Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea Magistrale in Matematica

A tighter mathematical programming formulation for an air traffic flow management problem with dynamic selection of the airspace configuration

Supervisor
Prof. Luigi De Giovanni

Candidate
Federico Baù

Matriculation Number
2006654

15 December 2023

Academic Year: 2022-2023

"Everything is hard before it is easy."

— Johann Wolfgang von Goethe

Abstract

The aim of this thesis is to find a tighter mathematical programming formulation for an air traffic flow management problem with dynamic selection of the airspace configuration. In particular, we start from an Integer Linear Programming model proposed in literature, we improve some of its constraints and we devise some valid inequalities. We thus obtain a tighter formulation that allows us to better approximate the region of the feasible solutions of the problem. The importance of finding a tight formulation lies in the fact that the methods for solving an integer linear programming problem are, in principle, more efficient if the formulation is tight. The ideal would be to find the tightest possible formulation, i.e. the convex hull of the solutions of the problem, but in practice this is very difficult to obtain. The formulation that we propose is not at all close to the ideal one, but it is significantly tighter than the starting one as shown by the computational experiments we carried out.

Contents

List of Figures

List of Tables

List of Acronyms

ANSP . . . . . Air Navigation Service Provider

ATAG . . . . . Air Transport Action Group

ATC . . . . . Air Traffic Control

ATFCM . . . . . Air Traffic Flow and Capacity Management

ATFM . . . . . Air Traffic Flow Management

ATM . . . . . Air Traffic Management

ATS . . . . . Air Traffic Services

GHG . . . . . Greenhouse Gas

IATA . . . . . International Air Transport Association

ICAO . . . . . International Civil Aviation Organization

ILP . . . . . Integer Linear Programming

LP . . . . . Linear Programming

LPR . . . . . Linear Programming Relaxation

MILP . . . . . Mixed Integer Linear Programming

NMOC . . . . . Network Manager Operations Centre

PANS . . . . . Procedures for Air Navigation Services

SAF . . . . . Sustainable Aviation Fuel

TBO . . . . . Trajectory Based Operations

1 Introduction

This first chapter is devoted to briefly describe the motivations behind our work, pointing out the increasing importance of air traffic flow management (with particular attention to the European scenario). A brief summary of the content of each chapter of this thesis as well as its main contributions are also provided.

1.1 Motivations

In our increasingly connected world, aviation plays an extremely important role, being the fastest worldwide transportation network. Through aviation, people can travel all around the world and international business is extremely facilitated. Of course, it is essential to ensure that flights are carried out safely. It is therefore also necessary that the flow of air traffic is managed efficiently and in such a way as to avoid potentially dangerous situations. Air traffic flow management (ATFM), described in Section 2.1, addresses this issue. But before delving into this more specific aspect, in this chapter we give a more general description of the motivations behind our work.

To understand better the importance of aviation in our society let us see some data. Before doing so it is important to remark that, due to a number of restrictions related to COVID-19 pandemic, the number of worldwide scheduled flights dropped down in 2020 and has not reached yet pre-COVID levels. According to the Air Transport Action Group (ATAG) this

might happen in 2024 [6]. Figure 1.1 clearly shows how the number of passengers carried by air in the European Union dropped down in 2020 (after having almost always increased in the previous 11 years).

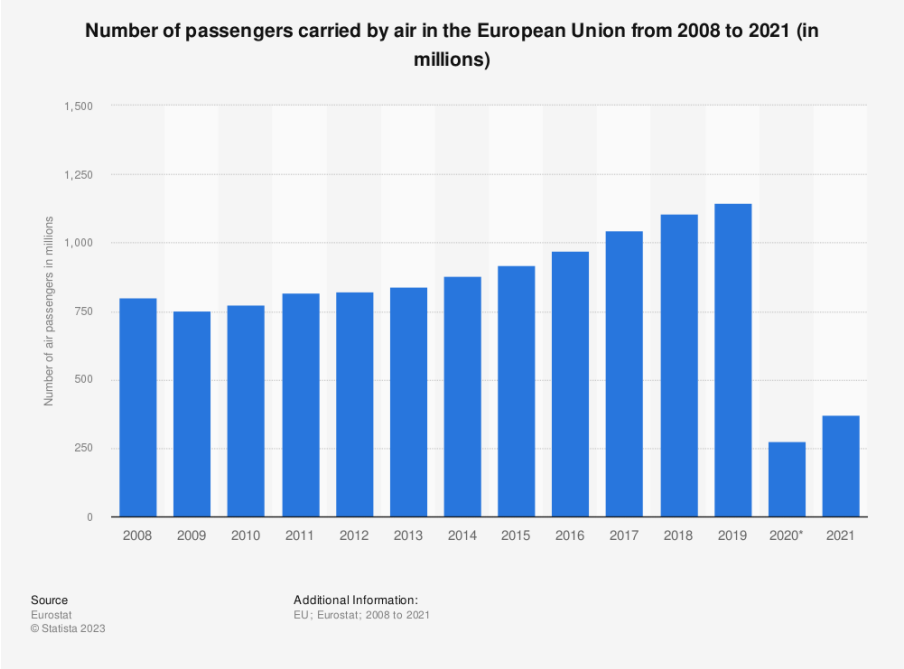

Figure 1.1: Number of passengers carried by air in the EU from 2008 to 2021 [24].

In 2018, always in accordance with [6]: "Air transport supported 13.5 million jobs and $991 billion in European economic activity. That is 3.6% of all employment and 4.4% of all GDP in European countries in 2018". Of these 13.5 million people, around 2.7 million were directly employed in the aviation sector. Furthermore in Europe, in 2018, there were: 9,112,303 flights; approximately 1.2 billion passengers; around 10 million tonnes of cargo. Moreover, before COVID-19 pandemic, air travel in Europe was "expected to continue to grow at about 2.1% per year over the next two decades" [6]. However, despite the effects of COVID-19 pandemic, it is still expected that there will be around 16 million flights in Europe by 2050, a 44% increase compared to the number of flights (11.1 million) performed in 2019 in Europe (this estimate corresponds approximately to an average growth of 1.2% per year) [12]. These data should give the reader an idea of the importance of the growing air transport industry.

According to [14], in 2022 in Europe, "en-route ATFM delays increased to levels observed

during the capacity crisis in 2018/19, notwithstanding traffic levels still being lower than in 2019". Furthermore, "10.4% of all flights were en-route ATFM delayed and the average en-route ATFM delay reached 1.74 minutes". From 2015 to 2021 the percentage of en-route delayed flights never reached 10% (it was 3.9% in 2015). Also the average en-route ATFM delay per flight (1.74 minutes in 2022, for a total en-route ATFM delay of approximately 15.9 million minutes) is a record from 2015 to 2022 (equalling 2018). In 2015 the average en-route ATFM delay per flight was 0.73 minutes. In 2022, however, punctuality deteriorated to the worst performance on record. In fact, "in July 2022, slightly more than half (52%) of the flights departed within 15 minutes of their scheduled departure time" [14]. All this happened despite "in 2022, traffic in the EUROCONTROL area increased by 48.3% compared to 2021 but remained 16.7% below the level of 2019". It is also worth noting that "the average delay per departure in 2022 was 17.6 minutes, with a peak of over 25 minutes in June and July. This represents a 33% increase compared to the average delay in 2019". The war in Ukraine (impacting on airspace availability) is one of the causes which contributed to 2022 bad performances [14].

Air transport industry has also a significant environmental impact. In 2019 aviation produced 4.3% of greenhouse gas (GHG) emissions in Europe and in 2018, globally, around 2.5% (approximately 1.04 billion tonnes, a quantity doubled since 1987 and quadrupled since 1966) of the total CO_2 emissions are attributable to air transport industry (at that time the highest percentage value ever recorded) [14, 21]. In Europe, despite having decreased by 57% in 2020 (due to the reduction in the number of flights provoked by COVID-19 pandemic), CO_2 emissions "increased again in line with traffic recovery reaching 81% of the level of 2019 in 2022" [14]. At global level, the International Civil Aviation Organization (ICAO) "adopted a Long-Term Aspirational Goal (LTAG) for international aviation of net-zero carbon emissions by 2050. International air transport association (IATA) member airlines approved in 2021 a resolution committing them to achieving net-zero carbon emissions from their operations by 2050 (Fly Net Zero)" [14]. Without normal fleet renewal and without any advancements in the use of sustainable aviation fuels (SAFs), in Europe CO_2 emissions might reach 279 million tonnes in 2050 (they were around 150 million tonnes in 2005 and around 200 million in 2019) [12].

All the previous data (and considerations) point out the importance of air traffic flow management. A good ATFM permits to limit delays, costs (keeping an airplane more time en-route increases costs related to fuel) and CO_2 (and, more in general, GHG) emissions. All of this must be done, obviously, in the safest possible way (structural capacities of airports must never

be exceeded and air traffic must always be under control).

1.2 Structure of the thesis and main contributions

Chapter 1: Introduction The first chapter is devoted to briefly describe the motivations behind our work, pointing out the increasing importance of air traffic flow management (with particular attention to the European scenario). A brief summary of the content of each chapter of this thesis as well as its main contributions are also provided.

Chapter 2: The Air Traffic Flow Management Problem In this second chapter, we introduce the ATFM problem. In particular, we describe it, outline the related planning of operations, see four possible ATFM strategies and present the specific ATFM problem treated in this thesis.

Chapter 3: Theoretical framework and computational tools In the first four sections of the third chapter, we recall some theoretical notions that will play a fundamental role in this thesis. In particular, the model we use to address the ATFM problem (as well as the models proposed in Operations Research literature that we recall in Chapter 4) is an integer linear programming model. For this reason, we review the notions of linear programming, integer linear programming and linear programming relaxation. We then illustrate two methods for solving integer linear programming problems and a technique, based on valid inequalities, which permits to tighten their formulation. To conclude this chapter, in the fifth section, we briefly describe the optimization software used in order to obtain the computational results presented in this thesis.

Chapter 4: State of the art In the first section of the fourth chapter, we briefly recall some of the most relevant integer linear programming models for ATFM present in literature. In the second section, we instead describe in detail the model presented in [26], which is the basis of the work developed in this thesis.

Chapter 5: A tighter formulation for the ATFM problem with dynamic selection of the airspace configuration The fifth chapter is the core of the thesis. In the first section, we prove the redundancy of the class of valid inequalities proposed in [26]. In the following three sections, we present our contributions in order to tighten the formulation presented in [26]. Our contributions are presented here from a theoretical viewpoint (the results of our computational

experiments are exhibited in Chapter 7) and basically consist in replacing some constants and adding new classes of valid inequalities. The new formulation thus obtained is explicitly written down in the fifth section. Finally, in the sixth section, we provide some estimates regarding the size of the new formulation.

Chapter 6: Model implementation and instance generation In the first section of this sixth chapter, we describe how the instances used for our computational experiments were generated. To conclude the chapter, in the second section, we illustrate the actual implementation of the model (described through the new formulation) in AMPL, an algebraic modelling language for mathematical programming.

Chapter 7: Computational results In the seventh chapter, we report some results, relating to our computational experiments, which show "how much" our contributions (presented in Chapter 5) allow us to tighten the formulation proposed in literature.

Chapter 8: Conclusions In the last chapter, we draw the conclusions of our work and propose some ideas for possible improvements and potential future developments.

Appendix In the appendix, we report the MATLAB file flights.m used to generate the spatial trajectories of the flights present in the instances on which we carried out our computational experiments. We also report the atfm.dat, atfm.mod and atfm.run files associated with Flights 10 as examples of the .dat, .mod and .run files used to implement the model presented in this thesis in AMPL.

2

The Air Traffic Flow Management Problem

In this second chapter, we will introduce the ATFM problem. In particular, we will describe it, outline the related planning of operations, see four possible ATFM strategies and present the specific ATFM problem treated in this thesis.

2.1 Air Traffic Flow Management

According to [18], ATFM is a service established with the aim of contributing to a safe, orderly and expeditious flow of air traffic by ensuring that air traffic control (ATC) capacity is used to the maximum extent possible, and that the traffic volume is compatible with the capacities declared by the appropriate air traffic services (ATS) authority. Its purpose is, therefore, to optimize air traffic flow according to ATC capacity without exceeding the capacities declared by the competent ATS authority (thus ensuring airlines to operate safe and efficient flights). ATFM is sometimes also called Air Traffic Flow and Capacity Management (ATFCM) and is part of the more general Air Traffic Management (ATM), which encompass also other type of services (for example air traffic services such as flight information services, alerting services, ...). The EUROCONTROL Network Manager Operations Centre (NMOC) plays today "a pivotal role in managing, streamlining and improving air traffic operations in Europe, with a strong network-minded approach" [13]. It constantly monitors the balance between traffic load and airspace capacity.

The ATFM activities are divided into the three following phases [19]:

Strategic phase During this phase, which begins about one year and ends about one week before the flight takes place, 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". Note that "this also includes avoiding imbalances between capacity and demand for events taking place a week or more in the future" [19].

Pre-tactical phase It takes place from one to six days before real time operations. In this phase the MNOC staff must "coordinate the definition of a daily plan aimed at optimising the overall ATM network performance and minimizing delay and cost, after a collaborative decision making process involving operational partners" 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" [19].

Tactical phase It occurs the day of operations (the same day in which the flight takes place). During this phase the daily plan, made the day before, is monitored and updated by the MNOC staff. "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 minimize delays. Flights taking place on that day receive the benefit of the flow management service, which includes inter alia 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" [19].

2.2 The problem and four possible strategies

As we have already seen, the ATFM problem consists in optimizing air traffic flow without exceeding the capacities declared by the competent ATS authority. Let us provide a more precise description of the problem. Let us consider an airport network. We have a certain number of scheduled flights, each of them with its own flight path. Every scheduled flight path is given by: the departure airport and the scheduled departure time; a sequence of sectors that (except for changes) will be crossed while travelling by air and, for each one of these sectors, the scheduled entry time; the destination airport and the scheduled arrival time. It is important to note that every flight path is defined by a 4D space-time trajectory. In this thesis, however, we will consider (for the sake of simplicity) every flight path as a 3D space-time trajectory since, as in [26],

we will not take into account the flight level dimension. In the ATFM problem faced in thesis, we want to minimize a cost defined by the weighted sum of all flights' delays without exceeding the various capacities. There are also ATFM problems that require to minimize other cost functions (see, for instance, [1] or [2]). To be more clear, let us explore the concept of cost in the ATFM problem considered in this thesis, as well as the concepts of sectors, capacities and airspace configurations.

2.2.1 The cost function

The ATFM problem faced in this thesis requires to minimize a weighted sum of all flights' delays without exceeding the various capacities described in Subsection 2.2.2. To understand better the previous sentence we need to do some considerations. First of all, let us observe that the delay of a flight can be seen as a deviation from the time component of its scheduled space-time trajectory. In the ATFM problem faced in this thesis only time deviations with respect to scheduled departure and arrival times matter. This means that, in the cost function we want to minimize, it does not matter if there is a delay in the intermediate stages between takeoff and landing, the only important thing is whether the flight leaves/arrives in time or not. However there are different ways in which a flight can accumulate delay. One of the most important ways is through the application of the so-called ground-holding policy, which consists in postponing the scheduled departure of the flight. Another possible strategy, which can be implemented for example in the event that the destination airport is momentarily congested, is given by the airborne-holding policy and consists in preventing the aircraft, ready to land, from landing for a certain amount of time. Other ways in which a flight can accumulate delay are, for example, traveling at lower speed (thus increasing time spent en-route) or changing the scheduled route. All these strategies will be presented in more detail in Subsection 2.2.3. In light of these considerations, it is easy to understand why the considered sum is weighted since, for example, 15 minutes of air-borne holding, requiring extra fuel consumption, are more expensive than 15 minutes of ground-holding.

2.2.2 Sectors, airspace configurations and capacities

Sectors are contiguous 3D portions of the airspace. We distinguish two different types of sectors (see [4]): elementary sectors and collapsed sectors. The former are pairwise disjoint and can be interpreted as the "atoms" in which the airspace is divided (see Figure 2.1), whereas the latter, which are not necessarily pairwise disjoint, can be thought as a sort of "molecules". In

particular, each collapsed sector is a contiguous 3D portion of the airspace given by the union of one or more elementary sectors (see Figure 2.2).

Figure 2.1: Three elementary sectors in northeastern Italy [4].

An airspace configuration (or, more simply, a configuration) is a partition of the set of elementary sectors into subsets corresponding to collapsed sectors. Since, generally, there exist several possible different partitions, it is usually possible to achieve many different configurations starting from the same elementary sectors. In this work, as in [26], the possibility of changing the chosen (or active) configuration during the day is contemplated. Airspace configurations play a fundamental role in this thesis since, by selecting the most convenient configuration, it is possible to reduce the value of the cost function. To better understand the importance of configurations, let us first introduce the concept of capacities.

At any time of the day, to each elementary sector is associated a capacity, which represents the maximum number of aircraft allowed to be simultaneously in that sector (in that moment of the day). The capacity of an elementary sector can be changed over time by the competent ATS authority (due, for example, to bad weather conditions). Similarly, at any time of the day every airport has a departure capacity and an arrival capacity (and also these can be changed over time). Moreover, every collapsed sector has a capacity that depends on the capacities of the contained elementary sectors and other factors related to, e.g., its shape or position in the airspace.

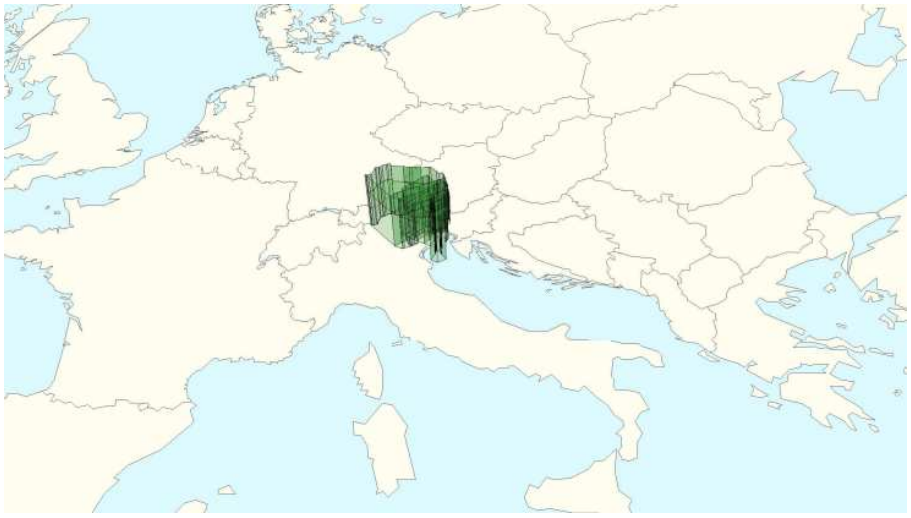

Figure 2.2: A collapsed sector in northeastern Italy [4].

Obviously the competent ATS authority establishes the various capacities in order to guarantee a high degree of safety (avoiding, in this way, dangerous situations). It is clear that the constraints given by the various limited capacities are some of the main causes of delays. The reason why we distinguish two different types of sectors is that, in reality, usually there are not enough controllers to monitor traffic on every elementary sector. For this reason we will assume that the controllers monitor the air traffic only on the collapsed sectors belonging to the temporarily active airspace configuration, as well as controlling also the traffic situation during the various takeoffs and landings phases, guaranteeing that for every airport, at any time of the day, both the departure and the arrival capacity will not be exceeded. In general, the capacity of a collapsed sector represents the maximum number of aircraft that the controllers of that collapsed sector can simultaneously monitor without much difficulty (the presence, at the same time, of too many flights inside a collapsed sector is a major risk factor). As one would expect, the capacity of a collapsed sector depends on many factors. Some of them are the size of the sector, its shape and the weather conditions affecting it.

In light of the previous considerations, we can finally understand the importance of airspace configurations. Suppose, for example, that there are only two contiguous elementary sectors that would be quite congested at the same time if we respected the scheduled flight paths. The choice to use, at that critical moment, an airspace configuration in which these two elementary sectors are part of the same collapsed sector could lead to exceeding the capacity of the latter,

thus forcing some flights to accumulate delays. Instead, by choosing a configuration such that the two elementary sectors are not part of the same collapsed sector, it may be possible to respect all the capacities of the collapsed sectors belonging to the configuration in question.

2.2.3 Possible ATFM strategies

In this subsection, we present four possible strategies to face the ATFM problem (describing also pros and cons of the first two approaches). The aim of these strategies is to avoid traffic congestion. An optimal balance between these different approaches is an essential element in ATFM. Our model will take into account only the first three strategies (re-routing is not an available option).

Ground-holding Applying ground-holding policy to a flight means deciding to postpone its departure (in this way we anticipate the delay on the ground at the departure airport). This policy does not require extra fuel consumption, so it is particularly advantageous both from an economic and an environmental viewpoint. It is also a particularly safe policy (since the aircraft is not in the air). The major drawback is that, deciding to anticipate the delay on the ground at the departure airport (in order to avoid future air traffic congestion), a certain amount of delay is immediately produced. This means that, if for some reason the expected air traffic congestion will not happen (for example because meanwhile, due to an improvement of weather conditions, the collapsed sectors capacities are increased), we have already caused an avoidable delay.

Airborne-holding Applying airborne-holding policy to a flight means deciding to postpone its landing, keeping in this way (for a certain amount of time) the aircraft airborne over the departure airport. Airborne-holding results in additional fuel consumption (this implies higher economical costs and a greater impact on the environment) and is more dangerous than ground-holding (since the aircraft is in the air). If on one hand airborne-holding presents these disadvantages, on the other hand this policy does not cause any avoidable delay (unlike ground-holding policy).

Speed control A flight is subject to speed control if it is possible to adjust its cruise speed. In this way the aircraft can reach a certain sector (or the destination airport) before or after an expected air traffic congestion. In our model speed control can be used only to slow down a flight. In other words, we assume the impossibility of increasing the scheduled speed of a flight. This means that it will not be possible to reduce a flight's

delay (and it will not be possible to increase the speed of an aircraft in order to avoid traffic congestion).

Re-routing Re-routing a flight means deciding to change its route (the flight will take another route, different from the scheduled one). This one is also a very important strategy to avoid air traffic congestion. In our model, however, re-routing is not an available option.

2.3 The ATFM problem with dynamic airspace configuration

In this section, we formalize the details of the specific ATFM problem treated in this thesis, which is the same one addressed in [26].

2.3.1 Elementary sectors, collapsed sectors and airspace configurations

As we have already seen, sectors are contiguous 3D portions of the airspace. They are divided into two different types: elementary sectors and collapsed sectors. Elementary sectors are a sort of "atoms" in which the airspace is divided. Let us denote with $\mathcal{J}$ the set whose elements are the elementary sectors. The elements of this set are pairwise disjoint and their union is the entire airspace. In other words, elementary sectors form a partition of the airspace. Each airport is not a sector, but it is entirely contained within an elementary sector.

Each collapsed sector is a contiguous 3D portion of the airspace given by the union of one or more elementary sectors (but not necessarily every union of one or more elementary sectors resulting in a contiguous 3D portion of the airspace is a collapsed sector). Let us denote with $\mathcal{H}$ the set whose elements are the collapsed sectors. For instance, let us consider $\mathcal{J} = \{a, b, c, d\}$ and $\mathcal{H} = \{C_1, C_2, C_3, C_4\}$, where $C_1 = \{a, b\}$, $C_2 = \{c, d\}$, $C_3 = \{a, c\}$ and $C_4 = \{b, d\}$.

An airspace configuration (or, more simply, a configuration) is a partition of the set $\mathcal{J}$ into subsets of $\mathcal{H}$. With respect to the previous example, there are only two possible airspace configurations: the first one is given by C_1 and C_2; the second one is given by C_3 and C_4. Since, in general, there are not enough controllers to monitor air traffic on every elementary sector,

choosing an airspace configuration is very useful (it permits to monitor traffic only on the collapsed sectors belonging to the active configuration). A list of configurations $\mathcal{M}$ is a set whose elements are airspace configurations (but not necessarily every possible airspace configuration belongs to $\mathcal{M}$). For instance, with respect to the example above, a list of configurations is given by $\mathcal{M} = \{M_1, M_2\}$, where $M_1 = \{C_1, C_2\}$ and $M_2 = \{C_3, C_4\}$.

2.3.2 Time representation

A long time period is divided into a set of smaller time periods of equal duration. For instance, we can decide to divide a 3 hour (180 minutes) period into 36 smaller time periods of equal duration. In this way by $t = 1$ we mean the time interval $[0, 5)$, by $t = 1$ we mean the time interval $[5, 10)$, and so on. Note that, realistically speaking, it is not possible to choose the most appropriate airspace configuration at each time period. For this reason, as in [26], let us 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.

2.3.3 Flights

The spatial trajectory of each flight is described (for some integer $n > 1$) by a sequence of the type $ABC, a_1, \ldots, a_n, XYZ$, where ABC is the departure airport, $a_1, \ldots, a_n$ are, in order, the elementary sectors that the flight will cross and XYZ is the destination airport. For each flight we assume that ABC (which is entirely contained within a_1) and XYZ (which is entirely contained within a_n) are different airports and that a_1 and a_n are different elementary sectors. For our purposes it will be much more practical (and more precise) to describe the spatial trajectory of a flight through a list containing elementary sectors rather than using a list containing collapsed sectors (see [26]).

2.3.4 Capacities

As we have already seen, at any time period t to each elementary sector is associated a capacity. However, since only some collapsed sectors are monitored in the model, we can directly consider the capacities associated to the collapsed sectors. We can assume (although it is not actually necessary) that the capacity of a collapsed sector is given by the sum of the capacities of the elementary sectors contained in that collapsed sector. The capacity of a collapsed sector at the time period t represents the maximum number of aircraft allowed to be in that sector

at time t. The capacity of a collapsed sector can change over time. Similarly, recalling Subsection 2.2.2, at time t every airport has a departure capacity (representing the maximum number of flights that are allowed to take off from the airport in the time period t) and an arrival capacity (representing the maximum number of flights that are allowed to land at the airport in the time period t).

2.3.5 Delays

As we have already seen, each flight has a scheduled flight path and, therefore, a scheduled departure time and a scheduled arrival time. Moreover, each flight has a scheduled entry time for every elementary sector that will be crossed while travelling by air. In our problem the available options to face the ATFM problem are ground-holding, airborne-holding and speed control (re-routing is not allowed). However, speed control can be applied to a flight only to make it slow down. As a consequence, it is not possible to use speed control in order to reduce a flight's delay. This means that, for each flight and for every elementary sector in its spatial trajectory, the scheduled time to cross that elementary sector coincides with the minimum time necessary (for that flight) to go through that sector. For each flight f let us denote with Δ_f the maximum delay in arrival allowed for flight f. Therefore, each elementary sector j belonging to the spatial trajectory of f must be reached within a very precise time window. The lower bound of this time window is given by the scheduled departure time plus the sum of all the scheduled time periods that the flight is expected to spend in the elementary sectors preceding j. The upper bound of this time window is given by the lower bound plus Δ_f. In this way we are able to guarantee the maximum flexibility on how a flight can spread the allowed delay.

2.3.6 Problem definition

Based on the elements presented so far, we define the ATFM problem with dynamic selection of the airspace configuration as follows. Given a set of scheduled flights and a list of configurations, the goal is to minimize a weighted sum of all flights' delays without exceeding the capacities of the various airports and collapsed sectors belonging to the active configuration. To do this, we can combine the strategies presented in Subsection 2.2.3 with the possibility of choosing, and possibly updating every now and then, which airspace configuration to adopt.

3

Theoretical framework and computational tools

Before presenting the model for the ATFM problem with dynamic selection of the airspace configuration, we recall the basic required theoretical and computational background. In the first four sections of this third chapter, we will recall some theoretical notions that will play a fundamental role in this thesis. In particular, the model we will use to address the ATFM problem (as well as the models proposed in Operations Research literature that we will recall in the next chapter) is an integer linear programming model. For this reason, we will review the notions of linear programming, integer linear programming and linear programming relaxation. We will then illustrate two methods for solving integer linear programming problems and a technique, based on valid inequalities, which permits to tighten their formulation. To conclude this chapter, in the fifth section, we will briefly describe the optimization software used in order to obtain the computational results presented in this thesis.

The theoretical part of this chapter will be mainly based on [10]. We point out that, in this thesis, $0 \in \mathbb{N}$ by convention.

3.1 Integer Linear Programming problems

First of all, let us recall what a linear programming (LP) problem is. A LP problem is a problem that can be written in the form

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

where $c, x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ (A, b, c are given parameters, whereas x is a vector of variables). The *feasible region* of (3.1) is given by the set $S := \{x \in \mathbb{R}^n : \ Ax \leq b, \ x \geq 0\}$, which is the set consisting of the points satisfying the *constraints* of (3.1). Each element of S is called a *solution* of the LP problem. We say that $\overline{x} \in S$ is an *optimal solution* of (3.1) if (and only if) $\forall x \in S$ we have that $c^T x \leq c^T \overline{x}$ ($\overline{x}$ maximizes the *objective function* $c^T x$ in S). The real number $c^T \overline{x}$ is called the *optimal value* of (3.1). If $S = \varnothing$, the problem admits no solution (we have an *infeasible* LP problem) and, obviously, no optimal solution (the feasible region is empty). Otherwise, if $S \neq \varnothing$, meaning that the LP problem is *feasible*, the feasible region is given by the intersection of finitely many half-spaces and, therefore, it is a convex *polyhedron* in $\mathbb{R}^n$, which can be unbounded or bounded. If the polyhedron is unbounded, then the LP problem may have an optimal solution (and consequently an optimal value) or may not ($\forall M \in \mathbb{R}$ we can find $x_M \in S$ such that $c^T x_M > M$, so the problem is *unbounded* and does not have an optimal value). If the polyhedron is bounded, then the LP problem has an optimal solution. As it is well known, if a LP problem has an optimal value, then there exists an optimal solution which is at a vertex. To find an optimal solution which is at a vertex (if the LP problem has an optimal value), we can use the well-known *simplex algorithm*, which can also be used to understand if a LP problem is infeasible or unbounded. This famous algorithm, developed by George Bernard Dantzig in 1947, is currently used to solve large-scale problems in all sorts of application areas [10]. The simplex algorithm, although very fast in practice (it is still nowadays a fundamental computational instrument to solve LP problems) does not guarantee a polynomial running time. However, it is possible to solve a LP problem with a polynomial algorithm using, for example, the *Karmarkar's interior point algorithm*. Therefore the complexity class of LP problems is **P** [10].

As we have already seen, there exist very efficient algorithms to solve LP problems in practice.

This is not true for integer linear programming (ILP) problems. An ILP problem is a problem that can be written in the form

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

where $c, x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ (A, b, c are given parameters, whereas x is a vector of variables). In the exact same way as we did for LP problems, we can define the notions of *objective function, constraints, feasible region, solution, optimal solution* and *optimal value* (the main difference is that the constraints also impose the integrality of the variables). Also an ILP can be *infeasible* or *feasible* and (if it is feasible) *unbounded* or not. However, the feasible region of a feasible ILP is not a polyhedron, but a discrete set of points (with integer coordinates) in $\mathbb{R}^n$. Sometimes only some variables are required to be integral, in this case we speak about mixed integer linear programming (MILP) problems. In this thesis, we will consider a particular type of ILP problems, in which all the variables are *binary* (restricted to take value 0 or 1). Binary variables are sometimes also called *binary decision variables*. Since ILP problems are in the **NP-hard** complexity class, they are in practice much more difficult to solve than LP problems [10].

The linear programming relaxation (LPR) of an ILP problem is the LP problem obtained removing the integrality constraints from the ILP problem. Let us see a simple example to clarify the situation. Let us consider the following ILP problem (taken from [10]):

$$\begin{aligned} \max \quad & 5.5x_1 + 2.1x_2 \\ \text{subject to} \quad & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 17 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z}. \end{aligned} \tag{3.3}$$

On one hand in Figure 3.1 are shown the feasible region of (3.3) (given by the black points and the red point) and its optimal solution (the red point). On the other hand, with respect to the LPR of the ILP problem (3.3), Figure 3.1 represents its feasible region (the light blue polyhedron, which in this case is a polygon) and its optimal solution (the yellow point). In fact, it is easy to check that the feasible region of the LPR of our ILP problem is the quadri-

lateral whose vertices have coordinates: $(0,0)$, $(0,2)$, $(2.125,0)$ and $(1.3,3.3)$. Since, as we already know, there is an optimal solution of the LPR at a vertex (which, in this case, is the only optimal solution of (3.3) as one can easily check, for example, with some basic considerations involving the level sets of the cost function), we can compute the values assumed by the cost function on the vertices of the polygon to find the optimal solution in question. The values are respectively: 0, 4.2, 11.6875 and 14.08. Therefore, the optimal solution of the LPR of our problem is $(1.3,3.3)$ (the yellow point in Figure 3.1) and the optimal value is 14.08. The feasible region of problem (3.3) consists of the polygon's points with integer coordinates. These 8 points (see the black points and the red point in Figure 3.1) have the following coordinates: $(0,0)$, $(0,1)$, $(0,2)$, $(1,0)$, $(1,1)$, $(1,2)$, $(1,3)$ and $(2,0)$. Here the cost function take, respectively, the following values: 0, 2.1, 4.2, 5.5, 7.6, 9.7, 11.8 and 11. Therefore, the optimal solution of our problem is $(1,3)$ (the red point in Figure 3.1) and the optimal value is 11.8.

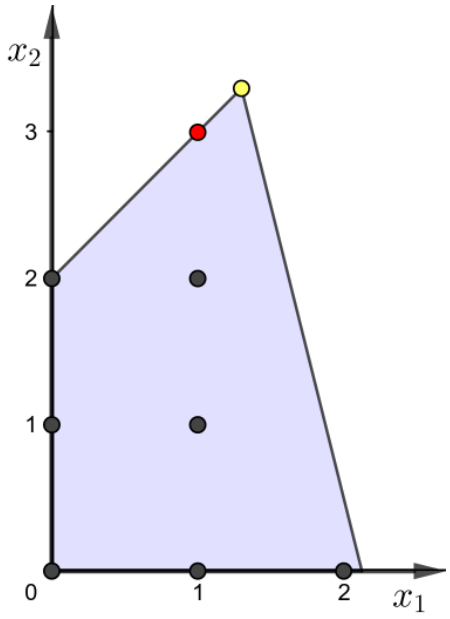

Figure 3.1: Geometrical representation of the ILP problem (3.3) and of its LPR [10].

In the previous example we were able to enumerate all the solutions of the ILP problem. This was possible because the problem was very simple (it is not at all practical, even for a computer, to follow this strategy for more complex ILP problems). We will see now two methods for solving ILP problems.

3.2 The branch-and-bound method

The branch-and-bound method is a general technique which allows to solve optimization problems by subdividing them into smaller sub-problems and utilizing a bounding function to discard a number of sub-problems that, certainly, will not contain the optimal solution. In particular, for our purposes, we will focus on the branch-and-bound method applied to ILP problems. In this context, to solve an ILP problem, the branch-and-bound method is used together with a method (usually the simplex algorithm) which permits to solve LP problems.

An example will help us to clarify how the branch-and-bound method works (we will use the same simple example presented in [10]). Let us consider again the LP problem (3.3). As we have already seen, the optimal solution of its LPR is given by $x_1 = 1.3$ and $x_2 = 3.3$. This optimal solution provides 14.08 as the optimal value of the LPR. Consequently, 14.08 is an upper bound on the optimal solution of the ILP problem. Branching on variable x_1, we obtain two ILP problems. The LPR of the one with the additional constraint $x_1 \leq 1$ has optimal solution $x_1 = 1$, $x_2 = 3$ with optimal value 11.8. Since this optimal solution is an integer solution we prune by integrality this sub-problem. Therefore, 11.8 is a lower bound on the optimal value of (3.3). On the other hand, the LPR of the sub-problem with the additional constraint $x_1 \geq 2$ has optimal solution $x_1 = 2$, $x_2 = 0.5$ with optimal value 12.05. Let us define $z := 5.5x_1 + 2.1x_2$ for the sake of simplicity. All the above steps are represented in the *enumeration tree* shown in Figure 3.2.

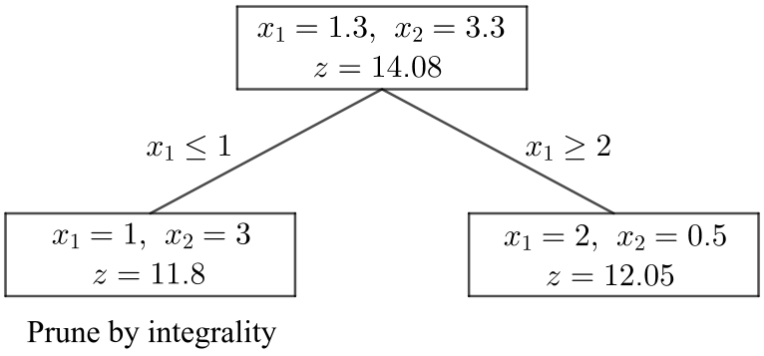

Figure 3.2: Branching on variable x_1 [10].

Note that, in the optimal solution of the sub-problem with the additional constraint $x_1 \geq 2$, x_2 is fractional, so this solution is not feasible to the integer program. Since, in this case, $z =$

12.05 is greater than 11.8 (the value of the best integer solution found so far), we need to investigate further the right branch of the enumeration tree represented in Figure 3.2. Thus we now branch on variable x_2. In this way we obtain two ILP problems, one with the additional constraint $x_2 \leq 0$ the other with the additional constraint $x_2 \geq 1$. The LPR of the second of these ILP problems is infeasible, so we prune this problem by infeasibility. The optimal solution of the LPR of the first of these ILP problems is given by $x_1 = 2.125$, $x_2 = 0$ and the optimal value is 11.6875. Since this value is smaller than the best lower bound 11.8, we prune by bound the corresponding node of the enumeration tree. The enumeration is now complete. Therefore, the optimal solution of (3.3) is given by $x_1 = 1$, $x_2 = 3$ and the optimal value is 11.8. The complete enumeration tree is represented in Figure 3.3.

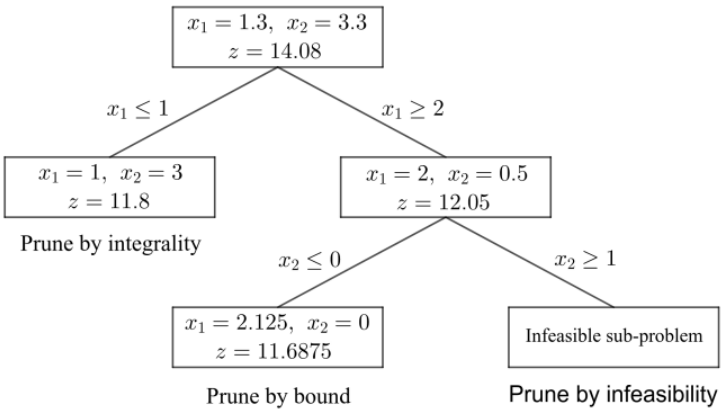

Figure 3.3: Example of a complete branch-and-bound tree [10].

To summarize what we have seen, the branch-and-bound method alternates two different phases: in the *branching* phase a feasible region is split into two parts; in the *bounding* phase a LP problem is solved (typically using the simplex algorithm) in order to find an upper bound for the considered region.

3.3 The cutting plane method

Suppose we want to solve an ILP problem that admits an optimal solution. Let us denote with F the feasible region of the ILP problem and with R_0 the feasible region of its LPR. Obviously $F \subseteq R_0$. Using, for example, the simplex method we can can find an optimal solution x_0 in R_0 (for the LPR of our ILP problem). If $x_0 \in F$, then we have found an optimal

solution (x_0) of the ILP problem, so we are done (without even applying the cutting plane method). Otherwise we have that $x_0 \notin F$ and we can use the cutting plane method. We have to find a *valid inequality*. This means that we have to find an inequality (namely a new constraint of the form $\alpha_1^T x \leq \beta_1$, where $\alpha_1 \in \mathbb{R}^n$ and $\beta_1 \in \mathbb{R}$) which, added to the original formulation of our ILP problem, does not modify F. This inequality will describe, obviously, an half-space of $\mathbb{R}^n$ which will be denoted by P_1 and (since the new constraint must be a valid inequality) will satisfy $F \cap P_1 = F$. However, to be a *cutting plane*, the inequality we are looking for must also (besides being a valid inequality) be violated by x_0 (meaning that $x_0 \notin P_1$).

Now suppose we have found such an inequality. Let $R_1 := P_1 \cap R_0$. We can solve the LP problem with feasible region R_1 with the simplex algorithm in order to find an optimal solution x_1. If $x_1 \in F$, then we have found an optimal solution (x_1) of our ILP problem, so we are done. Otherwise we find another cutting plane $\alpha_2^T x \leq \beta_2$ (where $\alpha_2 \in \mathbb{R}^n$ and $\beta_2 \in \mathbb{R}$), describing the half-space P_2. Now, being $\alpha_2^T x \leq \beta_2$ a cutting plane, $F \cap P_2 = F$ and $x_1 \notin P_2$. So we can define $R_2 := P_2 \cap R_1$ and repeat the procedure described above. Our hope is that, at a certain point, we will find $k \in \mathbb{N}$ such that $x_k \in F$. If this will happen (which is not, in general, guaranteed), then x_k will be an optimal solution of our ILP problem.

We have just described, from a general viewpoint, the way in which the cutting plane method operates. Remember that, in the above notation, $F \subseteq R_k \subseteq R_{k-1} \subseteq \cdots \subseteq R_1 \subseteq R_0$. Note that, at every iteration, there are infinitely many possible cutting planes to add. It is worth noting that there are cutting planes that always work, such as, for example, the so-called Gomory fractional cuts. In fact, using, at every iteration, a Gomory fracional cut it is possible to ensure that the cutting plane method will always terminate (we will always find, in the above notation, $k \in \mathbb{N}$ such that $x_k \in F$) [10]. Another possibility, given a specific ILP problem, is to search for ad hoc cutting planes. The latter, however, do not always guarantee that we will be able to find $k \in \mathbb{N}$ such that $x_k \in F$.

It is also possible to combine the cutting plane method with the branch-and-bound algorithm. This is done in the branch-and-cut method, "which is currently the most successful method for solving integer programs" [10]. Always citing [10], the branch-and-cut method "is obtained by adding a cutting-plane step before the branching step in the branch-and-bound algorithm".

3.4 Cover inequalities

Before introducing cover inequalities, let us recall the notion of convex hull. Given $S \subseteq \mathbb{R}^n$, the *convex hull* of S is the intersection of all the convex sets containing S. The convex hull is the unique smallest convex set containing S and is also the set of all convex combinations of points in S [22]. Assume we want to write a formulation for an ILP problem. There are infinitely many possible formulations (for our ILP problem) which result in the same feasible region. Let us consider two possible formulations of this type, let us say F_1 and F_2. By construction F_1 and F_2 have the same feasible region F. Let R_1 be the feasible region of the LPR of F_1. Let R_2 be the feasible region of the LPR of F_2. If $R_1 = R_2$, then F_1 and F_2 are substantially the *same* formulation. Otherwise, if $R_1 \neq R_2$, we say that F_1 and F_2 are *different* formulations. If $R_1 \subsetneq R_2$, then formulation F_1 is *tighter* than formulation F_2. The tightest possible formulation (the so-called *ideal formulation*) is the one whose LPR's feasible region coincides with the convex hull of F. Let us call H this formulation and R its feasible region. Formulation H has an extremely important property. If our ILP problem has an optimal solution, then solving the LPR of H (with, for example, the simplex algorithm) we find a certain point $\overline{x} \in R$ which always satisfy $\overline{x} \in F$. This means that, using formulation H, to solve our ILP problem it is sufficient to solve its LPR (and, as we have already seen, this is a huge advantage). However, in general, it is extremely difficult to find the formulation H. To be as clear as possible, in Figure 3.4 is shown the convex hull (the pink polygon) of the feasible region of problem (3.3).

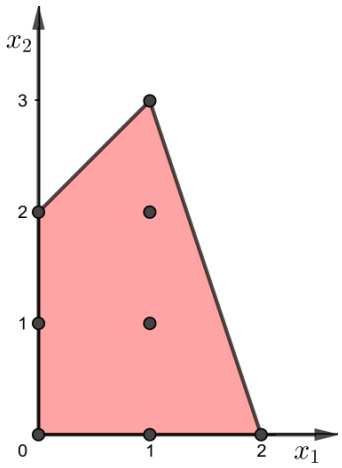

Figure 3.4: Convex hull of the feasible region of problem (3.3).

To tighten a given formulation of an ILP problem, we can add some valid inequalities, as defined in Section 3.3. An important class of valid inequalities, which will play a crucial role in this thesis, is given by the so-called *cover inequalities*. The remaining part of this section is based on [11].

3.4.1 Cover inequalities for the knapsack problem

Let us consider the so-called *knapsack problem*. In this problem, we suppose to have n items of weight $a_1, \dots, a_n \geq 0$ and profit $p_1, \dots, p_n \in \mathbb{R}$ respectively, and a bag (the knapsack) of maximum capacity $\beta \geq 0$. We want to find a subset of these n items that maximize the total profit without exceeding β. We also assume that $a_1, \dots, a_n, \beta \in \mathbb{Z}$. Defining binary variables $x_1, \dots, x_n$, where $x_i = 1$ if and only if item i is selected, we can give the following ILP formulation of the knapsack problem:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} p_i x_i \\ \text{subject to} \quad & \sum_{i=1}^{n} a_i x_i \leq \beta \\ & 0 \leq x_i \leq 1 \quad \forall i \in \{1, \dots, n\} \\ & x_i \in \mathbb{Z} \quad \forall i \in \{1, \dots, n\}. \end{aligned} \tag{3.4}$$

The feasible region of the LPR of problem (3.4) is usually much larger than the convex hull of the solutions of problem (3.4). For this reason we would like to find valid inequalities to strengthen the formulation. We call *cover* any subset of the n items that exceeds the knapsack capacity. This means that a cover is a subset $C \subseteq \{1, \dots, n\}$ satisfying

$$\sum_{i \in C} a_i > \beta.$$

Remembering that $a_1, \dots, a_n, \beta \in \mathbb{Z}$ (by assumption), we can rewrite the previous condition as

$$\sum_{i \in C} a_i \geq \beta + 1.$$

By definition, it is impossible to put all the elements of a cover C in the knapsack. Therefore we have the following valid inequality:

$$\sum_{i \in C} x_i \leq |C| - 1.$$

This inequality is called *cover inequality*.

If we add all possible cover inequalities to the original formulation of the knapsack problem, we obtain a better formulation. It can be shown, however, that this is not the ideal formulation yet. Since in the worst case there is one cover inequality for every non-empty subset of $\{1, \ldots, n\}$, we can have up to $2^n - 1$ cover inequalities. Even though in most practical cases only some of these subsets are actually covers, the number of covers is still too large. Consequently, adding all cover inequality is not practical at all (we cannot simply enumerate them and search for a violated one). Therefore, the idea is to add dynamically the cover inequalities only when they are really needed [11]. We will now see how to put in practice this idea.

Suppose that we have solved the LPR of problem (3.4) and we have found an optimal solution $\overline{x}$ that is not an integer vector. We want to check if there exists a cover inequality violated by $\overline{x}$ in order to (if it exists) add it to the formulation and solve the new LP problem. The problem of finding a cover inequality violated by $\overline{x}$, or deciding that none exists, is known as the *separation problem* for the cover inequalities [11]. Understanding if there is a cover inequality violated by $\overline{x}$ means understanding if we can find a subset $C \subseteq \{1, \ldots, n\}$ such that:

(i) $\sum_{i \in C} a_i \geq \beta + 1$;

(ii) $\sum_{i \in C} \overline{x}_i > |C| - 1$.

Condition (i) guarantees that the set C is a cover, while condition (ii) says that $\overline{x}$ does not satisfy the corresponding cover inequality. In order to understand if there is a cover inequality violated by $\overline{x}$ or not, let us introduce the binary variables $z_1, \ldots, z_n$, where $z_i = 1$ if and only if item i belongs to the cover C. We can rewrite condition (i) as follows:

$$\sum_{i=1}^{n} a_i z_i \geq \beta + 1.$$

Since, by definition, $\sum_{i=1}^{n} z_i = |C|$, we can rewrite condition (ii) as follows:

$$\sum_{i=1}^{n} \overline{x}_i z_i > -1 + \sum_{i=1}^{n} z_i \quad \text{or, equivalenly,} \quad \sum_{i=1}^{n} (1 - \overline{x}_i) z_i < 1.$$

Let us now consider the following ILP problem (where the $\overline{x}_i$s are known values and not variables):

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} (1 - \overline{x}_i) z_i \\ \text{subject to} \quad & \sum_{i=1}^{n} a_i z_i \geq \beta + 1 \\ & 0 \leq z_i \leq 1 \quad \forall i \in \{1, \ldots, n\} \\ & z_i \in \mathbb{Z} \quad \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{3.5}$$

The constraints guarantee that the variables z_i define a cover. If we solve now problem (3.5), assuming that it is feasible (otherwise there are no covers and, therefore, there are no cover inequalities), we find an optimal (integer) solution $\overline{z}$ with optimal value $\overline{v}$. There are now two possible different scenarios. If $\overline{v} < 1$, meaning that $\sum_{i=1}^{n} (1 - \overline{x}_i)\overline{z}_i < 1$, then (as we have seen above) the cover inequality defined by $\overline{z}$ is violated by $\overline{x}$. Otherwise, if $\overline{v} \geq 1$, there is not any cover inequality violated by $\overline{x}$ (since, in this case, $\sum_{i=1}^{n} (1 - \overline{x}_i) z_i \geq 1$ for any vector z defining a cover). So, to establish if there is a cover inequality violated by $\overline{x}$ (and find it), it is sufficient to solve the ILP problem (3.5). Let us observe that this problem is very similar to the ILP problem (3.4). One might therefore expect it would be better to solve directly the original problem rather than solving a similar problem just to find a new inequality to include in the formulation. However, this approach is much more promising in more general situations [11]. This means that it is possible to apply, with more efficiency from a computational viewpoint, this method to ILP problems different from the knapsack problem.

Before seeing how we can proceed to find cover inequalities for more general ILP problems, let us put in practice the above procedure on a trivial example. Let us consider the following

knapsack problem:

$$\begin{aligned} \max \quad & x_1 + 3x_2 \\ \text{subject to} \quad & x_1 + 2x_2 \leq 1 \\ & 0 \leq x_1, x_2 \leq 1 \\ & x_1, x_2 \in \mathbb{Z}. \end{aligned} \tag{3.6}$$

It is trivial to check that the feasible region is the subset of $\mathbb{R}^2$ containing only the points $(0, 0)$ and $(1, 0)$. The last one is the optimal solution (and the optimal value is 1). However, solving the LPR of problem (3.6), we obtain the optimal solution $x_1 = 0$, $x_2 = 0.5$, with optimal value 1.5. Since this solution is not integer, we look for a cover inequality. Following the above procedure (see (3.5)), we have to solve this ILP problem:

$$\begin{aligned} \min \quad & z_1 + 0.5z_2 \\ \text{subject to} \quad & z_1 + 2z_2 \geq 2 \\ & 0 \leq z_1, z_2 \leq 1 \\ & z_1, z_2 \in \mathbb{Z}. \end{aligned} \tag{3.7}$$

As one can trivially check, problem (3.7) admits only two solutions. Its feasible region is the subset of $\mathbb{R}^2$ containing only the points $(1, 1)$ and $(0, 1)$. The last one is the optimal solution (and the optimal value is 0.5). Therefore, we get the following cover inequality:

$$x_2 \leq 0.$$

Let us add it to the formulation in (3.6). The LPR of the new formulation is:

$$\begin{aligned} \max \quad & x_1 + 3x_2 \\ \text{subject to} \quad & x_1 + 2x_2 \leq 1 \\ & 0 \leq x_1 \leq 1 \\ & x_2 = 0. \end{aligned} \tag{3.8}$$

As one can trivially check, the optimal solution of problem (3.8) is $x_1 = 1$, $x_2 = 0$. Since it is an integer solution, it is also (as we already have seen) the optimal solution of problem (3.6). Figure 3.5 represents the feasible region of problem (3.6) (the two black points) and of

its LPR (the green triangle). Moreover, the feasible region of problem (3.8) is represented by the red segment (which includes the two black points).

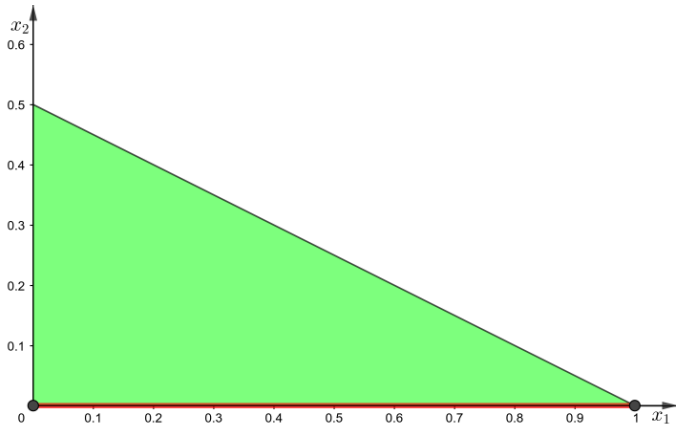

Figure 3.5: Feasible regions of problem (3.6), problem (3.8) and problem (3.6)'s LPR.

3.4.2 Cover inequalities for more general integer linear programming problems

Let us now see how it is possible to find cover inequalities also for more general ILP problems (not only knapsack problems). Let us consider an ILP problem that can be written in the form

$$\begin{aligned} \max \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & Bx \leq d \\ & 0 \leq x_i \leq 1 \quad \forall i \in \{1, \dots, n\} \\ & x_i \in \mathbb{Z} \quad \forall i \in \{1, \dots, n\}, \end{aligned} \tag{3.9}$$

where $c, x \in \mathbb{R}^n$, $A \in \mathbb{N}^{m \times n}$, $b \in \mathbb{N}^m$, $B \in \mathbb{R}^{k \times n}$ and $d \in \mathbb{R}^k$ (A, B, b, c, d are given parameters, whereas x is a vector of variables). Since $A \in \mathbb{N}^{m \times n}$ and $b \in \mathbb{N}^m$, every single constraint of the system $Ax \leq b$ can be seen as knapsack-type constraint. This means that, if we replace the system given by constraints $Ax \leq b$ and $Bx \leq d$ with any constraint among those present in $Ax \leq b$ (forgetting all the others), then we obtain an ILP problem that looks exactly like a knapsack problem. It is therefore possible to add to the formulation the cover

inequalities associated with each of the knapsack problems obtained this way (i.e., removing all constraints $Ax \leq b$ and $Bx \leq d$ except one constraint belonging to $Ax \leq b$) [11]. We can use the method described above in order to add the cover inequalities only when they are really needed, thus trying to contain the number of inequalities to be added.

Suppose that we have solved the LPR of problem (3.9), finding an optimal non-integer solution $\overline{x}$. At this point, for every knapsack problem obtained by removing all the constraints of the system (given by constraints $Ax \leq b$ and $Bx \leq d$) except one constraint among those present in $Ax \leq b$, we want to check if there is a cover inequality violated by $\overline{x}$. To do this, it suffices to solve a problem of the form (3.5). If this problem is feasible and the optimal value is smaller than 1, then we have just found a cover inequality violated by $\overline{x}$ that we can add to the formulation; otherwise there is no cover inequality violated by $\overline{x}$. We can then solve the new LPR of (3.9) (which includes the cover inequalities that have been added) and repeat this procedure until the current solution satisfies all cover inequalities. The entire process is well explained in Algorithm 3.1.

With Algorithm 3.1 we have to solve many problems of the form (3.5), which are ILP problems (thus hard to solve generally speaking). However, problem (3.5) is (quoting [11]) "one of the simplest integer linear programming problems and therefore, although in principle an exponential time might be needed to solve it, in practice an optimal solution can be found in a reasonable amount of time". As we have seen above, Algorithm 3.1 terminates when the current solution, assuming it exists, does not violate any cover inequality (pay attention that when this happens, $\overline{x}$ is not necessarily integer, since cover inequalities are not usually sufficient to describe the ideal formulation of the problem). It is, of course, possible to stop the algorithm before its natural termination if we think that the cover inequalities that we have already found are sufficient to provide a good formulation of the problem. However, if $\overline{x}$ is not integer, we can apply the branch-and-bound method. Adding some cover inequalities allows us to start from a better formulation and usually makes the branch-and-bound method terminate in a shorter time. This kind of approach, in which the formulation is strengthened with valid inequalities and then branch-and-bound is applied, is known as *cut-and-branch* [11]. In this thesis, the cut-and-branch technique will be used in order to get some computational results.

Algorithm 3.1: Generation of cover inequalities

Step 1: Solve the LPR of problem (3.9). If there is no solution, then **STOP**: problem (3.9) is infeasible. Otherwise let $\overline{x}$ be the optimal solution obtained.

Step 2: Check if $\overline{x}$ is integer. If this is true, then **STOP**: an optimal solution of problem (3.9) has been found. Otherwise proceed to the next step.

Step 3: For every constraint of the system $Ax \leq b$, solve the corresponding ILP problem (3.5) (where $a_1, \ldots, a_n$ and β are the coefficients and the right-hand side of the constraint). If for every constraint of the system $Ax \leq b$ the corresponding ILP problem (3.5) is infeasible, then **STOP**: there are no covers and, therefore, there are no cover inequalities. Otherwise, for every constraint of the system $Ax \leq b$ such that the corresponding ILP problem (3.5) is feasible, let $\overline{z}$ be the optimal integer solution (obtained by solving the corresponding ILP problem (3.5)) and $\overline{v}$ the corresponding optimal value.

Step 4: For every optimal value $\overline{v}$ found at the previous step, check whether $\overline{v} < 1$ holds. For each $\overline{v}$ such that $\overline{v} < 1$ holds, the corresponding $\overline{z}$ gives a cover inequality violated by $\overline{x}$ and this cover inequality is the one associated with the cover given by the set $C = \{i \in \{1, \ldots, n\} : \overline{z}_i = 1\}$.

Step 5: If for all the problems solved at **Step 3** we have that $\overline{v} \geq 1$, then there are no cover inequalities violated by $\overline{x}$. In this case **STOP**. Otherwise proceed to the next step.

Step 6: Add to formulation (3.9) all the cover inequalities found so far (which, to be clear, will become part of the new system $Bx \leq d$) and go back to **Step 1**.

3.5 Computational tools

In this section, we briefly illustrate the computational tools used in the development of this thesis. To run our computer simulations (consisting in solving many ILP problems and LP problems) we have used the optimization software IBM ILOG CPLEX (version 11.1.1 and version 12.8.0.0) through its AMPL (version 20080701) interface. The various instances used in our computational experiments have been generated using the software MATLAB. Some of these instances have been created by us (using version R2022b), whereas others are taken from [26] (and have been generated by version R2020b).

3.5.1 IBM ILOG CPLEX

IBM ILOG CPLEX, often informally referred to simply as CPLEX, is a solver for mathematical optimization. The CPLEX Optimizer was named after the simplex algorithm as implemented in the C programming language, although nowadays it also makes available other algorithms of mathematical programming and it provides interfaces to programming languages other than C (such as, for example, C#, C++, Java and Python). The CPLEX Optimizer was originally developed by Robert E. Bixby. It has been sold commercially since 1988 by CPLEX Optimization Inc., which was acquired by ILOG in 1997. ILOG was acquired by IBM afterwards (in January 2009). CPLEX is currently maintained and developed by IBM. The IBM ILOG CPLEX Optimizer is able to solve: ILP problems; very large LP problems (using either primal or dual variants of the simplex algorithm or the barrier interior point method); convex and non-convex quadratic programming problems; convex quadratically constrained problems. For more information on the IBM ILOG CPLEX Optimizer, visit [17].

3.5.2 AMPL

AMPL (the name is an acronym for "A Mathematical Programming Language") is a high-level algebraic modelling language for mathematical programming. It was designed and implemented by Robert Fourer, David Gay and Brian Kernighan at Bell Labotatories in 1985. AMPL has been developed in order to describe and solve high-complexity problems for large-scale mathematical computing (it permits to deal with, for example, large-scale optimization and scheduling-type problems). AMPL is not a solver, so it does not solve problems by itself. However, it writes files with complete details of the problem instances to be solved and invokes separate solvers (both open source and commercial software). Some of these solvers are

CBC, CPLEX, Gurobi and MINOS. A big advantage of AMPL is the closeness of its syntax to the mathematical notation of optimization problems (this permits to have a very succinct and readable definition of optimization problems). AMPL is available for many operating systems (such as Linux, macOS, Solaris, AIX and Windows). For more information on AMPL, visit [3].

3.5.3 MATLAB

MATLAB (an abbreviation of "MATrix LABoratory") is an environment, developed by MathWorks (and written in the programming languages C, C++ and MATLAB), for numerical computing and statistical analysis, which also includes the homonymous programming language (the latter, also developed by MathWorks, was originally designed by mathematician and computer programmer Cleve Moler in the late 1970s). The software MATLAB was first released as a commercial product in 1984 by MathWorks, Inc. (which was founded in order to develop the software). MATLAB permits to manipulate matrices, plot data and functions, implement algorithms and create user interfaces. It also allows interfacing with programs written in other languages. Nowadays, MATLAB has millions of users worldwide (it is widely used in both universities and industry). They come from various backgrounds (such as engineering, science and economics). The success of MATLAB is due to its many tools which can be used in the most varied applied fields of study. Moreover, a big advantage of MATLAB, is the possibility to run it on various operating systems, including Linux, macOS, Unix and Windows. For more information on MATLAB, visit [20].

4

State of the art

The ATFM problem has been addressed and solved through different approaches in the course of time. These various approaches lead to models that can be deterministic or stochastic, static or dynamic, linear or nonlinear, more detailed or less detailed (depending, for example, on how many of the ATFM strategies described in Subsection 2.2.3 are taken into account) and so on. The model to use must be chosen depending on which features and characteristics we want to highlight. One of these possible approaches is based on integer linear programming. Compared to other types of models, ILP models have some very useful features. In fact, they are: discrete; adequate to represent many real-world problems; solvable with standard (and relatively efficient) methods; usually easily readable; often easily generalizable. The model considered in this thesis is an ILP model.

In the first section of this fourth chapter, we will briefly recall some of the most relevant ILP models for ATFM present in literature. In the second section, we will instead describe in detail the model presented in [26], which will be the basis of the work developed in this thesis.

4.1 Some relevant ILP models for ATFM

Let us briefly recall (in chronological order) some of the most relevant ILP models for ATFM present in literature.

Helme (1992) [16]: In the ILP model here proposed there are only two possible strategies to minimize delays: ground-holding and airborne-holding. This means that, for each flight, the only possible choices are: decide whether or not to postpone its departure and, if so, for how long; decide whether or not to postpone its landing and, if so, for how long. It is not possible instead to adjust its cruise speed (speed control is not an available option) or to change its route (re-routing is not allowed). In other words, for each flight, it is possible to deviate from the scheduled trajectory only from a temporal viewpoint (and without using speed control).

Andreatta, Odoni and Richetta (1993) [5]: The model presented in this paper allows to use only the ground-holding policy to face the ATFM problem. In addition to speed control and re-routing, also airborne-holding is not an available option (since it is more expensive than ground-holding).

Bertsimas and Stock Patterson (1998) [8]: This model is particularly interesting for our purposes, since the formulation presented in [26], hence ours, is based on it. As in [16] (and also in [5]), for each flight, it is possible to deviate from the scheduled trajectory only from a temporal viewpoint (re-routing is not allowed, although the paper presents two possible approaches to add also re-routing policy to the model). However, besides ground-holding and airborne-holding policies, this model provides the possibility to adjust the cruise speed of every flight (speed control is an available option). A fundamental contribution present in this article is given by the introduction of a new type of binary variables. Fixed a flight f, a sector j (or an airport k) and a time t we have that $w^j_{f,t} = 1$ (or $w^k_{f,t} = 1$) if and only if f arrives at j (or f takes off from k, or f lands at k) *by* time t (and not, as it had always been done before, *at* time t). These new variables have some great advantages compared to those used until then. They permit to write down some constraints in a more clear (and elegant) way. Furthermore, the formulation is pretty tight (some constraints are facet defining for the convex hull of solutions): solving the LPR of the formulation in a number of instances the authors have found, almost always, an integral solution. There are three different types of connectivity described by the model's constraints: connectivity between sectors; connectivity between airports; connectivity in time. This model takes also into account continued flights (which are consecutive flights operated by a same aircraft).

Bertsimas and Stock Patterson (2000) [9]: In this model also re-routing (in addition to ground-holding, airborne-holding and speed control) is an available option to minimize the to-

tal cost of delays. Using this model to solve an instance of the ATFM problem requires a long computation time. This model is not suitable for being applied to real-world ATFM problems (the instances are too big).

Bertsimas, Lulli and Odoni (2011) [7]: This model is basically an improvement of [9], since it allows to solve bigger instances ("of size comparable to that of the entire U.S. air traffic management" [7]). Similarly to [8], this formulation is pretty tight. Ground-holding, airborne-holding, speed control and re-routing are all available strategies in this model. Note that re-rounting is possible thanks to sets of local conditions that enable to represent re-routing options compactly by only introducing some new constraints. Furthermore, the objective function presented in this paper favors a fair distribution of delays, trying to avoid excessive accumulation of delays on a restricted number of flights.

Agustín, Alonso-Ayuso, Escudero and Pizarro (2012) [1, 2]: This article is divided into two parts. In the first one the authors present an ILP (or, to be more precise, a MILP) model which, if necessary, allows for flight cancellation and re-routing. A wide range of possible objective functions is provided. Quoting [1]: "The global objective function to be optimized can include different terms, depending on the goal of the decision maker". For this reason, the article lists a number of possible terms to be combined together (with appropriate weights) in order to create the desired objective function to be minimized. Among them there are (besides a number of terms related to various delays), for example, the total cancelation cost (which is related to canceled flights) and the cost of using alternative flights routes (different from the scheduled ones). Also this formulation is pretty tight and permits to treat large-scale instances. In the second part of the article, taking a certain degree of uncertainty into account, a stochastic variation of the previous model is presented.

Fomeni, Lulli and Zografos (2017) [15]: This paper provides a binary ILP model that contributes to the optimization and optimum configuration of the TBO (an acronym for "Trajectory Based Operations") concept. TBO is "the concept of improving throughput, flight efficiency, flight times, and schedule predictability through better prediction and coordination of aircraft trajectories" [23]. The model considers the favourite 4D-trajectory of all the flights in the pre-tactical planning phase and provides an optimal pre-departure 4D-trajectory for each flight to be shared or negotiated with other stakeholders and subsequently handled throughout the flight. These trajectories are obtained by minimizing the deviation (relatively to time delay,

lateral and vertical deviation) from the original favourite trajectories. The novelty of this model is that, besides considering the complete space-time 4D-trajectory for each flight, it takes into account the priorities and the preferences of the ATM stakeholders [15].

4.2 An ILP model for the ATFM problem with dynamic airspace configuration

Let us now introduce the model presented in [26] in detail. This section is based on the fifth chapter of [26] and is very important for the continuation this thesis, since the aim of our work is to find a tighter formulation for the same specific ATFM problem faced in [26] and already recalled in Section 2.3.

4.2.1 Notation

Let us now recall the data and the notation present in the formulation proposed in [26] (they will also be used in this thesis). For the sake of simplicity (it will be useful later on), we will write $\widetilde{C}_h(t)$ instead of $C_h(t)$ (the latter notation is the one used in [26]). We will also write l_f^j instead of l_{fj}.

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

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

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

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

$\mathcal{M} = \{1, \dots, M\}$ is a list of configurations (recall Subsection 2.3.1).

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

$\mathcal{P}_m = \{h \in \mathcal{H} : \text{their union is the configuration } m \in \mathcal{M}\}$ is the set of collapsed sectors belonging to the configuration $m \in \mathcal{M}$.

$\mathcal{B}_h = \{j \in \mathcal{J} : \text{their union is the collapsed sector } h \in \mathcal{H}\}$ is the set of elementary sectors which form the collapsed sector $h \in \mathcal{H}$.

$N_f =$ number of elements belonging to the sequence that describes the spatial trajectory of flight f (remember what we have seen in Subsection 2.3.3). Note that, according to our assumptions, $N_f \geq 4$.

$$P(f,i) = \begin{cases} \text{the departure airport} & \text{if } i = 1, \\ \text{the } (i-1)\text{-th elementary sector in flight } f\text{'s path} & \text{if } 1 < i < N_f, \\ \text{the destination airport} & \text{if } i = N_f. \end{cases}$$

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

$D_k(t) =$ 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 in arrival allowed for flight f.

$c_f^g =$ cost of holding flight f (through ground-holding policy) on the ground for one unit of time.

$c_f^a =$ cost of holding flight f (through airborne-holding and/or speed control policies) in the air for one unit of time.

$l_f^k = 0$ is the scheduled number of time units that flight f must spend in its departure airport k. This quantity is equal to zero since, once it has taken off, flight f is immediately inside the first elementary sector of its path (remember that in this model, as we have already seen in Subsection 2.3.1, each airport is entirely contained within an elementary sector).

$l_f^j =$ scheduled number of time units that flight f must spend in the elementary sector j. We assume $l_f^j \geq 1$.

$\underline{T}_f^k =$ first feasible time period for flight f to take off from (or land at) airport k.

$\overline{T}_f^k = \underline{T}_f^k + \Delta_f =$ last feasible time period for flight f to take off from (or land at) airport k.

$T_f^k = \{\underline{T}_f^k, \ldots, \overline{T}_f^k\}$ is the set of feasible time periods for flight f to take off from (or land at) airport k. The minimum of the set T_f^k is $\underline{T}_f^k$, whereas the maximum is $\overline{T}_f^k$.

$\underline{T}_f^j =$ first feasible time period for flight f to arrive at elementary sector j.

$\overline{T}_f^j = \underline{T}_f^j + \Delta_f =$ last feasible time period for flight f to arrive at elementary sector j.

$T_f^j = \{\underline{T}_f^j, \ldots, \overline{T}_f^j\}$ is the set of feasible time periods for flight f to arrive at elementary sector j. The minimum of the set T_f^j is $\underline{T}_f^j$, whereas the maximum is $\overline{T}_f^j$.

$\tau =$ minimum number of consecutive time periods in which the (temporarily) chosen airspace configuration must remain active before it can be changed ($\tau \in \mathbb{N} \setminus \{0\}$).

$\widetilde{C}_h(t) = \left(\sum_{j \in \mathcal{B}_h} \left(\sum_{\substack{f \in \mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j', \\ 1<i<N_f}} 1 \right) \right) - S_h(t) =$ constant used to make capacity constraints redundant when related to h, t and a non-active airspace configuration. A constant $\widetilde{C}_h(t)$ is defined for each collapsed sector h and every time period t.

4.2.2 Decision variables

Let us now recall the two different types of decision variables present in the formulation proposed in [26] .

The first type is the same used in [8]. For the sake of clarity, we distinguish two different cases, depending on whether the variables in question give indications relating to an airport or an elementary sector crossed during the flight. In the first case, for every $f \in \mathcal{F}$, $k \in \mathcal{K} \cap P_f$ and $t \in T_f^k$, we define

$$w_{f,t}^k = \begin{cases} 1 & \text{if flight } f \text{ takes off from (or lands at) airport } k \text{ by time } t, \\ 0 & \text{otherwise.} \end{cases}$$

In the second case, for every $f \in \mathcal{F},\ j \in \mathcal{J} \cap P_f$ and $t \in T_f^j$, we define

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

The second type of decision variables is defined as follows. For every $m \in \mathcal{M}$ and $t \in \mathcal{T}$ we define

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

The variables $y_{m,t}$ allow to easily capture the fact that one only one configuration must be active in a certain period of time. Furthermore, as we will see, they permit (together with τ) to impose constraints related to the stability of an airspace configuration over time.

In the next subsection, we will write down explicitly the model's ILP formulation provided by [26]. We will later describe in detail this formulation. For the moment, we limit ourselves to observe that, from a formal viewpoint, in some constraints (as well as in the objective function) of the formulation undefined variables appear (see Subsection 4.2.3). To be more precise, some of these variables are of the type $w_{f,t}^j$, where $f \in \mathcal{F}$, $j \in P_f$ (note that, in this case, j can also be an airport) and $t \in \mathcal{T},\ t < \underline{T}_f^j$. Besides this type of variables, undefined variables appear in constraints (4.9) when $t = 1$. All these undefined variables appear only because, for the sake of clarity, the formulation presented is not as formal as possible (otherwise there would not be any issue). To fix this "problem" it is sufficient to set to zero all these "undefined variables" (which simply, in a more formal formulation, would not appear).

4.2.3 The model's formulation

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

subject to

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

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

$$\sum_{j\in\mathcal{B}_h}\left(\sum_{\substack{f\in\mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j',\\ 1<i<N_f,\, t\in T_f^j}} (w_{f,t}^j - w_{f,t}^{j'})\right) \le \tag{4.3}$$

$$\le S_h(t) + \widetilde{C}_h(t)(1-y_{m,t}) \qquad \forall\, m\in\mathcal{M},\, h\in\mathcal{P}_m,\, t\in\mathcal{T}$$

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

$$w_{f,t}^j - w_{f,t-1}^j \ge 0 \qquad \forall\, f\in\mathcal{F},\, j\in P_f,\, t\in T_f^j \tag{4.5}$$

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

$$w_{f,\overline{T}_f^j}^j = 1 \qquad \forall\, f\in\mathcal{F},\, j\in P_f \tag{4.7}$$

$$\sum_{m\in\mathcal{M}} y_{m,t} = 1 \qquad \forall\, t\in\mathcal{T} \tag{4.8}$$

$$y_{m,t} - y_{m,t-1} \le y_{m,u} \quad \forall\, m\in\mathcal{M},\, t\in\mathcal{T},\, u\in\{t+1,\dots,\min\{t+\tau-1,T\}\} \tag{4.9}$$

$$y_{m,t}\in\{0,1\} \qquad \forall\, m\in\mathcal{M},\, t\in\mathcal{T}. \tag{4.10}$$

4.2.4 The objective function

The objective function of the previous formulation is the same used in [8]. Recalling also what we seen in Subsection 2.2.1, the objective function presented in Subsection 4.2.3 is, at first sight, quite difficult to understand. The reason is that both [26] and [8] originally wanted to minimize the objective function

$$\sum_{f \in \mathcal{F}} \left[c_f^g g_f + c_f^a a_f \right],$$

where g_f is the total number of time units that flight f is held on the ground (through ground-holding policy) and a_f is the total number of time units that flight f is held in the air (through airborne-holding and/or speed control policies). Note that this objective function is a weighted sum of all flights' delays.

Let us show that this weighted sum is exactly the objective function present in Subsection 4.2.3. First of all, let us observe that g_f equals the difference between the actual departure time of flight f and its scheduled departure time. This means that

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

Similarly, we can observe that a_f can be expressed as the actual arrival time of flight f minus its scheduled arrival time minus the amount of time that flight f has been held on the ground. This means that

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

Therefore

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

Finally, rearranging terms on the right-hand side, we obtain

$$\sum_{f\in\mathcal{F}}\left[c_f^g g_f + c_f^a a_f\right] = \sum_{f\in\mathcal{F}}\Bigg[(c_f^g - c_f^a)\sum_{t\in T_f^k,\, k=P(f,1)} t(w_{f,t}^k - w_{f,t-1}^k) + c_f^a \sum_{t\in T_f^k,\, k=P(f,N_f)} t(w_{f,t}^k - w_{f,t-1}^k) + (c_f^a - c_f^g)d_f - c_f^a r_f\Bigg].$$

The last right-hand side is exactly the objective function present in Subsection 4.2.3.

4.2.5 The constraints

Let us now describe the formulation's constraints in detail. At the end of this subsection, we will also recall a class of valid inequalities proposed, but not used, in [26].

Capacity constraints

Constraints (4.1), (4.2) and (4.3) are the capacity constraints. In particular, for every $k \in \mathcal{K}$ and $t \in \mathcal{T}$, constraints (4.1) ensure that the number of flights which may take off from airport k at time t will not exceed $D_k(t)$ (the departure capacity of airport k at time t).

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

Similarly, for every $k \in \mathcal{K}$ and $t \in \mathcal{T}$, constraints (4.2) guarantee that the number of flights which may land at airport k at time t will not exceed $A_k(t)$ (the arrival capacity of airport k at time t).

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

While constraints (4.1) and (4.2) are also used in [8], constraints (4.3) are specific of the formulation presented in [26]. They guarantee that, at any time period t, for every collapsed sector h belonging to at least one configuration $m \in \mathcal{M}$, the total number of flights present in h at time t (represented by the left-hand side) will have an upper bound. Let us focus now our

attention on these upper bounds in order to fully understand constraints (4.3).

$$\sum_{j\in\mathcal{B}_h}\left(\sum_{\substack{f\in\mathcal{F}:\,P(f,i)=j,\,P(f,i+1)=j',\\ 1<i<N_f,\,t\in T_f^j}}(w_{f,t}^{j}-w_{f,t}^{j'})\right)\leq \tag{4.3}$$
$$\leq S_h(t)+\widetilde{C}_h(t)(1-y_{m,t})\qquad\forall\, m\in\mathcal{M},\ h\in\mathcal{P}_m,\ t\in\mathcal{T}$$

These various upper bounds are expressed by the right-hand sides. Let us now consider $\overline{m}\in\mathcal{M}$ and $\bar{t}\in\mathcal{T}$. If $y_{\overline{m},\bar{t}}=1$ (configuration $\overline{m}$ is active at time $\bar{t}$), then, for every $h\in\mathcal{P}_{\overline{m}}$, the right-hand side is simply given by $S_h(\bar{t})$. In other words, in this case we obtain the following constraints:

$$\sum_{j\in\mathcal{B}_h}\left(\sum_{\substack{f\in\mathcal{F}:\,P(f,i)=j,\,P(f,i+1)=j',\\ 1<i<N_f,\,\bar{t}\in T_f^j}}(w_{f,\bar{t}}^{j}-w_{f,\bar{t}}^{j'})\right)\leq S_h(\bar{t})\qquad\forall\, h\in\mathcal{P}_{\overline{m}}. \tag{4.11}$$

If, on the other hand, $y_{\overline{m},\bar{t}}=0$ (configuration $\overline{m}$ is not active at time $\bar{t}$), then we have

$$\sum_{j\in\mathcal{B}_h}\left(\sum_{\substack{f\in\mathcal{F}:\,P(f,i)=j,\,P(f,i+1)=j',\\ 1<i<N_f,\,\bar{t}\in T_f^j}}(w_{f,\bar{t}}^{j}-w_{f,\bar{t}}^{j'})\right)\leq \tag{4.12}$$
$$\leq\sum_{j\in\mathcal{B}_h}\left(\sum_{\substack{f\in\mathcal{F}:\,P(f,i)=j,\,P(f,i+1)=j',\\ 1<i<N_f}}1\right)\qquad\forall\, h\in\mathcal{P}_{\overline{m}}.$$

Note that constraints (4.12) are completely irrelevant (they are satisfied by every combination of binary variables). In fact, since we are using binary variables, each term of the form $w_{f,\bar{t}}^{j}-$

$w^{j'}_{f,\bar{t}}$ present in the left hand-side satisfies $w^{j}_{f,\bar{t}} - w^{j'}_{f,\bar{t}} \leq 1 - 0 = 1$. Therefore, we have that

$$\begin{aligned} \sum_{j \in \mathcal{B}_h} \left(\sum_{\substack{f \in \mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j', \\ 1<i<N_f,\, \bar{t} \in T^j_f}} (w^{j}_{f,\bar{t}} - w^{j'}_{f,\bar{t}}) \right) \leq \\ \leq \sum_{j \in \mathcal{B}_h} \left(\sum_{\substack{f \in \mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j', \\ 1<i<N_f,\, \bar{t} \in T^j_f}} 1 \right) \qquad \forall\, h \in \mathcal{P}_{\overline{m}}. \end{aligned} \tag{4.13}$$

Since, for every $h \in \mathcal{P}_{\overline{m}}$ and $j \in \mathcal{B}_h$, we (obviously) have

$$\begin{aligned} \{f \in \mathcal{F}:\ P(f,i) = j,\ P(f,i+1) = j',\ 1 < i < N_f,\ \bar{t} \in T^j_f\} \subseteq \\ \subseteq \{f \in \mathcal{F}:\ P(f,i) = j,\ P(f,i+1) = j',\ 1 < i < N_f\}, \end{aligned}$$

it follows that

$$\begin{aligned} \sum_{j \in \mathcal{B}_h} \left(\sum_{\substack{f \in \mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j', \\ 1<i<N_f,\, \bar{t} \in T^j_f}} 1 \right) \leq \\ \leq \sum_{j \in \mathcal{B}_h} \left(\sum_{\substack{f \in \mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j', \\ 1<i<N_f}} 1 \right) \qquad \forall\, h \in \mathcal{P}_{\overline{m}}. \end{aligned} \tag{4.14}$$

Combining (4.13) and (4.14) we understand that constraints (4.12) are completely irrelevant (they are satisfied by every combination of binary variables).

Connectivity constraints

There are two different groups of connectivity constraints (which are also used in [8]). The first of these two groups is given by constraints (4.4), which substantially express connectivity between elementary sectors. To be more precise, they guarantee that a flight f cannot enter the next elementary sector on its path (or land at its destination airport) until it has spent l^j_f time units (the minimum possible) traveling through elementary sector j, the current elementary sector in its path [26]. Note that, to be exact, here j can also be the departure airport of

flight f and in this case, remembering what we have seen in Subsection 4.2.1, $l_f^j = 0$.

$$w^{j'}_{f,t+l_f^j} - w^j_{f,t} \leq 0 \qquad \begin{cases} \forall\, f \in \mathcal{F},\ t \in T_f^j,\ j = P(f,i), \\ j' = P(f,i+1),\ 1 \leq i < N_f \end{cases} \tag{4.4}$$

The second group of connectivity constraints is given by constraints (4.5). They express connectivity in time:

$$w^j_{f,t} - w^j_{f,t-1} \geq 0 \qquad \forall\, f \in \mathcal{F},\ j \in P_f,\ t \in T_f^j. \tag{4.5}$$

In particular, constraints (4.5) stipulate that if flight f has arrived in elementary sector j (or has taken off from/landed at airport j if j is an airport) by time $t \in T_f^j$ (that is to say if $w^j_{f,t} = 1$), then $w^j_{f,t'} = 1$ for every $t' \in T_f^j$, $t' > t$.

Configuration constraints

Constraints (4.8) and (4.9) are the configuration constraints. They are specific of the formulation proposed in [26]. In particular, for every $t \in \mathcal{T}$, constraints (4.8) guarantee that exactly one configuration $m \in \mathcal{M}$ will be active at time t (remember that we are using binary variables).

$$\sum_{m \in \mathcal{M}} y_{m,t} = 1 \qquad \forall\, t \in \mathcal{T} \tag{4.8}$$

Regarding constraints (4.9), quoting [26], they ensure that "once a configuration is chosen, it must be maintained for at least τ periods of time before it can be changed".

$$y_{m,t} - y_{m,t-1} \leq y_{m,u} \quad \forall\, m \in \mathcal{M},\ t \in \mathcal{T},\ u \in \{t+1, \ldots, \min\{t+\tau-1, T\}\} \tag{4.9}$$

If, for instance, $T = 5$, $\mathcal{M} = \{1,2\}$ and $\tau = 2$, then the sequences of configurations satisfying constraints (4.9) are exactly the following: $(1,1,1,1,1)$; $(2,2,2,2,2)$; $(1,1,1,1,2)$; $(2,2,2,2,1)$; $(1,1,1,2,2)$; $(2,2,2,1,1)$; $(1,1,2,2,2)$; $(2,2,1,1,1)$; $(1,1,2,2,1)$; $(2,2,1,1,2)$. This last example should have clarified the role of constraints (4.9). Note also that, for the sake of clarity, the notation used in constraints (4.9) is not as formal as possible. In fact, in addition to what we have already seen in Subsection 4.2.2, when we write

$$\forall\, u \in \{t+1, \ldots, \min\{t+\tau-1, T\}\} \tag{4.15}$$

we actually mean

$$\forall\, u \in \{\widehat{u} \in \mathcal{T} :\ t+1 \leq \widehat{u} \leq \min\{t+\tau-1, T\}\}. \tag{4.16}$$

It is worth observing that if $t+1 > \min\{t+\tau-1, T\}$ (it happens when $t = T$ and/or $\tau = 1$), then the set appearing in (4.16) is the empty set. This means that if $\tau = 1$, then there are no constraints of the form (4.9) in the formulation. On the other hand, if $\tau > 1$, then constraints (4.9) appear in the formulation for every $t \in \mathcal{T}$, with $t \neq T$ (there are no constraints of the form (4.9) associated to $t = T$).

Other constraints

Constraints (4.7) stipulate that, for every $f \in \mathcal{F}$ and $j \in P_f$ (note that j can also be an airport), flight f must arrive at elementary sector j (or take off from/land at airport j if j is an airport) by time $\overline{T}^j_f$, which is the last feasible time period for flight f to arrive at elementary sector j (or to take off from/land at airport j).

$$w^j_{f,\overline{T}^j_f} = 1 \qquad \forall\, f \in \mathcal{F},\ j \in P_f \tag{4.7}$$

These constraints are necessary to avoid trivial solutions in which all the variables of the form $w^j_{f,t}$ (where j can also be an airport) are equal to zero. Constraints (4.7) do not appear explicitly in [26]. However, they have been used by Zanardelli in order to do his computational experiments (see, for instance, the "# basic constraints" on page 78 of [26]). Similarly, constraints (4.7) do not appear explicitly in [8] (here the various $w^j_{f,\overline{T}^j_f}$ are not seen as variables, but they are simply set to 1 as parameters before solving the problem). For the sake of clarity, we preferred to include directly constraints (4.7) in the formulation.

Finally, constraints (4.6) and (4.10) establish that all the variables involved in the formulation must be binary variables.

$$w^j_{f,t} \in \{0,1\} \qquad \forall\, f \in \mathcal{F},\ j \in P_f,\ t \in T^j_f \tag{4.6}$$

$$y_{m,t} \in \{0,1\} \qquad \forall\, m \in \mathcal{M},\ t \in \mathcal{T} \tag{4.10}$$

A class of valid inequalities

In addition to the constraints already described, in [26] a class of valid inequalities is also proposed in order to tighten the formulation. However, these inequalities are never actually used and, technically, they are not even proven to be valid inequalities. The inequalities in question are the following:

$$\sum_{\substack{j\in\mathcal{B}_h:\ P(f,i)=j,\\ 1<i<N_f}} (w^j_{f,t} - w^j_{f,t-1}) \leq 1 \qquad \forall\, m \in \mathcal{M},\ h \in \mathcal{P}_m,\ f \in \mathcal{F},\ t \in \mathcal{T}. \tag{4.17}$$

Inequalities (4.17) stipulate that, for every $m \in \mathcal{M}$, $h \in \mathcal{P}_m$, $f \in \mathcal{F}$ and $t \in \mathcal{T}$, flight f can enter at most one elementary sector $j \in \mathcal{B}_h$ at time t. As we will see in Section 5.1, constraints (4.17) are really valid inequalities, but they do not tighten the formulation at all.

4.2.6 Size of the formulation

To conclude this detailed description of the model presented in [26], we calculate the size of the formulation recalled in Subsection 4.2.3. To be more precise, we compute the exact number of variables involved in the formulation (providing also an upper bound) and the exact number of constraints involved in the formulation (providing also an upper bound). Similar calculations, in which, however, constraints (4.6), (4.7) and (4.10) were not taken into consideration, have already been done in [26].

Let us denote with D the maximum cardinality of the set of feasible times for flight f to arrive at elementary sector j (or take off from/land at airport j if j is an airport instead of an elementary sector) taken over all f and j. This means that

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

Remember that, in this model, for every $f \in \mathcal{F}$ and $j \in P_f$ (note that j can also be an airport) we have $T^j_f = \{\underline{T}^j_f, \dots, \overline{T}^j_f\}$, where $\overline{T}^j_f = \underline{T}^j_f + \Delta_f$. Therefore, for every $f \in \mathcal{F}$ and $j \in P_f$, we have $|T^j_f| = 1 + \Delta_f$. Consequently, in this model, we also have that

$$D = 1 + \max_{f\in\mathcal{F}} \Delta_f.$$

Let us denote with X the maximum number of elements belonging to the sequence that describes the spatial trajectory of flight f (remember what we have seen in Subsection 2.3.3) taken over all f. In other words, we have

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

Let us denote with Y the maximum number of collapsed sectors contained in configuration m taken over all m. This means that

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

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

The exact number of variables of the type $w^j_{f,t}$ (where j can also be an airport) involved in the formulation is given by

$$\sum_{f \in \mathcal{F}} \sum_{j \in P_f} |T^j_f|$$

or, equivalently (in this model), by

$$\sum_{f \in \mathcal{F}} \sum_{j \in P_f} (1 + \Delta_f).$$

Since

$$\sum_{f \in \mathcal{F}} \sum_{j \in P_f} |T^j_f| \leq \sum_{f \in \mathcal{F}} \sum_{j \in P_f} \left(\max_{f \in \mathcal{F},\, j \in P_f} |T^j_f| \right) = \sum_{f \in \mathcal{F}} \sum_{j \in P_f} D \leq \sum_{f \in \mathcal{F}} XD = |\mathcal{F}|XD,$$

$|\mathcal{F}|XD$ is an upper bound on the number of variables of the type $w^j_{f,t}$.

The exact number of variables of the type $y_{m,t}$ involved in the formulation is $|\mathcal{M}||\mathcal{T}|$. There-

fore, the exact number of variables involved in the formulation is given by

$$|\mathcal{M}||\mathcal{T}| + \sum_{f\in\mathcal{F}}\sum_{j\in P_f} |T_f^j| \tag{4.18}$$

or, equivalently (in this model), by

$$|\mathcal{M}||\mathcal{T}| + \sum_{f\in\mathcal{F}}\sum_{j\in P_f} (1+\Delta_f). \tag{4.19}$$

An upper bound on the total number of variables is given by

$$|\mathcal{M}||\mathcal{T}| + |\mathcal{F}|XD. \tag{4.20}$$

Let us now compute the exact number of constraints involved in the formulation. There are exactly:

- $|\mathcal{K}||\mathcal{T}|$ constraints of the form (4.1);
- $|\mathcal{K}||\mathcal{T}|$ constraints of the form (4.2);
- $|\mathcal{T}| \cdot \sum_{m\in\mathcal{M}} |\mathcal{P}_m|$ constraints of the form (4.3);
- $\sum_{f\in\mathcal{F}} \left(\sum_{j\in P_f\setminus\{P(f,N_f)\}} |T_f^j| \right)$ constraints of the form (4.4);
- $\sum_{f\in\mathcal{F}}\sum_{j\in P_f} |T_f^j|$ constraints of the form (4.5);
- $\sum_{f\in\mathcal{F}}\sum_{j\in P_f} |T_f^j|$ constraints of the form (4.6);
- $\sum_{f\in\mathcal{F}} N_f$ constraints of the form (4.7);
- $|\mathcal{T}|$ constraints of the form (4.8);
- $|\mathcal{M}| \cdot \sum_{t\in\mathcal{T}} \min\{t+\tau-1-(t+1)+1, T-(t+1)+1\} = |\mathcal{M}| \cdot \sum_{t\in\mathcal{T}} \min\{\tau-1, T-t\}$ constraints of the form (4.9);
- $|\mathcal{M}||\mathcal{T}|$ constraints of the form (4.10).

Therefore, the exact number of constraints involved in the formulation is

$$\begin{aligned} 2|\mathcal{K}||\mathcal{T}| + |\mathcal{T}| \cdot \sum_{m\in\mathcal{M}} |\mathcal{P}_m| + \sum_{f\in\mathcal{F}} \left(\sum_{j\in P_f\setminus\{P(f,N_f)\}} |T_f^j| \right) + 2\sum_{f\in\mathcal{F}} \sum_{j\in P_f} |T_f^j| \\ + \sum_{f\in\mathcal{F}} N_f + |\mathcal{M}| \cdot \sum_{t\in\mathcal{T}} \min\{\tau - 1, T - t\} + |\mathcal{T}|(1 + |\mathcal{M}|). \end{aligned} \tag{4.21}$$

An upper bound on the total number of constraints is given by

$$2|\mathcal{K}||\mathcal{T}| + |\mathcal{T}||\mathcal{M}|Y + |\mathcal{F}|(X-1)D + 2|\mathcal{F}|XD + |\mathcal{F}|X + |\mathcal{M}|(|\mathcal{T}|-1) + |\mathcal{T}|(1+|\mathcal{M}|),$$

which can be rewritten as

$$2|\mathcal{K}||\mathcal{T}| + |\mathcal{T}||\mathcal{M}|Y + 3|\mathcal{F}|XD - |\mathcal{F}|D + |\mathcal{F}|X + 2|\mathcal{M}||\mathcal{T}| - |\mathcal{M}| + |\mathcal{T}|$$

or, equivalently, as

$$(2|\mathcal{K}| + 2|\mathcal{M}| + |\mathcal{M}|Y + 1)|\mathcal{T}| + (3XD - D + X)|\mathcal{F}| - |\mathcal{M}|. \tag{4.22}$$

To conclude this subsection, let us compute the upper bounds provided by (4.20) and (4.22) in the case of a real-size instance. Suppose we have the following situation:

- $|\mathcal{F}| = 10{,}000$, representing $10{,}000$ flights;
- $|\mathcal{K}| = 20$, representing 20 of the most important European airports;
- $|\mathcal{J}| = 200$, representing 200 elementary sectors;
- $|\mathcal{H}| = 100$, representing 100 collapsed sectors;
- $|\mathcal{M}| = 5$, representing 5 different possible configurations;
- $|\mathcal{T}| = 168$, representing a 14 hour day subdivided into five-minute intervals;
- $D = 7$, representing the fact that 30 minutes is, for each flight $f \in \mathcal{F}$, an upper bound on the maximum delay in arrival allowed for flight f;
- $X = 10$, representing an upper bound of 8 elementary sectors in a flight's path;

- $Y = 50$, representing an upper bound on the number of collapsed sectors that can be found in each configuration.

Substituting these values into (4.20) and (4.22), we find that this instance will surely have no more than $700{,}840$ variables and $2{,}180{,}563$ constraints belonging to those present in Subsection 4.2.3. Note that, since $|\mathcal{F}|XD = 700{,}000$, the parameters that significantly affect the upper bounds just calculated are $|\mathcal{F}|$, X and D (if, for example, one of these parameters doubles, then both the upper bounds approximately double).

5

A tighter formulation for the ATFM problem with dynamic selection of the airspace configuration

This fifth chapter is the core of the thesis. In the first section, we will prove the redundancy of the class of valid inequalities proposed in [26] and recalled at the end of Subsection 4.2.5. In the following three sections, we will present our contributions in order to tighten the formulation presented in [26]. Our contributions will be presented here from a theoretical viewpoint (the results of our computational experiments will be exhibited in Chapter 7) and basically consist in replacing some constants and adding new classes of valid inequalities. The new formulation thus obtained will be explicitly written down in the fifth section. Finally, in the sixth section, we will provide some estimates regarding the size of the new formulation.

5.1 A class of redundant valid inequalities

Let us recall the class of valid inequalities proposed in [26] and also present at the end of Subsection 4.2.5. The inequalities in question are the following:

$$\sum_{\substack{j\in\mathcal{B}_h:\ P(f,i)=j,\\ 1<i<N_f}} (w^j_{f,t} - w^j_{f,t-1}) \leq 1 \qquad \forall\, m \in \mathcal{M},\ h \in \mathcal{P}_m,\ f \in \mathcal{F},\ t \in \mathcal{T}. \tag{4.17}$$

Inequalities (4.17) stipulate that, for every $m \in \mathcal{M}$, $h \in \mathcal{P}_m$, $f \in \mathcal{F}$ and $t \in \mathcal{T}$, flight f can enter at most one elementary sector $j \in \mathcal{B}_h$ at time t.

For the sake of clarity, let us now prove that constraints (4.17) are indeed valid inequalities. Note that this fact, although intuitively very sensible, has not been proven in [26].

Proposition 5.1. *Constraints* (4.17) *are valid inequalities for the formulation present in Subsection 4.2.3.*

Proof. Let us consider an instance of the formulation present in Subsection 4.2.3 and let $(\overline{w}, \overline{y})$ be a solution of this instance (instead of listing all the $\overline{w}^j_{f,t}$s and all the $\overline{y}_{m,t}$s of our solution we use the more compact notation $(\overline{w}, \overline{y})$). Our goal is to prove that $(\overline{w}, \overline{y})$ satisfies constraints (4.17). Let us consider $\overline{m} \in \mathcal{M}$, $\overline{h} \in \mathcal{P}_{\overline{m}}$, $\overline{f} \in \mathcal{F}$ and $\overline{t} \in \mathcal{T}$. For our purposes it is sufficient to prove that

$$\sum_{\substack{j\in\mathcal{B}_{\overline{h}}:\ P(\overline{f},i)=j,\\ 1<i<N_{\overline{f}}}} (\overline{w}^j_{\overline{f},\overline{t}} - \overline{w}^j_{\overline{f},\overline{t}-1}) \leq 1. \tag{5.1}$$

Since

$$\{j \in \mathcal{B}_{\overline{h}} :\ P(\overline{f}, i) = j,\ 1 < i < N_{\overline{f}}\} \subseteq \{P(\overline{f}, i) :\ 1 < i < N_{\overline{f}}\},$$

remembering that the solution $(\overline{w}, \overline{y})$ satisfies constraints (4.5), we obtain

$$\sum_{\substack{j\in\mathcal{B}_{\overline{h}}:\ P(\overline{f},i)=j,\\ 1<i<N_{\overline{f}}}} (\overline{w}^j_{\overline{f},\overline{t}} - \overline{w}^j_{\overline{f},\overline{t}-1}) \leq \sum_{1<i<N_{\overline{f}}} (\overline{w}^{P(\overline{f},i)}_{\overline{f},\overline{t}} - \overline{w}^{P(\overline{f},i)}_{\overline{f},\overline{t}-1}). \tag{5.2}$$

The right-hand side of (5.2) can be rewritten as

$$(\overline{w}^{P(\overline{f},2)}_{\overline{f},\overline{t}} - \overline{w}^{P(\overline{f},2)}_{\overline{f},\overline{t}-1}) + \sum_{2<i<N_{\overline{f}}} (\overline{w}^{P(\overline{f},i)}_{\overline{f},\overline{t}} - \overline{w}^{P(\overline{f},i)}_{\overline{f},\overline{t}-1}). \tag{5.3}$$

Since the solution $(\overline{w}, \overline{y})$ satisfies constraints (4.4), remembering that $l_{\overline{f}}^{P(\overline{f},i)} \geq 1$ for every i such that $1 < i < N_{\overline{f}}$ (because, since $1 < i < N_{\overline{f}}$, we know that $P(\overline{f}, i)$ is an elementary sector), we have

$$\begin{aligned}&(\overline{w}_{\overline{f},\overline{t}}^{P(\overline{f},2)} - \overline{w}_{\overline{f},\overline{t}-1}^{P(\overline{f},2)}) + \sum_{2<i<N_{\overline{f}}} (\overline{w}_{\overline{f},\overline{t}}^{P(\overline{f},i)} - \overline{w}_{\overline{f},\overline{t}-1}^{P(\overline{f},i)}) \leq \\ &\leq (\overline{w}_{\overline{f},\overline{t}}^{P(\overline{f},2)} - \overline{w}_{\overline{f},\overline{t}-1}^{P(\overline{f},2)}) + \sum_{2<i<N_{\overline{f}}} (\overline{w}_{\overline{f},\overline{t}-1}^{P(\overline{f},i-1)} - \overline{w}_{\overline{f},\overline{t}-1}^{P(\overline{f},i)}).\end{aligned} \tag{5.4}$$

The right-hand side of (5.4) can be rewritten (taking advantage of the telescopic summation) as

$$\overline{w}_{\overline{f},\overline{t}}^{P(\overline{f},2)} - \overline{w}_{\overline{f},\overline{t}-1}^{P(\overline{f},N_{\overline{f}}-1)}, \tag{5.5}$$

which obviously (since we are using binary variables) satisfies

$$\overline{w}_{\overline{f},\overline{t}}^{P(\overline{f},2)} - \overline{w}_{\overline{f},\overline{t}-1}^{P(\overline{f},N_{\overline{f}}-1)} \leq 1 - 0 = 1. \tag{5.6}$$

Combining (5.2), (5.3), (5.4), (5.5) and (5.6), we finally get (5.1). Therefore constraints (4.17) are valid inequalities for the formulation present in Subsection 4.2.3.

□

At this point, it is worth observing carefully the proof of Proposition 5.1. We have that (5.2), (5.3), (5.4), (5.5) and (5.6) hold even if $(\overline{w}, \overline{y})$ is a solution of the instance's LPR (it is sufficient to repeat the exact same reasoning since in this reasoning, even to deduce (5.6), it is enough that we use real variables satisfying the instance's LPR). As a consequence, also (5.1) holds even if $(\overline{w}, \overline{y})$ is a solution of the instance's LPR. Keeping this observation in mind we will be able to prove very quickly the next proposition.

Proposition 5.2. *Valid inequalities* (4.17) *are redundant for the LPR of the formulation present in Subsection 4.2.3.*

Proof. Let us consider an instance of the formulation present in Subsection 4.2.3 and let $(\widetilde{w}, \widetilde{y})$ be a solution of this instance's LPR (instead of listing all the $\widetilde{w}_{f,t}^{j}$s and all the $\widetilde{y}_{m,t}$s of our solution we use the more compact notation $(\widetilde{w}, \widetilde{y})$). Our goal is to prove that $(\widetilde{w}, \widetilde{y})$ satisfies constraints (4.17). Let us consider $\widetilde{m} \in \mathcal{M}$, $\widetilde{h} \in \mathcal{P}_{\widetilde{m}}$, $\widetilde{f} \in \mathcal{F}$ and $\widetilde{t} \in \mathcal{T}$. For our purposes

it is sufficient to prove that

$$\sum_{\substack{j\in\mathcal{B}_{\widetilde{h}}:\, P(\widetilde{f},i)=j,\\ 1<i<N_{\widetilde{f}}}} (\widetilde{w}^{j}_{\widetilde{f},\widetilde{t}} - \widetilde{w}^{j}_{\widetilde{f},\widetilde{t}-1}) \leq 1. \tag{5.7}$$

As we have previously observed, to prove (5.7) it is sufficient to repeat the exact same arguments we have already made to prove Proposition 5.1. Therefore valid inequalities (4.17) are redundant for the LPR of the formulation present in Subsection 4.2.3.

□

5.2 Tightened values for model's constants

The formulation proposed in [26], and reviewed (with slightly different notation) in Subsection 4.2.1, has a considerable drawback: its LPR's feasible region is "really far" from the convex hull of the solutions of the ILP problem. To be more precise, in every computational experiment we carried out, regardless of whether the specific instance of the ILP problem was feasible or not, the optimal value of its LPR was always equal to 0. This is due to the excessively high values of the constants $\widetilde{C}_h(t)$, which can be exploited to ensure that constraints (4.3), already described in Subsection 4.2.5, are satisfied. The goal of this section is, therefore, to find new constants $C_h(t)$, as small as possible, such that, replacing constants $\widetilde{C}_h(t)$ with constants $C_h(t)$ inside constraints (4.3), the feasible region of any instance of the formulation present in Subsection 4.2.3 does not change.

Let us recall how constants $\widetilde{C}_h(t)$ are defined in [26]. For every $h \in \mathcal{H}$ and $t \in \mathcal{T}$, $\widetilde{C}_h(t)$ is given by the difference between the number of times a flight $f \in \mathcal{F}$ will, sooner or later, enter an elementary sector belonging to $\mathcal{B}_h$ and $S_h(t)$.

$$\widetilde{C}_h(t) = \left(\sum_{j\in\mathcal{B}_h} \left(\sum_{\substack{f\in\mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j',\\ 1<i<N_f}} 1 \right) \right) - S_h(t) \qquad \forall\, h \in \mathcal{H},\ t \in \mathcal{T} \tag{5.8}$$

Let us now define new constants $C_h(t)$ as follows:

$$C_h(t) := \left(\sum_{j \in \mathcal{B}_h} \left(\sum_{\substack{f \in \mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j', \\ 1<i<N_f,\, t \in T_f^j}} 1 \right) \right) - S_h(t) \qquad \forall\, h \in \mathcal{H},\ t \in \mathcal{T}. \quad (5.9)$$

For every $h \in \mathcal{H}$ and $t \in \mathcal{T}$, $C_h(t)$ is given by the difference between the number of times a flight $f \in \mathcal{F}$ could be within, at time t, an elementary sector belonging to $\mathcal{B}_h$ and $S_h(t)$. Repeating the exact same argument we have already made to prove the validity of inequalities (4.14), we get

$$C_h(t) \leq \widetilde{C}_h(t) \qquad \forall\, h \in \mathcal{H},\ t \in \mathcal{T}. \quad (5.10)$$

Remembering that inequalities (4.13) hold for every combination of binary variables, we deduce that it is possible to replace constants $\widetilde{C}_h(t)$ with constants $C_h(t)$. This means that, recalling the role of constants $\widetilde{C}_h(t)$ described in Subsection 4.2.1, each constant $C_h(t)$ also make capacity constraints redundant when related to h, t and a non-active airspace configuration. Therefore, we have almost proven the following proposition.

Proposition 5.3. *Replacing constants $\widetilde{C}_h(t)$ with constants $C_h(t)$ inside constraints* (4.3) *the feasible region of any instance of the formulation present in Subsection 4.2.3 does not change.*

Proof. As we have just seen, for every $h \in \mathcal{H}$ and $t \in \mathcal{T}$, both $\widetilde{C}_h(t)$ and $C_h(t)$ make capacity constraints redundant when related to h, t and a non-active airspace configuration. Furthermore, capacity constraints related to h, t and an active airspace configuration are, in both cases, the same (remember constraints (4.11)).

□

It is important to use, as far as possible, small constants inside capacity constraints (4.3) since, obviously, in this way we can try to obtain a tighter formulation. Let us now prove a proposition in this regard.

Proposition 5.4. *Constants $C_h(t)$ are as small as possible. This means that, given an instance of the formulation present in Subsection 4.2.3, it is not possible, in general, to replace even just one of these constants with a smaller one in constraints* (4.3) *without changing the instance's feasible region.*

Proof. Let us consider a very simple instance of the formulation present in Subsection 4.2.3. In this instance there are two flights (f_1 and f_2), four airports (AMS, DUB, FRA and LHR), four elementary sectors (a, b, c and d), four collapsed sectors ($C_1 = \{a, c\}$, $C_2 = \{b, d\}$, $C_3 = \{a, b\}$ and $C_4 = \{c, d\}$), two configurations ($M_1 = \{C_1, C_2\}$ and $M_2 = \{C_3, C_4\}$) and four time periods ($\mathcal{T} = \{1, 2, 3, 4\}$). The capacities of a, b, c and d are always 1, 0, 0 and 1 respectively and, therefore, the capacities of C_1, C_2, C_3 and C_4 are always all equal to 1. Airports DUB, AMS, LHR and FRA are entirely contained within a, b, c and d respectively. The spatial trajectory of flight f_1 is DUB, a, b, AMS, whereas the spatial trajectory of flight f_2 is LHR, c, d, FRA. The departure capacities of both DUB and LHR are always equal to 1. The arrival capacities of both AMS and FRA are always equal to 1. Using the notation already presented in Subsection 4.2.1, for every $h \in \mathcal{H}$ and $t \in \mathcal{T}$, we have $S_h(t) = 1$. Moreover: $l^{DUB}_{f_1} = 0$; $l^{a}_{f_1} = 1$; $l^{b}_{f_1} = 1$; $l^{LHR}_{f_2} = 0$; $l^{c}_{f_2} = 1$; $l^{d}_{f_2} = 1$; $d_{f_1} = 1$; $d_{f_2} = 1$; $r_{f_1} = 3$; $r_{f_2} = 3$; $\Delta_{f_1} = 1$; $\Delta_{f_2} = 1$; $c^{a}_{f_1} = c^{a}_{f_2} = 3$; $c^{g}_{f_1} = c^{g}_{f_2} = 1$; $\tau = 1$. It is easy to check that, using the constants $C_h(t)$ previously defined, the optimal value of our instance is 0 (any delay can be avoided if, for example, the active configuration is always M_2).

However, limiting ourselves to replace $C_{C_1}(1) = (1+1)-1 = 1$ with $\overline{C}_{C_1}(1) = C_{C_1}(1) - 1 = 0$ in constraints (4.3), it is no longer possible to avoid any delay. In fact, using configuration M_1 at time $t = 1$, the constraint $w^{DUB}_{f_1,1} + w^{LHR}_{f_2,1} \leq S_{C_1}(1) + \overline{C}_{C_1}(1)(1 - y_{M_1,1}) = 1 + 0 \cdot (1 - 1) = 1$ (which, remembering what we have seen at the end of Subsection 4.2.2, belongs to constraints (4.3)) forces at least one of the two flights to postpone its departure. On the other hand, using configuration M_2 at time $t = 1$, the constraint $w^{DUB}_{f_1,1} + w^{LHR}_{f_2,1} \leq S_{C_1}(1) + \overline{C}_{C_1}(1)(1 - y_{M_1,1}) = 1 + 0 \cdot (1 - 0) = 1$ (which, remembering what we have seen at the end of Subsection 4.2.2, belongs to constraints (4.3)) forces at least one of the two flights to postpone its departure. Therefore, limiting ourselves to replace $C_{C_1}(1) = (1 + 1) - 1 = 1$ with $\overline{C}_{C_1}(1) = C_{C_1}(1) - 1 = 0$ in constraints (4.3), it is no longer possible to avoid any delay. This means that the optimal value of our instance is no longer 0. It is easy to check that the optimal value after replacement is 1. In fact, if the active configuration is always M_1, a solution is obtained simply by postponing, for example, the departure of flight f_2 at time $t = 2$ (so that flight f_2 will arrive at airport FRA at time $t = 4$, whereas flight f_1 will respect its scheduled arrival time). Consequently, since it must be a natural number other than zero, the optimal value after replacement is 1.

Figure 5.1 represents the airspace of the instance considered in this proof.

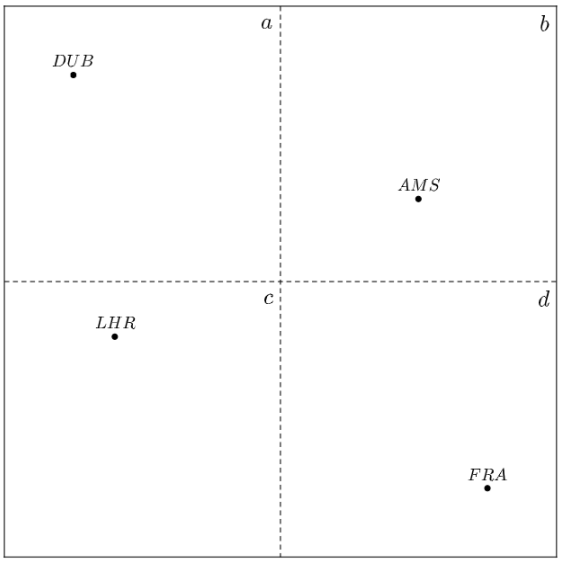

Figure 5.1: The airspace of the instance considered in the proof of Proposition 5.4.

□

As we have just proven, in order to tighten the formulation present in Subsection 4.2.3, the smallest constants that can be used in constraints (4.3) are the constants $C_h(t)$. We will see in Chapter 7 to what extent these new constants contribute to tighten, in the cases of the instances considered in this thesis, the formulation present in Subsection 4.2.3.

Finally, note that the possibility of replacing constants $\widetilde{C}_h(t)$ with constants $C_h(t)$ inside constraints (4.3) was pretty clear since Subsection 4.2.5. However, this possibility emerges so clearly only because in the formulation present in Subsection 4.2.3 we used a richer (but also heavier) notation for the summations than that used in [26] (the latter has the advantage of being lighter but the flaw of being less rigorous and, in this case, it does not allow you to notice this possibility so easily).

5.3 A new class of valid inequalities

Let us consider the following class of constraints:

$$\sum_{\substack{f\in\mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j',\\ 1<i<N_f,\, t\in T_f^j}} (w_{f,t}^{j} - w_{f,t}^{j'}) \leq \max_{h\in\mathcal{H}:\, j\in\mathcal{B}_h} S_h(t) \qquad \forall\, j\in\mathcal{J},\, t\in\mathcal{T}. \tag{5.11}$$

These constraints stipulate that, at any time period t and for every elementary sector j, the total number of flights present in j at time t will not exceed the maximum of the capacities of the collapsed sectors containing j. These are apparently rather weak constraints compared to constraints (4.3). In fact, for every $j \in \mathcal{J}$ and $t \in \mathcal{T}$, constraints (5.11) provide an upper bound on the number of flights present in j at time t, but j in an elementary sector (and, in general, not a collapsed sector) whereas the upper bound is a capacity of a collapsed sector that contains j (and, in general, this collapsed sector is strictly larger than j). One might therefore expect that, as it has already happened with constraints (4.17), constraints (5.11) will also be both valid inequalities for the formulation present in Subsection 4.2.3 and redundant constraints for the LPR of the same formulation. However, this is not true. Indeed, as we will see in this subsection, constraints (5.11) are really valid inequalities for the formulation present in Subsection 4.2.3 but they also have the advantage of not being redundant for the LPR of the same formulation.

First of all, let us prove the following proposition.

Proposition 5.5. *Constraints* (5.11) *are valid inequalities for the formulation present in Subsection 4.2.3.*

Proof. Let us consider an instance of the formulation present in Subsection 4.2.3 and let $(\overline{w}, \overline{y})$ be a solution of this instance (instead of listing all the $\overline{w}^j_{f,t}$s and all the $\overline{y}_{m,t}$s of our solution we use the more compact notation $(\overline{w}, \overline{y})$). Our goal is to prove that $(\overline{w}, \overline{y})$ satisfies constraints (5.11). Let us consider $\overline{j} \in \mathcal{J}$ and $\overline{t} \in \mathcal{T}$. For our purposes it is sufficient to prove that

$$\sum_{\substack{f \in \mathcal{F}:\, P(f,i)=\overline{j},\, P(f,i+1)=j', \\ 1<i<N_f,\, \overline{t} \in T^{\overline{j}}_f}} (\overline{w}^{\overline{j}}_{f,\overline{t}} - \overline{w}^{j'}_{f,\overline{t}}) \leq \max_{h \in \mathcal{H}:\, \overline{j} \in \mathcal{B}_h} S_h(\overline{t}). \tag{5.12}$$

Let us denote with $\overline{m}$ the configuration, active at time $\overline{t}$, associated to the solution $(\overline{w}, \overline{y})$. Let $\overline{h}$ be the collapsed sector, belonging to $\overline{m}$, that contains $\overline{j}$. Obviously, since

$$\{f \in \mathcal{F} :\ P(f,i) = \overline{j},\ P(f,i+1) = j',\ 1 < i < N_f,\ \overline{t} \in T^{\overline{j}}_f\} \subseteq$$

$$\subseteq \{f \in \mathcal{F} :\ P(f,i) = j,\ P(f,i+1) = j',\ j \in \mathcal{B}_{\overline{h}},\ 1 < i < N_f,\ \overline{t} \in T^{j}_f\},$$

we have that

$$\sum_{\substack{f\in\mathcal{F}:\, P(f,i)=\overline{j},\, P(f,i+1)=j',\\ 1<i<N_f,\, \overline{t}\in T_f^{\overline{j}}}} (\overline{w}^{\overline{j}}_{f,\overline{t}} - \overline{w}^{j'}_{f,\overline{t}}) \leq$$

$$\leq \sum_{j\in\mathcal{B}_{\overline{h}}} \left(\sum_{\substack{f\in\mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j',\\ 1<i<N_f,\, \overline{t}\in T_f^{j}}} (\overline{w}^{j}_{f,\overline{t}} - \overline{w}^{j'}_{f,\overline{t}}) \right). \tag{5.13}$$

Remembering that the solution $(\overline{w}, \overline{y})$ satisfies constraints (4.3) and that $y_{\overline{m},\overline{t}} = 1$, we have

$$\sum_{j\in\mathcal{B}_{\overline{h}}} \left(\sum_{\substack{f\in\mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j',\\ 1<i<N_f,\, \overline{t}\in T_f^{j}}} (\overline{w}^{j}_{f,\overline{t}} - \overline{w}^{j'}_{f,\overline{t}}) \right) \leq S_{\overline{h}}(\overline{t}). \tag{5.14}$$

Since $\overline{h} \in \{h \in \mathcal{H} :\ \overline{j} \in \mathcal{B}_h\}$, we (obviously) have that

$$S_{\overline{h}}(\overline{t}) \leq \max_{h\in\mathcal{H}:\, \overline{j}\in\mathcal{B}_h} S_h(\overline{t}). \tag{5.15}$$

Combining (5.13), (5.14) and (5.15), we finally get (5.12). Therefore constraints (5.11) are valid inequalities for the formulation present in Subsection 4.2.3.

□

Let us now investigate the relationship between constraints (4.3) and constraints (5.11) better. As we have just seen in Proposition 5.5, constraints (5.11) are valid inequalities for the formulation present in Subsection 4.2.3. One might therefore ask whether it is possible to replace constraints (4.3) with constraints (5.11) in the formulation present in Subsection 4.2.3 without changing the feasible region whatever the instance. The answer, as the reader might expect, is negative.

Proposition 5.6. *Replacing constraints* (4.3) *with constraints* (5.11) *in the formulation present in Subsection 4.2.3, the feasible region can change.*

Proof. Let us consider a simple instance (whose airspace is the same represented in Figure 5.1). In this instance there are four flights (f_1, f_2, f_3 and f_4), four airports (AMS, DUB, FRA and LHR), four elementary sectors (a, b, c and d), four collapsed sectors ($C_1 = \{a, c\}$, $C_2 =$

$\{b, d\}$, $C_3 = \{a, b\}$ and $C_4 = \{c, d\}$), two configurations ($M_1 = \{C_1, C_2\}$ and $M_2 = \{C_3, C_4\}$) and five time periods ($\mathcal{T} = \{1, 2, 3, 4, 5\}$). The capacities of a, b, c and d are always 1, 0, 0 and 1 respectively and, therefore, the capacities of C_1, C_2, C_3 and C_4 are always all equal to 1. Airports DUB, AMS, LHR and FRA are entirely contained within a, b, c and d respectively. DUB, a, b, AMS is the spatial trajectory of both f_1 and f_2. LHR, c, d, FRA is the spatial trajectory of both f_3 and f_4. The departure capacities of both DUB and LHR are always equal to 2. The arrival capacities of both AMS and FRA are always equal to 2. Using the notation already presented in Subsection 4.2.1, for every $h \in \mathcal{H}$ and $t \in \mathcal{T}$, we have $S_h(t) = 1$. Moreover: $l_{f_1}^{DUB} = l_{f_2}^{DUB} = 0$; $l_{f_1}^{a} = l_{f_2}^{a} = 1$; $l_{f_1}^{b} = l_{f_2}^{b} = 1$; $l_{f_3}^{LHR} = l_{f_4}^{LHR} = 0$; $l_{f_3}^{c} = l_{f_4}^{c} = 1$; $l_{f_3}^{d} = l_{f_4}^{d} = 1$; $d_{f_1} = d_{f_3} = 1$; $d_{f_2} = d_{f_4} = 2$; $r_{f_1} = r_{f_3} = 3$; $r_{f_2} = r_{f_4} = 4$; $\Delta_{f_1} = \Delta_{f_2} = \Delta_{f_3} = \Delta_{f_4} = 1$; $c_{f_1}^{a} = c_{f_2}^{a} = c_{f_3}^{a} = c_{f_4}^{a} = 3$; $c_{f_1}^{g} = c_{f_2}^{g} = c_{f_3}^{g} = c_{f_4}^{g} = 1$; $\tau = 1$.

It is easy to check that, using the formulation present in Subsection 4.2.3 (where the constants inside constraints (4.3) can be indifferently, as we have seen in Proposition 5.3, the $C_h(t)$s or the $\widetilde{C}_h(t)$s), this instance has a nonempty feasible region. In fact, using always configuration M_2, it is possible to respect the scheduled arrival time ($t = 3$) of both f_1 and f_3 and to postpone the departure of both f_2 and f_4 at time $t = 3$ so that both f_2 and f_4 arrive at their destination airports at time $t = 5$. Consequently there is at least one solution and the value of the objective function, associated with this solution, is 2. Note also that, since the airspace is the same represented in Figure 5.1, there cannot be three or more planes in the air at the same time. Otherwise both M_1 and M_2 (one of which must be active) would contain a collapsed sector with two flights simultaneously inside (and, remembering that $S_h(t) = 1$ for every $h \in \mathcal{H}$ and $t \in \mathcal{T}$, this is not possible due to constraints (4.3)). Therefore there can be at most two planes in the air at the same time and, as a consequence, 2 (the value of the objective function associated with the previously described solution) is also the optimal value.

On the other hand, replacing constraints (4.3) with constraints (5.11) in the formulation present in Subsection 4.2.3, the optimal value of this instance is 0 (and no longer 2). In fact, one can easily check that it is possible to respect the scheduled arrival time of each flight.

We have therefore proven that, in this instance, any optimal solution of the formulation obtained replacing constraints (4.3) with constraints (5.11) is not a solution of the formulation present in Subsection 4.2.3 (where the constants inside constraints (4.3) can be indifferently,

as we have seen in Proposition 5.3, the $C_h(t)$s or the $\widetilde{C}_h(t)$s). This concludes our proof.

□

To conclude this section, let us prove that constraints (5.11) are not redundant for the LPR of the formulation present in Subsection 4.2.3 (we will see in Chapter 7 to what extent the addition of these new constraints contribute to tighten, in the cases of the instances considered in this thesis, the formulation present in Subsection 4.2.3). This proof would not be necessary in light of the results we will see in Chapter 7. However, for the sake of clarity, we preferred to provide a proof of the next proposition (which, at first glance, may not seem so intuitive).

Proposition 5.7. *Constraints* (5.11) *are not redundant for the LPR of the formulation present in Subsection 4.2.3. This is true even if, inside constraints* (4.3)*, we use constants* $C_h(t)$ *instead of constants* $\widetilde{C}_h(t)$.

Proof. Our goal is to prove that there exists at least one instance such that, adding constraints (5.11) to the LPR of the formulation present in Subsection 4.2.3 (where the constants inside constraints (4.3) can be the $C_h(t)$s or the $\widetilde{C}_h(t)$s), its feasible region change. Let us consider the same instance that appears in the proof of Proposition 5.6 except for the values of the $S_h(t)$s. In fact, in this instance we have $S_h(t) = 0$ for every $h \in \mathcal{H}$ and $t \in \mathcal{T}$ (the airspace is the same represented in Figure 5.1). This means that, in this instance, the capacities of a, b, c and d are always all equal to 0.

Using the formulation present in Subsection 4.2.3, this instance's LPR admits (even if, inside constraints (4.3), we use constants $C_h(t)$ instead of constants $\widetilde{C}_h(t)$) the following solution: $w^{DUB}_{f_1,1} = 0.5$; $w^{DUB}_{f_1,2} = 1$; $w^{a}_{f_1,1} = 0.5$; $w^{a}_{f_1,2} = 1$; $w^{b}_{f_1,2} = 0.5$; $w^{b}_{f_1,3} = 1$; $w^{AMS}_{f_1,3} = 0.5$; $w^{AMS}_{f_1,4} = 1$; $w^{DUB}_{f_2,2} = 0.5$; $w^{DUB}_{f_2,3} = 1$; $w^{a}_{f_2,2} = 0.5$; $w^{a}_{f_2,3} = 1$; $w^{b}_{f_2,3} = 0.5$; $w^{b}_{f_2,4} = 1$; $w^{AMS}_{f_2,4} = 0.5$; $w^{AMS}_{f_2,5} = 1$; $w^{LHR}_{f_3,1} = 0.5$; $w^{LHR}_{f_3,2} = 1$; $w^{c}_{f_3,1} = 0.5$; $w^{c}_{f_3,2} = 1$; $w^{d}_{f_3,2} = 0.5$; $w^{d}_{f_3,3} = 1$; $w^{FRA}_{f_3,3} = 0.5$; $w^{FRA}_{f_3,4} = 1$; $w^{LHR}_{f_4,2} = 0.5$; $w^{LHR}_{f_4,3} = 1$; $w^{c}_{f_4,2} = 0.5$; $w^{c}_{f_4,3} = 1$; $w^{d}_{f_4,3} = 0.5$; $w^{d}_{f_4,4} = 1$; $w^{FRA}_{f_4,4} = 0.5$; $w^{FRA}_{f_4,5} = 1$; $y_{M_1,1} = y_{M_1,2} = y_{M_1,3} = y_{M_1,4} = y_{M_1,5} = y_{M_2,1} = y_{M_2,2} = y_{M_2,3} = y_{M_2,4} = y_{M_2,5} = 0.5$. Therefore, using the formulation present in Subsection 4.2.3, this instance's LPR has (even if, inside constraints (4.3), we use constants $C_h(t)$ instead of constants $\widetilde{C}_h(t)$) a nonempty feasible region.

However this solution does not satisfy constraints (5.11) and, more generally, no solution of this instance's LPR satisfy constraints (5.11). In this case, in fact, constraints (5.11) (remembering what we have seen at the end of Subsection 4.2.2, constraints (4.4) and our assumptions on

the l_f^js) force all the $w_{f,t}^j$s to be equal to 0, contradicting constraints (4.7). Therefore, adding constraints (5.11) to the formulation present in Subsection 4.2.3 (where the constants inside constraints (4.3) can be the $C_h(t)$s or the $\widetilde{C}_h(t)$s), this instance's LPR has no solution (the feasible region is empty).

We have therefore proven that in this instance, adding constraints (5.11) to the LPR of the formulation present in Subsection 4.2.3 (where the constants inside constraints (4.3) can be the $C_h(t)$s or the $\widetilde{C}_h(t)$s), the feasible region change. This concludes our proof.

□

5.4 A class of valid cover-like inequalities

Replacing constants $\widetilde{C}_h(t)$ with constants $C_h(t)$ inside constraints (4.3), we get the following constraints:

$$\sum_{j \in \mathcal{B}_h} \left(\sum_{\substack{f \in \mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j', \\ 1<i<N_f,\, t \in T_f^j}} (w_{f,t}^j - w_{f,t}^{j'}) \right) \leq \\ \leq S_h(t) + C_h(t)(1 - y_{m,t}) \qquad \forall\, m \in \mathcal{M},\ h \in \mathcal{P}_m,\ t \in \mathcal{T}. \tag{5.16}$$

Let us denote with $\mathfrak{F}$ the formulation obtained by adding constraints (5.11) and replacing constraints (4.3) with constraints (5.16) in the formulation present in Subsection 4.2.3.

Our goal is to apply an algorithm very similar to Algorithm 3.1 to formulation $\mathfrak{F}$ in order to obtain some cover inequalities. To do this, we observe that formulation $\mathfrak{F}$ can be rewritten in a form very similar to (3.9). First of all, in both $\mathfrak{F}$ and (3.9) the variables involved are binary variables. Let us denote with the compact notation $f(w)$ the objective function of formulation $\mathfrak{F}$. Since, using a not very formal notation (a formal notation is not necessary to express the following well-known property), $\min f(w) = -\max(-f(w))$, it is possible to replace $\min f(w)$ with $\max(-f(w))$ in formulation $\mathfrak{F}$ and solve the instance in question (remembering that, once the instance has been solved, it will be necessary to change the sign of the optimal value obtained if the latter exists). Alternatively, it is possible to observe that all the reasoning related to cover inequalities that we made in Section 3.4 holds even if we replace $\max$ with $\min$ in both (3.4) and (3.9). Therefore the objective function is in no way a prob-

lem for the purposes of rewriting formulation $\mathfrak{F}$ in the form (3.9). Remember that, in (3.9), constraints $Bx \leq d$ have no particular conditions on B and d. Constraints (4.1), (4.2), (4.4) and (5.11) can be easily seen as constraints of the type $Bx \leq d$. Also constraints (4.5) can be easily seen as constraints of the type $Bx \leq d$ (just multiply both members of each constraint by -1). Since a system of the type $\widetilde{B}x = \widetilde{d}$ is equivalent to the system given by $\widetilde{B}x \leq \widetilde{d}$ and $-\widetilde{B}x \leq -\widetilde{d}$, also constraints (4.7) and (4.8) can be easily seen as constraints of the type $Bx \leq d$. Also constraints (4.9) can be easily seen as constraints of the type $Bx \leq d$, it is sufficient to rewrite them as follows:

$$y_{m,t} - y_{m,t-1} - y_{m,u} \leq 0 \quad \forall\, m \in \mathcal{M},\ t \in \mathcal{T},\ u \in \{t+1, \ldots, \min\{t+\tau-1, T\}\}.$$

Therefore, temporarily ignoring constraints (5.16), formulation $\mathfrak{F}$ can be rewritten in the form (3.9). Note that, for the moment, no constraints of formulation $\mathfrak{F}$ have been interpreted as constraints of the system $Ax \leq b$ present in (3.9). Let us now focus on constraints (5.16). Keeping in mind how constants $C_h(t)$ are defined, constraints (5.16) can be rewritten as

$$\begin{aligned} C_h(t) \cdot y_{m,t} + \sum_{j \in \mathcal{B}_h} \left(\sum_{\substack{f \in \mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j', \\ 1<i<N_f,\, t \in T_f^j}} (w_{f,t}^{j} - w_{f,t}^{j'}) \right) \leq \\ \leq \sum_{j \in \mathcal{B}_h} \left(\sum_{\substack{f \in \mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j', \\ 1<i<N_f,\, t \in T_f^j}} 1 \right) \qquad \forall\, m \in \mathcal{M},\ h \in \mathcal{P}_m,\ t \in \mathcal{T}. \end{aligned} \tag{5.17}$$

Note that (also in this more convenient form) constraints (5.17) cannot be seen as constraints of the system $Ax \leq b$ present in (3.9). In fact, in general (remember what we have seen at the end of Subsection 4.2.2), the coefficients on the left-hand sides of constraints (5.17) are not always natural numbers. These coefficients are, typically, 1, 0 (associated with variables that do not appear on the left-hand sides), -1 and the $C_h(t)$s. The only problematic coefficients are the various -1 (which are strictly negative) and the $C_h(t)$s (which could be strictly negative). The first critical issue, related to the -1 coefficients, can be fixed with a simple trick. In fact, since we are using binary variables, constraints (4.4) imply that the various $w_{f,t}^{j} - w_{f,t}^{j'}$ are such that $w_{f,t}^{j} - w_{f,t}^{j'} \in \{0, 1\}$. Denoting the various $w_{f,t}^{j} - w_{f,t}^{j'}$ with the notation $u_{f,t}^{j,j'}$,

constraints (5.17) can be rewritten as

$$\begin{aligned} & C_h(t) \cdot y_{m,t} + \sum_{j \in \mathcal{B}_h} \left(\sum_{\substack{f \in \mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j', \\ 1<i<N_f,\, t \in T_f^j}} u_{f,t}^{j,j'} \right) \leq \\ & \leq \sum_{j \in \mathcal{B}_h} \left(\sum_{\substack{f \in \mathcal{F}:\, P(f,i)=j,\, P(f,i+1)=j', \\ 1<i<N_f,\, t \in T_f^j}} 1 \right) \qquad \forall\, m \in \mathcal{M},\ h \in \mathcal{P}_m,\ t \in \mathcal{T}. \end{aligned} \tag{5.18}$$

Pretending that the various $u_{f,t}^{j,j'}$ are binary variables, constraints (5.18) can almost be been seen as constraints of the system $Ax \leq b$ present in (3.9), the only critical aspect is related to the presence of constants $C_h(t)$, which could be strictly negative. However, despite the presence of constants $C_h(t)$, it is possible to look for cover inequalities for constraints (5.18) as if the latter were real knapsack-type constraints (recall (3.4)). Since constraints (5.18) are constraints of the form (5.19), the previous fact is an immediate consequence of the following simple proposition.

Proposition 5.8. *Given a constraint of the form*

$$\alpha x_{n+1} + \sum_{i=1}^{n} x_i \leq n, \tag{5.19}$$

where $n \in \mathbb{N}$ (if $n = 0$, then the constraint in question is nothing other than $\alpha x_1 \leq 0$), $\alpha \in \mathbb{Z}$ and $x_1, \ldots, x_{n+1} \in \{0, 1\}$ are binary variables, it is possible to look for cover inequalities in the exact same way described in Subsection 3.4.1.

Proof. If $\alpha \geq 0$, then there is nothing to prove (we are in the same conditions present in Subsection 3.4.1). But what happens if $\alpha < 0$? Also in this case, there is no problem in looking for cover inequalities since there are no covers (and, therefore, there are no cover inequalities). In fact, to find a cover inequality we need to find a cover. This means that in this case, remembering what we have seen in Subsection 3.4.1 (recall, in particular, (3.5)), we need to find for which values of the binary variables $z_1, \ldots, z_{n+1}$ we have that

$$\alpha z_{n+1} + \sum_{i=1}^{n} z_i \geq n + 1. \tag{5.20}$$

Since we are using binary variables and $\alpha < 0$, the maximum possible value for the left-hand side of (5.20) is n (setting $z_1, \ldots, z_n = 1$ and $z_{n+1} = 0$). Since there are no values of the binary variables $z_1, \ldots, z_{n+1}$ such that (5.20) holds, there are no covers. Since there are no covers, there cannot be cover inequalities and, therefore, there are no issues in looking for cover inequalities in the exact same way described in Subsection 3.4.1 (despite the fact that $\alpha < 0$). □

As a consequence of what we have just seen, it is possible to look for cover inequalities for constraints (5.18) as if the latter were real knapsack-type constraints (in the same way described in Section 3.4). Any cover inequality added to the formulation in this way will be expressed using the quantities $u_{f,t}^{j,j'}$. At this point, for each of these cover inequalities, it will be sufficient to replace the various $u_{f,t}^{j,j'}$ with the corresponding $w_{f,t}^{j} - w_{f,t}^{j'}$ in order to obtain the cover-like inequalities, which from now on we will simply call cover inequalities, that will be added to formulation $\mathfrak{F}$. In Algorithm 5.1, the procedure used to obtain the formulation that will be presented in the next section is summarized (to understand in more detail how cover inequalities are added to formulation $\mathfrak{F}$, the reader can also consult the code present in the appendix).

Algorithm 5.1: Generation of the formulation present in Section 5.5 for a given instance of formulation $\mathfrak{F}$

Step 0: Suppose we want to find an optimal solution for a given instance of formulation $\mathfrak{F}$.

Step 1: Solve the instance's LPR. If there is no solution, then **STOP**: the instance is infeasible. Otherwise let $(\overline{w}, \overline{y})$ be the optimal solution obtained.

Step 2: Check if $(\overline{w}, \overline{y})$ is integer. If this is true, then **STOP**: an optimal solution of the instance has been found. Otherwise proceed to the next step.

Step 3: For every constraint belonging to constraints (5.16) check whether there are cover inequalities violated by $(\overline{w}, \overline{y})$ (we have seen how it is possible to do this). If there are no cover inequalities violated by $(\overline{w}, \overline{y})$, then **STOP**. Otherwise add to formulation $\mathfrak{F}$ all the cover inequalities found so far and go back to **Step 1**.

Note that the formulation generated using Algorithm 5.1 is not at all immediate to obtain since, unlike both formulation $\mathfrak{F}$ and the formulation proposed in [26], the number of inequalities is huge and a separation procedure is appropriate. However, this is a tighter formulation.

We will see in Chapter 7 to what extent this use of cover inequalities contributes to tighten even further, in the cases of the instances considered in this thesis, the formulation present in Subsection 4.2.3. We will also see, again in Chapter 7, how the formulation generated using Algorithm 5.1 takes, given an instance of formulation $\mathfrak{F}$, quite a lot of time to be obtained.

To conclude this section, let us clarify an important point. As we have seen, with some tricks, constraints (5.16) can be treated as if they were real knapsack-type constraints. One of these tricks allowed us to eliminate the problem of the -1 coefficients. One might wonder whether, using essentially the same trick, other constraints of formulation $\mathfrak{F}$ could also be treated as if they were real knapsack-type constraints (in this way we could look for further cover inequalities). The answer is that, although it is possible to treat other constraints of formulation $\mathfrak{F}$ as if they were real knapsack-type constraints (this is quite evident, for example, for constraints (4.1), (4.2) and (5.11)), this does not allow to tighten even further formulation $\mathfrak{F}$ (and, actually, not even to tighten the formulation present in Subsection 4.2.3). This is due to the fact that all possible cover inequalities obtainable in this way are redundant for the LPR of formulation $\mathfrak{F}$ (and also for the LPR of the formulation present in Subsection 4.2.3). This observation we have just made is an immediate consequence of the following proposition.

Proposition 5.9. *Let us consider a constraint of the form*

$$\sum_{i=1}^{n} x_i \leq m, \tag{5.21}$$

where $m, n \in \mathbb{N}$, $n > 0$ *and* $x_1, \ldots, x_n \in \{0, 1\}$ *are binary variables. All possible cover inequalities obtainable by interpreting* (5.21) *as a knapsack-type constraint are, also relaxing the integrality constraints on the variables, redundant. This means that, besides being obviously valid inequalities, all possible cover inequalities obtainable by interpreting* (5.21) *as a knapsack-type constraint are satisfied by every combination of* $x_1, \ldots, x_n \in [0, 1]$ *that satisfies* (5.21).

Proof. If $m \geq n$, then there is nothing to prove (since there are no covers and, therefore, no cover inequalities). Consequently we can assume that $m < n$. Let us consider a cover C. Since C is a cover, we have that $|C| = \sum_{i \in C} 1 > m$ and, therefore,

$$m \leq |C| - 1. \tag{5.22}$$

The cover inequality associated with cover C is

$$\sum_{i \in C} x_i \leq |C| - 1. \tag{5.23}$$

Let us consider $\overline{x}_1, \ldots, \overline{x}_n \in [0, 1]$ satisfying (5.21). Since C is a cover, we have

$$\sum_{i \in C} \overline{x}_i \leq \sum_{i=1}^{n} \overline{x}_i. \tag{5.24}$$

Since $\overline{x}_1, \ldots, \overline{x}_n$ satisfy (5.21), we have that

$$\sum_{i=1}^{n} \overline{x}_i \leq m. \tag{5.25}$$

Combining (5.24), (5.25) and (5.22), we deduce that $\overline{x}_1, \ldots, \overline{x}_n$ satisfy (5.23). Since this reasoning holds for any combination of $x_1, \ldots, x_n \in [0, 1]$ that satisfies (5.21), we deduce that the cover inequality (5.23) is redundant.

Since the reasoning made for cover C holds for any possible cover inequality obtainable by interpreting (5.21) as a knapsack-type constraint, we have proven this proposition.

□

In the next section we will write down explicitly the formulation generated using Algorithm 5.1 (always keeping in mind what we have seen at the end of Subsection 4.2.2).

5.5 The new formulation

Based on the elements seen so far, we can finally write down explicitly our new formulation.

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

subject to

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

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

$$\sum_{j\in\mathcal{B}_h}\left(\sum_{\substack{f\in\mathcal{F}:\ P(f,i)=j,\ P(f,i+1)=j',\\ 1<i<N_f,\ t\in T_f^j}} (w_{f,t}^j - w_{f,t}^{j'})\right) \le \tag{5.16}$$

$$\le S_h(t) + C_h(t)(1-y_{m,t}) \qquad \forall\, m\in\mathcal{M},\ h\in\mathcal{P}_m,\ t\in\mathcal{T}$$

$$\sum_{\substack{f\in\mathcal{F}:\ P(f,i)=j,\ P(f,i+1)=j',\\ 1<i<N_f,\ t\in T_f^j}} (w_{f,t}^j - w_{f,t}^{j'}) \le \max_{h\in\mathcal{H}:\ j\in\mathcal{B}_h} S_h(t) \qquad \forall\, j\in\mathcal{J},\ t\in\mathcal{T} \tag{5.11}$$

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

$$w_{f,t}^j - w_{f,t-1}^j \ge 0 \qquad \forall\, f\in\mathcal{F},\ j\in P_f,\ t\in T_f^j \tag{4.5}$$

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

$$w_{f,\overline{T}_f^j}^j = 1 \qquad \forall\, f\in\mathcal{F},\ j\in P_f \tag{4.7}$$

$$\sum_{m\in\mathcal{M}} y_{m,t} = 1 \qquad \forall\, t\in\mathcal{T} \tag{4.8}$$

$$y_{m,t} - y_{m,t-1} \le y_{m,u} \quad \forall\, m\in\mathcal{M},\ t\in\mathcal{T},\ u\in\{t+1,\dots,\min\{t+\tau-1,T\}\} \tag{4.9}$$

$$y_{m,t}\in\{0,1\} \qquad \forall\, m\in\mathcal{M},\ t\in\mathcal{T} \tag{4.10}$$

cover inequalities generated using constraints (5.16). (5.26)

5.6 Size of the new formulation

Note that the variables that appear in the new formulation are exactly the same ones that appear in the formulation present in Subsection 4.2.3. Therefore, remembering what we have seen in Subsection 4.2.6, the exact number of variables involved in the new formulation is given by

$$|\mathcal{M}||\mathcal{T}| + \sum_{f\in\mathcal{F}}\sum_{j\in P_f} |T_f^j| \tag{4.18}$$

or, equivalently, by

$$|\mathcal{M}||\mathcal{T}| + \sum_{f \in \mathcal{F}} \sum_{j \in P_f} (1 + \Delta_f). \tag{4.19}$$

Furthermore, an upper bound on the total number of variables is given by

$$|\mathcal{M}||\mathcal{T}| + |\mathcal{F}|XD. \tag{4.20}$$

Regarding the exact number of constraints involved in the new formulation, we have already calculated (in Subsection 4.2.6) how many constraints (4.1), (4.2), (4.4), (4.5), (4.6), (4.7), (4.8), (4.9) and (4.10) there are exactly. Since the exact number of constraints (5.16) obviously coincides with the number of constraints (4.3), remembering what we have seen in Subsection 4.2.6, we deduce that the total number of constraints involved in the new formulation is given by the sum of (4.21) with the number of constraints (5.11) and (5.26). Since the exact number of constraints (5.11) is $|\mathcal{J}||\mathcal{T}|$, the total number of constraints involved in the new formulation is given by

$$\begin{aligned} &2|\mathcal{K}||\mathcal{T}| + |\mathcal{T}| \cdot \sum_{m \in \mathcal{M}} |\mathcal{P}_m| + \sum_{f \in \mathcal{F}} \left(\sum_{j \in P_f \setminus \{P(f,N_f)\}} |T_f^j| \right) + 2 \sum_{f \in \mathcal{F}} \sum_{j \in P_f} |T_f^j| \\ &+ \sum_{f \in \mathcal{F}} N_f + |\mathcal{M}| \cdot \sum_{t \in \mathcal{T}} \min\{\tau - 1, T - t\} + |\mathcal{T}|(1 + |\mathcal{M}|) + |\mathcal{J}||\mathcal{T}| + n_{cover}, \end{aligned} \tag{5.27}$$

where n_{cover} denotes the number of cover inequalities (5.26). We recall that, as discussed in Section 3.4, the number of constraints (5.26) is exponential in theory. However, only a generally small subset of them will be actually included in the formulation by the separation Algorithm 5.1, and their number cannot be determined a priori.

6

Model implementation and instance generation

In the first section of this sixth chapter, we will describe how the instances used for our computational experiments were generated. To conclude the chapter, in the second section, we will illustrate, through a description of the .mod, .dat and .run files, the actual implementation of the model (described through the new formulation) in AMPL.

6.1 Generation of the instances

The instances on which we carried out our computational experiments were generated exactly as in [26] and, as will be explained in this section, some of these are precisely instances used in [26]. All the instances on which we carried out our simulations were created using the same airspace. Furthermore, in each of these instances, there are always the same three possible airspace configurations.

6.1.1 Airspace and configurations

Figure 6.1 shows the airspace, taken from [26], of the instances used for our computational experiments. This means that in each instance there are 8 airports (DUB, CPH, LHR, AMS, CDG, FRA, ZRH and VIE) and 16 elementary sectors ($a, b, c, d, e, f, g, h, i, j, k, l, m, n, o$

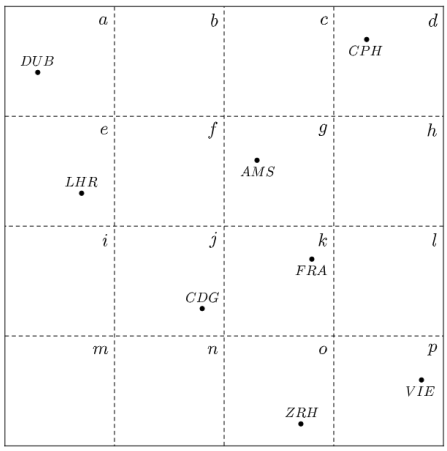

Figure 6.1: The airspace of the instances used for our computational experiments [26].

and p) arranged as shown in the figure.

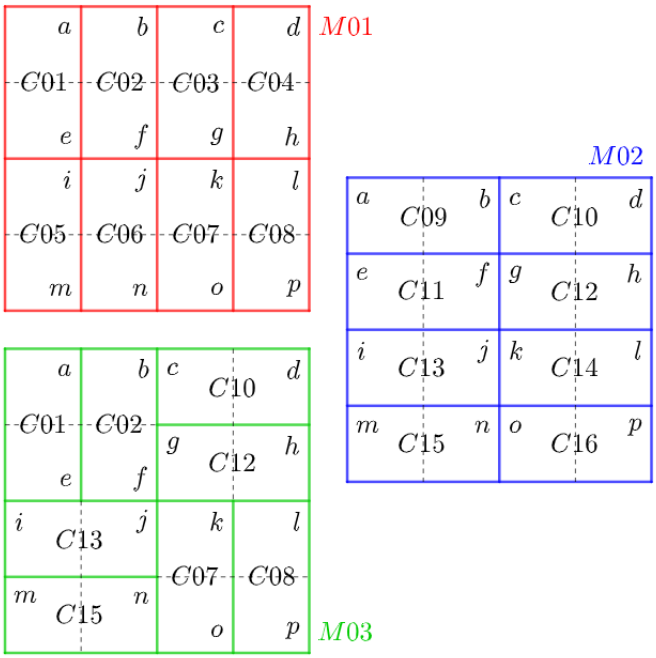

Figure 6.2: The three possible configurations in our instances [26].

Figure 6.2 represents the three possible configurations, taken from [26], present in each instance used for our computational experiments (to keep Figure 6.2 more readable the airports have not been represented). This means that in each instance there are 16 collapsed sectors

($C01 = \{a, e\}$, $C02 = \{b, f\}$, $C03 = \{c, g\}$, $C04 = \{d, h\}$, $C05 = \{i, m\}$, $C06 = \{j, n\}$, $C07 = \{k, o\}$, $C08 = \{l, p\}$, $C09 = \{a, b\}$, $C10 = \{c, d\}$, $C11 = \{e, f\}$, $C12 = \{g, h\}$, $C13 = \{i, j\}$, $C14 = \{k, l\}$, $C15 = \{m, n\}$ and $C16 = \{o, p\}$) and 3 configurations ($M01 = \{C01, C02, C03, C04, C05, C06, C07, C08\}$, $M02 = \{C09, C10, C11, C12, C13, C14, C15, C16\}$ and $M03 = \{C01, C02, C07, C08, C10, C12, C13, C15\}$).

6.1.2 Characteristics of the instances

We now list, using the same notation present in Chapter 4, the characteristics that all the instances used for our computational experiments share. In every instance we have:

- the same airspace, as in Figure 6.1;
- the same 3 different possible configurations ($|\mathcal{M}| = 3$), as in Figure 6.2;
- the same 8 airports ($|\mathcal{K}| = 8$);
- the same 16 elementary sectors ($|\mathcal{J}| = 16$);
- the same 16 collapsed sectors ($|\mathcal{H}| = 16$);
- a specific set of 256 flights ($|\mathcal{F}| = 256$);
- 36 time periods, representing a 6 hour day subdivided into ten-minute intervals ($|\mathcal{T}| = 36$);
- $D = 3$, representing the fact that 20 minutes is, for each flight $f \in \mathcal{F}$, an upper bound on the maximum delay in arrival allowed for flight f;
- $X = 18$, representing an upper bound of 16 elementary sectors in a flight's path (actually, in the instances considered in this thesis, no flight crosses more than 9 elementary sectors on its path but theoretically, based on how the instances have been generated, there could be up to 16 elementary sectors in a flight's path);
- $Y = 8$, representing the number of collapsed sectors present in each possible configuration;
- $D_k(t) = 8$ for every $k \in \mathcal{K}$ and $t \in \mathcal{T}$;
- $A_k(t) = 8$ for every $k \in \mathcal{K}$ and $t \in \mathcal{T}$;

- $c_f^g = 1$ for every flight $f \in \mathcal{F}$;
- $c_f^a = 3$ for every flight $f \in \mathcal{F}$.

The spatial trajectory of each flight f and the values of the related l_f^j, Δ_f, d_f, r_f, $\underline{T}_f^j$ and $\overline{T}_f^j$, for each elementary sector j, are randomly generated (in such a way as to satisfy, however, the properties and the relationships between them already described in Subsection 4.2.1). In particular, 20 possible random flight sets (and the related random trajectories and data) are generated, as we will detail in Subsection 6.1.3.

Regarding the capacities of the collapsed sectors, in our instances all the $S_h(t)$ have the same value, which, however, depends on the specific instance. In particular, we consider seven possible values of these capacities, namely 5, 10, 15, 20, 25, 30 and 35.

Our experiments will also consider different values of τ, that is the minimum number of consecutive time periods in which the (temporarily) chosen airspace configuration must remain active before it can be changed (as we have already seen in Subsection 4.2.1). In particular, we consider four different values, namely 1, 6, 12 and 36. We remark that $\tau = 1$ is equivalent to the case in which it is always possible to change configuration. This means that there are no constraints of the form (4.9). On the other hand, if $\tau = 36 = |\mathcal{T}|$ then, once a configuration has been chosen at time $t = 1$, it is no longer possible to change it.

Summarizing, our experiments will consider $20 \times 7 \times 4 = 560$ different instances of the ATFM problem with dynamic selection of the airspace configuration.

To conclude this subsection, we give the reader an idea of the size of the formulation proposed in [26] (and recalled in Subsection 4.2.3), and of the new formulation, in the case of the instances considered in this thesis.

Remembering (4.20) and (4.22), we find that these instances will surely have no more than 13,932 variables and 47,001 constraints using the formulation present in Subsection 4.2.3. Note that, since $|\mathcal{F}|XD = 13{,}824$, the parameters that significantly affect the upper bounds just calculated are $|\mathcal{F}|$, X and D (if, for example, one of these parameters doubles, then both the upper bounds approximately double). Actually, using this formulation, in the simulations

carried out for this thesis, there are slightly less than $3{,}000$ variables on average (with a maximum of $3{,}663$) and, on average, there are slightly less than $12{,}000$ constraints (with a maximum of $15{,}370$).

Recalling what we have already seen in Section 5.6, using the new formulation, all the considerations on the number of variables that we made for the formulation present in Subsection 4.2.3 still hold. Using the new formulation, in the simulations carried out for this thesis, there are about 320 constraints of the form (5.26) on average (with a maximum of 642) and there are always exactly $16 \times 36 = 576$ constraints of the form (5.11). Therefore, in the case of the computational experiments carried out for this thesis, the new formulation provides, on average, approximately 900 more constraints than the formulation present in Subsection 4.2.3 (this is an increase of less than 8%).

6.1.3 Generation of the spatial trajectories

Since in every instance used for our computational experiments there are 256 flights, it was not at all practical to generate the various trajectories manually. We therefore used the same procedure used in [26]. In particular, for every instance, we interpreted the airspace as a weighted undirected graph (see [25]), where each node represents an elementary sector or an airport. This graph was generated using MATLAB. In this graph, each node representing an airport has degree 1, as it is connected only to the elementary sector that contains it. Furthermore, if two elementary sectors are adjacent then there is an arc that connects the two nodes that represent them. Finally, a weight between 0 and 1 is randomly assigned to each arc. This last fact means that (with extremely high probability) each time we generate such a graph with MATLAB, we obtain a different graph (nodes and arcs are always the same, but not the weights). As in [26], 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. To create such a graph in MATLAB it is possible to use the function **graph()** whose parameters are given by: a list of source nodes; a list of corresponding target nodes; a list of weights associated with the arcs. The undirected graph associated to the airspace of the instances used for our computational experiments is represented in Figure 6.3.

After creating such a weighted undirected graph, we have generated the spatial trajectories of

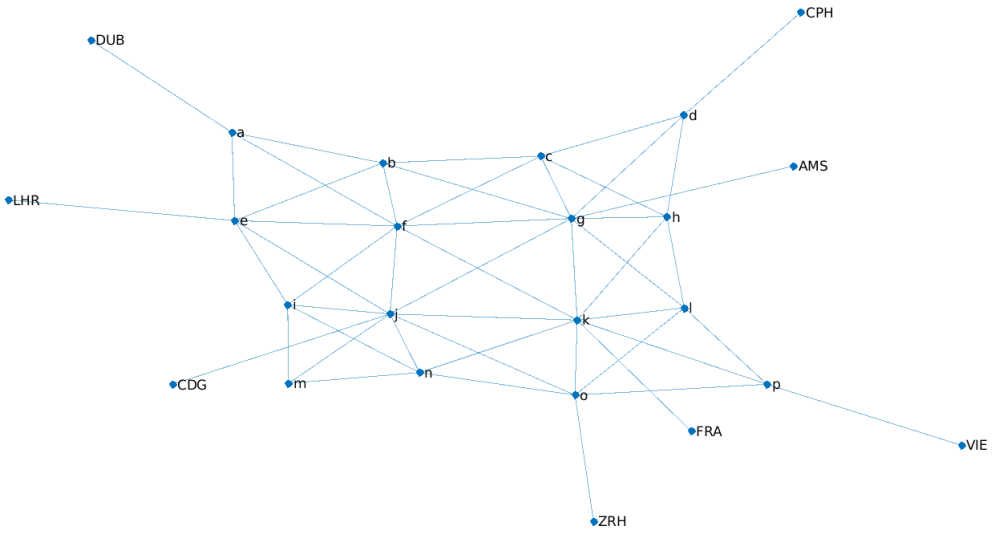

Figure 6.3: The undirected graph associated to the airspace of the instances used for our computational experiments [26].

256 flights following exactly the same procedure used in [26]. To do this, we have randomly chosen, for each of 128 flights, two distinct nodes representing the arrival and destination airports. Regarding the other 128 flights, after having identified 5 main airports (DUB, LHR, AMS, CDG and FRA), their arrival and destination airports were randomly chosen among the nodes representing these 5 main airports (always provided that, for each of these flights, the departure and arrival airports are different). For each flight, using the function (present in MATLAB) **shortestpath()** (which has as parameters the graph, the source node corresponding to the departure airport and the target node corresponding to the arrival airport), we have found the shortest path between the two nodes representing the departure and the destination airport. For each flight the path found in this way is its spatial trajectory. Note that these paths, once a source node and a target node are fixed, depend on the weights associated with the arcs of the graph (graphs with different weights lead, in general, to different paths). Our computational experiments have been conducted on 20 different graph settings of this type and, in particular, on 20 different flight sets with the related spatial trajectories and data. For simplicity, we will henceforth use the term "flight set" to refer to a set of flights together with the related spatial trajectories and data. Five of these flight sets are associated with the instances present in [26], whereas the other fifteen were generated by us. We will call these flight sets: Flights 1; . . . ; Flights 20. Flights 1, Flights 2, Flights 3, Flights 4 and Flights 5 are the five flight sets present in [26].

The procedure just described to generate spatial trajectories is implemented using the MATLAB file present in the appendix.

6.2 Implementation in AMPL

To perform our computational experiments, we have used the optimization software IBM ILOG CPLEX (version 11.1.1 and version 12.8.0.0) through its AMPL (version 20080701) interface. In this section, we describe the structure of the .mod, .dat and .run files used to implement the model in AMPL. These files contain, respectively, the definition of the model, the data and the procedures used to test the model. Examples of these files are provided in the appendix.

6.2.1 AMPL model file .mod

In the model file, the various sets, parameters, variables, objective functions and constraints are declared. In particular, the use of indexed parameters was extremely useful for subsequently implementing the separation of cover inequalities. Furthermore, again thanks to the use of indexes allowed by AMPL, we were able to declare only the variables really necessary for the formulation, avoiding the appearance of the "undefined variables" already described at the end of Subsection 4.2.2.

Indicating in brackets the notation used in the file, the following sets are declared: $\mathcal{M}$ (MAPS); $\mathcal{J}$ (ELEMENTARY); $\mathcal{H}$ (COLLAPSED); $\mathcal{F}$ (FLIGHTS); $\mathcal{K}$ (AIRPORTS); $\mathcal{P}_m$ (PARTITIONS[m]) for every $m \in \mathcal{M}$; $\mathcal{B}_h$ (BELONGING[h]) for every $h \in \mathcal{H}$; $\mathcal{T}$ (TIMES); P_f (PATHS[f]) for every $f \in \mathcal{F}$.

The declared parameters, fundamental for the implementation of the model, are the following: $D_k(t)$ (D[k,t]) for every $k \in \mathcal{K}$ and $t \in \mathcal{T}$; $A_k(t)$ (A[k,t]) for every $k \in \mathcal{K}$ and $t \in \mathcal{T}$; $S_h(t)$ (S[h,t]) for every $h \in \mathcal{H}$ and $t \in \mathcal{T}$; the various l_f^j (l[f,j]); Δ_f (delay[f]) for every $f \in \mathcal{F}$; d_f (d[f]) for every $f \in \mathcal{F}$; r_f (r[f]) for every $f \in \mathcal{F}$; a parameter representing the scheduled duration of flight f (duration[f]) for every $f \in \mathcal{F}$; c_f^g (cg[f]) for every $f \in \mathcal{F}$; c_f^a (ca[f]) for every $f \in \mathcal{F}$; the various $\underline{T}_f^j$ (tmin[f,j]); the various $\overline{T}_f^j$ (tmax[f,j]); $C_h(t)$ (C[h,t])

for every $h \in \mathcal{H}$ and $t \in \mathcal{T}$; τ (tau); for every constraint of the form (5.16) a parameter (number_cover_inequalities), necessary to implement the separation of cover inequalities, indicating the number of cover inequalities found (using the procedure described in Algorithm 5.1) treating that constraint as if it was a real knapsack-type constraint; some parameters (the various "cover_inequalities1" and the various "cover_inequalities2") necessary to store the cover inequalities found using Algorithm 5.1; some parameters (the various "wbar" and the various "ybar") necessary to store temporarily the various solutions found at **Step 1** of Algorithm 5.1; a parameter (flag) used to understand when to end the search for cover inequalities.

Other declared parameters, useful for collecting information from our simulations, are the following: a parameter indicating how many $w^j_{f,t}$ variables there are (counter_w); a parameter indicating how many $y_{m,t}$ variables there are (counter_y); a parameter indicating how many $w^j_{f,t}$ variables are integer in the solution (counter_w_int); a parameter indicating how many $w^j_{f,t}$ variables are fractional in the solution (counter_w_frac); a parameter indicating how many $y_{m,t}$ variables are integer in the solution (counter_y_int); a parameter indicating how many $y_{m,t}$ variables are fractional in the solution (counter_y_frac); a parameter (flag1) indicating the number of times **Step 3** of Algorithm 5.1 is executed; a parameter (flag2) indicating the number of problems of the form (3.5) solved using Algorithm 5.1; a parameter (flag3) indicating the total number of cover inequalities found using Algorithm 5.1.

The declared variables, fundamental for the implementation of the model, are the following: the various $w^j_{f,t}$ (w[f,j,t]); the various $y_{m,t}$ (y[m,t]); the variables (the various "z" and the various "x") used in the various problems of the form (3.5) solved using Algorithm 5.1. Other declared variables, useful for collecting information from our simulations, are the following: for every $f \in \mathcal{F}$ the total number g_f (g[f]) of time units that flight f is held on the ground through ground-holding policy; for every $f \in \mathcal{F}$ the total number a_f (a[f]) of time units that flight f is held in the air through airborne-holding and/or speed control policies.

The declared objective functions are the following: the objective function of our formulation (fun); the objective functions (sl1) of the various problems of the form (3.5) solved using Algorithm 5.1.

Finally, the declared constraints (besides the constraints, related to the binary nature of the variables, already included in the declaration of the variables) are the following: constraints

(4.1) (cons1); constraints (4.2) (cons2); constraints (5.16) (cons3); constraints (5.11) (cons4); constraints (4.4) (cons5); constraints (4.5) (cons6); constraints (4.7) (cons7); constraints (4.8) (cons8); constraints (4.9) (cons9); constraints (5.26) (conscover); the constraints (vsl1) of the various problems of the form (3.5) solved using Algorithm 5.1.

6.2.2 Data file .dat

In the data file, the elements are assigned to the various sets and the values are assigned to the various parameters. Here, indicating in brackets the notation used in the file, are assigned: configurations to set $\mathcal{M}$ (MAPS); flights to set $\mathcal{F}$ (FLIGHTS); airports to set $\mathcal{K}$ (AIRPORTS); elementary sectors to set $\mathcal{J}$ (ELEMENTARY); collapsed sectors to set $\mathcal{H}$ (COLLAPSED); collapsed sectors to set $\mathcal{P}_m$ (PARTITIONS[m]) for every $m \in \mathcal{M}$; elementary sectors to set $\mathcal{B}_h$ (BELONGING[h]) for every $h \in \mathcal{H}$; time periods to set $\mathcal{T}$ (TIMES); the spatial trajectory of flight f (PATHS[f]) for every $f \in \mathcal{F}$.

Moreover: the departure capacity of any airport at any time is considered constant (D=8); the arrival capacity of any airport at any time is considered constant (A=8); the cost of holding any flight on the ground, through ground-holding policy, for one unit of time is considered constant (cg=1); the cost of holding any flight in the air, through airborne-holding and/or speed control policies, for one unit of time is considered constant (ca=3); the values of the various l_f^j (l[f,j]), Δ_f (delay[f]), d_f (d[f]), r_f (r[f]), $\underline{T}_f^j$ (tmin[f,j]) and $\overline{T}_f^j$ (tmax[f,j]) are randomly generated (in such a way as to satisfy, however, the properties and the relationships between them already described in Subsection 4.2.1).

6.2.3 .run file

In our computational experiments all the capacities $S_h(t)$ have the same value. This means that, in each simulation carried out, $S_{h_1}(t_1) = S_{h_2}(t_2)$ for every $h_1, h_2 \in \mathcal{H}$ and $t_1, t_2 \in \mathcal{T}$. Given a flight set of our ATFM problem, running the .run file we solve it for different values of τ and of the various $S_h(t)$. Each time an instance is solved the following happens: the values of the constants $C_h(t)$ are computed; Algorithm 5.1 is applied (note that initially there are no constraints (5.26) which, therefore, will have to be found starting from the initial formulation, i.e. formulation $\mathfrak{F}$) in order to find constraints (5.26) and, therefore, obtain our new formulation together with an optimal solution (and, obviously, the optimal value) of its LPR; our instance is solved using the new formulation.

7

Computational results

The goal of this chapter is to highlight how our new formulation is significantly tighter than the one present in [26]. For this purpose, some computational results, obtained by working with the 560 instances described in Chapter 6, will be reported. In the first section of this seventh chapter, we will provide a general description of how our simulations were carried out and introduce some useful concepts to understand to what extent our contributions allow us to tighten the formulation proposed in [26]. In the second section, we will present the results obtained from our computational experiments and, finally, in the third section, we will comment on them.

7.1 Experiment settings

In the following, we will denote with $\mathfrak{F}_0$ the formulation present in Subsection 4.2.3, i.e. the formulation proposed in [26]. Let $\mathfrak{F}_1$ be the formulation obtained by replacing constraints (4.3) with constraints (5.16) in formulation $\mathfrak{F}_0$. Formulation $\mathfrak{F}_1$, compared to formulation $\mathfrak{F}_0$, simply has smaller values for the constants (remember what we have seen in Section 5.2). Let us denote with $\mathfrak{F}_2$ the formulation $\mathfrak{F}$ described at the beginning of Section 5.4. Formulation $\mathfrak{F}_2$ is obtained by adding constraints (5.11) to formulation $\mathfrak{F}_1$. Let $\mathfrak{F}_3$ be the formulation obtained by adding the cover inequalities, found using a procedure extremely similar to that described in Algorithm 5.1, to formulation $\mathfrak{F}_1$. Note that, in formulation $\mathfrak{F}_3$, there are no constraints of the form (5.11). Finally, let us denote with $\mathfrak{F}_4$ our new formulation, the one present in Section 5.5.

We recall that constraints (5.26) are dynamically added by Algorithm 5.1. By comparing these formulations, we will see to what extent our contributions, presented in Chapter 5, allow us to tighten formulation $\mathfrak{F}_0$.

It is also worth mentioning that the simulations related to formulations $\mathfrak{F}_0$, $\mathfrak{F}_1$ and $\mathfrak{F}_2$ were carried out using the optimization software IBM ILOG CPLEX (version 11.1.1) through its AMPL (version 20080701) interface, whereas the simulations related to formulations $\mathfrak{F}_3$ and $\mathfrak{F}_4$ were carried out using the optimization software IBM ILOG CPLEX (version 12.8.0.0) through its AMPL (version 20080701) interface.

Let us now introduce some terminology that will be used in the tables shown in the following section. We will use the term "Capacity" to refer to the value of all the various $S_h(t)$. Given a feasible instance of our ATFM problem (which is an ILP problem) whose optimal value (which is independent of the formulation used) $v_{optimal}$ is different from 0 and a formulation whose LPR has (regarding the instance in question) optimal value $v_{optimal}^{LPR}$, the integrality gap is defined as follows:

$$\text{integrality gap} := \frac{v_{optimal} - v_{optimal}^{LPR}}{v_{optimal}} \cdot 100.$$

The integrality gap (since $v_{optimal} \geq v_{optimal}^{LPR} \geq 0$ and $v_{optimal} > 0$) is a real number between 0 and 100 (extremes included) that indicates with a percentage the "relative distance" between $v_{optimal}$ and $v_{optimal}^{LPR}$ (the smaller this percentage, the "closer" $v_{optimal}$ and $v_{optimal}^{LPR}$ are). It is also worth remembering that, given an ILP problem to solve, the number of branch-and-bound nodes is the number of sub-problems (excluding the original problem) that are examined using the branch-and-bound method (or some variant thereof such as, for example, the branch-and-cut, recall Section 3.2 and Section 3.3) in order to solve the ILP problem (for example in Figure 3.3 there are 4 branch-and-bound nodes).

Finally, since in the next section we will compare some formulations with each other and this comparison will be carried out for four different values of τ, we describe here how the tables in the next section were constructed. It is worth mentioning that, depending on the values of τ and Capacity, the contents of the cells of these tables have been derived starting from the results obtained from the 20 different instances sharing these values of τ and Capacity (remember the description of the instances in Subsection 6.1.2).

Table 7.1, Table 7.6, Table 7.11 and Table 7.16 were obtained by solving 140 instances 5 times: one to solve our ATFM problem; four to solve the various LPRs (excluding the LPR of formulation $\mathfrak{F}_0$). In total, for each table, 700 simulations were carried out.

Table 7.2, Table 7.7, Table 7.12 and Table 7.17 were built on the basis of the same simulations used for Table 7.1, Table 7.6, Table 7.11 and Table 7.16 respectively.

Table 7.3, Table 7.8, Table 7.13 and Table 7.18 were built on the basis of 560 of the 700 simulations used for Table 7.1, Table 7.6, Table 7.11 and Table 7.16 respectively. For each table, the 140 simulations concerning our ILP ATFM problem were not used, but only the 560 simulations related to the various LPRs.

Table 7.4, Table 7.9, Table 7.14 and Table 7.19 were built on the basis of 280 of the 560 simulations used for Table 7.3, Table 7.8, Table 7.13 and Table 7.18 respectively. For each table, only the 280 simulations related to formulations $\mathfrak{F}_3$ and $\mathfrak{F}_4$ were used.

Table 7.5, Table 7.10, Table 7.15 and Table 7.20 were obtained by solving 140 instances 4 times, one for each formulation (excluding formulation $\mathfrak{F}_3$). In total, for each table, 560 simulations were carried out.

7.2 Comparison of formulations

In this section, we will mainly compare formulations $\mathfrak{F}_1$, $\mathfrak{F}_2$, $\mathfrak{F}_3$ and $\mathfrak{F}_4$. This will allow us to understand to what extent our contributions (presented in Chapter 5) make it possible to tighten, in the cases of the instances considered in this thesis, formulation $\mathfrak{F}_0$. As we have already seen in the previous section, this comparison between formulations will be carried out for four different values of τ.

7.2.1 The case $\tau = 1$

Table 7.1 reports, depending on the value of Capacity, the percentage of infeasible instances of our ILP ATFM problem in a column. Other four columns indicate, depending on the value of Capacity and on the formulation used, the percentage of infeasible instances whose LPR is feasible. We recall that, according to the instance description in Subsection 6.1.2, each row of

Table 7.1 consolidates the results over 20 instances (sharing the same τ and the same Capacity, but having different flight sets). As we have already seen at the beginning of Section 5.2, using formulation $\mathfrak{F}_0$, in every computational experiment we carried out, regardless of whether the specific instance of the ILP problem was feasible or not, the optimal value of its LPR was always equal to 0. For this reason formulation $\mathfrak{F}_0$ does not appear in Table 7.1. In Table 7.1, when Capacity is equal to 15, we have 19 infeasible instances (95% of 20) of our ATFM problem. Solving all these 19 instances using the LPR of formulation $\mathfrak{F}_1$ we always (100% of 19) find an optimal solution (the infeasibility of the instances of our ATFM problem is not detected at all). Solving all these 19 instances using the LPR of formulation $\mathfrak{F}_2$ we find an optimal solution in 5 (about 26.32% of 19) cases (the infeasibility of the instances of our ATFM problem is detected in 14 cases out of 19). Solving all these 19 instances using the LPR of formulation $\mathfrak{F}_3$ we find an optimal solution in 2 (about 10.53% of 19) cases (the infeasibility of the instances of our ATFM problem is detected in 17 cases out of 19). Solving all these 19 instances using the LPR of formulation $\mathfrak{F}_4$ we find an optimal solution in 1 (about 5.26% of 19) case (the infeasibility of the instances of our ATFM problem is detected in 18 cases out of 19). Table 7.1 therefore shows that each of our contributions allow us to tighten the formulation proposed in [26].

		% of infeasible instances whose LPR is feasible			
Capacity	% of infeasible instances	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_3$	$\mathfrak{F}_4$
5	100	15	0	0	0
10	100	90	0	0	0
15	95	100	26.32	10.53	5.26
20	30	100	16.67	16.67	16.67
25	10	100	50	50	0
30	5	100	100	0	0
35	5	100	100	0	0

Table 7.1: Infeasibility detection (in the case $\tau = 1$).

This fact is also confirmed in Table 7.2, which reports in a column the percentage of feasible instances of our ILP ATFM problem whose optimal value is different from 0. Other four columns indicate, depending on the value of Capacity and on the formulation used, the average integrality gap. In Table 7.2, when Capacity is equal to 25, we have 8 (40% of 20) feasible instances whose optimal value is different from 0. For these 8 instances the average integrality gap (rounded to two decimal places) is: 100% using formulation $\mathfrak{F}_1$; 55.31% using formulation $\mathfrak{F}_2$; 68.56% using formulation $\mathfrak{F}_3$; 35.60% using formulation $\mathfrak{F}_4$. In Table 7.2, the writing

"N/D" appears every time the average integrality gap is not defined.

		Average integrality gap (%)			
Capacity	% of feasible instances whose optimal value is $\neq 0$	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_3$	$\mathfrak{F}_4$
5	0	N/D	N/D	N/D	N/D
10	0	N/D	N/D	N/D	N/D
15	5	100	100	50	50
20	60	99.52	60.42	57.90	34.66
25	40	100	55.31	68.56	35.60
30	15	100	33.33	53.91	21.43
35	0	N/D	N/D	N/D	N/D

Table 7.2: Average integrality gap (in the case $\tau = 1$).

In Table 7.3 four columns indicate, depending on the value of Capacity and on the formulation used, the average percentage of fractional variables of the type $w^j_{f,t}$. Other four columns indicate, depending on the value of Capacity and on the formulation used, the average percentage of fractional variables of the type $y_{m,t}$. In Table 7.3, for example, when Capacity is equal to 20 and the formulation used is formulation $\mathfrak{F}_4$ we have that on average: about 5.27% of the various $w^j_{f,t}$ are fractional (on average there are slightly less than $2{,}900$ $w^j_{f,t}$ variables and of these, on average, about 150 are fractional); 10% of the various $y_{m,t}$ are fractional (there are always 108 $y_{m,t}$ variables and of these, on average, about 11 are fractional). The number of instances on which these two average percentages are computed, not present in Table 7.3 (to keep it more readable), is the number of instances whose LPR (using formulation $\mathfrak{F}_4$) is feasible, that is (in this case) 15 (this number can be deduced from Table 7.1). Note that formulation $\mathfrak{F}_4$ is, on average, the one with the smallest percentage of fractional $y_{m,t}$ variables. However, the use of cover inequalities increases the number of fractional $w^j_{f,t}$ variables (with formulation $\mathfrak{F}_4$ having fewer fractional $w^j_{f,t}$ variables than formulation $\mathfrak{F}_3$). In Table 7.3, the writing "N/D" appears every time the value present within the corresponding cell of the table is not defined.

Regarding formulation $\mathfrak{F}_4$ and (in particular) formulation $\mathfrak{F}_3$, the time needed to separate cover inequalities is relevant, as shown in Table 7.4. In this table two columns indicate, depending on the value of Capacity and on the formulation used, the average time taken to separate cover inequalities. Other two columns indicate, depending on the value of Capacity and on the formulation used, the average number of cover inequalities found. Also note that (except when Capacity is 30 or 35) in formulation $\mathfrak{F}_3$ there are, on average, more cover inequalities than in

	Average % of fractional $w^j_{f,t}$s				Average % of fractional $y_{m,t}$s			
Capacity	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_3$	$\mathfrak{F}_4$	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_3$	$\mathfrak{F}_4$
5	11.69	N/D	N/D	N/D	83.02	N/D	N/D	N/D
10	4.27	N/D	N/D	N/D	69.08	N/D	N/D	N/D
15	0.64	0.59	12.44	9.81	46.48	34.26	23.77	21.30
20	0.09	0.06	7.64	5.27	25.37	19.75	13.70	10
25	0	0	3.14	0.84	9.95	9.65	5.56	2.11
30	0	0	0.23	0.07	4.40	4.49	0.49	0.15
35	0	0	0	0	1.62	1.57	0	0

Table 7.3: Average percentage of fractional variables (in the case $\tau = 1$).

formulation $\mathfrak{F}_4$. Therefore there are no advantages in using formulation $\mathfrak{F}_3$ instead of formulation $\mathfrak{F}_4$. Note that all these averages are calculated over 20 instances, which, however, depend on the specific row of the table.

	Average time taken to find cover inequalities (in seconds)		Average number of cover inequalities found	
Capacity	$\mathfrak{F}_3$	$\mathfrak{F}_4$	$\mathfrak{F}_3$	$\mathfrak{F}_4$
5	1.106	0.014	57.25	0
10	13.518	0.015	388.8	0
15	59.453	8.867	473.5	81.1
20	240.450	54.785	343.7	127.85
25	75.577	19.922	96.3	55.3
30	18.294	3.482	23.75	23.8
35	3.276	1.504	15.15	15.35

Table 7.4: Average time (in seconds) taken to find cover inequalities and average number of cover inequalities found (in the case $\tau = 1$).

Finally, assuming that we have already found formulation $\mathfrak{F}_4$, the time needed, on average, to solve one of the instances of our ATFM problem (which is an ILP problem) is not excessively long, although longer than those required using other formulations (see Table 7.5). In Table 7.5, for completeness, we also report how many branch-and-bound nodes have been found, on average, in the various cases.

	Average solving time (in seconds)				Average number of branch-and-bound nodes			
Capacity	$\mathfrak{F}_0$	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_4$	$\mathfrak{F}_0$	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_4$
5	0.010	0.006	0.011	0.020	0	0	0	0
10	0.014	0.008	0.015	0.020	0	0	0	0
15	0.017	0.011	0.018	0.026	0.05	0.05	0	0
20	0.413	0.395	0.208	1.829	287	287	61.6	215.6
25	0.053	0.045	0.031	0.232	8.45	8.45	3.6	0
30	0.022	0.016	0.023	0.033	0	0	0	0
35	0.020	0.014	0.022	0.030	0	0	0	0

Table 7.5: Average solving time (in seconds) taken to solve our ATFM problem and average number of branch-and-bound nodes found (in the case $\tau = 1$).

7.2.2 The case $\tau = 6$

In Table 7.6, whose description is similar to that of Table 7.1, when Capacity is equal to 15, we have 19 infeasible instances (95% of 20) of our ATFM problem. Solving all these 19 instances using the LPR of formulation $\mathfrak{F}_1$ we always (100% of 19) find an optimal solution (the infeasibility of the instances of our ATFM problem is not detected at all). Solving all these 19 instances using the LPR of formulation $\mathfrak{F}_2$ we find an optimal solution in 5 (about 26.32% of 19) cases (the infeasibility of the instances of our ATFM problem is detected in 14 cases out of 19). Solving all these 19 instances using the LPR of formulation $\mathfrak{F}_3$ we find an optimal solution in 2 (about 10.53% of 19) cases (the infeasibility of the instances of our ATFM problem is detected in 17 cases out of 19). Solving all these 19 instances using the LPR of formulation $\mathfrak{F}_4$ we find an optimal solution in 1 (about 5.26% of 19) case (the infeasibility of the instances of our ATFM problem is detected in 18 cases out of 19). Table 7.6 therefore shows that each of our contributions allow us to tighten the formulation proposed in [26].

This fact is also confirmed in Table 7.7, whose description is similar to that of Table 7.2. Here, for example, when Capacity is equal to 25 we have 8 (40% of 20) feasible instances whose optimal value is different from 0. For these 8 instances the average integrality gap (rounded to two decimal places) is: 100% using formulation $\mathfrak{F}_1$; 55.31% using formulation $\mathfrak{F}_2$; 61.41% using formulation $\mathfrak{F}_3$; 30.47% using formulation $\mathfrak{F}_4$. In Table 7.7, the writing "N/D" appears every time the average integrality gap is not defined.
In Table 7.8, whose description is similar to that of Table 7.3, when Capacity is equal to 20 and the formulation used is formulation $\mathfrak{F}_4$ we have that on average: about 5.42% of the various

		% of infeasible instances whose LPR is feasible			
Capacity	% of infeasible instances	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_3$	$\mathfrak{F}_4$
5	100	15	0	0	0
10	100	90	0	0	0
15	95	100	26.32	10.53	5.26
20	30	100	16.67	16.67	16.67
25	10	100	50	50	0
30	5	100	100	0	0
35	5	100	100	0	0

Table 7.6: Infeasibility detection (in the case $\tau = 6$).

		Average integrality gap (%)			
Capacity	% of feasible instances whose optimal value is $\neq 0$	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_3$	$\mathfrak{F}_4$
5	0	N/D	N/D	N/D	N/D
10	0	N/D	N/D	N/D	N/D
15	5	100	100	50	50
20	65	99.61	64.49	53.86	34.35
25	40	100	55.31	61.41	30.47
30	15	100	33.33	31.09	21.43
35	0	N/D	N/D	N/D	N/D

Table 7.7: Average integrality gap (in the case $\tau = 6$).

$w^j_{f,t}$ are fractional (on average there are slightly less than 2,900 $w^j_{f,t}$ variables and of these, on average, about 160 are fractional); about 29.63% of the various $y_{m,t}$ are fractional (there are always 108 $y_{m,t}$ variables and of these, on average, about 32 are fractional). The number of instances on which these two average percentages are computed, not present in Table 7.8 (to keep it more readable), is the number of instances whose LPR (using formulation $\mathfrak{F}_4$) is feasible, that is (in this case) 15 (this number can be deduced from Table 7.6). Note that formulation $\mathfrak{F}_3$ and formulation $\mathfrak{F}_4$ are, on average, the ones with the smallest percentage of fractional $y_{m,t}$ variables. However, the use of cover inequalities increases the number of fractional $w^j_{f,t}$ variables (with formulation $\mathfrak{F}_4$ having fewer fractional $w^j_{f,t}$ variables than formulation $\mathfrak{F}_3$). In Table 7.8, the writing "N/D" appears every time the value present within the corresponding cell of the table is not defined.

	Average % of fractional $w^j_{f,t}$s				Average % of fractional $y_{m,t}$s			
Capacity	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_3$	$\mathfrak{F}_4$	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_3$	$\mathfrak{F}_4$
5	12.28	N/D	N/D	N/D	98.15	N/D	N/D	N/D
10	4.41	N/D	N/D	N/D	93.93	N/D	N/D	N/D
15	0.66	0.56	12.46	10.41	74.68	75.46	52.78	65.28
20	0.08	0.19	7.98	5.42	55.28	57.84	36.91	29.63
25	0	0	3.59	1.26	47.73	43.42	23.10	20.68
30	0	0	0.83	0.07	40.14	35.60	11.40	12.87
35	0	0	0	0	40.32	33.52	7.60	12.04

Table 7.8: Average percentage of fractional variables (in the case $\tau = 6$).

Regarding formulation $\mathfrak{F}_4$ and (in particular) formulation $\mathfrak{F}_3$, the time needed to separate cover inequalities is relevant, as shown in Table 7.9. The description of this table is similar to that of Table 7.4. Also note that (except when Capacity is 35) in formulation $\mathfrak{F}_3$ there are, on average, more cover inequalities than in formulation $\mathfrak{F}_4$. Therefore there are no advantages in using formulation $\mathfrak{F}_3$ instead of formulation $\mathfrak{F}_4$. Note that all these averages are calculated over 20 instances, which, however, depend on the specific row of the table.

	Average time taken to find cover inequalities (in seconds)		Average number of cover inequalities found	
Capacity	$\mathfrak{F}_3$	$\mathfrak{F}_4$	$\mathfrak{F}_3$	$\mathfrak{F}_4$
5	0.880	0.190	57.7	0
10	11.722	0.123	381.05	0
15	48.815	6.566	467.5	83.4
20	127.620	35.430	345.9	146.15
25	71.371	18.451	113.05	53.25
30	19.042	4.807	35.15	23.4
35	6.084	2.864	14.7	15.95

Table 7.9: Average time (in seconds) taken to find cover inequalities and average number of cover inequalities found (in the case $\tau = 6$).

Finally, assuming that we have already found formulation $\mathfrak{F}_4$, the time needed, on average, to solve one of the instances of our ATFM problem (which is an ILP problem) is not excessively long, although longer than those required using other formulations (see Table 7.10). In Table 7.10, for completeness, we also report how many branch-and-bound nodes have been found, on average, in the various cases.

	Average solving time (in seconds)				Average number of branch-and-bound nodes			
Capacity	$\mathfrak{F}_0$	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_4$	$\mathfrak{F}_0$	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_4$
5	0.010	0.006	0.011	0.020	0	0	0	0
10	0.014	0.010	0.015	0.021	0	0	0	0
15	0.018	0.013	0.018	0.028	0	0	0	0
20	0.225	0.213	0.156	1.500	30.6	30.6	10.95	150.05
25	0.079	0.070	0.045	0.364	2.7	2.7	0.9	0
30	0.025	0.018	0.025	0.038	0	0	0	0
35	0.022	0.016	0.025	0.035	0	0	0	0

Table 7.10: Average solving time (in seconds) taken to solve our ATFM problem and average number of branch-and-bound nodes found (in the case $\tau = 6$).

7.2.3 The case $\tau = 12$

In Table 7.11, whose description is similar to that of Table 7.1, when Capacity is equal to 15, we have 19 infeasible instances (95% of 20) of our ATFM problem. Solving all these 19 instances using the LPR of formulation $\mathfrak{F}_1$ we always (100% of 19) find an optimal solution (the infeasibility of the instances of our ATFM problem is not detected at all). Solving all these 19 instances using the LPR of formulation $\mathfrak{F}_2$ we find an optimal solution in 5 (about 26.32% of 19) cases (the infeasibility of the instances of our ATFM problem is detected in 14 cases out of 19). Solving all these 19 instances using the LPR of formulation $\mathfrak{F}_3$ we find an optimal solution in 2 (about 10.53% of 19) cases (the infeasibility of the instances of our ATFM problem is detected in 17 cases out of 19). Solving all these 19 instances using the LPR of formulation $\mathfrak{F}_4$ we find an optimal solution in 1 (about 5.26% of 19) case (the infeasibility of the instances of our ATFM problem is detected in 18 cases out of 19). Table 7.11 therefore shows that each of our contributions allow us to tighten the formulation proposed in [26].

This fact is also confirmed in Table 7.12, whose description is similar to that of Table 7.2. Here, for example, when Capacity is equal to 25 we have 8 (40% of 20) feasible instances whose optimal value is different from 0. For these 8 instances the average integrality gap (rounded to two decimal places) is: 100% using formulation $\mathfrak{F}_1$; 56.11% using formulation $\mathfrak{F}_2$; 61.13% using formulation $\mathfrak{F}_3$; 30.87% using formulation $\mathfrak{F}_4$. In Table 7.12, the writing "N/D" appears every time the average integrality gap is not defined.

In Table 7.13, whose description is similar to that of Table 7.3, when Capacity is equal to 20 and

		% of infeasible instances whose LPR is feasible			
Capacity	% of infeasible instances	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_3$	$\mathfrak{F}_4$
5	100	15	0	0	0
10	100	85	0	0	0
15	95	100	26.32	10.53	5.26
20	30	100	16.67	16.67	16.67
25	10	100	50	50	0
30	5	100	100	0	0
35	5	100	100	0	0

Table 7.11: Infeasibility detection (in the case $\tau = 12$).

		Average integrality gap (%)			
Capacity	% of feasible instances whose optimal value is $\neq 0$	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_3$	$\mathfrak{F}_4$
5	0	N/D	N/D	N/D	N/D
10	0	N/D	N/D	N/D	N/D
15	5	100	100	63.33	63.33
20	65	99.61	64.93	48.02	28.82
25	40	100	56.11	61.13	30.87
30	15	100	33.33	31.09	21.43
35	0	N/D	N/D	N/D	N/D

Table 7.12: Average integrality gap (in the case $\tau = 12$).

the formulation used is formulation $\mathfrak{F}_4$ we have that on average: about 5.50% of the various $w^j_{f,t}$ are fractional (on average there are slightly less than $2{,}900$ $w^j_{f,t}$ variables and of these, on average, about 160 are fractional); about 41.73% of the various $y_{m,t}$ are fractional (there are always 108 $y_{m,t}$ variables and of these, on average, about 45 are fractional). The number of instances on which these two average percentages are computed, not present in Table 7.13 (to keep it more readable), is the number of instances whose LPR (using formulation $\mathfrak{F}_4$) is feasible, that is (in this case) 15 (this number can be deduced from Table 7.11). Note that formulation $\mathfrak{F}_4$ is, on average, the one with the smallest percentage of fractional $y_{m,t}$ variables. However, the use of cover inequalities increases the number of fractional $w^j_{f,t}$ variables (with formulation $\mathfrak{F}_4$ having fewer fractional $w^j_{f,t}$ variables than formulation $\mathfrak{F}_3$). In Table 7.13, the writing "N/D" appears every time the value present within the corresponding cell of the table is not defined.

	Average % of fractional $w^j_{f,t}$s				Average % of fractional $y_{m,t}$s			
Capacity	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_3$	$\mathfrak{F}_4$	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_3$	$\mathfrak{F}_4$
5	14.87	N/D	N/D	N/D	99.69	N/D	N/D	N/D
10	4.39	N/D	N/D	N/D	99.89	N/D	N/D	N/D
15	0.94	0.88	12.90	11.58	96.67	95.37	56.79	52.31
20	0.12	0.19	8.15	5.50	79.58	72.28	48.58	41.73
25	0	0.21	3.89	1.48	58.94	50.10	37.72	28.40
30	0	0	0.82	0.07	55.88	51.02	24.90	15.94
35	0	0	0	0	54.77	47.50	19.64	16.08

Table 7.13: Average percentage of fractional variables (in the case $\tau = 12$).

Regarding formulation $\mathfrak{F}_4$ and (in particular) formulation $\mathfrak{F}_3$, the time needed to separate cover inequalities is relevant, as shown in Table 7.14. The description of this table is similar to that of Table 7.4. Also note that in formulation $\mathfrak{F}_3$ there are, on average, more cover inequalities than in formulation $\mathfrak{F}_4$. Therefore there are no advantages in using formulation $\mathfrak{F}_3$ instead of formulation $\mathfrak{F}_4$. Note that all these averages are calculated over 20 instances, which, however, depend on the specific row of the table.

	Average time taken to find cover inequalities (in seconds)		Average number of cover inequalities found	
Capacity	$\mathfrak{F}_3$	$\mathfrak{F}_4$	$\mathfrak{F}_3$	$\mathfrak{F}_4$
5	1.125	0.177	57.5	0
10	11.709	0.126	350.4	0
15	54.421	9.285	475.55	89.55
20	112.492	36.339	329.85	148.25
25	68.280	20.069	120.65	56.3
30	20.586	4.596	37.5	22.55
35	8.464	3.224	21.55	17.95

Table 7.14: Average time (in seconds) taken to find cover inequalities and average number of cover inequalities found (in the case $\tau = 12$).

Finally, assuming that we have already found formulation $\mathfrak{F}_4$, the time needed, on average, to solve one of the instances of our ATFM problem (which is an ILP problem) is not excessively long, although longer than those required using other formulations (see Table 7.15). In Table 7.15, for completeness, we also report how many branch-and-bound nodes have been found, on average, in the various cases.

	Average solving time (in seconds)				Average number of branch-and-bound nodes			
Capacity	$\mathfrak{F}_0$	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_4$	$\mathfrak{F}_0$	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_4$
5	0.015	0.009	0.016	0.021	0	0	0	0
10	0.015	0.010	0.016	0.022	0	0	0	0
15	0.021	0.015	0.022	0.089	0	0	0	0
20	0.120	0.110	0.108	1.489	9.1	9.1	6	215.3
25	0.123	0.112	0.065	0.539	3.55	3.55	1	0
30	0.027	0.020	0.028	0.044	0	0	0	0
35	0.024	0.018	0.025	0.041	0	0	0	0

Table 7.15: Average solving time (in seconds) taken to solve our ATFM problem and average number of branch-and-bound nodes found (in the case $\tau = 12$).

7.2.4 The case $\tau = 36$

In Table 7.16, whose description is similar to that of Table 7.1, when Capacity is equal to 15, we have 19 infeasible instances (95% of 20) of our ATFM problem. Solving all these 19 instances using the LPR of formulation $\mathfrak{F}_1$ we always (100% of 19) find an optimal solution (the infeasibility of the instances of our ATFM problem is not detected at all). Solving all these 19 instances using the LPR of formulation $\mathfrak{F}_2$ we find an optimal solution in 5 (about 26.32% of 19) cases (the infeasibility of the instances of our ATFM problem is detected in 14 cases out of 19). Solving all these 19 instances using the LPR of formulation $\mathfrak{F}_3$ we find an optimal solution in 0 (0% of 19) cases (the infeasibility of the instances of our ATFM problem is always detected). Solving all these 19 instances using the LPR of formulation $\mathfrak{F}_4$ we find an optimal solution in 0 (0% of 19) cases (the infeasibility of the instances of our ATFM problem is always detected). Table 7.16 therefore shows that each of our contributions allow us to tighten the formulation proposed in [26].

This fact is also confirmed in Table 7.17, whose description is similar to that of Table 7.2. Here, for example, when Capacity is equal to 25, we have 8 (40% of 20) feasible instances whose optimal value is different from 0. For these 8 instances the average integrality gap (rounded to two decimal places) is: 100% using formulation $\mathfrak{F}_1$; 57.78% using formulation $\mathfrak{F}_2$; 48.82% using formulation $\mathfrak{F}_3$; 24.70% using formulation $\mathfrak{F}_4$. In Table 7.17, the writing "N/D" appears every time the average integrality gap is not defined.

In Table 7.18, whose description is similar to that of Table 7.3, when Capacity is equal to 20 and

		% of infeasible instances whose LPR is feasible			
Capacity	% of infeasible instances	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_3$	$\mathfrak{F}_4$
5	100	5	0	0	0
10	100	85	0	0	0
15	95	100	26.32	0	0
20	45	100	44.44	22.22	22.22
25	10	100	50	0	0
30	5	100	100	0	0
35	5	100	100	0	0

Table 7.16: Infeasibility detection (in the case $\tau = 36$).

		Average integrality gap (%)			
Capacity	% of feasible instances whose optimal value is $\neq 0$	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_3$	$\mathfrak{F}_4$
5	0	N/D	N/D	N/D	N/D
10	0	N/D	N/D	N/D	N/D
15	5	100	100	63.33	63.33
20	50	99.29	65.83	47.86	33.35
25	40	100	57.78	48.82	24.70
30	20	100	62.50	33.37	28.57
35	0	N/D	N/D	N/D	N/D

Table 7.17: Average integrality gap (in the case $\tau = 36$).

the formulation used is formulation $\mathfrak{F}_4$ we have that on average: about 5.40% of the various $w^j_{f,t}$ are fractional (on average there are slightly less than $2{,}900$ $w^j_{f,t}$ variables and of these, on average, about 160 are fractional); about 51.28% of the various $y_{m,t}$ are fractional (there are always 108 $y_{m,t}$ variables and of these, on average, about 55 are fractional). The number of instances on which these two average percentages are computed, not present in Table 7.18 (to keep it more readable), is the number of instances whose LPR (using formulation $\mathfrak{F}_4$) is feasible, that is (in this case) 13 (this number can be deduced from Table 7.16). Note that formulation $\mathfrak{F}_4$ is, on average, the one with the smallest percentage of fractional $y_{m,t}$ variables. However, the use of cover inequalities increases the number of fractional $w^j_{f,t}$ variables (with formulation $\mathfrak{F}_4$ having fewer fractional $w^j_{f,t}$ variables than formulation $\mathfrak{F}_3$). In Table 7.18, the writing "N/D" appears every time the value present within the corresponding cell of the table is not defined.

	Average % of fractional $w^j_{f,t}$s				Average % of fractional $y_{m,t}$s			
Capacity	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_3$	$\mathfrak{F}_4$	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_3$	$\mathfrak{F}_4$
5	15.26	N/D	N/D	N/D	100	N/D	N/D	N/D
10	4.67	N/D	N/D	N/D	100	N/D	N/D	N/D
15	1.17	1.01	1.48	1.48	98.33	94.44	66.67	66.67
20	0.16	0	7.43	5.40	90	75.56	53.85	51.28
25	0.01	0.01	2.40	1.74	61.67	45.61	25.93	16.67
30	0	0	1.26	0.09	38.33	28.33	19.30	10.53
35	0	0	0	0	15	11.67	0	0

Table 7.18: Average percentage of fractional variables (in the case $\tau = 36$).

Regarding formulation $\mathfrak{F}_4$ and (in particular) formulation $\mathfrak{F}_3$, the time needed to separate cover inequalities is relevant, as shown in Table 7.19. The description of this table is similar to that of Table 7.4. Also note that in formulation $\mathfrak{F}_3$ there are, on average, more cover inequalities than in formulation $\mathfrak{F}_4$. Therefore there are no advantages in using formulation $\mathfrak{F}_3$ instead of formulation $\mathfrak{F}_4$. Note that all these averages are calculated over 20 instances, which, however, depend on the specific row of the table.

	Average time taken to find cover inequalities (in seconds)		Average number of cover inequalities found	
Capacity	$\mathfrak{F}_3$	$\mathfrak{F}_4$	$\mathfrak{F}_3$	$\mathfrak{F}_4$
5	0.644	0.174	19.75	0
10	11.436	0.116	350.55	0
15	33.465	5.628	413.95	81.05
20	90.666	33.405	301.05	128.55
25	56.310	19.891	106.95	64.75
30	19.919	3.781	36.65	27.95
35	4.195	2.121	21.05	21.05

Table 7.19: Average time (in seconds) taken to find cover inequalities and average number of cover inequalities found (in the case $\tau = 36$).

Finally, assuming that we have already found formulation $\mathfrak{F}_4$, the time needed, on average, to solve one of the instances of our ATFM problem (which is an ILP problem) is not excessively long, although longer than those required using other formulations (see Table 7.20). In Table 7.20, for completeness, we also report how many branch-and-bound nodes have been found, on average, in the various cases.

	Average solving time (in seconds)				Average number of branch-and-bound nodes			
Capacity	$\mathfrak{F}_0$	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_4$	$\mathfrak{F}_0$	$\mathfrak{F}_1$	$\mathfrak{F}_2$	$\mathfrak{F}_4$
5	0.017	0.010	0.017	0.022	0	0	0	0
10	0.016	0.010	0.017	0.023	0	0	0	0
15	0.020	0.014	0.021	0.118	0	0	0	0
20	0.064	0.055	0.060	0.228	0.35	0.35	0.55	0
25	0.105	0.094	0.088	0.216	0.45	0.45	0.35	0
30	0.032	0.025	0.032	0.050	0	0	0	0
35	0.029	0.022	0.030	0.042	0	0	0	0

Table 7.20: Average solving time (in seconds) taken to solve our ATFM problem and average number of branch-and-bound nodes found (in the case $\tau = 36$).

7.3 Comments on the computational results

Table 7.1, Table 7.6, Table 7.11 and Table 7.16 show how, using the LPR of formulation $\mathfrak{F}_4$, the infeasibility of the instances of our ATFM problem (which is an ILP problem) is often detected. In the worst case, the one in which $\tau = 36$ and Capacity is equal to 20, the infeasibility is detected in approximately 77.78% (7 out of 9) of the instances. This is a remarkable improvement compared to the LPR of formulation $\mathfrak{F}_0$. Using the latter, the infeasibility of the instances of our ATFM problem is never detected since, as we have already seen in Section 5.2, we always find 0 as the optimal value. Furthermore, these tables allow us to observe how each of our contributions is useful to tighten formulation $\mathfrak{F}_0$.

Also Table 7.2, Table 7.7, Table 7.12 and Table 7.17 allow us to observe how each of our contributions is useful to tighten formulation $\mathfrak{F}_0$. Except when Capacity is equal to 15 (case in which, for each of the four values of τ, the integrality gap is calculated on a single instance), the maximum average integrality gap, using formulation $\mathfrak{F}_4$, is approximately 35.60% (this happens in the case in which $\tau = 1$ and Capacity is equal to 25). This is a remarkable improvement compared to formulation $\mathfrak{F}_0$ (using the latter the integrality gap is always 100%).

It should be noted that formulation $\mathfrak{F}_4$ unfortunately also presents some flaws. First of all, also using the LPR of formulation $\mathfrak{F}_4$ (to solve the instances of our ATFM problem) we find a rather large number of fractional $y_{m,t}$ variables. Moreover, due to the use of cover inequalities, using the LPR of formulation $\mathfrak{F}_4$ (to solve the instances of our ATFM problem) we also find

a non-negligible number of fractional $w^j_{f,t}$ variables (see Table 7.3, Table 7.8, Table 7.13 and Table 7.18). Finally, the separation of cover inequalities, necessary to obtain formulation $\mathfrak{F}_4$, requires quite a long time to be performed (see Table 7.4, Table 7.9, Table 7.14 and Table 7.19). Furthermore, even once it is obtained, formulation $\mathfrak{F}_4$ requires a longer time, compared to other formulations, to solve the instances of our ILP ATFM problem (see Table 7.5, Table 7.10, Table 7.15 and Table 7.20).

8

Conclusions

In this thesis, starting from the ILP model presented in [26] for the ATFM problem with dynamic selection of the airspace configuration, we have proposed some contributions in order to obtain a tighter ILP formulation. This means that our new formulation obviously has, for each instance, the same feasible region as that of the formulation present in [26]. However, the LPR of our formulation is generally "closer" to the convex hull of the solutions of the ILP problem than the LPR of the formulation presented in [26]. With our new formulation, in fact, given an instance of the ATFM problem, it is often (in approximately 90% of cases) possible to identify its possible infeasibility by solving the LPR of our ILP problem instead of the ILP problem itself. Our new formulation also allows us to have a much smaller average integrality gap (less, in most cases, than 36%) compared to the formulation presented in [26] (in which the integrality gap, when defined, was always equal to 100% in all our computational experiments).

In order to obtain the improvements just described, starting from the formulation presented in [26], we proposed three different contributions. The first one consists in replacing some constants of the formulation with other smaller constants, the smallest possible. We then added to the formulation a new class of valid inequalities that allowed us to further tighten the formulation. Finally, to get closer to the convex hull of the solutions of the ILP problem, we also added some inequalities derived as cover inequalities from some of the ILP model constraints.

However, our new formulation also has some flaws. First of all, also using the LPR of our new formulation, we find a non-negligible number of fractional variables. Furthermore, due to the separation of cover inequalities, our new formulation requires quite a long time to be solved. Moreover, the model obtained after solving the linear relaxation of our new formulation further requires (on average) a longer time (compared to the formulation present in [26]) to provide the optimal integer solution of a given instance of the ATFM problem.

The flaws just exposed leave room for potential improvements and pave the way for future research possibilities. In particular, regarding the attempt to obtain an even tighter formulation (with a smaller integrality gap and fewer fractional variables) than ours, we suggest trying to find further classes of valid inequalities. It would probably be important that these classes of valid inequalities also involve configuration-related variables. The latter seem to be the variables that make it difficult to find a formulation whose LPR better approximates the convex hull of the solutions of the ILP problem. In fact, it is worth noting that the formulation present in [26], which is not tight at all, was built starting from a pretty tight formulation proposed by Bertsimas and Stock Patterson (see [8]). The formulation presented in [8], however, unlike the one present in [26], does not include this type of variables (note that the concept of airspace configuration itself is not present in [8]). It could therefore be precisely these variables, and the non-trivial challenge of connecting them with the others, that make it difficult to find a tight formulation regardless, as our simulations show, of how often it is possible to change the airspace configuration.

Finally, it might also be interesting to study what would happen if we introduced some variations in the model. One possibility could be, for example, to see what would happen if we added the option of re-routing a flight, reformulating the objective function in such a way as to penalize the choice of alternative routes compared to the scheduled ones.

Appendix

Here we report the MATLAB file flights.m used to generate the spatial trajectories of the flights present in the instances on which we carried out our computational experiments. We also report the atfm.dat, atfm.mod and atfm.run files associated with Flights 10 as examples of the .dat, .mod and .run files used to implement the model (described through our new formulation) in AMPL.

flights.m

```
%%%%% GENERATION OF SPATIAL TRAJECTIORIES FOR FLIGHTS

% GENERATION OF THE AIRSPACE, WHICH CAN BE SEEN AS A WEIGHTED UNDIRECTED GRAPH
% Note that the generation of weights is random
s = ["DUB" "CPH" "LHR" "AMS" "CDG" "FRA" "ZRH" "VIE" "a" "a" "a" "b" "b" "b" "b" "c"
     "c" "c" "c" "d" "d" "e" "e" "e" "f" "f" "f" "f" "g" "g" "g" "g" "h" "h" "i" "i"
     "i" "j" "j" "j" "j" "k" "k" "k" "k" "l" "l" "m" "n" "o"];
t = ["a" "d" "e" "g" "j" "k" "o" "p" "b" "e" "f" "c" "e" "f" "g" "d" "f" "g" "h" "g"
     "h" "f" "i" "j" "g" "i" "j" "k" "h" "j" "k" "l" "k" "l" "j" "m" "n" "k" "m" "n"
     "o" "l" "n" "o" "p" "o" "p" "n" "o" "p"];
rng('shuffle');
weights = rand(1,length(s));
G = graph(s,t,weights);
% plot(G,'EdgeLabel',G.Edges.Weight)
% Removing '%' from the previous line it is possible to plot the graph

% AIRPORTS AND MAIN AIRPORTS
AIRPORTS = ["DUB" "CPH" "LHR" "AMS" "CDG" "FRA" "ZRH" "VIE"];
PRIMARY = ["DUB" "LHR" "AMS" "CDG" "FRA"];

F = 256; % Number of flights

% GENERATION OF SPATIAL TRAJECTIORIES FOR THE FIRST 9 FLIGHTS
for i = 1:9
    d = PRIMARY(randi([1 length(PRIMARY)]));
```

```
    a = PRIMARY(randi([1 length(PRIMARY)]));
    while a == d
        a = PRIMARY(randi([1 length(PRIMARY)]));
    end
    P = shortestpath(G,d,a);
    fprintf('set PATHS[f00%d] = ',i);
    for j = 1:length(P)
        fprintf('%s ',P(j));
    end
    fprintf(';\n')
end

% GENERATION OF SPATIAL TRAJECTIORIES FOR FLIGHTS 10 TO 99
for i = 10:99
    d = PRIMARY(randi([1 length(PRIMARY)]));
    a = PRIMARY(randi([1 length(PRIMARY)]));
    while a == d
        a = PRIMARY(randi([1 length(PRIMARY)]));
    end
    P = shortestpath(G,d,a);
    fprintf('set PATHS[f0%d] = ',i);
    for j = 1:length(P)
        fprintf('%s ',P(j));
    end
    fprintf(';\n')
end

% GENERATION OF SPATIAL TRAJECTORIES FOR FLIGHTS 100 TO 128
for i = 100:128
    d = PRIMARY(randi([1 length(PRIMARY)]));
    a = PRIMARY(randi([1 length(PRIMARY)]));
    while a == d
        a = PRIMARY(randi([1 length(PRIMARY)]));
    end
    P = shortestpath(G,d,a);
    fprintf('set PATHS[f%d] = ',i);
    for j = 1:length(P)
        fprintf('%s ',P(j));
    end
    fprintf(';\n')
end
```

```

% GENERATION OF SPATIAL TRAJECTORIES FOR FLIGHTS 129 TO 256
for i = 129:F
    d = AIRPORTS(randi([1 length(AIRPORTS)]));
    a = AIRPORTS(randi([1 length(AIRPORTS)]));
    while a == d
        a = AIRPORTS(randi([1 length(AIRPORTS)]));
    end
    P = shortestpath(G,d,a);
    fprintf('set PATHS[f%d] = ',i);
    for j = 1:length(P)
        fprintf('%s ',P(j));
    end
    fprintf(';\n')
end

% PRINT FLIGHT NAMES ON THE SCREEN
for i = 1:9
    fprintf('f00%d ',i);
end
for i = 10:99
    fprintf('f0%d ',i);
end
for i = 100:F
    fprintf('f%d ',i);
end
fprintf(';\n');
```

atfm.dat

```
set MAPS := M01 M02 M03 ; # LIST OF CONFIGURATIONS
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 OF FLIGHTS
set AIRPORTS := DUB CPH LHR AMS CDG FRA ZRH VIE ; # SET OF AIRPORTS
set ELEMENTARY := a b c d e f g h i j k l m n o p ; # SET OF ELEMENTARY SECTORS
set COLLAPSED := C01 C02 C03 C04 C05 C06 C07 C08 C09 C10 C11 C12 C13 C14 C15 C16 ; #
      SET OF COLLAPSED SECTORS

# SETS OF COLLAPSED SECTORS BELONGING TO THE VARIOUS CONFIGURATIONS
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 ;

# SETS OF ELEMENTARY SECTORS WHICH FORM THE VARIOUS COLLAPSED SECTORS
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 OF TIME PERIODS

# SPATIAL TRAJECTORIES OF THE VARIOUS FLIGHTS
```

```
set PATHS[f001] = CDG j g k f e LHR ;
set PATHS[f002] = LHR e f k g j CDG ;
set PATHS[f003] = AMS g c b a DUB ;
set PATHS[f004] = CDG j g k FRA ;
set PATHS[f005] = CDG j g AMS ;
set PATHS[f006] = CDG j g k f e LHR ;
set PATHS[f007] = AMS g k f e LHR ;
set PATHS[f008] = DUB a b c g j CDG ;
set PATHS[f009] = AMS g k FRA ;
set PATHS[f010] = LHR e f k g AMS ;
set PATHS[f011] = FRA k f a DUB ;
set PATHS[f012] = FRA k g j CDG ;
set PATHS[f013] = DUB a b c g j CDG ;
set PATHS[f014] = LHR e f a DUB ;
set PATHS[f015] = FRA k f e LHR ;
set PATHS[f016] = CDG j g c b a DUB ;
set PATHS[f017] = LHR e f a DUB ;
set PATHS[f018] = LHR e f k g j CDG ;
set PATHS[f019] = CDG j g k f e LHR ;
set PATHS[f020] = FRA k g AMS ;
set PATHS[f021] = DUB a f k FRA ;
set PATHS[f022] = CDG j g AMS ;
set PATHS[f023] = AMS g j CDG ;
set PATHS[f024] = AMS g k f e LHR ;
set PATHS[f025] = CDG j g AMS ;
set PATHS[f026] = DUB a f k FRA ;
set PATHS[f027] = CDG j g AMS ;
set PATHS[f028] = DUB a b c g j CDG ;
set PATHS[f029] = DUB a b c g AMS ;
set PATHS[f030] = FRA k g AMS ;
set PATHS[f031] = LHR e f a DUB ;
set PATHS[f032] = AMS g c b a DUB ;
set PATHS[f033] = DUB a f e LHR ;
set PATHS[f034] = DUB a b c g j CDG ;
set PATHS[f035] = DUB a f k FRA ;
set PATHS[f036] = FRA k f e LHR ;
set PATHS[f037] = AMS g k FRA ;
set PATHS[f038] = FRA k g j CDG ;
set PATHS[f039] = FRA k g j CDG ;
set PATHS[f040] = AMS g j CDG ;
set PATHS[f041] = FRA k f e LHR ;
```

```
set PATHS[f042] = CDG j g c b a DUB ;
set PATHS[f043] = CDG j g k f e LHR ;
set PATHS[f044] = FRA k g AMS ;
set PATHS[f045] = AMS g c b a DUB ;
set PATHS[f046] = AMS g k FRA ;
set PATHS[f047] = FRA k f e LHR ;
set PATHS[f048] = DUB a b c g j CDG ;
set PATHS[f049] = FRA k g j CDG ;
set PATHS[f050] = CDG j g k f e LHR ;
set PATHS[f051] = CDG j g k FRA ;
set PATHS[f052] = CDG j g k f e LHR ;
set PATHS[f053] = DUB a b c g j CDG ;
set PATHS[f054] = LHR e f k g AMS ;
set PATHS[f055] = AMS g k FRA ;
set PATHS[f056] = LHR e f k FRA ;
set PATHS[f057] = DUB a b c g j CDG ;
set PATHS[f058] = LHR e f k g j CDG ;
set PATHS[f059] = LHR e f a DUB ;
set PATHS[f060] = DUB a f e LHR ;
set PATHS[f061] = AMS g c b a DUB ;
set PATHS[f062] = LHR e f a DUB ;
set PATHS[f063] = LHR e f k g AMS ;
set PATHS[f064] = AMS g c b a DUB ;
set PATHS[f065] = AMS g j CDG ;
set PATHS[f066] = AMS g k f e LHR ;
set PATHS[f067] = LHR e f a DUB ;
set PATHS[f068] = CDG j g k f e LHR ;
set PATHS[f069] = DUB a b c g AMS ;
set PATHS[f070] = LHR e f k g AMS ;
set PATHS[f071] = DUB a b c g AMS ;
set PATHS[f072] = DUB a b c g AMS ;
set PATHS[f073] = CDG j g AMS ;
set PATHS[f074] = AMS g k f e LHR ;
set PATHS[f075] = DUB a b c g AMS ;
set PATHS[f076] = LHR e f k g AMS ;
set PATHS[f077] = AMS g k f e LHR ;
set PATHS[f078] = CDG j g c b a DUB ;
set PATHS[f079] = FRA k f a DUB ;
set PATHS[f080] = CDG j g k f e LHR ;
set PATHS[f081] = CDG j g c b a DUB ;
set PATHS[f082] = DUB a f e LHR ;
```

```
set PATHS[f083] = LHR e f k g AMS ;
set PATHS[f084] = CDG j g AMS ;
set PATHS[f085] = DUB a f k FRA ;
set PATHS[f086] = CDG j g k FRA ;
set PATHS[f087] = AMS g k f e LHR ;
set PATHS[f088] = AMS g j CDG ;
set PATHS[f089] = DUB a f k FRA ;
set PATHS[f090] = FRA k g j CDG ;
set PATHS[f091] = DUB a f e LHR ;
set PATHS[f092] = LHR e f k g j CDG ;
set PATHS[f093] = AMS g j CDG ;
set PATHS[f094] = FRA k f a DUB ;
set PATHS[f095] = FRA k g AMS ;
set PATHS[f096] = AMS g c b a DUB ;
set PATHS[f097] = LHR e f k g j CDG ;
set PATHS[f098] = FRA k f a DUB ;
set PATHS[f099] = CDG j g k f e LHR ;
set PATHS[f100] = FRA k g AMS ;
set PATHS[f101] = LHR e f k g j CDG ;
set PATHS[f102] = FRA k g j CDG ;
set PATHS[f103] = FRA k g AMS ;
set PATHS[f104] = CDG j g k f e LHR ;
set PATHS[f105] = CDG j g k FRA ;
set PATHS[f106] = AMS g j CDG ;
set PATHS[f107] = AMS g j CDG ;
set PATHS[f108] = DUB a b c g AMS ;
set PATHS[f109] = DUB a b c g AMS ;
set PATHS[f110] = LHR e f k FRA ;
set PATHS[f111] = AMS g j CDG ;
set PATHS[f112] = FRA k f a DUB ;
set PATHS[f113] = CDG j g k f e LHR ;
set PATHS[f114] = CDG j g AMS ;
set PATHS[f115] = FRA k g AMS ;
set PATHS[f116] = FRA k g j CDG ;
set PATHS[f117] = CDG j g AMS ;
set PATHS[f118] = CDG j g AMS ;
set PATHS[f119] = LHR e f k g j CDG ;
set PATHS[f120] = LHR e f k g AMS ;
set PATHS[f121] = LHR e f a DUB ;
set PATHS[f122] = AMS g j CDG ;
set PATHS[f123] = FRA k f e LHR ;
```

```
set PATHS[f124] = FRA k f a DUB ;
set PATHS[f125] = FRA k g AMS ;
set PATHS[f126] = FRA k f e LHR ;
set PATHS[f127] = AMS g k f e LHR ;
set PATHS[f128] = AMS g c b a DUB ;
set PATHS[f129] = AMS g c b a DUB ;
set PATHS[f130] = FRA k f e LHR ;
set PATHS[f131] = LHR e f k g AMS ;
set PATHS[f132] = DUB a b c g j CDG ;
set PATHS[f133] = CPH d h g j CDG ;
set PATHS[f134] = ZRH o p VIE ;
set PATHS[f135] = DUB a b c g j CDG ;
set PATHS[f136] = VIE p l g h d CPH ;
set PATHS[f137] = CPH d h g k FRA ;
set PATHS[f138] = FRA k g j CDG ;
set PATHS[f139] = DUB a b c g l o ZRH ;
set PATHS[f140] = AMS g j CDG ;
set PATHS[f141] = CPH d h c b a DUB ;
set PATHS[f142] = CDG j g h d CPH ;
set PATHS[f143] = CDG j g l p VIE ;
set PATHS[f144] = LHR e f k g j CDG ;
set PATHS[f145] = DUB a b c g j CDG ;
set PATHS[f146] = VIE p k f e LHR ;
set PATHS[f147] = VIE p o ZRH ;
set PATHS[f148] = VIE p k f a DUB ;
set PATHS[f149] = VIE p l g AMS ;
set PATHS[f150] = DUB a f k FRA ;
set PATHS[f151] = ZRH o l g h d CPH ;
set PATHS[f152] = FRA k g AMS ;
set PATHS[f153] = AMS g k f e LHR ;
set PATHS[f154] = DUB a b c g j CDG ;
set PATHS[f155] = FRA k g AMS ;
set PATHS[f156] = CPH d h g j CDG ;
set PATHS[f157] = CDG j g k FRA ;
set PATHS[f158] = FRA k o ZRH ;
set PATHS[f159] = VIE p o ZRH ;
set PATHS[f160] = DUB a b c g AMS ;
set PATHS[f161] = FRA k f e LHR ;
set PATHS[f162] = FRA k g j CDG ;
set PATHS[f163] = AMS g k FRA ;
set PATHS[f164] = DUB a b c g l o ZRH ;
```

```
set PATHS[f165] = FRA k g AMS ;
set PATHS[f166] = DUB a f k p VIE ;
set PATHS[f167] = LHR e f k g h d CPH ;
set PATHS[f168] = LHR e f k p VIE ;
set PATHS[f169] = ZRH o p VIE ;
set PATHS[f170] = VIE p o ZRH ;
set PATHS[f171] = VIE p l g h d CPH ;
set PATHS[f172] = LHR e f k FRA ;
set PATHS[f173] = ZRH o l g AMS ;
set PATHS[f174] = CDG j g AMS ;
set PATHS[f175] = FRA k g h d CPH ;
set PATHS[f176] = LHR e f k o ZRH ;
set PATHS[f177] = LHR e f k p VIE ;
set PATHS[f178] = CDG j o ZRH ;
set PATHS[f179] = ZRH o k FRA ;
set PATHS[f180] = VIE p o ZRH ;
set PATHS[f181] = FRA k f e LHR ;
set PATHS[f182] = DUB a b c g l o ZRH ;
set PATHS[f183] = ZRH o k f e LHR ;
set PATHS[f184] = LHR e f k FRA ;
set PATHS[f185] = CPH d h c b a DUB ;
set PATHS[f186] = DUB a f e LHR ;
set PATHS[f187] = ZRH o l g h d CPH ;
set PATHS[f188] = ZRH o j CDG ;
set PATHS[f189] = CPH d h g j CDG ;
set PATHS[f190] = ZRH o l g AMS ;
set PATHS[f191] = CPH d h g k f e LHR ;
set PATHS[f192] = CPH d h g j CDG ;
set PATHS[f193] = DUB a b c g AMS ;
set PATHS[f194] = VIE p o ZRH ;
set PATHS[f195] = VIE p o ZRH ;
set PATHS[f196] = FRA k f e LHR ;
set PATHS[f197] = LHR e f k g AMS ;
set PATHS[f198] = CDG j g k f e LHR ;
set PATHS[f199] = LHR e f k g h d CPH ;
set PATHS[f200] = AMS g l p VIE ;
set PATHS[f201] = VIE p k f a DUB ;
set PATHS[f202] = CDG j g c b a DUB ;
set PATHS[f203] = VIE p o ZRH ;
set PATHS[f204] = DUB a b c g l o ZRH ;
set PATHS[f205] = ZRH o k FRA ;
```

```
set PATHS[f206] = DUB a b c h d CPH ;
set PATHS[f207] = FRA k o ZRH ;
set PATHS[f208] = LHR e f k g AMS ;
set PATHS[f209] = CPH d h c b a DUB ;
set PATHS[f210] = CDG j g h d CPH ;
set PATHS[f211] = FRA k g h d CPH ;
set PATHS[f212] = CDG j o ZRH ;
set PATHS[f213] = DUB a f e LHR ;
set PATHS[f214] = ZRH o k f e LHR ;
set PATHS[f215] = CDG j g c b a DUB ;
set PATHS[f216] = VIE p l g AMS ;
set PATHS[f217] = VIE p l g h d CPH ;
set PATHS[f218] = VIE p l g AMS ;
set PATHS[f219] = AMS g c b a DUB ;
set PATHS[f220] = ZRH o k f e LHR ;
set PATHS[f221] = AMS g l p VIE ;
set PATHS[f222] = CDG j g AMS ;
set PATHS[f223] = CDG j g c b a DUB ;
set PATHS[f224] = CDG j o ZRH ;
set PATHS[f225] = CPH d h g l p VIE ;
set PATHS[f226] = ZRH o l g h d CPH ;
set PATHS[f227] = VIE p l g h d CPH ;
set PATHS[f228] = VIE p o ZRH ;
set PATHS[f229] = FRA k g j CDG ;
set PATHS[f230] = LHR e f k g AMS ;
set PATHS[f231] = FRA k f e LHR ;
set PATHS[f232] = CPH d h g l p VIE ;
set PATHS[f233] = DUB a f k p VIE ;
set PATHS[f234] = AMS g l p VIE ;
set PATHS[f235] = VIE p l g j CDG ;
set PATHS[f236] = LHR e f k o ZRH ;
set PATHS[f237] = CPH d h c b a DUB ;
set PATHS[f238] = FRA k g j CDG ;
set PATHS[f239] = LHR e f k o ZRH ;
set PATHS[f240] = ZRH o k FRA ;
set PATHS[f241] = AMS g j CDG ;
set PATHS[f242] = AMS g c b a DUB ;
set PATHS[f243] = AMS g c b a DUB ;
set PATHS[f244] = ZRH o l g c b a DUB ;
set PATHS[f245] = CDG j o ZRH ;
set PATHS[f246] = CDG j g c b a DUB ;
```

```
279 set PATHS[f247] = DUB a b c g l o ZRH ;
280 set PATHS[f248] = FRA k g h d CPH ;
281 set PATHS[f249] = LHR e f a DUB ;
282 set PATHS[f250] = DUB a f k p VIE ;
283 set PATHS[f251] = CPH d h g j CDG ;
284 set PATHS[f252] = LHR e f k g j CDG ;
285 set PATHS[f253] = AMS g k FRA ;
286 set PATHS[f254] = DUB a f e LHR ;
287 set PATHS[f255] = VIE p l g AMS ;
288 set PATHS[f256] = AMS g l o ZRH ;
289
290 param D default 8 ; # DEPARTURE CAPACITY OF ANY AIRPORT AT ANY TIME
291 param A default 8 ; # ARRIVAL CAPACITY OF ANY AIRPORT AT ANY TIME
292 param cg default 1 ; # COST OF HOLDING ANY FLIGHT ON THE GROUND FOR ONE UNIT OF TIME
293 param ca default 3 ; # COST OF HOLDING ANY FLIGHT IN THE AIR FOR ONE UNIT OF TIME
294
295 # FOR EACH FLIGHT WE RANDOMLY GENERATE THE SCHEDULED NUMBER (> 0) OF TIME UNITS IT
      MUST SPEND IN EACH ELEMENTARY SECTOR INCLUDED IN ITS PATH (THIS NUMBER IS 0 IN
      THE CASE OF THE DEPARTURE AIRPORT). IN PARTICULAR, WE WILL MAKE SURE THAT IF A
      FLIGHT HAS TO CROSS AN ENTIRE ELEMENTARY SECTOR IT WILL TAKE LONGER THAN IF THE
      FLIGHT LANDS AT AN AIRPORT CONTAINED IN AN ELEMENTARY SECTOR
296 for {f in FLIGHTS} {
297   let l[f,first(PATHS[f])] := 0; # CASE OF THE DEPARTURE AIRPORT
298   for {j in PATHS[f]: ord0(j,PATHS[f]) > 1 and ord0(j,PATHS[f]) < card(PATHS[f])} {
299     if (prev(j,PATHS[f]) in ELEMENTARY and next(j,PATHS[f]) in ELEMENTARY) then {
300       let l[f,j] := floor(Uniform(3, 5));
301     }
302     else {
303       let l[f,j] := floor(Uniform(1, 3));
304     }
305   }
306 }
307
308 # FOR EACH FLIGHT WE: RANDOMLY GENERATE THE MAXIMUM DELAY ALLOWED IN ARRIVAL;
      CALCULATE (USING THE VARIOUS l[f,j]) ITS SCHEDULED DURATION; RANDOMLY GENERATE
      ITS SCHEDULED DEPARTURE AND ARRIVAL TIMES
309 for {f in FLIGHTS} {
310   let delay[f] := floor(Uniform(0, 3));
311   let duration[f] := 0;
312   for {j in PATHS[f]: ord0(j,PATHS[f]) > 1 and ord0(j,PATHS[f]) < card(PATHS[f])} {
313     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];
}

# FOR EACH FLIGHT AND EVERY ELEMENTARY SECTOR ON ITS PATH WE COMPUTE THE SET OF
    FEASIBLE TIMES FOR THAT FLIGHT TO ARRIVE IN THAT 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 MAPS ordered; # SET OF CONFIGURATIONS
set FLIGHTS; # SET OF FLIGHTS
set AIRPORTS; # SET OF AIRPORTS
set ELEMENTARY; # SET OF ELEMENTARY SECTORS
set COLLAPSED; # SET OF COLLAPSED SECTORS
set PARTITIONS {MAPS}; # SETS OF COLLAPSED SECTORS BELONGING TO THE VARIOUS
    CONFIGURATIONS
set BELONGING {COLLAPSED} ordered; # SETS OF ELEMENTARY SECTORS WHICH FORM THE
    VARIOUS COLLAPSED SECTORS
set TIMES ordered; # SET OF TIME PERIODS
set PATHS {FLIGHTS} ordered; # SPATIAL TRAJECTORIES OF THE VARIOUS FLIGHTS
param D {AIRPORTS, TIMES}; # D[k,t] = DEPARTURE CAPACITY OF AIRPORT k AT TIME t
param A {AIRPORTS, TIMES}; # A[k,t] = ARRIVAL CAPACITY OF AIRPORT k AT TIME t
param S {COLLAPSED, TIMES}; # S[h,t] = CAPACITY OF COLLAPSED SECTOR h AT TIME t
param l {f in FLIGHTS, j in PATHS[f]: ord0(j,PATHS[f]) < card(PATHS[f])}; # l[f,j] =
    SCHEDULED NUMBER OF TIME UNITS THAT FLIGHT f MUST SPEND IN j
param delay {FLIGHTS}; # delay[f] = MAXIMUM DELAY IN ARRIVAL ALLOWED FOR FLIGHT f
param d {FLIGHTS}; # d[f] = SCHEDULED DEPARTURE TIME OF FLIGHT f
param r {FLIGHTS}; # r[f] = SCHEDULED ARRIVAL TIME OF FLIGHT f
```

```
17 param duration {FLIGHTS}; # duration[f] = SCHEDULED DURATION OF FLIGHT f = r[f] - d[
       f]
18 param cg {FLIGHTS}; # cg[f] = COST OF HOLDING FLIGHT f ON THE GROUND FOR ONE UNIT
       OF TIME
19 param ca {FLIGHTS}; # ca[f] = COST OF HOLDING FLIGHT f IN THE AIR FOR ONE UNIT OF
       TIME
20 param tmin {f in FLIGHTS, PATHS[f]}; # tmin[f,j] = FIRST FEASIBLE TIME PERIOD FOR
       FLIGHT f TO ARRIVE IN (OR TAKE OFF FROM/LAND AT) j
21 param tmax {f in FLIGHTS, PATHS[f]}; # tmax[f,j] = LAST FEASIBLE TIME PERIOD FOR
       FLIGHT f TO ARRIVE IN (OR TAKE OFF FROM/LAND AT) j
22 param C {COLLAPSED, TIMES}; # C[h,t] = THEORETICAL MAXIMUM NUMBER OF FLIGHTS THAT
       COULD BE IN COLLAPSED SECTOR h AT TIME t
23 param tau > 0 integer; # MINIMUM NUMBER OF CONSECUTIVE TIME PERIODS IN WHICH THE (
       TEMPORARILY) CHOSEN CONFIGURATION MUST REMAIN ACTIVE BEFORE IT CAN BE CHANGED
24 param counter_w; # COUNTER OF VARIABLES w
25 param counter_y; # COUNTER OF VARIABLES y
26 param counter_w_int; # COUNTER OF VARIABLES w THAT ARE INTEGER
27 param counter_w_frac; # COUNTER OF VARIABLES w THAT ARE FRACTIONAL
28 param counter_y_int; # COUNTER OF VARIABLES y THAT ARE INTEGER
29 param counter_y_frac; # COUNTER OF VARIABLES y THAT ARE FRACTIONAL
30 param number_cover_inequalities {m in MAPS, h in PARTITIONS[m], t in TIMES} integer
       default 0; # number_cover_inequalities[m,h,t] = NUMBER OF COVER INEQUALITIES
       FOUND (USING THE PROCEDURE DESCRIBED IN ALGORITHM 5.1) TREATING THE
       CORRESPONDING CONSTRAINT (5.16) AS IF IT WAS A REAL KNAPSACK-TYPE CONSTRAINT
31
32 # PARAMETERS NECESSARY TO STORE THE COVER INEQUALITIES FOUND USING ALGORITHM 5.1
33 param cover_inequalities1 {m in MAPS, h in PARTITIONS[m], j in BELONGING[h], t in
       TIMES, 1..number_cover_inequalities[m,h,t], f in FLIGHTS: ord0(j,PATHS[f]) > 1
       and t >= tmin[f,j] and t <= tmax[f,next(j,PATHS[f])]-1} integer default 0;
34 param cover_inequalities2 {m in MAPS, h in PARTITIONS[m], t in TIMES, 1..
       number_cover_inequalities[m,h,t]} integer default 0;
35
36 # PARAMETERS NECESSARY TO STORE MOMENTARILY THE VARIOUS SOLUTIONS FOUND AT STEP 1
       OF ALGORITHM 5.1
37 param wbar {f in FLIGHTS, j in PATHS[f], t in TIMES: t >= tmin[f,j] and t <= tmax[f,
       j]} default 0;
38 param ybar {m in MAPS, t in TIMES} default 0;
39
40 param flag integer default 0; # PARAMETER USED TO UNDERSTAND WHEN TO END THE SEARCH
       FOR COVER INEQUALITIES
41 param flag1 integer default 0; # NUMBER OF TIMES STEP 3 OF ALGORITHM 5.1 IS EXECUTED
```

```
param flag2 integer default 0; # NUMBER OF PROBLEMS OF THE FORM (3.5) SOLVED USING
    ALGORITHM 5.1
param flag3 integer default 0; # NUMBER OF CONSTRAINTS (5.26) = TOTAL NUMBER OF
    COVER INEQUALITIES FOUND USING ALGORITHM 5.1
var w {f in FLIGHTS, j in PATHS[f], t in TIMES: t >= tmin[f,j] and t <= tmax[f,j]}
    binary; # VARIABLES w
var y {m in MAPS, t in TIMES} binary; # VARIABLES y
var g {f in FLIGHTS}; # g[f] = TOTAL NUMBER OF TIME UNITS THAT FLIGHT f IS HELD ON
    THE GROUND (THROUGH GROUND-HOLDING POLICY)
var a {f in FLIGHTS}; # a[f] = TOTAL NUMBER OF TIME UNITS THAT FLIGHT f IS HELD IN
    THE AIR (THROUGH AIRBORNE-HOLDING AND/OR SPEED CONTROL POLICIES)

# VARIABLES USED IN SOLVING THE VARIOUS PROBLEMS OF THE FORM (3.5) SOLVED USING
    ALGORITHM 5.1
var z {j in ELEMENTARY, t in TIMES, f in FLIGHTS: ord0(j,PATHS[f]) > 1 and t >= tmin
    [f,j] and t <= tmax[f,next(j,PATHS[f])]-1} binary;
var x {m in MAPS, t in TIMES} binary;

# 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])]
    then w[f,first(PATHS[f]),t-1] else 0))))+ca[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])] then w[f,last(PATHS[f]),t-1] else 0))))+(ca[f]-cg[f
    ])*d[f]-ca[f]*r[f]);

# CONSTRAINTS (4.1)
subject to cons1 {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 (4.2)
subject to cons2 {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];
```

```
65
66 # CONSTRAINTS (5.16)
67 subject to cons3 {m in MAPS, h in PARTITIONS[m], t in TIMES}:
68   sum {j in BELONGING[h]} sum {f in FLIGHTS: ord0(j,PATHS[f]) > 1} ((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]);
69
70 # CONSTRAINTS (5.11)
71 subject to cons4 {j in ELEMENTARY, t in TIMES}:
72   sum {f in FLIGHTS: ord0(j,PATHS[f]) > 1} ((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)) <= max {h in COLLAPSED: ord0(
       j,BELONGING[h]) >= 1} S[h,t];
73
74 # CONSTRAINTS (4.4)
75 subject to cons5 {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]}:
76   ((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;
77
78 # CONSTRAINTS (4.5)
79 subject to cons6 {f in FLIGHTS, j in PATHS[f], t in TIMES: t >= tmin[f,j] and t <=
       tmax[f,j]}:
80   (w[f,j,t]-(if t-1 >= tmin[f,j] then w[f,j,t-1] else 0)) >= 0;
81
82 # CONSTRAINTS (4.7)
83 subject to cons7 {f in FLIGHTS, j in PATHS[f], t in TIMES: t = tmax[f,j]}:
84   w[f,j,t] = 1;
85
86 # CONSTRAINTS (4.8)
87 subject to cons8 {t in TIMES}:
88   sum{m in MAPS} y[m,t] = 1;
89
90 # CONSTRAINTS (4.9)
91 subject to cons9 {m in MAPS, t in TIMES, u in t+1..min(t+tau-1,last(TIMES))}:
92   (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]);
```

```
93
94 # CONSTRAINTS (5.26) (THESE CONSTRAINTS WILL BE FOUND USING ALGORITHM 5.1)
95 subject to conscover{m in MAPS, h in PARTITIONS[m], t in TIMES, i in 1..
       number_cover_inequalities[m,h,t]}:
96   cover_inequalities2[m,h,t,i] * y[m,t] + sum {j in BELONGING[h]} (sum{f in FLIGHTS:
       ord0(j,PATHS[f]) > 1 and t >= tmin[f,j] and t <= tmax[f,next(j,PATHS[f])]-1}
       (cover_inequalities1[m,h,j,t,i,f] * ((if t <= tmax[f,j] then w[f,j,t] else 1) -
       (if t >= tmin[f,next(j,PATHS[f])] then w[f,next(j,PATHS[f]),t] else 0)))) <= -1
       + cover_inequalities2[m,h,t,i] + sum {j in BELONGING[h]} (sum{f in FLIGHTS:
       ord0(j,PATHS[f]) > 1 and t >= tmin[f,j] and t <= tmax[f,next(j,PATHS[f])]-1}
       cover_inequalities1[m,h,j,t,i,f]);
97
98
99 ### ELEMENTS FOR THE VARIOUS PROBLEMS OF THE FORM (3.5)
100
101 # OBJECTIVE FUNCTIONS FOR THE VARIOUS PROBLEMS OF THE FORM (3.5)
102 minimize sl1{m in MAPS, h in PARTITIONS[m], t in TIMES}: (1 - ybar[m,t]) * x[m,t] +
       sum {j in BELONGING[h]} (sum{f in FLIGHTS: ord0(j,PATHS[f]) > 1 and t >= tmin[f,
       j] and t <= tmax[f,next(j,PATHS[f])]-1} ((1 - ((if t <= tmax[f,j] then wbar[f,j,
       t] else 1) - (if t >= tmin[f,next(j,PATHS[f])] then wbar[f,next(j,PATHS[f]),t]
       else 0))) * z[j,t,f]));
103
104 # CONSTRAINTS FOR THE VARIOUS PROBLEMS OF THE FORM (3.5)
105 subject to vsl1{m in MAPS, h in PARTITIONS[m], t in TIMES}:
106   C[h,t] * x[m,t] + sum {j in BELONGING[h]} (sum{f in FLIGHTS: ord0(j,PATHS[f]) > 1
        and t >= tmin[f,j] and t <= tmax[f,next(j,PATHS[f])]-1} z[j,t,f]) >= 1 + S[h,t]
       + C[h,t];
```

atfm.run

```
1 reset;
2 model atfm.mod;
3 data atfm.dat;
4 option presolve 0;
5 option solver "/home/0/Software/CPLEX/ibm/ILOG/CPLEX_Studio128/cplex/bin/x86-64
      _linux/cplexamp";
6 problem master: w, y, g, a, fun, cons1, cons2, cons3, cons4, cons5, cons6, cons7,
      cons8, cons9, conscover; # THIS IS THE ATFM PROBLEM WE WANT TO SOLVE
```

```
problem slave1 {m in MAPS, h in PARTITIONS[m], t in TIMES}: x, z, sl1[m,h,t], vsl1[m,h,t]; # THESE ARE THE PROBLEMS OF THE FORM (3.5) THAT APPEAR USING ALGORITHM 5.1
# WE WILL SOLVE THE PROBLEM FOR DIFFERENT VALUES OF TAU
for {periods in {1,6,12,36}} {
  printf "\n*************************************************************************************************\n" > atfm.txt;
  let tau := periods;
  # WE WILL SOLVE THE PROBLEM FOR DIFFERENT VALUES OF THE CAPACITIES OF THE COLLAPSED SECTORS
  for {capacity in {5,10,15,20,25,30,35}} {
    option master.relax_integrality 1;
    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 C[h,t] := (sum {j in BELONGING[h]} sum {f in FLIGHTS: ord0(j,PATHS[f]) > 1 and tmin[f,j] <= t and t <= tmax[f,next(j,PATHS[f])] - 1} 1) - S[h,t];
      }
    }
    ### BEGINNING OF ALGORITHM 5.1
    solve master;
    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 flag := 1;
  if match (master.message, "infeasible") > 0 then {
    let flag := 0;
  }
  if (counter_w_frac = 0 and counter_y_frac = 0) then {
    let flag := 0;
  }
  ### BEGINNING OF STEP 3 OF ALGORITHM 5.1
  repeat until (flag = 0) {
    let flag := 0;
    for {f in FLIGHTS, j in PATHS[f], t in TIMES: t >= tmin[f,j] and t <= tmax[f,j
  ]} {
      let wbar[f,j,t] := w[f,j,t];
    }
    for {m in MAPS, t in TIMES} {
      let ybar[m,t] := y[m,t];
    }
    let flag1 := flag1 + 1;
    for {m in MAPS, h in PARTITIONS[m], t in TIMES} {
      solve slave1[m,h,t];
      let flag2 := flag2 + 1;
      if (match (slave1[m,h,t].message, "optimal") > 0 and sl1[m,h,t] < 1 - 1e-6)
      then {
        let flag := 1;
        let flag3 := flag3 + 1;
        let number_cover_inequalities[m,h,t] := number_cover_inequalities[m,h,t] +
        1;
        let cover_inequalities2[m,h,t,number_cover_inequalities[m,h,t]] := x[m,t];
        for {j in BELONGING[h], f in FLIGHTS: ord0(j,PATHS[f]) > 1 and t >= tmin[f
        ,j] and t <= tmax[f,next(j,PATHS[f])]-1} {
          let cover_inequalities1[m,h,j,t,number_cover_inequalities[m,h,t],f] := z
          [j,t,f];
        }
      }
    }
    solve master;
```

```
78      let counter_w := 0;
79      let counter_y := 0;
80      let counter_w_int := 0;
81      let counter_w_frac := 0;
82      let counter_y_int := 0;
83      let counter_y_frac := 0;
84      for {f in FLIGHTS, j in PATHS[f], t in TIMES: t >= tmin[f,j] and t <= tmax[f,j
        ]} {
85        let counter_w := counter_w + 1;
86        if (abs(w[f,j,t]-round(w[f,j,t])) > 1e-6) then {
87          let counter_w_frac := counter_w_frac + 1;
88        }
89        else {
90          let counter_w_int := counter_w_int + 1;
91        }
92      }
93      for {m in MAPS, t in TIMES} {
94        let counter_y := counter_y + 1;
95        if (abs(y[m,t]-round(y[m,t])) > 1e-6) then {
96          let counter_y_frac := counter_y_frac + 1;
97        }
98        else {
99          let counter_y_int := counter_y_int + 1;
100       }
101     }
102     if match (master.message, "infeasible") > 0 then {
103       let flag := 0;
104     }
105     if (counter_w_frac = 0 and counter_y_frac = 0) then {
106       let flag := 0;
107     }
108     ### END OF STEP 3 OF ALGORITHM 5.1
109   };
110   ### END OF ALGORITHM 5.1
111   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])] then w[f,first(PATHS[f]),t-1] else 0)))-d[f];
112   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])] then w[f,last(PATHS[f]),t-1] else 0)))-r[f]-g[f];
113   printf "\ntau = %d, ", tau > atfm.txt;
```

```
114     printf "capacity = %d, ", capacity > atfm.txt;
115     printf "fun = %f, ", fun > atfm.txt;
116     printf "ground = %f, ", sum{f in FLIGHTS} g[f] > atfm.txt;
117     printf "air = %f, ", sum{f in FLIGHTS} a[f] > atfm.txt;
118     printf "counter_w = %d, ", counter_w > atfm.txt;
119     printf "counter_w_int = %d, ", counter_w_int > atfm.txt;
120     printf "counter_w_frac = %d, ", counter_w_frac > atfm.txt;
121     printf "counter_y = %d, ", counter_y > atfm.txt;
122     printf "counter_y_int = %d, ", counter_y_int > atfm.txt;
123     printf "counter_y_frac = %d, ", counter_y_frac > atfm.txt;
124     print master.message > atfm.txt;
125     printf "time_solve = %f. \n", _total_solve_time > atfm.txt;
126     printf "flag1 = %d, ", flag1 > atfm.txt;
127     printf "flag2 = %d, ", flag2 > atfm.txt;
128     printf "flag3 = %d, ", flag3 > atfm.txt;
129     ### WE CAN NOW SOLVE THE ATFM PROBLEM USING OUR NEW FORMULATION
130     option master.relax_integrality 0;
131     solve master;
132     ### WE HAVE JUST SOLVED THE ATFM PROBLEM USING OUR NEW FORMULATION
133     let counter_w := 0;
134     let counter_y := 0;
135     let counter_w_int := 0;
136     let counter_w_frac := 0;
137     let counter_y_int := 0;
138     let counter_y_frac := 0;
139     for {f in FLIGHTS, j in PATHS[f], t in TIMES: t >= tmin[f,j] and t <= tmax[f,j]}
        {
140       let counter_w := counter_w + 1;
141       if (abs(w[f,j,t]-round(w[f,j,t])) > 1e-6) then {
142         let counter_w_frac := counter_w_frac + 1;
143       }
144       else {
145         let counter_w_int := counter_w_int + 1;
146       }
147     }
148     for {m in MAPS, t in TIMES} {
149       let counter_y := counter_y + 1;
150       if (abs(y[m,t]-round(y[m,t])) > 1e-6) then {
151         let counter_y_frac := counter_y_frac + 1;
152       }
153       else {
```

```
154         let counter_y_int := counter_y_int + 1;
155       }
156     }
157     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])] then w[f,first(PATHS[f]),t-1] else 0)))-d[f];
158     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])] then w[f,last(PATHS[f]),t-1] else 0)))-r[f]-g[f];
159     printf "\nILP" > atfm.txt;
160     printf "\ntau = %d, ", tau > atfm.txt;
161     printf "capacity = %d, ", capacity > atfm.txt;
162     printf "fun = %f, ", fun > atfm.txt;
163     printf "ground = %f, ", sum{f in FLIGHTS} g[f] > atfm.txt;
164     printf "air = %f, ", sum{f in FLIGHTS} a[f] > atfm.txt;
165     printf "counter_w = %d, ", counter_w > atfm.txt;
166     printf "counter_y = %d, ", counter_y > atfm.txt;
167     printf "BB_nodes = %d, ", num0(substr(master.message,match(master.message,"s\n")
        +2)) > atfm.txt;
168     print master.message > atfm.txt;
169     printf "time_solve = %f. \n", _total_solve_time > atfm.txt;
170     let flag1 := 0;
171     let flag2 := 0;
172     let flag3 := 0;
173     reset data w, y, g, a, x, z, number_cover_inequalities, cover_inequalities1,
        cover_inequalities2;
174   }
175 }
```

References

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

[2] A. Agustín, A. Alonso-Ayuso, L. F. Escudero and C. Pizarro, *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/. [Accessed on August 1, 2023].

[4] G. Andreatta, L. Capanna, L. De Giovanni and 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*, in Large-Scale Computation and Information Processing in Air Traffic Control: 125-168, L. Bianco and A. R. Odoni (eds). Springer-Verlag, 1993.

[6] ATAG. Available online at: https://aviationbenefits.org/media/167517/aw-oct-final-atag_abbb-2020-publication-digital.pdf, 2020. [Accessed on July 19, 2023].

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

[11] L. De Giovanni and M. Di Summa, *Methods and Models for Combinatorial Optimization. Cover inequalities.* Lecture notes, Università degli Studi di Padova, 2016. Available online at: https://www.math.unipd.it/~luigi/courses/metmodoc1617/mo9.cover.eng.pdf. [Last accessed on July 30, 2023].

[12] EUROCONTROL. EUROCONTROL Aviation Outlook 2050. Main Report. Available online at: https://www.eurocontrol.int/publication/eurocontrol-aviation-outlook-2050, 2022. [Accessed on July 19, 2023].

[13] EUROCONTROL. Introducing the EUROCONTROL Network Manager Operations Centre. Available online at: https://www.eurocontrol.int/publication/introducing-eurocontrol-network-manager-operations-centre-nmoc, 2019. [Accessed on July 23, 2023].

[14] EUROCONTROL. Performance Review Report (PRR) 2022. Available online at: https://www.eurocontrol.int/publication/performance-review-report-prr-2022, 2023. [Accessed on July 20, 2023].

[15] F. D. Fomeni, G. Lulli and K. Zografos, *An optimization model for assigning 4D-trajectories to flights under the TBO concept*, in 12th USA/Europe Air Traffic Management Research and Development Seminar: 332-341, Seattle, Washington, USA, 27-30 June 2017. The European Organisation for the Safety of Air Navigation (EUROCONTROL), 2017.

[16] M. P. Helme, *Reducing air traffic delay in a space-time network*, in 1992 IEEE International Conference on Sysytems, Man and Cybernetics: 236-242, Chicago, Illinois, USA, 18-21 October 1992. IEEE, 1992.

[17] IBM. https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer. [Accessed on August 1, 2023].

[18] ICAO. PANS - ATM (doc 4444), fourteenth edition. Available online at: https://skyrise.aero/wp-content/uploads/2017/03/ICAO-Doc-4444-EN.pdf, 2001. [Accessed on July 23, 2023].

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

[20] MathWorks. MATLAB. https://www.mathworks.com/products/matlab.html. [Accessed on August 1, 2023].

[21] H. Ritchie, *Climate change and flying: what share of global CO_2 emissions come from aviation?,* Published online at ourworldindata.org. Available online at: https://ourworldindata.org/co2-emissions-from-aviation, 2020. [Accessed on July 21, 2023].

[22] R. T. Rockafellar, *Convex Analysis,* Princeton Mathematical Series, vol.28. Princeton University Press, 1970, p.12.

[23] SKYbrary Aviation Safety. 4D Trajectory Concept. https://skybrary.aero/articles/4d-trajectory-concept. [Accessed on August 7, 2023].

[24] Statista. Available online at: https://www.statista.com/statistics/1118397/air-passenger-transport-european-union/, 2023. [Accessed on July 19, 2023].

[25] R. J. Wilson, *Introduction to Graph Theory, Fourth edition*. Addison Wesley Longman Limited, 1996.

[26] L. Zanardelli, *A mathematical programming model for air traffic flow management with dynamic selection of the airspace configuration*. Tesi di Laurea Magistrale in Matematica, Università degli Studi di Padova, 2020.

Acknowledgments

I would like to sincerely thank my supervisor, Professor Luigi De Giovanni, for his assistance during all the various phases that led to the realization of this thesis and for his complete availability.

The next acknowledgments will be written in Italian.

Grazie ai miei genitori che mi hanno sempre sostenuto, consigliato e incoraggiato durante tutto il mio percorso di studi (e non solo).

Grazie a mia sorella per il suo prezioso supporto morale (e per la sua capacità di "vivacizzare" l'ambiente).

Grazie a quella che è una delle mie sostenitrici più sfegatate: mia nonna Diomira.

Grazie infine a tutti quelli che, tra amici, parenti e conoscenti, mi hanno sempre supportato.