Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Master Degree in Mathematics

A mathematical programming model for air traffic flow management with dynamic selection of the airspace configuration

Supervisor
Prof. Luigi De Giovanni

Candidate
Luca Zanardelli
Matriculation number 1203113

11 December 2020
Academic Year 2019/2020

Abstract

In this thesis we present a new integer linear programming model for air traffic flow management. The goal of the model is to minimize the cost of flight delays. The introduction provides an overview of the air traffic flow management problem. The problem is addressed through a combination of actions such as ground-holding, airborne-holding, speed control and choice of the most appropriate configuration of the airspace. The configuration may be changed over time, so that the last action implies dynamism of the airspace configuration and allows controllers to monitor the flow of traffic more effectively. Nevertheless, to the best of our knowledge, the model we present in this thesis is the first one that exploits the use of dynamic airspace configurations. The new model is based on Integer Linear Programming formulation and it is compared to one state-of-the-art formulation using a single fixed configuration. Different model parameter settings are analysed, showing significant improvement in terms of reduced delays and related costs. Furthermore, theoretical considerations regarding the study of the linear relaxation of the new model are described. Finally, implementation details are provided.

Contents

Chapter 1

Introduction

1.1 Motivations

Aviation is one of the most global industries because it connects people, cultures and businesses across continents. In fact it provides the only rapid worldwide transportation network, making it essential for global business. Aviation generates economic growth, creates jobs and is decisive for international trade and tourism. According to recent estimates by the Air Transport Action Group (ATAG) [**6**], the total economic impact (direct, indirect, induced and tourism catalytic) of the European aviation industry has reached USD 823 billion. The aviation industry also supported a total of 12.2 million jobs in Europe. It provided 2.6 million direct jobs. According to the estimate of the ACI-Europe study, European airports contributed directly to the employment of 1.7 million people, earning a total of € 68.5 billion in 2013. In addition to jobs and income, these airports directly contributed a total of € 101.6 billion to national GDP. This is approximately 0.6% of the total GDP of Europe in 2013. In addition, indirect, induced and catalytic economic impacts (including tourism, trade, investments, etc.) are taken into account, which supported approximately 4.1% (€ 647.5 billion) of the total European GDP and 12.3 million jobs, with an annual income of € 356 billion. According to long-term traffic forecasts of the International Civil Aviation Organization (ICAO) [**6**], dating back to before the COVID-19 pandemic, passenger and freight traffic in Europe is expected to grow by 3.3% and 2.5% per year respectively until 2045.

As reported in [**13**], in 2019, traffic growth was 0.9%, with a total of over 11.1 million flights. The punctuality of airline arrivals improved over the previous year to 78%. All-causes delay decreased by 11% with a reduction in both primary and reactionary delays [**13**]. Airlines were better prepared with spare planes and crews, and there were fewer interruptions to the network. There were over 1.4 million flights delayed by ATFM regulation in 2019, a 5% increase compared to 2018. About 40% of these flights were delayed by more than 15 minutes, a decrease of 4% compared to 2018. Average daily

ATFM delay in 2019 decreased by 6% compared to 2018. The en-route ATFM delay decreased by 8% and the airport ATFM delay increased by about 2% compared to 2018. The main reasons for en-route ATFM delay in 2019 were en-route ATC capacity (32%), en-route ATC staffing (17%) and en-route weather (15%). Airport weather conditions (12%) and airport capacity (8%) were the main causes of delays attributed to airports. Direct emissions from aviation account for around 3% of the EU's total greenhouse gas emissions and over 2% of global emissions. If global aviation were a country, it would rank in the top 10 emitters. Someone flying from Paris to New York and back generates approximately the same level of emissions as an average person in the EU by heating their home for a full year [**13**]. In 2020, annual global emissions from international aviation are about 70% higher than in 2005. The International Civil Aviation Organization (ICAO) predicts that, in the absence of additional measures by 2050, they could grow by over 300% [**13**]. Air traffic flow management is essential. It allows to ensure and maintain optimal air traffic management, limiting delays, improving the punctuality and efficiency of aircraft carriers, keeping the management of airports and airspace sectors always within the limits of declared capacity. An intervention to regularize the amount of air traffic will be implemented when necessary to ensure that this amount of traffic never exceeds the structural capacity of an airport or sector, furthermore it allows to exploit these capabilities in the most efficient way.

1.2 Structure of the thesis and main contributions

Chapter 1: Introduction In the first chapter we will give an overview of European air traffic, and its impact on economic and environmental aspects. We will also provide the description of the contents of each chapter of the thesis.

Chapter 2: The ATFM problem In the first part of this chapter we will illustrate how operations are conducted to manage the flow of air traffic. In the second part we will more formally introduce the Air Traffic Flow Management (ATFM) problem within the Air Traffic Management (ATM) scope. In particular, we will introduce ground holding, airborne holding, speed control and re-rounting policies, as well as the opportunity of dynamically changing the capacity configuration of the airspace, which has been recently implemented in some control centres as a strategy to adapt control resources to air traffic and obtain improved air traffic flow.

Chapter 3: Linear programming and computational tools In the first part of this chapter we will present the fundamental mathematical tool in carrying out this work: Integer Linear Programming (ILP). We will describe two methods for solving linear integer programming problems. In the second part of the chapter, we will describe the software we used to implement mathematical model and solve them

on test instances.

Chapter 4: State of the art In the first part of this chapter we will illustrate some existing ILP formulations for ATFM, starting from approaches dating back to 1992 up to more recent years. In the second part, we will describe in detail the model from literature that underlies our work.

Chapter 5: A model for ATFM with dynamic configurations This chapter describes the main original contribution of the thesis. We will present a new approach to the air traffic flow management problem, using integer linear programming, to include dynamic airspace configuration as a policy to improve network efficiency. In particular, we will describe in detail new capacity and configuration constraints.

Chapter 6: Model implementation and instance generation The thesis presents a computational study and, in this chapter, we will illustrate the procedure we used to generate the instances to be tested using optimization software to implement and solve ILP models.

Chapter 7: Computational results In this chapter we report the results on three different classes of instances of the air traffic flow management problem. We will provide an analysis of the improvement that can be obtained with different parameter settings and some results on computational efficiency.

Chapter 8: Conclusions In this last chapter we will comment on the results both from a computational and theoretical point of view, underlining the improvements that can be made through this new approach to ATFM.

Appendix: In the appendix we report the AMPL code of the model and the data related to a sample instance.

Chapter 2

The air traffic flow management problem

In this chapter we will describe the planning of operations in the context of air traffic flow management. We will also illustrate the problem in detail, highlighting the operational contest and the most common strategies for solving it.

2.1 Operations management in air traffic control

Air Traffic Flow Management (ATFM) is a service established with the aim of contributing to a safe, orderly and rapid flow of air traffic by ensuring that the Air Traffic Controllers (ATC) capacity is used to the maximum extent possible and that the traffic volume is compatible with the declared capacities by the appropriate Air Traffic Services (ATS) authority. Its purpose is therefore to optimize the flow of air traffic based on air traffic control capability, allowing airlines to operate safe and efficient flights. The Network Manager Operations Center (NMOC) constantly checks the balance between airspace capacity and traffic load. The Air Traffic Flow and Capacity Management (ATFCM) activities are divided into three phases [**10**]:

Strategic phase It starts about one year before the flight takes place and it ends one week before real time operations. During this phase, the NMOC helps the Air Navigation Service Providers (ANSPs) to predict what capacity they will need to provide in each of their air traffic control centres. This also includes avoiding imbalances between capacity and demand for events taking place a week or more in the future;

Pre-tactical phase It starts few days before real time operations. The task of the NMOC staff is to coordinate the definition of a daily plan aimed at optimising the overall ATM network performance and minimising delay and cost, and inform

operational partners about the ATFCM measures that will be in force in European airspace on the following day via the publication of the agreed plan for the day of operations;

Tactical phase It takes place on the day of operations. The staff checks and updates the daily plan made the day before based on current reality. The staff continues working on capacity optimisation according to real time traffic demand, and where aircraft are affected by a regulation, offers alternative solutions to minimise delays. Flights taking place on that day receive the benefit of the flow management service, which includes, among other things, the allocation of individual aircraft departure slots, re-routings to avoid bottlenecks and alternative flight profiles in an attempt to maximise flight efficiency and make the best use of the available capacity.

2.2 The problem and some solution approaches

ATM manages the flight paths and the relations between them. The flight path provides information about the airport of departure, the airport of arrival, the sectors crossed, the time of departure, arrival and entry in each sector. We must specify that the flight path is defined by a 4D space-time trajectory. We will model, for simplicity, the trajectories of our problem as 3D space-time trajectories (we don't consider the flight level dimension).

2.2.1 Cost of delay

The main objective of ATM is to avoid delays, which can be seen as deviations from the temporal component of "ideal" trajectory. The most relevant time deviations are those related to the scheduled departure and arrival, as they are those that involve the main costs of delay. Summing up, regarding the relevant of delays, it is not important whether or not there is a delay in the intermediate phases between take-off and landing, but only if the flight leaves or arrives on time. Hence, the delay costs are mainly related to the delays on the ground at the departure and flight delays upon arrival. The second type of delay is usually more expensive, as it implies extra fuel consumption.

2.2.2 Capacity and airspace configurations

One of the main problems causing delays is the limited capacity of the airports and sectors. The term capacity means the ability to provide Air Navigation Services (ANS) with a certain volume of air traffic, while maintaining a high level of safety. This term is related to the concept of sector. For the purpose of our study, we consider the airspace model based on the concepts of elementary sector and collapsed sector. An elementary sector is a 3D portion of the airspace. Figure 2.1 shows three elementary sectors in the area north-east of Italy. A collapsed sector is the union of one or more elementary

Figure 2.1: Three elementary sectors [**4**].

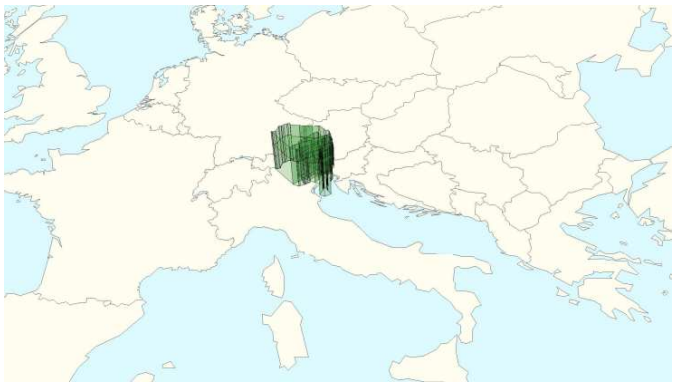


Figure 2.2: A collapsed sector [**4**].

sectors that form a 3D connected portion of the airspace. Figure 2.2 shows an example of a collapsed sector. An airspace configuration is a partition of elementary sectors into subsets corresponding to collapsed sectors. Based on the same elementary sectors, different configurations can be achieved.

The idea is that there are generally not enough controllers to monitor traffic on every elementary sector. We will therefore consider that the controllers monitor the traffic on the collapsed sectors and therefore that the term capacity will be linked to that of the collapsed sector. The capacity of a collapsed sector is the maximum number of flights that the collapsed sector controllers can monitor without exceeding a default maximum workload. The same definition can be given for airports capacity. The capacity of a sector depends, among other things, on its size and geometry but the capacity is dynamic and changes over time to allow ATC to operate in safe conditions.

2.2.3 Control strategies

In general, two main strategies are used to avoid air traffic congestion. The first is commonly known as ground holding. The idea of ground holding is to anticipate the delay on the ground at the departure airport. The second strategy is known as airborne holding. It consists in allowing a flight to hold on airborne before landing at its arrival airport. Proceeding in this way implies additional fuel consumption, with both economic and environmental consequences. An optimal balance between ground and air holding is a key element in ATFM. Another strategy is known as re-routing. The idea of the change of route is to divert a flight on a different route from the one planned, keeping the departure and arrival airports, in order to possibly improve the general state of air traffic flows. For instance, a flight that is scheduled to pass through a congested sector may be re-routed to not occupy this sector. Speed control is a strategy that allows a flight to arrive at a specified point before or after expected congestion. In this thesis we will propose a mathematical model to describe the problem just presented. In this model we will consider the possibility of avoiding congestion by means of ground holding, airborne holding, speed control and the use of dynamic airspace configurations.

Chapter 3

Linear programming and computational tools

The problem of air traffic flow management has been addressed with various approaches. One of them, which is where we are going to tackle it, is integer linear programming. In the first section of this chapter we will provide the main ideas on the subject and illustrate two methods for solving integer linear programs. In the second section we will describe the optimization software that will make possible the practical development of our work.

3.1 Integer linear programming

For the development of this section we mainly referred to [**11**] and [**17**]. A Linear Program (LP) is a problem of the form

$$\begin{aligned} \max \quad & c^\top x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{3.1}$$

where $c, x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. The problem consists in finding a point $\overline{x}$ which maximizes $c^\top x$, which is called the objective function, and which belongs to the set $\{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$, which is called feasible region. A point belonging to this set is called an feasible point. Any point $\overline{x}$ which belongs to the feasible region and maximizes the objective function is a solution of the LP problem (3.1). Geometrically, the feasible region is a convex polyhedron. A linear function is a convex function, which implies that every local minimum is a global minimum; similarly, a linear function is a concave function, which implies that every local maximum is a global maximum. A linear program does not necessarily admit a solution. In fact, two other cases are possible: if

the constraints of a linear program are mutually contradictory, there are no points that satisfy all the constraints and thus the feasible region is the empty set. In this case the problem has no solution and is said to be infeasible. In case the problem may be improved indefinitely without violating the constraints and bounds, it is said to be unbounded. The simplex algorithm is the classical method to solve the optimization problem of linear programming. This algorithm works very well for problems of the form (3.1), where variables can take any real value. Let us now consider the case in which variables can only take integer values.

A pure Integer Linear Program (ILP) is a problem of the form

$$\begin{aligned} \max \quad & c^\top x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \quad \text{integral} \end{aligned} \tag{3.2}$$

where, as before, $c, x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. The feasible region in this case is given by the set $S = \{x \in \mathbb{Z}^n : Ax \leq b,\, x \geq 0\}$. The linear relaxation of S is the set $S' = \{x \in \mathbb{R}^n : Ax \leq b,\, x \geq 0\}$. The linear programming relaxation of (3.2) is the linear program $\max\{c^\top x : x \in S'\}$. Problems in which only a subset of variables is bound to be integral are called mixed integer linear programs (MILP). In our work, we will consider particular pure integer linear programs, in which the variables are restricted to take the value 0 or 1. There is a substantial difference in terms of computational complexity between LP and ILP.

In this section, we will discuss two methods that have proved satisfactory for solving integer programs. These two approaches are based on simple ideas but are the heart of the software dedicated to the integer programming.

3.1.1 The branch-and-bound method

The branch-and-bound method is not a solution technique restricted to integer programming problems. It is based on the concept that the total set of feasible solutions can be divided into smaller subsets of solutions. These smaller subsets can then be evaluated until the best solution is found. When this method is applied to an integer programming problem, it is used in combination with the normal noninteger solution method.

Let us take a simple example to illustrate this method. Consider the following problem:

$$\begin{aligned} \max \quad & z = 5.5x_1 + 2.1x_2 \\ \text{subject to} \quad & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 17 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \text{ integer.} \end{aligned}$$

The solution of the integer linear programming relaxation is $x_1 = 1.3, x_2 = 3.3$ with objective value 14.08. Thus 14.08 is an upper bound on the optimal solution of the problem. Branching on variable x_1, we create two integer programs. The linear programming relaxation of the one with the additional constraint $x_1 \leq 1$ has solution $x_1 = 1, x_2 = 3$ with value 11.8, ad it is consequently pruned by integrality. Hence 11.8 is a lower bound on the value of an optimal solution of the integer program. The linear programming relaxation of the subproblem with the additional constraint $x_1 \geq 2$ has solution $x_1 = 2, x_2 = 0.5$ and objective value 12.05. These steps are summarized in the enumeration tree shown in Figure 3.1.

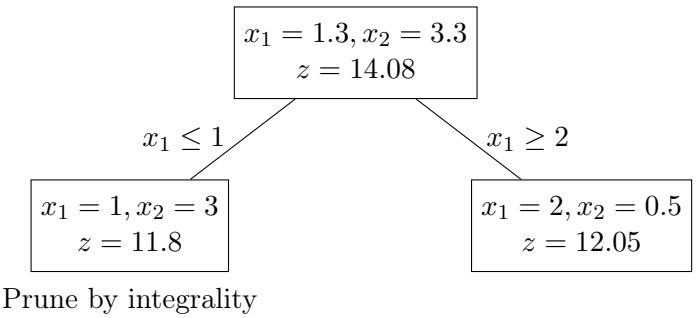


Figure 3.1: Branching on variable x_1

Note that the value of x_2 is not integer, so this solution is not feasible to the integer program. Since its objective function is higher than 11.8 (the value of the best integer solution found so far), we need to continue the search on the right branch of the tree. Therefore we branch on variable x_2. We create two new integer programs, the first with the additional constraint $x_2 \leq 0$, the second with $x_2 \geq 1$. The linear programming relaxation of the first one has solution $x_1 = 2.125, x_2 = 0$ with value 11.6875. Since this value is smaller than the best lower bound 11.8, the corresponding node of the enumeration tree is pruned by bound. The linear programming relaxation of the second one is infeasible, so this problem is pruned by infeasibility. The enumeration is complete and the optimal solution is $x_1 = 1, x_2 = 3$ with value 11.8. The complete enumeration tree is shown in Figure 3.2.

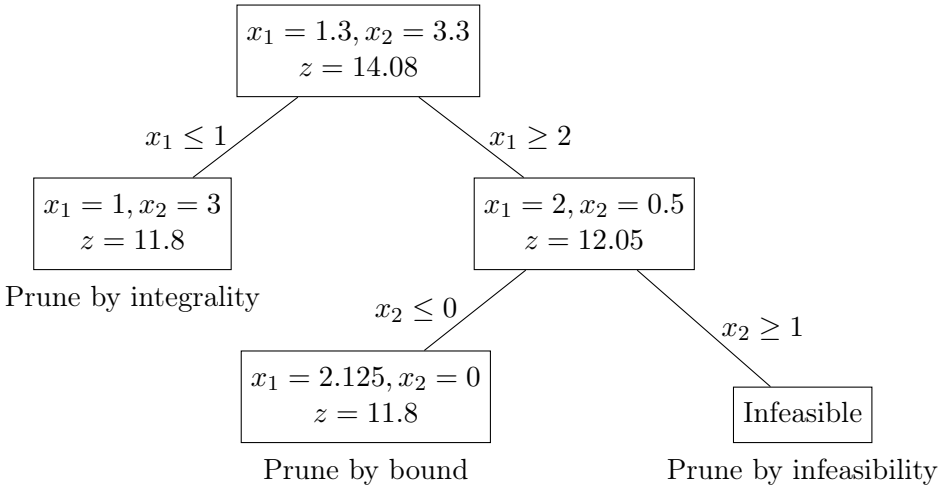


Figure 3.2: Example of a branch-and-bound tree

In summary, the branch-and-bound method works by alternating two phases: in the branching phase a feasible region is split into two feasible subregions, and in the bounding phase a linear program is solved using the simplex method, thus finding an upper bound for the region considered.

3.1.2 The cutting plane method

The basic concept of the cutting plane method is to cut off parts of the feasible region of the linear programming relaxation, so that the optimal integer solution becomes an extreme point and therefore can be found by the simplex method or by other methods for solving linear programs. Many types of cuts have been developed. We now briefly present Gomory's cuts. In general, if nonnegative integer variables $x_1, \ldots, x_n$ satisfy the equation

$$\sum_{j=1}^{n} a_j x_j = a_0$$

where $a_0 \notin \mathbb{Z}$, the Gomory fractional cut is

$$\sum_{j=1}^{n} (a_j - \lfloor a_j \rfloor) x_j \geq a_0 - \lfloor a_0 \rfloor. \tag{3.3}$$

This inequality is satisfied by any $x \in \mathbb{Z}_+^n$ satisfying the equation $\sum_{j=1}^n a_j x_j = a_0$ because $\sum_{j=1}^n a_j x_j = a_0$ implies $\sum_{j=1}^n (a_j - \lfloor a_j \rfloor) x_j = a_0 - \lfloor a_0 \rfloor + k$ for some integer k, also $k \geq 0$ since the the left-hand side is nonnegative.

We illustrate the method with an example. Let us consider the same linear program,

which is:

$$\begin{aligned} \max \quad & z = 5.5x_1 + 2.1x_2 \\ \text{subject to} \quad & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 17 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \text{ integer.} \end{aligned} \tag{3.4}$$

We first introduce slack variables x_3 and x_4 to turn inequality constraints into equalities. The problem becomes to maximize z subject to

$$\begin{array}{rrrrrcl} z & -5.5x_1 & -2.1x_2 & & & = & 0 \\ & -x_1 & +x_2 & +x_3 & & = & 2 \\ & 8x_1 & +2x_2 & & x_4 & = & 17 \\ & x_1, x_2, x_3, x_4 \geq 0 \text{ integer.} & & & & & \end{array}$$

Note that x_3 and x_4 can be constrained to be integer since the data in the constraints of the problem are all integers.

Solving the linear programming relaxation using, for example, the simplex method, we obtain the optimal tableau:

$$\begin{array}{rrrrcl} z & & +0.58x_3 & +0.76x_4 & = & 14.08 \\ & x_2 & +0.8x_3 & +0.1x_4 & = & 3.3 \\ x_1 & & -0.2x_3 & +0.1x_4 & = & 1.3 \\ & x_1, x_2, x_3, x_4 \geq 0. & & & & \end{array}$$

Its basic solution is $x_1 = 1.3, x_2 = 3.3, x_3 = x_4 = 0$ with objective value $z = 14.08$. Since the values of x_1 and x_2 are fractional, this is not a solution of (3.4). We can generate a cut from the constraint $x_2 + 0.8x_3 + 0.1x_4 = 3.3$ in the above tableau according to (3.3). We get:

$$0.8x_3 + 0.1x_4 \geq 0.3,$$

but since $x_3 = 2 + x_1 - x_2$ and $x_4 = 17 - 8x_1 - 2x_2$, we can express the Gomory fractional cut we just found in terms of x_1 and x_2. This yields $x_2 \leq 3$.

Adding this cut to the linear programming relaxation, we get:

$$\begin{aligned} \max \quad & z = 5.5x_1 + 2.1x_2 \\ \text{subject to} \quad & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 17 \\ & x_2 \leq 3 \\ & x_1, x_2 \geq 0. \end{aligned}$$

We introduce a slack variable x_5 to turn the constraint $x_3 \leq 2$ into equality. By the same reasoning as before, x_5 can also be constrained to be integer. Solving this linear program, we find the optimal tableau:

$$\begin{array}{rrrrrcl} z & & & +0.6875x_4 & +0.725x_5 & = & 13.8625 \\ & & x_3 & +0.125x_4 & -1.25x_5 & = & 0.375 \\ & x_1 & & +0.125x_4 & -0.25x_5 & = & 1.375 \\ & & x_2 & & x_5 & = & 3 \end{array}$$
$$x_1, x_2, x_3, x_4, x_5 \geq 0.$$

Its basic solution in the (x_1, x_2)-space is $x_1 = 1.375, x_2 = 3$ with objective value $z = 13.8625$. Since x_1 is not integer, we need to generate another cut. From the constraint $x_1 + 0.125x_4 - 0.25x_5 = 1.375$, we generate the new fractional cut using (3.3), which is $0.125x_4 + 0.75x_5 \geq 0.375$. Replacing $x_4 = 17 - 8x_1 - 2x_2$ and $x_5 = 3 - x_2$ we get:

$$x_1 + x_2 \leq 4.$$

Adding this cut and solving again the new linear program, we find a new optimal solution $x_1 = 1.5, x_2 = 2.5$ with objective value $z = 13.5$. This solution is again not integer. Two more iterations are needed to obtain the optimal solution $x_1 = 1, x_2 = 3$ with objective value $z = 11.8$.

3.2 Optimization software

In this section we describe the optimization software we use both from a historical and technical point of view. For our purposes, we used the software ILOG CPLEX Optimization Studio (version 11.1.1) through its AMPL (version 20080701) interface. We also used MATLAB (version R2020b).

3.2.1 IBM ILOG CPLEX Optimization Studio

IBM ILOG CPLEX Optimization Studio (often informally referred to simply as CPLEX) is an optimization software package. The CPLEX Optimizer was named for the simplex method as implemented in the C programming language, although today it also supports other types of mathematical optimization and offers interfaces different from C. Originally developed by Robert E. Bixby, it has been marketed since 1988 by CPLEX Optimization Inc., acquired by ILOG in 1997, which was itself acquired by IBM in January 2009. CPLEX is actively maintained and developed within IBM. The IBM ILOG CPLEX Optimizer solves integer programming problems, very large linear programming problems using either primal or dual variants of the simplex method or the barrier interior point method, convex and non-convex quadratic programming problems, and convex

quadratically constrained problems [**12**].

3.2.2 AMPL

AMPL, acronym for A Mathematical Programming Language, is a high-level language developed by the Bell laboratories to describe and solve large and complicated mathematical programming problems (for example, optimization and scheduling problems). AMPL does not solve problems by itself, but instead writes files with full details of the problem instances to be solved and invokes separate solvers (such as CPLEX, Gurobi and MINOS). One advantage of AMPL is the similarity of its syntax to the mathematical notation of optimization problems. This allows for a very concise and readable definition of problems in the domain of optimization. AMPL was created by Robert Fourer, David Gay and Brian Kernighan. AMPL is available for many popular operating systems including Linux, macOS, Solaris, AIX, and Windows. The translator is proprietary software maintained by AMPL Optimization LLC [**3**].

3.2.3 MATLAB

MATLAB (short for Matrix Laboratory) is an environment for numerical computation and statistical analysis written in C, which also includes the programming language of the same name created by MathWorks. MATLAB allows you to manipulate matrices, visualize functions and data, implement algorithms, create user interfaces, and interface with other programs. MATLAB is used by millions of people in industry and universities due to its many tools to support the most diverse applied fields of study and runs on various operating systems, including Windows, macOS, Linux and Unix [**16**].

Chapter 4

State of the art

The problem of air traffic flow management has been addressed and solved over the years through different approaches. One of these is operations research and in particular integer linear programming. In the first part of this chapter, we will give a quick overview on the evolution of linear integer programming models for solving the air traffic flow management problem. Then, we will see in detail one of these models, namely the one by Bertsimas and Stock Patterson, which will be the basis of our work. In particular, we will provide additional details with respect to [**8**], as deduced from the implementation of the model.

4.1 Historical notes

The following chronological list shows some of the most important integer linear programming models for air traffic flow management.

Helme (1992) [15]: This article proposes a linear program in order to minimize delays through ground holding and airborne delay. The trajectories are predetermined and cannot be changed during the flight. Furthermore, the speed control during the flight is not allowed. The decision variables of the model establish whether or not a flight is to be held on the ground at the time of departure and, if so, for how long and whether or not a flight is to be held over the destination airport and, if so, for how long.

Andreatta, Odoni and Richetta (1993) [5]: This work proposes solutions to the problem of air traffic flow through ground-holding models, avoiding to consider the possibility of airborne delays as they are generally more expensive.

Bertsimas and Stock Patterson (1998) [8]: This paper introduces the possibility of speed control during the flight. Also in this case, the flight paths are fixed. This model determines, for each flight, the optimal departure time and transit time in

the sectors of the route. Furthermore, the decision variables used in this model capture the three types of connectivity: connectivity between sectors, connectivity between airports, and connectivity in time. Finally, the formulation of the problem is particularly strong. This paper is the basis of our work and we will study it in detail in the next section.

Bertsimas and Stock Patterson (2000) [9]: This paper introduces the possibility for a flight to re-route, that is, deciding which of the possible flight paths minimizes the total cost of the delay. Speed control during the flight is allowed.

Bertsimas, Lulli and Odoni (2011) [7]: This work is basically an extension of the previous one and was able to solve bigger instances. The model covers all the phases of each flight and solves for an optimal combination of flow management actions, including ground holding, re-routing, speed control, and airborne delay. A distinguishing feature of the model is that it allows for rerouting decisions. This is achieved through the imposition of sets of local conditions that make it possible to represent rerouting options in a compact way by only introducing some new constraints.

Agustín, Alonso-Ayuso and Escudero (2012) [1, 2]: The model allows for flight cancelation and re-routing, if necessary. It considers several types of objective functions to minimize, namely, the number of flights exceeding a given time delay (that can be zero), separable and nonseparable ground holding and air delay costs, penalization of alternative routes to the scheduled one for each flight, time unit delay cost to arrive to the nodes (that are, air sectors and airports) and penalization for advancing arrival to the nodes over the schedule.

Fomeni, Lulli and Zografos (2017) [14]: This article presents a model that contributes to the optimization and optimal configuration of the trajectory based operations (TBO). TBO is the concept of improving productivity, flight efficiency, flight times through better prediction and coordination of aircraft trajectories. A 0-1 integer programming model is developed with the aim of assigning a 4D-trajectory to each flight with the aim of optimizing the efficiency of the ATM system. The model considers the preferred 4D-trajectory of all pre-tactical flights and exits an optimal 4D-trajectory for each flight. The TBO concept implies that these 4D-trajectories need to be discussed with other stakeholders and handled accordingly during the flight. The novelty of this model is that it considers both the complete 4D-trajectories for each flight, and the preferences and priorities of the ATM stakeholders.

4.2 Bertsimas and Stock Patterson's model

As mentioned in the previous section, this model is the basis of our work and can be found in [**8**]. The precise form of the air traffic flow management problem addressed by this model is as follows: given a configuration of departures, arrivals and flight paths (which in this case are a sequence of sectors along with a departure airport and one of arrival), find a trajectory configuration that minimizes flight delays, both on the ground and on airborne, subject to airspace capacity limitations. Allowed strategies are: ground maintenance, flight maintenance, speed control. In other words, the goal of the problem is to decide how much time each flight will spend on the ground and in the air to minimize the total cost of delay. Let us formally present the data of the problem:

$$\mathcal{F} = \{1, \ldots, F\} \text{ is a set of flights,}$$
$$\mathcal{K} = \{1, \ldots, K\} \text{ is a set of airports,}$$
$$\mathcal{J} = \{1, \ldots, J\} \text{ is a set of sectors,}$$
$$\mathcal{T} = \{1, \ldots, T\} \text{ is a set of time periods,}$$
$$\mathcal{C} = \{(f', f) : f' \text{ is continued by flight } f\} \text{ is a set of pairs of flights that are continued,}$$
$$N_f = \text{number of sectors in flight } f\text{'s path,}$$
$$P(f,i) = \begin{cases} \text{the departure airport,} & \text{if } i = 1, \\ \text{the } (i-1)^{st} \text{ in flight } f\text{'s path,} & \text{if } 1 < i < N_f, \\ \text{the arrival airport,} & \text{if } i = N_f, \end{cases}$$
$$P_f = \{P(f,i) : 1 \leq i \leq N_f\},$$
$$D_k(t) = \text{departure capacity of airport } k \text{ at time } t,$$
$$A_k(t) = \text{arrival capacity of airport } k \text{ at time } t,$$
$$S_j(t) = \text{capacity of sector } j \text{ at time } t,$$
$$d_f = \text{scheduled departure time of flight } f,$$
$$r_f = \text{scheduled arrival time of flight } f,$$
$$s_f = \text{turnaround time of an airplane after flight } f,$$
$$c_f^g = \text{cost of holding flight } f \text{ on the ground for one unit of time,}$$
$$c_f^a = \text{cost of holding flight } f \text{ in the air for one unit of time,}$$
$$l_{fj} = \text{number of time units that flight } f \text{ must spend in sector } j,$$
$$T_f^j = \text{set of feasible times for flight } f \text{ to arrive to sector } j = [\underline{T}_f^j, \overline{T}_f^j],$$
$$\underline{T}_f^j = \text{first time period in the set } T_f^j, \text{ and}$$
$$\overline{T}_f^j = \text{last time period in the set } T_f^j.$$

We define a decision variable as follows. For every $f \in \mathcal{F}$, $j \in P_f$ and $t \in T_f^j$:

$$w_{ft}^j = \begin{cases} 1 & \text{if flight } f \text{ arrives at sector } j \text{ by time } t, \\ 0 & \text{otherwise.} \end{cases}$$

This definition using by and not at is fundamental for understanding the formulation. Also recall that we have also defined for each flight a list P_f including the departure airport, the appropriate sectors and the arrival airport, so that the variable w_{ft}^j will only be defined for those elements j in the list P_f. Furthermore, we have defined T_f^j as the set of feasible times for flight f to arrive to sector j, so that the variable w_{ft}^j will only be defined for those times within T_f^j. Thus, in the formulation whenever the variable w_{ft}^j is used, it is assumed that (f, j, t) is a feasible combination. Since flight f has to arrive at sector j by the last possible time in its time window, we can impose $w^j_{f\overline{T}_f^j} = 1$.

The objective of the formulation is to minimize total delay cost. We will now show how to derive it. Noticing that the first sector for every flight is the departure airport, the total number of time units that flight f is held on the ground can be expressed as the actual departure time minus the scheduled departure time, that is,

$$g_f = \sum_{t \in T_f^k,\, k=P(f,1)} t(w_{ft}^k - w_{f,t-1}^k) - d_f.$$

Noticing that the last sector for every flight is the arrival airport, the total number of time units that flight f is held on airborne can be expressed as the actual arrival time minus the scheduled arrival time minus the amount of time that the flight has been held on the ground, that is,

$$a_f = \sum_{t \in T_f^k,\, k=P(f,N_f)} t(w_{ft}^k - w_{f,t-1}^k) - r_f - g_f.$$

We define our objective function as follows:

$$\min \sum_{f \in \mathcal{F}} [c_f^g g_f + c_f^a a_f].$$

Finally, by replacing the expressions of g_f and a_f in the latter expression and rearranging the terms, we can present the objective function along with the complete formulation of the problem.

$$\min \sum_{f\in\mathcal{F}}[(c_f^g - c_f^a) \sum_{t\in T_f^k,\, k=P(f,1)} t(w_{ft}^k - w_{f,t-1}^k) + c_f^a \sum_{t\in T_f^k,\, k=P(f,N_f)} t(w_{ft}^k - w_{f,t-1}^k) + (c_f^g - c_f^a)d_f - c_f^a r_f]$$

subject to

$$\sum_{f\in\mathcal{F}:\, P(f,1)=k} (w_{ft}^k - w_{f,t-1}^k) \le D_k(t) \qquad k\in\mathcal{K},\, t\in\mathcal{T}, \tag{4.1}$$

$$\sum_{f\in\mathcal{F}:\, P(f,N_f)=k} (w_{ft}^k - w_{f,t-1}^k) \le A_k(t) \qquad k\in\mathcal{K},\, t\in\mathcal{T}, \tag{4.2}$$

$$\sum_{f\in\mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j',\, i<N_f} (w_{ft}^j - w_{ft}^{j'}) \le S_j(t) \qquad j\in\mathcal{J},\, t\in\mathcal{T}, \tag{4.3}$$

$$w_{f,t+l_{fj}}^{j'} - w_{ft}^j \le 0 \qquad \begin{cases} f\in\mathcal{F},\, t\in T_f^j,\, j=P(f,i), \\ j'=P(f,i+1),\, i<N_f, \end{cases} \tag{4.4}$$

$$w_{ft}^k - w_{f',t-s_{f'}}^k \le 0 \qquad \begin{cases} (f',f)\in\mathcal{C},\, t\in T_f^k, \\ k=P(f,1)=P(f',N_{f'}), \end{cases} \tag{4.5}$$

$$w_{ft}^j - w_{f,t-1}^j \ge 0 \qquad f\in\mathcal{F},\, j\in P_f,\, t\in T_f^j, \tag{4.6}$$

$$w_{ft}^j \in \{0,1\} \qquad f\in\mathcal{F},\, j\in P_f,\, t\in T_f^j. \tag{4.7}$$

The first three sets of constraints take into account the capacities of various elements considered.

Constraints (4.1) guarantee that the number of flights which may take off from airport k at time t, will not exceed the departure capacity of airport k at time t.

Similarly, constraints (4.2) ensure that the number of flights which may arrive at airport k at time t, will not exceed the arrival capacity of airport k at time t.

Constraints (4.3) guarantee that the sum of all flights which may feasibly be in sector j at time t will not exceed the capacity of sector j at time t. A peculiarity of this model that we noticed during the implementation is that the system allows an aircraft to delay on airborne by stationing above the departure airport. In fact, due to this artifice, the aircraft occupies neither a sector nor a runway for take-off. In other words, it may happen that an aircraft sees the airport as a normal sector in which no capacity is defined and which therefore gives the possibility to stay on airborne above it without entering the next sector and consequently avoiding the congestion of the runways for take-off and of

the real sectors. Staying on airborne above the departure airport is however considered as an airborne delay.

Constraints (4.4) represent connectivity between sectors. They stipulate that if a flight arrives at sector j' by time $t + l_{fj}$, then it must have arrived at sector j by time t where j and j' are contiguous sectors in flight f's path. These are the constraints that allow speed control. Note that $l_{f,P(f,1)} = 0$, which means that as soon as a flight takes off it is considered immediately inside the first sector. Another peculiarity of this model is that these last constraints allow an aircraft to manage the flight delay even using the intermediate sectors even if, in general, it is preferred to manage it all in the last sector, the one in which the arrival airport is located.

Constraints (4.5) ensure that if flight f departs from airport k by time t, then flight f' must have arrived at airport k by time $t - s_{f'}$.

Constraints (4.6) represent connectivity in time. Hence, if a flight has arrived by time t, then $w^j_{ft'}$ has to have a value of 1 for all later time periods, $t' \geq t$.

Chapter 5

A model for ATFM with dynamic configurations

In this chapter we will formalize our approach to the ATFM problem, enriching the model of Bertsimas and Stock Patterson [8] and highlighting the differences with respect to it.

5.1 An overview of the ATFM problem

5.1.1 Collapsed sectors and airspace configurations

As introduced in Subsection 2.2.2, the airspace is divided into elementary sectors, which can be seen as the pieces of a puzzle. Each airport is contained entirely in an elementary sector. A configuration is a way to merge elementary sectors in order to create larger sectors. In general, if the airspace is divided into J elementary sectors, we do not have as many controllers to control each elementary sector.

In mathematical terms, if $\mathcal{J}$ is the set containing the elementary sectors then a configuration is a partition of $\mathcal{J}$. The subsets of $\mathcal{J}$ that form the partition are the collapsed sectors. Suppose, for example, that $\mathcal{J} = \{a, b, c, d\}$.

Let $\mathcal{H}$ be the set containing the collapsed sectors, for example, $\mathcal{H} = \{C_1, C_2, C_3, C_4\}$, where $C_1 = \{a, b\}, C_2 = \{c, d\}, C_3 = \{a, c\}$ and $C_4 = \{b, d\}$. It is necessary that, for each collapsed sector, the union of the elementary sectors that compose it is connected.

A configuration is given, for example, by $C_1 = \{a, b\}$ and $C_2 = \{c, d\}$. Another example of configuration is given by $C_3 = \{a, c\}$ and $C_4 = \{b, d\}$. No other configurations can exist with the elements thus defined.

As said before, in general, the controllers are less than the elementary sectors to be controlled, so we need to merge the elementary sectors into collapsed sectors, in order to better control them. We will define a list of configurations $\mathcal{M}$ (with reference to the example, given by $\mathcal{M} = \{M_1, M_2\}$ where $M_1 = \{C_1, C_2\}$ and $M_2 = \{C_3, C_4\}$), in order

to be able to choose the best one, that is the one that minimizes the total cost of delays.

5.1.2 Time representation

We divide a long time period into smaller time periods, for example a 3 hour period into 36 time periods of 5 minutes each. So, remaining in the example, since 3 hours equals 180 minutes, by $t = 1$ we mean the interval $[0, 5)$, by $t = 2$ we mean the interval $[5, 10)$, and so on. Time discretization is fundamental for modeling the problem through an ILP. Obviously some details of the ATFM are not fully captured, but we are only interested in the macroscopic aspects.

A time-related aspect to consider is that it is not realistic to be able to choose the most appropriate configuration each time period. In order to manage this criticality we define the parameter τ, which indicates the minimum number of consecutive time periods in which the chosen configuration must remain active before it can be changed.

5.1.3 Flights

The trajectory of each flight can be described using the sequence ABC, a, b, c, d, XYZ, where ABC is the departure airport, a, b, c and d are, in order, the elementary sectors that the flight will cross and XYZ is the destination airport. We will see that it is more convenient to continue describing the trajectory of a flight through a list of elementary sectors and not by a list of collapsed sectors.

5.1.4 Capacity

The concept of capacity is associated with that of the collapsed sector. The capacity of a collapsed sector at the t time period represents the number of flights that can be within that collapsed sector at the t period. The capacity of collapsed sectors depends on time, in fact in general it is not constant.

5.1.5 Airports

An airport is, of course, always the beginning or the end of a flight path. We will associate two parameters with each airport. The first is the departure capacity in a t time period, that is how many flights can take off from a given airport in the t time period. The second is the arrival capacity in a t time period, that is how many flights can land at a given airport in the t time period.

5.1.6 Delays

Each flight has a scheduled departure time and a scheduled arrival time. For each flight, we know how long it takes to cross a certain sector. So, knowing the departure time, we

can calculate the minimum time a flight will arrive in a certain sector. At this point, it is natural to introduce the possibility that a flight may be delayed. For each f flight we define a Δ_f time period which indicates how long that flight may be delayed upon arrival.

As in the Bertsimas and Stock Patterson model, each flight f has a window of time to reach a certain elementary sector j. The lower bound of this interval is given by the departure time plus the sum of all the time periods that the flight must spend in the sectors preceding j. The upper bound is given by the lower bound plus the delay Δ_f: in this way, we guarantee the maximum flexibility on how a flight can spread the allowed delay.

5.2 Problem data

We are ready to present the data of our formulation and related notation. Although some notation has already been introduced in the formulation of Bertsimas and Stock Patterson, we rewrite them for the sake of clarity:

$\mathcal{F} = \{1, \ldots, F\}$ is the set of flights,

$\mathcal{K} = \{1, \ldots, K\}$ is the set of airports,

$\mathcal{H} = \{1, \ldots, H\}$ is the set of collapsed sectors,

$\mathcal{J} = \{1, \ldots, J\}$ is the set of elementary sectors,

$\mathcal{M} = \{1, \ldots, M\}$ is the set of configurations,

$\mathcal{T} = \{1, \ldots, T\}$ is the set of time periods,

$\mathcal{P}_m = \{h \in \mathcal{H}$: their union is the configuration $m \in \mathcal{M}\}$ are the collapsed which form the configuration $m \in \mathcal{M}$,

$\mathcal{B}_h = \{j \in \mathcal{J}$: their union is the collapsed $h \in \mathcal{H}\}$ are the elementary sectors which form the collapsed $h \in \mathcal{H}$,

$N_f =$ number of sectors in flight f's path,

$$P(f,i) = \begin{cases} \text{the departure airport,} & \text{if } i = 1, \\ \text{the } (i-1)^{st} \text{ in flight } f\text{'s path,} & \text{if } 1 < i < N_f, \\ \text{the arrival airport,} & \text{if } i = N_f, \end{cases}$$

$$P_f = \{P(f,i) : 1 \leq i \leq N_f\},$$

$D_k(t) =$ departure capacity of airport k at time t,

$A_k(t) =$ arrival capacity of airport k at time t,

$S_h(t) =$ capacity of collapsed sector h at time t,

$d_f =$ scheduled departure time of flight f,

$r_f =$ scheduled arrival time of flight f,

$\Delta_f =$ maximum delay allowed for the flight f,

$c_f^g =$ cost of holding flight f on the ground for one unit of time,

$c_f^a =$ cost of holding flight f in the air for one unit of time,

$l_{fj} =$ number of time units that flight f must spend in elementary sector j,

$T_f^j =$ set of feasible times for flight f to arrive to sector $j = [\underline{T}_f^j, \overline{T}_f^j]$,

$\underline{T}_f^j =$ first time period in the set T_f^j,

$\overline{T}_f^j = \underline{T}_f^j + \Delta_f =$ last time period in the set T_f^j,

$\tau =$ minimum number of consecutive time periods a configuration must remain active, and

$$C_h(t) = \left(\sum_{j \in \mathcal{B}_h} \sum_{f \in \mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j',\, i<N_f} 1 \right) - S_h(t) = \text{constant used to make capacity constraints redundant when related to a non-active configuration.}$$

5.3 Decision variables

We now present the decision variables. The first is the same used in the Bertsimas and Stock Patterson model and we recall its definition for clarity. For every $f \in \mathcal{F}$, $j \in P_f$ and $t \in T_f^j$ we define

$$w_{ft}^j = \begin{cases} 1 & \text{if flight } f \text{ arrives at sector } j \text{ by time } t, \\ 0 & \text{otherwise.} \end{cases}$$

Let us now define a new decision variable to manage the choice of configuration over time. For every $m \in \mathcal{M}$ and $t \in \mathcal{T}$ we define

$$y_{mt} = \begin{cases} 1 & \text{if configuration } m \text{ is active at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

We decided to use variables y_{mt} because they easily capture the fact that one and only one configuration must be active in a certain period of time. Moreover, these variables allow to manage the stability of a configuration over time, because, as mentioned in Subsection 5.1.2, it is not realistic to allow to choose the best one at each time period.

5.4 Objective function

As for the objective function, we use the same proposal in the work of Bertsimas and Stock Patterson. We recall it, remembering that it is the weighted sum of the two types of delay of each single flight f, which must be minimized:

$$\min \sum_{f \in \mathcal{F}} [c_f^g g_f + c_f^a a_f].$$

In the next section we will report the extended form of this objective function, as discussed in Section 4.2, along with the rest of the formulation.

5.5 Mathematical formulation

We are ready to present the model:

$$\begin{aligned}\min \sum_{f\in\mathcal{F}}[(c_f^g - c_f^a) &\sum_{t\in T_f^k,\, k=P(f,1)} t(w_{ft}^k - w_{f,t-1}^k)\\ &+ c_f^a \sum_{t\in T_f^k,\, k=P(f,N_f)} t(w_{ft}^k - w_{f,t-1}^k)\\ &+ (c_f^g - c_f^a)d_f - c_f^a r_f]\end{aligned}$$

subject to

$$\sum_{f\in\mathcal{F}:\, P(f,1)=k} (w_{ft}^k - w_{f,t-1}^k) \le D_k(t) \qquad k\in\mathcal{K},\, t\in\mathcal{T}, \tag{5.1}$$

$$\sum_{f\in\mathcal{F}:\, P(f,N_f)=k} (w_{ft}^k - w_{f,t-1}^k) \le A_k(t) \qquad k\in\mathcal{K},\, t\in\mathcal{T}, \tag{5.2}$$

$$\begin{aligned}\sum_{j\in\mathcal{B}_h} \sum_{f\in\mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j',\, i<N_f} &(w_{ft}^j - w_{ft}^{j'})\\ &\le S_h(t) + C_h(t)(1-y_{mt}) \qquad m\in\mathcal{M},\, h\in\mathcal{P}_m,\, t\in\mathcal{T},\end{aligned} \tag{5.3}$$

$$w_{f,t+l_{fj}}^{j'} - w_{ft}^j \le 0 \qquad \begin{cases} f\in\mathcal{F},\, t\in T_f^j,\, j=P(f,i),\\ j'=P(f,i+1),\, i<N_f, \end{cases} \tag{5.4}$$

$$w_{ft}^j - w_{f,t-1}^j \ge 0 \qquad f\in\mathcal{F},\, j\in P_f,\, t\in T_f^j, \tag{5.5}$$

$$w_{ft}^j \in \{0,1\} \qquad f\in\mathcal{F},\, j\in P_f,\, t\in T_f^j, \tag{5.6}$$

$$\sum_{m\in\mathcal{M}} y_{mt} = 1 \qquad t\in\mathcal{T}, \tag{5.7}$$

$$y_{mt} - y_{m,t-1} \le y_{mu} \quad m\in\mathcal{M},\, t\in\mathcal{T},\, u\in\{t+1,\ldots,\min(t+\tau-1,T)\}, \tag{5.8}$$

$$y_{mt}\in\{0,1\} \qquad m\in\mathcal{M},\, t\in\mathcal{T}. \tag{5.9}$$

5.5.1 Airport capacity constraints

Constraints (5.1) and (5.2) limit the capacity of departure and arrival airports:

$$\sum_{f\in\mathcal{F}:P(f,1)=k} (w^k_{ft} - w^k_{f,t-1}) \leq D_k(t) \qquad k \in \mathcal{K},\, t \in \mathcal{T}, \tag{5.1}$$

$$\sum_{f\in\mathcal{F}:P(f,N_f)=k} (w^k_{ft} - w^k_{f,t-1}) \leq A_k(t) \qquad k \in \mathcal{K},\, t \in \mathcal{T}. \tag{5.2}$$

They are the same as those considered in the Bertsimas and Stock Patterson model. We recall that constraints (5.1) guarantee that the number of flights which may take off from airport k at time t, will not exceed the departure capacity of airport k at time t. Similarly, constraints (5.2) ensure that the number of flights which may arrive at airport k at time t, will not exceed the arrival capacity of airport k at time t.

5.5.2 Collapsed sector capacity constraints

Constraints (5.3) limit the capacity of collapsed sectors, and they are specific of the formulation proposed in this thesis:

$$\begin{aligned}\sum_{j\in\mathcal{B}_h} \sum_{f\in\mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j',\, i<N_f} (w^j_{ft} - w^{j'}_{ft}) \\ \leq S_h(t) + C_h(t)(1 - y_{mt}) \qquad m \in \mathcal{M},\, h \in \mathcal{P}_m,\, t \in \mathcal{T}.\end{aligned} \tag{5.3}$$

Constraints (5.3) ensure that, for each collapsed sector h, the total number of flights that may feasibly cross it will not exceed the capacity of h at any time t, that is $S_h(t)$. This number is expressed by the left-hand side of (5.3), since it counts the number of flights that has entered collapsed sector h (from any elementary sector not contained in the collapsed sector) by time t and has not yet entered in the successive collapsed sector (in any elementary sector not contained in the collapsed sector). The right-hand side of (5.3) indicates the capacity of the collapsed sector h at time t in the active configuration m which is active at time t.

Moreover, the right-hand side of (5.3) shows the relationship between these constraints and the constraints (5.7), which manage the choice of configuration and will be illustrated in Subsection 5.5.4. Let us see them in detail through an example. Suppose that $\mathcal{M} =$

$\{1, 2\}$. Constraints (5.3) can be rewritten as:

$$\sum_{j\in\mathcal{B}_h} \sum_{f\in\mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j',\, i<N_f} (w^j_{ft} - w^{j'}_{ft})$$

$$\leq S_h(t) + C_h(t)(1 - y_{1,t}) \qquad h \in \mathcal{P}_1,\, t \in \mathcal{T},$$

$$\sum_{j\in\mathcal{B}_h} \sum_{f\in\mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j',\, i<N_f} (w^j_{ft} - w^{j'}_{ft})$$

$$\leq S_h(t) + C_h(t)(1 - y_{2,t}) \qquad h \in \mathcal{P}_2,\, t \in \mathcal{T}.$$

Now suppose that for a certain period of time $\bar{t}$ we have $y_{1,\bar{t}} = 1$ and $y_{2,\bar{t}} = 0$. Remembering that

$$C_h(t) = \left(\sum_{j\in\mathcal{B}_h} \sum_{f\in\mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j',\, i<N_f} 1 \right) - S_h(t),$$

we can rewrite the latest constraints as:

$$\sum_{j\in\mathcal{B}_h} \sum_{f\in\mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j',\, i<N_f} (w^j_{f,\bar{t}} - w^{j'}_{f,\bar{t}}) \leq S_h(\bar{t}) \qquad h \in \mathcal{P}_1, \tag{5.10}$$

$$\begin{aligned} &\sum_{j\in\mathcal{B}_h} \sum_{f\in\mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j',\, i<N_f} (w^j_{f,\bar{t}} - w^{j'}_{f,\bar{t}}) \\ &\qquad \leq \sum_{j\in\mathcal{B}_h} \sum_{f\in\mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j',\, i<N_f} (1 - 0) \qquad h \in \mathcal{P}_2. \end{aligned} \tag{5.11}$$

Since the right-hand side of (5.11) is an upper bound of the left-hand side, constraints (5.11) are irrelevant and therefore constraints (5.10) are the only ones that are considered. In fact, they are the constraints related to configuration 1, which is the one active at time $\bar{t}$ because $y_{1,\bar{t}} = 1$ and $y_{2,\bar{t}} = 0$.

5.5.3 Connectivity constraints

These two groups of constraints are borrowed from the formulation of Bertsimas and Stock Patterson.

Constraints (5.4) represent connectivity between elementary sectors:

$$w^{j'}_{f,t+l_{fj}} - w^j_{ft} \leq 0 \qquad \begin{cases} f \in \mathcal{F},\, t \in T^j_f,\, j = P(f,i), \\ j' = P(f,i+1),\, i < N_f. \end{cases} \tag{5.4}$$

They ensure that a flight cannot enter the next elementary sector on its path until it has spent l_{fj} time units (the minimum possible) traveling through elementary sector j, the current elementary sector in its path.

Constraints (5.5) represent connectivity in time:

$$w^j_{ft} - w^j_{f,t-1} \geq 0 \qquad f \in \mathcal{F},\, j \in P_f,\, t \in T^j_f. \tag{5.5}$$

Thus, if a flight has arrived by time t, then w^j_{ft} has to have a value of 1 for all later time periods, $t' \geq t$.

5.5.4 Configuration constraints

Constraints (5.7) and (5.8) manage the choice of the configuration and its maintenance over time:

$$\sum_{m \in \mathcal{M}} y_{mt} = 1 \qquad t \in \mathcal{T}, \tag{5.7}$$

$$y_{mt} - y_{m,t-1} \leq y_{mu} \quad m \in \mathcal{M},\, t \in \mathcal{T},\, u \in \{t+1, \ldots, \min(t+\tau-1, T)\}. \tag{5.8}$$

Constraints (5.7) ensure that, for any time t, one and only one configuration m can be active (notice that $y_{mt} \in \{0, 1\}$). Constraints (5.8) stipulate that once a configuration is chosen, it must be maintained for at least τ periods of time before it can be changed. We must consider $y_{mt} = 0$ if $t < 1$ or $t > T$. For clarity, let us give some examples of sequences of admissible configurations. If $T = 10$, $\mathcal{M} = \{1, 2, 3\}$ and $\tau = 4$ we could have $(1, 1, 1, 1, 2, 2, 2, 2, 3, 3)$, $(2, 2, 2, 2, 2, 1, 1, 1, 1, 3)$ or $(3, 3, 3, 3, 3, 3, 1, 1, 1, 1)$ but of course we couldn't have $(1, 1, 1, 1, 2, 2, 2, 3, 3, 3)$, $(2, 2, 2, 3, 3, 3, 3, 1, 1, 1)$ or $(1, 2, 2, 2, 2, 2, 2, 2, 2, 2)$.

5.6 A class of valid inequalities

We wonder if the formulation can have similar characteristics to that of Berstimas and Stock Patterson, from the point of view of the quality of linear relaxation. Recall that the linear relaxation of the Bertsimas and Stock Patterson model is consistently strong, in fact the optimal solution of the relaxed linear problem is almost always the same as for the relaxed linear program. We note that the constraints (5.3) have changed with respect to the Bertsimas and Stock Patterson formulation. In fact, from a structural point of view, given a configuration $m \in \mathcal{M}$ and a time period $t \in \mathcal{T}$, the variables are linked to the elementary sectors. Conversely, in the Bertsimas and Stock Patterson model, the variables are linked to the collapsed sectors (in this case the sectors to be considered are the collapsed sectors). This generates structural differences, as demonstrated by the following example. Let us consider the situation of the Figure 5.1. The red arrows

represent the path of flight f. It takes off from airport k_1, located in the elementary sector c, then passes through the elementary sectors d and b, and finally lands at airport k_2, located in the elementary sector a. We write, for a fixed period of time $t \in \mathcal{T}$, the constraints related to the capacity of the sectors α, β and γ, which we assume to be constant and equal to 1, in both cases. Under these assumptions, in our model, these constraints are:

$$w^a_{ft} - w^{k_2}_{ft} \leq 1 \quad \text{for sector } \alpha,$$

$$w^c_{ft} - w^d_{ft} + w^b_{ft} - w^a_{ft} \leq 1 \quad \text{for sector } \beta,$$

$$w^d_{ft} - w^b_{ft} \leq 1 \quad \text{for sector } \gamma.$$

In the Bertsimas and Stock Patterson model the only sectors to consider are α, β and γ (we ignore the elementary sectors a, b, c and d) and therefore we have:

$$w^\alpha_{ft} - w^{k_2}_{ft} \leq 1 \quad \text{for sector } \alpha,$$

$$w^\beta_{ft} - w^\gamma_{ft} + w^\beta_{ft} - w^\alpha_{ft} = 2w^\beta_{ft} - w^\gamma_{ft} - w^\alpha_{ft} \leq 1 \quad \text{for sector } \beta,$$

$$w^\gamma_{ft} - w^\beta_{ft} \leq 1 \quad \text{for sector } \gamma.$$

We note that in the first case we have 3 constraints and 5 variables but in the second we have 3 constraints and 4 variables, showing that the new model cannot be obtained from the Bertsimas and Stock Patterson one by simply renaming variables. This prompted us to look for new valid inequalities to improve the formulation.

Let us consider a new category of variables, defined in [**8**]. For every $f \in \mathcal{F}$, $j \in P_f$ and $t \in T^j_f$ we define

$$u^j_{ft} = \begin{cases} 1 & \text{if flight } f \text{ arrives at sector } j \text{ at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

These variables can be expressed as follows:

$$u^j_{ft} = w^j_{ft} - w^j_{f,t-1}$$

and vice versa,

$$w^j_{ft} = \sum_{t' \leq t} u^j_{ft'}.$$

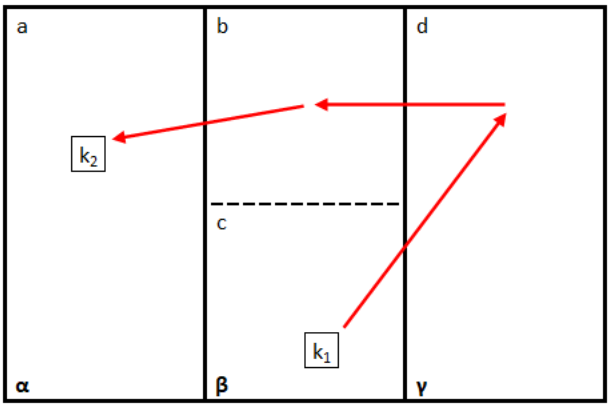


Figure 5.1: Comparison of the capacity constraints in the two formulations.

We state that

$$\sum_{j\in\mathcal{B}_h:P(f,i)=j,1<i<N_f} w^j_{ft} \leq 1 \qquad m \in \mathcal{M},\, h \in \mathcal{P}_m,\, f \in \mathcal{F},\, t \in \mathcal{T},$$

which can be rewritten as:

$$\sum_{j\in\mathcal{B}_h:P(f,i)=j,1<i<N_f} (w^j_{ft} - w^j_{f,t-1}) \leq 1 \qquad m \in \mathcal{M},\, h \in \mathcal{P}_m,\, f \in \mathcal{F},\, t \in \mathcal{T}. \tag{5.12}$$

The group of inequalities (5.12) states that, given a configuration m, if a flight f enters a collapsed sector h at time t then it can enter it through one and only one of the elementary sectors j that form the collapsed sector.

5.7 Size of the formulation

Let D be the maximum cardinality of the set of feasible times for flight f to be in sector j taken over all f and j, that is,

$$D = \max_{f\in\mathcal{F},j\in P_f} |T^j_f|.$$

Let

$$X = \max_{f\in\mathcal{F}} N_f,$$

be the maximum number of sectors that a flight passes through along its path, taken over all flights. Let Z be the maximum number of collapsed sectors contained in all

configurations, that is,

$$Y = \max_{m \in \mathcal{M}} |\mathcal{P}_m|.$$

Let $|\mathcal{F}|$ be the total number of flights, $|\mathcal{T}|$ be the total number of time periods, $|\mathcal{K}|$ be the total number of airports, $|\mathcal{J}|$ be the total number of elementary sectors, $|\mathcal{H}|$ be the total number of collapsed sectors and $|\mathcal{M}|$ be the total number of configurations.

The exact number of variables w_{ft}^j is $\sum_f \sum_{j \in P_f} |T_f^j|$ since each flight has a different number of elementary sectors and number of feasible time intervals associated with it. An upper bound on the number of variables w_{ft}^j will be $|\mathcal{F}|DX$. The exact number of variables y_{mt} is $|\mathcal{M}||\mathcal{T}|$. Hence, an upper bound on the number of total variables is given by

$$|\mathcal{F}|DX + |\mathcal{M}||\mathcal{T}|.$$

The exact number of constraints is

$$|\mathcal{K}||\mathcal{T}| + |\mathcal{K}||\mathcal{T}| + |\mathcal{T}| \sum_m \mathcal{P}_m + \sum_{f \in \mathcal{F}} \sum_{j \in P_f} |T_f^j| + \sum_{f \in \mathcal{F}} \sum_{j \in P_f} |T_f^j|$$
$$+|\mathcal{T}| + |\mathcal{M}| \sum_{t \in \mathcal{T}} \min(t + \tau - 1 - (t+1) + 1, T - (t+1) + 1),$$

which can be rewritten as

$$2|\mathcal{K}||\mathcal{T}| + |\mathcal{T}| \sum_m \mathcal{P}_m + 2 \sum_{f \in \mathcal{F}} \sum_{j \in P_f} |T_f^j| + |\mathcal{T}| + |\mathcal{M}| \sum_{t \in \mathcal{T}} \min(\tau - 1, T - t).$$

An upper bound on the number of constraints can then be calculated as

$$2|\mathcal{K}||\mathcal{T}| + |\mathcal{T}|Y + 2|\mathcal{F}|DX + |\mathcal{T}| + |\mathcal{M}||\mathcal{T}|,$$

which can be rewritten as

$$2|\mathcal{F}|DX + (2|\mathcal{K}| + |\mathcal{M}| + Y + 1)|\mathcal{T}|.$$

In order to get a feeling of the size of the formulation for one of the instances considered in this thesis, let us consider the following example:

1. $|\mathcal{K}| = 8$, representing 8 of the most important European airports.
2. $|\mathcal{F}| = 256$, representing 256 flights.
3. $|\mathcal{J}| = 16$, representing 16 elementary sectors.
4. $|\mathcal{H}| = 16$, representing 16 collapsed sectors.
5. $|\mathcal{T}| = 36 = 6 \times 6$, representing a 6 hour day with ten minute intervals.

6. $|\mathcal{M}| = 3$, representing 3 possible configurations.
7. $D = 2$, representing an upper bound of ten minutes that a flight can be late to any given sector.
8. $X = 8$, representing an upper bound of at most eight elementary sectors in a flight's path.
9. $Y = 8$, an upper bound on the number of eight collapsed sectors that can be found in each configuration.

For this example the number of variables is at most 4204 and the number of constraints is at most 9200.

To get an idea of the size of the formulation for a realistic instance, let us consider the following example:

1. $|\mathcal{K}| = 20$, representing 20 of the most important European airports.
2. $|\mathcal{F}| = 10000$, representing 10000 flights.
3. $|\mathcal{J}| = 200$, representing 200 elementary sectors.
4. $|\mathcal{H}| = 100$, representing 100 collapsed sectors.
5. $|\mathcal{T}| = 144 = 12 \times 12$, representing a 12 hour day with five-minute intervals.
6. $|\mathcal{M}| = 4$, representing 4 possible configurations.
7. $D = 6$, representing an upper bound of half an hour that a flight can be late to any given sector.
8. $X = 8$, representing an upper bound of at most eight elementary sectors in a flight's path.
9. $Y = 40$, an upper bound on the number of 40 collapsed sectors that can be found in each configuration.

For this example the number of variables is at most 485776 and the number of constraints is at most 972240. The critical quantities that significantly affect the number of variables and constraints are $D, |\mathcal{F}|$ and X. If for example any of these parameters doubles, both the number of variables and the number of constraints approximately double.

Chapter 6

Model implementation and instance generation

6.1 Model implementation

For the implementation of our model, we have used ILOG CPLEX Optimization Studio [**12**] using its AMPL interface [**3**] as the modelling language. Each instance of the problem is described by a common .mod file, where we define sets, parameters, variables, objective function and constraints, and by a .dat file, which lists the actual elements of the sets and assigns values to the parameters. These two files are then readable by the AMPL code through which we interface with the CPLEX solver. AMPL processes the specific model arising from these data and, then, calls CPLEX to solve it. MATLAB has been used to support the automatic generation of .dat file for different instances. Examples of .dat and .mod are reported in the Appendix.

6.2 Generation of flight trajectories

Regarding the definition of flight paths P_f, where $f \in \mathcal{F}$, we use the following approach, which we implemented in MATLAB. We can see the airspace as a weighted undirected graph, where each node represents an elementary sector or an airport. In particular, each node representing an airport has degree 1, as it is connected only to the elementary sector that contain it. If two elementary sectors are adjacent, then there is an arc that connects the two nodes that represent them. A weight is randomly assigned to each arc, which changes each time we generate a new instance. The weight assigned to the arc connecting two nodes represents how convenient it can be to move from one node to another through that arc. If this number is close to 0, then it is convenient to choose this arc. Conversely, if this number is close to 1 then it is not convenient. In MATLAB, we can build such a graph using the function `graph()`, which has as parameters the list of source nodes,

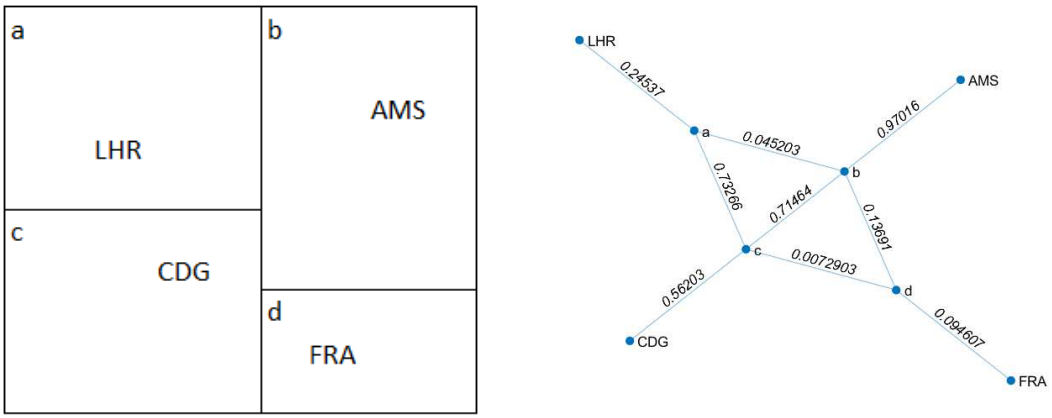


Figure 6.1: An example of airspace and its associated graph.

the list of corresponding termination nodes and the list of weights associated with the arcs. An example of a graph associated with an airspace is shown in Figure 6.1. For the generation of flight paths, we proceed in the following way. We randomly choose a node that represents the departure airport, then we randomly choose another node that represents the arrival airport and, with the function `shortestpath()` (which has as parameters the graph, the departure node and the arrival node), we calculate the shortest path between these two nodes. The result is the flight path. We note that, since the weights are randomly assigned each time the function `shortestpath()` is called, we potentially allow to have, for a fixed pair of airports, paths through different sectors.

6.3 Data for the airspace elements

We list in AMPL the set of configurations $\mathcal{M}$, the set of flights $\mathcal{F}$, the set of airports $\mathcal{K}$, the set of elementary sectors $\mathcal{J}$, the set of collapsed sectors $\mathcal{H}$, the set $\mathcal{P}_m$ of collapsed sectors which form the map $m \in \mathcal{M}$, the set $\mathcal{B}_h$ of elementary sectors which form the collapsed $h \in \mathcal{H}$ and the set of times $\mathcal{T}$.

For the remaining data, we assume that airport capacities are constant over time. Regarding the capacity of the collapsed sectors, we will test the model both in the case of constant and variable capacities over time.

We will also assume that the costs of ground delay and airborne delay are constant, taking into account that, generally, the second delay is more expensive than the first.

We recall that the parameter τ represents the minimum number of consecutive time periods a configuration must remain active.

The time periods each flight has to spend in each of the elementary sectors included

in its path are randomly generated. For each flight and for each elementary sector, we choose between two numbers that are set at the beginning.

Moreover, we randomly generate the maximum delay time of each flight, calculate the flight duration based on how much time each flight spends in each sector, and finally randomly generate the take-off and landing times of each flight.

Finally, on the basis of the parameters already described, we calculate the feasible time window for each flight to arrive in each elementary sector included in its path. In particular, the lower bound of this interval is given by the departure time plus the sum of all the time periods that the flight must spend in the in the previous sectors. The upper bound is given by the lower bound plus the delay.

For any further information, we refer to the Appendix (in particular the .dat file), where we will report a complete example written in AMPL.

Chapter 7

Computational results

The aim to the computational test is to understand if the model proposed in Chapter 5 actually improves tha air traffic flow management. For the implementation we used ILOG CPLEX Optimization Studio (version 11.1.1) **[12]** using its AMPL interface (version 20080701) **[3]** as the modeling language. The AMPL code we have implemented is shown in the Appendix. First of all, we solve each instance of the ATFM problem using dynamic configurations and, then, we compare the results to those obtained by using a fixed configuration. We are looking for evidence to demonstrate that it is convenient to give the possibility to take advantage from multiple configuration at the pre-tactical stage. Furthermore, we will also study the linear relaxation of both in the model with dynamic configurations and in the one where the configuration is fixed, which corresponds to the original formulation of Bertsimas and Stock Patterson. For simplicity, we will assume that the capacities (airports and collapsed sectors) are constant over time.

7.1 Instance 1: nominal case

In this instance we consider a squared airspace composed of 16 elementary sectors (a, b, c, d, e, f, g, h, i, j, k, l, m, n, o and p) and containing 8 airports (DUB, CPH, LHR, AMS, CDG, FRA, ZRH and VIE). In Figure 7.1 we can see the graph associated with this airspace, as described in Section 6.2.

We consider 16 collapsed sectors (C01, C02, C03, C04, C05, C06, C07, C08, C09, C10, C11, C12, C13, C14, C15 and C16). This airspace is shown in Figure 7.2. We also consider three possible configurations (M01, M02 and M03), each formed by 8 collapsed sectors (some of these collapsed sectors are used in two configurations), as we can see in Figure 7.3. Each collapsed sector consists of two elementary sectors. Suppose the capacity of the collapsed sectors is equal to each other and is constant over time. We consider 36 time periods lasting 10 minutes each, for a total of 6 hours. We tested the model on five random instances, each with 256 flights. At least half of them have one

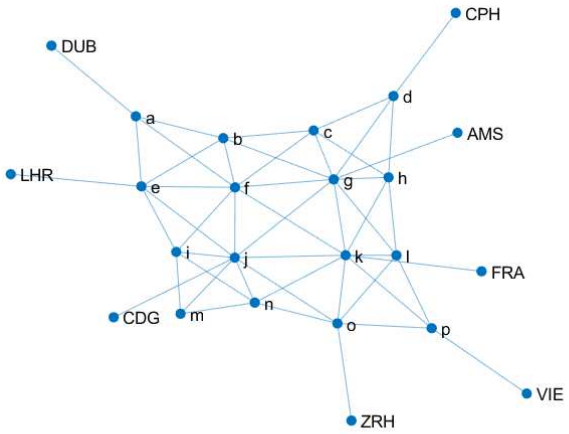


Figure 7.1: Graph associated with the airspace.

between DUB, LHR, AMS, CDG and FRA as their departure and arrival airports. For each of these five instances, we collected the optimal value of the objective function (if it exists) and the solving time, varying τ, which is the minimum number of consecutive time periods a configuration must remain active, in the set $\{6, 12, 36\}$ and the capacity of the collapsed sectors in the set $\{15, 16, \ldots, 35\}$. Note that if $\tau = 36$ it means that the configuration must remain fixed for all time. Hence, for each of the five instances we ran $3 \times (35 - 15 + 1) = 63$ tests. Now, by setting the capacity value of each collapsed sector and that of τ, we can read the following tables as follows. For a given τ and capacity configuration:

- # represents the number of times (out of 5) in which we have found an optimal solution;
- min. represents the minimum value of the objective function (if it exists) on the 5 tests carried out;
- max. represents the maximum value of the objective function (if it exists) on the 5 tests carried out;
- avg. represents the average of the value of the optimal functions found (the average is carried out on the number of optimal solutions found);
- avg. savings represents the average of the individual savings (expressed as a percentage) calculated with respect to the case with $\tau = 36$;
- avg. solving times represents the average time (in seconds) to solve the instance.

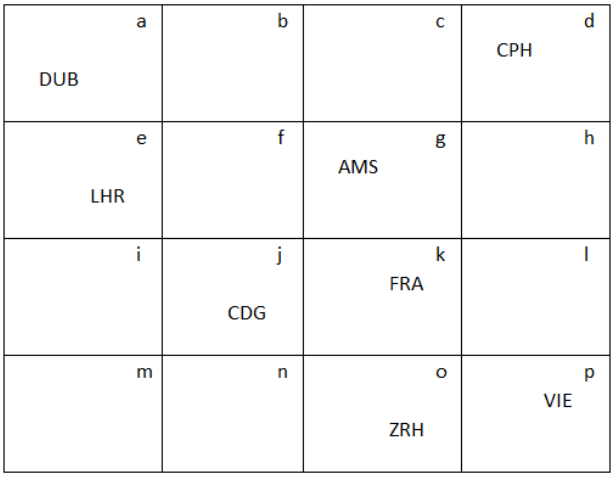


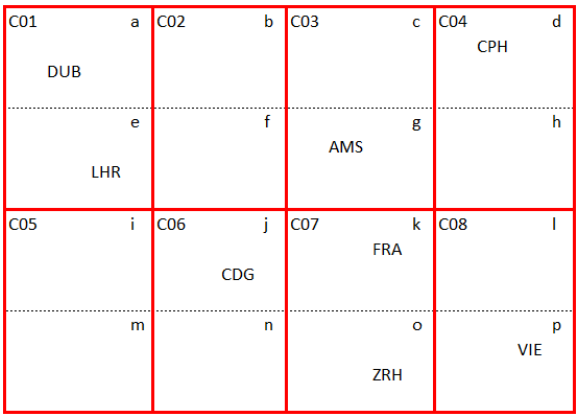
Figure 7.2: The airspace considered.

We note that giving the possibility to change the configuration saves on the total cost. In particular, by setting $\tau = 36$, the model returns the cost that we would get by fixing the best configuration, assuming we know what it is in advance.

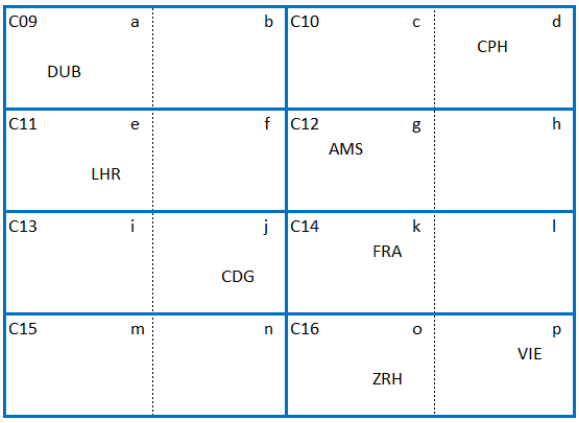
Table 7.1 indicates that the five instances considered are very different from each other. In fact, there are some where the cost of delay is very low and others where it is very high. This is due to a high degree of randomness in the generation of flights and related parameters.

In Table 7.2, we can see that in the first significant cases (with capacities 20 and 21) we obtain a saving of about 20% in the case $\tau = 6$ and of about 10% in the case $\tau = 12$. Furthermore, we observe that when the capacity increases, the choice of configuration becomes irrelevant, in particular if the capacity is at least 27 there is no difference between setting $\tau = 6$ or $\tau = 12$.

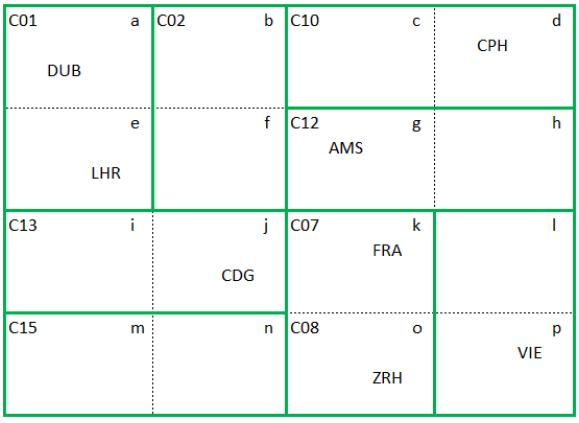
Finally, in Table 7.3 we can see that the running times are almost always less than one second.



(a)



(b)



(c)

Figure 7.3: The three configurations

	$\tau = 6$				$\tau = 12$				$\tau = 36$			
capacity	#	min.	max.	avg.	#	min.	max.	avg.	#	min.	max.	avg.
15	0	N/A	N/A	N/A	0	N/A	N/A	N/A	0	N/A	N/A	N/A
16	0	N/A	N/A	N/A	0	N/A	N/A	N/A	0	N/A	N/A	N/A
17	0	N/A	N/A	N/A	0	N/A	N/A	N/A	0	N/A	N/A	N/A
18	1	64	64	64.0	1	64	64	64.0	1	73	73	73.0
19	1	33	33	33.0	1	33	33	33.0	1	35	35	35.0
20	3	20	46	29.7	3	20	57	35.3	2	21	34	27.5
21	3	10	19	13.7	3	12	22	15.3	3	12	30	18.0
22	3	4	10	6.7	3	4	11	7.0	3	4	13	7.7
23	4	1	38	11.8	4	1	48	14.5	3	1	8	3.7
24	5	0	50	14.4	5	0	52	15.0	5	0	56	17.0
25	5	0	28	8.0	5	0	30	8.4	5	0	34	9.6
26	5	0	19	5.2	5	0	20	5.4	5	0	23	6.2
27	5	0	13	3.4	5	0	13	3.4	5	0	15	3.8
28	5	0	7	2.0	5	0	7	2.0	5	0	8	2.2
29	5	0	3	1.0	5	0	3	1.0	5	0	4	1.2
30	5	0	1	0.4	5	0	1	0.4	5	0	2	0.6
31	5	0	0	0.0	5	0	0	0.0	5	0	0	0.0
32	5	0	0	0.0	5	0	0	0.0	5	0	0	0.0
33	5	0	0	0.0	5	0	0	0.0	5	0	0	0.0
34	5	0	0	0.0	5	0	0	0.0	5	0	0	0.0
35	5	0	0	0.0	5	0	0	0.0	5	0	0	0.0

Table 7.1: General results for Instance 1.

		avg. savings (%)	
capacity	#	$\tau = 6$	$\tau = 12$
15	0	N/A	N/A
16	0	N/A	N/A
17	0	N/A	N/A
18	1	12.3	12.3
19	1	5.7	5.7
20	2	18.6	9.7
21	3	17.8	8.9
22	3	7.7	5.1
23	3	8.3	4.2
24	5	7.5	6.0
25	5	6.6	5.4
26	5	6.0	5.1
27	5	2.7	2.7
28	5	2.5	2.5
29	5	5.0	5.0
30	5	10.0	10.0
31	5	0.0	0.0
32	5	0.0	0.0
33	5	0.0	0.0
34	5	0.0	0.0
35	5	0.0	0.0

Table 7.2: Average savings (in %) compared to the fixed configuration test in the case of Instance 1.

	avg. solving times (s)		
capacity	$\tau = 6$	$\tau = 12$	$\tau = 36$
15	0.010	0.015	0.015
16	0.016	0.015	0.016
17	0.016	0.016	0.016
18	0.872	0.113	0.045
19	0.242	0.443	0.038
20	0.386	0.235	0.040
21	0.132	0.165	0.038
22	0.406	0.340	0.085
23	1.007	0.791	0.095
24	0.429	0.756	0.420
25	0.294	0.566	0.466
26	0.203	0.429	0.381
27	0.100	0.217	0.333
28	0.083	0.211	0.176
29	0.060	0.063	0.062
30	0.041	0.039	0.055
31	0.032	0.036	0.045
32	0.030	0.033	0.041
33	0.028	0.031	0.040
34	0.028	0.031	0.039
35	0.028	0.032	0.041

Table 7.3: Average solving times (in seconds) in the case of Instance 1.

7.2 Instance 2: weather disturbance from north to south

Let us consider the situation seen in the first instance and keep all the parameters, except the capacity of the collapsed sectors, which we assume they vary over time, simulating a weather disturbance that affects the airspace from North to South. Again, for each of these five instances, we collected the optimal value of the objective function (if it exists) and the solving time, varying τ in the set $\{6, 12, 36\}$ and the capacity of the nominal collapsed sectors in the set $\{20, 21, \ldots, 50\}$. In particular, with reference to Figure 7.2:

- $t \in \{1, \ldots, 9\}$: the weather disturbance affects the elementary sectors a, b, c and d. The respective collapsed sectors involved are C01, C02, C03, C04, C09 and C10. These sectors, while affected by the disturbance, have reduced capacity by 1/3 compared to the nominal capacity. Then, for example, if the nominal capacity of the collapsed sector C02 at time 4 is 25, its actual capacity is $\lfloor \frac{2}{3} 25 \rfloor = 16$;

- $t \in \{10, \ldots, 18\}$: the weather disturbance affects the elementary sectors e, f, g and h. The respective collapsed sectors involved are C01, C02, C03, C04, C11 and C12. These sectors, while affected by the disturbance, have reduced capacity by 1/3 compared to the nominal capacity;

- $t \in \{19, \ldots, 27\}$: the weather disturbance affects the elementary sectors i, j, k and l. The respective collapsed sectors involved are C05, C06, C07, C08, C13 and C14. These sectors, while affected by the disturbance, have reduced capacity by 1/3 compared to the nominal capacity;

- $t \in \{28, \ldots, 36\}$: the weather disturbance affects the elementary sectors m, n, o and p. The respective collapsed sectors involved are C05, C06, C07, C08, C15 and C16. These sectors, while affected by the disturbance, have reduced capacity by 1/3 compared to the nominal capacity.

As for Table 7.4, compared to the previous situation, in order to obtain feasible problems, we need, on average, larger nominal values of the capacity of the collapsed sectors.

Also, we can see in Table 7.4 that there is a strong variability between the five instances.

From Table 7.5, we observe that it is possible to save even more than 50% (both with $\tau = 6$ and with $\tau = 12$) if the capacity is between 29 and 31. The dynamic configuration of the airspace proves to be very efficient in this case of capacities varying over time. In the next section, we will try to understand if, assuming the direction of the weather disturbance changes we will obtain similar results.

Finally, also in this case we can observe in Table 7.6 that the running times are almost always less than one second.

	$\tau = 6$				$\tau = 12$				$\tau = 36$			
capacity	#	min	max	avg	#	min	max	avg	#	min	max	avg
20	0	N/A	N/A	N/A	0	N/A	N/A	N/A	0	N/A	N/A	N/A
21	0	N/A	N/A	N/A	0	N/A	N/A	N/A	0	N/A	N/A	N/A
22	0	N/A	N/A	N/A	0	N/A	N/A	N/A	0	N/A	N/A	N/A
23	1	36	36	36.0	1	46	46	46.0	1	46	46	46.0
24	1	18	18	18.0	1	24	24	24.0	1	24	24	24.0
25	1	18	18	18.0	1	23	23	23.0	1	23	23	23.0
26	2	10	50	30.0	2	11	52	31.5	1	11	11	11.0
27	2	5	28	16.5	2	6	28	17.0	2	6	59	32.5
28	2	5	28	16.5	2	6	28	17.0	2	6	59	32.5
29	3	1	47	21.3	3	1	59	25.3	2	3	28	15.5
30	3	0	27	12.0	3	0	27	12.0	3	1	33	16.7
31	3	0	27	12.0	3	0	27	12.0	3	1	33	16.7
32	3	0	16	6.3	3	0	18	7.0	3	0	19	9.0
33	4	0	25	8.8	4	0	25	9.0	4	0	28	10.5
34	4	0	25	8.8	4	0	25	9.0	4	0	28	10.5
35	4	0	12	4.5	4	0	12	4.8	4	0	15	5.8
36	5	0	44	10.4	5	0	44	10.4	5	0	45	11.4
37	5	0	44	10.4	5	0	44	10.4	5	0	45	11.4
38	5	0	28	6.0	5	0	28	6.0	5	0	30	7.2
39	5	0	19	3.8	5	0	19	3.8	5	0	20	4.4
40	5	0	19	3.8	5	0	19	3.8	5	0	20	4.4
41	5	0	13	2.6	5	0	13	2.6	5	0	13	2.6
42	5	0	7	1.4	5	0	7	1.4	5	0	8	1.6
43	5	0	7	1.4	5	0	7	1.4	5	0	8	1.6
44	5	0	3	0.6	5	0	3	0.6	5	0	4	0.8
45	5	0	1	0.2	5	0	1	0.2	5	0	2	0.4
46	5	0	1	0.2	5	0	1	0.2	5	0	2	0.4
47	5	0	0	0.0	5	0	0	0.0	5	0	0	0.0
48	5	0	0	0.0	5	0	0	0.0	5	0	0	0.0
49	5	0	0	0.0	5	0	0	0.0	5	0	0	0.0
50	5	0	0	0.0	5	0	0	0.0	5	0	0	0.0

Table 7.4: General results for Instance 2.

		avg. savings (%)	
capacity	#	$\tau = 6$	$\tau = 12$
20	0	N/A	N/A
21	0	N/A	N/A
22	0	N/A	N/A
23	1	21.7	0.0
24	1	25.0	0.0
25	1	21.7	0.0
26	1	9.1	0.0
27	2	34.6	26.3
28	2	34.6	26.3
29	2	54.8	54.8
30	3	54.0	54.0
31	3	54.0	54.0
32	3	26.1	22.6
33	4	30.0	27.7
34	4	30.0	27.7
35	4	33.6	30.0
36	5	9.3	9.3
37	5	9.3	9.3
38	5	17.3	17.3
39	5	21.0	21.0
40	5	21.0	21.0
41	5	0.0	0.0
42	5	2.5	2.5
43	5	2.5	2.5
44	5	5.0	5.0
45	5	10.0	10.0
46	5	10.0	10.0
47	5	0.0	0.0
48	5	0.0	0.0
49	5	0.0	0.0
50	5	0.0	0.0

Table 7.5: Average savings (in %) compared to the fixed configuration test in the case of Instance 2.

	avg. solving times (s)		
capacity	$\tau = 6$	$\tau = 12$	$\tau = 36$
20	0.014	0.023	0.024
21	0.782	0.241	0.052
22	0.744	0.301	0.109
23	0.399	0.362	0.060
24	0.164	0.426	0.257
25	0.136	0.326	0.274
26	0.319	0.464	0.171
27	0.233	0.244	0.146
28	0.242	0.304	0.122
29	0.277	0.300	0.110
30	0.210	0.324	0.083
31	0.197	0.351	0.074
32	0.212	0.331	0.186
33	0.171	0.237	0.231
34	0.149	0.240	0.205
35	0.416	0.445	0.100
36	0.268	0.442	0.485
37	0.187	0.349	0.469
38	0.160	0.253	0.268
39	0.185	0.243	0.237
40	0.182	0.214	0.232
41	0.214	0.190	0.201
42	0.103	0.146	0.162
43	0.081	0.135	0.107
44	0.052	0.111	0.093
45	0.049	0.055	0.070
46	0.047	0.052	0.069
47	0.046	0.050	0.067
48	0.045	0.048	0.060
49	0.044	0.049	0.060
50	0.043	0.047	0.058

Table 7.6: Average solving times (in seconds) in the case of Instance 2.

7.3 Instance 3: weather disturbance from west to east

Let us consider the situation seen in the first instance and keep all the parameters, except the capacity of the collapsed sectors, which we assume they vary over time, simulating a weather disturbance that affects our airspace from West to East. Again, for each of these five instances, we collected the same results as before, varying τ in the set $\{6, 12, 36\}$ and the capacity of the collapsed sectors in the set $\{20, 21, \ldots, 50\}$. Let us see the situation in detail, again with reference to Figure 7.2:

- $t \in \{1, \ldots, 9\}$: the weather disturbance affects the elementary sectors a, e, i and m. The respective collapsed sectors involved are C01, C05, C09, C11, C13 and C15. These sectors, while affected by the disturbance, have reduced capacity by 1/3 compared to the nominal capacity;

- $t \in \{10, \ldots, 18\}$: the weather disturbance affects the elementary sectors b, f, j and n. The respective collapsed sectors involved are C02, C06, C09, C11, C13 and C15. These sectors, while affected by the disturbance, have reduced capacity by 1/3 compared to the nominal capacity;

- $t \in \{19, \ldots, 27\}$: the weather disturbance affects the elementary sectors c, g, k and o. The respective collapsed sectors involved are C03, C07, C10, C12, C14 and C16. These sectors, while affected by the disturbance, have reduced capacity by 1/3 compared to the nominal capacity;

- $t \in \{28, \ldots, 36\}$: the weather disturbance affects the elementary sectors d, h, l and p. The respective collapsed sectors involved are C04, C08, C10, C12, C14 and C16. These sectors, while affected by the disturbance, have reduced capacity by 1/3 compared to the nominal capacity.

Regarding Tables 7.7 and 7.9, the same considerations made for the previous examples apply.

Instead, as regards Table 7.8, we notice that we get better results than the first example, but worse than the case in which the perturbation moved from West to East. This is due to the randomness in the generation of flight routes and related data, so we cannot assume that the two cases in which there is a weather disturbance are symmetrical.

	$\tau = 6$				$\tau = 12$				$\tau = 36$			
capacity	#	min	max	avg	#	min	max	avg	#	min	max	avg
20	0	N/A	N/A	N/A	0	N/A	N/A	N/A	0	N/A	N/A	N/A
21	0	N/A	N/A	N/A	0	N/A	N/A	N/A	0	N/A	N/A	N/A
22	0	N/A	N/A	N/A	0	N/A	N/A	N/A	0	N/A	N/A	N/A
23	0	N/A	N/A	N/A	0	N/A	N/A	N/A	0	N/A	N/A	N/A
24	0	N/A	N/A	N/A	0	N/A	N/A	N/A	0	N/A	N/A	N/A
25	1	33	33	33.0	1	33	33	33.0	0	N/A	N/A	N/A
26	2	17	36	26.5	2	17	49	33.0	1	27	27	27.0
27	3	10	29	19.3	3	10	29	21.0	3	14	37	26.7
28	3	7	29	18.3	3	7	29	20.0	3	9	37	25.0
29	4	5	46	19.3	4	5	46	19.8	3	5	20	13.3
30	4	3	27	10.3	4	3	27	10.5	4	3	33	13.3
31	4	2	27	10.0	4	2	27	10.3	4	2	33	13.0
32	5	1	63	16.8	5	1	63	17.2	5	1	63	18.4
33	5	0	36	9.2	5	0	36	9.4	5	0	36	10.0
34	5	0	36	9.2	5	0	36	9.4	5	0	36	10.0
35	5	0	22	5.6	5	0	22	5.8	5	0	22	6.0
36	5	0	15	3.6	5	0	15	3.6	5	0	15	3.6
37	5	0	15	3.6	5	0	15	3.6	5	0	15	3.6
38	5	0	10	2.2	5	0	10	2.2	5	0	10	2.2
39	5	0	7	1.4	5	0	7	1.4	5	0	7	1.4
40	5	0	7	1.4	5	0	7	1.4	5	0	7	1.4
41	5	0	4	0.8	5	0	4	0.8	5	0	5	1.0
42	5	0	2	0.4	5	0	2	0.4	5	0	3	0.6
43	5	0	2	0.4	5	0	2	0.4	5	0	3	0.6
44	5	0	1	0.2	5	0	1	0.2	5	0	2	0.4
45	5	0	0	0.0	5	0	0	0.0	5	0	1	0.2
46	5	0	0	0.0	5	0	0	0.0	5	0	1	0.2
47	5	0	0	0.0	5	0	0	0.0	5	0	0	0.0
48	5	0	0	0.0	5	0	0	0.0	5	0	0	0.0
49	5	0	0	0.0	5	0	0	0.0	5	0	0	0.0
50	5	0	0	0.0	5	0	0	0.0	5	0	0	0.0

Table 7.7: General results for Instance 3.

		avg. savings (%)	
capacity	#	$\tau = 6$	$\tau = 12$
20	0	N/A	N/A
21	0	N/A	N/A
22	0	N/A	N/A
23	0	N/A	N/A
24	0	N/A	N/A
25	0	N/A	N/A
26	1	37.0	37.0
27	3	25.7	21.2
28	3	23.6	19.1
29	3	15.0	11.7
30	4	17.0	15.0
31	4	17.0	15.0
32	5	17.4	15.3
33	5	21.8	20.0
34	5	21.8	20.0
35	5	22.9	20.0
36	5	0.0	0.0
37	5	0.0	0.0
38	5	0.0	0.0
39	5	0.0	0.0
40	5	0.0	0.0
41	5	4.0	4.0
42	5	6.7	6.7
43	5	6.7	6.7
44	5	10.0	10.0
45	5	20.0	20.0
46	5	20.0	20.0
47	5	0.0	0.0
48	5	0.0	0.0
49	5	0.0	0.0
50	5	0.0	0.0

Table 7.8: Average savings (in %) compared to the fixed configuration test in the case of Instance 3.

	avg. solving times (s)		
capacity	$\tau = 6$	$\tau = 12$	$\tau = 36$
20	0.009	0.015	0.017
21	0.018	0.017	0.016
22	0.016	0.018	0.016
23	0.069	0.096	0.020
24	0.266	0.375	0.213
25	0.201	0.348	0.112
26	0.328	0.345	0.219
27	0.364	0.236	0.141
28	0.122	0.218	0.152
29	0.309	0.356	0.214
30	0.353	0.593	0.122
31	0.401	0.577	0.148
32	0.422	0.304	0.191
33	0.199	0.267	0.168
34	0.143	0.229	0.164
35	0.091	0.236	0.181
36	0.061	0.115	0.202
37	0.068	0.101	0.142
38	0.044	0.071	0.188
39	0.042	0.069	0.079
40	0.035	0.064	0.079
41	0.033	0.054	0.064
42	0.032	0.035	0.060
43	0.030	0.035	0.048
44	0.031	0.034	0.053
45	0.029	0.034	0.050
46	0.027	0.032	0.046
47	0.026	0.034	0.049
48	0.027	0.030	0.044
49	0.028	0.030	0.045
50	0.029	0.030	0.045

Table 7.9: Average solving times (in seconds) in the case of Instance 3.

7.4 Linear programming relaxation

In order to compare the quality of the formulation, from a polyhedral point of view, to the one of Bertsimas and Stock Patterson, we have studied the linear relaxation of the problem, fixing a configuration among the three available, eliminating the variables y and the constraints connected to them. We tried to understand if, as for Bertsimas and Stock Patterson model, the formulation is almost always integral, and in fact the answer is negative. This seems to confirm the results in Section 5.6. Let us consider the situation in Instance 1, where we tested five random instances. Let us examine the first of these instances and fix the three configurations one at a time. In Table 7.10, we note that, by fixing the configuration M01, all the values of the objective function in the relaxed problem correspond to those of the integer problem, except for two cases, where we find 28.5 instead of 29 and 8.5 instead of 9. In brackets we report the percentage of fractional w variables with respect to the total. In general the new capacity constraint can remove the integrality of continuous relaxation. In Table 7.11, related to the fifth instance considered in Instance 1, we observe that, by fixing the configuration M02, all the values of the objective function in the relaxed problem correspond to those of the integer problem, except for one case, where we find 25.5 instead of 26. The same applies if we fix the configuration M03. In both cases the percentage of fractional variables w remains around 1%.

	M01		M02		M03	
capacity	IP	LR	IP	LR	IP	LR
≤ 20	N/A	N/A	N/A	N/A	N/A	N/A
21	N/A	N/A	30	30	N/A	N/A
22	N/A	N/A	13	13	N/A	N/A
23	29	28.5 (1.4%)	8	8	21	21
24	15	15	3	3	12	12
25	9	8.5 (0.6%)	1	1	7	7
26	5	5	0	0	5	5
27	4	4	0	0	4	4
28	3	3	0	0	3	3
29	2	2	0	0	2	2
30	1	1	0	0	1	1
≥ 31	0	0	0	0	0	0

Table 7.10: A first example of comparison between optimal solutions of the integer and relaxed problems, in the case of fixed configuration. In parentheses, the percentage of fractional variables, if other than zero.

	M01		M02		M03	
capacity	IP	LR	IP	LR	IP	LR
≤ 23	N/A	N/A	N/A	N/A	N/A	N/A
24	N/A	N/A	26	25.5 (1.4%)	26	25.5 (1.2%)
25	47	47	13	13	13	13
26	25	25	8	8	8	8
27	12	12	4	4	4	4
28	7	7	3	3	3	3
29	3	3	2	2	2	2
30	2	2	1	1	1	1
31	1	1	0	0	0	0
≥ 32	0	0	0	0	0	0

Table 7.11: A second example of comparison between optimal solutions of the integer and relaxed problems, in the case of fixed configuration. In parentheses, the percentage of fractional variables, if other than zero.

Chapter 8

Conclusions

The problem of air traffic flow management is very important and it has been addressed in several literature works. We focus on an approach based on Integer Linear Programming, and we have seen that, over the years, there has been an evolution in the way the problem has been treated. In fact, starting from just including ground holding in the models, airborne holding, speed control and the possibility of re-routing have been introduced. In this thesis we have explored a further strategy. In fact, in addition to ground holding, airborne holding and speed control, we have introduced the possibility of choosing the most appropriate airspace configuration over time, favoring the one that allows a less expensive management of the flow of air traffic. This idea arises from the fact that in general, since there are fewer controllers than the number of elementary sectors to be controlled, it may be convenient to merge groups of adjacent elementary sectors in order to consider larger sectors, known as collapsed sectors. There is no single way to merge elementary sectors. In our tests, we give some possible airspace configurations, giving the solver the possibility to choose the most appropriate ones over time. In this sense, starting from a well-known work by Bertsimas and Stock Patterson, we have provided a new Integer Linear Programming model and implemented it in AMPL. We have tested it on various random instances and we have seen that the results actually reassure us that we are getting improvements. We have seen that the dynamism of the configuration allows us to save up to more than 50% compared to the case in which the configuration of the airspace is fixed. Moreover, the results suggest that the model is more performing in the case where the capacity of the collapsed sectors is variable over time, for example because of weather disturbances. Finally, through the study of linear relaxation, we have shown that our formulation is not as strong as that of the Bertsimas and Stock Patterson model, from which we started. We have seen in particular that in some cases the number of fractional variables is different from zero and therefore a possible future development is to try to deepen the study of linear relaxation, towards the definition of valid inequalities to strengthen linear relaxation.

Appendix

In the following, we report the .dat AMPL file of an instance related to the case in which the presence of a weather disturbance insisting on the airspace is introduced. Then, we will report the .mod AMPL file implementing the model presented in Chapter 5 and a .run AMPL file including a script to test the instance on different combinations of capacity and parameter τ values.

atfm.dat

```
set MAPS := M01 M02 M03 ;

set FLIGHTS := f001 f002 f003 f004 f005 f006 f007 f008 f009 f010 f011 f012 f013 f014 f015
f016 f017 f018 f019 f020 f021 f022 f023 f024 f025 f026 f027 f028 f029 f030 f031 f032 f033
f034 f035 f036 f037 f038 f039 f040 f041 f042 f043 f044 f045 f046 f047 f048 f049 f050 f051
f052 f053 f054 f055 f056 f057 f058 f059 f060 f061 f062 f063 f064 f065 f066 f067 f068 f069
f070 f071 f072 f073 f074 f075 f076 f077 f078 f079 f080 f081 f082 f083 f084 f085 f086 f087
f088 f089 f090 f091 f092 f093 f094 f095 f096 f097 f098 f099 f100 f101 f102 f103 f104 f105
f106 f107 f108 f109 f110 f111 f112 f113 f114 f115 f116 f117 f118 f119 f120 f121 f122 f123
f124 f125 f126 f127 f128 f129 f130 f131 f132 f133 f134 f135 f136 f137 f138 f139 f140 f141
f142 f143 f144 f145 f146 f147 f148 f149 f150 f151 f152 f153 f154 f155 f156 f157 f158 f159
f160 f161 f162 f163 f164 f165 f166 f167 f168 f169 f170 f171 f172 f173 f174 f175 f176 f177
f178 f179 f180 f181 f182 f183 f184 f185 f186 f187 f188 f189 f190 f191 f192 f193 f194 f195
f196 f197 f198 f199 f200 f201 f202 f203 f204 f205 f206 f207 f208 f209 f210 f211 f212 f213
f214 f215 f216 f217 f218 f219 f220 f221 f222 f223 f224 f225 f226 f227 f228 f229 f230 f231
f232 f233 f234 f235 f236 f237 f238 f239 f240 f241 f242 f243 f244 f245 f246 f247 f248 f249
f250 f251 f252 f253 f254 f255 f256 ;

set AIRPORTS := DUB CPH LHR AMS CDG FRA ZRH VIE ;

set ELEMENTARY := a b c d e f g h i j k l m n o p ;
```

```
set COLLAPSED := C01 C02 C03 C04 C05 C06 C07 C08 C09 C10 C11 C12 C13 C14 C15 C16 ;

set PARTITIONS[M01] := C01 C02 C03 C04 C05 C06 C07 C08 ;
set PARTITIONS[M02] := C09 C10 C11 C12 C13 C14 C15 C16 ;
set PARTITIONS[M03] := C01 C02 C07 C08 C10 C12 C13 C15 ;

set BELONGING[C01] := a e ;
set BELONGING[C02] := b f ;
set BELONGING[C03] := c g ;
set BELONGING[C04] := d h ;
set BELONGING[C05] := i m ;
set BELONGING[C06] := j n ;
set BELONGING[C07] := k o ;
set BELONGING[C08] := l p ;
set BELONGING[C09] := a b ;
set BELONGING[C10] := c d ;
set BELONGING[C11] := e f ;
set BELONGING[C12] := g h ;
set BELONGING[C13] := i j ;
set BELONGING[C14] := k l ;
set BELONGING[C15] := m n ;
set BELONGING[C16] := o p ;

set TIMES := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 ;

set PATHS[f001] = LHR e i f a DUB ;
set PATHS[f002] = LHR e i j CDG ;
set PATHS[f003] = FRA k g d c b e LHR ;
set PATHS[f004] = LHR e i j CDG ;
set PATHS[f005] = FRA k n o j CDG ;
set PATHS[f006] = CDG j i e LHR ;
set PATHS[f007] = FRA k g f a DUB ;
set PATHS[f008] = AMS g f a DUB ;
set PATHS[f009] = CDG j i e LHR ;
set PATHS[f010] = FRA k g f a DUB ;
set PATHS[f011] = DUB a f g k FRA ;
set PATHS[f012] = LHR e i f a DUB ;
set PATHS[f013] = AMS g d c b e LHR ;
```

```
set PATHS[f014] = LHR e b c d g AMS ;
set PATHS[f015] = LHR e b c d g k FRA ;
set PATHS[f016] = AMS g k FRA ;
set PATHS[f017] = DUB a f j CDG ;
set PATHS[f018] = LHR e b c d g AMS ;
set PATHS[f019] = AMS g f j CDG ;
set PATHS[f020] = CDG j f g AMS ;
set PATHS[f021] = AMS g f j CDG ;
set PATHS[f022] = CDG j o n k FRA ;
set PATHS[f023] = LHR e b c d g AMS ;
set PATHS[f024] = FRA k g d c b e LHR ;
set PATHS[f025] = AMS g f a DUB ;
set PATHS[f026] = AMS g f a DUB ;
set PATHS[f027] = AMS g f a DUB ;
set PATHS[f028] = AMS g k FRA ;
set PATHS[f029] = CDG j f a DUB ;
set PATHS[f030] = FRA k n o j CDG ;
set PATHS[f031] = CDG j f a DUB ;
set PATHS[f032] = LHR e b c d g k FRA ;
set PATHS[f033] = FRA k g AMS ;
set PATHS[f034] = AMS g d c b e LHR ;
set PATHS[f035] = AMS g f j CDG ;
set PATHS[f036] = FRA k n o j CDG ;
set PATHS[f037] = DUB a f g k FRA ;
set PATHS[f038] = FRA k g f a DUB ;
set PATHS[f039] = LHR e b c d g k FRA ;
set PATHS[f040] = LHR e b c d g AMS ;
set PATHS[f041] = DUB a f g k FRA ;
set PATHS[f042] = DUB a f g k FRA ;
set PATHS[f043] = DUB a f i e LHR ;
set PATHS[f044] = AMS g f j CDG ;
set PATHS[f045] = LHR e i j CDG ;
set PATHS[f046] = LHR e i j CDG ;
set PATHS[f047] = DUB a f g k FRA ;
set PATHS[f048] = AMS g f j CDG ;
set PATHS[f049] = AMS g k FRA ;
set PATHS[f050] = DUB a f i e LHR ;
set PATHS[f051] = CDG j f a DUB ;
set PATHS[f052] = DUB a f g AMS ;
```

```
set PATHS[f053] = AMS g f j CDG ;
set PATHS[f054] = DUB a f i e LHR ;
set PATHS[f055] = DUB a f g AMS ;
set PATHS[f056] = CDG j i e LHR ;
set PATHS[f057] = AMS g f a DUB ;
set PATHS[f058] = AMS g f j CDG ;
set PATHS[f059] = LHR e i j CDG ;
set PATHS[f060] = FRA k g AMS ;
set PATHS[f061] = AMS g k FRA ;
set PATHS[f062] = CDG j f a DUB ;
set PATHS[f063] = FRA k n o j CDG ;
set PATHS[f064] = FRA k g d c b e LHR ;
set PATHS[f065] = LHR e i f a DUB ;
set PATHS[f066] = AMS g f j CDG ;
set PATHS[f067] = LHR e b c d g k FRA ;
set PATHS[f068] = LHR e i j CDG ;
set PATHS[f069] = LHR e i j CDG ;
set PATHS[f070] = CDG j f a DUB ;
set PATHS[f071] = CDG j f a DUB ;
set PATHS[f072] = AMS g k FRA ;
set PATHS[f073] = DUB a f g k FRA ;
set PATHS[f074] = CDG j i e LHR ;
set PATHS[f075] = DUB a f j CDG ;
set PATHS[f076] = DUB a f i e LHR ;
set PATHS[f077] = AMS g d c b e LHR ;
set PATHS[f078] = CDG j o n k FRA ;
set PATHS[f079] = FRA k n o j CDG ;
set PATHS[f080] = AMS g d c b e LHR ;
set PATHS[f081] = AMS g f a DUB ;
set PATHS[f082] = DUB a f g k FRA ;
set PATHS[f083] = FRA k g f a DUB ;
set PATHS[f084] = FRA k g f a DUB ;
set PATHS[f085] = AMS g f a DUB ;
set PATHS[f086] = AMS g d c b e LHR ;
set PATHS[f087] = CDG j f a DUB ;
set PATHS[f088] = DUB a f g k FRA ;
set PATHS[f089] = AMS g f a DUB ;
set PATHS[f090] = CDG j o n k FRA ;
set PATHS[f091] = AMS g f j CDG ;
```

```
set PATHS[f092] = DUB a f i e LHR ;
set PATHS[f093] = CDG j f a DUB ;
set PATHS[f094] = AMS g f a DUB ;
set PATHS[f095] = FRA k g f a DUB ;
set PATHS[f096] = AMS g f j CDG ;
set PATHS[f097] = LHR e i f a DUB ;
set PATHS[f098] = CDG j i e LHR ;
set PATHS[f099] = CDG j f g AMS ;
set PATHS[f100] = AMS g f j CDG ;
set PATHS[f101] = AMS g f j CDG ;
set PATHS[f102] = LHR e b c d g AMS ;
set PATHS[f103] = CDG j f a DUB ;
set PATHS[f104] = DUB a f g k FRA ;
set PATHS[f105] = LHR e i f a DUB ;
set PATHS[f106] = AMS g d c b e LHR ;
set PATHS[f107] = LHR e i j CDG ;
set PATHS[f108] = FRA k n o j CDG ;
set PATHS[f109] = CDG j f g AMS ;
set PATHS[f110] = CDG j f g AMS ;
set PATHS[f111] = FRA k g f a DUB ;
set PATHS[f112] = LHR e b c d g AMS ;
set PATHS[f113] = CDG j i e LHR ;
set PATHS[f114] = LHR e i j CDG ;
set PATHS[f115] = LHR e i j CDG ;
set PATHS[f116] = AMS g d c b e LHR ;
set PATHS[f117] = FRA k g AMS ;
set PATHS[f118] = CDG j f a DUB ;
set PATHS[f119] = AMS g d c b e LHR ;
set PATHS[f120] = AMS g f a DUB ;
set PATHS[f121] = LHR e i f a DUB ;
set PATHS[f122] = LHR e i f a DUB ;
set PATHS[f123] = DUB a f i e LHR ;
set PATHS[f124] = FRA k g f a DUB ;
set PATHS[f125] = CDG j f a DUB ;
set PATHS[f126] = DUB a f i e LHR ;
set PATHS[f127] = AMS g f a DUB ;
set PATHS[f128] = CDG j f g AMS ;
set PATHS[f129] = FRA k g d c b e LHR ;
set PATHS[f130] = ZRH o n k g d CPH ;
```

```
set PATHS[f131] = VIE p l g AMS ;
set PATHS[f132] = CDG j f g AMS ;
set PATHS[f133] = ZRH o n k g AMS ;
set PATHS[f134] = LHR e i j o ZRH ;
set PATHS[f135] = VIE p o ZRH ;
set PATHS[f136] = ZRH o n k FRA ;
set PATHS[f137] = AMS g f a DUB ;
set PATHS[f138] = CPH d g k FRA ;
set PATHS[f139] = DUB a f g AMS ;
set PATHS[f140] = DUB a f g AMS ;
set PATHS[f141] = LHR e i j o ZRH ;
set PATHS[f142] = FRA k g f a DUB ;
set PATHS[f143] = CPH d g AMS ;
set PATHS[f144] = AMS g k FRA ;
set PATHS[f145] = CDG j o n k FRA ;
set PATHS[f146] = FRA k n o j CDG ;
set PATHS[f147] = DUB a f g l p VIE ;
set PATHS[f148] = LHR e b c d CPH ;
set PATHS[f149] = FRA k n o ZRH ;
set PATHS[f150] = LHR e i j o ZRH ;
set PATHS[f151] = DUB a f g l p VIE ;
set PATHS[f152] = CPH d g AMS ;
set PATHS[f153] = CDG j i e LHR ;
set PATHS[f154] = FRA k n o ZRH ;
set PATHS[f155] = FRA k g d c b e LHR ;
set PATHS[f156] = VIE p l g AMS ;
set PATHS[f157] = CPH d g AMS ;
set PATHS[f158] = LHR e b c d g k FRA ;
set PATHS[f159] = AMS g k FRA ;
set PATHS[f160] = FRA k g l p VIE ;
set PATHS[f161] = AMS g f j CDG ;
set PATHS[f162] = CDG j f a DUB ;
set PATHS[f163] = CPH d g k FRA ;
set PATHS[f164] = LHR e i f a DUB ;
set PATHS[f165] = ZRH o n k FRA ;
set PATHS[f166] = CPH d g k n o ZRH ;
set PATHS[f167] = AMS g d CPH ;
set PATHS[f168] = FRA k n o j CDG ;
set PATHS[f169] = AMS g k n o ZRH ;
```

```
set PATHS[f170] = CPH d g AMS ;
set PATHS[f171] = ZRH o j CDG ;
set PATHS[f172] = CDG j i e LHR ;
set PATHS[f173] = VIE p l g d CPH ;
set PATHS[f174] = LHR e b c d g k FRA ;
set PATHS[f175] = DUB a f g l p VIE ;
set PATHS[f176] = CPH d c b e LHR ;
set PATHS[f177] = VIE p l g f a DUB ;
set PATHS[f178] = CDG j i e LHR ;
set PATHS[f179] = ZRH o j i e LHR ;
set PATHS[f180] = AMS g k FRA ;
set PATHS[f181] = ZRH o j CDG ;
set PATHS[f182] = DUB a f j CDG ;
set PATHS[f183] = VIE p l g d c b e LHR ;
set PATHS[f184] = DUB a f g AMS ;
set PATHS[f185] = CDG j o ZRH ;
set PATHS[f186] = FRA k g f a DUB ;
set PATHS[f187] = CDG j f g d CPH ;
set PATHS[f188] = CDG j o p VIE ;
set PATHS[f189] = LHR e b c d g AMS ;
set PATHS[f190] = FRA k n o ZRH ;
set PATHS[f191] = DUB a f i e LHR ;
set PATHS[f192] = AMS g f j CDG ;
set PATHS[f193] = DUB a f g AMS ;
set PATHS[f194] = DUB a f j o ZRH ;
set PATHS[f195] = ZRH o n k g AMS ;
set PATHS[f196] = AMS g f j CDG ;
set PATHS[f197] = CDG j o p VIE ;
set PATHS[f198] = CPH d g k n o ZRH ;
set PATHS[f199] = DUB a f i e LHR ;
set PATHS[f200] = FRA k g l p VIE ;
set PATHS[f201] = ZRH o p VIE ;
set PATHS[f202] = ZRH o n k g AMS ;
set PATHS[f203] = CPH d g l p VIE ;
set PATHS[f204] = CPH d g k n o ZRH ;
set PATHS[f205] = AMS g d c b e LHR ;
set PATHS[f206] = AMS g k FRA ;
set PATHS[f207] = VIE p l g k FRA ;
set PATHS[f208] = DUB a f i e LHR ;
```

```
set PATHS[f209] = CDG j o ZRH ;
set PATHS[f210] = CDG j f a DUB ;
set PATHS[f211] = DUB a f j o ZRH ;
set PATHS[f212] = CPH d g f a DUB ;
set PATHS[f213] = AMS g f a DUB ;
set PATHS[f214] = DUB a f g d CPH ;
set PATHS[f215] = DUB a f i e LHR ;
set PATHS[f216] = DUB a f i e LHR ;
set PATHS[f217] = LHR e b c d CPH ;
set PATHS[f218] = LHR e b c d g k FRA ;
set PATHS[f219] = DUB a f g d CPH ;
set PATHS[f220] = DUB a f i e LHR ;
set PATHS[f221] = ZRH o j CDG ;
set PATHS[f222] = ZRH o n k g AMS ;
set PATHS[f223] = ZRH o j f a DUB ;
set PATHS[f224] = FRA k g AMS ;
set PATHS[f225] = ZRH o j i e LHR ;
set PATHS[f226] = VIE p l g f a DUB ;
set PATHS[f227] = AMS g f j CDG ;
set PATHS[f228] = CDG j o p VIE ;
set PATHS[f229] = CPH d g k n o ZRH ;
set PATHS[f230] = VIE p l g d c b e LHR ;
set PATHS[f231] = ZRH o p VIE ;
set PATHS[f232] = DUB a f j CDG ;
set PATHS[f233] = LHR e b c d g AMS ;
set PATHS[f234] = CPH d g l p VIE ;
set PATHS[f235] = CDG j o p VIE ;
set PATHS[f236] = DUB a f g k FRA ;
set PATHS[f237] = ZRH o n k FRA ;
set PATHS[f238] = CDG j f g AMS ;
set PATHS[f239] = AMS g f j CDG ;
set PATHS[f240] = ZRH o p VIE ;
set PATHS[f241] = CPH d c b e LHR ;
set PATHS[f242] = VIE p l g k FRA ;
set PATHS[f243] = CPH d g k FRA ;
set PATHS[f244] = CDG j f g d CPH ;
set PATHS[f245] = FRA k g d c b e LHR ;
set PATHS[f246] = AMS g f j CDG ;
set PATHS[f247] = VIE p l g d CPH ;
```

```
set PATHS[f248] = CDG j f g d CPH ;
set PATHS[f249] = VIE p l g d c b e LHR ;
set PATHS[f250] = CDG j o p VIE ;
set PATHS[f251] = CPH d g l p VIE ;
set PATHS[f252] = LHR e b c d g k FRA ;
set PATHS[f253] = CPH d g AMS ;
set PATHS[f254] = ZRH o n k g AMS ;
set PATHS[f255] = AMS g d CPH ;
set PATHS[f256] = ZRH o n k FRA ;

param D default 8 ;
param A default 8 ;

param cg default 1 ;
param ca default 3 ;

\* We define the minimum time each flight must spend in each elementary sector included in
its path.  Obviously it will spend 0 periods of time in the departure airport, since once
it has taken off, the flight is immediately considered in the first elementary sector (the
one in which the departure airport is contained).  In other cases, we randomly generate these
times.  In particular, we will ensure that if a flight has to cross an entire sector it will
take longer than if the flight lands at an airport in the sector */

for {f in FLIGHTS} {
  let l[f,first(PATHS[f])] := 0;
  for {j in PATHS[f]:  ord0(j,PATHS[f]) > 1 and ord0(j,PATHS[f]) < card(PATHS[f])} {
    if (prev(j,PATHS[f]) in ELEMENTARY and next(j,PATHS[f]) in ELEMENTARY) then {
      let l[f,j] := floor(Uniform(3, 5));
    }
    else {
    let l[f,j] := floor(Uniform(1, 3));
    }
  }
}

\* We randomly generate the maximum delay time of each flight, calculate the flight duration
based on how much time each flight spends in each sector, and finally randomly generate the
take-off and landing times of each flight */
```

```
for {f in FLIGHTS} {
  let delay[f] := floor(Uniform(0, 3));
  let duration[f] := 0;
  for {j in PATHS[f]:  ord0(j,PATHS[f]) > 1 and ord0(j,PATHS[f]) < card(PATHS[f])} {
    let duration[f] := duration[f] + l[f,j];
  }
  let d[f] := floor(Uniform(first(TIMES), last(TIMES) - delay[f] - duration[f] + 1));
  let r[f] := d[f] + duration[f];
}

\* We calculate the set of feasible times for each flight to arrive in a given elementary
sector */

for {f in FLIGHTS} {
  let tmin[f,first(PATHS[f])] := d[f];
  let tmax[f,first(PATHS[f])] := tmin[f,first(PATHS[f])] + delay[f];
  for {j in PATHS[f] :  ord0(j,PATHS[f]) > 1} {
    let tmin[f,j] := tmin[f,prev(j,PATHS[f])] + l[f,prev(j,PATHS[f])];
    let tmax[f,j] := tmin[f,j] + delay[f];
  }
}
```

atfm.mod

```
# set of configurations
set MAPS;

# set of flights
set FLIGHTS;

# set of airports
set AIRPORTS;
# set of elementary sectors
set ELEMENTARY;

# set of collapsed sectors
set COLLAPSED;

# set of collapsed sectors whose union forms a certain configuration
set PARTITIONS {MAPS};

# set of elementary sectors whose union forms a certain collapsed sector
```

```
set BELONGING {COLLAPSED};

# set of time periods
set TIMES ordered;

# path of a certain flight
set PATHS {FLIGHTS} ordered;

# D[k,t] = departure capacity of airport k at time t
param D {AIRPORTS, TIMES};

# A[k,t] = arrival capacity of airport k at time t
param A {AIRPORTS, TIMES};

# S[c,t] = capacity of collapsed sector h at time t
param S {COLLAPSED, TIMES};

# l[f,j] = number of time units that flight f must spend in elementary sector j
param l {f in FLIGHTS, j in PATHS[f]:  ord0(j,PATHS[f]) < card(PATHS[f])};

# delay[f] = maximum delay allowed for the flight f
param delay {FLIGHTS};

# d[f] = scheduled departure time of flight f
param d {FLIGHTS};

# r[f] = scheduled arrival time of flight f
param r {FLIGHTS};

# duration[f] = r[f] - d[f] param duration {FLIGHTS};

# cg[f] = cost of holding flight f on the ground for one unit of time
param cg {FLIGHTS};

# ca[f] = cost of holding flight f in the air for one unit of time
param ca {FLIGHTS};

# tmin[f,j] = first period of time for flight f to enter to elementary sector j
param tmin {f in FLIGHTS, PATHS[f]};

# tmax[f,j] = last period of time for flight f to enter to elementary sector j
param tmax {f in FLIGHTS, PATHS[f]};

# theoretical maximum number of flights that could be in the collapsed sector h at time t
param C {COLLAPSED, TIMES};

# minimum number of consecutive time periods a configuration must remain active
param tau > 0 integer;

# counter of variables w
```

```
param counter_w;

# counter of variables y
param counter_y;

# counter of variables w that are integer
param counter_w_int;

# counter of variables w that are fractional
param counter_w_frac;

# counter of variables y that are integer
param counter_y_int;

# counter of variables y that are fractional
param counter_y_frac;

# variables w
var w {f in FLIGHTS, j in PATHS[f],t in TIMES:t >= tmin[f,j] and t <= tmax[f,j]} binary;

# variables y
var y {m in MAPS, t in TIMES} binary;

# number of time periods of ground delay
var g {f in FLIGHTS};

# number of time periods of airborne delay
var a {f in FLIGHTS};

# objective function
minimize fun :  sum{f in FLIGHTS} ((cg[f]-ca[f])*(sum{t in TIMES : t >= tmin[f,first(PATHS[f])]
and t <= tmax[f,first(PATHS[f])]} (t*(w[f,first(PATHS[f]),t]-(if t-1 >= tmin[f,first(PATHS[f])]
and t-1 <= tmax[f,first(PATHS[f])] then w[f,first(PATHS[f]),t-1] else if t-1
<= tmin[f,first(PATHS[f])]-1 then 0 else 1))))+ca[f]*(sumt in TIMES : t >= tmin[f,last(PATHS[f])]
and t <= tmax[f,last(PATHS[f])] (t*(w[f,last(PATHS[f]),t]-(if t-1 >= tmin[f,last(PATHS[f])]
and t-1 <= tmax[f,last(PATHS[f])] then w[f,last(PATHS[f]),t-1] else if t-1
<= tmin[f,last(PATHS[f])]-1 then 0 else 1))))+(ca[f]-cg[f])*d[f]-ca[f]*r[f]);

# basic constraints
s.t.  v0 {f in FLIGHTS, j in PATHS[f], t in TIMES: t = tmax[f,j]}:
w[f,j,t] = 1;

# Constraints on airport departure capacity
s.t.  v1 {k in AIRPORTS, t in TIMES}:
sum {f in FLIGHTS: k = first(PATHS[f])} ((if t >= tmin[f,k] and t <= tmax[f,k] then w[f,k,t]
else if t <= tmin[f,k]-1 then 0 else 1)-(if t-1 >= tmin[f,k] and t-1 <= tmax[f,k] then w[f,k,t-1]
else if t-1 <= tmin[f,k]-1 then 0 else 1)) <= D[k,t];
```

```
# Constraints on airport arrival capacity
s.t.  v2 {k in AIRPORTS, t in TIMES}:
sum {f in FLIGHTS: k = last(PATHS[f])} ((if t >= tmin[f,k] and t <= tmax[f,k] then w[f,k,t]
else if t <= tmin[f,k]-1 then 0 else 1)-(if t-1 >= tmin[f,k] and t-1 <= tmax[f,k] then w[f,k,t-1]
else if t-1 <= tmin[f,k]-1 then 0 else 1)) <= A[k,t];

# Constraints on the capacity of collapsed sectors
s.t.  v3 {m in MAPS, h in PARTITIONS[m], t in TIMES}:
sum {j in BELONGING[h]} sum {f in FLIGHTS: ord0(j,PATHS[f]) > 1 and ord0(j,PATHS[f])
< card(PATHS[f])} ((if t >= tmin[f,j] and t <= tmax[f,j] then w[f,j,t] else if t <= tmin[f,j]-1
then 0 else 1)-(if t >= tmin[f,next(j,PATHS[f])] and t <= tmax[f,next(j,PATHS[f])] then
w[f,next(j,PATHS[f]),t] else if t <= tmin[f,next(j,PATHS[f])]-1 then 0 else 1))
<= S[h,t]+C[h,t]*(1-y[m,t]);

# Constraints on connectivity between elementary sectors
s.t.  v4 {f in FLIGHTS, j in PATHS[f], t in TIMES: ord0(j,PATHS[f]) >= 1 and ord0(j,PATHS[f])
< card(PATHS[f]) and t >= tmin[f,j] and t <= tmax[f,j]}:
((if t+l[f,j] >= tmin[f,next(j,PATHS[f])] and t+l[f,j] <= tmax[f,next(j,PATHS[f])] then
w[f,next(j,PATHS[f]),t+l[f,j]] else if t+l[f,j] <= tmin[f,next(j,PATHS[f])]-1 then 0 else
1)-w[f,j,t]) <= 0;

# Constraints on connectivity in time (5)
s.t.  v5 {f in FLIGHTS, j in PATHS[f], t in TIMES: t >= tmin[f,j] and t <= tmax[f,j]}:
(w[f,j,t]-(if t-1 >= tmin[f,j] and t-1 <= tmax[f,j] then w[f,j,t-1] else if t-1 <= tmin[f,j]-1
then 0 else 1)) >= 0;

# Constraints on the choice of configuration (7)
s.t.  v7 {t in TIMES}:
sum{m in MAPS} y[m,t] = 1;

# Constraints on maintaining a configuration (8)
s.t.  v8 {m in MAPS, t in TIMES, u in t+1..min(t+tau-1,last(TIMES))}:
(y[m,t]-(if t-1 < first(TIMES) then 0 else y[m,t-1])) <= (if u > last(TIMES) then 0 else y[m,u]);
```

atfm.run

```
reset;
model atfm.mod;
data atfm.dat;
option presolve 0;
option solver cplexamp;
```

```
option relax_integrality 0;
for {periods in {6, 12, 36}} {
  printf "\n*********************************\n" >atfm.txt;
  let tau := periods;
  for {capacity in 20..50} {
    let counter_w := 0;
    let counter_y := 0;
    let counter_w_int := 0;
    let counter_w_frac := 0;
    let counter_y_int := 0;
    let counter_y_frac := 0;
    for {h in COLLAPSED} {
      for {t in TIMES} {
        let S[h,t] := capacity;
      }
    }
    let {h in COLLAPSED, t in TIMES: (h = 'C01' or h = 'C02' or h = 'C03' or h = 'C04' or
h = 'C09' or h = 'C10') and t >= 1 and t <= 9} S[h,t] := floor(2/3 * S[h,t]);
    let {h in COLLAPSED, t in TIMES: (h = 'C01' or h = 'C02' or h = 'C03' or h = 'C04' or
h = 'C11' or h = 'C12') and t >= 10 and t <= 18} S[h,t] := floor(2/3 * S[h,t]);
    let {h in COLLAPSED, t in TIMES: (h = 'C05' or h = 'C06' or h = 'C07' or h = 'C08' or
h = 'C13' or h = 'C14') and t >= 19 and t <= 27} S[h,t] := floor(2/3 * S[h,t]);
    let {h in COLLAPSED, t in TIMES: (h = 'C05' or h = 'C06' or h = 'C07' or h = 'C08' or
h = 'C15' or h = 'C16') and t >= 28 and t <= 36} S[h,t] := floor(2/3 * S[h,t]);
    for {h in COLLAPSED} {
      for {t in TIMES} {
        let C[h,t] := (sum {j in BELONGING[h]} sum {f in FLIGHTS: ord0(j,PATHS[f]) > 1 and
ord0(j,PATHS[f]) < card(PATHS[f])} 1) - S[h,t];
      }
    }
    solve;
    for {f in FLIGHTS, j in PATHS[f], t in TIMES: t >= tmin[f,j] and t <= tmax[f,j]} {
      let counter_w := counter_w + 1;
      if (abs(w[f,j,t]-round(w[f,j,t])) > 1e-6) then {
        let counter_w_frac := counter_w_frac + 1;
      }
      else {
        let counter_w_int := counter_w_int + 1;
      }
```

```
    }
    for {m in MAPS, t in TIMES} {
      let counter_y := counter_y + 1;
      if (abs(y[m,t]-round(y[m,t])) > 1e-6) then {
        let counter_y_frac := counter_y_frac + 1;
      }
      else {
        let counter_y_int := counter_y_int + 1;
      }
    }
    let {f in FLIGHTS} g[f] := sum{t in TIMES: t >= tmin[f,first(PATHS[f])] and t
<= tmax[f,first(PATHS[f])]} (t*(w[f,first(PATHS[f]),t]-(if t-1 >= tmin[f,first(PATHS[f])]
and t-1 <= tmax[f,first(PATHS[f])] then w[f,first(PATHS[f]),t-1] else if t-1
<= tmin[f,first(PATHS[f])]-1 then 0 else 1)))-d[f];
    let {f in FLIGHTS} a[f] := sum{t in TIMES: t >= tmin[f,last(PATHS[f])] and t
<= tmax[f,last(PATHS[f])]} (t*(w[f,last(PATHS[f]),t]-(if t-1 >= tmin[f,last(PATHS[f])] and
t-1 <= tmax[f,last(PATHS[f])] then w[f,last(PATHS[f]),t-1] else if t-1
<= tmin[f,last(PATHS[f])]-1 then 0 else 1)))-r[f]-g[f];
    printf "\ntau = %d, ", tau >atfm.txt;
    printf "capacity = %d, ", capacity >atfm.txt;
    printf "fun = %f, ", fun >atfm.txt;
    printf "ground = %f, ", sum{f in FLIGHTS} g[f] >atfm.txt;
    printf "air = %f, ", sum{f in FLIGHTS} a[f] >atfm.txt;
    printf "counter_w = %d, ", counter_w >atfm.txt;
    printf "counter_w_int = %d, ", counter_w_int >atfm.txt;
    printf "counter_w_frac = %d, ", counter_w_frac >atfm.txt;
    printf "counter_y = %d, ", counter_y >atfm.txt;
    printf "counter_y_int = %d, ", counter_y_int >atfm.txt;
    printf "counter_y_frac = %d, ", counter_y_frac >atfm.txt;
    printf "BB_nodes = %d, ", num0(substr(solve_message,match(solve_message,"s\n")+2)) >atfm.txt;
    printf "time_solve = %f.  \n", _total_solve_time >atfm.txt;
  }
}
```

References

[1] A. Agustín, A. Alonso-Ayuso, L. F. Escudero, C. Pizarro, et al. On air traffic flow management with rerouting. Part I: Deterministic case. *European Journal of Operational Research*, 219(1):156–166, 2012.

[2] A. Agusín, A. Alonso-Ayuso, L. F. Escudero, C. Pizarro, et al. On air traffic flow management with rerouting. Part II: Stochastic case. *European Journal of Operational Research*, 219(1):167–177, 2012.

[3] AMPL. `https://ampl.com/products/ampl/`

[4] G. Andreatta, L. Capanna, L. De Giovanni e L. Righi. Rapporto sul progetto di ricerca Follow-up di Optiframe. *Rapporto Interno Consorzio Futuro in Ricerca*, 2018.

[5] G. Andreatta, A. R. Odoni and O. Richetta. Models for the Ground-Holding Problem. *Large-Scale Computation and Information Processing in Air Traffic Control*. Springer-Verlag, 1993.

[6] ATAG. `https://www.icao.int/sustainability/Documents/AVIATION-BENEFITS-2019-web.pdf`, 2019.

[7] D. Bertsimas, G. Lulli, and A. Odoni. An integer optimization approach to large-scale air traffic flow management. *Operations Research*, 59(1):211–227, 2011.

[8] D. Bertsimas and S. Stock Patterson. The air traffic flow management problem with enroute capacities. *Operations research*, 46(3):406–422, 1998.

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

[10] Y. Chynchenko, V. Kharchenko. Concept of air traffic flow and capacity management in European region. *Proceedings of National Aviation University*, (56): 7–12, 2013

[11] M. Conforti, G. Cornuejols, and G. Zambelli. *Integer Programming*. Springer International Publishing, 2014.

[12] CPLEX. https://www.ibm.com/it-it/products/ilog-cplex-optimization-studio/details

[13] EUROCONTROL. Performance Review Report (PRR) 2019 https://www.eurocontrol.int/sites/default/files/2020-06/eurocontrol-prr-2019.pdf, 2020.

[14] F. D. Fomeni, G. Lulli, and K. Zografos. An optimization model for assigning 4d-trajectories to flights under the tbo concept. *Twelfth USA/Europe Air Traffic Management Research and Development Seminar (ATM2017)*, pages 26–30, 2017.

[15] M. P. Helme. Reducing air traffic delay in a space-time network. *1992 IEEE International Conference on Systems, Man, and Cybernetics*, pages 236–242. IEEE, 1992.

[16] MATLAB. https://it.mathworks.com/products/matlab.html

[17] Pan Li. https://www.math.cuhk.edu.hk/course_builder/1415/math3220/L5.pdf, 2015.