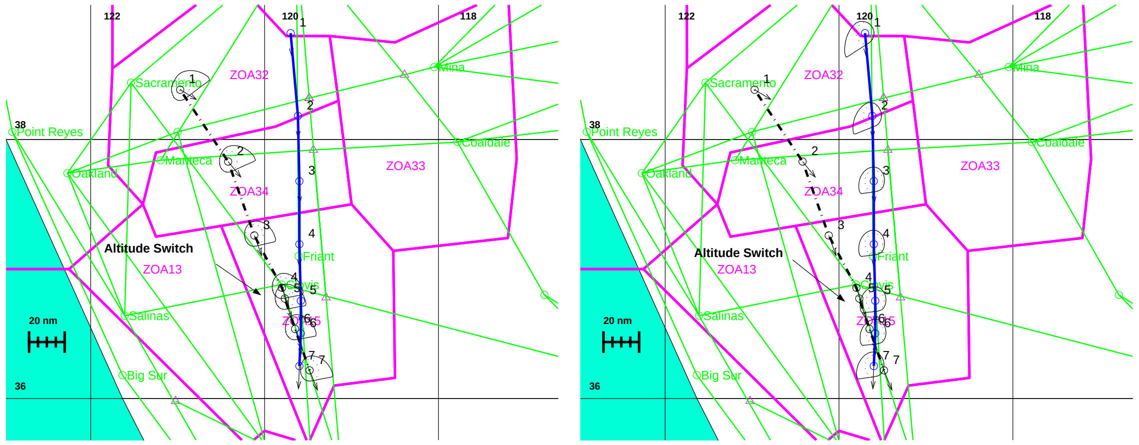


Figure 9.9: Aircraft 1 (solid), en route from Reno/Tahoe Airport to Los Angeles Airport; aircraft 2 (dash-dotted), en route from Sacramento Airport to San Diego Airport; both aircraft are cruising at 33,000ft when aircraft 1 is commanded to climb to 37,000ft by the ATC, at the position labeled by 4. At position 5, both aircraft are in the $\mathcal{G}(\tau)$ of the other aircraft.

- *Conflict detected with interpolation.* Because of the sampling rate of ETMS data (one update every 3 min.), our method might miss occurrences of $(x_r, y_r, \psi_r) \in \mathcal{G}(\tau)$. Interpolation of the flight path between the sample points might however provide a *detected conflict* from the previous category. Since this was not observed directly from the data, but from an interpolation, we call this "conflict detected with interpolation". An implementation in ATC systems would not suffer from such problems because the update rate would be much faster.

- *False alarms.* When our method detects a threat, but no reaction from the ATC was observed in the ETMS data, we call the situation a false alarm. It is due to the fact that our method is conservative: one of the aircraft did not do the worst possible action ω_b^* which is accounted for in our algorithm.

- *Not applicable.* Any pair of aircraft for which no threat is detected. Most of the aircraft pairs will fall into this category, because they are very far from each other.

We realized a sample study for one hour of traffic for all altitudes between floors 250 and 370 (25,000 ft and 37,000 ft respectively) for the complete Oakland ARTCC, and classified all aircraft pairs using the categories above. The result is presented in Table 9.3. The number of "not applicable" pairs is obviously greater than all other numbers; it illustrates the fact that the number of conflicting pairs of aircraft is small in comparison with the total

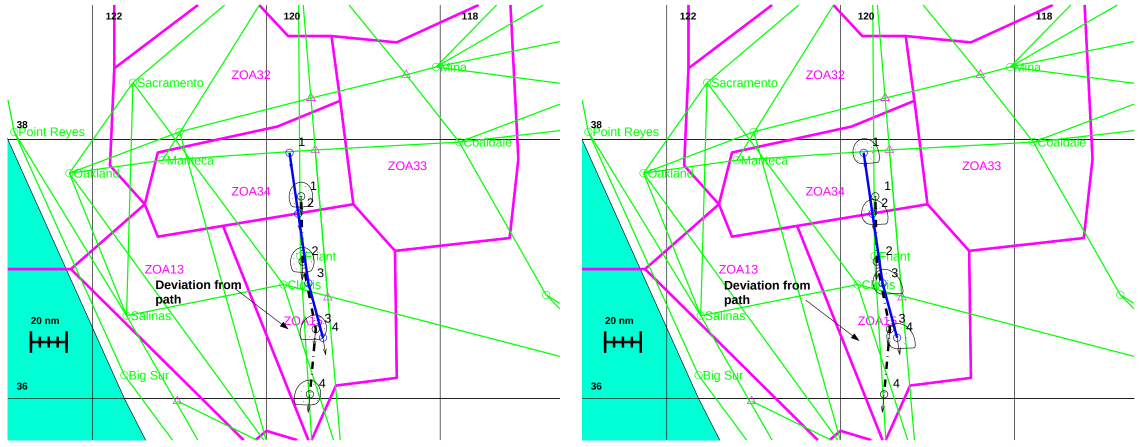


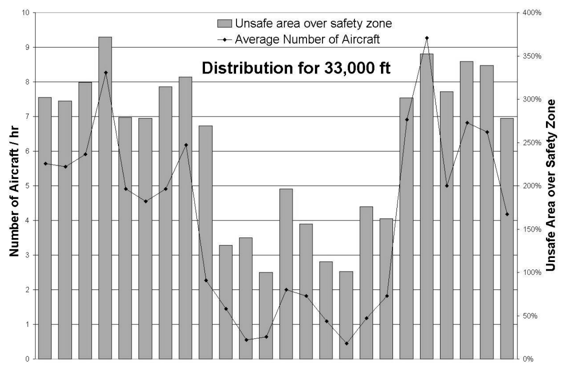
Figure 9.10: Aircraft 1 (solid) and aircraft 2 (dash-dotted) are both en route from Seattle-Tacoma Airport to Los Angeles Airport, cruising at 33,000ft. Keeping aircraft on their path (Jetway 7) would lead to LOS, predicted by our algorithm, using interpolation between 3 and 4. The maneuver chosen by ATC (aircraft 2 deviates from its path, and joins again) is not obvious from the data, but avoids LOS.

number of pairs of airborne aircraft. We see from Table 9.3 that our method only gives two false alarms out of 33 cases, which is satisfying. Figures 9.6 to 9.10 illustrate some of the detected conflicts and also conflicts detected with interpolation accounted in Table 9.3. We also observe that most of the cases we found in this study were conflict avoidance scenarios resolved by altitude changes. Very often, these altitude changes coincide with descent of aircraft into destination airports, as in Figures 9.6, 9.7 and 9.8.

To further justify the usefulness of the method proposed, we also need to show that our results do not saturate the airspace display with unsafe sets, i.e. that the cumulative size of the unsafe areas does not cover too large a portion of the airspace. We determine the total airspace area which our method computes as unsafe, at a given time (i.e. cumulating all aircraft pairs) and plot its value divided by the area of just the protected zones $\mathcal{G}_0$, over a 24 hour period. The result is shown in Figure 9.11 for two flight altitudes. We see that the results never exceed a factor of 4: the total unsafe area marked by our algorithm is never bigger than 4 times the area enclosed by the safety disks of all aircraft. This confirms our belief that this method is applicable to a real ATC system, in that it does not produce too many false alarms or saturate the display with large unsafe sets.

Floor	D.C.	D.C.w.I	F.A.	N.A	T.C.P.
250	3	2	0	205	210
270	1	1	0	256	258
290	2	2	0	288	292
310	0	1	0	186	187
330	7	2	1	198	208
350	6	6	1	361	374
370	0	0	0	61	61
tot.	19	14	2	1555	1590

Table 9.3: Sample study for one hour of ETMS data for flight levels ranging from 25,000ft to 37,000ft. All aircraft pairs are classified in one of the four categories: detected conflicts (D.C), detected conflicts with interpolation (D.C.w.I), false alarms (F.A.), not applicable (N.A). The sum is shown in the last column as Total Counted Pairs (T.C.P.).



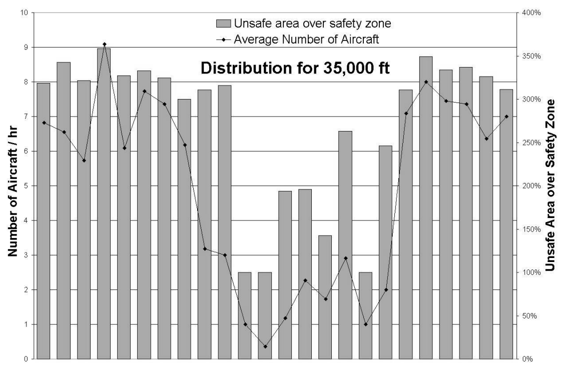


Figure 9.11: Average number of aircraft per hour at a given altitude, over a 24 hours period. Ratio of the area labeled potentially unsafe $\mathcal{G}(\tau)$ over the sum of the protected areas ($\mathcal{G}_0$). This ratio does not exceed 4, which confirms that our method provides results which are compatible with current ATC displays (there will be no saturation of the airspace display with unsafe sets).

Chapter 10

Summary, extensions and open problems

In this very brief and final chapter, we summarize the work presented in this dissertation and outline some future directions. Some problems are direct extensions of the work presented here; others are more fundamental questions or open problems, which might be of interest on their own.

10.1 Summary

After a brief presentation of the current Air Traffic Control system in the US, we presented a new Lagrangian model of sector air traffic flow. The Lagrangian approach describes the trajectories of each aircraft in the system. The model uses the framework of hybrid systems, which enables us to decompose the trajectories into specific maneuvers, which we identified as currently in use by Air Traffic Controllers. We modeled the behavior of human Air Traffic Controllers with a cost function, incorporating the different levels of priority for the tasks which they accomplish in controlling the air traffic flow. The model was successfully validated against ETMS data.

A subset of this model was used to describe arrival traffic in the vicinity of large airports. The problem of routing and sequencing the aircraft was posed as a hybrid system controller synthesis problem. It was then reduced to a MILP. Numerical experiments have shown that using commercially available software to solve it can lead to unpredictably large computation times, and is therefore not suitable for an online implementation.

A new polynomial time algorithm was developed for a subproblem of the general MILP. The algorithm relies on an analytical solution which we derived for the case in which the order of arrival of the aircraft is known a priori. Numerical experiments have confirmed the applicability of the method for numbers of aircraft up to 100, which is an upper bound for realistic scenarios. 3- and 5-approximation algorithms were developed for two other subproblems of the general MILP, in which the aircraft are allowed to achieve an integer number of holding patterns before being released back into the system. The two algorithms minimize the sum of arrival times of all aircraft, and the arrival time of the last aircraft. To our best knowledge, despite striking similarities with problems available in the literature, these problems and algorithms are new.

A new Eulerian model of air traffic flow was presented. This approach models the air traffic as the flow of a fluid in pipes. The density (of the fluid) represents the density of aircraft. This approach is suitable for high altitude traffic scenarios, in which large numbers of aircraft are involved, and for which the goal is to control the density of aircraft. A PDE was derived to describe the evolution of the density over time, inspired by the Lighthill-Whitham-Richards PDE usually introduced in the context of highway traffic. A control paradigm was developed to control systems driven by networks of PDEs. It relies on the computation of the adjoint of the system of coupled PDEs governing the network. It was successfully applied to the air traffic control case of interest.

Finally, Hamilton-Jacobi formulations of reachability analysis were investigated, which can be used to compute unsafe sets around aircraft in the NAS. First, a link between viability theory and Hamilton-Jacobi formulations of reachability was made. Such a link had already been made for the static HJE. It is to our best knowledge the first for the time dependent HJE. A characteristics-based method was developed to compute analytical solutions of the HJE, which provides a solution to the Mayer problem. Some applications of this method have significantly clarified questions raised in the past about convergence of level set methods and disappearance of level sets in numerical computations. Finally, these formulations were applied to ETMS data and enable us to show that the differential game setting can be used efficiently to assess alert levels in the NAS.

10.2 Extensions of this work

10.2.1 Modeling

Modeling is an open topic for at least two reasons. First, it might be useful to refine the model for applications other than the ones presented in this dissertation. We will outline some future directions below. Second, once the model is in place, validation can improve our confidence in the model when its results confirm observations; further tests which we also outline below might help validate additional features of our model, in particular with respect to its predictive capabilities.

Lagrangian air traffic model

Even though aircraft dynamics could be included in the Lagrangian model, we did not do it. We feel that the dynamics of the aircraft is relevant for the collision avoidance scenarios, for which the precise knowledge of aircraft motion can turn out to be crucial; we do not feel that they would bring any significant improvement to the MILP formulation of the routing scheduling problems.

- The model of the human Air Traffic Controller could be improved to include sector specific requirements, based on traffic flow observations made at relevant ARTCCs. Our approach for this work was to avoid disjunctions of "if", "then", "else" statements in favor of a more systematic optimization approach. It is, however, clear that modeling a human with cost functions has its limits, and methods inspired by artificial intelligence or human factors might significantly improve the fidelity with which the simulator reproduces the flow. From an ATC perspective, there is a lot of interest in building predictive models which could simulate how humans react under stress.

  The next level of modeling which this work did not address is modeling the Traffic Management Unit, which takes strategic decisions at the center level (ARTCC). The modeling at this level, and its interaction with the sector level would be facilitated by the existence of the playbooks mentioned earlier, which also implies that the model is center dependent.

- Validation of a model is of equal importance. A recent study [84] classified days in the NAS into classes: "good days", "congested days", etc. This study could enable one to test the predictive capabilities of the model (and extensions) under stress and normal conditions, which would be very useful. Another achievement would be to validate the maneuver assignment based on observations realized at any ARTCC. For example: correlating the recorded decisions of a human Air Traffic Controller with the simulation from ETMS data, from the same day for the same airspace, would provide strong statements for the validity of our cost function and solution mechanism.

Eulerian air traffic model

A common problem of Eulerian networks is the treatment of the destination of the different agents (aircraft or cars). This was not an issue for the present case, since the diverging link was only used for control purposes. A challenging extension of this work would be to incorporate several major destinations in the network (and not only Chicago as in the present case). This difficulty was also noted by Menon [110] and seems to appear in highway networks as well.

A possible extension of the Eulerian approach is to add one more dimension to the problem, and treat the air traffic flow as a two dimensional problem. This approach was suggested by Menon [111]. The difficulty of multiple origins and destinations becomes even more crucial in this setting, but its advantage is the possibility to control a surface density (vs. a line density, i.e. along a line, as in the present work).

The control methodology applied here can also be applied to other physical systems which can be modeled by a system of coupled PDEs. We recently applied it successfully to highway networks [23]. Another possible extension would be networks of irrigation channels, as presented in [104].

10.2.2 Combinatorial optimization algorithms

It was recently suggested to us by one of the authors of [41] to adapt the algorithm of the one interval case to a problem of dimension two, sometimes referred to as the *square packing problem.*

The square packing problem consists of placing N squares of width Δ into N boxes (which can overlap), such that the squares do not overlap. In mathematical terms, consider a given set of $l_i \in \mathbb{R}^2$, and $u_i \in \mathbb{R}^2$, where $i \in \{1, \cdots, N\}$. In the two dimensional plane, u_i represents the upper right corner of box i, and l_i the lower left corner of box i. Given $\Delta > 0$, the square packing problem consists of finding a set of x_i, $i \in \{1, \cdots, N\}$ satisfying

$$l_i \preceq x_i \preceq u_i$$

with the constraint $||x_i - x_j||_\infty \geq \Delta$ for all $i \neq j$ (i.e. either $|x_{i1} - x_{j1}| \geq \Delta$ or $|x_{i2} - x_{j2}| \geq \Delta$), with usual notations of [41]. Using a modification of the analytical solution found for the aircraft case, it is clear that solving this feasibility problem leads directly to the computation of the largest possible Δ. An open problem is to see how the feasibility algorithm used for the aircraft case can be extended to this two dimensional problem.

An extension of the approximation algorithms presented in Chapter 5 is to use other objective functions (for example, total number of holding patterns, or total number of aircraft put on a holding pattern). It would be of great interest to extend this algorithm to the case in which the payoff is the sum of the delays (and not the arrival times). Minimizing the sum of arrival times or minimizing the sum of the delays is the same. Of course, the approximation ratio is different. Finding the approximation ratio for this new objective function would be of interest in order to relate the output of the algorithm to metrics currently in use in Air Traffic Control.

10.3 Open problems

Some fundamental questions remain open. Solving them would constitute a significant scientific contribution. We list them below.

- Prove that the combinatorial optimization problems of Chapters 4 and 5, with more than one interval of feasible arrival times, are NP-complete with respect to the number of aircraft and/or the number of intervals.

- Prove that the analytically constructed solution of the Mayer problem is the viscosity solution of the HJE. The only guarantee provided by the method is that the solution is continuous. In cases in which it is not continuously differentiable, it is at best a weak solution. It is possible to prove a posteriori that a constructed solution is the viscosity solution (for example using definitions from [74, 57]). However, proving that this is true in general (i.e. for any such solution constructed with our method) is an open problem. Very few analytical expressions of viscosity solutions of the HJE are known (a few simple examples are provided in [14]). It would also be interesting to relate this work to the theory of generalized characteristics [109] which itself is related to the concept of the viscosity solution.

- Prove that Theorem 6 follows directly from the definition of the discriminating kernel [46]. This involves generalizing the proof of Theorem 6 in Section 8.2.3 to the case of differential games.

Appendix A

Overapproximation algorithms for the reachable set

This appendix provides an overapproximation algorithm to compute the reachable set. In [16], we sketched several proofs to show that this algorithm is overapproximative, which means that its result will enclose the true reachable set, an important property if the reachable set represents unsafe regions, as in Chapters 8 and 9.

A.1 Overapproximation method for constrained viability

A crucial underlying assumption made in Chapters 8 and 9 is that the problem is unconstrained, i.e. that the state can take any value within $\mathbb{R}^n$. Most of the known PDE-based techniques, such as the HJE methods presented above, rely on this assumption. If the domain is bounded, i.e. the state space is not $\mathbb{R}^n$, but a subset $K \subset \mathbb{R}^n$, the previous result becomes invalid. An intuitive reason for this fact is that for a given dynamics and a given target $\mathcal{G}_0 \subset K$, it might not be possible to reach $\mathcal{G}_0$ while staying in the constraint set K. This fact is illustrated in Figure A.1.

The difficulty of incorporating constraints into a reachability problem was one of the major motivating factors in the emergence of viability theory [6]. The subsequent viability results, including optimal control, Lyapunov theory, differential games and finally hybrid systems [8,

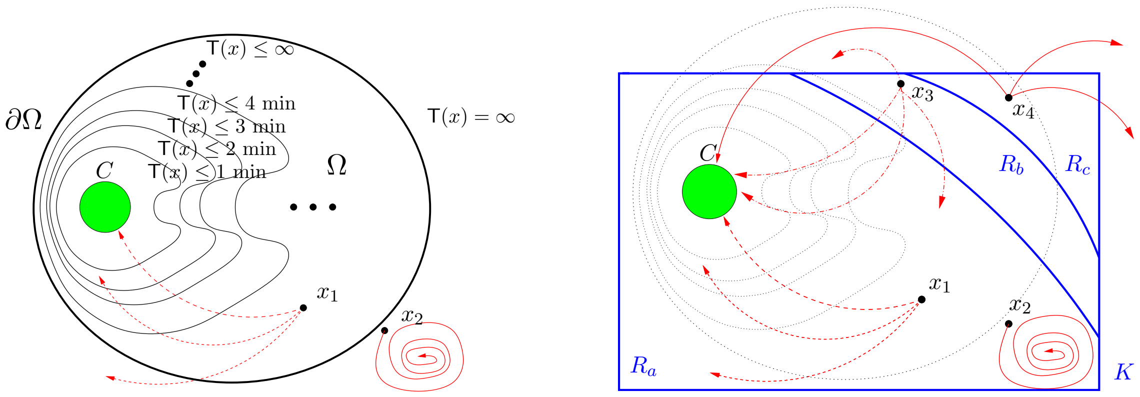

$\mathsf{T}(x) \leq \infty$
$\partial\Omega$
$\mathsf{T}(x) \leq 4$ min
$\mathsf{T}(x) \leq 3$ min
$\mathsf{T}(x) \leq 2$ min
$\mathsf{T}(x) \leq 1$ min
$\mathsf{T}(x) = \infty$
C
Ω
x_1
x_2
x_3
x_4
R_a
R_b
R_c
K

Figure A.1: **Left:** Diagram of the isolines of the unconstrained minimum time-to-reach function $\mathsf{T}(x)$. Ω is the set in which this function is finite (and therefore, there exist a trajectory leading into the target). For example there exists a trajectory leading from x_1 to the target, but there does not exist a trajectory leading from x_2 to the target. **Right:** If in addition, the trajectories are constrained to remain in a set $K \subset \mathbb{R}^n$, K might be divided into several subsets: in R_a the points are such that any trajectory emanating from them stays in K forever or reaches $\mathcal{G}_0$ in finite time (for example x_1 or x_2); in R_b some of the trajectories exit K; in R_c, it is not possible to reach $\mathcal{G}_0$ without exiting K. Despite the fact that $\mathsf{T}(x_3) < \infty$, the minimum time-to-reach the target from x_3 while staying in K is infinite.

44, 60, 61] enabled a systematic mathematical treatment of constraints. In the rest of this chapter, we will use these results to show how to incorporate constraints into a reachability problem.

A.1.1 Constrained reachability using minimum time-to-reach functions

We use the formalism developed in Section 8.1 and consider a dynamical system with control input a:

$$\begin{cases} \dot{x}(t) = f(x(t), a(t)), \quad t > 0 \\ x(0) = x \end{cases} \tag{A.1}$$

where $a(\cdot) \in \mathfrak{A}(t)$ as defined in Assumption 1; f is continuous in a and Lipschitz-continuous in x. We rewrite (A.1) in set valued form as a differential inclusion:

$$\begin{cases} \dot{x}(t) \in F(x(t)) \ := \{f(x,a)\}_{a \in \mathcal{A}} \\ x(0) = x \end{cases} \tag{A.2}$$

The set of solutions of (A.2) (equivalently of (A.1)) is denoted $\mathcal{S}_F(x)$. Since (A.1) and (A.2) are strictly equivalent [10], we will use them interchangeably. In order to adopt the traditional notations of viability theory, we change the notations slightly with respect to the previous sections. Let $K \subset \mathbb{R}^n$ be a constraint set and C be a closed set in K (C was formerly called $\mathcal{G}_0$). A generic element of $\mathcal{S}_F(x)$ will be denoted $x(\cdot)$ (and was formerly denoted $\zeta_F(\cdot; x)$). Consider the following problem: find the set of initial conditions x for which there exists a trajectory starting at x remaining in K and reaching C in finite time. In mathematical terms, we seek

$$W_C^K = \{x \in K : \exists x(\cdot) \in \mathcal{S}_F(x),\ \exists t \geq 0, x(t) \in C \wedge (\forall s < t, x(s) \in K)\} \tag{A.3}$$

We define the constrained minimum time-to-reach function accordingly:

$$\theta_C^K(x) = \inf_{x(\cdot)\in\mathcal{S}_F(x)} \inf\{t \in \mathbb{R}^+ : \quad x(t) \in C \quad \wedge \quad (\forall s < t,\ x(s) \in K)\} \tag{A.4}$$

and note that $\mathsf{T}(x) = \theta_C^K(x)$ when $K = \mathbb{R}^n$, where $\mathsf{T}(\cdot)$ was defined in Section 8.1. Note that $\theta_C^K(x) = +\infty$ if all the trajectories originating at x leave K before reaching the target, or stay in K forever without reaching C.

Fact 2. *W_C^K may be computed using the minimum time-to-reach function:*

$$W_C^K = \operatorname{Dom}\left(\theta_C^K\right) := \{x \in K : \theta_C^K(x) < +\infty\} \tag{A.5}$$

where Dom $(\cdot)$ *denotes the domain of definition of the function θ_C^K, or the set of points at which it is defined (here that is the set of points at which it is finite).*

The following can be found in [46] and is the same as (8.21) in the case with constraints:

Proposition 4. *Assume that in (A.2), F is uppersemicontinuous* with compact convex nonempty values and that K and C are closed. Then*

$$\mathsf{Epi}(\theta_C^K) = \mathsf{Viab}_\Phi\left(K \times \mathbb{R}^+\right) \tag{A.6}$$

where $\mathsf{Epi}(\theta_C^K) := \{(x, y) \in K \times \mathbb{R}^+ : y \geq \theta_C^K(x)\}$ *denotes the epigraph of the function θ_K^C,*

*A set valued map $F : X \rightsquigarrow X$ with compact values is uppersemicontinuous iff $\forall x_0 \in X$, and $\forall \varepsilon > 0$, $\exists \eta > 0$ such that $\forall x \in x_0 + \eta\mathcal{B}$, $F(x) \subset F(x_0) + \varepsilon\mathcal{B}$.

i.e. the set of points above its graph, and where

$$\Phi(x) = \begin{cases} F(x) \times \{-1\} & \text{if} \quad x \notin C \\ \overline{\mathsf{ch}}\{F(x) \times \{-1\}, \{0,0\}\} & \text{if} \quad x \in C. \end{cases} \tag{A.7}$$

In (A.7), $\overline{\mathsf{ch}}$ denotes the closure of the convex hull ch of the set between brackets (i.e. the closure of the smallest convex set containing it). Proposition 4 states that the set of points above the graph of the minimum time-to-reach function is the set of initial states $(x, y) \in K \times \mathbb{R}^+$ such that the trajectories $(x(\cdot), y(\cdot)) \in \mathcal{S}_\Phi((x, y))$ reach $C \times \mathbb{R}^+$ in finite time. Even if we do not make direct use of (8.15),(A.6),(A.7) in the present work, they have proved crucial in the development of the techniques used here. Indeed, Proposition 4 links the minimum time-to-reach function to the viability kernel and therefore enables the use of the *viability kernel algorithm* (Frankowska and Quincampoix [75]) whose numerical implementation (Saint-Pierre [139]) provides a guaranteed overapproximation of Epi $\left(\theta_C^K\right)$. In subsequent work, Cardaliaguet et al. [44] tailored the viability kernel algorithm to the computation of the minimum time-to-reach function. In [141], Saint-Pierre proposes a further simplification this algorithm. In the next section, we present our numerical algorithm inspired by [141].

A.1.2 Approximation Algorithm

We present an algorithm to underapproximate the minimum time-to-reach function under state constraints, θ_C^K, adapted from [46, 141] for our design. The proof of this algorithm has been sketched in [16]. The inclusion of state constraints will allow us to ignore the problem of boundary conditions. It gives good insight into the approximation procedure: the algorithm computes the exact minimum time-to-reach function for a discrete time dynamics defined on a discrete state space. Hence, we begin by showing how we can define a fully discrete dynamics whose trajectories are *good* approximations of the trajectories of system (A.2) (good will be defined later).

Numerical Approximation of Continuous Dynamics

We endow $X := \mathbb{R}^N$ with the Euclidean norm $\|\cdot\|$, and we denote by $\mathcal{B}$ the unit ball under this norm. For $h > 0$, we set $X_h = \left(h\mathbb{N}/\sqrt{N}\right)^N$, where $\mathbb{N}$ is the set of natural numbers. Then $\forall x \in X$, $\exists x_h$ in the ball $x + h\mathcal{B}$. Hence, X_h is a discrete approximation of the state space X. The following claims defines approximations of $\mathcal{S}_F(x)$ of the system (A.2) by the set of trajectories $\mathcal{S}_\Gamma(x_h)$ of discrete dynamics $\Gamma: X_h \rightsquigarrow X_h$.

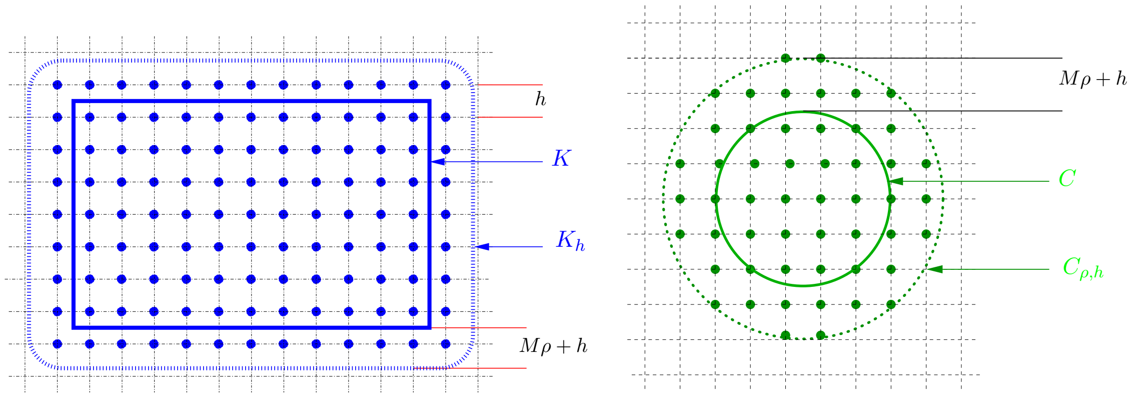


Figure A.2: **Left:** Discretization of space $X_h = h\mathbb{N}^2$ and the domain : $K_h = (K + h\mathcal{B}) \cup X_h$. **Right:** discretization of the target: $C_{\rho,h} = (C + (M\rho + h)\mathcal{B}) \cup X_h$.

Claim 1 (Relationship between continuous and discrete trajectories). *Assume that $F : X \rightsquigarrow X$ is upper semicontinuous with nonempty convex compact values and is l-Lipschitz. Assume moreover that there exists $M > 0$ such that for all $x \in K$, $\sup_{y\in F(x)} \|y\| \leq M$. For a mesh $h > 0$ and a time step $\rho > 0$, we define discrete dynamics on X_h:*

$$x_h^{n+1} \in \Gamma_{\rho,h}(x_h^n) := [x_h^n + \rho\,(\,F(x_h^n) + r(\rho, h)\mathcal{B}\,)] \cap X_h, \tag{A.8}$$

where $r(\rho, h) = lh + Ml\rho + 2\frac{h}{\rho}$, and we define the set of trajectories of this system as $\mathcal{S}_{\Gamma_{\rho,h}}(x_h)$. Then a trajectory $x(\cdot)$ of system (A.2) defines trajectories of system (A.8) in the following way:

$$\forall \{x_h^n\} \in \{\{y_h^n\} : \forall n \in \mathbb{N},\, y_h^n \in (\,x(n\rho) + h\mathcal{B}\,)\}\,, \qquad \{x_h^n\} \in \mathcal{S}_{\Gamma_{\rho,h}}(x_h), \tag{A.9}$$

and a trajectory $\{x_h^n\} \in \mathcal{S}_{\Gamma_{\rho,h}}(x_h)$ is close to a trajectory $x(\cdot) \in \mathcal{S}_F(x_h)$ in the following *sense:* $\forall t \geq 0$

$$\|x(t) - \hat{x}(t)\| \leq \begin{cases} \left((\ M + r(\rho,h)\ )\,\rho + \frac{r(\rho,h)}{l} \right) (\mathrm{e}^{lt} - 1) & \textit{if} \quad l > 0 \\ 2\,\frac{h}{\rho}\,t & \textit{if} \quad l = 0 \end{cases} \tag{A.10}$$

where $\hat{x}(t) = x_h^n + \frac{x_h^{n+1} - x_h^n}{\rho}\,(t - n\rho)$, *for* $n \in \mathbb{N}$ *and* $t \in [n\rho, (n+1)\rho]$, *represents a continuous trajectory interpolating points in* $\{x_h^n\}$.

This claim states that for all ρ and h, the dynamics (A.8) is an *overapproximation* of dynamics (A.2) in the following sense: all trajectories of system (A.2), when discretized with time step ρ and projected on X_h, are trajectories of (A.8); and all the trajectories of (A.8), when interpolated as in $\hat{x}(t)$ above, are approximations of trajectories of (A.2), with an upper bound on the error given by an increasing function of $\eta(\rho,h)$ (with $\eta(\rho,h) = 2h/\rho$ if $l = 0$) and of time. Therefore, the smaller the $\eta(\rho,h)$, the better the approximation. Moreover, $\eta(\rho,h)$ tends to 0 if and only if ρ, h and h/ρ tend to 0. This will be used in the approximation algorithm for the minimum time-to-reach function.

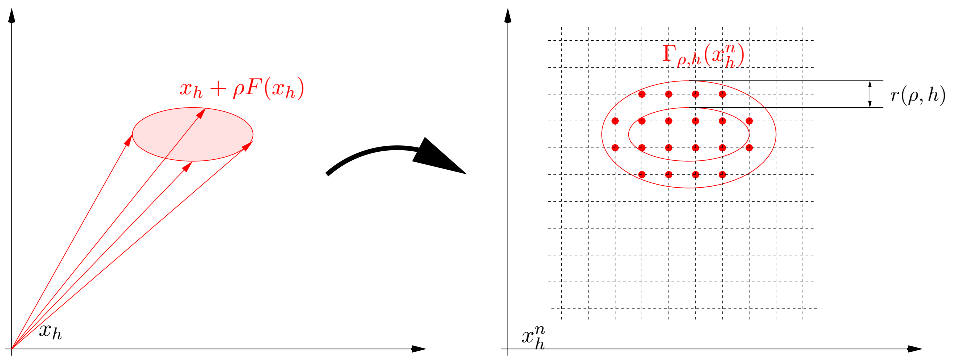


Figure A.3: The set valued dynamics F is discretized as shown in (A.8)

Approximation of the minimum time-to-reach function.

We will now define a fully discrete target problem which approximates the target problem defined for continuous time and state space, and will show relationships between the discrete minimum time-to-reach function and the continuous one. We shall use Θ to denote the discrete approximation of θ_C^K, its sub/superscripts will depend on context (and will always correspond to the sub/superscripts of θ).

We begin by defining a discrete approximation, and the sense in which discrete functions can converge to continuous functions.

Definition 2. *We say that a discrete function $\Psi_h : X_h \to \mathbb{R}$ is an* underapproximation *of a function $\theta : X \to \mathbb{R}$ if*

$$\forall x_h \in X_h, \quad \forall x \in (x_h + h\mathcal{B}), \qquad \Psi_h(x_h) \leq \theta(x),$$

and for a family (or a sequence) indexed by the set Ξ (containing elements called ξ) denoted $\{\Psi_{h,\xi}\}$, we write $\lim_{(h,\xi)\to(0^+,\xi_0)} \Psi_{h,\xi} = \theta$ if

$$\forall x \in X, \qquad \lim_{(h,\xi)\to(0^+,\xi_0)} \sup_{x_h \in (x+h\mathcal{B})\cap X_h} \Psi_{h,\xi}(x_h) \;=\; \theta(x).$$

If moreover $\{\Psi_{h,\xi}\}$ is a underapproximation of θ for all $h > 0$ and all $\xi \in \Xi$, we say that it defines a converging underapproximation scheme of θ.

Claim 2 (Discrete function is converging underapproximation of continuous). *Let $K \subset X$ be a closed set of constraints and $C \subset X$ be a closed target. Under the assumptions and notations of Claim 1 and for $\rho > 0$ and $h > 0$, we denote $C_{\rho,h} = (C + (M\rho + h)\mathcal{B}) \cap X_h$ and $K_h = (K + h\mathcal{B}) \cap X_h$, and we define the discrete minimum time-to-reach function:*

$$\Theta_{C_{\rho,h}}^{K_h}(x_h) = \inf_{\{x_h^n\}\in\mathcal{S}_{\Gamma_{\rho,h}}(x_h)} \inf\{n \in \mathbb{N} : x_h^n \in C_{\rho,h} \wedge \forall m < n,\, x_h^m \in K_h\} \tag{A.11}$$

Then $\left\{\Theta_{C_{\rho,h}}^{K_h}\right\}$ defines a converging underapproximation scheme of θ_C^K.

Corollary 3. *The minimum time-to-reach function θ_C^K can be underapproximated with use of a discrete function Θ^n, using the following algorithm:*

$$\Theta^0(x_h) \;=\; \begin{cases} 0 & \textit{if} \quad x_h \in K_h, \\ +\infty & \textit{else} \end{cases} \tag{A.12}$$

$$\Theta^{n+1}(x_h) \;=\; \begin{cases} \rho + \inf_{y_h\in\Gamma(x_h)} \Theta^n(y_h) & \textit{if} \quad x_h \notin C_{\rho,h}, \\ \Theta^n(x_h) & \textit{else} \end{cases} \tag{A.13}$$

Indeed, $\Theta_{C_{\rho,h}}^{K_h}(x_h) = \lim_{n\to+\infty} \Theta^n(x_h)$.

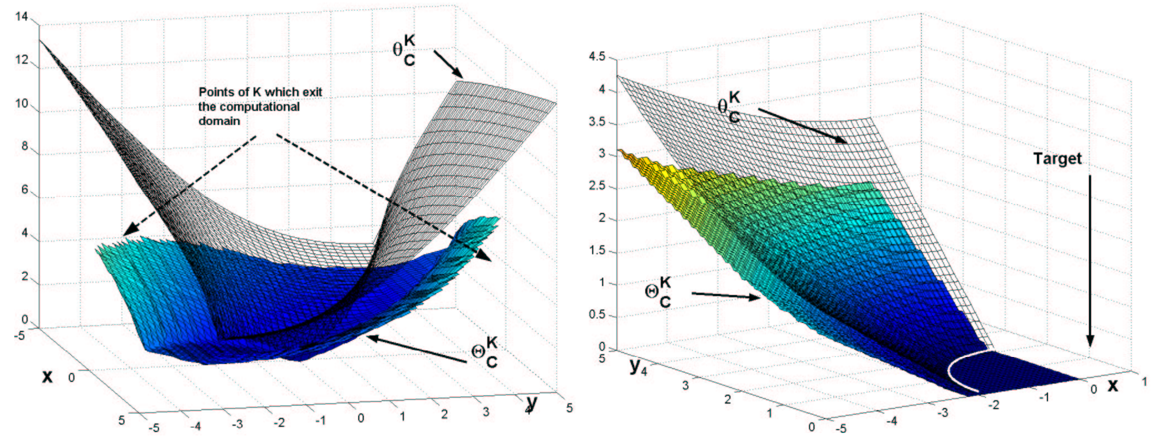


Figure A.4: **Left:** Numerical underapproximation of the value function of the double integrator problem (A.14), obtained by the the scheme presented in this paper. Computation realized on a 400 $\times$ 400 grid on $[-5, 5] \times [-5, 5]$, converged in 56 iterations, $\rho = 0.09$. **Right:** Numerical underapproximation of the value function of the wall pursuit evasion game (A.15). Computation realized on a 200 $\times$ 200 grid on $[-5, 0] \times [0, 5]$, converged in 36 iterations, $\rho = 0.09$. In both cases, three points out of four omitted in the plot for clarity. The numerical underapproximation of (A.15) is more accurate than that of (A.14). This is due to the zero Lipschitz constant of the dynamics of (A.15).

The choice of the two parameters ρ and h is a matter of trial and error. However, when one of them is fixed, an interesting hint for setting the other is to minimize either $r(\rho, h)$ or $\eta(\rho, h)$ which appear in Claim 1 and are indicators of the accuracy of the approximation by $\Gamma_{\rho,h}$.

A.1.3 Numerical validation

The claims of the previous section do not provide direct information about the rate of convergence of the algorithm, as other numerical techniques do (see for example Mitchell and al. [116]). Furthermore, the bounds shown in (A.10) are very large (therefore bad for numerical purposes), and do not give any indication how the solution behaves within them. We want to be able to quantify how close the underapproximation we provide is from the viscosity solution of the constrained Hamilton-Jacobi equation corresponding to the problem (see for example [14, 8]). We now show computations realized on two textbook-style examples for which we are able to compute analytically the viscosity solution.

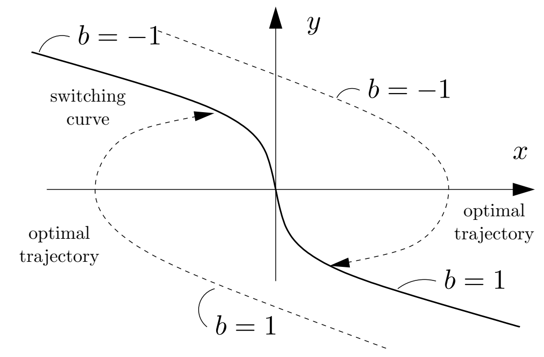


Figure A.5: Steering problem, after Bryson [43]. The switching curve determines the optimal control to reach $(0,0)$ in minimum time.

Steering Problem (after Bryson [43]). The dynamical system for this problem is: $(\dot{x},\dot{y}) = (y,b)$ where $b \in [-1,1]$, $(x,y) \in \mathbb{R}^2$. The viscosity solution $\theta^{\mathbb{R}^2}_{(0,0)}$ of the corresponding static HJI PDE for this dynamics is given by:

$$\begin{cases} \theta^{\mathbb{R}^2}_{(0,0)}(x,y) = -y + \sqrt{2y^2 - 4x} & \text{if } x + \frac{1}{2}y|y| \leq 0 \\ \theta^{\mathbb{R}^2}_{(0,0)}(x,y) = +y + \sqrt{2y^2 + 4x} & \text{if } x + \frac{1}{2}y|y| \geq 0 \end{cases} \tag{A.14}$$

The numerical results obtained from the underapproximation algorithm are compared to the viscosity solution $\theta^{\mathbb{R}^2}_{(0,0)}$ in Figure A.4 (left). Clearly, the numerical result is below the analytical. The error is due to the Lipschitz constant of this example as related back to (A.10). We note here that Saint-Pierre [139] has developed powerful techniques to alleviate this problem.We did not implement them: they are computationally expensive and our goal here is fast overapproximations.

Wall pursuit evasion game (after Isaacs [85]). We treat this problem as a control problem by forcing the evader to run in one direction and reducing the space to one quadrant. In this context: $(\dot{x},\dot{y}) = (-w\cos d, \text{sign}(y) - w\sin d = 1 - w\sin d)$ with $(x,y) \in \mathbb{R}^- \times \mathbb{R}^+$: The pursuer has speed $w > 1$ and can move any direction d. Isaacs' *retrograde path equations* method enables reducing this problem to solving $(l + w\theta(x,y))^2 = x^2 + (y + \theta(x,y))^2$, which provides the viscosity solution (A.15) of the corresponding static HJI PDE, shown in Figure A.4 (right), as well as its numerical underapproximation obtained with our algorithm.

$$\theta^{\mathbb{R}^-\times\mathbb{R}}_{B(0,l)\cap\mathbb{R}^-\times\mathbb{R}}(x,y) = \frac{1}{w^2-1}\left[-(lw - |y|) + \sqrt{(|y|w - l)^2 + (w^2-1)x^2}\right] \tag{A.15}$$

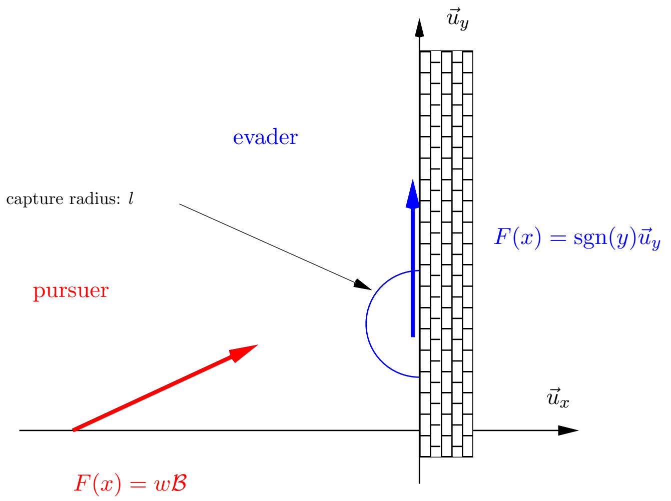


Figure A.6: Relative dynamics in the wall pursuit evasion game of Isaacs [85].

A.2 Conflict detection in merging traffic using underapproximations of the minimum time-to-reach function

We are now interested in performing fast computations of safety zones of aircraft for Air Traffic Management systems. Guaranteed underapproximation of those sets is crucial: certification of a conflict resolution protocol always requires a proof of its safety. We will here show the application of our technique to aircraft conflict resolution problems very frequently encountered in the Sector 33 airspace of the Oakland ARTCC.

Sector 33 is one of the busiest high altitude sectors in the US. It is at the junction of jetways coming from and going to Los Angeles, San Francisco, Oakland, San Jose, Las Vegas and is a collector of traffic from the east coast. At waypoint COALDALE in this sector, aircraft coming from Las Vegas may frequently conflict with aircraft going to the east coast at the same flight level.

We consider the following subproblem (the notations refer to Figure A.9 and 3.1). Let the local flight plan of aircraft 1 be jetway 92 towards COALDALE and then jetway 58 towards San Francisco, while the flight plan of aircraft 2 is jetway 58 through COALDALE . If the aircraft are in danger of a LOS, meaning coming closer than 5 nautical miles horizontally and 1000ft vertically to each other, the Air Traffic Controller will either reroute horizontally

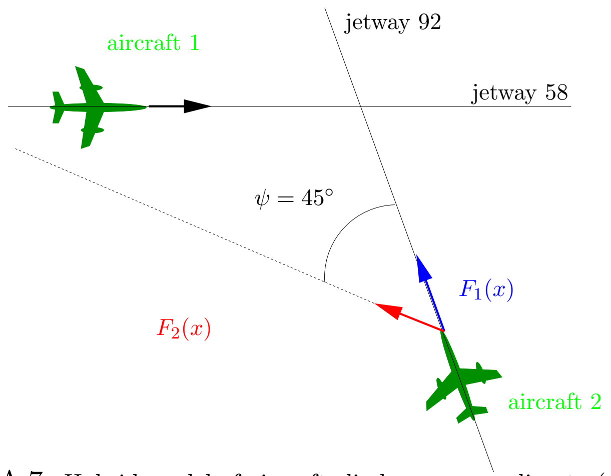


Figure A.7: Hybrid model of aircraft climb, corresponding to (A.17).

or climb one of the aircraft (i.e. will provide only one discrete action). The goal here is to develop advisories for Air Traffic Controllers, so that aircraft do not lose separation.

Let $x \in \mathbb{R}^2$ be the planar relative coordinate of the aircraft 1 w.r.t. aircraft 2, $v_{92} \in \mathbb{R}^2$ the velocity vector of aircraft 1 along jetway 92, and $v_{58} \in \mathbb{R}^2$ the velocity vector of aircraft 2 along jetway 58. We allow uncertainty in speed (due to winds, gusts, inaccuracy of sensors), with uncertainty bound of Mach $M = 0.05$.

$$\dot{x} = v_{58} - (v_{92} + 0.05 \cdot c \cdot \mathcal{B}) \qquad \text{Mode 1} \tag{A.16}$$

where c is the speed of sound and $\mathcal{B}$ is the unit ball in $\mathbb{R}^2$. We can now apply the results of Section A.1. Let us consider aircraft 1 as the evader and compute the safe set of its allowed positions (i.e. for which no loss of separation can occur). The unsafe set is the set of points which can eventually enter a disk target C around aircraft 2 of radius 5 nautical miles. Let us denote $\theta^{\text{mode}\,1}$ the minimal time-to-reach function for target C (there are no state constraints here). Then the safe set is $\mathbb{R}^2 \setminus \text{Dom}\left(\theta^{\text{mode}\,1}\right)$.

Conflict resolution via heading change (hybrid model).

A possible Air Traffic Controller choice is to make aircraft 1 "cut" between jetway 92 and jetway 58. This avoids the conflict and shortens the path of aircraft 1, and is the preferred option of Air Traffic Controllers in general. This can be modeled as a second mode of aircraft 1, now rotated by an angle ψ to the west:

$$\dot{x} = v_{58} - R_{\psi} \cdot (v_{92} + 0.05 \cdot c \cdot \mathcal{B}) \qquad \text{Mode 2} \tag{A.17}$$

where R_{ψ} is the standard $\mathbb{R}^2$ rotation matrix of angle ψ. Let us denote $\theta^{\text{mode}\,2}$ the minimal time function to reach C in this dynamics. An illustration is given in Figure A.7.

The Air Traffic Controller's policy is the following: if aircraft 1 is safe in mode 1, stay in mode 1; else if it is safe in mode 2, switch to mode 2; if both modes are unsafe, switching can be used to increase the time during which the distance between the two aircraft is guaranteed to be greater than 5 nautical miles.

Denote by θ^{hybrid} the function representing the minimum guaranteed time before loss of separation, and by F_1 and F_2 the set valued dynamics associated to the two modes, and by $\mathcal{S}_{F_1,F_2}(x,T)$ the set of trajectories originating at x for which switching from mode 1 to mode 2 occurs once. Then, at time T, we have:

$$\forall x \in X, \qquad \theta^{\text{hybrid}}(x) \;=\; \sup_{T>0} \; \inf_{x(\cdot)\in\mathcal{S}_{F_1,F_2}(x,T)} \; \inf\{t > 0 : x(t) \in C\} \tag{A.18}$$

with safe set $\mathbb{R}^2 \backslash \text{Dom}\left(\theta^{\text{hybrid}}\right)$. By definition, $\text{Dom}\left(\theta^{\text{hybrid}}\right) = \text{Dom}\left(\theta^{\text{mode}\,1}\right) \cap \text{Dom}\left(\theta^{\text{mode}\,2}\right)$ and $\forall x \in \mathbb{R}^2$, $\theta^{\text{hybrid}}(x) \geq \max\{\theta^{\text{mode}\,1}, \theta^{\text{mode}\,2}\}$. An algorithm for underapproximating θ^{hybrid} is presented in the next section.

Conflict resolution via floor climbing (reset model).

The second possible choice of the Air Traffic Controller is to climb aircraft 1. It takes about 3 minutes to climb an aircraft from one floor to the next floor. If there are no aircraft on the next floor and there is enough time to climb the aircraft, then the problem is solved. Let us investigate the case in which there is another aircraft on the next floor (aircraft 3).

Let aircraft 1 be on floor 350 (35,000 ft) on jetway 92 towards COALDALE , aircraft 2 be on floor 350 on jetway 58 towards COALDALE and aircraft 3 on floor 370 (37,000) on jetway 58

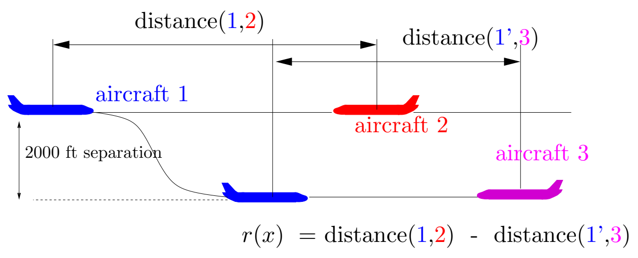


Figure A.8: Reset model of aircraft climb, corresponding to (A.19).

towards COALDALE (see Figure A.10). Given the positions of aircraft 2 and 3 on jetway 58 at their respective altitudes, we want to find the set of locations at which both collision cannot be avoided, and collision can be avoided by either climbing or staying at the same level. We assume that aircraft 2 and 3 are separated horizontally by a vector δ (regardless of their altitude) and fly at the same speed (which is usually the case on high altitude jetways). If it takes T_{climb} seconds to climb from floor 350 to floor 370 and the horizontal speed during climbing is unchanged, let r be the following *reset function*:[†]

$$r(\vec{x}) = \vec{x} + \vec{\delta} + T_{\text{climb}}\vec{v}_{92} \tag{A.19}$$

Then climbing aircraft 1 from floor 350 to floor 370 is equivalent to a reset. Let us call x the relative position of aircraft 1 w.r.t. aircraft 2: if $\theta(x) < T_{\text{climb}}$, there is not enough time to climb aircraft 1 without causing loss of separation with aircraft 2: the situation is unsafe. Otherwise, the aircraft can be climbed, and the algorithm in the next section will take this reset into account. Intuitively, the reset reinitializes the parameters by translating them by δ plus $T_{\text{climb}}v_{92}$, which is the ground distance needed to climb. As in the hybrid case, we will define θ^{reset} as the new minimal time-to-reach function which incorporates the possible reset within the execution of the automaton, and we will compute the set of points which are still unsafe when climbing is allowed, either because there is not enough time to climb, or the aircraft climbs to an unsafe zone on the next floor. An illustration is given in Figure A.8.

If we denote by $\mathcal{S}_{F_1,r}(x, T)$ the set of trajectories originating at x for which resetting occurs

[†]It is assumed that the reset function has no fixed point. This prevents the system from "jumping indefinitely". It is easy to check that the function (A.19) satisfies this assumption for realistic values of $\vec{\delta}$, T_{climb} and $\vec{v}_{92}$.

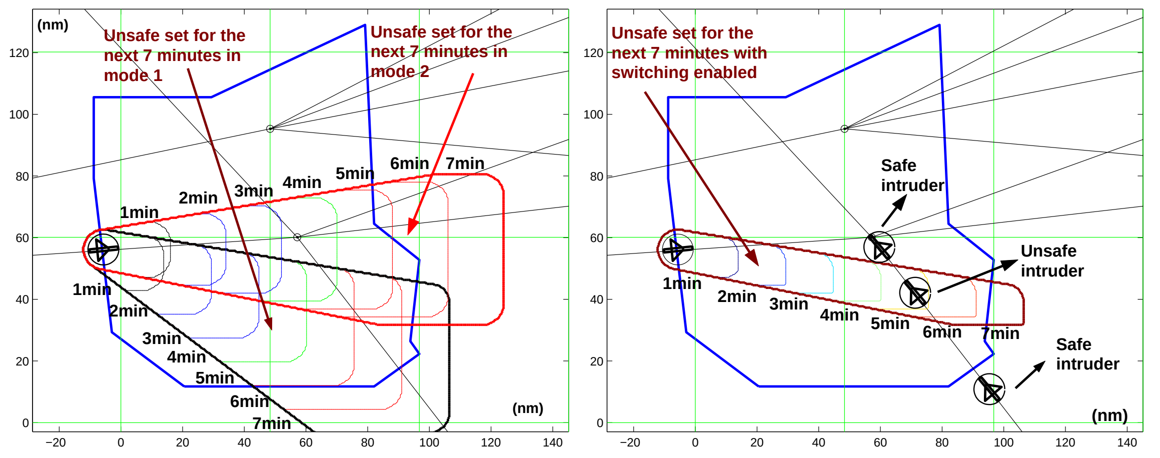


Figure A.9: Result of the conflict avoidance protocol with hybrid switching enabled. Computation realized on a 350 × 350 grid. **Left:** Reachability computation for the next 7 minutes for both modes (superimposed). Isolines are in increments of one minute in relative coordinates (which is why the distance between the isolines seems bigger than the one minute achievable distance of one aircraft at Mach $M = 0.85$ in absolute coordinates). If the intruder is in the intersection of the two unsafe sets, it cannot avoid loss of separation with the set of two maneuvers. If it is in one of the unsafe sets, switching avoids loss of separation. Otherwise, any of the two modes is safe. **Right:** Same as left with switching enabled. Position of first and third intruder are safe relative to these two dynamics. The second intruder cannot avoid loss of separation with only these two maneuvers.

at time T, we have

$$\forall x \in X, \qquad \theta^{\text{reset}}(x) = \sup_{T>0} \inf_{x(\cdot)\in\mathcal{S}_{F_1,r}(x,T)} \inf\{t > 0 : x(t) \in C\} \qquad \text{(A.20)}$$

and the safe set is $\mathbb{R}^2 \backslash \text{Dom}\left(\theta^{\text{reset}}\right)$. An algorithm for underapproximating θ^{reset} is presented in the next section.

Computing Safe Sets for Hybrid and Reset Systems

The algorithm presented below stems from Corollary 3 and provides an underapproximation of θ^{hybrid}. It is based on the fact that, given that the system starts in mode 1 and may switch to mode 2 at any time, $\theta^{\text{hybrid}}(x) = \theta^{\text{mode}\,2}(x)$ if $\theta^{\text{mode}\,2}(x) \geq \theta^{\text{mode}\,1}(x)$ because mode 2 guarantees safety longer than mode 1, otherwise $\theta^{\text{hybrid}}(x) \geq \theta^{\text{mode}\,1}(x)$ since mode 1 is safer now, and switching to mode 2 later may increase the time for which the system is safe.

Claim 3 (Approximation of the hybrid minimum time-to-reach function). *Let $C \subset X$ be a closed target. We assume that F_1 and F_2 satisfy the assumptions of Claim 1. For $\rho > 0$, $h > 0$ and $i \in \{1, 2\}$, we define the fully discrete dynamics $\Gamma^i_{\rho,h}$ and the discrete* *minimum time-to-reach functions* $\Theta^{mode\,i}_{\rho,h}$ *as in Claims 1 and 2. Let* $\mathcal{S}_{\rho,h} := \{x_h \in X_h : \Theta^{mode\,1}_{\rho,h}(x_h) \leq \Theta^{mode\,2}_{\rho,h}(x_h)\}$*. Then a converging underapproximation scheme for* θ^{hybrid} *is given by* $\Theta^{hybrid}_{\rho,h}(x_h) = \lim_{n\to+\infty} \Theta^{n}_{\rho,h}(x_h)$*, where*

$$\left[\begin{array}{lcl} \Theta^0_{\rho,h}(x_h) & = & \sup\{\Theta^{mode\,1}_{\rho,h}(x_h), \Theta^{mode\,2}_{\rho,h}(x_h)\} \\ \Theta^{n+1}_{\rho,h}(x_h) & = & \begin{cases} \rho + \inf_{y_h \in \Gamma^1_{\rho,h}(x_h)} \Theta^n_{\rho,h}(y_h) & \text{if} \quad x_h \notin \mathcal{S}_{\rho,h} \\ \Theta^n_{\rho,h}(x_h) & \text{else} \end{cases} \end{array}\right. \tag{A.21}$$

In the case of the reset model, the reasoning is similar to the hybrid case. Indeed, if a trajectory starts at x_0 with a reset at time T to x_1, we know that it cannot reach the target before $T + T_{\text{climb}} + \theta^{\text{mode}\,1}(x_1)$. In order to avoid loss of separation during climbing, we set

$$\theta^R(x) = \begin{cases} T_{\text{climb}} + \theta^{\text{mode}\,1}(r(x)) & \text{if } \theta^{\text{mode}\,1}(x) \geq T_{\text{climb}} \\ 0 & \text{else} \end{cases} \tag{A.22}$$

Then $\theta^R(x)$ plays the same role as $\theta^{\text{mode}\,2}$ in the hybrid model.

Claim 4 (Approximation of the reset minimum time-to-reach function). *Let* $C \subset X$ *be a closed target. We assume that* F *satisfies the assumptions of Claim 1 and that the reset function* $r : X \to X$ *is* λ*-Lipschitz continuous. For* $\rho > 0$*,* $h > 0$*, we define* $\Gamma_{\rho,h}$ *and the discrete minimum time-to-reach function* $\Theta_{\rho,h}(x_h)$ *as in Claim 1. We also define the discrete reset function*

$$R_h(x_h) := (\ r(x_h) + (1+\lambda)h\mathcal{B}\ ) \cap X_h$$

Then a converging underapproximation scheme for θ^R *is given by*

$$\Theta^R_{\rho,h}(x_h) := \begin{cases} \inf_{y_h \in R_h(x_h)} \Theta^{mode\,1}_{\rho,h}(y_h) + \frac{T_{climb}}{\rho} & \text{if} \quad \Theta^{mode\,1}_{\rho,h}(x_h) \geq \frac{T_{climb}}{\rho}, \\ 0 & \text{else} \end{cases}$$

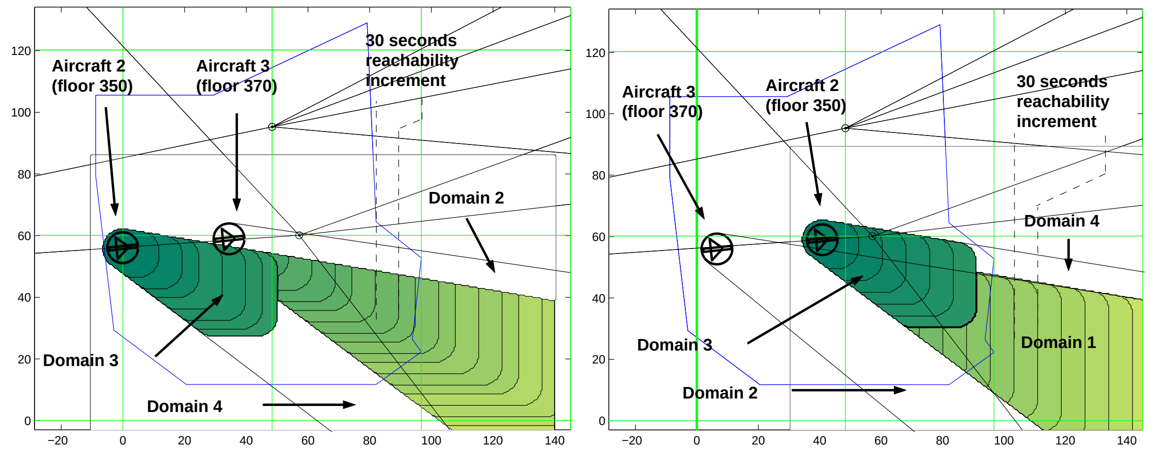


Figure A.10: Results of the conflict avoidance maneuver with reset enabled. Computation realized on a 350 × 350 grid with $T_{\text{climb}} = 3\text{min}$. Aircraft 2 and aircraft 1 are on floor 350. Aircraft 3 is on floor 370 approximately 35 miles ahead of (behind) aircraft 2. The different domains of the diagram have the following interpretation: if aircraft 1 is in Domain 1, there is no way conflict can be avoided by either climbing to 370 or staying on 350. If it is in Domain 2, it should stay there, for climbing will generate a conflict. In Domain 3, there is not enough time to climb, so conflict will occur on floor 350. In Domain 4, conflict can be avoided by climbing. Outside of these four domains, any altitude is safe. Each isoline represents a 30 sec. increment in the time-to-reach function w.r.t. the target (in relative dynamics).

Furthermore, if $\mathcal{S}_h := \{x_h \in X_h : \quad \Theta_{\rho,h}(x_h) < \Theta^R_{\rho,h}(x_h)\}$, *then a converging underapproximation for* θ^{reset} *is given by* $\Theta_{\rho,h}(x_h) = \lim_{n\to+\infty} \Theta^n(x_h)$, *with*

$$\Theta^0_{\rho,h}(x_h) = \sup\{\Theta_{\rho,h}(x_h), \Theta^R_{\rho,h}(x_h)\} \tag{A.23}$$

$$\Theta^{n+1}_{\rho,h}(x_h) = \begin{cases} \rho + \inf_{y_h \in \Gamma_{\rho,h}(x_h)} \Theta^n_{\rho,h}(y_h) & \textit{if} \quad x_h \notin \mathcal{S}_h, \\ \Theta^n_{\rho,h}(x_h) & \textit{else} \end{cases} \tag{A.24}$$

Results obtained for both the hybrid and the reset case are shown in Figures A.9 and A.10.

Appendix B

Single interval scheduling algorithms

The first part of this appendix presents a proof of Carlier's algorithm, adapted from Carlier's original article [49]. It also tries to recreate the numerical experiment in this article. The second part of the appendix presents the proof of the modified Baptiste algorithm used in Chapter 5, Section 5.2.6. This proof is an adaptation of the original proof of Baptiste [12].

B.1 The Carlier algorithm

This Appendix presents a modified version of Carlier's algorithm [49, 51] for solving the feasibility of the optimization program (5.2) for a given Δ, when $n_i = 1$. We were not able to reprove Carlier's original algorithm (we think that the proposed example in the paper has a minor error), thus, the algorithm presented here includes a modification by us. We present this complete algorithm and our proof (which is a slightly modified version of Carlier's) here, as the original algorithm is published in French and may not be accessible to some.

We reproduce the work of Carlier [49, 51] as much as possible, using the original notation of the author, and modifying it when necessary. When not defined, the notation refer to Chapter 5. The problem solved by Carlier is the following.

Let I be a set of jobs. We want to find an ordering[*] $\mathcal{T} = (t_i)_{i \in I}$ *such that*

1. $t_i \geq a_i$*, job i cannot start before time a_i.*
2. $t_i + D \leq b_i$*, job i cannot end after time b_i.*
3. $t_i < t_j \Longrightarrow t_j \geq t_i + D$*: two jobs cannot be executed simultaneously.*

A few definitions and some notation need to be given and will be used in the algorithm below. The algorithm recursively builds an ordering $\mathcal{T}^x$ satisfying the conditions above, where the recursion is run on x which represents time (therefore, x will be proceed from $\min_i a_i$ to $\max_j b_j$).

- $K_x = \{i | a_i + D \leq x\}$ is the set of jobs which can be finished before time x.
- Let $\mathcal{T}^x$ be an ordering of the jobs constructed by the algorithm; we call U_x the set of jobs ordered by $\mathcal{T}^x$: $U_x = \{i | i \text{ ordered by } \mathcal{T}^x\}$.
- $H = K_x - U_{x-D}$ is the set of jobs which can be finished before time x (i.e. from K_x), which have not been ordered by $\mathcal{T}^{x-D}$ (i.e. not in U_{x-D}).
- m is the most urgent job of H: $m = \arg\min_{h \in H} b_h$.
- The delay of an ordering $\mathcal{T}$ (with corresponding set of indices U) is: $y = \max_{u \in U}(t_u + D)$. The delay of $\mathcal{T}$ is the time this ordering takes to complete.
- An ordering $\mathcal{T}$ (with corresponding set of indices U) is said to be *active* iff there does not exist a job w in the complement $\overline{U}$ of U such that $\max_{u \in U}(t_u + D) > b_w - D$. In other words, there does not exist a job w in the complement of U which cannot be processed before its deadline b_w because w cannot start before the completion time (delay) of the jobs in U. Mathematically, it equivalently reads: $\max_{u \in U}(t_u + D) \leq \min_{u \in \overline{U}} b_w - D$.
- An ordering $\mathcal{T}$ (with corresponding set of indices U) is said to be *x-active* iff it is active and has a delay less than x.

[*]In French: “ordonnancement”

- $\{\mathcal{T}^{x-D}+m\}$ is the ordering obtained by adding the most urgent job m to the ordering $\mathcal{T}^{x-D}$, if m is executed as soon as possible. As soon as possible means the job m should start at $t_m = \max\{a_m, y\}$, where $y = \max_{u \text{ in } \mathcal{T}^{x-D}}(t_u+D)$ is the delay of the ordering $\mathcal{T}^{x-D}$.

- $T(x)$ is the set of orderings which are active and which have a delay less than x.

Definition 3. (Carlier) *The operator "preferable to" $\succeq$ is defined the following way: for two sets U and V of elements of I, $U \succeq V$ if:*

1. $p = |U| \geq r = |V|$

2. $b_{u_1} \leq b_{v_1}, \cdots, b_{u_r} \leq b_{v_r}$ *if* $b_{u_1} \leq b_{u_2} \cdots \leq b_{u_p}$ *and* $b_{v_1} \leq b_{v_2} \cdots \leq b_{v_r}$.

Theorem 9 (after Carlier [49, 51]). *$\forall x \in \{\min_{i\in I} a_i, \cdots, \max_{i\in I} b_i\}$, the set $\mathcal{T}^x$ constructed by the algorithm below is x-active and is preferable to any other x-active ordering.*

Algorithm Carlier

Input: N jobs, with release times a_i and deadlines b_i.
Output: An ordering (i.e. schedule) with optimal makespan.

```
1   $\forall x \in \{\min_{i\in I} a_i, \cdots, \min_{i\in I} a_i + D - 1\}$, $\mathcal{T}^x := \emptyset$, $U_x := \emptyset$
2   for $x = \min_{i\in I} a_i + D : \max_{i\in I} b_i$
3       $K_x := \{i | a_i + D \leq x\}$, $H = K_x - U_{x-D}$
        if $H = \emptyset$
4           $\mathcal{T}^x = \mathcal{T}^{x-D}$
        else
5           $m = \arg\min_{h\in H} b_h$
            if $\{\mathcal{T}^{x-D} + m\}$ is active
6               $\mathcal{T}^x = \{\mathcal{T}^{x-D} + m\}$
            else
7               $\mathcal{T}^x = \mathcal{T}^{x-1}$
            end
        end
    end
8   if $U_x \neq I$ no solution else return $\mathcal{T}^x$ end
```

Proof — The proof is done by induction on x. Obviously, $\forall x \in \{\min_{i\in I} a_i, \cdots, \min_{i\in I} a_i + D - 1\}$, the property is true, since $\mathcal{T}^x := \emptyset$ and $U_x := \emptyset$.

Suppose the property is true for all $z \leq x-1$. We want to show that it is true for $\mathcal{T}^x$. There are two cases:

Case 1: $H = \emptyset$.

We know that $K_x \subseteq U_{x-D}$ because $K_x - U_{x-D} = \emptyset$. Since the set of jobs ordered at time $x-D$ is at most the number of jobs that *can* be ordered at time $x-D$, $U_{x-D} \subseteq K_{x-D} \subseteq K_x$, because K_x is monotonically increasing. Combining the above inclusions, we have $K_x = U_{x-D}$. This means that all jobs executable before time x are ordered by U_{x-D} (i.e. by $\mathcal{T}^{x-D}$). Also, $\mathcal{T}^{x-D} \in T(x-D) \subseteq T(x)$, therefore $\mathcal{T}^{x-D} \in T(x)$. This implies that $\mathcal{T}^{x-D}$ is a maximal element of $T(x)$ for the preorder $\succeq$.

Case 2: $H \neq \emptyset$.

H contains at least one element which can be finished before x not ordered by U_{x-D}. Let $\mathcal{T}' = \{\mathcal{T}^{x-D} + m\}$; let $\mathcal{T}$ be any ordering in $T(x)$, and call l the last job which was ordered by $\mathcal{T}$, and $\mathcal{T} - \{l\}$ the ordering obtained by withdrawing job l from $\mathcal{T}$, U the set of jobs ordered by $\mathcal{T}$.

- We first show that if $\mathcal{T}'$ is active, then $\mathcal{T}' \succeq \mathcal{T}$. We use Carlier's [49, 51] lemma on the preorder $\succeq$:

  Lemma: (Carlier [49, 51]) Let C and E be two sets such that $C \succeq E$. Let $u \notin C$ and $v \notin E$. Suppose that $\forall w \in \overline{C}$, $b_u \leq b_w$. Then $\{u\} \cup C \succeq \{v\} \cup E$. $\square$.

  We can apply the previous lemma to the present problem: take $I = K_x$, $C = U_{x-D}$, $E = U - \{l\}$, $u = m$, $v = l$. In the following, the complement of a set $K \subseteq I$ is defined as $\overline{K} = I \backslash K$. Clearly, $u \notin C$, $v \notin E$, $\forall w \in \overline{C}$, $b_u \leq b_w$ because by construction, $m = u$ is the most urgent job. Finally $U_{x-D} \succeq U - \{l\}$ by the induction assumption. Therefore $\mathcal{T}' \succeq \mathcal{T}$, that is, $\mathcal{T}'$ is preferable to any element of $T(x)$. Since $\mathcal{T}'$ is active, it must also be x-active. To see this, we need to prove that the delay of $\mathcal{T}'$ is less than x. Calling y the delay of $\mathcal{T}^{x-D}$, we have $y \leq x - D$. Since $y \leq \min_{w\in\overline{\mathcal{T}^{x-D}}} b_w - D$, because $\mathcal{T}^{x-D}$ is $x-D$ active and $m \in \overline{\mathcal{T}^{x-D}} \cap K_x$, m can be executed before x, i.e. $t_m + D \leq x$. In conclusion, $\mathcal{T}'$ is x-active and preferable to all elements of $T(x)$.

- Second, we show that if $\mathcal{T}'$ is not active, then $\mathcal{T}^x = \mathcal{T}^{x-1}$. We want to show that $\forall \mathcal{T} \in T(x)$, $\mathcal{T}^{x-1} \succeq \mathcal{T}$.
  - If $\max_{u\in\mathcal{T}}(t_u + D) \le x - 1$, this result is immediate by induction.
  - Otherwise $\max_{u\in\mathcal{T}}(t_u + D) = x$. We call l the last job ordered by $\mathcal{T}$ (see Figure B.1). By induction, we know that $\mathcal{T}^{x-D} \succeq \mathcal{T} - \{l\}$. This implies [†] $\overline{\mathcal{T} - \{l\}} \succeq \overline{\mathcal{T}^{x-D}}$.

    Because $\{\mathcal{T}^{x-D} + m\}$ is not active, $\exists \overline{w} \in \overline{\{\mathcal{T}^{x-D} + m\}}$ such that $b_{\overline{w}} - D < x$. We have $\overline{w} \in \mathcal{T}$ since otherwise $\overline{w}$ can only start after x (the delay of $\mathcal{T}$) or $b_{\overline{w}} - D \ge x$, which is a contradiction. Thus, $\overline{w}$ must have been processed before x and $\overline{w} \in K_x$. Since m was the most urgent task added to $\mathcal{T}^{x-D}$, $b_m \le b_{\overline{w}} < x + D$ (that is, both m and $\overline{w}$ are in K_x when m is added).

    Now we have: $m \in \overline{\mathcal{T}^{x-D}}$ and $\overline{w} \in \overline{\mathcal{T}^{x-D}}$. Since $\overline{\mathcal{T} - \{l\}} \succeq \overline{\mathcal{T}^{x-D}}$, $\overline{\mathcal{T} - \{l\}}$ contains at least two jobs i and j, indexed by the order $b_i \le b_j$, such that $b_i \le b_m < x + D$ and $b_i \le b_{\overline{w}} < x + D$. Furthermore, since both i and j are in $\overline{\mathcal{T} - \{l\}}$, one of them must be scheduled after l, say job j, with completion time at least $x + D$. This implies $b_j \ge x + D$, which contradicts the previous inequality. Therefore there cannot be $\mathcal{T} \in T(x)$ of exact delay x.

We have proved that regardless of how $\mathcal{T}^x$ was constructed, it is always x-active, and orders more jobs than any other x-active ordering $\mathcal{T}$. If a solution to the original problem exists (i.e. the problem is feasible), we know that the algorithm will find it, because for any x, it finds one x-active ordering with the maximal number of jobs. It suffices to march x until $\max_{i\in I} b_i$. □

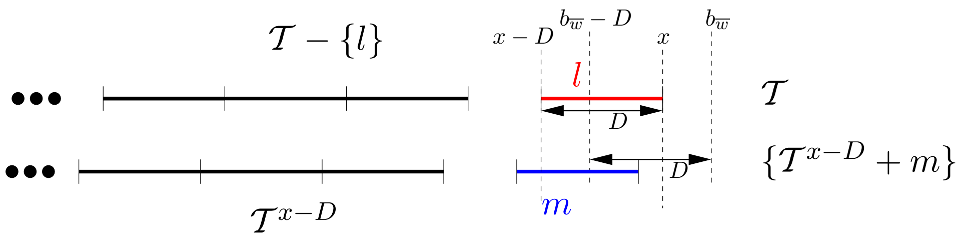


Figure B.1: Case 2 from the proof of Theorem 9, with $\{\mathcal{T}^{x-D} + m\}$ not active. Orderings $\mathcal{T}$ of delay x exactly and $\{\mathcal{T}^{x-D} + m\}$ at iteration x. The last job l of $\mathcal{T}$ is such that $t_l + D = x$.

[†] Carlier also shows $A \succeq B \Rightarrow \overline{B} \succeq \overline{A}$.

Theorem 10. *1. The computational complexity of calculating $\mathcal{T}^x$ at each step is $O(N)$. 2. It is possible to modify the algorithm such that the overall computational complexity is $O(N^3)$.*

Proof —

1. In step 3, the computational complexity of calculating K_x, for all x from $\min_{i \in I} a_i + D$ to $\max_{i \in I} b_i$ is $O(N)$ (since it requires stepping through all a_i). Therefore, in the **for** loop of step 2, it is $O(1)$. The computation of H is at most $O(N)$, if the two sets K_x and U_{x-D} are constructed with increasing indices (i.e. the sets are indexed such that performing $K_x - U_{x-D}$ only requires checking the upper indices of K_x and U_{x-D}). All other operations are $O(1)$ except computing $m = \arg\min_{h \in H} b_h$. The cost of computing $\mathcal{T}^x$ is thus $O(N)$.

2. It suffices to compute the $\mathcal{T}^x$ for $x = a_i + kD$, with $i \in I = \{1, \ldots, N\}$ and $k \in I = \{0, \ldots, N-1\}$. The total number of x is N^2, which gives the algorithm a complexity of $O(N^3)$.

□

Remark:

- An upper bound on the number of $\mathcal{T}^x$ is $\max_{i \in I} b_i - \min_{i \in I} a_i$.
- The complexity can be reduced to $O(N^2 \min(N, D))$ with an additional test on U_x. At each iteration, we add the test $U_x = K_x$. If this is true, $x := \min_{i \in I - K_x} a_i$. The number of such jumps is at most N. Between two jumps, a number ν_i of jobs is scheduled, for a duration D each. $\sum_{i=1}^{N} \nu_i = N$, thus the number of x to step through for these jobs is at most ND. The total number of x to step through is thus $N + ND$. The complexity of the algorithm is thus $O(N^2 \min\{N, D\})$. In another article [50], Carlier shows that it is even possible to reduce the complexity of the algorithm to $O(N \log(N) \min(N, D))$.

We run Carlier's numerical example on the present algorithm, as an illustration of the algorithm. Consider six jobs, with following release times: $a_1 = 0$, $a_2 = 2$, $a_3 = 7$, $a_4 = 9$, $a_5 = 10$, $a_6 = 24$, and the following deadlines: $b_1 = 32$, $b_2 = 35$, $b_3 = 22$, $b_4 = 20$, $b_5 = 23$, $b_6 = 30$. The processing time of each job is $D = 5$. The following array now shows the results obtained on this example, with the algorithm above. For $x = 19$, $\{\mathcal{T}_{14}+3\} = \{1, 4, 3\}$ is not active, according to the definition of active: the delay of this ordering is 19. This does not enable the later scheduling of job number 5: $b_5 = 23 < 19 + D = 24$. Therefore, $\mathcal{T}_{19} = \mathcal{T}_{18}$. Similarly, for $x = 20$, $U_{15} = \{1, 2, 4\}$, $m = 3$, but $\{1, 2, 4 + 3\}$ is not active, because its delay is 20, and $b_5 = 23 < 20 + D = 25$. $\{\mathcal{T}_{15} + m\}$ is not active, therefore $\mathcal{T}_{20} = \mathcal{T}_{19}$. The same is true for $\mathcal{T}_{21}$. For $x = 22$, it becomes active again, and the jobs can be added properly until a full ordering is produced for $x = 32$ (see Figure B.2).

B.2 Proof of the modified Baptiste algorithm

We now provide a proof of Proposition 2 (Chapter 5).

$\boxed{\textbf{Case 1: } r_k \notin [s, e)}$

We have $J_k \notin U_k(s, e)$ by definition of $U_k(s, e)$. We know that $U_k(s, e) \subseteq U_{k-1}(s, e) + \{J_k\}$. Since $J_k \notin U_k(s, e)$, this implies $U_k(s, e) \subseteq U_{k-1}(s, e)$. The reverse inclusion $U_{k-1}(s, e) \subseteq U_k(s, e)$ holds by definition of $U_k(s, e)$, and we therefore have $U_k(s, e) = U_{k-1}(s, e)$. By definition of $C_k(s, e, l)$, this implies that $C_k(s, e, l) = C_{k-1}(s, e, l)$.

$\boxed{\textbf{Case 2: } r_k \in [s, e)}$

We have $J_k \in U_k(s, e)$. We call

$$C' = \min \left\{ C_{k-1}(s, e, l) \, , \min_{\substack{s' \in \Theta \\ 1 \le q \le l-1 \\ \max(r_k, s+\Delta) \le s' \le \min(d_k, e) - \Delta}} \left(C_{k-1}(s, s', l - q) + s' + C_{k-1}(s', e, q - 1) \right) \right\}$$

We want to prove that $C' = C_k(s, e, l)$. We do it in two steps.

Step 1: $C' \ge C_k(s, e, l)$.

- If $C' = C_{k-1}(s, e, l)$, the inclusion $U_{k-1}(s, e) \subseteq U_k(s, e)$ implies $C' = C_{k-1}(s, e, l) \ge C_k(s, e, l)$

- If $C' = C_{k-1}(s, s', l-q) + s' + C_{k-1}(s', e, q-1)$ for some s' and q achieving the min, we separate the jobs into three subsets: X, the $l-q$ first jobs; job J_k; and Y, the set of $q-1$ jobs after s'. We first show that $X \cup Y \cup \{J_k\} \subseteq U_k(s,e)$.

  1. $\forall j \in X$, $J_j \in U_{k-1}(s,s') \subseteq U_k(s,s') \subseteq U_k(s,e)$.
  2. $\forall j \in Y$, $J_j \in U_{k-1}(s',e) \subseteq U_k(s',e) \subseteq U_k(s,e)$.
  3. $J_k \in U_k(s,e)$ by assumption.

  Therefore, $X \cup Y \cup \{J_k\} \subseteq U_k(s,e)$, and $|X| + |Y| + 1 = l - 1 + q - 1 + 1 = l$. From this, we see that $X \cup Y \cup \{J_k\}$ is a particular element of the set among which the min is taken in the definition of C' above. Therefore, $C' \geq C_k(s,e,l)$ which is the min among all elements of this set.

Step 2: $\underline{C' \leq C_k(s,e,l)}$ Call B the instantiation of the l jobs which realizes $C_k(s,e,l)$.

- If $J_k \notin B$, then $B \subseteq U_k(s,e) \backslash \{J_k\} \subseteq U_{k-1}(s,e)$, and $|B| = l$. B is also an instantiation of $C_{k-1}(s,e,l)$, therefore $C_k(s,e,l) \geq C_{k-1}(s,e,l)$. By definition of C', we have $C_{k-1}(s,e,l) \geq C'$. Therefore $C_k(s,e,l) \geq C'$.

- If $J_k \in B$, we first show that $\forall j \in Y$, $r_j > s'$, where s' is the starting time of job J_k.

  Let j be in Y. Assume $r_j \leq s'$. We know that $r_k \leq s'$ by construction. We know that $j \leq k$, because $J_j \in Y$ (and $Y \subseteq B \subseteq U_k(s,e)$; in $U_k(s,e)$, all jobs have index less than k). By assumption, the deadlines have been indexed chronologically, therefore, $d_j \leq d_k$. Call z the completion time of job j. We have $d_j \geq z$. Summarizing all information above, this means that both J_j and J_k can start at s' and finish at z. Therefore switching J_j and J_k is possible and will not change the sum of the starting times of all jobs. The conclusion of this paragraph is that *any job of Y with release time less than s' can be put in X without change of cost.* We can therefore assume that all jobs J_j of Y have a starting time $r_j > s'$

  Let J_j be in Y. Because $J_j \in U_k(s,e)$, $j \leq k$. Since $J_j \in Y$, J_j is scheduled after J_k by the previous paragraph. Therefore, $j \leq k-1$. This implies $J_j \in U_{k-1}(s',e)$. Therefore $Y \subseteq U_{k-1}(s',e)$. Let us call $C(Y)$ the cost of scheduling all jobs of Y ($C(Y)$ is the sum of their starting times). $C(Y) \geq C_{k-1}(s', e, |Y|)$ by definition of $C.(,\cdot,\cdot,\cdot)$. For X, similarly, $J_j \in X \subseteq B \subseteq U_k(s,e)$ implies $r_j \geq s$. Also, $r_j < s'$ (because $r_j \leq s' - \Delta$).

$j \neq k$ because $J_j \in X$, therefore $j \leq k-1$. From this we deduce $X \subseteq U_{k-1}(s, s')$. This implies $C(X) \geq C_{k-1}(s, s', |X|)$ where $C(X)$ is the cost of scheduling the jobs of X.

$$C(X) \geq C_{k-1}(s, s', |X|)$$
$$C(Y) \geq C_{k-1}(s', e, |Y|)$$

Therefore, writing explicitly the contributions of the different terms in B, we have:

$$C_k(s, e, l) = C(X) + s' + C(Y) \geq C_{k-1}(s, s', |X|) + s' + C_{k-1}(s', e, |Y|) \geq C'$$

The last inequality results from the min in the definition of C'.

x	K	H	m	$\{\mathcal{T}_{x-D}+m\}$ active	U_x	$\mathcal{T}_x$
5	1	1	1	yes	1	0
6	1	1	1	yes	1	0
7	1 2	1 2	1	yes	1	0
8	1 2	1 2	1	yes	1	0
9	1 2	1 2	1	yes	1	0
10	1 2	2	2	yes	1 2	0 5
11	1 2	2	2	yes	1 2	0 5
12	1 2 3	2 3	3	yes	1 3	0 7
13	1 2 3	2 3	3	yes	1 3	0 7
14	1 2 3 4	2 3 4	4	yes	1 4	0 9
15	1 2 3 4 5	3 4 5	4	yes	1 2 4	0 5 10
16	1 2 3 4 5	3 4 5	4	yes	1 2 4	0 5 10
17	1 2 3 4 5	2 4 5	4	yes	1 3 4	0 7 12
18	1 2 3 4 5	2 4 5	4	yes	1 3	0 7 12
19	1 2 3 4 5	2 3 5	3	no	1 3 4	0 7 12
20	1 2 3 4 5	3 5	3	no	1 3 4	0 7 12
21	1 2 3 4 5	3 5	3	no	1 3 4	0 7 12
22	1 2 3 4 5	2 5	5	yes	1 3 4 5	0 7 12 17
23	1 2 3 4 5	2 5	5	yes	1 3 4 5	0 7 12 17
24	1 2 3 4 5	2 5	5	yes	1 3 4 5	0 7 12 17
25	1 2 3 4 5	2 5	5	yes	1 3 4 5	0 7 12 17
26	1 2 3 4 5	2 5	5	yes	1 3 4 5	0 7 12 17
27	1 2 3 4 5	2	2	no	1 3 4 5	0 7 12 17
28	1 2 3 4 5	2	2	no	1 3 4 5	0 7 12 17
29	1 2 3 4 5 6	2 6	6	yes	1 3 4 5 6	0 7 12 17 24
30	1 2 3 4 5 6	2 6	6	yes	1 3 4 5 6	0 7 12 17 24
31	1 2 3 4 5 6	2 6	6	yes	1 3 4 5 6	0 7 12 17 24
32	1 2 3 4 5 6	2 6	6	yes	1 3 4 5 6	0 7 12 17 24
33	1 2 3 4 5 6	2 6	6	yes	1 3 4 5 6	0 7 12 17 24
34	1 2 3 4 5 6	2	2	yes	1 3 4 5 6 2	0 7 12 17 24 29

Table B.1: Numerical experiment for the example of Carlier. The result is shown in Figure B.2.

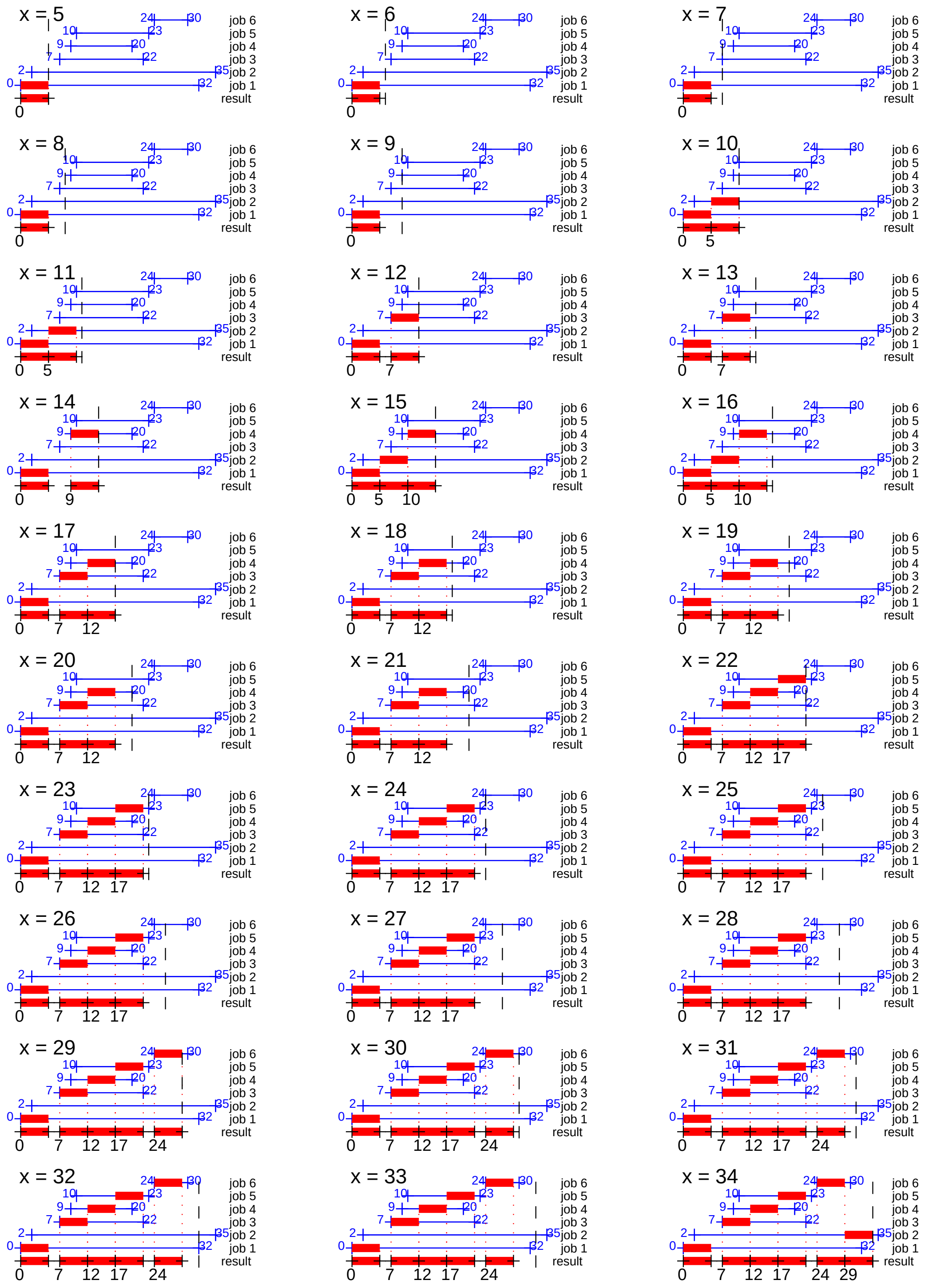


Figure B.2: Successive orderings obtained with the modified Carlier algorithm. A strip on row i means that job i has been scheduled at the time corresponding to the beginning of the strip. Row 0 shows the set of all ordered jobs. At $x = 19$, $\{\mathcal{T}^{14} + m\}$ is not active according to our new definition, therefore, $U_{19} = U_{18}$.

Bibliography

[1] O. M. Aamo and M. Krstic. *Flow Control by Feedback.* Springer-Verlag, 2002.

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

[3] R. Alur, C. Courcoubetis, T.A.Henzinger, P.-H. Ho and X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. In G. Cohen and J.-P. Quadrat, editors, *Proceedings of the 11th International Conference on Analysis and Optimization of Systems: Discrete-event Systems*, number 199 in Lecture Notes in Computer Science, pages 331–351. Springer Verlag, 1994.

[4] K Amonlirdviman, N. Khare, D. Tree, J. D. Axelrod, and C. J. Tomlin. Using mathematical models to understand planar cell polarity. Submitted to *Nature*, Oct. 2003.

[5] R. Ansorge. What does the entropy condition mean in traffic flow theory? *Transportation Research*, 24B(2):133–143, 1990.

[6] J.-P. Aubin. *Viability Theory.* Systems and Control: Foundations and Applications. Birkhäuser, Boston, MA, 1991.

[7] J.-P. Aubin. Systems of Hamilton-Jacobi-Bellman equations: A viability approach. Lecture Notes, `http://viab.dauphine.fr/`, June-July, 2001.

[8] J.-P. Aubin, A. M. Bayen, N. Bonneuil, and P. Saint-Pierre. *Elements of Viability Theory.* To appear (Springer-Verlag), 2005.

[9] J.-P. Aubin and A. Cellina. *Differential inclusions.* Springer-Verlag, New York, NY, 1984.

[10] J.-P. Aubin and H Frankowska. *Set Valued Analysis*. Birkhäuser, Boston, MA, 1990.

[11] B. Bamieh, F. Paganini, and M. A. Daleh. Distributed control of spatially-invariant systems. *IEEE Transactions on Automatic Control*, 47(7):1091–1107, 2002.

[12] P. Baptiste. Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine when processing times are equal. *Journal of Scheduling*, 2:245–252, 1999.

[13] M. Bardi. Some applications of viscosity solutions to optimal control and differential games. In I. Capuzzo-Dolcetta and P. L. Lions, editors, *Viscosity Solutions and Applications*, number 1660 in Lecture Notes in Mathematics, pages 44–97. Springer, 1995.

[14] M. Bardi and I. Capuzzo-Dolcetta. *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman equations*. Birkhäuser, Boston, MA, 1997.

[15] T. Başar and G. J. Olsder. *Dynamic noncooperative game theory*. Mathematics in science and engineering. SIAM, Philadelphia, PA, 1994.

[16] A. M. Bayen, E. Crück, and C. J. Tomlin. Guaranteed overapproximations of unsafe sets for continuous and hybrid systems: solving the Hamilton-Jacobi equation using viability techniques. In C. J. Tomlin and M. Greenstreet, editors, *Hybrid Systems: Computation and Control*, LNCS 2289, pages 90–104. Springer-Verlag, 2002.

[17] A. M. Bayen, P. Grieder, G. Meyer, and C. J. Tomlin. Lagrangian delay predictive model for sector-based air traffic flow. To appear 2004 in the *AIAA Journal of Guidance, Control and Dynamics*.

[18] A. M. Bayen, P. Grieder, H. Sipma, G. Meyer, and C. J. Tomlin. Delay predictive models of the National Airspace System using hybrid control theory. In *2002 American Control Conference*, volume 1, pages 767–772, Anchorage, AK, May 2002.

[19] A. M. Bayen, P. Grieder, and C. J. Tomlin. A control theoretic predictive model for sector-based air traffic flow. In *AIAA Conference on Guidance, Navigation and Control*, Monterey, CA, Aug. 2002. AIAA Paper 2002–5011.

[20] A. M. Bayen, I. Mitchell, M.K. Oishi, and C. J. Tomlin. Reachability analysis and controller synthesis for autopilot design. In preparation for *AIAA Journal of Guidance, Control and Dynamics*, 2004.

[21] A. M. Bayen, R. Raffard, and C. J. Tomlin. Adjoint-based constrained control of Eulerian transportation networks: application to Air Traffic Control. Submitted to the *2004 American Control Conference*.

[22] A. M. Bayen, R. Raffard, and C. J. Tomlin. Eulerian network model of air traffic flow in congested areas. Submitted to the *2004 American Control Conference*.

[23] A. M. Bayen, R. Raffard, and C. J. Tomlin. Network congestion alleviation using adjoint hybrid control: application to highways. Submitted to *Hybrid Systems: Computation and Control 2004*.

[24] A. M. Bayen, S. Santhanam, I. Mitchell, , and C. J. Tomlin. A differential games formulation of alert levels in ETMS data for high altitude traffic. In *AIAA Conference on Guidance, Navigation and Control*, Austin, TX, Aug. 2003. Paper 2003–5341.

[25] A. M. Bayen and C. J. Tomlin. A case study in air traffic management. In D. Al Gobaisi, editor *Encyclopedia of Life Support Systems*, UNESCO-EOLSS Publishers Co. Ltd., Ref: 6:43:28:8. To appear, 2004.

[26] A. M. Bayen and C. J. Tomlin. A construction procedure using characteristics for viscosity solutions of the Hamilton-Jacobi equation. In *Proceedings of the* 40^{th} *IEEE Conference on Decision and Control*, pages 1657–1662, Orlando, Dec. 2001.

[27] A. M. Bayen and C. J. Tomlin. Real-time discrete control law synthesis for hybrid systems using MILP: applications to congested airspaces. In *American Control Conference*, pages 4620–4626, Denver, CO, June 2003.

[28] A. M. Bayen, C. J. Tomlin, J. Zhang, and Y. Ye. An approximation algorithm for scheduling aircraft with holding time. In preparation for the 43^{rd} IEEE Conference on Decision and Control.

[29] A. M. Bayen, C. J. Tomlin, J. Zhang, and Y. Ye. MILP formulation and polynomial-time algorithm for an aircraft scheduling problem. To appear in the 42^{nd} *Conference on Decision and Control*, 2003.

[30] A. Bemporad, F. Borrelli, and M. Morari. Piecewise linear optimal controllers for hybrid systems. In *2000 American Control Conference*, pages 1190–1194, Chicago, IL, June 2000.

[31] A. Bemporad, F. Borrelli, and M. Morari. On the optimal control law for linear discrete time hybrid systems. In C.J. Tomlin and M. Greenstreet, editors, *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science2289, pages 105–119. Springer Verlag, 2002.

[32] P. Bernhard and B. Larrouturou. Etude de la barrière pour un problème de fuite optimale dans le plan. Technical Report RR-1131, INRIA, Sophia Antipolis, (France), Dec. 1989.

[33] D. Bertsimas and S. Stock. The air traffic flow management problem with enroute capacities. *Operations Research*, 46(3):406–422, 1998.

[34] D. Bertsimas and J. N. Tsitsiklis. *Introduction to linear optimization.* Athena Scientific, Belmont, MA, 1997.

[35] T. R. Bewley. Flow control: new challenges for a new renaissance. *Progress in Aerospace Science*, 37:21–58, 2001.

[36] T. R. Bewley, R. Temam, and M. Ziane. A general framework for robust control in fluid mechanics. *Physica, D*, 138:360–392, 2000.

[37] K. Bilimoria. A geometric optimization approach to aircraft conflict resolution. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Denver, CO, Aug. 2000.

[38] K. Bilimoria and H. Q. Lee. Aircraft conflict resolution with an arrival time constraint. In *AIAA Guidance, Navigation, and Control Conference*, Monterey, CA, Aug. 2002. Paper 2002-4444.

[39] K. Bilimoria, B. Sridhar, G. Chatterji, K. Seth, and S.Graabe. FACET: Future ATM Concepts Evaluation Tool. In *Proceedings of the 3rd USA/Europe Air Traffic Management R&D Seminar*, Naples, Italy, June 2001.

[40] M. Boulvin, A. Vande Wouwer, R. Lepore, C. Renotte, and M. Remy. Modeling and control of cement grinding processes. *IEEE Transactions on Control Systems Technology*, 11(5):715–725, 2003.

[41] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2003.

[42] M. S. Branicky, V. S. Borkar, and S. K. Mitter. A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1):31–45, 1998.

[43] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control, Optimization, Estimation and Control.* Taylor and Francis, 1975.

[44] P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre. Optimal Times for Constrained Nonlinear Control Problems without Local Controllability. *Applied Mathematics and Optimization*, 36:21–42, 1997.

[45] P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre. Set-valued numerical analysis for optimal control and differential games. In M. Bardi, T.E.S. Raghavan, and T. Parthasarathy, editors, *Stochastic and Differential Games: Theory and Numerical Methods*, Annals of the International Society of Dynamic Games. Birkhäuser, Boston, MA, 1999.

[46] P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre. Set-valued numerical analysis for optimal control and differential games. In M. Bardi, T.E.S. Raghavan, and T. Parthasarathy, editors, *Stochastic and Differential Games: Theory and Numerical Methods*, Annals of the International Society of Dynamic Games, pages 177–247. Birkhäuser, 1999.

[47] P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre. Numerical schemes for dicontinuous value functions of optimal control. *Set Valued Analysis*, 8(1–2):111–126, 2000.

[48] J. Carlier. Problème à une machine, rep. no. 78.05. Technical report, Institut de Programmation, Université Pierre et Marie Curie, Paris, France, 1978.

[49] J. Carlier. Problèmes d'ordonnancement à durées égales. *QUESTIO*, 5(4):219–229, 1981.

[50] J. Carlier. The one machine sequencing problem. *European Journal of Operational Research*, 11:42–47, 1982.

[51] J. Carlier. *Problèmes d'ordonnancement à contraintes de ressources: algorithmes et complexité*. PhD thesis, Université Pierre et Marie Curie (Paris VI), 1984.

[52] L. I. Cerviño, T. R. Bewley, J. B. Freund, and S. K. Lele. Perturbation and adjoint analyses of flow-acoustic interactions in an unsteady 2d jet. In *Center for Turbulence Research, Proceedings of the Summer Program 2002*, pages 27–40, 2002.

[53] G. Chatterji, B. Sridhar, and D. Kim. Analysis of ETMS data qualiy for traffic flow management decisions. In *Proceedings of the AIAA Conference on Guidance, Navigation and Control*, Austin, TX, Aug. 2003. Paper 2003-5626.

[54] B. Chen, C. N. Potts, and G. J. Woeginger. A review of machine scheduling: Complexity, algorithms and approximability. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, volume 3, pages 21–169. Kluwer Academic Publishers, Boston, MA, 1998.

[55] C. Chen, Z. Jia, and P. Varaiya. Causes and cures of highway congestion. *IEEE Control Systems Magazine*, 21(4):26–33, 2001.

[56] J. Chuzhoy, R. Ostrovsky, and Y. Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. In *IEEE Symposium on Foundations of Computer Science*, pages 348–356, 2001.

[57] F. H. Clarke, Yu. S. Ledyaev, R. J. Stern, and P. R. Wolenski. *Nonsmooth Analysis and Control Theory.* Graduate Texts in Mathematics, Springer, New York, 1998.

[58] M. G. Crandall, L. C. Evans, and P.-L. Lions. Some properties of viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American Mathematical Society*, 282(2):487–502, 1984.

[59] M. G. Crandall and P.-L. Lions. Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American Mathematical Society*, 277(1):1–42, 1983.

[60] E. Crück. *Contrôle et jeux différentiels des systèmes hybrides et impulsionnels : Problèmes de cible et de viabilité*. PhD thesis, Université de Bretagne Occidentale, Brest, France, 2003.

[61] E. Crück, M. Quincampoix, and P. Saint-Pierre. Pursuit-evasion games with hybrid dynamics. In *The fourth international conference "Tools for Mathematical Modelling"*, June 2003.

[62] C. Daganzo. The cell transmission model: a dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research*, 28B(4):269–287, 1994.

[63] C. Daganzo. The cell transmission model, part II: network traffic. *Transportation Research*, 29B(2):79–93, 1995.

[64] C. Daganzo. A finite difference approximation of the kinematic wave model of traffic flow. *Transportation Research*, 29B(4):261–276, 1995.

[65] G. B. Dantzig and D. R. Fulkerson. Minimizing the number of tankers to meet a fixed schedule. *Naval Research Logistic Quarterly*, 1:217–222, 1954.

[66] M. A. Demetriou, A. Paskaleva, O. Vayena, and H. Doumanidis. Scanning actuator guidance scheme in a 1-D thermal manufacturing process. *IEEE Transactions on Control Systems Technology*, 11(5):757–764, 2003.

[67] D. Dugail, E. Feron, and K. Bilimoria. Conflict-free conformance to En-Route flow-rate constraints. In *Proceedings of the AIAA Conference on Guidance, Navigation and Control*, Monterey, CA, Aug. 2002.

[68] D. Dugail, Z.-H. Mao, and E. Feron. Stability of intersecting aircraft flows under centralized and decentralized conflict avoidance rules. In *Proceedings of the AIAA Conference on Guidance, Navigation, and Control*, Montreal, Canada, Aug. 2001.

[69] T. Erlebach and F. Spieksma. Simple algorithms for a weighted interval selection problem. In *Proceedings of the 11th Annual International Symposium on Algorithms and Computation (ISAAC 2000)*, LNCS 1969, pages 228–240. Springer-Verlag, 2000.

[70] L. C. Evans. *Partial Differential Equations.* American Mathematical Society, Providence, Rhode Island, 1998.

[71] L. C. Evans and P. E. Souganidis. Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations. *Indiana University Mathematics Journal*, 33(5):773–797, 1984.

[72] C. A. Floudas. *Nonlinear and mixed-integer programming - fundamentals and applications.* Oxford University Press, Oxford, UK, 1995.

[73] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: a modeling language for mathematical programming.* Boyd and Fraser, Danvers, MA, 1999.

[74] H. Frankowska. Lower Semicontinuous Solutions of Hamilton-Jacobi-Bellman Equations. *SIAM Journal of Control and Optimization*, 31(1):257–272, 1993.

[75] H. Frankowska and M. Quincampoix. Viability kernels of differential inclusions with constraints: Algorithm and applications. *Mathematics of Systems, Estimation and Control*, 1(3):371–388, 1991.

[76] M. R. Garey and D. S. Johnson. *Computers and intractability, a guide to the theory of NP-Completeness.* W.H. Freeman and Company, New York, NY, 1979.

[77] R. Y. Gazit. *Aircraft Surveillance and Collision Avoidance using GPS.* PhD thesis, Department of Aeronautics and Astronautics, Stanford University, 1996.

[78] I. Gertsbakh and H. I. Stern. Minimal resources for fixed and variable job schedules. *Operations Research*, 26:68–85, 1978.

[79] M. B Giles and N. A. Pierce. Analytic adjoint solutions for the quasi-one-dimensional Euler equations. *Journal of Fluid Mechanics*, 426:327–345, 2001.

[80] G. Gomes and R. Horowitz. A study of two onramp metering schemes for congested freeways. In *Proceedings of the American Control Conference*, Denver, CO, June 2003.

[81] S. Gutman and G. Leitmann. Optimal strategies in the neighborhood of collision course. *AIAA Journal*, 14(9):1210–1212, 1976.

[82] C. Hirsch. *Numerical Computation of Internal and External Flows.* John Wiley & and Sons, 1988.

[83] J. Histon and R. J. Hansman. The impact of structure on cognitive complexity in air traffic control. Technical Report ICAT-2002-4, Massachusetts Institute of Technology, June 2002.

[84] B. Hoffmann, J. Krozel, S. Penny, A. Roy, and K. Roth. A cluster analysis to classify days in the National Airspace System. In *Proceedings of the AIAA Conference on Guidance, Navigation and Control*, Austin, TX, Aug. 2003. Paper 2003-5711.

[85] R. Isaacs. *Differential Games*. Dover (John Wiley & and Sons), 1999 (1965).

[86] V. Jain and I. E. Grossmann. Algorithms for hybrid MILP / CP models for a class of optimization problems. *INFORMS Journal on Computing*, 13:258–276, 2001.

[87] A. Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3(3):233–260, 1988.

[88] A. Jameson. Computational aerodynamics for aircraft design. *Science*, 245:361–371, 1989.

[89] A. Jameson. Control theory for optimum design of aerodynamic shapes. In *Proceedings of the 29th IEEE Conference on Decision and Control*, pages 176–179, 1990.

[90] A. Jameson. Analysis and design of numerical schemes for gas dynamics 1: Artificial diffusion, upwind biasing, limiters and their effect on accuracy and multigrid convergence. *International Journal of Computational Fluid Dynamics*, 4:171–218, 1995.

[91] A. Jameson. Analysis and design of numerical schemes for gas dynamics 2: Artificial diffusion and discrete shock structure. *International Journal of Computational Fluid Dynamics*, 4:1–38, 1995.

[92] A. Jameson. A perspective on computational algorithms for aerodynamic analysis and design. *Progress in Aerospace Sciences*, 37:197–243, 2001.

[93] Jeppesen. High Altitude Enroute charts. Feb. 2000, `http://www.jeppesen.com`.

[94] S. Kahne and I. Frolow. Air traffic management: Evolution with technology. *IEEE Control Systems Magazine*, 16(4):12–21, 1996.

[95] J. Krishnan and P. A. Iglesias. Analysis of the signal transduction properties of a module of spatial sensing in eukaryotic chemotaxis. *Bulletin of Mathematical Biology*, 65(1):95–128, 2003.

[96] J. Kuchar and L. Yang. A review of conflict detection and resolution modeling methods. *IEEE Transactions on Intelligent Transportation Systems*, 1(4):179–189, 2000.

[97] R. Lachner. *Echtzeitsynthese optimaler Strategien für Differentialspiele schneller Dynamik mit Anwendungen bei der Kollisionvermeidung.* PhD thesis, Technische Universität Clausthal, Germany, 2002.

[98] G. Leitmann. A simple differential game. *Journal of Optimization Theory and Applications*, 2(4):220–225, 1968.

[99] K. Leoviriyakit and A. Jameson. Aerodynamic shape optimization of wings including planform variations. In *41st AIAA Aerospace Sciences Meeting and Exhibit.* Paper 2003-0210, 2003.

[100] J. Lewin. *Differential games : theory and methods for solving game problems with singular surfaces.* Springer-Verlag, New York, NY, 1994.

[101] H. R. Lewis and C. H. Papadimitriou. *Elements of the theory of computation.* Prentice Hall, Upper Saddle River, NJ, 1982.

[102] M. J. Lighthill and G. B. Whitham. On kinematic waves. II. A theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London*, 229(1178):317–345, 1956.

[103] J. C. Lin and D. K. Gifford. VisualFlight: The air traffic control data analysis system. Technical report, Massachusetts Institute of Technology, May 2002.

[104] X. Litrico. Robust IMC flow control of SIMO dam-river open-channel systems. *IEEE Transactions on Control Systems Technology*, 10(5):432–437, 2002.

[105] J. Lygeros. On the relation of reachability to minimum cost optimal control. In *Proceedings of the IEEE Conference on Decision and Control*, pages 1910–1915, Las Vegas, Nevada, USA, Dec. 10-13 2002.

[106] J. Lygeros. Reachability, viability and invariance: An approach based on minimum cost optimal control. Technical Report CUED/F-INFENG/TR.444, Department of Engineering, University of Cambridge, 2002.

[107] Z. H. Mao, E. Feron, and K. Bilimoria. Stability of intersecting aircraft flows under decentralized conflict avoidance rules. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Denver, CO, Aug. 2000.

[108] I. M. Y. Mareels, E. Weyer, and S. K. Ooi. Irrigation networks: A systems engineering approach. In *Proceedings of the 4th International Conference on Control and Automation*, Montreal, CA, June 2003.

[109] A. Melikyan. *Generalized characteristics of first order PDEs : applications in optimal control and differential games.* Birkhäuser, Boston, MA, 2002.

[110] P. K. Menon, G. Sweriduk, and K. Bilimoria. A new approach for modeling, analysis and control of air traffic flow. In *Proceedings of the AIAA Conference on Guidance, Navigation and Control*, Monterey, CA, Aug. 2002.

[111] P. K. Menon, G. D. Sweriduk, T. Lam, V. H. L. Cheng, and K. Bilimoria. Air traffic flow modeling, analysis and control. In *Proceedings of the AIAA Conference on Guidance, Navigation and Control*, Austin, TX, Aug. 2003. Paper 2003-5712.

[112] A. W. Merz. The game of two identical cars. *Journal of Optimization Theory and Applications*, 9(5):324–343, 1972.

[113] P. Mishra and G. Pappas. Flying hot potatoes. In *2002 American Control Conference*, Anchorage, AK, May 2002.

[114] I. Mitchell. Games of two identical vehicles. Technical Report SUDAAR 740, Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, July 2001.

[115] I. Mitchell. *Application of Level Set Methods to Control and Reachability Problems in Continuous and Hybrid Systems.* PhD thesis, Scientific Computing and Computational Mathematics, Stanford University, 2002.

[116] I. Mitchell, A. M. Bayen, and C. J. Tomlin. Computing reachable sets for continuous dynamic games using level set methods. Submitted Jan. 2002 to the *IEEE Transactions on Automatic Control*.

[117] I. Mitchell, A. M. Bayen, and C. J. Tomlin. Validating a Hamilton-Jacobi approximation to hybrid system reachable sets. In M.D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, LNCS 2034, pages 418–432. Springer-Verlag, 2001.

[118] I. Mitchell and C. Tomlin. Level set methods for computation in hybrid systems. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control*, number 1790 in Lecture Notes in Computer Science, pages 310–323. Springer Verlag, 2000.

[119] K. T. Mueller, D. R. Schleicher, and K. Bilimoria. Conflict detection and resolution with traffic flow constraints. In *AIAA Guidance, Navigation, and Control Conference*, Monterey, CA, Aug. 2002. Paper 2002-4445.

[120] L. Munoz, X. Sun, R. Horowitz, and L. Alvarez. Traffic density estimation with the cell transmission model. In *Proceedings of the American Control Conference*, Denver, CO, June 2003.

[121] G. F. Newell. A simplified theory of kinematic waves in highway traffic, part I: general theory. *Transportation Research*, 27B(4):281–287, 1993.

[122] A. Nilim, L. El-Ghaoui, V. Duong, and M. Hansen. Trajectory-based air traffic management (TB-ATM) under weather uncertainty. In *Proceedings of the 4rd USA/Europe Air Traffic Management R&D Seminar*, Santa Fe, NM, December 2001.

[123] M. S. Nolan. *Fundamentals of Air Traffic Control, 3rd Edition.* Brooks/Cole Publishing, Pacific Grove, CA, 1999.

[124] M. Oishi, C. J. Tomlin, V. Gopal, and D. Godbole. Addressing multiobjective control: Safety and performance through constrained optimization. In M. D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science 2034, pages 459–472. Springer Verlag, 2001.

[125] M. K. Oishi, I. Mitchell, A. M. Bayen, and C. J. Tomlin. System verification: Application to user interface design. Submitted 2003 to the *IEEE Transactions on Control Systems Technology.*

[126] M. K. Oishi, I. Mitchell, A. M. Bayen, C. J. Tomlin, and A. Degani. Hybrid verification of an interface for an automatic landing. In *Proceedings of the* 41^{th} *IEEE Conference on Decision and Control*, pages 1607–1613, Las Vegas, Dec. 2002.

[127] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces.* Springer Verlag, New York, NY, 2002.

[128] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[129] M. Pachter and T Miloh. The geometric approach to the construction of the barrier surface in differential games. *Computers and Mathematics with Applications*, 13(1-3):47–67, 1987.

[130] L. Pallottino, A. Bicchi, and E. Feron. Mixed Integer Programming for aircraft conflict resolution. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Montreal, Canada, Aug. 2001.

[131] C. G. Panayiotou and C. G. Cassandras. A sample path approach for solving the ground holding policy problem in air traffic control. *IEEE Transactions on Control Systems Technology*, 9(3):510–523, 2001.

[132] N. Petit. *Systèmes à retard. Platitude en génie des procédés et contrôle de certaines équations des ondes.* PhD thesis, Ecole Nationale Supérieure des Mines de Paris, 2000.

[133] S. Resmerita, M. Heymann, and G. Meyer. A framework for conflict resolution in air traffic management. To appear in the *Proceedings of the* 42^{d} *Conference on Decision and Control*, 2003.

[134] A. Richards, J. Bellingham, M. Tillerson, and J. How. Co-ordination and control of multiple uavs. In *AIAA Conference on Guidance, Navigation and Control*, Monterey, CA, Aug. 2002.

[135] A. Richards and J. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *2002 American Control Conference*, Anchorage, AK, May 2002.

[136] A. Richards, J. How, and E. Feron. Plume avoidance maneuver planning using mixed integer linear programming. In *AIAA Conference on Guidance, Navigation and Control*, Montreal, Canada, Aug. 2001.

[137] A. Richards, T. Schouwenaars, J. How, and E. Feron. Spacecraft trajectory planning with collision and plume avoidance using mixed-integer linear programming. *AIAA Journal of Guidance, Control and Dynamics*, 25(4):755–764, 2002.

[138] P. I. Richards. Shock waves on the highway. *Operations Research*, 4(1):42–51, 1956.

[139] P. Saint-Pierre. Approximation of the viability kernel. *Applied Mathematics and Optimization*, 29:187–209, 1994.

[140] P. Saint-Pierre. Viability, optimality and stability of dynamical systems and estimation of convergence of numerical schemes. *Pliska Studia Mathematica Bulgarica*, 12:213–226, 1998.

[141] P. Saint-Pierre. Approche ensembliste des systèmes dynamiques, regards qualitatifs et quantitatifs. *Matapli, Société de Mathématiques Appliquées et Industrielles*, 66, 2001.

[142] J. A. Sethian. *Level Set Methods and Fast Marching Methods.* Cambridge University Press, New York, NY, 1999.

[143] R. D. Shaver. The growing congestion in the national airspace system (NAS) : How do we measure it? Are current plans sufficient to constrain its growth? If not, what else can we do? *Air Traffic Control Quarterly*, 10(3):169–195, 2002.

[144] R. F. Stengel. *Optimal Control and Estimation.* Dover, New York, NY, 1994.

[145] W. A. Strauss. *Partial Differential Equations.* John Wiley & and Sons, New York, NY, 1992.

[146] R. S. Teo, J. S. Jang, and C. J. Tomlin. Tests performed on Dragonfly Aircraft, NASA Ames, Sep. 2002.

[147] C. J. Tomlin. *Hybrid Control of Air Traffic Management Systems.* PhD thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1998.

[148] C. J. Tomlin, S. P. Boyd, I. Mitchell, A. M. Bayen, M. Johannson, and L. Xiao. Computational tools for the verification of hybrid systems. In T. Samad and G. Balas, editors, *Software-Enabled Control*. John Wiley. To appear, 2004.

[149] C. J. Tomlin, J. Lygeros, and S. Sastry. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–970, 2000.

[150] C. J. Tomlin, I. Mitchell, A. M. Bayen, and M. K. Oishi. Computational techniques for the verification and control of hybrid systems. *Proceedings of the IEEE*, 91(7):986–1001, July 2003.

[151] C. J. Tomlin, I. Mitchell, and R. Ghosh. Safety verification of conflict resolution maneuvers. *IEEE Transactions on Intelligent Transportation Systems*, 2(2):110–120, 2001.

[152] C. J. Tomlin, G. J. Pappas, and S. Sastry. Conflict resolution for air traffic management: A study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, 1998.

[153] V. V. Vazirani. *Approximation Algorithms.* Springer Verlag, Berlin, Germany, 2001.

[154] Y. Wang, M. Papageorgiou, and A. Messmer. Motorway traffic state estimation based on extended Kalman filter. In *Proceedings of the 2003 European Control Conference*, Cambridge, U.K., Sep. 2003.

[155] H. P. Williams and S. C. Brailsford. Computational logic and integer programming. In J. Beasley, editor, *Advances in Linear and Integer Programming*, pages 249–281. Oxford University Press, Oxford, UK, 1996.

[156] J. B. Yerashunas, J. A. De Abreu-Garcia, and T. T. Hartley. Control of lateral motion in moving webs. *IEEE Transactions on Control Systems Technology*, 11(5):684–693, 2003.

[157] `http://cherokee.stanford.edu/~bayen`.