NEW MODELS AND METHODS OF MATHEMATICAL OPTIMIZATION IN AIR TRAFFIC FLOW MANAGEMENT

DAVID GARCÍA HEREDIA

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor in Mathematical Engineering

Universidad Carlos III de Madrid

ADVISORS:
Elisenda Molina Ferragut
Antonio Alonso Ayuso

TUTOR:
Elisenda Molina Ferragut

January 2021

A mi familia.
En especial, a mi madre.

Nobody, but Slydini.

– D. Vernon

ACKNOWLEDGMENTS

As any important personal project, this thesis is the result of hours and hours of hard work and dedication plus, and here comes the important part of the sum, the unconditional support and patient of many people.

Therefore, it would not be fair not to start thanking Elisenda and Antonio for all they have done to make this project a reality. This includes the great advice on what ideas to explore, possible ways to do it, what should be our goals, what books are essential in Operations Research to learn it, the corrections of my not uncommon mistakes, the personal advice, the financial support from the research projects for my internship or to participate in congresses, and many, many more. For all that, thank you very much. You can believe me when I say that, although the thesis was my responsibility as a student, it would never have existed without you guys!

And as everything has a beginning, at this point I would like to make a special mention to Prof. F. Javier Otamendi, from the Rey Juan Carlos University. Not only because of the influence that his subject had on me when I chose my master's degree specialization, but also because he introduced me to this world of research. Thank you for that!

I would like to thank the Universidad Carlos III de Madrid, and particularly the department of statistics, for the possibility of having studied my master's and doctorate there, as well as for the financial support, including research, conferences and internship. I greatly appreciate all that, including the work of all my professors during the master's program, from whom I have learned so much.

I am also grateful to all my colleagues in the statistics department. For one reason or another, I always felt good around them. A special mention goes to Prof. Ignacio Cascos, because in addition to everything I have learned from him about teaching and statistics, he also gave me good advice when I was facing important decisions. I truly appreciated that! Surely there are many other people of the department that I would

also like to mention, such as David Delgado, who still does not forgive my "betrayal" ;), or Mike Wiper (*e4, best by test*), but I will hold myself back for the sake of brevity.

In this part of the thesis I do not forget Prof. Manuel Laguna. Not only for the opportunity of working with him, but also for the kind treatment that he gave me when I was overseas, miles away from home. Thanks for everything! Besides, I had the opportunity to get to know that beautiful city called Boulder, and some wonderful people whom I began to call roommates, but whom I now have the pleasure of calling friends.

Speaking of which! At this point I really want to thanks all my friends for the support and warmth that I have received from them. Some of them I have known since child. Some of them I met during my degree at the URJC. And others during my master's and doctorate. And although I must admit that I have not been able to be with them as much time as I would have liked (believe me when I say that this work has taken me many, many hours), I have always enjoyed every moment that I have spent with them. I do not have space to mention you all guys, so I will just say that, if I were born again, I would definitely make the same decisions that led me to meet each one of you. I really appreciate you guys!

And since the most important part is always left to the end, now is the time to thank my family. If without my advisors this thesis would have never existed, without my family I would never have reached what I have reached, including starting this project. Growing up in one family or another has a tremendous impact on anyone's career, and I had the fortune of having a winning ticket. That is why I want to start thanking my parents, Félix and María del Mar, for all the support in the academic and personal side. Without their education and dedication I would not be here, that is for sure. I also want to thank my Aunt Yolanda, who always took care of me with unconditional support. Thank you for everything! Also to my brother Sergio, the genius of the family, of whom I could not be more proud. And to conclude, I would like to thank my grandparents. Because if I can afford to be here today, it is only because they did not give up when things were not that easy. Thank you for that.

David

This research was partially funded by projects MTM2015-63710-P and RTI2018-094269-B-I00 (A. Alonso-Ayuso and D. García-Heredia) from the Government of Spain.

CONTENTS PUBLISHED AND PRESENTED

The author of this thesis appears as a co-author in all the contributions listed below.

PUBLICATIONS

The materials from the following sources are included in the thesis. Their inclusion is not indicated by typographical means or references.

- García-Heredia, David; Alonso-Ayuso, Antonio; and Molina, Elisenda. (2019). "A Combinatorial model to optimize air traffic flow management problems". In: *Computers & Operations Research*, 112, 104768.
  - Url: https://doi.org/10.1016/j.cor.2019.104768.
  - Partially included in Chapter 2.
- García-Heredia, David; Molina, Elisenda; Laguna, Manuel; and Alonso-Ayuso, Antonio. (2020). "A solution method for the shared Resource Constrained Multi-Shortest Path Problem". In: *UC3M Working Papers in Statistics and Econometrics*.
  - Url: https://e-archivo.uc3m.es/handle/10016/30793.
  - Partially included in Chapters 3 and 4.
  - Note: Submitted for publication.

CONFERENCES

- García-Heredia, David; Molina, Elisenda; and Alonso-Ayuso, Antonio. A new MIP model for the Air Traffic Flow Management Problem. *In 2nd European Conference on Stochastic Optimization*, September 2017, Rome (Italy).
- García-Heredia, David; Molina, Elisenda; and Alonso-Ayuso, Antonio. New mathematical optimization models for Air Traffic Flow Management. *In 1st Spanish Young Statisticians and Operational Researches Meeting*, November 2017, Granada (Spain).

- García-Heredia, David; Molina, Elisenda; and Alonso-Ayuso, Antonio. New mathematical optimization models for Air Traffic Flow Management. *In XXXVII Congreso Nacional de Estadística e Investigación Operativa*, May 2018, Oviedo (Spain).
- García-Heredia, David; Alonso-Ayuso, Antonio; Laguna, Manuel; and Molina, Elisenda. A matheuristic for the Common Capacity Constrained Multi Shortest Path Problem. *In XXXVIII Congreso Nacional de Estadística e Investigación Operativa*, September 2019, Alcoi (Spain).

CONTENTS

Abstract xv

Resumen xvii

1 INTRODUCTION 1

2 AIR TRAFFIC FLOW MANAGEMENT PROBLEM 7

2.1 Introduction . . . 8

2.2 Literature Review . . . 10

2.3 Problem Description . . . 13

2.4 Previous Models . . . 15

2.4.1 Notation . . . 16

2.4.2 ATFMP by nodes . . . 17

2.4.3 ATFMRP by nodes . . . 22

2.4.4 ATFMRP by arcs . . . 26

2.5 New Mathematical Formulation . . . 33

2.5.1 Cost analysis in the objective function . . . 38

2.5.2 Model extension . . . 41

2.6 Alternative Formulation . . . 42

2.7 Graph of Conflicts . . . 47

3 SHARED RESOURCE CONSTRAINED MULTI-SHORTEST PATH PROBLEM 51

3.1 Introduction . . . 52

3.2 Problem description . . . 54

3.2.1 Notation . . . 54

3.2.2 Preliminaries: Shortest Path Problem and Resource Constrained Shortest Path Problem . . . 55

3.2.3 Mathematical formulation . . . 56

3.2.4 Key features . . . 57

3.2.5 Application to project scheduling . . . 60

3.3 Solution Methods . . . 64

3.3.1 Introduction to heuristic algorithms . . . 64

3.3.2 Matheuristic algorithm . . . 66

3.3.3 Lagrangian Relaxation . . . 80

4 COMPUTATIONAL EXPERIENCE 89
4.1 ATFM data sets . . . 89
4.1.1 Raw data transformation . . . 90
4.1.2 Sectors and route waypoints . . . 91
4.1.3 Flight plans . . . 92
4.1.4 4D-networks . . . 94
4.1.5 Capacity constraints . . . 95
4.1.6 Instances dimensions . . . 97
4.2 Computational Results . . . 98
4.2.1 Integer Programming Results . . . 98
4.2.2 Matheuristic and Lagrangian Relaxation Results . . . 101
4.2.3 Flight Plans Modifications . . . 114
5 CONCLUSIONS AND FUTURE RESEARCH 119
BIBLIOGRAPHY 125

LIST OF FIGURES

Figure 1 Illustrative example with three airports and four air sectors. . . . 14
Figure 2 ATFMP by nodes for $f = 1$ in the example. 19
Figure 3 ATFMRP by nodes for Case 3 and $f = 1$ in the example. 23
Figure 4 ATFMRP by arcs for $f = 1$ in the example. 29
Figure 5 Illustration of the penalization functions. 41
Figure 6 Illustration of dynamic sector configuration. 42
Figure 7 Example of 3D flight plan. 43
Figure 8 4D-network example. 44
Figure 9 4D-network for two continued flights. 45
Figure 10 Illustrative example of the Shortest Path Problem. 55
Figure 11 Illustrative example of the Resource Constrained Shortest Path Problem. 56
Figure 12 Illustrative example of the Shared Resource Constrained Multi-Shortest Path Problem. 59
Figure 13 Example of a sequence of activities for a project. 62
Figure 14 Extended network for the project in Figure 13. 63
Figure 15 Illustrative example of solving the SRC-MSPP with the algorithm (part I). 69
Figure 16 Illustrative example of solving the SRC-MSPP with the algorithm (part II). 75
Figure 17 Example of sectors and waypoints. 91
Figure 18 Capacity requirements per time period for the most demanded sectors on January 16, 2019. 93
Figure 19 Effect of the pool size for instances of size 30%. 103
Figure 20 Results of the algorithm for instances of size 65% that exact methods could also solve. 104
Figure 21 Results using a random cost structure. 105
Figure 22 Deviation from Lagrangian lower bounds. 108

Figure 23 Percentage of solution time in each phase of the proposed solution procedure. 110
Figure 24 Improvement achieved after each phase of the proposed procedure. 111

LIST OF TABLES

Table 1 Travel times for $f = 1$ in the example. 19
Table 2 Dimensions of the instances in our test set. 97
Table 3 Gurobi results for instances of size 30%. 99
Table 4 Gurobi results for instances of size 65%. 100
Table 5 Parameters values in experimentation. 102
Table 6 Subset of instances solved only by the algorithm. 109
Table 7 Algorithm results for instances of size 30% when using a pool of 200 solutions. 113
Table 8 Algorithm results for instances of size 65% when using a pool of 300 solutions. 114
Table 9 Flight modifications in the solution of the algorithm for instances of size 30%. 116
Table 10 Flight modifications in the solution of the algorithm for instances of size 65%. 117
Table 11 Flight modifications in the solution of the algorithm for instances of size 100%. 118

ABSTRACT

In this thesis we address the problem of Air Traffic Flow Management (ATFM). In brief, this problem consists of finding optimal schedules and routes for a set of flights in such a way that, when the flight plans are executed, no region of the airspace has more aircraft flying over it than allowed by security restrictions. Likewise, no airport should be assigned more departures or arrivals than it can handle.

In the thesis, continuing a research line originated some decades ago, we cope with this problem using mathematical optimization. The thesis content is organized as follows.

In Chapter 2 we give a detailed description of the ATFM problem and review some of the most recent works that also employ mathematical programming to tackle the problem. The chapter also contains our modeling proposals for the ATFM problem. These consist of two new and equivalent 0-1 mathematical programming formulations. The formulations are shown to be an easy way to model different complex situations arising in practice, and permit to solve some limitations of the state-of-the-art models. The chapter concludes presenting a novel methodology to detect, beforehand, routes that will be part of the optimal solution.

In Chapter 3 we generalize some of the results obtained in Chapter 2. Concretely, we introduce a family of shortest path problems that, to the best of our knowledge, has not been previously investigated: the Shared Resource Constrained Multi-Shortest Path Problem. In the chapter we show how to use this family of shortest path problems to solve some type of project scheduling problems. This way, the results obtained in the previous chapter for the ATFM problem are extended to a broader family of scheduling problems. In Chapter 3 we also discuss two different solution methods for the family of shortest path problems presented. The first method consists of a matheuristic algorithm, while the second one is based on two Lagrangian Relaxations of the problem.

Chapter 4 contains an extensive computational experience to validate the results presented in the previous chapters. Moreover, the chapter includes the creation of ATFM instances which have been released for free disposal.

Finally, in Chapter 5 we summarize the main conclusions and contributions accomplished in the thesis. Future lines of research that this work opens are also discussed there.

RESUMEN

En esta tesis tratamos el problema de la Gestión de Flujo del Tráfico Aéreo (ATFM). De manera breve, este problema consiste en encontrar una planificación temporal y de rutas óptima para un conjunto de vuelos de manera que, cuando se ejecuten los planes de vuelo, ninguna región del espacio aéreo tenga más aeronaves volando sobre ella que las permitidas por las restricciones de seguridad. Asimismo, no se debe asignar a ningún aeropuerto más salidas o llegadas de las que pueda manejar.

En la tesis, continuando una línea de investigación originada hace algunas décadas atrás, abordamos este problema mediante optimización matemática. El contenido de la tesis está organizado de la siguiente manera.

En el capítulo 2 damos una descripción detallada del problema del ATFM y revisamos algunos de los trabajos más recientes que también emplean optimización matemática para abordar el problema. El capítulo también contiene nuestras propuestas de modelización para el problema del ATFM. Estas consisten en dos nuevas y equivalentes formulaciones 0-1 de optimización matemática. En el capítulo se muestra como dichas formulaciones permiten modelar de manera sencilla diferentes situaciones complejas que surgen en la práctica, y cómo permiten resolver algunas limitaciones de los modelos que conforman el estado del arte. El capítulo concluye presentado una nueva metodología para detectar, de antemano, rutas que formarán parte de la solución óptima.

En el capítulo 3 generalizamos algunos de los resultados obtenidos en el capítulo 2. Concretamente, introducimos una familia de problemas de camino mínimo que, hasta nuestro entender, no ha sido investigada previamente: el problema de múltiples caminos mínimos con recursos compartidos y restringidos. En el capítulo mostramos cómo usar esta familia de problemas de camino mínimo para resolver algunos problemas de planificación de proyectos. De esta manera, los resultados obtenidos en el capítulo anterior para el problema del ATFM se extienden a una familia más amplia de problemas de planificación. En el capítulo 3 también presentamos dos métodos de resolución diferentes para la familia de problemas de camino mínimo presentada. El primer método consiste en un algoritmo matheurístico, mientras que el segundo se basa en dos Relajaciones Lagrangianas del problema.

El capítulo 4 contiene una extensa experiencia computacional para validar los resultados presentados en los capítulos anteriores. Además, el capítulo incluye la creación de instancias para el problema del ATFM que han sido liberadas para su libre disposición.

Finalmente, en el capítulo 5 resumimos las principales conclusiones y contribuciones realizadas en la tesis. También se discuten las futuras líneas de investigación que este trabajo abre.

1

INTRODUCTION

One of the challenges that the flight industry is currently facing is air traffic saturation. In short, a saturation of air traffic occurs when the number of flights scheduled to fly a concrete region of the airspace or airport surpasses the maximum quantity allowed by security restrictions. This situation mainly arises because of the current high-traffic demand, and short-term factors like weather, which negatively affect flying conditions.

The current practice to tackle this problem is to modify some flight plans a few hours in advance to their execution. That way, flights can proceed at the expense of some of them having some delay at the departure/landing or having to fly a longer route. These modifications are included in what is denoted as Air Traffic Flow Management (ATFM) and are proposed by the operator of the ATFM system (central decision-maker).

As air traffic involves millions of flights along one year, the cost associated with this problem reaches billions, making crucial how decisions are made to tackle it. In this respect, this thesis, continuing a research line originated some decades ago, addresses the ATFM problem from a mathematical optimization perspective. The basic question to answer, which becomes the cornerstone around which the mathematical model is built, is: How to reschedule flights minimizing costs, so capacity limits in airports and the airspace are respected at all times?

Attempting to answer this question is what motivated the two objectives with which this thesis started:

1. Contributing with an Integer Programming (IP) formulation that solved some limitations of the current state of the art for the decisions involved at intermediate points of the flight route.

2. Proposing solution methods that allow tackling ATFM instances of the size found in practice in acceptable computational times for the industry.

Working on these objectives led us to incorporate two new ones:

3. Extending the results obtained for the ATFM problem to a broader family of scheduling problems. Particularly, to tackle some type of project scheduling problems.

4. Creating a collection of ATFM instances using publicly available sources that contributed to establishing a benchmark data set. That way, other authors can easily test their proposals and reproduce others' work.

In the thesis, the first objective is tackled in Chapter 2. We start this chapter introducing the reader to the problem of air traffic saturation, its consequences and the mechanisms that exist to deal with it. The chapter also contains a review of the most recent works that cope with the ATFM problem from a mathematical optimization point of view. The works closer to the approach developed in this thesis are analyzed and discussed in more detail. Concretely, those of Bertsimas and Patterson [16]; Bertsimas, Lulli, and Odoni [20]; and Agustín, Alonso-Ayuso, Escudero, and Pizarro [4].

After that introduction, in Chapter 2 we present our modeling proposals for the ATFM problem: two new and equivalent 0-1 mathematical programming formulations. The first formulation is obtained by modifying the variables' definition employed in Agustín et al. [4]. In that work, the authors formulate each flight route as a sequence of arcs and use the following binary variables: $x^t_{f,m,n}$. Variable $x^t_{f,m,n}$ is equal to one if, in the solution for flight f, arc (m, n) becomes part of the route (space information), and geographical point/node n has been reached *by* time t (time information). Note that *by time* means at time t or earlier, making of the variable a step variable. In our proposal, we define variables so they also include time information about geographical point/node m, and they are formulated as impulse variables (*at* time instead of *by* time). Concretely, our variables are of the form: $z^{t_1,t_2}_{f,m,n}$. These two modifications in the variables lead to: 1) A better modeling of the flight routes, which results into solutions with less abrupt modifications with respect to the original flight plans, 2) The possi-

bility of including non-linear cost while keeping the model linear, and 3) Most of the constraints defining facets of the polytope.

The second IP formulation exploits the latter condition to formulate the problem not as a general combinatorial one, but as a shortest path problem in multiple networks with limited shared resources. In this second formulation, each network is associated with an aircraft and its potential schedule modifications. Networks have the form of time-expanded or 4D-networks. That is, the nodes of the networks combine space (geographical point) and time information (time at which the point is reached). The shortest path structure of the second formulation enlarges the range of solving strategies, which facilitates to address the second objective of the thesis in the next chapter. Chapter 2 concludes with a novel methodology based on a graph of conflicts to detect, beforehand, routes that will be part of the optimal solution.

In Chapter 3 we tackle the second and third objective listed for the thesis. In the chapter, generalizing some of the results obtained in Chapter 2, we start introducing a new family of shortest path problems that, to the best of our knowledge, has not been previously investigated in literature. The goal in this problem is to find, for each network within a collection, a path between two given nodes that minimizes the total cost (the sum of costs of the arcs forming the paths), while not exceeding the limits of a set of shared resources. We have denoted this problem as the Shared Resource Constrained Multi-Shortest Path Problem (SRC-MSPP). The mathematical formulation of the SRC-MSPP consists of the classic flow conservation constraints for each network, and a collection of capacity constraints that limit the resource consumption. As discussed in the chapter, the SRC-MSPP resembles the Resource Constrained Shortest Path Problem (RCSPP), but important differences exist between both, e. g., the structure of the solutions or that the RCSPP involves only one network, while the SRC-MSPP multiple of them with arcs from different networks accessing the same resources.

In the chapter we show how to use the SRC-MSPP to solve some types of project scheduling problems. The goal in these problems is to schedule the activities conforming a project is such a way that resource limits, as well as precedence relationships between activities, are respected. The type of scheduling problems that we consider consist of multiple projects with: 1) their activities sequenced in serial, and 2) simultaneously incorporating multiple features such as processing speeds of the activities, lag times, alternative sequences of activities, etc. Note that these features are usually studied in isolation (i. e., not simultaneously) in the project scheduling literature.

In Chapter 3 we also present two different solution methods for the SRC-MSPP. The first method is a matheuristic algorithm designed to take advantage of the modern computer architecture with multiple cores. The algorithm consists of three phases: 1) Generation of feasible solutions, 2) Combination of solutions, and 3) Solution improvement. In the first phase, the procedure to generate feasible solutions is built around the idea of penalizing the usage of arcs causing resource infeasibility for a given solution, and recomputing the shortest paths. The procedure is repeated until finding a feasible solution or reaching a maximum number of iterations. In the second phase, the solutions obtained in the previous phase are combined by means of an IP solver. The combination is made paying attention to the arcs of the networks, which introduces some neat properties as detailed in the chapter. The last phase of the algorithm tries to improve the solution obtained in the second phase by fixing a large proportion of it and exploring the resulting subregion of the solution space. The exploration is also done using an IP solver.

Respect to the second solution method studied, this is based on two Lagrangian Relaxations of the problem. The first relaxation is obtained after dualizing the capacity constraints limiting the resource consumption. This makes that, for a fixed vector of Lagrangian multipliers, just a collection of shortest path problems has to be solved. The second relaxation is obtained after dualizing a copy of the decision variables. This allows splitting the problem into two: One consisting of a collection of shortest path problems, and the other consisting of multiple multidimensional knapsack problems. In either case, the resulting relaxations allow to compute lower bounds for the problem and assess the quality of solutions obtained by the algorithm. The strength of these bounds is established using some theoretical results in optimization literature.

All the work developed in the thesis is empirically validated in Chapter 4, where a series of computational experiments are conducted. For the experiments, we created (last objective of the thesis) a group of ATFM instances using publicly available sources. The instances, which comprise different sizes and difficulties, have been released for free disposal to permit other authors to easily test their proposals and reproduce others' work. Respect to the goals with the experiments, these include: Obtaining solutions for the problem using IP solvers, studying the performance of the solution methods proposed analyzing their insights, and validating the flight plans obtained with the formulations presented.

Finally, in Chapter 5 we close the thesis summarizing the main conclusions and contributions accomplished, as well as exposing the future lines of research that this work opens.

2

AIR TRAFFIC FLOW MANAGEMENT PROBLEM

This chapter introduces and motivates the core problem of this thesis: The problem of air traffic saturation and how to deal with it at a tactical decision level by using mathematical optimization. A review of how the problem has been addressed so far in optimization literature is also presented. The review includes a more extensive discussion of those works in which this thesis is inspired. Their evolution and characteristics are compared, so the benefits of the proposal we present later on this chapter can be better assessed.

Respect to the latter, we introduce two new and equivalent 0-1 mathematical formulations for the Air Traffic Flow Management Rerouting Problem. The second one, derived from the former, is based on 4D-networks, which allows us to consider the problem not as a general combinatorial one, but as a shortest path problem in multiple networks with limited shared resources. This fact introduces several neat features, enlarges the range of solving strategies and motivates future chapters of the thesis. Among the decisions considered in the models are ground and air delays, changes in the speed of the aircraft and alternative routes. The proposed models, in comparison with the current state of the art, are shown to be an easy way to model different real complex situations (e. g., a more realistic representation of costs and decisions involved, as well as dynamic sector configuration). Finally, a novel methodology based on a graph of conflicts to detect, beforehand, routes that will be part of the optimal solution, is proposed. All these

methodological contributions have been published in García-Heredia, Alonso-Ayuso, and Molina [53].

2.1 INTRODUCTION

According to the International Air Transport Association [79, 80], different regions of the world, such as Europe or the US, are facing the problem of air traffic saturation, i. e., the number of scheduled flights is sometimes larger than the capacity of the airspace and/or the airports. This gap between capacity and demand mainly arises because of two reasons [44]: 1) A high traffic demand, and 2) Short-term factors like weather, which negatively affect airspace capacities.

To ensure that scheduled flights can proceed while security (capacity) restrictions are met at all time, some changes such as delays or alternative routes have to be proposed to the initial flight plan of some aircraft merely a few hours in advance (tactical decisions). For the case of Europe, the magnitude, consequences and trend of air traffic saturation are exposed in the following figures [45–47, 78]:

1. An average of 30,000 flights per day was recorded in 2018.
2. The current forecast is that air traffic keeps growing: From 11 million flights in 2018 to 16.2 million in 2040. Under this scenario, it is estimated that 1.5 million flights (accounting for 8% of the demand) will not be accommodated in 2040, meaning that 160 million passengers will not be able to fly.
3. In 2018, there was an average departure delay of 14.7 minutes per flight, a 17% more than in 2017.
4. The number of flights arriving within 15 minutes of their scheduled time decreased to 75.7% in 2018, a 3.9% less respect to 2017.
5. In 2018, while air traffic increased by 3.8% over 2017, en-route delays more than doubled (104%), going from 0.88 to 1.74 minutes per flight. The main two causes of these delays were attributed to capacity (37.4%) and weather (25.4%) factors.
6. Every year, there is an economic cost of billions of euros due to these problems. For example, in 2012 a cost of €5.2 billion was reported.

To mitigate these problems, Air Traffic Management (ATM) is currently organized on three decision levels: Strategic, tactical and operational [44]. The strategic level involves the scheduling of flights that will take place in a few months' time. The goal is to maximize the available capacity in order to cope with the projected demand. The ATM part in charge of this is called Airspace Management. The tactical decision level, which comprises the scope of this thesis, encompasses measures taken hours (up to one day) prior to the flight plan execution. During this phase, the traffic demand for the day is analyzed and compared to the predicted available capacity. The initial plan, developed during the strategic phase, is adjusted accordingly, resulting in the Air Traffic Daily Plan. The goal of this plan is to guarantee a safe, orderly, and expeditious flow of traffic while minimizing the impact of the decisions made. The ATM part in charge of this level is referred to as Air Traffic Flow Management (ATFM). Finally, the operational level consists of the decisions made during the flight plan execution. Decisions at this level are made by air traffic controllers and are mainly focused on: 1) Guaranteeing that the plans developed in the previous phases are fulfilled, making modifications if any disruption occurs, and 2) Collision avoidance, that is, ensuring a minimum separation between aircraft during the flight plan. The ATM part in charge of this level is called Air Traffic Control.

In Europe, for example, ATM is the responsibility of EUROCONTROL (European Organization for the Safety of Air Navigation), which plays the role of central network manager. Particularly, the Network Manager Operations Centre is in charge of ATFM operations.

The relevance of ATM in reducing air traffic congestion is such that, since early 2000, large-scale projects have been developed to enhance it. For example, in 2004, the European Commission launched the Single European Sky (SES), an initiative to improve the way Europe's airspace is managed. Its purpose is to reform the European ATM so as to ensure that future traffic demand can be (safely) met while reducing costs and improving environmental performance. Similarly, in 2007, the Federal Aviation Administration (FAA) started in the US the Next Generation Air Transportation System (NextGen), a collection of initiatives aiming to transform the US National Airspace System (NAS) to increase safety and efficiency.

2.2 LITERATURE REVIEW

During the last decades, ATM problems have attracted a lot of attention from academic research. Not only because of its relevance to the industry, but also because of its complexity: the problem involves thousands of flights, multiple decisions to be made and high interaction between them, leading to propagation effects in the underlying network. All this has motivated researches to address the problem from a mathematical optimization perspective.

As previously mentioned, this thesis copes with the problem at its tactical decision level (ATFM). A literature review on the topic until 2010 can be found in Agustín, Alonso-Ayuso, Escudero, and Pizarro [3]. There it is exposed the evolution of this problem in the literature over the last years. From the Single-Airport Ground-Holding Problem (SAGHP), where the goal is to coordinate departures and arrivals for one single airport; to the most modern version: the Air Traffic Flow Management Problem (ATFMP), where the airspace and multiple airports are considered at the same time.

Among the works prior to 2010, we highlight those of Bertsimas and Patterson [16] and Lulli and Odoni [97]. The former is one of the most influential papers that has been published about ATFM. The 0-1 mathematical optimization model proposed in that paper, based on the so-called step variables, considers: 1) Capacity limits in the airspace and airports, 2) Continued flights performed by the same aircraft, and 3) Decisions about assigning ground and air delays at a minimum cost. The proposed formulation is so tight that most of the time it produces an integer solution when the problem is relaxed. For their part, Lulli and Odoni [97], propose a model that obtains a *fair* distribution of the delays among the different airline companies, ensuring that not all of the delays are transferred to only one company or one type of flight.

Since 2010, several new proposals have appeared in literature. In Bertsimas, Lulli, and Odoni [20], the authors extend the Bertsimas and Patterson [16] model by considering the usage of alternative routes. These modifications result in the so-called Air Traffic Flow Management Rerouting Problem (ATFMRP). Likewise, Agustín, Alonso-Ayuso, Escudero, and Pizarro [4] also extend the Bertsimas and Patterson [16] model, but, contrary to Bertsimas et al. [20], their variables are formulated based on the flight network arcs instead of the nodes. This model provides an improvement with respect to the previous ones: besides rerouting and air delays, increases in the aircraft's speed, as well as flight cancellations, are explicitly formulated and penalized. The computational experi-

ence presented shows that the majority of the problems can be solved in the root of the Branch and Bound tree within a few minutes.

Balakrishnan and Chandran [11] formulate a model considering the same possible flight modifications that Agustín et al. [4], but using a different variable definition. In particular, each decision variable corresponds to the usage (or not) of a candidate time-space flight trajectory. This considerably reduces the number of constraints in the problem, but at the expense of exponentially increasing the number of variables. Because of this, the authors solve the problem through column generation in combination with a variable rounding heuristic to generate integer solutions.

Ozgur and Cavcar [103] develop a model that, instead of considering airspace capacities as the previous ones, it directly focuses on aircraft separation (operational level-wise). This results in a difficult-to-solve model where simplifications such as no flight connections or changes of speed are assumed, making that the only decisions considered are about ground holding time.

In Jovanović, Tošić, Čangalović, and Stanojević [83], opposite to the previous works where the models were formulated from a centralized point of view (the ATFM network manager is the decision-maker), the authors propose a model for a collaborative-decision-making (CDM) process between airlines and the network manager. Concretely, the authors formulate a bi-level optimization model that establishes tolls for the usage of each air sector (disjoint regions in which the airspace is divided), so in the end, airlines are prone to choose those routes which are optimal from the network manager perspective. Note that this idea of CDM in ATFM is starting to being studied by some authors, but it is still far from being the main line of research nowadays.

Akgunduz, Jaumard, and Moeini [8] propose a model that tries to combine tactical and operational decision levels by incorporating collision avoidance constraints. The resulting model is extremely complicated, even without considering all the complexities at the operational level, so only small instances can be solved. A remarkable characteristic of their proposal is that it formulates the fuel-consumption-rate as a function of speed (which is not constant).

Ivanov, Netjasov, Jovanović, Starita, and Strauss [81] use ground holding to minimize the effect of propagated delays and improve airport slot adherence, i. e., they choose, among the set of optimal solutions, that which requires fewer changes with respect to the original flight plan.

Diao and Chen [38] propose, instead of using airspace constraints based on sectors, using constraints based on airways, limiting one aircraft per airway. The authors claim

that this idea could improve the capacity of the airspace while ensuring safe separation. They solve the problem through a Dantzig-Wolfe decomposition.

Xiao, Cai, and Abbass [115] develop a multi-objective optimization model aiming at minimizing delays and the workload of air traffic controllers. A heuristic algorithm is introduced to solve the problem.

Another multi-objective optimization model is proposed in Dal Sasso, Fomeni, Lulli, and Zografos [34], but this time to consider different stakeholders' preferences. Their model gets to include Airspace Users' priorities, allowing airlines to decide how to distribute their allocated delays among their flights. In the computational experience, they obtain the Pareto frontier of non-dominated solutions so these can be discussed afterward in a CDM process.

The previous work is later continued in Dal Sasso, Fomeni, Lulli, and Zografos [35]. Among the novelties included on it are: 1) Mechanisms to collect priorities so the airlines do not have to reveal their cost structure, and 2) A simulated annealing algorithm to solve the problem.

A different way in which the ATFMP has been addressed is through multicommodity flow networks, i. e., networks where each flight represents a commodity. Among the research carried out in this area, we highlight the works by Bertsimas and Patterson [17], Chen, Hu, Zhang, Yin, and Han [29], Chen, Cao, and Sun [31], and Wei, Cao, and Sun [113]. The difference between this approach and the previous works is that the usage of commodities does not specify to which particular flights are assigned the proposed modifications in the flight plans. Thus, a subsequent, normally complex procedure is needed to reconstruct the flight routes. Furthermore, multicommodity models seem to be more difficult to solve than those based on modifying concrete flights, being necessary to employ techniques such as Lagrangian Relaxation.

Note that all the previous works, as well as this thesis, tackle the deterministic version of the ATFM problem, that is, that in which no source of uncertainty is considered when obtaining the final flight plan. This standpoint, although dominant in the literature, is not the only one.

Agustín, Alonso-Ayuso, Escudero, and Pizarro [5] extend the work in Agustín et al. [4] considering, by means of scenario trees, uncertainty in the capacities of airports and air sectors. Instances with about 400 flights and up to 48 scenarios are solved.

Balakrishnan and Chandran [11] also deal with uncertainty in the capacity of sectors and airports, solving cases with 17,500 flights and up to 25 scenarios.

Chang, Solak, Clarke, and Johnson [27] develop a two-stage formulation for the ATFM and a heuristic algorithm to solve it. The authors, instead of considering multiple sectors (like in deterministic versions) and hundreds of flights (like in Agustín et al. [5]), they consider only one sector and up to 52 flights. However, they solve problems with more than 30,000 scenarios.

Chen, Chen, and Sun [30] model uncertainty using chance constraints. They defend this approach as a way to handle big instances of the problem while avoiding the large formulations arising when using scenario trees, or the too conservative solutions resulting from robust optimization. A computational experience solving instances with 3,050 flights is reported.

Corolli, Lulli, Ntaimo, and Venkatachalam [33] develop two different two-stage formulations for the ATFM when considering uncertainty in the capacities. The difference between the formulations is the type of decisions allowed at each stage. As a characteristic of the models, air delays are only considered in the contiguous sectors to the landing airport of each flight. To solve the problem, they develop an algorithm based on relaxing the integrality constraints of the second-stage variables and introducing them when the solution is fractional. Instances with 9 scenarios and almost 3,200 flights are solved.

To conclude this section devoted to literature review, we point out that recently, in Shone, Glazebrook, and Zografos [106] most of the works related to stochastic modeling in air traffic management have been summarized, including those specific to stochastic optimization in ATFM.

2.3 PROBLEM DESCRIPTION

Now, for the sake of future discussion, we formally define the ATFM problem. Let us consider a given set of airports $\mathcal{K}$, an airspace divided in a set $\mathcal{J}$ of disjoint air sectors and a set of flights $\mathcal{F}$ departing from an airport, traversing different air sectors, and arriving at an airport in the system. The planning horizon considered at this level is that one day, following the most common approach in literature, is discretized into a set of time periods (usually 5 to 15 minutes in length). Sectors, as well as airports, have a limited capacity, i. e., security restrictions limit the number of flights that can use these elements at each time period. When the capacity required to execute the original flight plans is greater than available, e. g., due to weather conditions or a peak in

traffic demand, the operator of the ATFM system (central decision-maker) must propose some changes to the original plans, while minimizing the global costs derived from these. An important feature within this problem is the existence of continued flights, i. e., situations in which a flight must wait to take off until a previous one has landed, usually because both are operated by the same aircraft. This fact causes an effect which propagates decisions to the entire network of flights.

A flight plan determines the route and time scheduled for each flight. That is, the airports and waypoints (air traffic control points at the boundary or inside each sector) that a flight must cross, as well as the scheduled time to pass through each of them. This definition leads us to represent the set of all possible routes of each flight $f \in \mathcal{F}$ by a directed graph $\mathcal{G}_f = (\mathcal{N}_f, \mathcal{A}_f)$, where $\mathcal{N}_f$ is the set of nodes (airports and waypoints), and $\mathcal{A}_f$ the set of arcs defining the scheduled and potential alternative routes.

For the sake of illustration, let us consider the example depicted in Figure 1.

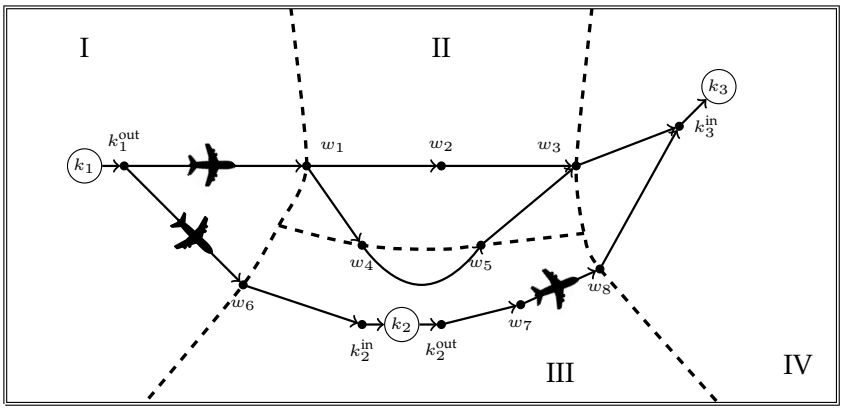


Figure 1: Illustrative example with three airports and four air sectors.

There are four air sectors (I, II, III and IV), three airports (k_1, k_2 and k_3) and eight waypoints (w_1, …, w_8). Additionally, for each airport k_i we consider two boundary nodes, k_i^{in} and k_i^{out}, representing the entrance to and the exit from the airport k_i, respectively. This example shows three flights, $f = 1$ departing from k_1 and going to k_3, $f = 2$ from k_1 to k_2, and $f = 3$ from k_2 to k_3. Note that for the former, two alternative routes have been considered. Thus, $\mathcal{G}_1 = (\mathcal{N}_1, \mathcal{A}_1)$ is defined as follows:

$$\begin{aligned}\mathcal{N}_1 =& \{k_1, k_1^{\text{out}}, w_1, w_2, w_3, w_4, w_5, k_3^{\text{in}}, k_3\},\\ \mathcal{A}_1 =& \big\{(k_1, k_1^{\text{out}}), (k_1^{\text{out}}, w_1), (w_1, w_2), (w_1, w_4),\\ & (w_2, w_3), (w_3, k_3^{\text{in}}), (w_4, w_5), (w_5, w_3), (k_3^{\text{in}}, k_3)\big\}.\end{aligned} \tag{2.1}$$

Each flight has a time period assigned for taking off from its departure airport and a scheduled number of time periods for traversing each arc in the graph. Using this information, it is possible to obtain the scheduled arrival time at each node in the route (including the airport).

As previously said, if when using the scheduled/original flight plan for all flights in the network, the capacity at an airport or air sector is violated, then, decisions to modify some of the flight plans have to be considered. For each flight f, the following decisions can be made: 1) Assigning ground delays, 2) Making changes in the speed of the aircraft while it traverses one or more arcs in its route, and 3) Selecting an alternative route. In order to consider changes in the speed, let us denote by $\ell_{f,m,n}$ the scheduled number of time periods that flight f spends traversing arc (m,n). Then, delays and increases in the speed can be formulated by defining $\overline{\ell}_{f,m,n}$ and $\underline{\ell}_{f,m,n}$, upper and lower bounds, respectively, on the number of time periods that flight f can spend traversing arc (m,n), for $(m,n) \in \mathcal{A}_f$ ($\underline{\ell}_{f,m,n} \leq \ell_{f,m,n} \leq \overline{\ell}_{f,m,n}$).

The modification of the flight plans can combine ground delays, speed changes and rerouting, such that a flight may be affected by all of these measures simultaneously. Each decision has a cost associated with it and the goal of the problem is to obtain a feasible flight plan for each flight at a minimum cost while respecting the capacity limits in airports and sectors. Note that, since continued flights are operated by the same aircraft, all decisions concerning a continued flight can also affect its following flight. Then, if a flight is delayed, its following flight could suffer some delay. Therefore, variables that account for the availability of the aircraft in the case of continued flights must be considered.

2.4 PREVIOUS MODELS

The mathematical formulation that we develop in this thesis is a continuation of the research line established by Bertsimas and Patterson [16], Bertsimas et al. [20], and Agustín et al. [4]. Before presenting our proposal in Section 2.5, we briefly introduce each of these models to show the evolution of the problem and the contributions made in this respect.

2.4.1 *Notation*

Sets

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

$\mathcal{K}$, set of airports.

$\mathcal{J}$, set of air sectors.

$\mathcal{F}$, set of flights.

$\mathcal{Q} = \{(f', f)\}$, set of continued flights such that $f' \in \mathcal{F}$ is continued by flight $f \in \mathcal{F}$.

$\mathcal{N}_f$, set of nodes that define all the available routes for flight $f \in \mathcal{F}$.

$\mathcal{A}_f = \big\{(m, n) \,|\, m, n \in \mathcal{N}_f\big\}$, set of arcs that define all the available routes, i. e., traffic sequencing, for flight $f \in \mathcal{F}$. The subset $\mathcal{A}_f^* \subseteq \mathcal{A}_f$ includes the arcs that define the scheduled route for flight f.

$\mathcal{N}^j$, set of nodes that belong to sector $j \in \mathcal{J}$.

$\mathcal{A}_f^j = \{(m, n) \in \mathcal{A}_f |\, m, n \in \mathcal{N}_f \cap \mathcal{N}^j\}$, set of arcs for flight $f \in \mathcal{F}$ that belong to sector $j \in \mathcal{J}$.

$\Gamma_f^-(n) = \{m \,|\, (m, n) \in \mathcal{A}_f\}$, set of nodes such that (m, n) is an in-coming arc of node $n \in \mathcal{N}_f$ for flight $f \in \mathcal{F}$.

$\Gamma_f^+(n) = \{m \,|\, (n, m) \in \mathcal{A}_f\}$, set of nodes such that (n, m) is an out-going arc of node $n \in \mathcal{N}_f$ for flight $f \in \mathcal{F}$.

$\mathcal{T}_f^n$, set of feasible time periods for flight f to arrive at node n, $f \in \mathcal{F}$, $n \in \mathcal{N}_f$. Let us denote by T_f^n the last element in $\mathcal{T}_f^n$.

Parameters

$\tau_f^d \in \mathcal{T}$, scheduled departure time for flight $f \in \mathcal{F}$.

$\tau_f^a \in \mathcal{T}$, scheduled arrival time at its destination for flight $f \in \mathcal{F}$ when it follows its scheduled route.

$k_f^d \in \mathcal{K}$, departure airport for flight $f \in \mathcal{F}$.

k^{out}, airport boundary node for the departure of any flight, $k \in \mathcal{K}$.

$k_f^a \in \mathcal{K}$, arrival airport for flight $f \in \mathcal{F}$.

k^{in}, airport boundary node for the arrival of any flight, $k \in \mathcal{K}$.

$\ell_{f,m,n}$, scheduled travel time (i. e., number of time periods) for flight $f \in \mathcal{F}$ to traverse arc $(m,n) \in \mathcal{A}_f$. Note 1: $\ell_{f,k_f^d,k_f^{d,\text{out}}} = 0$ and $\ell_{f,k_f^{a,\text{in}},k_f^a} = 0$. Note 2: $\tau_f^a = \tau_f^d + \sum_{(m,n)\in\mathcal{A}_f^*} \ell_{f,m,n}$.

$\overline{\ell}_{f,m,n}$ and $\underline{\ell}_{f,m,n}$, maximum and minimum travel time (i. e., number of time periods) that is allowed for flight $f \in \mathcal{F}$ to traverse arc $(m,n) \in \mathcal{A}_f$, respectively. Note: $\underline{\ell}_{f,m,n} \leq \ell_{f,m,n} \leq \overline{\ell}_{f,m,n}$, for every arc in $\mathcal{A}_f$. In particular, $\underline{\ell}_{f,k_f^d,k_f^{d,\text{out}}} = \ell_{f,k_f^d,k_f^{d,\text{out}}} = \overline{\ell}_{f,k_f^d,k_f^{d,\text{out}}} = 0$ and $\underline{\ell}_{f,k_f^{a,\text{in}},k_f^a} = \ell_{f,k_f^{a,\text{in}},k_f^a} = \overline{\ell}_{f,k_f^{a,\text{in}},k_f^a} = 0$.

$\tau_{f',f}$, turnaround time, i. e., time needed for preparing flight f after the arrival of f', $(f',f) \in \mathcal{Q}$. Note: it is assumed that $\tau_{f'}^a + \tau_{f',f} \leq \tau_f^d$.

$C_k^{t,\text{dep}}$, departure capacity of airport $k \in \mathcal{K}$ at time $t \in \mathcal{T}$.

$C_k^{t,\text{arr}}$, arrival capacity of airport $k \in \mathcal{K}$ at time $t \in \mathcal{T}$.

C_k^t, joint departure and arrival capacity of airport $k \in \mathcal{K}$ at time $t \in \mathcal{T}$. Note: $C_k^t < C_k^{t,\text{dep}} + C_k^{t,\text{arr}}$.

C_j^t, capacity of sector $j \in \mathcal{J}$ at time $t \in \mathcal{T}$.

2.4.2 *ATFMP by nodes: Bertsimas & Patterson*

As mentioned in the previous section, Bertsimas and Patterson [16] is one of the most influential works in ATFM literature. Proof of that is the number of posterior works which are based on it. The model considers ground and air delays, but not speed increases or the usage of alternatives routes. Furthermore, instead of associating nodes with geographical points like in the example shown in Figure 1 (page 14), it defines one node per sector entrance, so reaching one node means entering the associated sector.

Using this formulation, graph $\mathcal{G}_f = (\mathcal{N}_f, \mathcal{A}_f)$ for flight $f = 1$ in Figure 1, instead of like in (2.1) (page 14), it becomes:

$$\mathcal{N}_1 = \{k_1, I, II, IV, k_3\},$$
$$\mathcal{A}_1 = \big\{(k_1, I), (I, II), (II, IV), (IV, k_3)\big\}.$$

Note that:

- The alternative route that used sector III no longer exists.
- As alternative routes are not considered, each graph $\mathcal{G}_f$ is indeed a directed path where arcs are of the form $(n, n+1) \in \mathcal{A}_f$.
- As said, there is a one-to-one correspondence between each node in $\mathcal{N}_f$ and the airports and sectors that flight f has to cross in its route.

The latter implies that for a given sector $j \in \mathcal{J}$ in f's route, j uniquely identifies one of the route nodes. Similarly, given $j \in \mathcal{J}$ in f's route, the immediately subsequent node in the route can be identified by $j + 1$, which corresponds to the contiguous sector of j. This fact will be used in the formulation of the model.

Variables

$v^t_{f,n} = 1$, if flight f arrives at node n *by* time t, and 0 otherwise, $\forall f \in \mathcal{F}, n \in \mathcal{N}_f, t \in \mathcal{T}^n_f$.

Note that the variables (due to the usage of "by" in the definition) are step variables, that is, they indicate if a flight f has arrived at node n at time t or earlier. Thus, if $v^t_{f,n} = 1$, then $v^{t+1}_{f,n} = \ldots = v^{T^n_f}_{f,n} = 1$.

Example ATFM

For the sake of illustration when discussing the different models presented in the thesis, we elaborated an example based on the situation depicted in Figure 1 (page 14) for flight $f = 1$. Consider that the scheduled departure time for that flight is $\tau^d_1 = 1$, and that is possible to delay its departure for at most two extra periods. That is, the flight cannot depart later than $t = 3$. Consider also that the scheduled and maximum travel times for each arc of the route are those shown in Table 1.

Noting that $\mathcal{T}^{k_1}_1$ contains the possible departure times of the flight, i. e., $\mathcal{T}^{k_1}_1 = \{1, 2, 3\}$, the rest of the sets $\mathcal{T}^n_1$ can be easily computed using the travel time information in

Table 1: Travel times for $f = 1$ in the example.

	(k_1, I)	(I, II)	(II, IV)	(IV, k_3)
$\ell_{1,m,n}$	0	1	3	2
$\bar{\ell}_{1,m,n}$	0	2	5	2

Table 1. Concretely, given node n and $\mathcal{T}_1^n$, $\mathcal{T}_1^{n+1} = \{\underline{T}_1^n + \ell_{1,n,n+1}, \ldots, T_1^n + \bar{\ell}_{1,n,n+1}\}$, where $\underline{T}_1^n$ is the first time period at which node n can be reached, i. e., $\underline{T}_1^n = \min\{t \in \mathcal{T}_1^n\}$. Recall that $T_f^n = \max\{t \in \mathcal{T}_f^n\}$.

Figure 2 shows a graph representation of the flight route of the example, together with the sectors crossed by the flight, travel times and sets $\mathcal{T}_1^n$. The vertical dashed lines represent the sector limits.

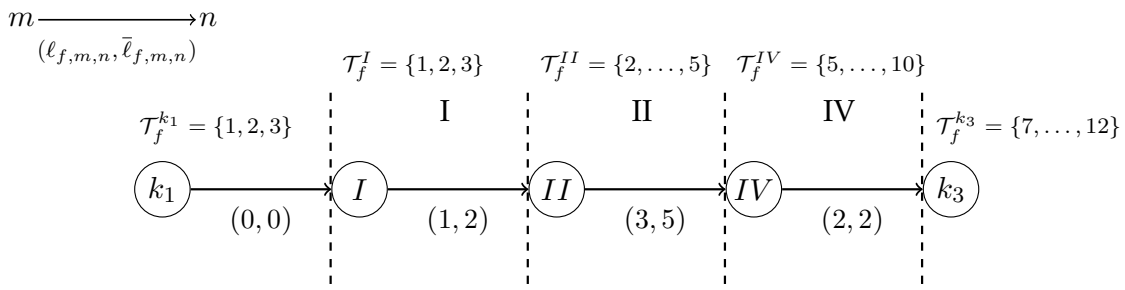


Figure 2: ATFMP by nodes for $f = 1$ in the example.

To conclude the example for this formulation, consider that the new flight plan for $f = 1$ includes the following modifications: One period of departure delay and two periods of air delay when traversing sector II (arc (II, IV)). In that case, decisions variables would take the following values:

$$
\begin{array}{lllll}
v^1_{1,k_1} = 0 & v^1_{1,I} = 0 & v^2_{1,II} = 0 & v^5_{1,IV} = 0 & v^7_{1,k_3} = 0 \\
v^2_{1,k_1} = 1 & v^2_{1,I} = 1 & v^3_{1,II} = 1 & v^6_{1,IV} = 0 & v^8_{1,k_3} = 0 \\
v^3_{1,k_1} = 1 & v^3_{1,I} = 1 & v^4_{1,II} = 1 & v^7_{1,IV} = 0 & v^9_{1,k_3} = 0 \\
 & & v^5_{1,II} = 1 & v^8_{1,IV} = 1 & v^{10}_{1,k_3} = 1 \\
 & & & v^9_{1,IV} = 1 & v^{11}_{1,k_3} = 1 \\
 & & & v^{10}_{1,IV} = 1 & v^{12}_{1,k_3} = 1
\end{array}
$$

Objective function

Deviations from the original flight plan are penalized in the objective function by accounting for ground and air delays. The total ground delay assigned to flight f is computed as the difference between the actual and planned departure time:

$$g_f = \sum_{t \in \mathcal{T}_f^k : k = k_f^d} (t - \tau_f^d)(v_{f,k}^t - v_{f,k}^{t-1}) = \sum_{t \in \mathcal{T}_f^k : k = k_f^d} t(v_{f,k}^t - v_{f,k}^{t-1}) - \tau_f^d.$$

Note that in the sum, all the terms $(v_{f,k}^t - v_{f,k}^{t-1})$ will be equal to zero, but one, which will be equal to 1. Concretely, if the flight departs at time t', $(v_{f,k}^{t'} - v_{f,k}^{t'-1}) = 1$ and $g_f = t' - \tau_f^d$.

Similarly, the delay in air is the difference between the actual and planned arrival time minus the delay due to ground holding:

$$a_f = \sum_{t \in \mathcal{T}_f^k : k = k_f^a} (t - \tau_f^a - g_f)(v_{f,k}^t - v_{f,k}^{t-1}) = \sum_{t \in \mathcal{T}_f^k : k = k_f^a} t(v_{f,k}^t - v_{f,k}^{t-1}) - \tau_f^a - g_f.$$

These two quantities are multiplied by cost coefficients c_f^g and c_f^a, respectively, leading to the following objective function seeking for minimization:

$$\min \sum_{f \in \mathcal{F}} (c_f^g g_f + c_f^a a_f). \tag{2.2}$$

Note that ground delays, as they are safer than air delays, should be preferred, i. e., $c_f^g < c_f^a$.

Capacity constraints

$$\sum_{\substack{f \in \mathcal{F}: \\ \{k_f^d = k, t \in \mathcal{T}_f^k\}}} (v_{f,k}^t - v_{f,k}^{t-1}) \leq C_k^{t,\text{dep}} \quad \forall k \in \mathcal{K}, t \in \mathcal{T}, \tag{2.3}$$

$$\sum_{\substack{f \in \mathcal{F}: \\ \{k_f^a = k, t \in \mathcal{T}_f^k\}}} (v_{f,k}^t - v_{f,k}^{t-1}) \leq C_k^{t,\text{arr}} \quad \forall k \in \mathcal{K}, t \in \mathcal{T}, \tag{2.4}$$

$$\sum_{\substack{f \in \mathcal{F}: \\ \{k_f^a = k, t \in \mathcal{T}_f^k\}}} (v_{f,k}^t - v_{f,k}^{t-1}) + \sum_{\substack{f \in \mathcal{F}: \\ \{k_f^d = k, t \in \mathcal{T}_f^k\}}} (v_{f,k}^t - v_{f,k}^{t-1}) \leq C_k^t \quad \forall k \in \mathcal{K}, t \in \mathcal{T}, \tag{2.5}$$

$$\sum_{f \in \mathcal{F}} (v_{f,j}^t - v_{f,j+1}^t) \leq C_j^t \quad \forall j \in \mathcal{J}, t \in \mathcal{T}, \tag{2.6}$$

Flight structure constraints

$$v_{f,n+1}^{t+\ell_{f,n,n+1}} - v_{f,n}^{t} \leq 0 \quad \forall f \in \mathcal{F}, n \in \mathcal{N}_f \setminus \{k_f^a\}, t \in \mathcal{T}_f^n, \tag{2.7}$$

$$v_{f,k_f^d}^{t} - v_{f',k_{f'}^a}^{t-\tau_{f',f}} \leq 0 \quad \forall (f', f) \in \mathcal{Q}, t \in \mathcal{T}_f^{k_f^d}, \tag{2.8}$$

$$v_{f,n}^{t} - v_{f,n}^{t-1} \geq 0 \quad \forall f \in \mathcal{F}, n \in \mathcal{N}_f, t \in \mathcal{T}_f^n, \tag{2.9}$$

$$v_{f,k_f^d}^{T_f^{k_f^d}} = 1 \quad \forall f \in \mathcal{F}, \tag{2.10}$$

Variables' domain

$$v_{f,n}^{t} \in \{0, 1\} \quad \forall f \in \mathcal{F}, n \in \mathcal{N}_f, t \in \mathcal{T}_f^n. \tag{2.11}$$

In the model, constraints (2.3)-(2.6) guarantee that capacity in airports and air sectors is respected at all times. Constraints (2.7) establish connectivity between the route nodes and ensure that each flight f spends at least the scheduled time ($\ell_{f,n,n+1}$) going from one node to another. Constraints (2.8) are for the connection of continued flights (f', f): flight f cannot depart before flight f' has landed and spent $\tau_{f',f}$ time units at the airport preparing the aircraft. Constraints (2.9) are for time connectivity: they state that if a flight has not arrived at time t to a node of the route, then neither has it arrived before. Finally, constraints (2.10) ensure that all flights take off.

Example ATFM (continuation)

Note that in the model, there are no constraints limiting the maximum number of time periods for going from one node to the subsequent one. Therefore, it is possible to obtain solutions that are feasible from a mathematical point of view (no constraint violation), but not from a physical one (e. g., excessive speed reduction). This is because the authors do not work with maximum travel times ($\bar{\ell}_{f,m,n}$), but just with the latest possible time to reach each node (sets $\mathcal{T}_f^n$). However, sets $\mathcal{T}_f^n$ only control (indirectly) the total delay up to a node, but not where the delay occurs, i. e., all the air delay might be assigned to occur in one arc of the route, which is not feasible from a physical point of view.

To illustrate this point, see that in the example, no constraint forbids spending 6 time periods (instead of just 5 as exposed in Table 1) in arc (II, IV). That is, the following solution is also feasible (changes with respect to the previous one in bold):

$$
\begin{array}{lllll}
v^1_{1,k_1} = 0 & v^1_{1,I} = 0 & v^2_{1,II} = 0 & v^5_{1,IV} = 0 & v^7_{1,k_3} = 0 \\
v^2_{1,k_1} = 1 & v^2_{1,I} = 1 & v^3_{1,II} = 1 & v^6_{1,IV} = 0 & v^8_{1,k_3} = 0 \\
v^3_{1,k_1} = 1 & v^3_{1,I} = 1 & v^4_{1,II} = 1 & v^7_{1,IV} = 0 & v^9_{1,k_3} = 0 \\
 & & v^5_{1,II} = 1 & \boldsymbol{v^8_{1,IV} = 0} & \boldsymbol{v^{10}_{1,k_3} = 0} \\
 & & & v^9_{1,IV} = 1 & v^{11}_{1,k_3} = 1 \\
 & & & v^{10}_{1,IV} = 1 & v^{12}_{1,k_3} = 1
\end{array}
$$

2.4.3 *ATFMRP by nodes: Bertsimas, Lulli & Odoni*

Bertsimas, Lulli, and Odoni [20] extend the previous work by considering the usage of alternative routes and penalizing deviations from the original flight plan more accurately. They maintain the same variables' definition ($v^t_{f,n}$), and the one-to-one relationship between route nodes and sectors.

Example ATFM (continuation)

Let us continue with the example presented before, but this time to illustrate how rerouting would be taken into account with variables $v^t_{f,n}$. The first thing to mention is that due to: i) Defining the variables by nodes instead arcs (unlike, for example, Agustín et al. [4] or this thesis), and ii) Maintaining the one-to-one relationship between nodes and sectors, this new model can only incorporate rerouting in a restricted way. Let see that with the next three cases:

- Case 1: The situation of the example, i. e., that shown in Figure 1 (page 14) for flight $f = 1$, cannot be modeled because there is a reentry[1] to sector II. See detailed explanation after the mathematical formulation.
- Case 2: If in the example, instead of having the alternative route as: $\{(I, II), (II, III), (III, II), (II, IV)\}$, it were $\{(I, II), (II, III), (III, IV)\}$, i. e., without reentry, it still could not be modeled. This is because two sectors adjacent to sector IV would be used. See detailed explanation after the mathematical formulation.

1 Note that reentries are common when working with non-convex sectors.

- Case 3: If the alternative route were $\{(I, III), (III, IV)\}$, then it could be modeled. Note in Figure 1, however, that this alternative route is much longer than those described in the previous two cases.

For the sake of illustration, we will consider that the situation of the example is that described in Case 3. That is, the alternative route is not $\{(I, II), (II, III), (III, II), (II, IV)\}$, but $\{(I, III), (III, IV)\}$. For that situation, the graph representation of the flight routes, jointly with travel times and sets $\mathcal{T}_f^n$ is shown in Figure 3.

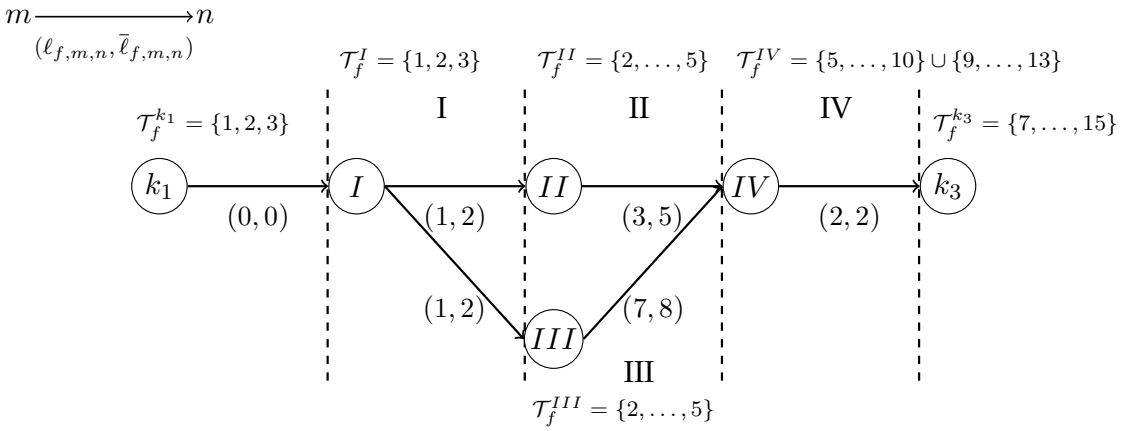


Figure 3: ATFMRP by nodes for Case 3 and $f = 1$ in the example.

Note that set $\mathcal{T}_1^{IV}$ is formed by the union of the possible arrival times to node IV from both routes.

To conclude the example, consider that the solution of the model includes: i) Departing with no delay, ii) Using the alternative route, and iii) Delaying the flight one period when traversing sector III (arc (III, IV)). In that case, all the variables associated with node II would be equal to zero. For the rest, their values would be:

$$
\begin{array}{lllll}
v^1_{1,k_1} = 1 & v^1_{1,I} = 1 & v^2_{1,III} = 1 & v^5_{1,IV} = 0 & v^7_{1,k_3} = 0 \\
v^2_{1,k_1} = 1 & v^2_{1,I} = 1 & v^3_{1,III} = 1 & \vdots & \vdots \\
v^3_{1,k_1} = 1 & v^3_{1,I} = 1 & v^4_{1,III} = 1 & v^9_{1,IV} = 0 & v^{11}_{1,k_3} = 0 \\
 & & v^5_{1,III} = 1 & v^{10}_{1,IV} = 1 & v^{12}_{1,k_3} = 1 \\
 & & & \vdots & \vdots \\
 & & & v^{13}_{1,IV} = 1 & v^{15}_{1,k_3} = 1
\end{array}
$$

Objective function

Respect to the previous work, this model introduces a substantial improvement in the objective function. Delays are penalized in a superlinear way: $g_f^{1+\epsilon_1} + a_f^{1+\epsilon_2}$, where $0 < \epsilon_1 < \epsilon_2 < 1$. Note that with this new form of penalization, in addition to preferring ground to air delays ($\epsilon_1 < \epsilon_2$), a fairer distribution of these is also achieved. It is preferable to delay (either in ground or air) two flights one time period each rather than one flight two periods. Furthermore, as stated in Bertsimas et al. [20], if ϵ_1 and ϵ_2 are closed to zero, the following approximation can be done:

$$g_f^{1+\epsilon_1} + a_f^{1+\epsilon_2} = g_f^{1+\epsilon_1} + a_f^{1+\epsilon_2} + g_f^{1+\epsilon_2} - g_f^{1+\epsilon_2} \cong h_f^{1+\epsilon_2} - (g_f^{1+\epsilon_2} - g_f^{1+\epsilon_1}),$$

where $h_f = g_f + a_f$, is the total delay. Using $h_f^{1+\epsilon_2}$ over $g_f^{1+\epsilon_1} + a_f^{1+\epsilon_2}$ makes it preferable to assign one unit of ground delay to one flight and one unit of air delay to another, than both delays to a single flight. Note that each term is still computed linearly:

$$h_f^{1+\epsilon_2} = \sum_{t \in \mathcal{T}_f^k : k = k_f^d} (t - \tau_f^a)^{1+\epsilon_2} (v_{f,k}^t - v_{f,k}^{t-1}),$$

$$g_f^{1+\epsilon_2} - g_f^{1+\epsilon_1} = \sum_{t \in \mathcal{T}_f^k : k = k_f^d} \left((t - \tau_f^d)^{1+\epsilon_2} - (t - \tau_f^d)^{1+\epsilon_1} \right) (v_{f,k}^t - v_{f,k}^{t-1}).$$

As a result, the objective function becomes:

$$\min \sum_{f \in \mathcal{F}} \left(h_f^{1+\epsilon_2} - (g_f^{1+\epsilon_2} - g_f^{1+\epsilon_1}) \right). \tag{2.12}$$

Capacity constraints

$$\sum_{\substack{f \in \mathcal{F}: \\ \{k_f^d = k, t \in \mathcal{T}_f^k\}}} (v_{f,k}^t - v_{f,k}^{t-1}) \leq C_k^{t,\text{dep}} \quad \forall k \in \mathcal{K}, t \in \mathcal{T}, \tag{2.13}$$

$$\sum_{\substack{f \in \mathcal{F}: \\ \{k_f^a = k, t \in \mathcal{T}_f^k\}}} (v_{f,k}^t - v_{f,k}^{t-1}) \leq C_k^{t,\text{arr}} \quad \forall k \in \mathcal{K}, t \in \mathcal{T}, \tag{2.14}$$

$$\sum_{\substack{f \in \mathcal{F}: \\ \{k_f^a = k, t \in \mathcal{T}_f^k\}}} (v_{f,k}^t - v_{f,k}^{t-1}) + \sum_{\substack{f \in \mathcal{F}: \\ \{k_f^d = k, t \in \mathcal{T}_f^k\}}} (v_{f,k}^t - v_{f,k}^{t-1}) \leq C_k^t \quad \forall k \in \mathcal{K}, t \in \mathcal{T}, \tag{2.15}$$

$$\sum_{f \in \mathcal{F}} \max \left\{ 0, v_{f,j}^t - \sum_{n \in \Gamma_f^+(j)} v_{f,n}^t \right\} \leq C_j^t \quad \forall j \in \mathcal{J}, t \in \mathcal{T}, \tag{2.16}$$

Flight structure constraints

$$v^t_{f,n} - \sum_{m \in \Gamma^-_f(n)} v^{t-\ell_{f,m,n}}_{f,m} \leq 0 \quad \forall f \in \mathcal{F}, n \in \mathcal{N}_f \setminus \{k^d_f\}, t \in \mathcal{T}^n_f, \tag{2.17}$$

$$v^{T^n_f}_{f,n} - \sum_{m \in \Gamma^+_f(n)} v^{T^m_f}_{f,m} \leq 0 \quad \forall f \in \mathcal{F}, n \in \mathcal{N}_f \setminus \{k^a_f\}, \tag{2.18}$$

$$\sum_{m \in \Gamma^+_f(n)} v^{T^m_f}_{f,m} \leq 1 \quad \forall f \in \mathcal{F}, n \in \mathcal{N}_f \setminus \{k^a_f\}, \tag{2.19}$$

$$v^t_{f,k^d_f} - v^{t-\tau_{f',f}}_{f',k^a_{f'}} \leq 0 \quad \forall (f', f) \in \mathcal{Q}, t \in \mathcal{T}^{k^d_f}_f, \tag{2.20}$$

$$v^t_{f,n} - v^{t-1}_{f,n} \geq 0 \quad \forall f \in \mathcal{F}, n \in \mathcal{N}_f, t \in \mathcal{T}^n_f, \tag{2.21}$$

$$v^{T^{k^d_f}_f}_{f,k^d_f} = 1 \quad \forall f \in \mathcal{F}, \tag{2.22}$$

Variables' domain

$$v^t_{f,n} \in \{0, 1\} \quad \forall f \in \mathcal{F}, n \in \mathcal{N}_f, t \in \mathcal{T}^n_f. \tag{2.23}$$

In the model, constraints (2.13)-(2.16) are for the capacity limits. The maximum is required in (2.16) because $v^t_{f,j} - \sum_{n \in \Gamma^+_f(j)} v^t_{f,n}$ is negative whenever flight f arrives at one of the subsequent sectors of j without flying through sector j. Constraints (2.17) stipulate that a flight cannot arrive at node n by time t if it has not arrived at any of its predecessor nodes by time $t - \ell_{f,m,n}$. Constraints (2.18) and (2.19) make flights reach one of their subsequent nodes no later than allowed. Constraints (2.20)-(2.22) are exactly the same as constraints (2.8)-(2.10) in the previous model.

In the paper, the authors also present a series of valid inequalities (cuts) to strengthen the model formulation. And although no comparison data is provided in the paper, the authors claim to have verified the benefits of using the inequalities in a few tests conducted.

Example ATFM (continuation)

Once presented the formulation, it can be explained why the alternative route situations described in Case 1 and 2 cannot be modeled with this formulation. For Case 1 there are two problems:

1. Consider that the alternative route is used and that node II is reached at t_1, node III at t_2, and node II is reached again (in the reentry) at t_3, i. e., $v^{t_1}_{f,II} = 1$, $v^{t_2}_{f,III} = 1$ and $v^{t_3}_{f,II} = 1$, with $t_1 < t_2 < t_3$. Due to the variables' definition, once node III is reached, in constraints (2.16) the term $\sum_{n\in\Gamma^+_1(II)} v^t_{1,n}$ will always be positive for $t > t_2$. Therefore, for $t \geq t_3 > t_2$:

$$v^t_{1,II} - \sum_{n\in\Gamma^+_1(II)} v^t_{1,n} = v^t_{1,II} - (v^t_{1,III} + v^t_{1,IV}) = 1 - 1 - v^t_{1,IV} \leq 0,$$

   that is, when reentry, no capacity consumption will be taken into account.

2. When using the alternative route, both, nodes III and IV become part of the solution, i. e., $v^{T^{III}_f}_{f,III} = v^{T^{IV}_f}_{f,IV} = 1$. However, this would cause constraint (2.19) for node II to be violated: $\sum_{m\in\Gamma^+_f(II)} v^{T^m_f}_{f,m} = v^{T^{III}_f}_{f,III} + v^{T^{IV}_f}_{f,IV} = 2$. Note that these constraints forbid, for a given sector j, using more than one adjacent sector to it.

The latter reason is what also prevents the formulation from being able to model Case 2. Note that for Case 1, making a copy II' of node II such that the alternative route becomes $\{(II, III), (III, II'), (II, IV)\}$, does not solve the problem. Besides not respecting the one-to-one relationship between nodes and sectors, it does not solve the second issue listed.

Note also that the potential problem mentioned for Bertsimas and Patterson [16] about obtaining feasible solutions from a mathematical point of view, but not from a physical one, still exists in this one.

As it will be shown, formulations based on arcs instead of nodes avoid all these issues, which make them more suitable to address the ATFM problem. Furthermore, they also avoid the usage of the *maximum* in the sector capacity constraints (2.16).

2.4.4 *ATFMRP by arcs: Agustín, Alonso-Ayuso, Escudero & Pizarro*

Agustín, Alonso-Ayuso, Escudero, and Pizarro [4] present a model that also extends that of Bertsimas and Patterson [16], but with some improvements respect to the one in Bertsimas et al. [20]. Concretely:

1. Variables are defined by arcs instead of nodes. This turns out to provide: i) A better modeling framework (e. g., it solves the modeling rerouting problems discussed before) and, ii) A finer control of the flight plans modifications. In fact, as exposed

in Agustín [2], the ATFMRP formulated by nodes can be viewed as a particular case of the ATFMRP formulated by arcs.

2. Related to the previous point, as deviations from the original flight plan can be measured more accurately, a more realistic objective function is proposed. Particularly, deviations of the flight plans at intermediate nodes of the route are also considered and penalized.

3. In addition to delays, increases in aircraft's speed are also considered.

4. The model does not associate nodes with sectors, but with geographical points of the airspace. And these geographical points are the ones associated with the boundary or interior of each sector (recall in Figure 1, page 14). This subtle difference allows to: i) Represent flight routes more accurately (more waypoints can conform the routes), and ii) Separate the flight routes from the sector configuration, so the latter can change along the time horizon (details in Subsection 2.5.2).

To handle entry and exiting operations to sectors with this formulation based on arcs, boundary nodes are defined to belong to all the sectors they connect. That is, a node can belong to more than one sector. For example, waypoint w_1 in Figure 1 would belong to sectors I and II. Additionally, airports are also represented by arcs. Particularly, by the departure and landing arcs (k, k^{out}) and (k^{in}, k), respectively. See details in the example later on.

Additional sets

$\mathcal{T}_{*,f}^{m,n}$, set of feasible time periods for flight $f \in \mathcal{F}$ to arrive at node n through arc $(m,n) \in \mathcal{A}_f$. Let us denote by $T_{*,f}^{m,n}$ the last element in $\mathcal{T}_{*,f}^{m,n}$.

$\widetilde{\mathcal{T}}_f^{m,n}$, smallest interval containing $\mathcal{T}_{*,f}^{m,n}$. The distinction between the two sets is because, due to the rerouting option, $\mathcal{T}_{*,f}^{m,n}$ may be formed of no consecutive elements (e. g., $\mathcal{T}_{*,f}^{m,n} = \{5,7,9\}$ and $\widetilde{\mathcal{T}}_f^{m,n} = \{5,6,7,8,9\}$).

$\widetilde{\mathcal{T}}_f^{n}$, smallest interval containing $\mathcal{T}_f^{n}$.

$\mathcal{N}_f^{j+} = \{n |\ n \in \mathcal{N}_f^j,\ \Gamma_f^-(n) \cap \mathcal{N}_f^j = \emptyset\}$, set of in-coming nodes to sector $j \in \mathcal{J}$ in the route of flight $f \in \mathcal{F}$.

$\mathcal{N}_f^{j-} = \{n |\ n \in \mathcal{N}_f^j,\ \Gamma_f^+(n) \cap \mathcal{N}_f^j = \emptyset\}$, set of out-going nodes from sector $j \in \mathcal{J}$ in the route of flight $f \in \mathcal{F}$.

Additional parameters

$c_f^{m,n}$, cost of flight f for using arc (m,n), $f \in \mathcal{F}$, $(m,n) \in \mathcal{A}_f$.

$c_{f,t}^{n}$, cost of flight f for arriving at node n at time period t, $f \in \mathcal{F}$, $n \in \mathcal{N}_f \setminus \{k_f^d\}$, $t \in \widetilde{\mathcal{T}}_f^n$.

c_f^{n+}, c_f^{n-}, costs of flight f for using any arc heading to node n more or less time periods than scheduled, respectively, $f \in \mathcal{F}$, $n \in \mathcal{N}_f \setminus \{k_f^d, k_f^{a,\text{in}}, k_f^a\}$.

c_f^{a+}, c_f^{a-}, costs of flight f for flying more or less time periods than scheduled, respectively, $f \in \mathcal{F}$.

Variables

$x_{f,m,n}^t = 1$, if flight f arrives at node n using arc (m,n) *by* time t, and 0 otherwise, $\forall f \in \mathcal{F}, (m,n) \in \mathcal{A}_f, t \in \widetilde{\mathcal{T}}_f^{m,n}$.

ξ_f^{n+}, ξ_f^{n-}, real non-negative variables that measure, respectively, the positive and negative difference between the time spent by flight f traversing arc (m,n) and the scheduled one ($\ell_{f,m,n}$), $f \in \mathcal{F}$, $(m,n) \in \mathcal{A}_f$.

η_f^+, η_f^-, real non-negative variables that measure, respectively, the positive and negative difference between the planned and actual trip length for flight $f \in \mathcal{F}$.

Note that the $x_{f,m,n}^t$ variables are still formulated as step variables. Note also the following relationship between these variables and the $v_{f,n}^t$ ones:

$$v_{f,n}^t = \sum_{m \in \Gamma_f^-(n)} x_{f,m,n}^t \quad \forall f \in \mathcal{F}, n \in \mathcal{N}_f, t \in \mathcal{T}_f^n.$$

Example ATFM (continuation)

Let us analyze how the example discussed for the previous models would be formulated with these new variables. First of all, notice that now $\mathcal{G}_1$ is as in (2.1) (page 14). Figure 4 shows a graphical representation of it together with information about the travel times[2]. Note that in the figure, the vertical dashed lines divide each boundary node in two, meaning that each node belongs to both sectors it connects. Note also that, as previously said, airports are now represented by departure and landing arcs.

2 For illustration purposes, the travel times are different from the previous examples.

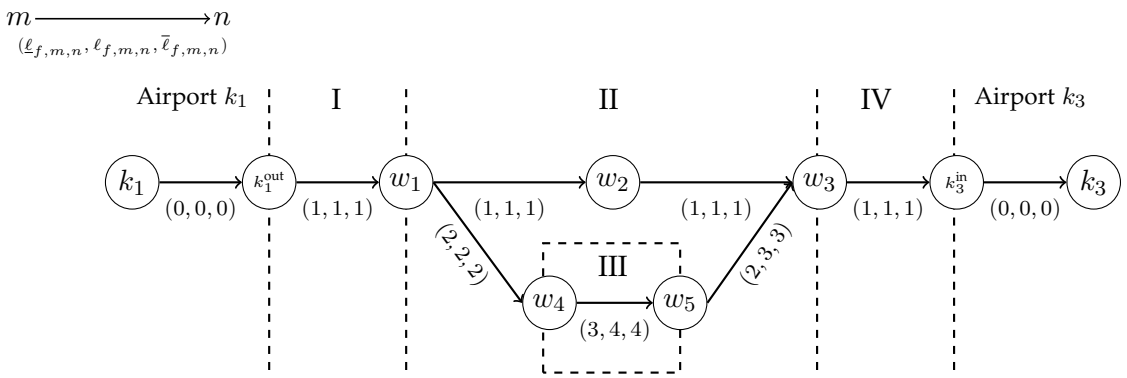


Figure 4: ATFMRP by arcs for $f = 1$ in the example.

With the information in the figure, and remembering that the flight was scheduled to depart at $\tau_1^d = 1$ and not later than $t = 3$, we can obtain the different sets required:

Sets $\mathcal{N}^j$, $\mathcal{N}^{j+}$ and $\mathcal{N}^{j-}$

Notice how a node belong to all the sectors it connects.

$$\begin{aligned}
\mathcal{N}_1^I &= \{k_1^{\text{out}}, w_1\} & \mathcal{N}_1^{I+} &= \{k_1^{\text{out}}\} & \mathcal{N}_1^{I-} &= \{w_1\} \\
\mathcal{N}_1^{II} &= \{w_1, w_2, w_3, w_4, w_5\} & \mathcal{N}_1^{II+} &= \{w_1, w_5\} & \mathcal{N}_1^{II-} &= \{w_3, w_4\} \\
\mathcal{N}_1^{III} &= \{w_4, w_5\} & \mathcal{N}_1^{III+} &= \{w_4\} & \mathcal{N}_1^{III-} &= \{w_5\} \\
\mathcal{N}_1^{IV} &= \{w_3, k_3^{\text{in}}\} & \mathcal{N}_1^{IV+} &= \{w_3\} & \mathcal{N}_1^{IV-} &= \{k_3^{\text{in}}\}
\end{aligned}$$

Sets $\mathcal{A}^j$

$$\begin{aligned}
\mathcal{A}_1^I &= \{(k_1^{\text{out}}, w_1)\} & \mathcal{A}_1^{II} &= \{(w_1, w_2), (w_2, w_3), (w_1, w_4), (w_5, w_3)\} \\
\mathcal{A}_1^{III} &= \{(w_4, w_5)\} & \mathcal{A}_1^{IV} &= \{(w_3, k_3^{\text{in}})\}
\end{aligned}$$

Sets $\mathcal{T}_f^n$ and $\widetilde{\mathcal{T}}_f^n$

For the sake of conciseness, we just show these sets for nodes k_1, w_2, w_5, w_3 and k_3^{in}. Note that for w_3 and k_3^{in}, $\mathcal{T}_f^n$ and $\widetilde{\mathcal{T}}_f^n$ are different.

$$\begin{aligned}
\mathcal{T}_1^{k_1} &= \{1, 2, 3\} & \widetilde{\mathcal{T}}_1^{k_1} &= \{1, 2, 3\} \\
\mathcal{T}_1^{w_2} &= \{3, 4, 5\} & \widetilde{\mathcal{T}}_1^{w_2} &= \{3, 4, 5\} \\
\mathcal{T}_1^{w_5} &= \{7, 8, 9, 10\} & \widetilde{\mathcal{T}}_1^{w_5} &= \{7, 8, 9, 10\} \\
\mathcal{T}_1^{w_3} &= \{4, 5, 6\} \cup \{9, \ldots, 13\} & \widetilde{\mathcal{T}}_1^{w_3} &= \{4, \ldots, 13\} \\
\mathcal{T}_1^{k_3^{\text{in}}} &= \{5, 6, 7\} \cup \{10, \ldots, 14\} & \widetilde{\mathcal{T}}_1^{k_3^{\text{in}}} &= \{5, \ldots, 14\}
\end{aligned}$$

Sets $\mathcal{T}_{*,f}^{m,n}$ and $\widetilde{\mathcal{T}}_{f}^{m,n}$

Again, the sets are just shown for a few nodes. Note that for the example, $\mathcal{T}_{*,f}^{m,n}$ and $\widetilde{\mathcal{T}}_{f}^{m,n}$ are only different in the last case.

$$\mathcal{T}_{*,1}^{w_1,w_2} = \{3,4,5\} \qquad \widetilde{\mathcal{T}}_{1}^{w_1,w_2} = \{3,4,5\}$$
$$\mathcal{T}_{*,1}^{w_2,w_3} = \{4,5,6\} \qquad \widetilde{\mathcal{T}}_{1}^{w_2,w_3} = \{4,5,6\}$$
$$\mathcal{T}_{*,1}^{w_4,w_5} = \{7,8,9,10\} \qquad \widetilde{\mathcal{T}}_{1}^{w_4,w_5} = \{7,8,9,10\}$$
$$\mathcal{T}_{*,1}^{w_5,w_3} = \{9,\ldots,13\} \qquad \widetilde{\mathcal{T}}_{1}^{w_5,w_3} = \{9,\ldots,13\}$$
$$\mathcal{T}_{*,1}^{w_3,k_3^{\text{in}}} = \{5,6,7\} \cup \{10,\ldots,14\} \qquad \widetilde{\mathcal{T}}_{1}^{w_3,k_3^{\text{in}}} = \{5,\ldots,14\}$$

To conclude the example, we show the values that the variables would take if the flight took off on schedule, it used the alternative route and it were advanced 1 time period when traversing arc (w_4, w_5) and another when traversing arc (w_5, w_3). For arcs $\{(k_1, k_1^{\text{out}}), (k_1^{\text{out}}, w_1), (w_1, w_4), (w_4, w_5)\}$, we had that:

$$x^1_{1,k_1,k_1^{\text{out}}} = 1 \qquad x^2_{1,k_1^{\text{out}},w_1} = 1 \qquad x^4_{1,w_1,w_4} = 1 \qquad x^7_{1,w_4,w_5} = 1$$
$$x^2_{1,k_1,k_1^{\text{out}}} = 1 \qquad x^3_{1,k_1^{\text{out}},w_1} = 1 \qquad x^5_{1,w_1,w_4} = 1 \qquad x^8_{1,w_4,w_5} = 1$$
$$x^3_{1,k_1,k_1^{\text{out}}} = 1 \qquad x^4_{1,k_1^{\text{out}},w_1} = 1 \qquad x^6_{1,w_1,w_4} = 1 \qquad x^9_{1,w_4,w_5} = 1$$
$$x^{10}_{1,w_4,w_5} = 1$$

and for arcs $\{(w_5, w_3), (w_3, k_3^{\text{in}}), (k_3^{\text{in}}, k_3)\}$:

$$x^9_{1,w_5,w_3} = 1 \qquad x^5_{1,w_3,k_3^{\text{in}}} = 0 \qquad x^5_{1,k_3^{\text{in}},k_3} = 0$$
$$x^{10}_{1,w_5,w_3} = 1 \qquad \vdots \qquad \vdots$$
$$x^{11}_{1,w_5,w_3} = 1 \qquad x^9_{1,w_3,k_3^{\text{in}}} = 0 \qquad x^9_{1,k_3^{\text{in}},k_3} = 0$$
$$x^{12}_{1,w_5,w_3} = 1 \qquad x^{10}_{1,w_3,k_3^{\text{in}}} = 1 \qquad x^{10}_{1,k_3^{\text{in}},k_3} = 1$$
$$x^{13}_{1,w_5,w_3} = 1 \qquad \vdots \qquad \vdots$$
$$x^{14}_{1,w_3,k_3^{\text{in}}} = 1 \qquad x^{14}_{1,k_3^{\text{in}},k_3} = 1.$$

All the variables associated with arcs (w_1, w_2) and (w_2, w_3) would be equal to zero. And the values for the *xi* and *eta* variables would be: $\xi_1^{w_5-} = 1$, $\xi_1^{w_3-} = 1$, $\eta_1^{+} = 5$, $\eta_1^{-} = 0$, and the rest equal to zero.

Objective function

In Agustín et al. [4] multiple terms conform the objective function. Below we just present the most relevant ones, referring the reader to the paper for the rest.

1. Cost of using each arc, including that for alternative routes:

$$\sum_{f\in\mathcal{F}}\sum_{(m,n)\in\mathcal{A}_f} c_f^{m,n} x_{f,m,n}^{T_{*,f}^{m,n}}.$$

2. Cost of early and late arrivals at each node:

$$\sum_{f\in\mathcal{F}}\sum_{n\in\mathcal{N}_f\setminus\{k_f^d\}}\sum_{m\in\Gamma_f^-(n)}\sum_{t\in\widetilde{\mathcal{T}}_f^{m,n}} c_{ft}^n(x_{f,m,n}^t - x_{f,m,n}^{t-1}).$$

3. Cost of deviating from the planned speed:

$$\sum_{f\in\mathcal{F}}\sum_{n\in\mathcal{N}_f\setminus\{k_f^d,k_f^{a,\text{in}},k_f^a\}} (c_f^{n+}\xi_f^{n+} + c_f^{n-}\xi_f^{n-}).$$

4. Cost of deviating from the scheduled travel time:

$$\sum_{f\in\mathcal{F}}(c_f^{a+}\eta_f^+ + c_f^{a-}\eta_f^-).$$

   Note that the last three terms of the objective function reward, not only arriving without any delay, but also without changing the planned speed.

5. Cost to reward ground versus air delays:

$$\sum_{f\in\mathcal{F}}\Big[\sum_{t\in\mathcal{T}_{*,f}^{k_f^{a,\text{in}},k_f^a}} c_{ft}^{k_f^a}\big(x_{f,k_f^{a,\text{in}},k_f^a}^t - x_{f,k_f^{a,\text{in}},k_f^a}^{t-1}\big) - \sum_{t\in\mathcal{T}_{*,f}^{k_f^d,k_f^{d,\text{out}}}} c_{ft}^{k_f^d}\big(x_{f,k_f^d,k_f^{d,\text{out}}}^t - x_{f,k_f^d,k_f^{d,\text{out}}}^{t-1}\big)\Big].$$

   The reward occurs when late arrivals (positive minuend) are partially or totally caused by late departures (positive subtrahend).

Capacity constraints

$$\sum_{\substack{f\in\mathcal{F}:\\ \{k_f^d=k,t\in\widetilde{\mathcal{T}}_f^{k,k^{\text{out}}}\}}} \left(x^t_{f,k,k^{\text{out}}} - x^{t-1}_{f,k,k^{\text{out}}}\right) \le C_k^{t,\text{dep}} \quad \forall k\in\mathcal{K}, t\in\mathcal{T}, \tag{2.24}$$

$$\sum_{\substack{f\in\mathcal{F}:\\ \{k_f^a=k,\, t\in\widetilde{\mathcal{T}}_f^{k^{\text{in}},k}\}}} \left(x^t_{f,k^{\text{in}},k} - x^{t-1}_{f,k^{\text{in}},k}\right) \le C_k^{t,\text{arr}} \quad \forall k\in\mathcal{K}, t\in\mathcal{T}, \tag{2.25}$$

$$\sum_{\substack{f\in\mathcal{F}:\\ \{k_f^d=k,t\in\widetilde{\mathcal{T}}_f^{k,k^{\text{out}}}\}}} \left(x^t_{f,k,k^{\text{out}}} - x^{t-1}_{f,k,k^{\text{out}}}\right)+$$

$$\sum_{\substack{f\in\mathcal{F}:\\ \{k_f^a=k,\, t\in\widetilde{\mathcal{T}}_f^{k^{\text{in}},k}\}}} \left(x^t_{f,k^{\text{in}},k} - x^{t-1}_{f,k^{\text{in}},k}\right) \le C_k^t \quad \forall k\in\mathcal{K}, t\in\mathcal{T}, \tag{2.26}$$

$$\sum_{f\in\mathcal{F}}\Big(\sum_{n\in\mathcal{N}_f^{j+}}\sum_{m\in\Gamma_f^-(n)} x^t_{f,m,n} - \sum_{n\in\mathcal{N}_f^{j-}}\sum_{m\in\Gamma_f^-(n)} x^t_{f,m,n}\Big) \le C_j^t \;\; \forall j\in\mathcal{J}, t\in\mathcal{T}, \tag{2.27}$$

Flight structure constraints

$$\sum_{m\in\Gamma_f^+(n)} x^{t+\underline{\ell}_{f,n,m}}_{f,n,m} \le \sum_{m\in\Gamma_f^-(n)} x^t_{f,m,n} \le \sum_{m\in\Gamma_f^+(n)} x^{t+\overline{\ell}_{f,n,m}}_{f,n,m}$$
$$\forall f\in\mathcal{F}, n\in\mathcal{N}_f\setminus\{k_f^d,k_f^a\}, t\in\widetilde{\mathcal{T}}_f^n, \tag{2.28}$$

$$x^{T^{k_f^d,k_f^{d,\text{out}}}_{*,f}}_{f,k_f^d,k_f^{d,\text{out}}} = 1 \quad \forall f\in\mathcal{F}, \tag{2.29}$$

$$x^{t-1}_{f,m,n} - x^t_{f,m,n} \le 0 \quad \forall f\in\mathcal{F}, (m,n)\in\mathcal{A}_f, t\in\mathcal{T}^{m,n}_{*,f}, \tag{2.30}$$

$$x^{t-1}_{f,m,n} - x^t_{f,m,n} = 0 \quad \forall f\in\mathcal{F}, (m,n)\in\mathcal{A}_f, t\in\widetilde{\mathcal{T}}^{m,n}_f\setminus\mathcal{T}^{m,n}_{*,f}, \tag{2.31}$$

$$x^t_{f,k_f^d,k_f^{d,\text{out}}} - x^{t-\tau_{f',f}}_{f',k_f^{a,\text{in}},k_f^a} \le 0 \quad \forall (f',f)\in\mathcal{Q}, t\in\widetilde{\mathcal{T}}_f^{k_f^d,k_f^{d,\text{out}}}, \tag{2.32}$$

Delay constraints

$$\sum_{m\in\Gamma_f^+(n)}\Big(\sum_{t\in\widetilde{\mathcal{T}}_f^{n,m}} t(x^t_{f,n,m} - x^{t-1}_{f,n,m})\Big) - \sum_{m\in\Gamma_f^-(n)}\Big(\sum_{t\in\widetilde{\mathcal{T}}_f^{m,n}} t(x^t_{f,m,n} - x^{t-1}_{f,m,n})\Big) -$$
$$\sum_{m\in\Gamma_f^+(n)} \ell_{f,n,m} x^{T^{n,m}_{*,f}}_{f,n,m} - \xi_f^{n+} + \xi_f^{n-} = 0 \quad \forall f\in\mathcal{F}, n\in\mathcal{N}_f\setminus\{k_f^d,k_f^{a,\text{in}},k_f^a\}, \tag{2.33}$$

$$\sum_{t\in\widetilde{\mathcal{T}}_f^{k_f^{a,\text{in}},k_f^a}} t(x^t_{f,k_f^{a,\text{in}},k_f^a} - x^{t-1}_{f,k_f^{a,\text{in}},k_f^a}) - \sum_{t\in\widetilde{\mathcal{T}}_f^{k_f^d,k_f^{d,\text{out}}}} t(x^t_{f,k_f^d,k_f^{d,\text{out}}} - x^{t-1}_{f,k_f^d,k_f^{d,\text{out}}}) -$$

$$(\tau_f^a - \tau_f^d) x^{T^{*,f}_{k_f^{a,\text{in}},k_f^a}}_{f,k_f^{a,\text{in}},k_f^a} - \eta_f^+ + \eta_f^- = 0 \quad \forall f \in \mathcal{F}, \tag{2.34}$$

Variables' domain

$$\xi_f^{n+}, \xi_f^{n-} \geq 0 \quad \forall f \in \mathcal{F}, n \in \mathcal{N}_f \backslash \{k_f^d, k_f^{a,\text{in}}, k_f^a\}, \tag{2.35}$$

$$\eta_f^+, \eta_f^- \geq 0 \quad \forall f \in \mathcal{F}, \tag{2.36}$$

$$x^t_{f,m,n} \in \{0,1\} \quad \forall f \in \mathcal{F}, (m,n) \in \mathcal{A}_f, t \in \widetilde{\mathcal{T}}_f^{m,n}. \tag{2.37}$$

In the model, constraints (2.24)-(2.27) ensure capacity limits. Constraints (2.28) entail that if a flight reaches a node by time t, then it reaches one of the subsequent ones by a time period between $t + \underline{\ell}_{f,n,m}$ and $t + \overline{\ell}_{f,n,m}$. Constraints (2.29) ensure that all flights take off. Constraints (2.30) are for time connectivity. Constraints (2.31) forbid to make decisions in non-feasible time periods. Constraints (2.32) stand for connectivity between continued flights. Constraints (2.33) and (2.34) are to compute delays at intermediate waypoints of the route.

Note that with this formulation, the rerouting problems described in Case 1 and 2 for the previous model (ATFMRP by nodes) are no longer an issue: i) Capacity constraints, as accounting for arcs, can distinguish between multiples entries to sector II, and ii) The constraints causing the issue for Case 2 are not needed in the model.

Finally, to conclude the discussion, we mention that, despite the improvements introduced by this latter model, it still has some limitations related to the modifications of the flight plans at intermediate waypoints of the route. These limitations, fully discussed in the next section, motivated us to develop two new mathematical formulations for the ATFM problem.

2.5 NEW MATHEMATICAL FORMULATION

Note from the discussion of the previous section that the advantages of Agustín et al. [4] over the model of Bertsimas et al. [20] mainly arise from the definition of the variables. Defining the variables by arcs instead of nodes allows collecting more information, which results in better modeling. Based on this idea, we propose a new formulation

that, by adding to the variables' definition of Agustín et al. [4] information about the time leaving the tail node, and not only about the time reaching the head one, allows for a better cost representation and control of the decisions involved. Particularly, our proposal solves an important limitation of the previous models: Knowing the number of time periods spent at each arc of the route. This means, as fully discuss after presenting the first mathematical formulation, 1) having more control over how much each flight deviates from its initial flight plan, 2) being able to penalized changes of speed in a non-linear way (as occurs in practice[3]), and 3) proposing solutions which require less abrupt changes (e. g., smoother speed changes).

In addition to providing a more realistic modeling of the problem, our formulation turns out to be equivalent to a shortest path problem in multiple networks with limited shared resources. This fact results in most of the constraints of our ATFM formulation being facets, and in a larger variety of solving strategies, which motivates future chapters of the thesis.

Additional sets and parameters

- $\mathcal{F}_{\bar{p}} = \{f \in \mathcal{F} | \nexists f' \in \mathcal{F} : (f', f) \in \mathcal{Q}\}$, set of flights without a predecessor.
- $\mathcal{T}_f^{m,n} = \{(t_1, t_2) | \ t_1 \in \mathcal{T}_f^m, \ t_2 \in \mathcal{T}_f^n, \ \underline{\ell}_{f,m,n} \leq t_2 - t_1 \leq \overline{\ell}_{f,m,n}\}$, set of feasible pair time combinations for flight $f \in \mathcal{F}$ to travel through arc $(m, n) \in \mathcal{A}_f$ starting at t_1 and finishing at t_2. Note: since $\underline{\ell}_{f,m,n} = \overline{\ell}_{f,m,n} = 0$ for $(m, n) = (k_f^d, k_f^{d,\text{out}})$ and $(m, n) = (k_f^{a,\text{in}}, k_f^a)$, then $\mathcal{T}_f^{m,n} = \{(t, t) | \ t \in \mathcal{T}_f^n\}$, for those special arcs.
- $c_{f,m,n}^{t_1,t_2}$, penalization for flight $f \in \mathcal{F}$ for using arc $(m, n) \in \mathcal{A}_f$ departing from node m at t_1 and arriving at node n at t_2, $(t_1, t_2) \in \mathcal{T}_f^{m,n}$.

Note that for every arc $(m, n) \in \mathcal{A}_f$ in the air space, $c_{f,m,n}^{t_1,t_2}$ represents the penalization of speed decreases of the aircraft for $t_2 - t_1 > \ell_{f,m,n}$, whereas for $t_2 - t_1 < \ell_{f,m,n}$ it represents the penalization of speed increases. The rerouting cost can be considered by penalizing arcs in $\mathcal{A}_f \setminus \mathcal{A}_f^*$. For the special arcs $(k_f^{a,\text{in}}, k_f^a)$ and $(k_f^d, k_f^{d,\text{out}})$, $c_{f,k_f^d,k_f^{d,\text{out}}}^{t,t}$ represents the penalization of a ground delay of $t - \tau_f^d$ time periods, and $c_{f,k_f^{a,\text{in}},k_f^a}^{t,t}$ represents the penalization of late (for $t > \tau_f^a$) or early (for $t < \tau_f^a$) arrivals.

The existence of continued flights has a propagation effect in the network. In particular, for each pair $(f', f) \in \mathcal{Q}$, two different situations may appear: 1) Flight f' arrives

3 As exposed in Akgunduz et al. [8], the rate of fuel consumption is not a linear function of speed, so it should not be penalized in a linear way.

on time (or, in some cases, with a small delay), then the plan of flight f is not affected; and 2) Flight f' arrives late, so flight f departs with some delay. Now, let us consider the following sets associated with each pair $(f', f) \in \mathcal{Q}$:

- For each $t \in \mathcal{T}_f^{k_f^d}$, $\mathcal{T}_{f',f}^t = \left\{t' \,|\, t' \in \mathcal{T}_{f'}^{k_{f'}^a} \text{ such that } t = \max\{\tau_f^d, t' + \tau_{f',f}\}\right\}$. If flight f' arrives in one of these periods, then t is the first period at which the aircraft assigned to f is ready to take off.

For the sake of clarity in the definition of sets $\mathcal{T}_{f',f}^t$, let us consider a pair of flights $(f', f) \in \mathcal{Q}$ such that f' can arrive at $\mathcal{T}_{f'}^{k_{f'}^a} = \{5, 6, 7, 8\}$, and f can depart at $\mathcal{T}_f^{k_f^d} = \{7, 8, 9\}$, but waiting a minimum of $\tau_{f',f} = 1$ time period for preparing the aircraft. Thus, if flight f' arrives at $t' = 5$, flight f must wait until $t = 7$ before departing. Therefore, $\mathcal{T}_{f',f}^7 = \{5, 6\}$, $\mathcal{T}_{f',f}^8 = \{7\}$, $\mathcal{T}_{f',f}^9 = \{8\}$. Note that $\left\{\mathcal{T}_{f',f}^t, t \in \mathcal{T}_f^{k_f^d}\right\}$ defines a partition of $\mathcal{T}_{f'}^{k_{f'}^a}$.

Variables

$z_{f,m,n}^{t_1,t_2} = 1$, if flight f leaves node m at time t_1 and arrives at node n at time t_2, and 0 otherwise, $\forall f \in \mathcal{F}, (m,n) \in \mathcal{A}_f, (t_1, t_2) \in \mathcal{T}_f^{m,n}$.

$y_f^t = 1$, if the aircraft that operates flight f is ready for taking off at the beginning of time t, and 0 otherwise, $\forall f \in \mathcal{F}, t \in \mathcal{T}_f^{k_f^d}$.

As previously mentioned, our proposal extends the variables' definition of Agustín et al. [4] by including information about the time leaving the tail node. Actually, the following relationship exists between variables:

$$x_{f,m,n}^t = \sum_{\substack{(t_1,t_2) \in \mathcal{T}_f^{m,n}: \\ t_2 \leq t}} z_{f,m,n}^{t_1,t_2} \quad \forall f \in \mathcal{F}, (m,n) \in \mathcal{A}_f, t \in \widetilde{\mathcal{T}}_f^{m,n}.$$

Note also that the new variables are impulse variables (*at time*), instead of step ones (*by time*). These two changes in the variables' definition will result, as later exposed, in a shortest path problem structure with limited shared resources.

Remark 1. *We assume that for flights $f \in \mathcal{F}_{\bar{p}}$, the aircraft is always ready for departing at τ_f^d (i.e., $y_f^{\tau_f^d} = 1$), while for subsequent flights, the aircraft is not ready from the beginning, being only available if the previous flight has arrived well in advance (i.e., $y_f^{\tau_f^d} = 0, \forall f \in \mathcal{F} \setminus \mathcal{F}_{\bar{p}}$).*

Example ATFM (continuation)

To illustrate how the new z-variables work, as well as how the new sets $\mathcal{T}_f^{m,n}$ are computed, we continue with the example employed in the previous section of the thesis. Recall that the example was based on the situation depicted for flight $f = 1$ in Figure 1 (page 14), that a maximum departure delay of two periods was permitted, and that a graph representation when defining the variables by arcs can be seen in Figure 4 (page 29). Using the time information exposed in the latter figure, sets $\mathcal{T}_f^{m,n}$ are as follows:

$$\mathcal{T}_1^{k_1,k_1^{\text{out}}} = \{(1,1),(2,2),(3,3)\}$$
$$\mathcal{T}_1^{k_1^{\text{out}},w_1} = \{(1,2),(2,3),(3,4)\}$$
$$\mathcal{T}_1^{w_1,w_2} = \{(2,3),(3,4),(4,5)\}$$
$$\mathcal{T}_1^{w_2,w_3} = \{(3,4),(4,5),(5,6)\}$$
$$\mathcal{T}_1^{w_1,w_4} = \{(2,4),(3,5),(4,6)\}$$
$$\mathcal{T}_1^{w_4,w_5} = \{(4,7),(4,8),(5,8),(5,9),(6,9),(6,10)\}$$
$$\mathcal{T}_1^{w_5,w_3} = \{(7,9),(7,10),(8,10),(8,11),(9,11),(9,12),(10,12),(10,13)\}$$
$$\mathcal{T}_1^{w_3,k_3^{\text{in}}} = \{(4,5),(5,6),(6,7),(9,10),(10,11),(11,12),(12,13),(13,14)\}$$
$$\mathcal{T}_1^{k_3^{\text{in}},k_3} = \{(5,5),(6,6),(7,7),(10,10),(11,11),(12,12),(13,13),(14,14)\}$$

Assuming that in the example the flight departs with one period of delay and that it reaches its destination using the main route, the following variables would be equal to one:

$$y_1^1 \quad y_1^2 \quad z_{1,k_1,k_1^{\text{out}}}^{2,2} \quad z_{1,k_1^{\text{out}},w_1}^{2,3}$$
$$z_{1,w_1,w_2}^{3,4} \quad z_{1,w_2,w_3}^{4,5} \quad z_{1,w_3,k_3^{\text{in}}}^{5,6} \quad z_{1,k_3^{\text{in}},k_3}^{6,6},$$

and all the rest would be equal to zero. Note that $y_1^2 = 1$ due to the one period of departure delay.

From the example, the following two properties of the variables become clearer:

1. Each z-variable contains all the information about the consequences of that variable being equal to one: If it implies a speed change, the usage of an alternative route or a late arrival/departure (in the case of airports). Therefore, as detailed in Subsection 2.5.1, each modification of the flight plan can be penalized in a non-linear way while keeping the model linear.

2. By defining the variables as impulse variables, for each arc (m,n) in the solution, only one of the corresponding variables is equal to one. That is, if (m,n) belongs to the final route of flight f, then:

$$\sum_{(t_1,t_2)\in\mathcal{T}_f^{m,n}} z_{f,m,n}^{t_1,t_2} = 1.$$

Objective function

$$\min \sum_{f\in\mathcal{F}} \sum_{(m,n)\in\mathcal{A}_f} \sum_{(t_1,t_2)\in\mathcal{T}_f^{m,n}} c_{f,m,n}^{t_1,t_2} z_{f,m,n}^{t_1,t_2}, \tag{2.38}$$

Capacity constraints

$$\sum_{\substack{f\in\mathcal{F}:\\ \{k_f^d=k,t\in\mathcal{T}_f^k\}}} z_{f,k,k^{\text{out}}}^{t,t} \le C_k^{t,\text{dep}}, \quad \forall k\in\mathcal{K}, t\in\mathcal{T}, \tag{2.39}$$

$$\sum_{\substack{f\in\mathcal{F}:\\ \{k_f^a=k,t\in\mathcal{T}_f^k\}}} z_{f,k^{\text{in}},k}^{t,t} \le C_k^{t,\text{arr}}, \quad \forall k\in\mathcal{K}, t\in\mathcal{T}, \tag{2.40}$$

$$\sum_{\substack{f\in\mathcal{F}:\\ \{k_f^d=k,t\in\mathcal{T}_f^k\}}} z_{f,k,k^{\text{out}}}^{t,t} + \sum_{\substack{f\in\mathcal{F}:\\ \{k_f^a=k,t\in\mathcal{T}_f^k\}}} z_{f,k^{\text{in}},k}^{t,t} \le C_k^t, \quad \forall k\in\mathcal{K}, t\in\mathcal{T}, \tag{2.41}$$

$$\sum_{f\in\mathcal{F}} \sum_{(m,n)\in\mathcal{A}_f^j} \sum_{\substack{(t_1,t_2)\in\mathcal{T}_f^{m,n}:\\ t_1\le t<t_2}} z_{f,m,n}^{t_1,t_2} \le C_j^t, \quad \forall j\in\mathcal{J}, t\in\mathcal{T}. \tag{2.42}$$

Flight structure constraints

$$y_f^{\tau_f^d} = 1, \quad \forall f\in\mathcal{F}_{\bar{p}}, \tag{2.43}$$

$$y_f^t = y_f^{t+1} + z_{f,k_f^d,k_f^{d,\text{out}}}^{t,t}, \quad \forall f\in\mathcal{F}_{\bar{p}}, t\in\mathcal{T}_f^{k_f^d}\setminus\{T_f^{k_f^d}\}, \tag{2.44}$$

$$y_f^t = z_{f,k_f^d,k_f^{d,\text{out}}}^{t,t}, \quad \forall f\in\mathcal{F}_{\bar{p}}, t = T_f^{k_f^d}, \tag{2.45}$$

$$y_f^{\tau_f^d} = 0, \quad \forall f\in\mathcal{F}\setminus\mathcal{F}_{\bar{p}}, \tag{2.46}$$

$$y_f^t + \sum_{t'\in\mathcal{T}_{f',f}^t} z_{f',k_{f'}^{a,\text{in}},k_{f'}^a}^{t',t'} = y_f^{t+1} + z_{f,k_f^d,k_f^{d,\text{out}}}^{t,t}, \quad \forall (f',f)\in\mathcal{Q}, t\in\mathcal{T}_f^{k_f^d}\setminus\{T_f^{k_f^d}\}, \tag{2.47}$$

$$y_f^t + \sum_{t' \in \mathcal{T}^t_{f',f}} z^{t',t'}_{f',k^{a,\text{in}}_{f'},k^a_{f'}} = z^{t,t}_{f,k^d_f,k^{d,\text{out}}_f}, \quad \forall (f',f) \in \mathcal{Q}, t = T_f^{k_f^d}, \tag{2.48}$$

$$\sum_{m \in \Gamma_f^-(n)} \sum_{(t',t) \in \mathcal{T}_f^{m,n}} z^{t',t}_{f,m,n} = \sum_{m' \in \Gamma_f^+(n)} \sum_{(t,t') \in \mathcal{T}_f^{n,m'}} z^{t,t'}_{f,n,m'},$$

$$\forall f \in \mathcal{F}, n \in \mathcal{N}_f \setminus \{k_f^d, k_f^a\}, t \in \mathcal{T}_f^n. \tag{2.49}$$

Variables' domain

$$z^{t_1,t_2}_{f,m,n} \in \{0,1\}, \quad \forall f \in \mathcal{F}, (m,n) \in \mathcal{A}_f, (t_1,t_2) \in \mathcal{T}_f^{m,n}, \tag{2.50}$$

$$y_f^t \in \{0,1\}, \quad \forall f \in \mathcal{F}, t \in \mathcal{T}_f^{k_f^d}. \tag{2.51}$$

In the model, the objective function (2.38) accounts for the operation costs, including decisions about delays, rerouting or changes in the speed. Constraints (2.39)-(2.42) enforce capacity limits in airports and sectors. Constraints (2.43) state that all flights without a predecessor must have a ready-to-fly aircraft assigned since the scheduled departure time. Constraints (2.44) state that if a flight has its aircraft ready at time period t, then it can be scheduled for departing at that time period or delayed an additional one. Constraints (2.45) force the flight to take off if its aircraft is still ready at the beginning of the last time period it can depart. Constraints (2.46) state that a flight with a predecessor does not have its own aircraft ready to depart at the scheduled departure time (since it is operated by the same aircraft as its predecessor, see Remark 1). Constraints (2.47) and (2.48) are similar to constraints (2.44) and (2.45), but for flight f with a predecessor f'. In these cases, the right-hand side of each restriction takes into account that flight f can depart (or be delayed one additional time period) at time period t if its aircraft is ready from the previous time period ($y_f^t = 1$) or its predecessor flight f' has arrived well enough in advance ($\sum_{t' \in \mathcal{T}^t_{f',f}} z^{t',t'}_{f',k^{a,\text{in}}_{f'},k^a_{f'}} = 1$). Constraints (2.49) guarantee that once the aircraft reaches waypoint n, then it leaves it at the same time period.

2.5.1 *Cost analysis in the objective function*

The priority aim of ATFM models is to find flight plans such that the capacity of the different elements (airports and sectors) is never exceeded. This is achieved by proposing a set of constraints that adequately define a feasible region. Moreover, objective function coefficients should be oriented towards choosing a solution which does not differ

excessively from the initial flight plan and that, in addition, does so in the most balanced possible way. The solutions that are sought must: 1) Prefer ground delays to air delays; 2) Propose speed changes as smooth as possible; 3) Propose alternative routes when the original route is not available; and 4) Penalize late arrivals. In addition, these proposals should be such that changes are fairly distributed between different flights (for example, a solution in which two flights have a delay of one period is preferable to a solution in which a single flight concentrates two periods of delay). One of the advantages of the proposed model over previous ones in the literature, e. g., Bertsimas et al. [20] or Agustín et al. [4], is precisely the flexibility to fix the coefficients in the objective function in order to take these facts into account.

Notice that, since each variable $z^{t_1,t_2}_{f,m,n}$ contains information about the time leaving the tail node m and reaching the head node n, these variables determine the speed at which the aircraft traverses arc (m,n). This implies that we can penalize changes in the speed in a non-linear way, which is what happens in reality. To better understand this, consider the following example. Let us consider two aircraft, one flying from node m to node n, and another flying from m' to n'. Imagine that the scheduled travel time for the first route is 20 minutes, while the second one is 25 minutes. Now consider that both aircraft must be delayed 5 minutes in their respective routes. If we penalized delays based on this quantity (as the current state of the art does), both cases would have the same cost even though they should not because the first case requires a reduction in the speed of 25%, while in the second requires one of 20%. Thanks to our variables we know the percentage of reduction beforehand, so we can establish penalizations based on this non-linear quantity while maintaining linearity in our model. As far as we know, the only work which also incorporates this idea of non-linear speed penalization is Akgunduz et al. [8].

We have considered the following penalization structure. Let us consider varaible $z^{t_1,t_2}_{f,m,n}$. If this variable takes value 1, then the corresponding flight leaves waypoint m at time period t_1 and reaches waypoint n at time period t_2. The penalization $c^{t_1,t_2}_{f,m,n}$ is such that:

- If $m = k^d_f$ (ground delay penalization), we use, based on Bertsimas et al. [20], $c^{t,t}_{f,k^d_f,k^{d,\text{out}}_f} = c_g|t - \tau^d_f|^{1+\epsilon_1}$, where c_g is a constant to penalize ground delays. For $0 < \epsilon_1 < 1$ this is a convex and superlinear penalization, looking for a better distribution of the delays between several flights (the penalization is greater for one flight delayed two periods than for two flights delayed one period each).

- If $n = k_f^a$ (arrival delay penalization): $c^{t,t}_{f,k_f^{a,\text{in}},k_f^a} = c_a|t - \tau_f^a|^{1+\epsilon_2}$, where c_a is a constant to penalize air delays. As in the previous case, this penalization seeks to distribute small delays in arrivals among several flights as opposed to concentrating large delays on a few flights.

- In all other cases (airspace arcs): $c^{t_1,t_2}_{f,m,n} = c_r(t_2 - t_1) + c_v\left|\frac{t_2-t_1-\ell_{m,n}}{\ell_{m,n}}\right|^{1+\epsilon_3}$. This penalization includes several elements:

  - c_r is a constant to penalize the use of arcs which do not belong to the main route. Therefore, $c_r = 0$ for arcs in the main route and $c_r > 0$ for the rest. Notice that by multiplying this quantity by $(t_2 - t_1)$ the time spent in the alternative route is taken into account.
  - As in the departure and arrival delay penalization, ϵ_3 seeks to penalize the concentration of speed changes in a few arcs.
  - By dividing by $\ell_{m,n}$, the length of the arc is also taken into account: the less the scheduled time to cross an arc, the greater the penalization for changing the flight time one unit. That is, greater velocity changes have a greater penalization.
  - Finally, c_v is a constant to penalize speed changes.

  Note that, while penalization for delays in arrivals and departures are common in the literature, penalization at intermediate points is not straightforward. In fact, very few works consider it. In particular, Agustín et al. [4] penalize deviations in the arrivals at each waypoint using a linear function, but without considering the length of the arc (they do not penalize the change in speed, but the delay in the arrival at the waypoints).

In general, since ground delays are preferred to air delays, it seems to be recommendable to set $c_g < c_v$. Figure 5 shows the impact of different values of the ϵ parameter. The picture at the top represents the penalization assigned to departure and arrival delays for $c_g = c_a = 1$. The picture at the bottom represents the penalization assigned to the increment in the time spent (velocity change) for a flight in traversing an arc, for $\ell_{f,m,n} = 2$, $c_r = 0$ and $c_v = 1$. Note that $\epsilon = 0$ corresponds to the linear case.

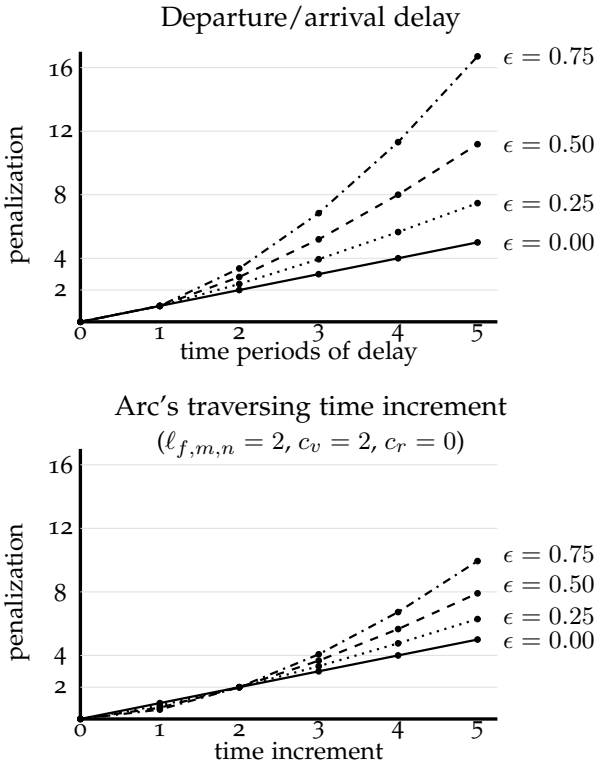


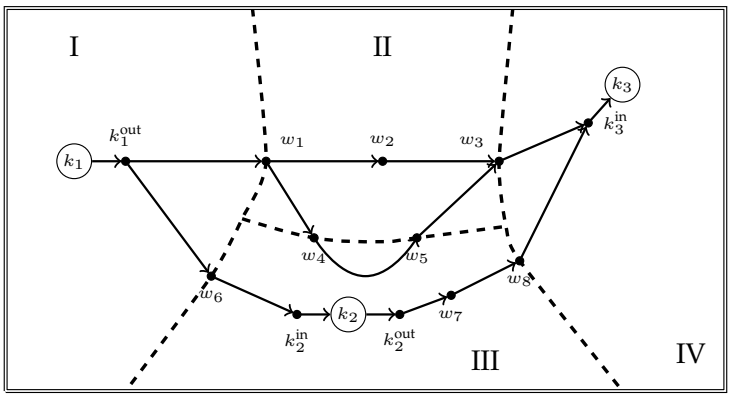
Figure 5: Illustration of the penalization functions.

2.5.2 *Model extension*

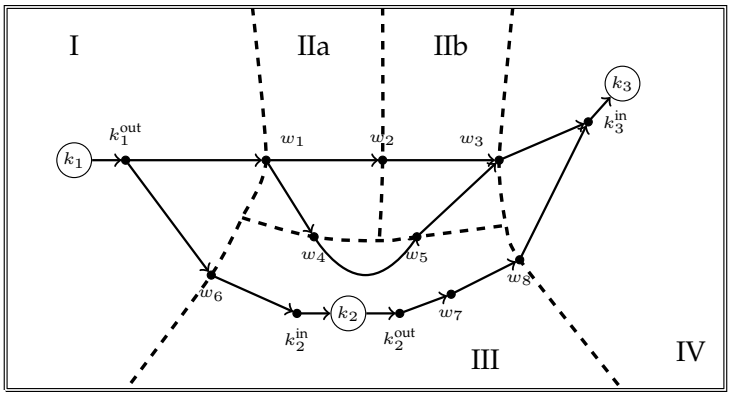
Usually, sectors are defined taking into account the maximum number of flights that air traffic controllers working in a single sector can safely manage. Initially, the airspace is divided into the so-called elementary sectors, which can be assembled into larger units called collapsed sectors. A partition of the entire airspace, defined by a specific combination of elementary and/or collapsed sectors, is called sector configuration (see Baumgartner [12]). Therefore, along the planning horizon, different sector configurations can be used in order to adapt the nominal capacity of the sectors to the expected air traffic demand. Unlike previous models in the literature that use a fixed sector configuration during the whole planning horizon, the model that we present here allows for a dynamic one.

This is because our model considers flight routes as sequences of waypoints not necessarily associated with air sector boundaries (some of them can lie in the interior), making the flight structure independent of the air sector configuration. As an example,

let us suppose that in the instance illustrated in Figure 1, instead of having a static airspace division in four air sectors, {I, II, III, IV}, the Air Traffic Control service considers that, from a given time period t^*, sector II must be split into two different sectors, say IIa and IIb (see Fig. 6). Then, the set of sectors is $\mathcal{J}$={I, II, IIa, IIb, III, IV} that is not a partition of the airspace. But considering $\mathcal{J}_t$={I, II, III, IV} for $t < t^*$ and $\mathcal{J}_t$={I, IIa, IIb, III, IV} for $t \geq t^*$, $\mathcal{J}_t$ defines a sector configuration for each t. Now, w_2 is an exit waypoint from sector II_a and an entering point to sector II_b.



$$\mathcal{J}_t = \{I, II, III, IV\}$$



$$\mathcal{J}_t = \{I, IIa, IIb, III, IV\}$$

Figure 6: Illustration of dynamic sector configuration.

2.6 ALTERNATIVE FORMULATION

Model (2.38)–(2.51), thanks to indexing the z-variables by arcs $(m, n) \in \mathcal{A}_f$ and by the time leaving m and reaching n, allows to represent the flight plans using a time-

expanded or 4D-network. That is, a network $\widetilde{\mathcal{G}} = (\widetilde{\mathcal{N}}, \widetilde{\mathcal{A}})$, where each node $n \in \widetilde{\mathcal{N}}$ represents a position in space and a time period in which that position can be reached; and where each arc $a \in \widetilde{\mathcal{A}}$ corresponds to a particular decision variable with a cost associated.

For instance, for a given flight f, graph $\mathcal{G}_f = (\mathcal{A}_f, \mathcal{N}_f)$, shown[4] in Figure 7, represents the set of two available routes that f can follow departing from airport k_1 to airport k_3.

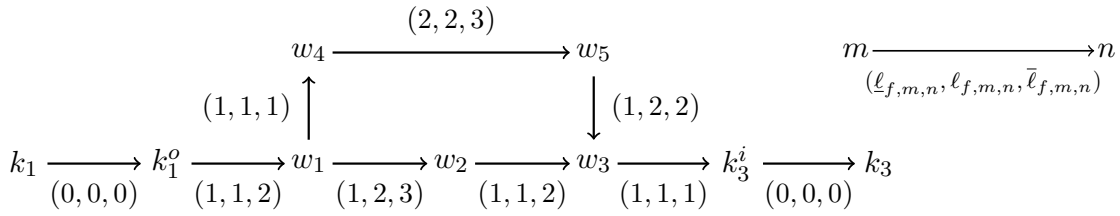


Figure 7: Example of 3D flight plan.

The figure shows, for each arc, the minimum, the scheduled and the maximum number of time periods that flight f needs for traversing it. Using this information and considering that, for example, the flight is scheduled to take off at $t = 1$ and that the maximum ground delay is one time period, it is possible to build the 4D-network $\widetilde{\mathcal{G}}_f = (\widetilde{\mathcal{N}}_f, \widetilde{\mathcal{A}}_f)$ represented in Figure 8. In that network, each circled-node $nt \in \widetilde{\mathcal{N}}_f$ is the result of combining each node $n \in \mathcal{N}_f$ with the possible time periods to reach it, $t \in \mathcal{T}_f^n$.

There exists an arc from node mt_1 to node nt_2 in the following two cases:

- If $m \neq n$ and it is feasible for flight f to go from node $m \in \mathcal{N}_f$ to node $n \in \mathcal{N}_f$, departing at t_1 and arriving at t_2. It is not difficult to see that there is a one-to-one correspondence between the arcs $\widetilde{\mathcal{A}}_f$ of this form and the z-variables in model (2.38)–(2.51). To be specific, arc (mt_1, nt_2) is associated with variable $z_{f,m,n}^{t_1,t_2}$, for all $(m, n) \in \mathcal{A}_f$, $(t_1, t_2) \in \mathcal{T}_f^{m,n}$.

- If $m = n = k_f^d$, $t_1 \in \mathcal{T}_f^{k_f^d} \setminus \{T_f^{k_f^d}\}$ and $t_2 = t_1 + 1$. Each of these arcs (the vertical one in the figure) represent a one-time period ground delay for the flight. Notice that there is also a one-to-one correspondence between these arcs and the y-variables

4 Due to aesthetic reasons, when drawing the text inside the nodes, instead of using k_1^{out} and k_3^{in}, we will use k_1^o and k_3^i respectively.

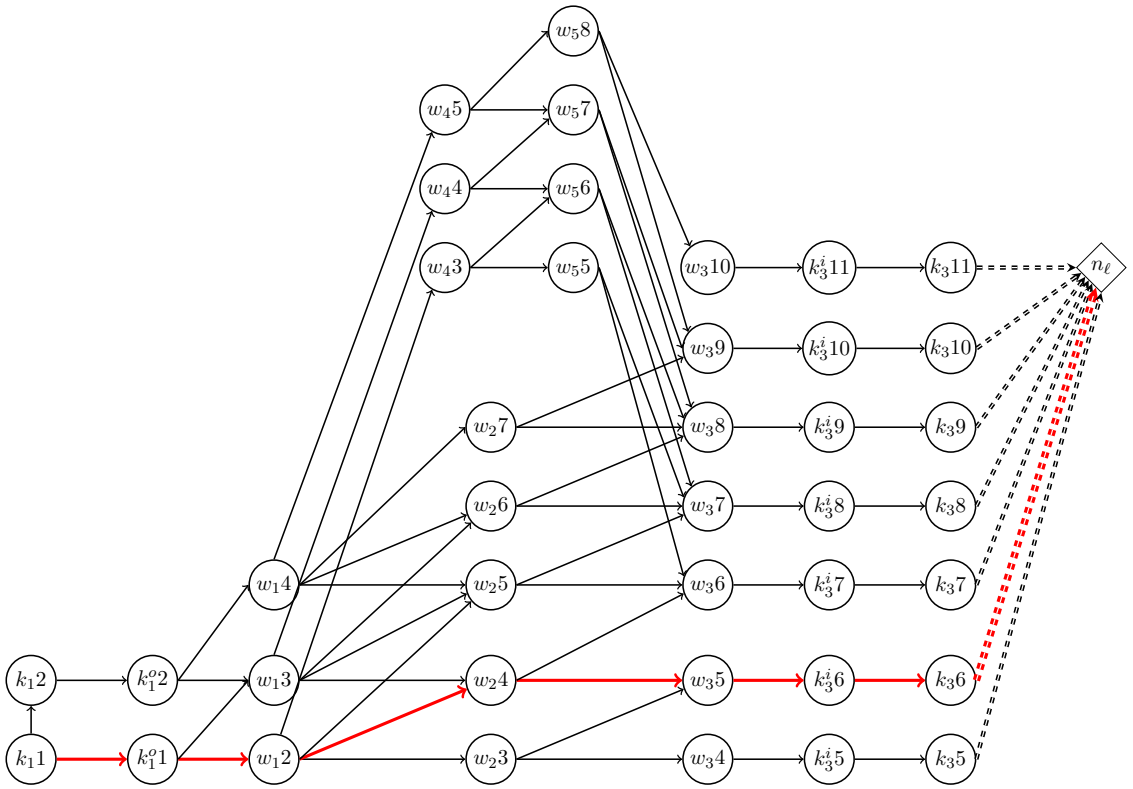


Figure 8: 4D-network example.

in model (2.38)–(2.51): when delayed, the aircraft is ready at the beginning of next time period.

In Figure 8, it can also be appreciated that the network includes the (auxiliary) diamond node n_ℓ, which represents the end of aircraft operations and which is linked to every arrival airport node by double-line arcs. These arcs have no decision variables associated.

Note that there is a one-to-one correspondence between each possible flight plan in the example and each path of the 4D-network from $k_1 1$ (in general, $k_f^d \tau_f^d$) to n_ℓ. In particular, the initial scheduled plan is represented using bold red arcs in Figure 8.

Regarding costs, since each non-dashed arc in $\widetilde{\mathcal{A}}_f$ has a correspondence with a decision variable in model (2.38)–(2.51), a cost $\tilde{c}_a$ for each arc $a = (mt_1, nt_2) \in \widetilde{\mathcal{A}}_f$ is assigned such that $\tilde{c}_a = c_{f,m,n}^{t_1,t_2}$, $(m,n) \in \mathcal{A}_f$, $(t_1,t_2) \in \mathcal{T}_f^{m,n}$. The cost for the double-line arcs is set to 0. As a result, the shortest path from $k_1, 1$ to n_ℓ represents the cheapest (presumably the original) flight plan for flight f.

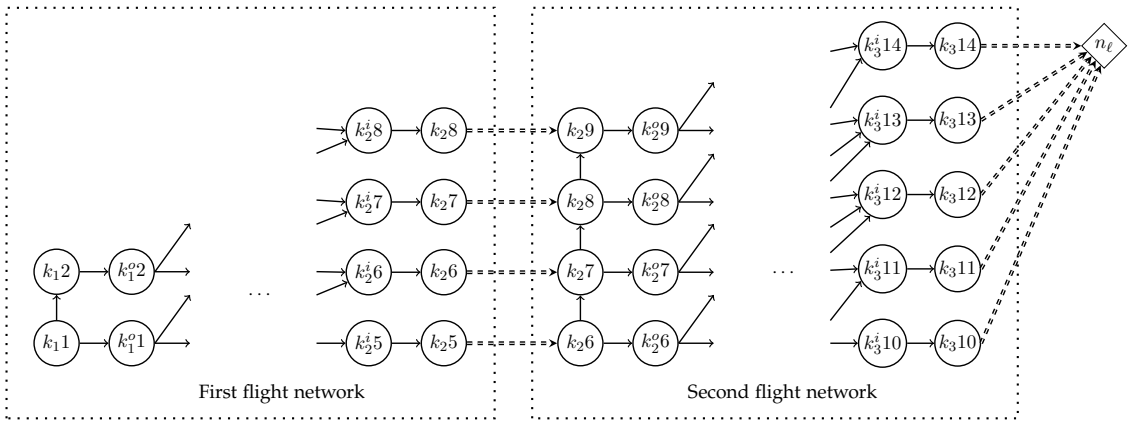


Figure 9: 4D-network for two continued flights.

The 4D-network can be extended to cases where an aircraft operates several continued flights. This is done by defining for each aircraft, instead of flight, a network $\widetilde{\mathcal{G}}_s = (\widetilde{\mathcal{N}}_s, \widetilde{\mathcal{A}}_s)$, where $s \in \mathcal{S} = \{1, \ldots, |\mathcal{S}|\}$ stands for the s-th aircraft.

For the sake of illustration, let us consider the example of two continued flights ($\tau_{f',f} = 1$) depicted in Figure 9. There, the nodes and arcs defining each flight remain as in the one-flight 4D-network. The connection between the two flights is done by introducing new double-line arcs representing the transition from the first flight to the second one. Concretely, if t is the first time period at which the second flight, f, can depart from k_f^d given the arrival of the first flight, f', at time t' to airport $k_{f'}^a$, then nodes $k_{f'}^a t'$ and $k_f^d t$ are linked. That is, $(k_{f'}^a t', k_f^d t) \in \widetilde{\mathcal{A}}_s$, $\forall\, k_f^d t \in \widetilde{\mathcal{N}}_f$, $t' \in \mathcal{T}^t_{f',f}$ and $k_{f'}^a t' \in \widetilde{\mathcal{N}}_{f'}$.

Note that as these new double-line arcs do not represent any decision, they do not have any variable associated. In fact, the 4D-network can be shrunk by merging every set of nodes connected by these double-line arcs into just one node. In Figure 9, as in the one-flight 4D-network case, each path from $k_1 1$ to n^ℓ represents a specific flight plan for the set of flights operated by the aircraft.

From the previous figures, it is clear that by using this representation based on the 4D-networks $\widetilde{\mathcal{G}}_s = (\widetilde{\mathcal{N}}_s, \widetilde{\mathcal{A}}_s)$, $s \in \mathcal{S}$, the ATFMRP can be formulated as a shortest path problem in multiple networks (one per each aircraft $s \in \mathcal{S}$) with limited shared resources (capacity constraints). To that end, let us introduce the following notation:

$\mathcal{F}_s$, set of flights f that belong to network $\widetilde{\mathcal{G}}_s$, $s \in \mathcal{S}$, $f \in \mathcal{F}$. Notice that $\{\mathcal{F}_s | s \in \mathcal{S}\}$ defines a partition of $\mathcal{F}$. Note also that there exists a unique first flight $f_0^s \in \mathcal{F}_s \cap \mathcal{F}_{\bar{p}}$ and a unique last flight $f_\ell^s \in \mathcal{F}_s$ operated by aircraft s.

$\widetilde{\mathcal{A}}^{\text{dep}}_{s,k,t} = \{(kt, k^{\text{out}}t) \in \widetilde{\mathcal{A}}_s |\ f \in \mathcal{F}_s :\ k = k^d_f,\ (t,t) \in \mathcal{T}_f^{k,k^{\text{out}}}\}$, set of arcs referred to time period t which represent the departure from airport k in the s-th network, $s \in \mathcal{S}$, $k \in \mathcal{K}$, $t \in \mathcal{T}$.

$\widetilde{\mathcal{A}}^{\text{arri}}_{s,k,t} = \{(k^{\text{in}}t, kt) \in \widetilde{\mathcal{A}}_s |\ f \in \mathcal{F}_s :\ k = k^a_f,\ (t,t) \in \mathcal{T}_f^{k^{\text{in}},k}\}$, set of arcs referred to time period t which represent the arrival at airport k in the s-th network, $s \in \mathcal{S}$, $k \in \mathcal{K}$, $t \in \mathcal{T}$.

$\widetilde{\mathcal{A}}^{\text{sec}}_{s,j,t} = \{(mt_1, nt_2) \in \widetilde{\mathcal{A}}_s |\ f \in \mathcal{F}_s :\ (m,n) \in \mathcal{A}^j_f,\ (t_1,t_2) \in \mathcal{T}_f^{m,n} : t_1 \leq t < t_2\}$, set of arcs referred to time period t which belong to sector j in the s-th network, $s \in \mathcal{S}$, $j \in \mathcal{J}$, $t \in \mathcal{T}$.

Thus, the ATFMRP admits the following formulation:

$$\min \sum_{s\in\mathcal{S}} \sum_{a\in\widetilde{\mathcal{A}}_s} \tilde{c}_{s,a}\rho_{s,a}, \tag{2.52}$$

subject to:

$$\sum_{a\in\Lambda^-(n)} \rho_{s,a} - \sum_{a\in\Lambda^+(n)} \rho_{s,a} = \begin{cases} 1, & \text{if } n = k^d_{f^s_0}\tau^d_{f^s_0}, \\ -1, & \text{if } n = n^s_\ell, \\ 0, & \text{otherwise.} \end{cases} \quad \forall s \in \mathcal{S}, n \in \widetilde{\mathcal{N}}_s \tag{2.53}$$

$$\sum_{s\in\mathcal{S}} \sum_{a\in\widetilde{\mathcal{A}}^{\text{dep}}_{s,k,t}} \rho_{s,a} \leq C^{t,\text{dep}}_k, \quad \forall k \in \mathcal{K}, t \in \mathcal{T}, \tag{2.54}$$

$$\sum_{s\in\mathcal{S}} \sum_{a\in\widetilde{\mathcal{A}}^{\text{arri}}_{s,k,t}} \rho_{s,a} \leq C^{t,\text{arr}}_k, \quad \forall k \in \mathcal{K}, t \in \mathcal{T}, \tag{2.55}$$

$$\sum_{s\in\mathcal{S}} \sum_{a_1\in\widetilde{\mathcal{A}}^{\text{arri}}_{s,k,t}} \rho_{s,a_1} + \sum_{s\in\mathcal{S}} \sum_{a_2\in\widetilde{\mathcal{A}}^{\text{dep}}_{s,k,t}} \rho_{s,a_2} \leq C^t_k, \ \ \forall k \in \mathcal{K}, t \in \mathcal{T}, \tag{2.56}$$

$$\sum_{s\in\mathcal{S}} \sum_{a\in\widetilde{\mathcal{A}}^{\text{sec}}_{s,j,t}} \rho_{s,a} \leq C^t_j, \quad \forall j \in \mathcal{J}, t \in \mathcal{T}, \tag{2.57}$$

$$\rho_{s,a} \in \{0,1\}, \quad \forall s \in \mathcal{S}, a \in \widetilde{\mathcal{A}}_s, \tag{2.58}$$

where $\rho_{s,a}$ is a binary variable that takes value 1 if arc $a \in \widetilde{\mathcal{A}}_s$ belongs to the solution path in network $s \in \mathcal{S}$ and 0 otherwise, and $\Lambda^-(n)$ and $\Lambda^+(n)$ are the sets of outgoing and incoming arcs for node $n \in \widetilde{\mathcal{N}}_s$, respectively.

Proposition 1. *Models* (2.38)–(2.51) *and* (2.52)–(2.58) *are equivalent.*

Proof. The proof follows straightforward from the network construction made before. □

As a direct consequence, it can be concluded that the ATFMRP belongs to class $\mathcal{NP}-$complete. This is because the constrained shortest path problem is $\mathcal{NP}-$complete (Ahuja, Magnanti, and Orlin [6, Appx. B]). Furthermore, as the matrix associated with constraints (2.53) is totally unimodular, the extreme points of the feasible region defined by them are integer (Bertsimas and Tsitsiklis [18, Ch. 7]). As it will be shown in Chapter 4, this fact leads to very tight Linear Programming (LP) relaxations, permitting commercial Mix Integer Programming (MIP) solvers to solve fairly big instances of the problem in a personal computer within reasonable computational times.

2.7 GRAPH OF CONFLICTS

To reduce the size of the mathematical formulation, we propose a methodology based on the construction of a conflict graph. By using this graph, the problem could eventually be split into disjoint subproblems to be solved independently. Furthermore, in some cases, the size of the problem can be reduced by identifying when the main route of a flight has no conflict with any other flight and, therefore, can be fixed and removed from the problem.

The cornerstone of the proposal is that capacity constraints define potential conflicts between aircraft: two aircraft may not be able to occupy the same sector/airport at the same time. To be concrete, given two aircraft s, $s' \in \mathcal{S}$, we say that there is a potential conflict between them if there is at least one arc in $\widetilde{\mathcal{G}}_s$ and one arc in $\widetilde{\mathcal{G}}_{s'}$ with a non-zero coefficient in the same capacity constraint.

We distinguish three types of conflicts between two aircraft: those involving the main routes, those involving the main route of one aircraft and a secondary route of the other, and those involving only secondary routes. We will represent these conflicts through a multigraph $\mathcal{H} = (\mathcal{S}, \mathcal{M})$, where $\mathcal{S}$ is the set of aircraft and $\mathcal{M}$ is the set of arcs such that $(s, s') \in \mathcal{M}$ if aircraft s and s' are involved in a potential capacity conflict. In order to use $\mathcal{H}$ for reducing the size of the problem, let us first introduce the following elements:

$\widetilde{\mathcal{A}}_s^* \subseteq \widetilde{\mathcal{A}}_s$, subset of arcs belonging to $\widetilde{\mathcal{A}}_s$ that define the main route, $s \in \mathcal{S}$.

$\mathcal{R}$, set of indexes of the capacity constraints. Note that if it can be proven, using some preprocessing process, that a capacity constraint is redundant (i. e., the maximum number of aircraft that can eventually be flying through the constraint's associated sector and time period is not greater than its capacity), then, the constraint index can be removed from $\mathcal{R}$.

$\widetilde{\mathcal{A}}^r$, set of arcs in $\cup_{s\in\mathcal{S}}\widetilde{\mathcal{A}}_s$ with non-zero coefficients in constraint $r \in \mathcal{R}$.

The set of arcs $\mathcal{M}$ is the union of three disjoint sets: $\mathcal{M}^{1,1}$, $\mathcal{M}^{1,2}$ and $\mathcal{M}^{2,2}$, which represent each of the three types of conflicts considered:

$\mathcal{M}^{1,1}$, arcs connecting two aircraft (nodes) that have at least one conflict in their main routes. Formally, $(s,s'),(s',s) \in \mathcal{M}^{1,1} \Leftrightarrow \exists r \in \mathcal{R}$ such that $\widetilde{\mathcal{A}}^r \cap \widetilde{\mathcal{A}}^*_s \neq \emptyset$ and $\widetilde{\mathcal{A}}^r \cap \widetilde{\mathcal{A}}^*_{s'} \neq \emptyset$, for $s, s' \in \mathcal{S}$, $s \neq s'$.

$\mathcal{M}^{1,2}$, arcs connecting two aircraft that have at least one conflict between the main route of the first of them and an alternative route of the second one. Mathematically, $(s,s') \in \mathcal{M}^{1,2} \Leftrightarrow \exists r \in \mathcal{R}$ such that $\widetilde{\mathcal{A}}^r \cap \widetilde{\mathcal{A}}^*_s \neq \emptyset$ and $\widetilde{\mathcal{A}}^r \cap (\widetilde{\mathcal{A}}_{s'} \setminus \widetilde{\mathcal{A}}^*_{s'}) \neq \emptyset$, for $s, s' \in \mathcal{S}$, $s \neq s'$. Note that in general $(s,s') \nLeftrightarrow (s',s)$.

$\mathcal{M}^{2,2}$, arcs connecting two aircraft that have at least one conflict in their alternative routes. Mathematically, $(s,s'),(s',s) \in \mathcal{M}^{2,2} \Leftrightarrow \exists r \in \mathcal{R}$ such that $\widetilde{\mathcal{A}}^r \cap (\widetilde{\mathcal{A}}_s \setminus \widetilde{\mathcal{A}}^*_s) \neq \emptyset$ and $\widetilde{\mathcal{A}}^r \cap (\widetilde{\mathcal{A}}_{s'} \setminus \widetilde{\mathcal{A}}^*_{s'}) \neq \emptyset$, for $s, s' \in \mathcal{S}$, $s \neq s'$.

To decompose the original problem into independent subproblems and/or reduce its size by using the multigraph of conflicts, we use the following result.

Proposition 2. *Let $\mathcal{H} = (\mathcal{S}, \mathcal{M})$ be the multigraph of conflicts associated with a given ATFMRP problem. Then:*

(i) *Model (2.52)–(2.58) can be solved by optimizing as many independent subproblems as connected components there are in $\mathcal{H}$; each subproblem formed by the 4D-networks of the aircraft conforming the component.*

(ii) *For each singleton connected component of $\mathcal{H}$, the main route of all flights operated by the associated aircraft is part of the optimal solution.*

(iii) *If for an aircraft $s \in \mathcal{S}$, $\nexists s' \in \mathcal{S} : (s,s') \in \mathcal{M}^{1,1}$ and $(s,s') \in \mathcal{M}^{1,2}$, then the main route of all flights operated by s is part of the optimal solution.*

(iv) *If for a connected component of* $\mathcal{H}$ $\nexists s, s' \in \mathcal{S} : (s, s') \in \mathcal{M}^{1,1}$*, then the main routes of all flights operated by the aircraft of that component are part of the optimal solution.*

Proof. The proof is straightforward taking into account that only capacity constraints (2.54)–(2.57) have non-zero coefficients in more than one aircraft. Thus, by construction, if two aircraft belong to different connected components, none of the flights operated by these aircraft can use the same sector/airport at the same time period in any of their feasible 4D-routes. Consequently, (i) and (ii) hold. By construction, if condition (iii) holds, the main route of aircraft s is free of conflicts and, thus, it can be used in the optimal solution. Condition (iv) is very restrictive since it implies that there is no conflict between the main routes of the aircraft forming the component. In such a case, it is impossible that there exists a violation of capacity constraints when the flights are using their main routes and, therefore, the main route of all the involved flights can be included in the optimal solution. □

We conclude the chapter by pointing out to the reader that a computational experience to discuss the empirical behavior of the proposals made in this chapter will be presented in Chapter 4. There it will be shown how the formulations proposed, in addition to the modeling advantages exposed in this chapter, allows commercial MIP solvers to tackle ATFM instances up to a certain size in times that are feasible for the industry.

3

SHARED RESOURCE CONSTRAINED MULTI-SHORTEST PATH PROBLEM

Motivated by the 4D-network ATFM formulation presented before, this chapter introduces a new family of constrained shortest path problems that, as far as we know, has not been previously addressed in literature: the Shared Resource Constrained Multi-Shortest Path Problem (SRC-MSPP). In addition to its mathematical formulation and characteristics, in the chapter it is shown how the SRC-MSPP permits to model project scheduling problems where the activities of the projects are sequenced in serial, and multiple extensions are considered at the same time (e.g., lag times, different execution modes, etc). The chapter also includes two different solution methods for the SRC-MSPP. The first one is a matheuristic algorithm designed to take advantage of the modern computer architecture with multiple cores. The algorithm consists of three phases: 1) Generation of feasible solutions, 2) Combination of solutions, and 3) Solution improvement. The second solution method studied consists of two Lagrangian Relaxations for the SRC-MSPP. The first one is based on dualizing the capacity constraints existing in the problem, and the second one on dualizing a copy of the decision variables. Most of the content exposed in this chapter was developed in collaboration with prof. Manuel Laguna during an internship and can be found in García-Heredia, Molina, Laguna, and Alonso-Ayuso [54].

3.1 INTRODUCTION

The Shortest Path Problem (SPP) is very well known in the Operations Research literature. The goal is to find the minimum-cost path connecting an origin node and a destination node in a network. The variety of applications is what makes this problem relevant. Applications range from computing the best route in GPS navigators, to obtaining minimum risk routes for hazmat shipments (Carotenuto, Giordani, and Ricciardelli [24]), or routing protocols in IP networks (Pióro, Szentesi, Harmatos, Jüttner, Gajowniczek, and Kozdrowski [104]). Moreover, it may also appear as a subproblem of other optimization problems, e.g., in Air Traffic Flow Management as demonstrated in this thesis.

The SPP is commonly solved with algorithms that have a polynomial-time complexity such as Dijkstra's (Dijkstra [39]) and Bellman-Ford (Bellman [15]). Particularly, in directed acyclic networks, the SPP can be solved in a complexity that is linear in the number of arcs by using the Bellman principle of optimality (Bellman [14]).

All this has propitiated the development of several extensions of the SPP to address more complex situations. Some of these extensions are:

- The k-shortest path problem (Eppstein [48] and Yen [116]), where the goal is to find the k paths of minimum cost between two nodes. An application of this occurs in hazmat shipments.

- The multi-objective shortest path problem (Aneja and Nair [10], Guerriero and Musmanno [67], and Raith and Ehrgott [105]), where the goal is to find efficient (non-dominated) solutions according to more than one criterion. An application of this occurs when trying to balance time and fuel consumption.

- The dynamic shortest path problem (Ahuja, Orlin, Pallottino, and Scutella [7] and Chabini [25]), where properties such as the cost of each arc may change along time. For example, shipping costs may vary throughout a day.

- The Resource Constrained Shortest Path Problem (RCSPP), where the consumption of various resources is associated with each arc of the network. The objective is to find the Shortest Path (SP) connecting two nodes, while ensuring that available resource quantities are not exceeded (Beasley and Christofides [13] and Handler and Zang [72]).

The latter is one of the most studied extensions of the SPP. In addition to the straightforward application of finding the shortest path subject to time or fuel limits, the RCSPP with one resource has been embedded as a pricing subproblem in column-generation procedures for vehicle routing problems (see Chabrier [26] or Montoya, Guéret, Mendoza, and Villegas [100]).

The RCSPP belongs to the $\mathcal{NP}-$complete class (Ahuja et al. [6, Appx. B]) and it is usually solved by means of specialized algorithms (see, for instance, Dumitrescu and Boland [43], Garcia [56], Lozano and Medaglia [95] or Horváth and Kis [76]). The basic idea behind these algorithms is to identify and prune dominated and infeasible paths in order to reduce the size of the problem to be able to solve it via dynamic programming or similar approaches.

In this chapter we address an extension of the RCSPP that, as far as we know, has not appeared in the literature: the Shared Resource Constrained Multi-Shortest Path Problem (SRC-MSPP). The problem consists of finding, for each network within a collection, a path between a given source (or origin) node and a given sink (or destination) node that minimizes the total cost (the sum of costs of the arcs forming the paths), while not exceeding a limit in the usage of a set of common resources shared by all networks. This extension is motivated by the results obtained in the previous chapter of the thesis about ATFM.

The goal in this chapter is to formally describe and study the SRC-MSPP, present a $0-1$ Mathematical Optimization model, show how it can be applied to some scheduling problems (besides ATFM ones), and study different solution methods for it.

The motivation for the development of solution method for the SRC-MSPP is that exact mathematical programming solvers are not able to solve instances of the size found in practice and that, as we discuss in Subsection 3.2.4, it is not possible to simply apply the solution methods available in the literature that have been developed for the resource constrained shortest path problem.

Applications of the SRC-MSPP include the aforementioned ATFM problem, or the train rescheduling problem (Josyula, Törnquist Krasemann, and Lundberg [82] and Törnquist [109, 110]). Actually, we noticed that this type of scheduling problems correspond to project scheduling problems (Hartmann and Briskorn [74]) with all the activities sequenced in serial, but with multiple extensions included at the same time (see discussion in Subsection 3.2.5), which is not customary in the literature of project scheduling problems.

3.2 PROBLEM DESCRIPTION

Before formally formulating the SRC-MSPP, we introduce notation and mathematical formulations for the SPP and RCSPP.

3.2.1 *Notation*

Sets

$\mathcal{S} = \{1, \ldots, |\mathcal{S}|\}$, set of index for the networks considered in the problem.

$\mathcal{N}_s$, set of nodes of the s-th network, $s \in \mathcal{S}$.

$\mathcal{A}_s$, set of arcs of the s-th network, $s \in \mathcal{S}$.

$\mathcal{C}_s$, set costs of the s-th network such that, for each arc $a = (m, n) \in \mathcal{A}_s$, $c_a \in \mathcal{C}_s$ represents the cost of using arc a to go from node $m \in \mathcal{N}_s$ to node $n \in \mathcal{N}_s$, $s \in \mathcal{S}$.

$\mathcal{G}_s = (\mathcal{N}_s, \mathcal{A}_s, \mathcal{C}_s)$, s-th network[1] of the problem, $s \in \mathcal{S}$.

$\mathcal{G}_{\mathcal{S}} = \{\mathcal{G}_s\}_{s \in \mathcal{S}}$, set of networks in the problem.

$\Lambda_s^+(n), \Lambda_s^-(n)$, set of incoming and outgoing arcs, respectively, of node $n \in \mathcal{N}_s$, $s \in \mathcal{S}$.

$\mathcal{R}$, set of constrained resources.

Parameters

o_s, d_s, origin (or source) and destination (or sink) nodes, respectively, of the s-th network, $o_s, d_s \in \mathcal{N}_s$, $s \in \mathcal{S}$.

q_a^r, consumption of resource $r \in \mathcal{R}$ by arc $a \in \{\mathcal{A}_s\}_{s \in \mathcal{S}}$.

W^r, availability of resource $r \in \mathcal{R}$.

Variables

$x_a = 1$, if arc $a \in \{\mathcal{A}_s\}_{s \in \mathcal{S}}$ is part of the solution, and 0 otherwise.

1 In this chapter, we use $\mathcal{G}$ instead of $\widetilde{\mathcal{G}}$ to denote networks.

3.2.2 *Preliminaries: Shortest Path Problem and Resource Constrained Shortest Path Problem*

For the sake of clarity and future comparison, we first expose the IP formulations of the shortest path problem and resource constrained shortest path problem. For the SPP, this is shown below[2].

$$\min \sum_{a \in \mathcal{A}} c_a x_a, \tag{3.1}$$

subject to:

$$\sum_{a \in \Lambda^-(n)} x_a - \sum_{a \in \Lambda^+(n)} x_a = \begin{cases} 1, & \text{if } n = o, \\ -1, & \text{if } n = d, \\ 0, & \text{otherwise.} \end{cases} \quad \forall n \in \mathcal{N} \tag{3.2}$$

$$x_a \in \{0, 1\} \quad \forall a \in \mathcal{A}. \tag{3.3}$$

In this formulation, (3.1) establishes the objective of minimizing the cost of the arcs selected, while (3.2) are the classical flow conservation constraints which guarantee that the arcs in the solution form a path from o to d.

As an example, consider the network shown in Figure 10, where below each arc is represented its associated cost. In the network, the shortest path from o to d is indicated with bold red arcs, and it has a total cost of 5 units.

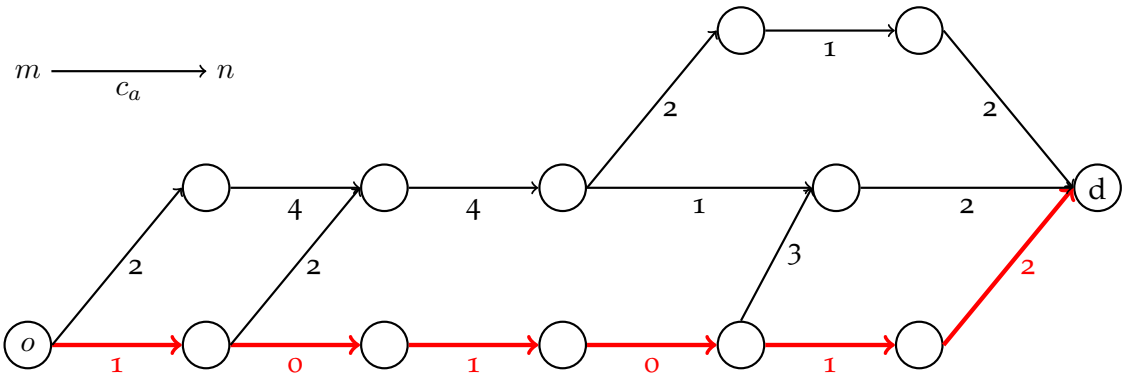


Figure 10: Illustrative example of the Shortest Path Problem.

2 Note that as both problems, the SPP and RCSPP, deal only with one network, the subscript s employed when defining notation in the previous subsection is not needed.

The IP formulation of the RCSPP is obtained by adding to model (3.1)-(3.3) the following capacity (knapsack) constraints:

$$\sum_{a \in \mathcal{A}} q_a^r x_a \leq W^r, \quad \forall r \in \mathcal{R}. \tag{3.4}$$

To illustrate the RCSPP, consider the SPP example exposed before, but now, with each arc consuming capacity from one limited resource. Assume that the availability of this resource is 22 units. This new situation, together with the arcs resource consumption, is depicted in Figure 11. From the data exposed in the figure, it is clear that the SP obtained for the example in Figure 10, when there was no resource limit, is no longer feasible (it has a cost of 5 and a resource consumption of 24 units). The optimal solution for this RCSPP example (shown with bold red arcs in the figure), has a cost of 7 and a resource consumption of 21 units.

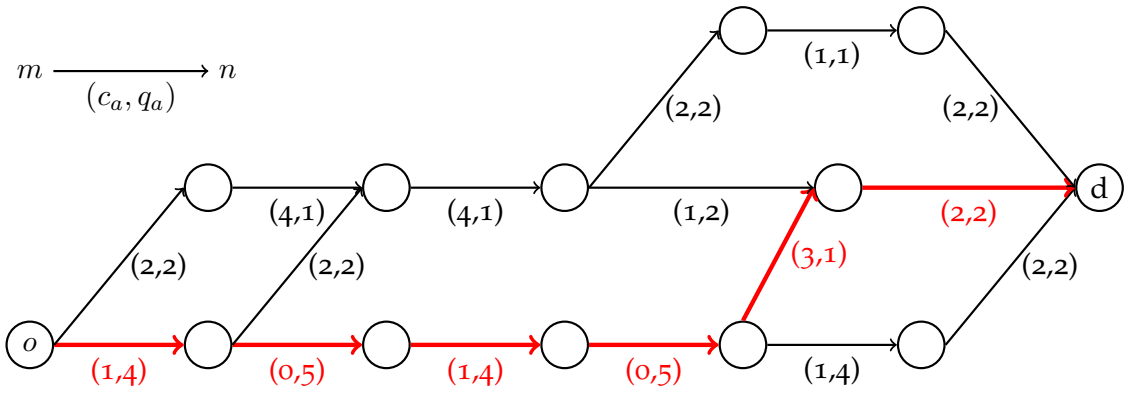


Figure 11: Illustrative example of the Resource Constrained Shortest Path Problem.

3.2.3 *Mathematical formulation*

The goal in the SRC-MSPP is to find for each network in $\mathcal{G}_S$ a path between a given source node o_s and a given sink node d_s, $s \in \mathcal{S}$, that minimizes the overall cost while not exceeding the capacity of a set of resources shared by all networks. The problem can be formulated as an integer program as follows:

$$\min \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}_s} c_a x_a, \tag{3.5}$$

subject to:

$$\sum_{a \in \Lambda_n^-(n)} x_a - \sum_{a \in \Lambda_n^+(n)} x_a = \begin{cases} 1, & \text{if } n = o_s, \\ -1, & \text{if } n = d_s, \\ 0, & \text{otherwise.} \end{cases} \quad \forall s \in \mathcal{S}, n \in \mathcal{N}_s \tag{3.6}$$

$$\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}_s} q_a^r x_a \leq W^r \quad \forall r \in \mathcal{R}, \tag{3.7}$$

$$x_a \in \{0, 1\}, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}_s. \tag{3.8}$$

The objective function (3.5) minimizes the cost of the arcs in the solution. Equations (3.6) are the flow conservation constraints for each network. These guarantee that the arcs in the solution of each network form a path from origin to destination. Constraints (3.7) enforce the resource limits. The model finishes with the integrality constraints (3.8).

3.2.4 *Key features*

We now highlight the main features of the shared resource constrained multi-shortest path problem, while pointing out the differences with the resource constrained shortest path problem.

First of all notice that in contrast to the RCSPP, where the arcs in a single path compete for a set of available resources, in the SRC-MSPP, the arcs in paths from multiple networks must share the constrained set of resources (see constraints (3.7)).

In second place, the structure of the solutions is also different. In the RCSPP, a solution (i.e., a single path) is infeasible when, for one or more resources, it consumes more than what is available. That is, the feasibility of a path depends only on the arcs in the path and it can be detected without any additional information (local conditions). By contrast, while the path for each single network in a solution to the SRC-MSPP may be feasible (local conditions), the entire solution may not be feasible when considering the aggregated resource consumption of all paths (global conditions). In other words, instead of infeasible paths, SRC-MSPP deals with infeasible combinations of paths.

To illustrate this point, consider the ATFM problem. Under typical circumstances, all elements in a network have enough capacity to handle at least one aircraft. Therefore, no route by itself is ever infeasible (local conditions). This means that a route cannot be eliminated when considered in isolation. The feasibility of a route depends on other routes in a chosen set. Therefore, there are certain route combinations that are feasible and others that are infeasible.

Note that the SRC-MSPP formulation admits that each arc, when used, consumes from all the available resources, i.e., $q_a^r > 0 \ , \forall s \in \mathcal{S}, a \in \mathcal{A}_s, r \in \mathcal{R}$. However, in real applications of the SRC-MSPP (such as train rescheduling) this is not the case. In those situations, each arc commonly represents an activity that to be completed requires from a particular subset of resources, but not from all of them. That is, in practice, the situation for the SRC-MSPP is such that each arc of each network does not consume from all the resources, but just from a subset.

To illustrate the characteristics of the SRC-MSPP, we elaborated an example based on the situation exposed in Figure 12. There are two networks where each of the arcs is accessing to two resources. Note that in the arcs, the resources consumed are indicated by superscripts. So for instance, 4^{r_1} means that the usage of the arc consumes 4 units of resource r_1. From the superscripts in the figure, it can be seen that in the example there is a total of 5 resources: $r_1, \ldots, r_5$. Assume that all the resources have a limit of 12 units, i. e., $W^r = 12 \ \forall r \in \{r_1, \ldots, r_5\}$. In the figure, the optimal solution for each network is shown with bold red arcs. For the top network the cost is 12, while for bottom one is 6. That is, the optimal solution has a total cost of 18 units. Below it is shown the desegregated resource consumption of the solution:

$$
\begin{aligned}
r_1 &: \overbrace{4+2}^{\text{top network}} + \overbrace{4}^{\text{bottom network}} = 10, \\
r_2 &: \overbrace{2+2}^{\text{top network}} + \overbrace{3+5}^{\text{bottom network}} = 12, \\
r_3 &: \overbrace{2+1}^{\text{top network}} + \overbrace{2}^{\text{bottom network}} = 5, \\
r_4 &: \overbrace{1+1+2}^{\text{top network}} + \overbrace{2+1}^{\text{bottom network}} = 7, \\
r_5 &: \overbrace{1+2+1}^{\text{top network}} + \overbrace{1+2}^{\text{bottom network}} = 7.
\end{aligned}
$$

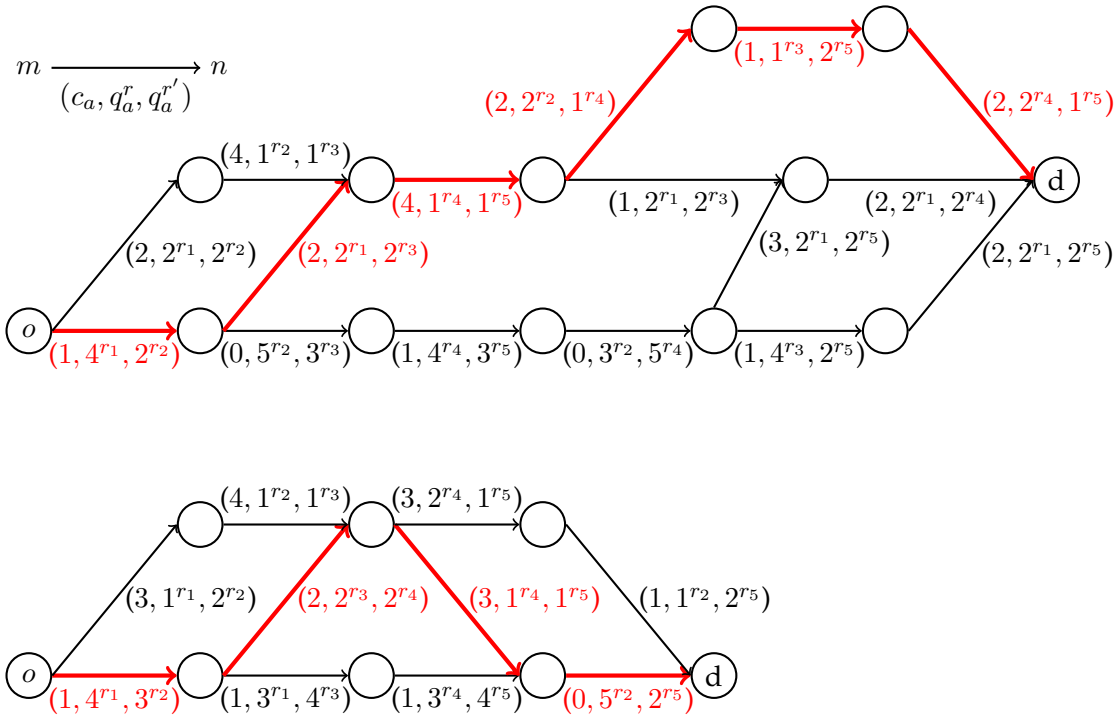


Figure 12: Illustrative example of the Shared Resource Constrained Multi-Shortest Path Problem.

From the example, the following two aforementioned characteristics become clearer: i) Any network's route may be feasible when considered in isolation, but not (necessarily) when combined with other networks' routes, and ii) As happens in scheduling problems, not all the arcs (activities) require from all the resources.

We conclude this subsection with two remarks. The first one is that, from the IP formulation (3.5)–(3.8), it is clear that the ATFMRP formulated through time-expanded networks (recall in Section 2.6) is a particular case where the SRC-MSPP arises in practice. In the following subsection we describe how the SRC-MSPP permits to model other types of scheduling problems. The second remarks is that, as previously said, the methods for the RCSPP are largely based on identifying and pruning dominated and infeasible paths, instead of combinations of paths, which make them not applicable to the problem addressed here. This, jointly with the fact that exact mathematical programming solvers are not able to solve some instances of the size found in practice, motivated us to study in the next section different solution methods for the SRC-MSPP.

3.2.5 *Application to project scheduling*

The SRC-MSPP can be applied to solve resource-constrained project scheduling problems. In its simplest version (see Kolisch and Padman [89], Kolisch and Hartmann [88], Hartmann and Briskorn [74] or Zheng and Wang [118]), the problem consists of finding a schedule that minimizes the project's completion time (i.e., the makespan), where a project is characterized by a set of activities with precedence relationships and resource availability. A feasible schedule of activities is such that it does not violate the precedence constraints and it does not consume more than the available resources.

Several extensions of this problem have been proposed in literature. For example, Confessore, Giordani, and Rismondo [32], Gonçalves, Mendes, and Resende [66] or Krüger and Scholl [90] address the problem with multiple projects and common resources needs.

The multi-mode extension occurs when several executions modes are considered, i.e., when the execution mode determines the resource requirements and completion times. Examples of this extension can be found in Kuster, Jannach, and Friedrich [91] or Kellenbrink and Helber [84], where the authors deal with an even more elaborate multi-mode scheme in which the activities required to complete a project might change (i.e., they are not fixed) and that the choice of modes is independent for each activity.

Lag times between activities have also been studied in project scheduling (Chassiakos and Sakellaropoulos [28], Klein [86], and Klein and Scholl [87]). The existence of lag times implies that, when one activity is finished, the following one has to wait some time before starting being executed.

Extensions related to when activities can be executed have also been developed. This has been studied considering time windows in which resources are not available (e.g., during holidays), as in Lu, Ren, Wang, and Zhu [96], and considering forbidden time periods in which activities cannot be executed, as in Drexl, Nissen, Patterson, and Salewski [42]. As discussed by Hartmann and Briskorn [74], in contrast to time windows where the availability of resources affects all activities, forbidden periods are particular to each activity.

The basic characteristics of the project scheduling problems that can be modeled with the SRC-MSPP are:

1. Multiple projects sharing resources whose availability might vary during the planning horizon.

2. The activities of the projects are sequenced in serial.

3. Each project has a starting time window and activities can only be scheduled within a given time interval.

4. There is no idle time between activities, i.e., once an activity has been completed, the following activity must start immediately.

We note, as discussed later on, that there are workarounds in order to apply SRC-MSPP to a scheduling problem that does not conform with the last characteristic.

The following additional characteristics may also be considered:

1. Time lags.

2. Multi-modes associated with individual activities.

3. Flexibility in the type of objective function, e.g., minimize the makespan or the sum of the cost of executing each activity on a given mode (Achuthan and Hardjawidjaja [1]).

We now discuss how the SRC-MSPP is capable of capturing the various characteristics of these project scheduling problems and its extensions.

3.2.5.1 *Modeling framework*

Consider a set of projects, each of them consisting of a set of serial activities. A project is completed once all its activities have been completed. Projects must be completed within a planning horizon represented as a set of discrete time periods. Each project must start within a time window and the duration of each activity has a minimum, a scheduled, and a maximum number of periods. For each time period that an activity is being processed, it consumes from a set of resources. Preemption of projects and activities is not allowed, i.e., once a project or activity has started, it must be completed. Idle time between activities can be modeled as dummy activities to be completed before starting the next activity. Completing a dummy activity in no time represents the case of not having considered the idle time option. Serial precedence relationships between projects might exist, such that, for a given sequence of projects, a project cannot start until the previous one has been fully completed.

The objective is to find a feasible schedule for all the activities included in the projects, so they are completed at minimum cost while respecting resources' capacity restrictions.

Therefore, the problem is to determine the starting processing time of each project and the processing speed of activities.

Since each project is made of a sequence of activities, it can be represented as a directed graph where arcs correspond to activities and nodes represent start and completion events. Figure 13 shows an example of a 4-activity project represented as an activity-on-arc graph, where node $n_{\tilde{a}}$ represents the completion of activity $\tilde{a}$ and the three values below each arc represent the number of time periods for the minimum, the scheduled, and the maximum duration of the activity. In this example, the four activities are scheduled to be completed in 1, 2, 3, and 1 time periods.

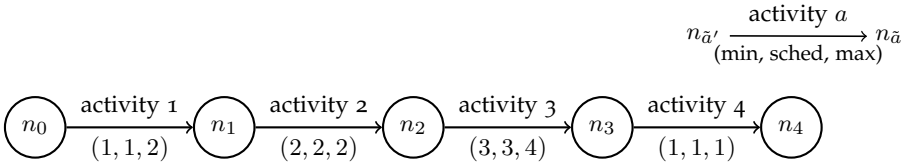


Figure 13: Example of a sequence of activities for a project.

The start time for each activity in a project can be derived from the possible start times of the project and the possible duration of the activity. For instance, assume that possible start times for the project in Figure 13 are $\{1, 2, 3\}$, then the possible start times for each activity are:

$$\{1, 2, 3\}, \{2, 3, 4, 5\}, \{4, \ldots, 7\}, \{7, \ldots, 11\},$$

and the possible completion times for the project are $\{8, \ldots, 12\}$.

An extended network is built by expanding each node in the original graph by the possible start times of each activity. For instance, activity 1 in Figure 13 has three possible start times (i.e., 1, 2, and 3). Node n_0 represents the start of activity 1. Therefore, the extended network will have three nodes representing the three possible start times for this activity (see nodes labeled (n_0, 1), (n_0, 2), and (n_0, 3) in Figure 14). This is done for all the nodes in the original network. Arcs in the extended network represent the possible activity duration. For instance, the arc from node (n_0, 2) to node (n_1, 4) represents the case when activity 1 starts in period 2 and has a duration of two periods.

The extended network has an origin (o) and a destination (d) node. The origin node is linked to every tail node of the first activity and the destination node is linked to every head node of the last activity, as shown in Figure 14.

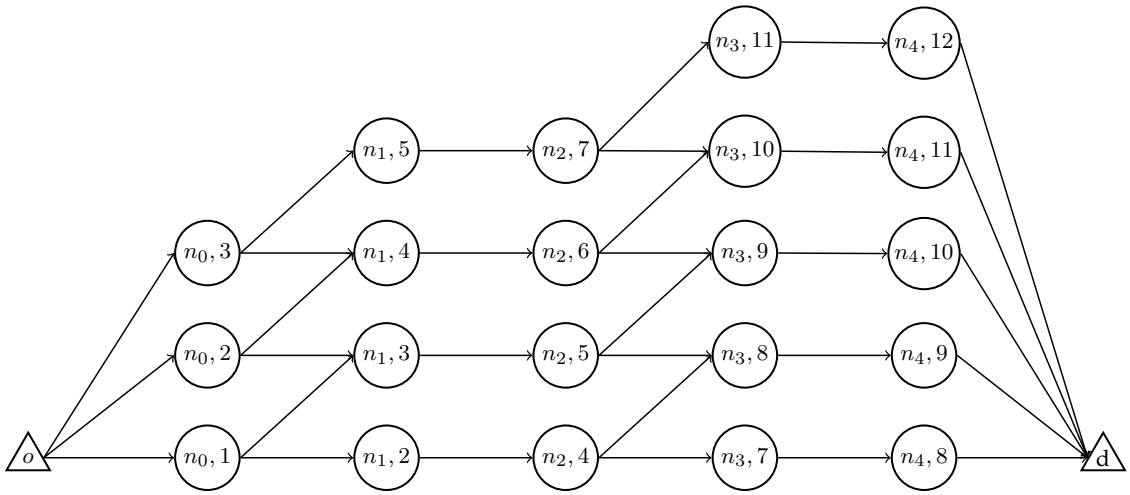


Figure 14: Extended network for the project in Figure 13.

A feasible schedule for a project corresponds to a path from the origin to the destination in its associated extended network. A value can be assigned to each arc of the extended network in such a way that it represents the cost of starting the activity in the time period indicated in its tail node and finishing the activity in the time period indicated in its head node. The resource unconstrained problem of finding the optimal schedule of activities is equivalent to finding the shortest path in the extended network. However, since in most settings, projects share resources, the constrained scheduling problem for all projects can be solved as an instance of the SRC-MSPP.

As can be seen, the idea of transforming the graph of serial activities into an extended network is equivalent to the transformation shown in Section 2.6 for the particular case of ATFM. Thus, alternative sequences of activities can also be easily modeled.

From the previous discussion it is clear that the ATFM problem can be viewed as a particular case of project scheduling problem. Let us specified this. In ATFM, each aircraft, which may perform multiple continued flights, can be viewed as a project. As multiple aircraft exist, the multiple project extension arises. For each aircraft, the sequence of serial activities to schedule are the airports and sectors that the aircraft has to reach/cross to carry out the flights that it has assigned. Recall that this route schedule conforms what is called a flight plan. The decisions conforming a flight plan are: the departure time (which is restricted to a time window), the usage of the main route or an alternative one (alternative sequences of activities existing to complete the project), the speed of the aircraft when traversing each air sector (execution modes), and

the landing time (completion time of the project). Furthermore, each aircraft must cross each sector within a given time interval (forbidden time periods, in project scheduling). When an aircraft lands, there is a setup time between flights (turnaround time) that can be equated to the lag time extension in project scheduling. Different objectives can be considered, such as minimizing the cost of the routes or minimizing late landings. Finally, sectors and airports conform the set of limited resources whose capacity may change along time.

3.3 SOLUTION METHODS

We now introduce two solution methods for the shared resource constrained multi-shortest path problem. The first one is matheuristic algorithm consisting of three phases: 1) Generation of feasible solutions, 2) Combination of solutions, and 3) Solution improvement. The second method consists of two Lagrangian Relaxations, one based on dualizing the capacity constraints, and the other based on dualizing a copy of the decision variables.

3.3.1 *Introduction to heuristic algorithms*

Multiple problems in combinatorial optimization are cataloged as $\mathcal{NP}$-complete or $\mathcal{NP}$-hard, meaning, from a practical point of view, that no polynomial-time algorithm is known to solve them. Nowadays, however, software and hardware permit to solve many instances of these problems to optimality (or nearly) using exact mathematical programming methods (e.g., by means of MIP solvers).

Despite this good news, there are still multiple problem instances for which MIP solvers fail to obtain a *good* solution in a *feasible* amount of time. For these situations, heuristics algorithms are employed. Heuristics algorithms (see in Martí, Pardalos, and Resende [98]), are solution methods oriented to find acceptable solutions in a reasonable amount of time. As a drawback, heuristic methods generally do not find the optimal solution, and they lack a procedure to prove optimality (Glover and Kochenberger [59]).

The cornerstone of heuristic optimization is the idea of local search, that is, given an initial solution, explore a neighborhood of this solution (i. e., a subset of the space of solutions) in such a way that a better solution is achieved. The procedure is repeated until no further improvement is obtained.

One of the inconveniences of heuristics algorithms is that they quickly get trapped in local optimum leading to low-quality solutions. To avoid this early termination issue, heuristics are usually embedded in a metaheuristic framework. From Sörensen and Glover [107]: "*A metaheuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms*". The goal of a metaheuristic is to guide the local search in such a way that: 1) It balances exploration and exploitation of the solution space, and 2) It is capable of escaping from local optimum. Some the most famous and tested matheuristics are (see more in Glover and Kochenberger [59] and Martí et al. [98]):

- Ant Colony Optimization (Dorigo [40], Dorigo, Maniezzo, and Colorni [41], and López-Ibáñez, Stützle, and Dorigo [94]).
- Genetic Algorithms (García-Martínez, Rodriguez, and Lozano [55] and Holland [75]).
- Tabu Search (Glover and Laguna [60] and Glover [62–64]).
- Variable Neighborhood Search (Hansen and Mladenović [73] and Mladenović and Hansen [99]).
- Scatter Search (Glover [61, 65] and Laguna and Martí [92]).
- Simulated Annealing (Geman and Geman [57], Hajek [71], Kirkpatrick, Gelatt, and Vecchi [85], and Nikolaev and Jacobson [101]).

Metaheuristics have been successfully applied to optimization problems, but the usage of one or another heavily depends on the problem to be solved (Wolpert and Macready [114]).

Because of the good performance of exact and metaheuristic methods, a new family of optimization algorithms based on a hybridization of both has arisen. These algorithms are denoted as matheuristic algorithms. As exposed in Fischetti and Fischetti [50], matheuristic algorithms are characterized by the central role of the mathematical model, which is the cornerstone around which the heuristic is built. Just the opposite of what happens in metaheuristics, where the mathematical model is disregard, or simply used for illustrative purposed or performance comparison with exact methods.

Due to its nature, matheuristic algorithms have been developed for specific problems, as usually do with metaheuristics (e.g., Billaut, Della Croce, and Grosso [21], Della

Croce, Salassa, and T'kindt [37], Fanjul-Peyro, Perea, and Ruiz [49], and Toffolo, Santos, Carvalho, and Soares [108]); and for becoming part of black-box solvers, as usually do with exact methods (e.g., Danna, Rothberg, and Le Pape [36] and Fischetti and Lodi [51]).

Matheuristic algorithms, as taking advantage of two perspectives, provide a powerful and interesting framework for developing algorithms to tackle optimization problems. In the matheuristic algorithm described in the next subsection, we employ a heuristic search to generate feasible solutions, while exact methods are used to explore subregions of the solution space that can improve the solutions.

3.3.2 *Matheuristic algorithm*

In this Subsection, we propose a three-phase matheuristic algorithm to solve the SRC-MSPP. We first give an overview of the procedure and then introduce notation and provide a detailed description of each phase.

In the first phase, a pool $\boldsymbol{\mathcal{X}} = \{\mathcal{X}^1, \ldots, \mathcal{X}^K\}$ of solutions is generated. A solution k in the pool is represented as $\mathcal{X}^k = \{\mathcal{X}_s^k\}_{s \in \mathcal{S}}$, where $\mathcal{X}_s^k \subseteq \mathcal{A}_s$ is a set of arcs defining a path from o_s to d_s in network $\mathcal{G}_s$. This phase attempts to generate solutions $\mathcal{X}^k \in \boldsymbol{\mathcal{X}}$ that are feasible with respect to the original model (3.5)-(3.8). However, as we discuss below, the resulting pool may include some solutions that violate one or more capacity constraints.

In the second phase, the solutions in $\boldsymbol{\mathcal{X}}$ are combined to obtain new (and perhaps better) feasible solutions, denoted by $\mathcal{X}^{\text{II}}$. This is done by solving the original IP model (3.5)-(3.8) with the arcs in $\boldsymbol{\mathcal{X}}$. By construction, $\mathcal{X}^{\text{II}}$ cannot be worse than the best feasible solution in $\boldsymbol{\mathcal{X}}$.

The third phase applies a local search to $\mathcal{X}^{\text{II}}$. The local search attempts to close the gap between the cost of each path in $\mathcal{X}^{\text{II}}$ and its corresponding lower bound. The lower bound for each network can be found by solving the resource constrained shortest path problem. For instances of the SRC-MSPP for which all paths are feasible for an individual network (e. g., in ATFM), the solution of the RCSPP is equivalent to the solution of the SPP. Therefore, the lower bound for an individual network can be found by solving the SPP instead of the RCSPP. The goal of the local search is to find improved solutions, which tend to be those for which the gaps are balanced across all networks. Solution $\mathcal{X}^{\text{III}}$ denotes the outcome of the local search.

Algorithm 1 shows a pseudo-code of the proposed procedure.

Algorithm 1 Matheuristic

1: **function** MATHEURISTIC
2: $\boldsymbol{\mathcal{X}} \leftarrow$ `GeneratePool`$(\mathcal{G}_{\mathcal{S}}, \mathcal{R}, maxIter, minIterIP, maxNets, penalty, \alpha, \beta, K)$;
3: $\mathcal{X}^{\text{II}} \leftarrow$ `SolutionCombination`$(\mathcal{G}_{\mathcal{S}}, \mathcal{R}, \boldsymbol{\mathcal{X}})$;
4: $\mathcal{X}^{\text{III}} \leftarrow$ `LocalSearch`$(\mathcal{G}_{\mathcal{S}}, \mathcal{R}, \mathcal{X}^{\text{II}}, \delta, \gamma)$;
5: **return** $\mathcal{X}^{\text{III}}$;
6: **end function**

Notation for the algorithm

Parameters

$maxIter$, maximum number of failed attempts to generate a feasible solution.

$minIterIP$, minimum number of failed attempts between calls to the IP solver. The IP solver is used, as shown later, to help the heuristic to find feasible solutions.

$maxNets$, maximum number of networks with penalized arcs in the previous iteration to consider using the IP solver.

$penalty$, value to penalize the usage of arcs contributing to solution infeasibility.

α, probability of penalizing a set of arcs that has been identified as contributing to the solution infeasibility.

β, percentage of networks, whose arcs have not been penalized in the previous iteration, that are fixed to their current paths when calling the IP solver.

K, number of solutions to generate in the first phase of the algorithm.

Δ_s, difference, for the s-th network, between its solution cost after the second phase of the algorithm and its corresponding lower bound, $s \in \mathcal{S}$.

δ, number of networks for which the algorithm tries to improve their solution in the third phase. The networks with larger Δ_s values are the ones to be improved.

γ, number of networks that the algorithm will use to trade resources with the δ networks above.

Sets

$\mathcal{X}_s^k \subseteq \mathcal{A}_s$, set of arcs defining a path from o_s to d_s in the s-th network for the k-th solution generated, $s \in \mathcal{S}$, $k \in \{1, \ldots, K\}$.

$\mathcal{X}^k = \{\mathcal{X}_s^k\}_{s \in \mathcal{S}}$, k-th solution generated, $k \in \{1, \ldots, K\}$.

$\boldsymbol{\mathcal{X}} = \{\mathcal{X}^1, \ldots, \mathcal{X}^K\}$, set of solutions generated in the first phase of the algorithm. For analogy with other metaheuristics such as Scatter Search, we will refer to $\boldsymbol{\mathcal{X}}$ as pool instead of set.

$\mathcal{X}^{II}$, solution generated in the second phase of the algorithm.

$\mathcal{X}^{III}$, solution generated in the third and last phase of the algorithm.

$\mathcal{X}^{LB}$, lower-bound solution to the problem.

$\mathcal{G}_s^* = (\mathcal{V}_s, \mathcal{A}_s, \mathcal{C}_s^*)$, s-th network of the problem with the set of costs different (due to the penalization process of the algorithm) than the original network $\mathcal{G}_s$, $s \in \mathcal{S}$.

$\mathcal{G}_{\mathcal{S}}^* = \{\mathcal{G}_s^*\}_{s \in \mathcal{S}}$, set of networks with modified costs in the problem. $\mathcal{G}_s^* = \mathcal{G}_s$ for those networks whose costs have not been modified.

$\mathcal{A}_s^r \subseteq \mathcal{A}_s$, subset of arcs in network $s \in \mathcal{S}$ that use resource $r \in \mathcal{R}$. Note that an arc may belong to more than one $\mathcal{A}_s^r$.

$\mathcal{S}^* \subseteq \mathcal{S}$, network indexes with at least one penalized arc in the previous iteration.

$\mathcal{S}^\beta$, network indexes as described for parameter β above.

$\mathcal{S}^\delta$, network indexes as described for parameter δ above.

$\mathcal{S}^\gamma$, network indexes as described for parameter γ above.

Example for the Algorithm

For the sake of clarity when exposing function `GeneratePool`, the most complicated one in the algorithm, we elaborated an example based on the situation depicted in Figure 15. There are 3 networks, where the bold red arcs indicate the shortest paths from o to d. In the example, there is only one resource with capacity equal to 1, and the only arcs that may access it are those indicated in the figure, i. e., a_1, a_2, a_3, a_4 and a_5. In the example,

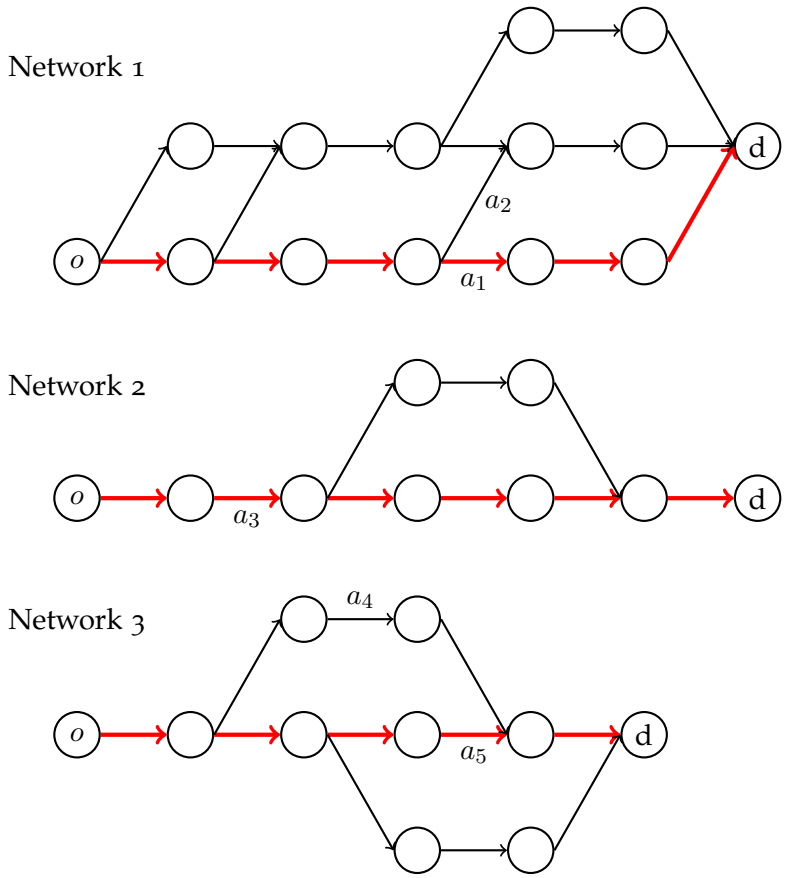


Figure 15: Illustrative example of solving the SRC-MSPP with the algorithm (part I).

each arc consumes 1 unit of the resource. Therefore, the model will have the following capacity constraint:

$$\overbrace{x_{a_1} + x_{a_2}}^{\text{Network 1}} + \overbrace{x_{a_3}}^{\text{Network 2}} + \overbrace{x_{a_4} + x_{a_5}}^{\text{Network 3}} \leq 1. \tag{3.9}$$

Note that the current SP solution is not feasible because the consumption of the limited resource is 3 units ($x_{a_1} + x_{a_3} + x_{a_5}$). With this situation, let us described the algorithm and how it would work for the example.

3.3.2.1 *Phase I: The `GeneratePool` Function*

The goal of the first phase of our procedure is to generate a pool of feasible solutions $\mathcal{X}$. The procedure (shown in Algorithm 2) starts with the solution of SPP[3] (i.e., model (3.1)–

3 We tackle SRC-MSPP instances for which the W^r values are such that no individual shortest path violates the capacity constraints, making the RCSPP for each network equivalent to solving the SPP. Additionally,

(3.3)) for each network (line 2). The total cost associated with the collection of all the obtained shortest paths, $\mathcal{X}^{\text{LB}}$, is a lower bound for the original problem. If this collection of shortest paths meets all the capacity constraints, then $\mathcal{X}^{\text{LB}}$ is an optimal solution to the original problem and the procedure terminates (lines 3–5). Otherwise (i.e., at least one capacity constraint has been violated), a for-loop to generate K feasible solutions is executed (lines 8–11). At each iteration of the loop, the `FeasibleSol` function attempts to generate a feasible solution using $\mathcal{X}^{\text{LB}}$ as a starting seed. This for-loop is amenable to parallel execution in the presence of multiple cores because the calls to `FeasibleSol` are independent. `FeasibleSol` is a non-deterministic iterative procedure that attempts to create a feasible solution $\mathcal{X}^k$ from a starting solution $\mathcal{X}^{\text{LB}}$ that does not meet the capacity constraints of the original model (3.5)-(3.8). Due to its non-deterministic nature, it is expected that `FeasibleSol` will generate a different solution every time is called.

Algorithm 2 Generating pool $\boldsymbol{\mathcal{X}}$

1: **function** GENERATEPOOL($\mathcal{G}_{\mathcal{S}}, \mathcal{R}, maxIter, minIterIP, maxNets, penalty, \alpha, \beta, K$)
2: $\quad\mathcal{X}^{\text{LB}} \leftarrow$ `ShortestPath`($\mathcal{G}_{\mathcal{S}}$);
3: $\quad feasible \leftarrow$ `CheckFeasibility`($\mathcal{R}, \mathcal{X}^{\text{LB}}$);
4: $\quad$**if** $feasible ==$ `true` **then**
5: $\quad\quad\boldsymbol{\mathcal{X}} \leftarrow \{\mathcal{X}^{\text{LB}}\}$;
6: $\quad$**else**
7: $\quad\quad\boldsymbol{\mathcal{X}} \leftarrow \emptyset$;
8: $\quad\quad$**for** $k = 1, \ldots, K$ **do**
9: $\quad\quad\quad\mathcal{X}^k \leftarrow$ `FeasibleSol`($\mathcal{G}_{\mathcal{S}}, \mathcal{R}, \mathcal{X}^{\text{LB}}, maxIter, minIterIP, maxNets, penalty, \alpha, \beta$);
10: $\quad\quad\quad\boldsymbol{\mathcal{X}} \leftarrow \boldsymbol{\mathcal{X}} \cup \{\mathcal{X}^k\}$;
11: $\quad\quad$**end for**
12: $\quad$**end if**
13: $\quad$**return** $\boldsymbol{\mathcal{X}}$;
14: **end function**

Algorithm 3 shows the steps associated with the function that attempts to produce a feasible solution from the collection of shortest paths $\mathcal{X}^{\text{LB}}$. The procedure identifies, for the current solution, the arcs that are contributing to the infeasibility of the solution and adds a penalty to the cost of a random subset of these arcs. Then, the SPP is solved for each penalized network (denoted as $\mathcal{G}_s^*$), producing shortest paths that exclude most or all of the penalized arcs. These steps are repeated in search of a feasible solution. If this fails, a final attempt to find a feasible solution is made by way of solving a reduced

since the networks in our computational testing are acyclic, we solve the SPP using Bellman principle of optimality.

version of the original integer programming model. In this reduced version, some of the arcs are fixed and only a subset of the arcs is included as decision variables in the model. This step does not guarantee a feasible solution because the variable fixing may render the model infeasible. The search for a feasible solution ends once one is found or after a specified limit on the number of failed attempts. For reasons that will become clear below, adding an infeasible solution to pool $\boldsymbol{\mathcal{X}}$ is not an issue as long as the pool contains at least one feasible solution.

Algorithm 3 Producing a feasible solution from $\mathcal{X}^{\text{LB}}$

function FEASIBLESOL($\mathcal{G}_{\mathcal{S}}, \mathcal{R}, \mathcal{X}, maxIter, minIterIP, maxNets, penalty, \alpha, \beta$)
  $\mathcal{A}_{\mathcal{S}}^{\mathcal{R}} \leftarrow \{\mathcal{A}_s^r \mid s \in \mathcal{S}, r \in \mathcal{R}\}$;
  $\mathcal{G}_{\mathcal{S}}^* \leftarrow \mathcal{G}_{\mathcal{S}}$;
  $feasible \leftarrow$ `false`;
  $nIter \leftarrow 0$;
  $nIterIP \leftarrow minIterIP$;
  while $feasible ==$ `false` & $nIter \leq maxIter$ **do**
    $\mathcal{S}^*, \mathcal{G}_{\mathcal{S}}^*, \mathcal{A}_{\mathcal{S}}^{\mathcal{R}} \leftarrow$ `PenalizeArcs`$(\mathcal{X}, \mathcal{G}_{\mathcal{S}}^*, \mathcal{R}, \mathcal{A}_{\mathcal{S}}^{\mathcal{R}}, penalty, \alpha)$;
    if $\mathcal{S}^* == \emptyset$ **then**
      $\mathcal{A}_{\mathcal{S}}^{\mathcal{R}} \leftarrow \{\mathcal{A}_s^r \mid s \in \mathcal{S}, r \in \mathcal{R}\}$;
    else
      if $|\mathcal{S}^*| \leq maxNets$ & $nIterIP \geq minIterIP$ **then**
        $\mathcal{X}^{trial} \leftarrow$ `SolveIPModel`$(\mathcal{R}, \mathcal{G}_{\mathcal{S}}, \mathcal{S}^*, \mathcal{X}, \beta)$;
        $feasible \leftarrow$ `CheckFeasibility`$(\mathcal{R}, \mathcal{X}^{trial})$;
        if $feasible == true$ **then**
          $\mathcal{X} \leftarrow \mathcal{X}^{trial}$;
        else
          $nIterIP \leftarrow 0$;
        end if
      else
        $\mathcal{X} \leftarrow$ `ShortestPath`$(\mathcal{G}_{\mathcal{S}}^*)$;
        $feasible \leftarrow$ `CheckFeasibility`$(\mathcal{R}, \mathcal{X})$;
      end if
    end if
    $nIterIP \leftarrow nIterIP + 1$;
    $nIter \leftarrow nIter + 1$;
  end while
  return $\mathcal{X}$;
end function

The `FeasibleSol` function takes as input the set of networks ($\mathcal{G}_{\mathcal{S}}$), the set of resources ($\mathcal{R}$), the collection of shortest paths associated with the lower bound ($\mathcal{X} = \mathcal{X}^{\text{LB}}$), the

maximum number of failed attempts ($maxIter$), the number of failed attempts between calls to the IP solver for the reduced model ($minIterIP$), maximum number of networks with penalized arcs in the last iteration to consider using the IP solver ($maxNets$), the penalty value ($penalty$), the probability of penalizing a set of arcs that has been identified as contributing to the solution infeasibility (α), and the percentage of networks, whose arcs have not been penalized in the last iteration, that are fixed to their current paths when calling the IP solver (β).

Lines 2–6 initialize the local elements of the `FeasibleSol` function. Let $\mathcal{A}_s^r \subseteq \mathcal{A}_s$ be the subset of arcs in network s that use resource r. We point out that an arc may belong to more than one $\mathcal{A}_s^r$. $\mathcal{A}_{\mathcal{S}}^{\mathcal{R}}$ contains the unpenalized subsets $\mathcal{A}_s^r$. At the beginning, no arc subsets have been penalized and therefore all arcs subsets are included in $\mathcal{A}_{\mathcal{S}}^{\mathcal{R}}$. In the example, $\mathcal{A}_{\mathcal{S}}^{\mathcal{R}} = \big\{\{a_1, a_2\}, \{a_3\}, \{a_4, a_5\}\big\}$. $\mathcal{G}_{\mathcal{S}}^* = \{(\mathcal{N}_s, \mathcal{A}_s, \mathcal{C}_s^*)\}_{s \in \mathcal{S}}$ consists of all the penalized networks and starts as a copy of $\mathcal{G}_{\mathcal{S}}$, indicating that at the beginning no arcs have been penalized (i.e., $\mathcal{C}_s^* = \mathcal{C}_s$). The $feasible$ Boolean variable keeps track of the feasibility of the solution obtained at the current iteration. The $nIter$ counter is the number of failed attempts to produce a feasible solution, and $nIterIP$ counts the number of failed attempts since the last time the IP model was executed.

After the initialization, a while-loop (lines 7–27) that attempts to create a feasible solution out of the current $\mathcal{X}$ begins. In the first step of the loop, the `PenalizeArcs` function (see pseudo-code and detailed description at the end of this Phase I exposition) seeks to identify and penalize (using a non-deterministic procedure) subsets of arcs that are contributing to infeasibility in the current solution. For that, `PenalizeArcs` creates a list of resources for which their capacity is exceeded by the current solution. We will refer to this as the list of infeasible resources. Then, for each resource r^* in the list of infeasible resources, the procedure loops through all networks and with probability α, penalizes each unpenalized subset $\mathcal{A}_s^{r^*} \in \mathcal{A}_{\mathcal{S}}^{\mathcal{R}}$ if at least one arc in $\mathcal{A}_s^{r^*}$ is also in $\mathcal{X}$. The subset penalization consists of adding a $penalty$ value to the current cost of all arcs in $\mathcal{A}_s^{r^*}$. That is, the penalization process is cumulative.

Note that some of the arcs in a penalized subset $\mathcal{A}_s^{r^*}$ might not be in the current solution $\mathcal{X}$. The reason for penalizing these inactive arcs is to decrease their attractiveness in subsequent iterations, since their addition to the solution may cause a resource that has been made feasible to become infeasible again. The penalization process may stop before considering all subsets that use an infeasible resource. This happens when we have penalized enough subsets that the sum of their resource requirements is at least as large as the amount by which the resource is infeasible.

`PenalizeArcs` returns the set of network indexes with at least one penalized arc in the current iteration ($\mathcal{S}^* \subseteq \mathcal{S}$), the set of networks with penalized costs ($\mathcal{G}^*_{\mathcal{S}}$), and an updated $\mathcal{A}^{\mathcal{R}}_{\mathcal{S}}$ set in which the penalized arc subsets have been removed. The latter is to not penalize the same subsets $\mathcal{A}^r_s$ more than once and to foster diversity in the search. Note that, since an arc can consume more than one resource, removing a subset does not completely remove an arc. That is, an arc that has been penalized for consuming one resource may belong to an unpenalized subset of another resource. Thus, an arc can be penalized multiple times, one per infeasible constraint to which it belongs. The algorithm allows penalizing an arc multiple times to further discourage the usage of those arcs with the largest contribution to the infeasibility of the solution

In the example introduced before, this first part of the algorithm will work as follows. As there is only one infeasible resource, the list of infeasible resources is formed by a single element. For that resource, the next steps happen:

Step 1: Check conditions for penalizing arcs in Network 1:

1. $\mathcal{A}^{r^*}_1 \in \mathcal{A}^{\mathcal{R}}_{\mathcal{S}}$? That is, have the arcs in Network 1 (which can access to resource r^*) not been penalized before for using r^*? If true, check the next condition; else, go to the next step.
2. Is there at least one arc in $\mathcal{A}^{r^*}_1$ also in $\mathcal{X}$? That is, Is there at least one arc of Network 1 consuming the resource and, therefore, contributing to infeasibility in constraint (3.9)? If true, continue; else, go to the next step.

In this case, both conditions are true, so arcs a_1 and a_2 are candidates to be penalized. Now the algorithm randomly decides if penalizing them or not. Consider that it does. Then, both arcs are penalized and subset $\mathcal{A}^{r^*}_1$ is removed from $\mathcal{A}^{\mathcal{R}}_{\mathcal{S}}$. Now this set becomes $\mathcal{A}^{\mathcal{R}}_{\mathcal{S}} = \{\{a_3\}, \{a_4, a_5\}\}$.

Step 2: Check conditions for penalizing arcs in Network 2 as in the previous step. In this case, both conditions hold again, so arc a_3 is a candidate to be penalized. Consider that at the random step the algorithm decides to penalize the arc. Then arc a_3 is penalized and subset $\mathcal{A}^{r^*}_2$ removed from $\mathcal{A}^{\mathcal{R}}_{\mathcal{S}}$. After that, $\mathcal{A}^{\mathcal{R}}_{\mathcal{S}} = \{\{a_4, a_5\}\}$

Step 3: If the current penalized arcs (a_1, a_2 and a_3) were no longer used after the penalization, the constraint would become feasible and penalizing more would be counterproductive: We would discourage other networks from using their shortest paths when unnecessary. Therefore, at this point, as specified in the algorithm description, the penalization of this resource stops. Obviously, for the situation of

the example, a_3 will always be used (it is the only way for Network 2 to reach its destination), so the early stopping does not have the desired effect. This situation was shown on purpose in the example to illustrate that this stopping rule also has a heuristic nature aiming at penalizing as few as possible.

Step 4: If there were more elements in the list of infeasible resources, the algorithm would repeat the previous steps for them. As in the example there are not, this first penalization round finishes. Note that for this iteration $\mathcal{S}^* = \{$Network 1, Network 2$\}$.

Since `PenalizeArcs` removes penalized arc subsets from $\mathcal{A}^{\mathcal{R}}_{\mathcal{S}}$ after each iteration, a point may be reached in which $\mathcal{A}^{\mathcal{R}}_{\mathcal{S}}$ is of a size that `PenalizeArcs` might return an empty $\mathcal{S}^*$ set. This can occur, for example, if just a few sets remain in $\mathcal{A}^{\mathcal{R}}_{\mathcal{S}}$ and the algorithm decides (in the random step) to not penalize them. If this occurs, $\mathcal{A}^{\mathcal{R}}_{\mathcal{S}}$ is reset to include all arcs (line 10). This is to avoid the algorithm getting trapped in an infeasible solution.

As long as $\mathcal{S}^*$ is not empty, an attempt to find a feasible solution is made. The attempt takes on two different forms. An exact method solves a reduced version of the integer programming model (lines 12–20) or new shortest paths are found for the networks in $\mathcal{G}^*_{\mathcal{S}}$ (lines 21–22). The exact method is used when the number of penalized networks is small enough (i.e.: $|\mathcal{S}^*| \leq maxNets$) and the number of attempts to reach feasibility by recomputing the shortest paths reaches $minIterIP$. The exact method solves a reduced version of the original model (3.5)–(3.8) in which we fix a large percentage of the variables in the original problem. We start by selecting $\beta\%$ of the networks[4] in $\mathcal{S} \setminus \mathcal{S}^*$. The selection is made balancing solution quality (networks with worse objective function are selected) and diversity (networks are selected at random). We alternate the use of these two criteria. We denote this set as $\mathcal{S}^\beta$. Then, variables in the set $\{x_a \,|\, a \in \{\mathcal{A}_s\}_{s\in\mathcal{S}^\beta}\}$ are fixed to 1 if $a \in \mathcal{X}$, and to 0 otherwise. This means that the paths for the networks in $\mathcal{S}^\beta$ are fixed as dictated by the current solution. Therefore, the only variables in the reduced model are those associated with the arcs in $\{\mathcal{A}_s\}_{s\in\mathcal{S}\setminus\mathcal{S}^\beta}$. The reduced model uses the original cost values for the objective function calculation and a resource availability that is reduced by the resources requirements of the fixed variables.

4 In our original algorithmic design, we fixed all paths in networks without penalized arcs, i.e., all networks in $\mathcal{S} \setminus \mathcal{S}^*$. However, this proved to be too restrictive, frequently making the reduced model infeasible. The β parameter allows us to include some additional networks in the formulation and increase the flexibility of the model.

In the case of our example, after the penalization step, the SP would be recomputed. The new resulting solution is shown with bold red arcs in Figure 16. There it can be seen that the new solution is still infeasible, having only changed for Network 1. Note that if a_2 had not been penalized, the new shortest path of Network 1 might have included a_2, making that Network 1 continued contributing to infeasibility. Hence the importance of penalizing subsets instead of individual arcs.

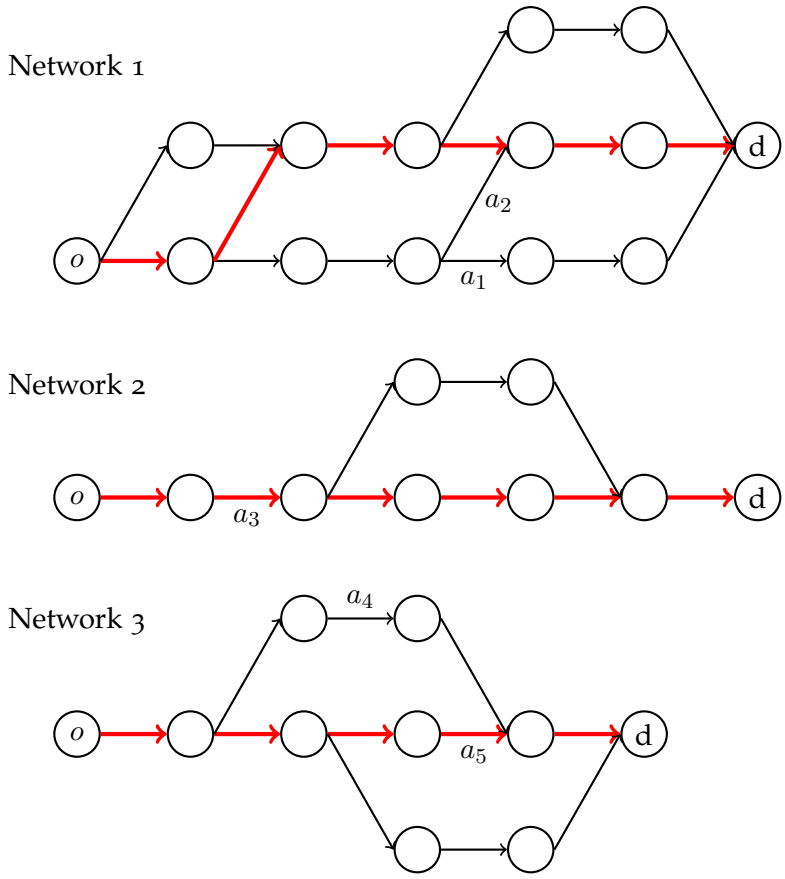


Figure 16: Illustrative example of solving the SRC-MSPP with the algorithm (part II).

After obtaining a new solution (by either of the methods described above), the procedure checks for feasibility (lines 14 and 22). If the solution is feasible, then the procedure ends and returns $\mathcal{X}$. If the solution is not feasible, then the current solution changes only if the SPP method was used to find it. That is, the current solution is not changed when the IP model is used, but no feasible solution exists due to the variable fixing. The process ends after $maxIter$ failed attempts and it returns the current infeasible solution. If the solution pool does not include any feasible solutions, then `SolutionCombination`

function in Algorithm 1 might not return a feasible solution. The probability of observing this, however, decreases with the size of the solution pool. In fact, with the size that we used in our computational experiences, the procedure never encountered this situation.

For our example, given that the solution keeps being infeasible, the next iteration of the while loop would be as follows.

Step 1: Penalization step. Because of the previous discussion, at this iteration only the arcs of Network 3 are candidates to be penalized. Consider that at the random step, the algorithm decides to penalize them. Then, a_4 and a_5 are penalized and $\mathcal{A}_3^{r^*}$ is removed from $\mathcal{A}_{\mathcal{S}}^{\mathcal{R}}$. Note that if the arcs of Network 3 had not been penalized, set $\mathcal{A}_{\mathcal{S}}^{\mathcal{R}}$ would have been reset at the next iteration because $\mathcal{S}^*$ would be empty.

Step 2: As $\mathcal{S}^* = \{\text{Network 3}\}$ is not empty, the SP is recomputed. On this occasion, the solution would be the same as in Figure 16, except for Network 3, which will reach d using the arcs at the bottom of the network.

Step 3: As the new solution is feasible (only arc a_3 consumes the resource), this call to `FeasibleSol` ends.

Before describing additional elements of the proposed procedure, we would like to make a brief comment on the use of the IP model to solve the reduced problem. Our original design of the `FeasibleSol` function did not include this component. We added it after preliminary computational experiences showed that, for small $|\mathcal{S}^*|$ values, feasibility could be reached faster and with better solution quality by solving the reduced problem instead of continuing to penalize arcs and finding the revised shortest paths. In this design, the IP exact solver is meant to be invoked occasionally. That is, it is not meant to be the first option. Therefore, the values of the $maxNets$ and $minIterIP$ parameters must be chosen accordingly. The $maxNets$ parameter controls the size of the subproblem. Given that we are using an exact solver, we need to limit the size of the model that we are asking the solver to tackle. The $minIterIP$ parameter is a proxy for directly monitoring the changes in $\mathcal{S}^*$. Note that if the IP model fails to produce a feasible solution with a particular $\mathcal{S}^*$, it only makes sense to invoke it again after $\mathcal{S}^*$ has experienced some changes. Instead of keeping track of the changes in $\mathcal{S}^*$, we experimentally adjusted the value of $minIterIP$ to allow the arc penalization function to change the composition of $\mathcal{S}^*$.

Pseudo-code for function `PenalizeArcs`

To conclude this first phase of the algorithm, we show in Algorithm 4 the pseudocode for function `PenalizeArcs`, employed before to identify and penalize subsets of arcs that are contributing to infeasibility in the current solution.

Algorithm 4 Penalizing arcs contributing to infeasibility

1: **function** PENALIZEARCS($\mathcal{X}$, $\mathcal{G}^*_{\mathcal{S}}$, $\mathcal{R}$, $\mathcal{A}^{\mathcal{R}}_{\mathcal{S}}$, $penalty$, α)
2: $\mathcal{S}^* \leftarrow \emptyset$;
3: $\mathcal{R}^* \leftarrow \{r \in \mathcal{R} \mid \sum_{a \in \mathcal{X}} q^r_a - W^r > 0\}$;
4: **for** $r \in \mathcal{R}^*$ **do**
5: $total \leftarrow \sum_{a \in \mathcal{X}} q^r_a - W^r$;
6: **for** $s \in \mathcal{S}$ **do**
7: **if** $\mathcal{A}^r_s \in \mathcal{A}^{\mathcal{R}}_{\mathcal{S}}$ & $\mathcal{A}^r_s \cap \mathcal{X} \neq \emptyset$ **then**
8: **if** `Rand()` $< \alpha$ **then**
9: $c^*_a \leftarrow c^*_a + penalty,\ \forall\ a \in \mathcal{A}^r_s, c^*_a \in \mathcal{C}^*_n$;
10: $\mathcal{A}^{\mathcal{R}}_{\mathcal{S}} \leftarrow \mathcal{A}^{\mathcal{R}}_{\mathcal{S}} \setminus \{\mathcal{A}^r_s\}$;
11: $\mathcal{S}^* \leftarrow \mathcal{S}^* \cup \{s\}$;
12: $total \leftarrow total - \sum_{a \in \mathcal{A}^r_s \cap \mathcal{X}} q^r_a$;
13: **if** $total \leq 0$ **then**
14: break;
15: **end if**
16: **end if**
17: **end if**
18: **end for**
19: **end for**
20: **return** $\mathcal{S}^*$, $\mathcal{G}^*_{\mathcal{S}}$, $\mathcal{A}^{\mathcal{R}}_{\mathcal{S}}$;
21: **end function**

In the first two lines of the algorithm, $\mathcal{S}^*$ will save the networks' indexes whose arcs are penalized, and $\mathcal{R}^*$ is the set of resources whose capacity is exceeded by the current solution.

Then, the algorithm loops for the elements in $\mathcal{R}^*$ and $\mathcal{S}$. In each loop $r \in \mathcal{R}^*$, the local variable $total$ (line 5) represents the amount by which resource r is exceeded.

$\mathcal{A}^r_s$ is a candidate to be penalized if: 1) It has not been penalized in previous iterations ($\mathcal{A}^r_s \in \mathcal{A}^{\mathcal{R}}_{\mathcal{S}}$), and 2) At least one of its arcs is in the current solution ($\mathcal{A}^r_s \cap \mathcal{X} \neq \emptyset$). If both conditions hold, then subset $\mathcal{A}^r_s$ is penalized with probability α.

When penalizing, the cost of the arcs in $\mathcal{A}^r_s$ are increased by $penalty$ (line 9), subset $\mathcal{A}^r_s$ is removed from $\mathcal{A}^{\mathcal{R}}_{\mathcal{S}}$, set $\mathcal{S}^*$ is updated, and $total$ is reduced by how much the penalized subset contributes to infeasibility. If $total$ becomes less or equal to zero, then

the penalization of the resource stops. This early stopping rule is based on the idea that, if the penalized arcs were not used any longer, then the constraint would not be violated, and penalizing more would be counterproductive (it would discourage other networks form keep using their best current path, when unnecessary).

Note that penalizing by subsets is valid as far as not every arc of every network consumes from all the resources (recall discussion in Subsection 3.2.4). If that was not the case, then penalizing through subsets would not be recommended. This is because all the arcs in a network would belong to the same subset and, therefore, they would be penalized at the same time, making no difference between penalizing or not. For those cases, the penalization has to be done only for the arcs that belong to the current solution, not for the subsets with at least one arc in the current solution.

At the end, the algorithm returns sets $\mathcal{S}^*$, $\mathcal{G}^*_{\mathcal{S}}$, $\mathcal{A}^{\mathcal{R}}_{\mathcal{S}}$.

3.3.2.2 *Phase II: The `SolutionCombination` Function*

As discussed at the beginning of Subsection 3.3.2, the `SolutionCombination` function combines the solutions in $\boldsymbol{\mathcal{X}}$ and produces a new solution $\mathcal{X}^{\text{II}}$, as long as at least one solution in the pool is feasible. The combination process consists of solving a reduced version of the original IP model (3.5)–(3.8), where the only variables are those associated with arcs in the pool of solutions. That is, the set of variables in the model is $\{x_a\}_{a\in\boldsymbol{\mathcal{X}}}$.

This form of combination of solutions has two key properties. First, the resulting solution $\mathcal{X}^{\text{II}}$ is at least as good as the best feasible in $\boldsymbol{\mathcal{X}}$. In our computational experiments, we observed that $\mathcal{X}^{\text{II}}$ was always better than any of the solutions in $\boldsymbol{\mathcal{X}}$. Second, $\mathcal{X}^{\text{II}}$ is guaranteed to be no worse than any solution found as a combination of the paths associated with the solutions in $\boldsymbol{\mathcal{X}}$. This is due to the generation of $\mathcal{X}^{\text{II}}$ by a combination of arcs (instead of paths) that allows forks and joints to be produced at the nodes of the subnetwork induced by the arcs in $\boldsymbol{\mathcal{X}}$, leading to paths that are not in the pool of solutions. Our experiments with various designs led us to conclude that this combination method is superior to others in terms of the quality of the combined solution and the computational time to find it.

3.3.2.3 *Phase III: The `LocalSearch` Function*

In preliminary experimentation we observed that, for a given solution $\mathcal{X}^{\text{II}}$ resulting from the combination method, some networks used paths whose cost was much higher

(relative to the known lower bound) than the cost associated with other networks. For each network s, let $\Delta_s = \sum_{a \in \mathcal{A}_s} c_a(x_a^{\text{II}} - x_a^{\text{LB}})$ denote this difference.

We concluded that the $\mathcal{X}^{\text{II}}$ solutions tend to be unbalanced, with a relatively small number of networks with much larger Δ_s values than others. We therefore developed the `LocalSearch` function taking into account the structure of the $\mathcal{X}^{\text{II}}$ solutions. In particular, `LocalSearch` focuses on improving the paths in networks with relatively large Δ_s values. This is done at the possible expense of worsening the delta values of other networks. The search, in other words, is for a balanced solution. That is, one for which the collection of Δ_s values for all networks have less variance.

Algorithm 5 Improving solution $\mathcal{X}^{\text{II}}$

1: **function** LocalSearch($\mathcal{G}_\mathcal{S}, \mathcal{R}, \mathcal{X}^{\text{II}}, \delta, \gamma$)
2: $\quad\Delta_s \leftarrow \sum_{a \in \mathcal{A}_s} c_a(x_a^{\text{II}} - x_a^{\text{LB}})\ \forall s \in \mathcal{S}$;
3: $\quad\Delta \leftarrow \{\Delta_s\}_{s \in \mathcal{S}}$;
4: $\quad\Delta \leftarrow$ `SortDescending`(Δ);
5: $\quad\mathcal{S}^\delta \leftarrow \{s \in \mathcal{S} \mid \Delta_s \geq \Delta[\delta]\}$;
6: $\quad\mathcal{S}^\gamma \leftarrow \emptyset$;
7: $\quad$**for** $s \in \mathcal{S} \setminus \mathcal{S}^\delta$ **do**
8: $\quad\quad$**if** $\exists r \in \mathcal{R}, a \in \mathcal{A}_s, a' \in \{\mathcal{A}_{s'}\}_{s' \in \mathcal{S}^\delta} : q_a^r > 0\ \&\ q_{a'}^r > 0$ **then**
9: $\quad\quad\quad\mathcal{S}^\gamma \leftarrow \mathcal{S}^\gamma \cup \{s\}$;
10: $\quad\quad$**end if**
11: $\quad\quad$**if** $|\mathcal{S}^\gamma| == \gamma$ **then**
12: $\quad\quad\quad$break;
13: $\quad\quad$**end if**
14: $\quad$**end for**
15: $\quad\mathcal{X}^{\text{III}} \leftarrow$ `SolveIPModel`$(\mathcal{R}, \mathcal{G}_\mathcal{S}, \mathcal{S}^\delta \cup \mathcal{S}^\gamma, \mathcal{X}^{\text{II}})$;
16: $\quad$**return** $\mathcal{X}^{\text{III}}$;
17: **end function**

The local search uses two parameters, δ and γ to operate on $\mathcal{X}^{\text{II}}$. The former is the number of networks for which the algorithm tries to improve their solution. The latter parameter is the number of networks that the algorithm will use to trade resources with the previous δ ones.

The procedure, shown in Algorithm 5, starts by identifying the set of δ networks with the largest Δ_s values (lines 2–5). In Algorithm 5, $\Delta[\delta]$ is the δ-th element in the descending-ordered set Δ. Set $\mathcal{S}^\delta$ contains the networks for which the local search is trying to improve their Δ_s values. The procedure then selects, from all the networks not in $\mathcal{S}^\delta$, using a first-match rule, γ networks ($\mathcal{S}^\gamma$), each of them sharing at least one

resource with one or more networks in $\mathcal{S}^{\delta}$ (lines 6–14). The networks in $\mathcal{S}^{\gamma}$ are used as "partners" for the networks in $\mathcal{S}^{\delta}$ in order to trade off the use of resources. The values of the parameters associated with the local search are such that $\delta \ll \gamma$.

We define $\mathcal{S}^{\text{LS}} = \mathcal{S}^{\delta} \cup \mathcal{S}^{\gamma}$ and solve the original model (3.5)–(3.8) by fixing the paths in $\mathcal{S} \setminus \mathcal{S}^{\text{LS}}$. That is, variables in the set $\{x_a \,|\, a \in \{\mathcal{A}_s\}_{s \in \mathcal{S} \setminus \mathcal{S}^{\text{LS}}}\}$ are fixed to 1 if $a \in \mathcal{X}^{\text{II}}$, and to 0 otherwise; and the remaining variables (i.e., $\{x_a \,|\, a \in \{\mathcal{A}_s\}_{s \in \mathcal{S}^{\text{LS}}}$) are the only ones in the IP model. The solution of the IP model is denoted by $\mathcal{X}^{\text{III}}$. This solution is guaranteed to be no worse than $\mathcal{X}^{\text{II}}$. We have observed that the local search, as defined above, is often able to improve upon the solution constructed by the combination method, except in those cases when $\mathcal{X}^{\text{II}}$ is near-optimal. We have experimented with a local search that focuses only on the reduced set of networks with the worst Δ_s values (i.e., the networks in $\mathcal{S}^{\delta}$) and determined that this strategy provides very little room for improvement because most of the resources are committed to the paths that are fixed prior to solving the IP model.

3.3.3 *Lagrangian Relaxation*

The previous algorithm, as most heuristic-based procedures, will not always provide an optimal solution to the problem, but a feasible one. That is, the output of the algorithm will be an upper bound to the optimum value. As it will be shown in Chapter 4, for ATFM instances of moderate size, commercial MIP solvers can obtain optimal solutions, so the quality of the algorithm can be easily assessed. However, for bigger instances, solvers fail to even solve the LP relaxation of the problem so, to measure quality, a procedure to obtain lower bounds is required. For this, we study two Lagrangian Relaxations of the SRC-MSPP. Before discussing them, we first summarize what a Lagrangian Relaxation is and some known theoretical results on the topic.

3.3.3.1 *Theoretical Framework*

The following subsection has been elaborated using as main references[5] Geoffrion [58], Fisher [52], Bertsimas and Tsitsiklis [18, Ch. 11.4] and Guignard [68]. During the exposition, bold lowercase letters represent columns vectors, and bold uppercase letters matrices.

Let us start defining what the relaxation of an optimization problem is:

5 The interested reader is referred to them for a more detailed description.

Definition 1. Problem $RP = \min\{g(\boldsymbol{x}) \,|\, \boldsymbol{x} \in \mathcal{W}_{RP}\}$ is a relaxation of problem $P = \min\{f(\boldsymbol{x}) \,|\, \boldsymbol{x} \in \mathcal{W}_P\}$, with the same decision variables $\boldsymbol{x}$, if and only if:

(i) The feasible set of RP contains that of P, i. e., $\mathcal{W}_P \subseteq \mathcal{W}_{RP}$.

(ii) Over the feasible set of P, the objective function of RP dominates that of P, i. e., $\forall\, \boldsymbol{x} \in \mathcal{W}_P$, $g(\boldsymbol{x}) \leq f(\boldsymbol{x})$.

Denoting by $\boldsymbol{x}_P^*$ and $\boldsymbol{x}_{RP}^*$ the optimal solution of P and RP, respectively, it follows that: $g(\boldsymbol{x}_{RP}^*) \leq f(\boldsymbol{x}_P^*)$.

Consider, for the sake of discussion, that problem P is of the following form:

$$\min_{\boldsymbol{x}}\{\boldsymbol{c}'\boldsymbol{x} \,|\, \boldsymbol{A}_1\boldsymbol{x} \leq \boldsymbol{b}_1, \boldsymbol{A}_2\boldsymbol{x} \leq \boldsymbol{b}_2, \boldsymbol{x} \in \mathcal{W}\}, \tag{P}$$

where $\boldsymbol{x}$ is the (column) vector of variables, $\boldsymbol{c}$ the vector of costs coefficients, $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ the right-hand side constraint vectors, $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ matrices of conformable dimensions, and $\mathcal{W}$ the set containing the sign and integer restrictions on $\boldsymbol{x}$. In the following, it is assumed that constraints $\boldsymbol{A}_1\boldsymbol{x} \leq \boldsymbol{b}_1$ are "complicated", that is, if they did not exist, the problem could be solved efficiently. As an example, if in the SRC-MSPP there were no capacity constraints, just a collection of shortest path had to be solved.

Among the possible relaxations for problem (P), there is the so-called Lagrangian Relaxation (LR):

Definition 2. The Lagrangian Relaxation of problem (P), relative to constraints $\boldsymbol{A}_1\boldsymbol{x} \leq \boldsymbol{b}_1$, is:

$$\min_{\boldsymbol{x}}\{\boldsymbol{c}'\boldsymbol{x} + \boldsymbol{\lambda}'(\boldsymbol{A}_1\boldsymbol{x} - \boldsymbol{b}_1) \,|\, \boldsymbol{A}_2\boldsymbol{x} \leq \boldsymbol{b}_2, \boldsymbol{x} \in \mathcal{W}\}, \tag{LR_λ}$$

where $\boldsymbol{\lambda}$ is a (fixed) vector of non-negative real numbers called Lagrangian multipliers, and constraints $\boldsymbol{A}_1\boldsymbol{x} \leq \boldsymbol{b}_1$ are said to be dualized.

Note that (LR_λ) fulfills Definition 1: Condition (i) clearly holds, while (ii) follows from the facts that $\boldsymbol{\lambda} \geq \boldsymbol{0}$ and, for every feasible solution of (P), $\boldsymbol{A}_1\boldsymbol{x} - \boldsymbol{b}_1 \leq \boldsymbol{0}$. As a result, for any $\boldsymbol{\lambda} \geq \boldsymbol{0}$, any optimal solution of (LR_λ), called Lagrangian solution and denoted by $\boldsymbol{x}_\lambda^*$, provides a lower bound on the optimal value of (P). The goal is to find the tightest of such as bounds, which is called the Lagrangian dual problem:

Definition 3. The Lagrangian dual of problem (P), relative to constraints $\boldsymbol{A}_1\boldsymbol{x} \leq \boldsymbol{b}_1$, is:

$$\max_{\boldsymbol{\lambda} \geq \boldsymbol{0}} z(\boldsymbol{\lambda}), \tag{LR^*}$$

where $z(\boldsymbol{\lambda})$ is the optimal cost of (LR_λ).

Under the assumption that the convex hull of the non-dualized constraints, denoted by $CH(\{\boldsymbol{x} \in \mathcal{W} | \boldsymbol{A}_2\boldsymbol{x} \leq \boldsymbol{b}_2\})$, is a bounded polyhedron (as happens in the SRC-MSPP), $z(\boldsymbol{\lambda})$ can be written as:

$$z(\boldsymbol{\lambda}) = \min_{i=\{1,\ldots,M\}} \boldsymbol{c}'\boldsymbol{x}^i + \boldsymbol{\lambda}'(\boldsymbol{A}_1\boldsymbol{x}^i - \boldsymbol{b}_1),$$

where $\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^M\}$ is the set of extreme points of $CH(\{\boldsymbol{x} \in \mathcal{W} | \boldsymbol{A}_2\boldsymbol{x} \leq \boldsymbol{b}_2\})$. From this representation it can be seen that $z(\boldsymbol{\lambda})$ is a concave picewise linear function (since it is the minimum of a finite collection of linear functions of $\boldsymbol{\lambda}$), and that problem (LR^*), therefore, can be recast as a linear programming, but with a very large number of constraints.

Unlike the case of the LP relaxation, if a Lagrangian solution $\boldsymbol{x}^*_\lambda$ turns out to be feasible for problem (P), it does not mean to be optimal. It just provides bounds on where the optimal cost will be: $[\boldsymbol{c}'\boldsymbol{x}^*_\lambda + \boldsymbol{\lambda}'(\boldsymbol{A}_1\boldsymbol{x}^*_\lambda - \boldsymbol{b}_1),\ \boldsymbol{c}'\boldsymbol{x}^*_\lambda]$. Optimality can be assessed when complementary slackness holds, i. e., $\boldsymbol{\lambda}'(\boldsymbol{A}_1\boldsymbol{x}^*_\lambda - \boldsymbol{b}_1) = \boldsymbol{0}$. Unfortunately, this a sufficient, not a necessary condition, so $\boldsymbol{x}^*_\lambda$ may be feasible and optimal for (P) and not satisfy complementary slackness.

The following theorem provides a significant result about the strength of the bound achieved by solving the Lagrangian dual problem.

Theorem 1. *The optimal value (z^{LR^*}) of the Lagrangian dual problem is equal to the optimal cost of the following linear programming problem:*

$$\min_{\boldsymbol{x}}\{\boldsymbol{c}'\boldsymbol{x} \,|\, \boldsymbol{A}_1\boldsymbol{x} \leq \boldsymbol{b}_1, CH(\{\boldsymbol{x} \in \mathcal{W} | \boldsymbol{A}_2\boldsymbol{x} \leq \boldsymbol{b}_2\})\}.$$

Proof. See in Geoffrion [58]. □

Corollary 1. *Let z^{IP} and z^{LP} denote the optimal cost of problem (P) and its LP relaxation, respectively. Then:*

$$z^{LP} \leq z^{LR^*} \leq z^{IP},$$

that is, the Lagrangian dual problem provides a lower bound that is always at least as good as that coming from the LP relaxation.

Corollary 2. *If the non-dualized constraints have the intregrality property, i. e.: $CH(\{\boldsymbol{x} \in \mathcal{W} | \boldsymbol{A}_2\boldsymbol{x} \leq \boldsymbol{b}_2\}) = \{\boldsymbol{x} \,|\, \boldsymbol{A}_2\boldsymbol{x} \leq \boldsymbol{b}_2\}$, then the optimal cost of the Lagrangian and LP relaxation are equal: $z^{LP} = z^{LR^*}$.*

Under the scenario described in the last corollary, solving the LR instead of the LP relaxation of (P) may still be of interest because: 1) Sometimes, the LP cannot be computed easily, and 2) The procedure to solve the LR can be combined with a heuristic to get feasible integer solutions for (P).

Note that in all the previous discussion, just one set of constraints was dualized ($\boldsymbol{A}_1\boldsymbol{x} \leq \boldsymbol{b}_1$). The motivation, as previously mentioned, was that this set was formed by complicated constraints, while the non-dualized ones had a special structure that made of the remaining problem an easier one to solve. On some occasions, however, there are two (or more) sets of constraints with a special structure and common variables linking them. In that situation, it is possible to split the common variables and dualized the copy constraints. This particular type of relaxation is called Lagrangian Decomposition and works as follows:

1. Define the following equivalent problem of (P) (same optimal values, but different variable spaces):

$$\min_{\boldsymbol{x},\boldsymbol{u}}\{\boldsymbol{c}'\boldsymbol{x} \mid \boldsymbol{A}_1\boldsymbol{x} \leq \boldsymbol{b}_1, \boldsymbol{x} \in \mathcal{W}, \boldsymbol{A}_2\boldsymbol{u} \leq \boldsymbol{b}_2, \boldsymbol{u} \in \mathcal{W}, \boldsymbol{x} = \boldsymbol{u}\}. \tag{P'}$$

2. Dualized the copy constraints and solve two independent subproblems:

$$\begin{aligned}&\min_{\boldsymbol{x},\boldsymbol{u}}\{\boldsymbol{c}'\boldsymbol{x} + \boldsymbol{\lambda}'(\boldsymbol{u} - \boldsymbol{x}) \mid \boldsymbol{A}_1\boldsymbol{x} \leq \boldsymbol{b}_1, \boldsymbol{x} \in \mathcal{W}, \boldsymbol{A}_2\boldsymbol{u} \leq \boldsymbol{b}_2, \boldsymbol{u} \in \mathcal{W}\} = \\ &\min_{\boldsymbol{x}}\{(\boldsymbol{c}' - \boldsymbol{\lambda}')\boldsymbol{x} \mid \boldsymbol{A}_1\boldsymbol{x} \leq \boldsymbol{b}_1, \boldsymbol{x} \in \mathcal{W}\} + \min_{\boldsymbol{u}}\{\boldsymbol{\lambda}'\boldsymbol{u} \mid \boldsymbol{A}_2\boldsymbol{u} \leq \boldsymbol{b}_2, \boldsymbol{u} \in \mathcal{W}\}.\end{aligned} \tag{LD_λ}$$

Notice that on this occasion $\boldsymbol{\lambda}$ is not constrained in sign and that obtaining a Lagrangian solution that is feasible for (P') directly implies optimality (complementary slackness holds).

For the Lagrangian Decomposition, the following theorem states the strength of the bound achieved:

Theorem 2. *The optimal value (z^{LD^*}) of the Lagrangian Decomposition dual problem is equal to the optimal cost of the following linear programming problem:*

$$\min_{\boldsymbol{x}}\{\boldsymbol{c}'\boldsymbol{x} \mid CH(\{\boldsymbol{x} \in \mathcal{W} | \boldsymbol{A}_1\boldsymbol{x} \leq \boldsymbol{b}_1\}) \cap CH(\{\boldsymbol{x} \in \mathcal{W} | \boldsymbol{A}_2\boldsymbol{x} \leq \boldsymbol{b}_2\})\}.$$

Proof. See in Guignard and Kim [69]. □

Corollary 3. *If one of the two subproblems has the integrality property, then z^{LD^*} is equal to the best of the two LR bounds corresponding to dualizing either $\boldsymbol{A}_1\boldsymbol{x} \leq \boldsymbol{b}_1$ or $\boldsymbol{A}_2\boldsymbol{x} \leq \boldsymbol{b}_2$. If both subproblems have the integrality property, then $z^{LD^*} = z^{LP}$.*

To solve the Lagrangian dual problem and obtain the optimal vector of Lagrangian multipliers $\boldsymbol{\lambda}^*$, several methods exist in the literature. One of them is the Subgradient method, which is based on obtaining subgradients of $z(\boldsymbol{\lambda})$ to iteratively update the Lagrangian multipliers in a steepest ascent fashion. Before describing it, we define what a subgradient is and present two theoretical results employed in the algorithm.

Definition 4. A vector $\boldsymbol{\phi}$ is a subgradient of a concave function $f : \mathbb{R}^n \mapsto \mathbb{R}$ at point $\boldsymbol{x}^0 \in \mathbb{R}^n$ if:

$$f(\boldsymbol{x}) \leq f(\boldsymbol{x}^0) + \boldsymbol{\phi}'(\boldsymbol{x} - \boldsymbol{x}^0) \quad \forall \boldsymbol{x} \in \mathbb{R}^n.$$

The set of all subgradients of f at $\boldsymbol{x}^0$, denoted by $\partial f(\boldsymbol{x}^0)$, is the subdiferential of f at $\boldsymbol{x}^0$.

Proposition 3. *Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be a concave function. A vector $\boldsymbol{x}^*$ maximizes f over its domain if and only if $\mathbf{0} \in \partial f(\boldsymbol{x}^*)$.*

Proof. See in Bertsimas and Weismantel [19, Ch. 4.4] □

Proposition 4. *Let $\boldsymbol{x}^*_\lambda$ be a Lagrangian solution of problem (LR_λ) for the vector of multipliers $\boldsymbol{\lambda}$. Then, a subgradient of $z(\boldsymbol{\lambda})$ is given by $\boldsymbol{A}_1\boldsymbol{x}^*_\lambda - \boldsymbol{b}_1$.*

Proof. See in Bertsimas and Weismantel [19, Ch. 4.4] □

The subgradient method consists of the following iterative algorithm:

1. Choose a starting vector of multipliers $\boldsymbol{\lambda}^j$, where j stands for the iteration index. At the beginning, $j = 0$.
2. Solve problem (LR_λ) for $\boldsymbol{\lambda}^j$ and let $\boldsymbol{\phi}^j = \boldsymbol{A}_1\boldsymbol{x}^*_\lambda - \boldsymbol{b}_1$.
3. If $\boldsymbol{\phi}^j = \mathbf{0}$, then $\mathbf{0} \in \partial z(\boldsymbol{x}^*_\lambda)$ and the algorithm terminates because $\boldsymbol{\lambda}^j$ is optimal. Else, let $\boldsymbol{\lambda}^{j+1} = \boldsymbol{\lambda}^j + \theta^j\boldsymbol{\phi}^j$, where θ^j is a positive stepsize parameter at iteration j. Increment j and go to step 2.

In the algorithm, the stopping criterion of $\boldsymbol{\phi}^j = \mathbf{0}$ is rarely met, so others such as the number of iterations or distance between multipliers are incorporated[6]. Additionally, if sign restrictions exist over the multipliers, each component of $\boldsymbol{\lambda}$ that violates it after the last step of the algorithm, is set to 0.

6 As exposed in Guignard [68], with this algorithm, the sequence $\|\boldsymbol{\lambda}^j - \boldsymbol{\lambda}^*\|$ is monotone non-increasing.

To establish θ^j, different rules exist that guarantee the convergence of the algorithm to the dual optimum (z^{LR^*}). One of the most popular is:

$$\theta^j = \mu_j \frac{z^{\text{LR}^*} - z(\boldsymbol{\lambda}^j)}{\|\boldsymbol{\phi}^j\|^2},$$

where $\mu_j \in (0, 2)$. Unfortunately, this rule requires of the unknown value of z^{LR^*}, so in practice the best estimation available is used.

To overcome the difficulty of having to know z^{LR^*}, in Bragin, Luh, Yan, Yu, and Stern [23] and Zhao, Luh, and Wang [117] is presented a modification of the previous algorithm called Surrogate Lagrangian Relaxation. In this method, the best estimation of z^{LR^*} is only used at the first iteration of the algorithm, not being necessary the rest of the times. Furthermore, the method guarantees the convergence of the algorithm without having to solve (LR_λ) to optimality at each iteration. As a drawback, other parameters have to be set by the user.

3.3.3.2 *Lagrangian Relaxation and Decomposition for the SRC-MSPP*

From the previous subsection it is clear that the most natural Lagrangian Relaxation for the IP formulation of the SRC-MSPP, model (3.5)-(3.8) on page 57, is the one resulting from dualizing the capacity constraints:

$$\min \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}_s} c_a x_a + \sum_{r\in\mathcal{R}} \lambda_r \Big(\sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}_s} q_a^r x_a - W^r \Big), \tag{3.10}$$

subject to:

$$\sum_{a\in\Lambda_n^-(n)} x_a - \sum_{a\in\Lambda_n^+(n)} x_a = \begin{cases} 1, & \text{if } n = o_s, \\ -1, & \text{if } n = d_s, \\ 0, & \text{otherwise.} \end{cases} \quad \forall s \in \mathcal{S}, n \in \mathcal{N}_s \tag{3.11}$$

$$x_a \in \{0, 1\}, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}_s. \tag{3.12}$$

So for a given vector of (non-negative) multipliers $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_{|\mathcal{R}|}]'$, problem (3.10)–(3.12) is a shortest path problem in $|\mathcal{S}|$ networks. Notice that as the flow constraints (3.11) define a polyhedron of integer vertices, the bound provided by the Lagrangian dual problem will be equal to that coming from the LP relaxation of model (3.5)-(3.8) (recall Corollary 2).

In the computational experience we employed this relaxation to obtain lower bounds with which to compare the solutions generated by the matheuristic.

An alternative Lagrangian Relaxation

Alternatively to the previous relaxation, it is possible to formulate a new one that exploits the structure of the capacity constraints in the SRC-MSPP. Note that these constraints have the structure of a multiple (possibly multidimensional) knapsack problem. For example, in the ATFM problem, there are multiple bins/knapsacks where an arc can be assigned to (e. g., same departure airport, but different departure time), and when assigned, it may consume (multidimensional part) more than one resource, e. g., a sector for multiple periods of time.

This structure is exploited by means of the Lagrangian Decomposition of the SRC-MSPP. Here we present the mathematical formulation of this relaxation, leaving obtaining computational results as future research.

To obtain the relaxation, the first step is to make a copy of the decision variables, so the flow and capacity constraints can be split into two separated groups:

SRC-MSPP with copy of the variables

$$\min \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}_s} c_a x_a, \tag{3.13}$$

subject to:

$$\sum_{a\in\Lambda_n^-(n)} x_a - \sum_{a\in\Lambda_n^+(n)} x_a = \begin{cases} 1, & \text{if } n = o_s, \\ -1, & \text{if } n = d_s, \\ 0, & \text{otherwise.} \end{cases} \quad \forall s \in \mathcal{S}, n \in \mathcal{N}_s \tag{3.14}$$

$$x_a = x'_a \quad \forall s \in \mathcal{S}, a \in \mathcal{A}_s, \tag{3.15}$$

$$\sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}_s} q_a^r x'_a \leq W^r \quad \forall r \in \mathcal{R}, \tag{3.16}$$

$$x_a, x'_a \in \{0,1\}, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}_s. \tag{3.17}$$

Then, we dualize constraints (3.15) and obtain the next relaxation:

$$\min \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}_s} \Big(c_a x_a + \lambda_a (x_a - x'_a) \Big), \quad \text{subject to: (3.14), (3.16) and (3.17).}$$

This relaxation can be split into the following two subproblems:

Subproblem 1: Multiple Shortest paths

$$\min \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}_s} (c_a + \lambda_a) x_a, \tag{3.18}$$

subject to:

$$\sum_{a\in\Lambda_n^-(n)} x_a - \sum_{a\in\Lambda_n^+(n)} x_a = \begin{cases} 1, & \text{if } n = o_s, \\ -1, & \text{if } n = d_s, \\ 0, & \text{otherwise.} \end{cases} \quad \forall s \in \mathcal{S}, n \in \mathcal{N}_s \tag{3.19}$$

$$x_a \in \{0,1\}, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}_s. \tag{3.20}$$

Subproblem 2: Generalized knapsack problem

$$\min -\sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}_s} \lambda_a x'_a, \tag{3.21}$$

subject to:

$$\sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}_s} q_a^r x'_a \leq W^r \quad \forall r \in \mathcal{R}, \tag{3.22}$$

$$x'_a \in \{0,1\}, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}_s. \tag{3.23}$$

Note that in real applications this subproblem will usually be formed by multiple independent knapsack problems. As an example, in the ATFM problem, each sector and airport is an independent (multiple multidimensional) knapsack problem. Furthermore, to increase the possibilities of obtaining solutions for which $x_a = x'_a \ \forall s \in \mathcal{S}, a \in \mathcal{A}_s$, some constraints based on the nature of the problem may be added to this second subproblem. E. g., for the ATFM problem, it can be added constraints that force a flight to not depart more than once.

4

COMPUTATIONAL EXPERIENCE

In this chapter we discuss, through a series of computational experiments, the empirical behavior of the work developed in the previous chapters of the thesis. We start presenting the ATFM data sets employed, which were generated using publicly available sources and released for free disposal. Afterwards, we discuss the computational results obtained when formulating the ATFM as an SRC-MSPP, and solving it by means of exact methods and the matheuristic proposed in Chapter 3. Results for the Lagrangian Relaxation resulting from dualizing the capacity constraints are also discussed. The chapter concludes with a thorough analysis of the elements of the algorithm to assess their contribution, and showing the flight plans modifications produced by the solution.

4.1 ATFM DATA SETS

In ATFM, unlike other optimization problems, there is not an established collection of data sets that helps authors to test their proposals, reproduce others' work, etc. This clearly complicates the research labor, making that the authors have to spend a lot of time preparing the data sets for the computational experience.

To contribute to change that, we generated a collection of ATFM data sets using publicly available sources. The data sets can be found, jointly with the code of the algorithm described in Subsection 3.3.2, in https://github.com/DavidGarHeredia/SRC-MSPP.

As far as we know, the only publication that also released the data sets employed in the computational experience is that of Dal Sasso et al. [34]. However, these data sets, opposite to ours, are completely synthetic, meaning that no real data sources were used to generate them. Testing our proposal also with these data sets is left as future research.

In the following, we describe the process that we followed to generate the mentioned ATFM data sets.

4.1.1 *Raw data transformation*

As said, we used publicly available sources to create our data sets. Concretely, from [111] we obtained information about domestic flights in the US, and from [112] information about the airports' location (latitude and longitude). The information about the flights corresponds to the 16th of January, May, and September of 2019. The choice of the 16th is due to the high volume of air traffic on that day for each of these months. Henceforth, we will refer to these data sources as raw data sets.

Among the flight information available in the raw data sets, we were interested[1] in: The departure and landing time, the origin and destination airport, if the flight was canceled or diverted (i. e., it landed at a different destination than the originally scheduled), and the number of flights operated by each aircraft (continued flights).

We made the following modifications to the raw data sets:

1. Instead of expressing the time in hours and minutes, it was transformed to a number between 0 and $24\text{h} * 60\text{min/h} = 1440$.

2. The following flights were removed:
   a) Flights with missing information (*NAs*).
   b) Canceled and diverted flights.
   c) Flights outside the contiguous territory of the US (mainly Hawaii, Alaska and Guatemala). This is for simplicity when creating the sectors (see later on).

Respect to deleting flights, if there had been any flight using an airport not listed in the airport data set, it would have also been removed. This is because the coordinates

1 Other information contained in these data sets was not used and therefore it will not be described here. Nonetheless, the interested reader can find it in [111].

of the airport, which are required to create the routes and sectors afterward, would be missing. However, we did not encounter this situation.

Note that to generate ATFM flight plans, the information contained in the raw data sets is not enough. It is also needed information about: i) The sectors and their capacity, ii) The turnaround time between continued flights (recall parameter $\tau_{f',f}$ in Section 2.4), and iii) The route of each flight. As this information is not available, we had to estimate it as commonly do in ATFM literature (e. g., [4, 11, 20]). The procedure to do that is discussed later on, but we point out here that, as a consequence of not having information about the route of each flight, the landing time will be substituted by the departure time plus the time required to travel the estimated route. This is to have coherence in time.

4.1.2 *Sectors and route waypoints*

To be able to create the flight plans with the information contained in the raw data sets, we first had to model the scenario where the flights take place: The contiguous territory of the US. For that, we divided the airspace into 400 sectors using a 20×20 grid, as illustratively shown in Figure 17. The boundaries of the grid are set according

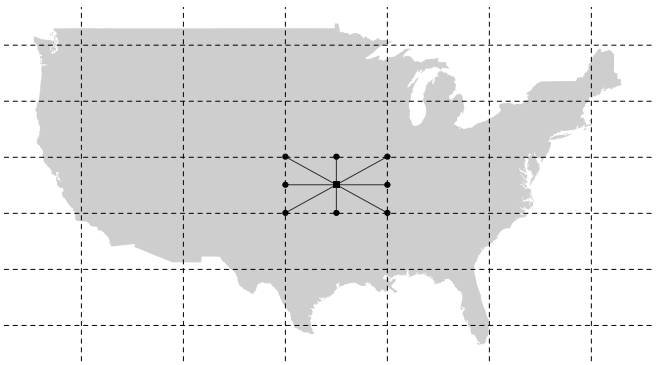


Figure 17: Example of sectors and waypoints.

to maximum and minimum latitude and longitude of the airports in the data set ± 1 degree. That way, all the airports are contained in the grid and the boundaries do not pass through them (the ± 1 provides the slackness).

As illustrated in the figure, associated with each sector there are nine waypoints, eight on the boundary, to define entry and exit points; and one in the middle, to connect the boundary-waypoints by bi-directional arcs (solid lines in the figure). Each airport is also connected to its sector boundary-waypoints by bi-directional arcs. This gives rise to a space graph that will be used later on to establish the flights routes. In this graph, the length of each arc is set as the Haversine distance between the waypoints connected by the arc.

4.1.3 *Flight plans*

So far, the information at disposal is that of the raw data sets and the one created in the previous subsection about sectors and potential routes. However, to create the flight plans, some extra information referring to time is required. Concretely:

- The maximum departure delay for each flight. We set this value to 90 minutes. Notice that the scheduled departure time plus these 90 minutes establish the departure time window of every flight.
- The speed of each aircraft. We define a common speed of 885 km/h (0.742 Mach). Usually, commercial aircraft travel at 0.8 M, but due to time discretization, this speed resulted in less sensible results. Note that the aircraft speed and the length of each arc define the scheduled travel time ($\ell_{m,n}$).
- The maximum speed variation permitted when traversing an arc ($\underline{\ell}_{m,n}$, $\overline{\ell}_{m,n}$). We set this value to 25% of the scheduled travel ($\ell_{m,n}$). Note that this limits the latest landing time to 25% of the travel time plus the ground delay.
- The turnaround time ($\tau_{f',f}$) for continued flights. We set this value to the difference between the scheduling departure time of f and the scheduled landing time of f'. We noticed, however, that sometimes this value was too big, making that if flight f' suffered some delay, waiting the turnaround time would not allow the continued flight f to depart within its departure time window. To avoid that situation, we reduced $\tau_{f',f}$ in those cases. The reduction value was obtained with the following rule $\min(6, \max(\tau_{f',f} - 1, 0))$. Value 6 was empirically set to obtain sensible results.
- Time discretization. We did it in time periods of 5 minutes. When no integer numbers were obtained from transforming minutes to time periods, we used the

following rounding rules: 1) For the travel times ($\ell_{m,n}$), we always rounded up, 2) For the changes of speed ($\underline{\ell}_{m,n}$, $\overline{\ell}_{m,n}$), we rounded up when the decimal part was equal or above to 0.75, else we rounded down. Rounding up with 0.5 may represent an excessive change of speed, hence our decision.

With all this information, the flight plans can be generated. The main route of each flight is defined as the shortest path between its departure and landing airport in the graph constructed in the previous subsection. To have time coherence, the landing time, as earlier pointed out, is changed to its departure time plus the time required to reach its destination when traveling the main route at the scheduled speed.

Respect to the alternative routes, notice that obtaining them through the straightforward approach of establishing the $k-$shortest paths from origin to destination may not be valid because, due to how the space graph is built, these shortest paths may use the same sectors that the main route, making them useless. Thus, we proceed as follows. First, using the main route, we obtained the minimum capacity required at sectors to avoid changes in the flight plans (see example in Figure 18).

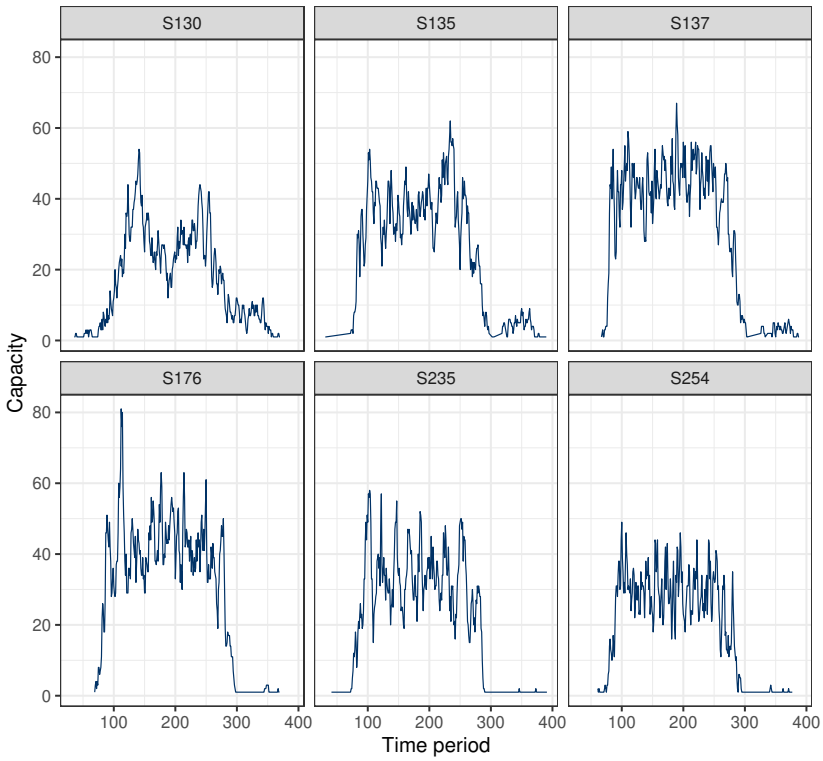


Figure 18: Capacity requirements per time period for the most demanded sectors on January 16, 2019.

Then, to alleviate these capacity requirements, we generated up to four alternative routes for each flight passing through at least one of the twenty most demanded sectors. To explain how these alternative routes are obtained, consider that flight f passes through N of these twenty sectors. The procedure sets a high[2] cost (i. e., travel time) for the arcs of the space graph passing through these N sectors. An exception occurs when the sector contains the departure or landing airport of the flight. Setting a high cost for the arcs of those sectors is worthless because it is not possible to set an alternative route that does not use them. After setting the high cost, the shortest path from origin to destination for flight f is recomputed, obtaining an alternative route that passes through different sectors than the main one. To obtain another route, the arcs of the closest sector to the landing airport are set to their original cost and the SP is recomputed. The procedure continues until no penalized sectors remain or four alternative routes have been obtained.

4.1.4 *4D-networks*

Once created the flight plans, the 4D-networks to formulate the ATFM as an SRC-MSPP can be obtained by incorporating the time information into the nodes of the networks as exposed in Sections 2.6 and 3.2.5. Then, the 4D-networks are sorted in topological order. That is, the nodes of the network are labeled in such a way that no arc has its tail node with a label greater than its head node. The output at this phase is a table where each arc corresponds to a row and contains the following information (columns of the table):

- Arc ID. An integer from 0 to $|\cup_{s\in\mathcal{S}}\mathcal{A}_s| - 1$ to identify the arc.
- Network ID. An integer from 0 to $|\mathcal{S}| - 1$ indicating to which network belongs the arc.
- Flight ID. An integer from 0 to $|\mathcal{F}| - 1$ indicating to which flight belongs the arc.
- Previous flight ID. An integer indicating the ID of the predecessor flight. If the flight has not predecessor, we set this value to -1.
- Tail node of the arc. An integer from 0 to $|\mathcal{N}_s| - 1$ indicating the tail node of the arc. Recall that this label comes from the topological sorting.

2 We used $10,000$ units.

- Head node of the arc. An integer from 0 to $|\mathcal{N}_s| - 1$ indicating the head node of the arc
- Cost of using the arc. Using the notation of Subsection 2.5.1: i) For ground delays, $\epsilon_1 = 0.25$ and $c_g = 1$, ii) For late arrivals, $\epsilon_2 = 0.75$ and $c_a = 2$, and iii) For speed changes, $\epsilon_3 = 0.5$, $c_v = 100$ and $c_r = 10$. The speed change cost was additionally reduced by 5 units (without allowing negative quantities), because it showed a better empirical behavior in the final flight plans obtained.
- Begin time. An integer between 0 and 1440 indicating the time at which the tail node was left when using this arc.
- End time. An integer between 0 and 1440 indicating the time at which the head node is reached when using this arc.
- Speed change. When the arc refers to a departure or landing operation, this shows the number of time periods that differ from scheduled, i. e., $t_2 - t_1 - \ell_{m,n}$. When the arc refers to a change of speed in the air, it shows $\frac{t_2 - t_1 - \ell_{m,n}}{\ell_{m,n}}$, (recall in Subsection 2.5.1).
- Route ID. An integer from 1 to 5 indicating to which route belongs the arc. A value of 1 refers to the main route, while a value from 2 to 5 to the rest of the alternative routes (if any).
- Phase of the flight. Due to the dummy nodes existing for origin and destination (triangular nodes in Figure 14, page 63), and the arcs connecting continued flights, there are seven possibilities: Dummy Source, departure, ground delay, air, land, union and dummy sink.
- Sector or airport employed by the arc. Arcs associated with no sector (e. g., those connecting continued flights) have the label *none*.

4.1.5 *Capacity constraints*

To generate the ATFM data sets, the only part that remains is creating the capacity constraints for airports and sectors. The output table described in the previous subsection has all the information needed (resources and time at which they are used) to formulate

the left-hand side of these constraints. Thus, the only part left is to generate the capacity levels (right-hand sides) of each resource.

As this point is critical for the ATFM problem, we generated different scenarios (capacity levels) to test the robustness of the formulation and the proposed algorithm. Particularly, for each flight plan, we tested twelve scenarios, which we grouped into three categories (easy, medium, and difficult).

We first created a base scenario with capacities based on the minimum value required for the original flight plan to be feasible. For example, for sector S135 in Figure 18, that value is 62. We noticed that with this rule, however, the less demanded sectors/airports ended up with a really low capacity level, so we decided to set a minimum value for sectors, departures and landings. Grouping the constraints by these 3 categories (sectors, departures and landings), the minimum value of each group was computed as the 10% trimmed mean of the capacities assigned with the rule described at the beginning of the paragraph. As an example, in the flight plan of 2019/Jan/16, the minimum value was 15 aircraft for sectors, and 2 for departure and landing operations. The creation of the base scenario concludes by including constraints that limit the number of simultaneous departures and landings occurring at each airport (recall parameter C_k^t in Section 2.4). The capacity value for each of these constraints was set equal to 80% of the total departure and arrival capacity of the corresponding airport. Note that due to this last group of constraints, the plan resulting from all the flights using their main route is no longer feasible.

The twelve abovementioned scenarios are created by applying different capacity reductions to the base scenario. Easy cases simulate the effect of bad weather moving across sectors and causing capacity reductions (Bertsimas et al. [20]). Concretely, the generation of these instances starts with one of the five most demanded sectors (chosen at random) and two contiguous sectors. The capacity of these three sectors (and their airports) is reduced during five consecutive time periods. Then, the capacity reduction is moved to three other contiguous sectors for another five time periods. The process is repeated for the entire time horizon. The reductions were set at 10%, 20%, 30% and 40% of the base capacity, leading to a total of four scenarios. Medium cases are based on the easy ones with an additional capacity reduction randomly applied to 50% of the elements (sectors and airports) and enforced during the entire planning horizon. The additional reduction of the capacity of each element is randomly chosen between 1% and 20%. For the difficult cases, the randomly generated capacity reduction is applied

to all elements. At the end of the penalization process, a minimum capacity of one aircraft was set for each element in all the scenarios.

4.1.6 *Instances dimensions*

In order to build a test set with various problem sizes, we created versions with fewer flights than those contained in the raw data sets. Concretely with 30% and 65% of the total number of aircraft. Note that the generation of capacity levels (previous subsection) is flight plan-depended, so they had to be repeated for these smaller plans. The dimensions of the instances in our test set are shown in Table 2. Note that each arc

Table 2: Dimensions of the instances in our test set.

Flight plan	Size	#Flights	#Networks	#Arcs	#Nodes	#Resources
Jan	30%	5,596	1,329	3,116,931	1,482,732	136,887
May	30%	6,951	1,368	4,043,046	1,868,888	145,711
Sep	30%	6,610	1,368	3,6706,44	1,729,221	142,734
Jan	65%	11,788	2,879	7,195,904	3,336,779	173,249
May	65%	13,752	2,963	9,717,333	4,341,805	178,451
Sep	65%	13,530	2,964	8,597,363	3,922,250	185,519
Jan	100%	18,100	4,429	11,934,419	5,446,911	184,404
May	100%	20,634	4,558	14,475,487	6,457,815	204,724
Sep	100%	20,581	4,559	13,993,972	6,248,302	201,451

corresponds to a variable in the IP formulation of the SRC-MSPP (model (3.5)-(3.8) on page 57), each node to a flow constraint, and each resource to a capacity constraint. The large number of capacity constraints is due to the product of the number of capacity elements and periods in the planning horizon. There are twelve scenarios for each case in the table, resulting in a total of 108 instances. Since most of the constraints in the problem define facets (flow constraints), a strong LP relaxation is expected when attempting to solve the problem using exact methods. Our computational experiments corroborate this important characteristic of the model, which others have pointed out as related to time-expanded network IP formulations (Boland and Savelsbergh [22]).

4.2 COMPUTATIONAL RESULTS

In this section we discuss the results of a series of computational experiments performed using the ATFM instances created before. The experiments are oriented to study the empirical behavior of the IP formulation of the SRC-MSPP, the matheuristic proposed to solve it, and the Lagrangian Relaxation resulting from dualizing the capacity constraints (recall in Subsection 3.3.3.2).

We perform all experiments on an HP Z230 Tower Workstation, with 4 cores (processor i7-4770 3.40GHz) and 32 GB of RAM. Our MIP solver is Gurobi 9.0 ([70]). The matheuristic algorithm was coded in C++, compiled using the GNU compiler 6.3, and run on a Debian 9 Operating System. The for-loop to generate the pool of solutions (Algorithm 2, page 70) was parallelized using OpenMP ([102]) with 8 threads.

To take into account the random elements in the proposed matheuristic, we solved each problem instance five times. We represent the associated variability with violin plots, a well-known extension of the boxplot that shows the distribution of the data. In violin plots, the quantiles corresponding to 25%, 50% (median), and 75% are depicted with a solid line, while a dark diamond shows the mean.

In addition, we set the MIP gap in Gurobi at various levels depending on the purpose for calling this optimizer: solving full model (0.5%), generating the pool of solutions (1%), combining the solutions (1%), and applying local search (0.5%).

4.2.1 *Integer Programming Results*

We now discuss the results when solving the ATFM instances through exact methods (MIP solver). The goals with this part of the experiments are: 1) To corroborate the hypothesis of the good behavior of the IP formulation for the SRC-MSPP given that most of the constraints define facets, and 2) To obtain the optimal solutions to assess the goodness of the algorithm later on.

The results exposed are without using the graph of conflict methodology introduced in Section 2.7. Preliminary results shown that there was not time gain when employing it. Actually, on several occasions it was counterproductive, being faster, although not significantly, solving the problem without this preprocessing. We partially think that a more efficient implementation of the code of that methodology could reverse the situation, but for the moment, the conclusion is that it does not produce any benefit.

We attempted to solve the IP model for the 108 instances in our test set with Gurobi. However, 44 of the runs terminated in an out-of-memory error and without an integer solution. These instances (recall in Table 2) correspond to the full flight plans (size 100%) and for the 65% reduced flight plans for the May and September days under the difficult scenario. Table 3 and 4 show the results of the Gurobi runs.

Table 3: Gurobi results for instances of size 30%.

Flight plan	Difficulty	Reduction	z^{IP}	z^{LP}	Gap	t	nn	$col_{\%}$	$row_{\%}$
Jan	easy	10%	53.94	53.94	0.00%	121.54	0	17.81%	38.34%
Jan	easy	20%	140.71	140.71	0.00%	123.41	0	17.81%	38.34%
Jan	easy	30%	344.77	344.77	0.00%	125.96	0	17.81%	38.34%
Jan	easy	40%	799.52	796.83	0.34%	136.46	1	17.81%	38.34%
Jan	medium	10%	1,418.49	1,417.74	0.05%	180.84	1	17.94%	38.39%
Jan	medium	20%	1,557.70	1,556.20	0.10%	216.62	1	17.94%	38.39%
Jan	medium	30%	1,846.41	1,843.63	0.15%	217.09	1	17.94%	38.39%
Jan	medium	40%	2,570.51	2,566.67	0.15%	338.83	1	17.94%	38.39%
Jan	difficult	10%	2,332.08	2,325.09	0.30%	312.98	1	18.12%	38.62%
Jan	difficult	20%	2,394.94	2,386.05	0.37%	322.54	1	18.12%	38.62%
Jan	difficult	30%	2,506.20	2,496.11	0.40%	342.96	1	18.12%	38.62%
Jan	difficult	40%	2,779.11	2,771.58	0.27%	371.94	1	18.12%	38.62%
May	easy	10%	68.70	68.70	0.00%	165.34	0	16.32%	35.76%
May	easy	20%	195.58	195.58	0.00%	161.97	0	16.32%	35.76%
May	easy	30%	577.15	577.15	0.00%	177.37	0	16.32%	35.76%
May	easy	40%	1,456.42	1,456.42	0.00%	185.56	0	16.32%	35.77%
May	medium	10%	1,388.94	1,388.94	0.00%	206.63	0	16.39%	35.80%
May	medium	20%	1,559.76	1,559.76	0.00%	223.57	0	16.39%	35.80%
May	medium	30%	1,955.98	1,955.98	0.00%	236.27	0	16.39%	35.80%
May	medium	40%	2,744.99	2,743.30	0.06%	347.99	1	16.39%	35.81%
May	difficult	10%	2,684.09	2,672.23	0.44%	481.28	1	16.50%	35.88%
May	difficult	20%	2,757.35	2,749.10	0.30%	474.33	1	16.50%	35.88%
May	difficult	30%	3,077.13	3,071.69	0.18%	830.47	1	16.50%	35.89%
May	difficult	40%	3,774.89	3,759.51	0.41%	1,105.56	1	16.50%	35.90%
Sep	easy	10%	82.33	82.33	0.00%	152.28	0	17.38%	37.69%
Sep	easy	20%	152.74	152.74	0.00%	148.01	0	17.38%	37.69%
Sep	easy	30%	379.26	379.26	0.00%	153.40	0	17.38%	37.69%
Sep	easy	40%	892.31	892.31	0.00%	169.04	0	17.39%	37.70%
Sep	medium	10%	1,003.58	1,003.58	0.00%	153.13	0	17.40%	37.65%
Sep	medium	20%	1,105.11	1,105.11	0.00%	160.34	0	17.40%	37.65%
Sep	medium	30%	1,287.59	1,287.59	0.00%	162.91	0	17.40%	37.64%
Sep	medium	40%	1,635.60	1,635.60	0.00%	177.98	0	17.40%	37.65%
Sep	difficult	10%	2,470.21	2,461.54	0.35%	432.28	1	17.51%	37.70%
Sep	difficult	20%	2,619.64	2,608.15	0.44%	436.89	1	17.51%	37.70%
Sep	difficult	30%	2,900.44	2,885.52	0.52%	544.09	1	17.51%	37.70%
Sep	difficult	40%	3,477.41	3,462.06	0.44%	607.60	1	17.51%	37.70%

Table 4: Gurobi results for instances of size 65%.

Flight plan	Difficulty	Reduction	z^{IP}	z^{LP}	Gap	t	nn	$col_{\%}$	$row_{\%}$
Jan	easy	10%	63.94	63.94	0.00%	303.15	0	16.61%	36.13%
Jan	easy	20%	137.28	137.28	0.00%	306.54	0	16.61%	36.13%
Jan	easy	30%	462.09	462.09	0.00%	315.67	0	16.61%	36.13%
Jan	easy	40%	1,368.88	1,366.80	0.15%	418.51	1	16.61%	36.14%
Jan	medium	10%	2,190.44	2,185.87	0.21%	707.22	1	16.68%	36.18%
Jan	medium	20%	2,315.25	2,315.25	0.00%	647.73	0	16.68%	36.18%
Jan	medium	30%	2,814.98	2,812.20	0.10%	909.54	1	16.68%	36.18%
Jan	medium	40%	3,992.40	3,983.01	0.24%	1,160.82	1	16.68%	36.19%
Jan	difficult	10%	3,921.93	3,912.03	0.25%	1,806.97	1	16.75%	36.28%
Jan	difficult	20%	4,079.09	4,067.67	0.28%	1,956.49	1	16.75%	36.28%
Jan	difficult	30%	4,445.04	4,434.00	0.25%	2,105.56	1	16.75%	36.28%
Jan	difficult	40%	5,146.37	5,126.39	0.39%	2,599.7	1	16.75%	36.28%
May	easy	10%	113.83	113.83	0.00%	432.83	0	14.87%	33.35%
May	easy	20%	412.49	412.49	0.00%	442.17	0	14.87%	33.35%
May	easy	30%	1,134.42	1,134.42	0.00%	474.99	0	14.87%	33.35%
May	easy	40%	2,751.35	2,745.71	0.21%	744.24	1	14.87%	33.35%
May	medium	10%	2,100.63	2,092.90	0.37%	970.32	1	14.91%	33.39%
May	medium	20%	2,572.71	2,570.74	0.08%	1,043.58	1	14.91%	33.39%
May	medium	30%	3,688.33	3,679.38	0.24%	1,241.99	1	14.91%	33.39%
May	medium	40%	5,812.79	5,812.79	0.00%	1,832.07	0	14.91%	33.39%
Sep	easy	10%	41.06	41.06	0.00%	387.21	0	15.73%	34.74%
Sep	easy	20%	46.67	46.67	0.00%	372.20	0	15.73%	34.74%
Sep	easy	30%	71.64	71.64	0.00%	369.44	0	15.73%	34.74%
Sep	easy	40%	114.07	114.07	0.00%	368.10	0	15.73%	34.74%
Sep	medium	10%	2,413.05	2,406.19	0.29%	769.83	1	15.78%	34.76%
Sep	medium	20%	2,523.37	2,516.18	0.29%	817.48	1	15.78%	34.76%
Sep	medium	30%	2,794.17	2,786.20	0.29%	863.04	1	15.78%	34.76%
Sep	medium	40%	3,583.07	3,569.65	0.38%	999.19	1	15.78%	34.76%

The first three columns of each table show the date of the flight plan, the level of difficulty and the capacity reduction due to the storm. This is followed by z^{IP} and z^{LP}, the objective function value of the integer solution and LP relaxation, respectively. Gap is the integrality gap, defined as $100\frac{z^{\text{IP}}-z^{\text{LP}}}{z^{\text{LP}}}$. Column t is the wall time in seconds to obtain the integer solution, and nn the number of nodes explored in the Branch & Bound (B&B) tree. A value of 0 means that the solution was obtained at the root node. Note that nn, along with Gap, measures the strength of the IP formulation. $col_{\%}$ and $row_{\%}$ represent the percentage of columns and rows that Gurobi deleted in the preprocessing phase.

The following observations relate to the values in both tables:

1. Gurobi's solution times are within a range that is acceptable in practice for the ATFM problem (43 minutes in the worst case).

2. Storm capacity reduction has a significant impact on the solution time. For example, for the 30%-size instances of January of medium difficulty, the variability in time is $(338.83 - 180.84)/338.83 = 0.466$, that is, a 46% of the worst time. It can be corroborated that a variability larger than 20% occurs in 10 out of the 16 possible cases in the tables.

3. The size and difficulty of the problem have even a bigger impact on the time required to solve the problem. For the size, compare for example the solution time of instance May, medium difficulty and 10% reduction in both tables. For the difficulty, compare for example in Table 3 the solution time of the instance Sep, medium difficulty and 10% reduction with that of Sep, difficult and 10%.

4. The integrality gap is always below 0.6%, being 0% in 30 out of 64 instances. In those cases, the problem was solved at the root node $nn = 0$. This shows the strength of the LP relaxation of the formulation for the problem.

5. The last two columns show that Gurobi achieved a significant reduction of the original size of the problem.

Note that the results were obtained for the cost structure discussed in the data set generation and Subsection 2.5.1, which is particularly suited for ATFM problems. To test the sensitivity of Gurobi to the cost structure in the IP formulation, we generated costs from a continuous uniform distribution $(0, 100)$ for the arcs in the networks in our problem test set. This new cost structure turned out to increase the difficulty of the problems in such a way that, out of the 108 instances, Gurobi was able to solve only 24. All of these instances belong to the sets of size 30%. The results for the experiments with this cost structure are exposed later on jointly with the performance of the algorithm.

4.2.2 *Matheuristic and Lagrangian Relaxation Results*

We now discuss the computational results obtained when solving the ATFM instances with the matheuristic and the Lagrangian relaxation. This discussion includes: the parameter setting of the algorithm, the performance analysis of the solution methods and insights into their performance.

4.2.2.1 *Parameter setting*

We used ParamILS (Hutter, Hoos, and Stützle [77]) to set the values for the parameters of our matheuristic procedure, except for the size of the solution pool, which is discussed below. ParamILS is an automated system for parameter setting and algorithm configuration. The goal of this system is to optimize a target algorithm's performance on a given class of problem instances by varying a set of ordinal and/or categorical parameters. We measured performance as a function of quality and solution time. We made slight adjustments to the settings found by ParamILS based on our knowledge of our search procedure. The parameter values that we used for our experimentation are shown in Table 5. The penalty value is about 4 times the maximum cost in the networks.

Table 5: Parameters values in experimentation.

$maxIter$	$minIterIP$	$maxNets$	$penalty$	α	β	δ	γ
100	20	$5\%\|\mathcal{S}\|$	6000	50%	95%	2%	30%

To set the size of the solution pool, we conducted a series of experiments. Concretely, we used the instances of size 30% to test four different pool size values (16, 32, 48, and 64). The results are summarized in Figure 19 (a) and (b), where violin plots show, for each difficulty level and pool size, the distribution of the optimalitiy gap and the speed-up with respect to Gurobi's solutions. Given the $m-$th instance and the $k-$th execution of our heuristic procedure on that instance, the optimality gap is computed as $100\frac{z^{\mathrm{h}}_{m,k}-z^{\mathrm{IP}}_{m}}{z^{\mathrm{IP}}_{m}}$, where $z^{\mathrm{h}}_{m,k}$ and z^{IP}_{m} denote the objective function values for the solutions found with our heuristic and with Gurobi, respectively. Similarly, the speed-up is computed as $\frac{t^{\mathrm{IP}}_{m}}{t^{\mathrm{h}}_{m,k}}$, where $t^{\mathrm{h}}_{m,k}$ and t^{IP}_{m} denote the wall time required by our procedure and the one required by Gurobi, respectively.

Figure 19 (a) shows the importance of the pool size as the difficulty of the problem increases. Larger pool sizes achieve smaller optimality gaps with lower variability. For instance, for a pool size of 64, the optimality gap is always less than 5%.

The dashed line in Figure 19 (b) shows the speed-up value of 1, which is when the heuristic solution time is the same as Gurobi's. Values below this line mean that the optimal solution was found and confirmed faster than the running time of the heuristic. This happened in some of the easy cases, particularly, for the two largest pool sizes. The heuristic finished before Gurobi in the medium and difficult instances. The run-

time difference is more significant in difficult cases, where the minimum speed-up is 1.5.

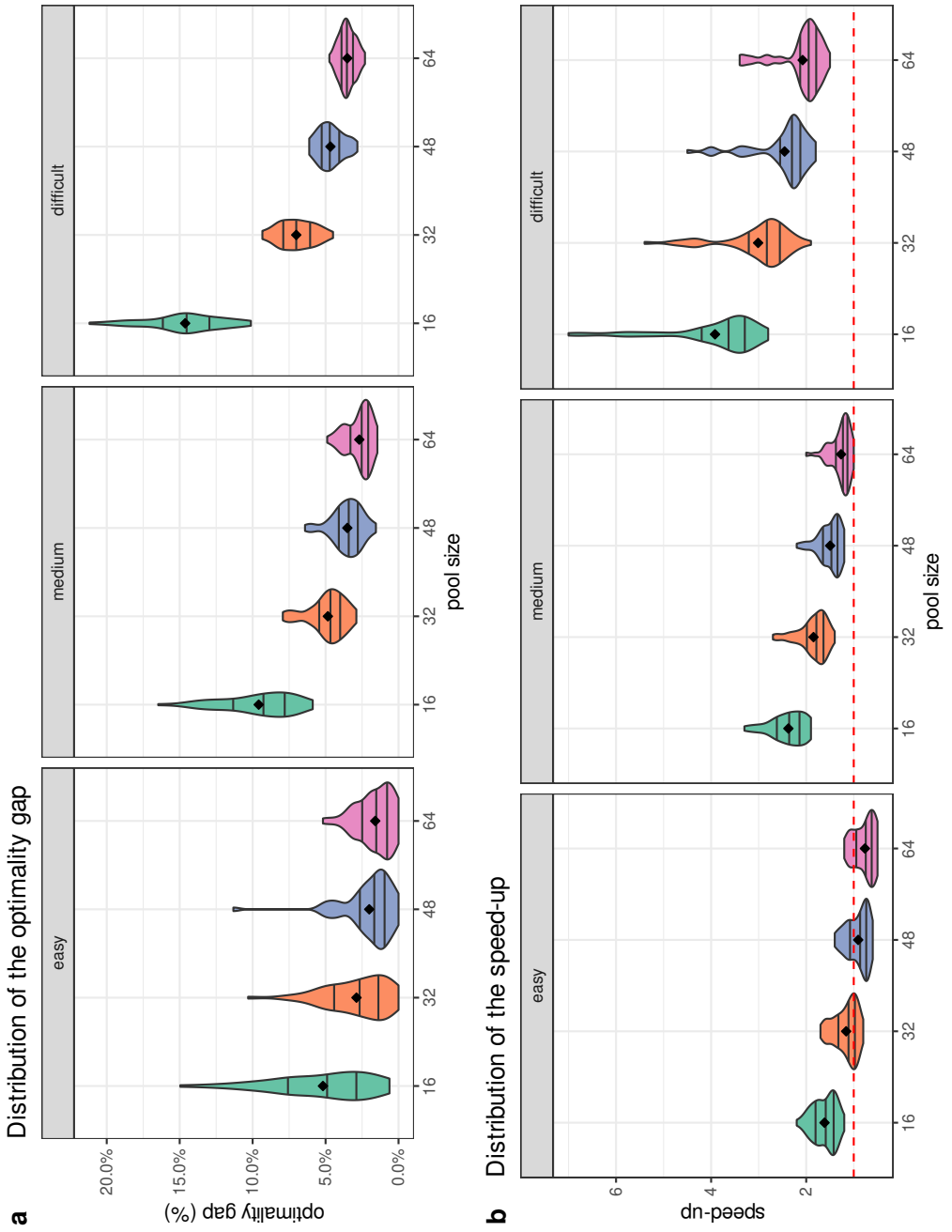


Figure 19: Effect of the pool size for instances of size 30%.

In terms of computational effort, we observed that our parallelization scheme for generating the pool of solutions scaled linearly. That is, twice as much time is required to generate the same amount of solutions with half the threads. This fact will be important when analyzing the insights of the algorithm later on.

Extrapolating from the experiments with instances of size 30% and in order to balance solution quality and computational effort, we chose a pool of 80 solutions for instances of size 65% and a pool of 96 solutions for instances with the complete flight plans.

4.2.2.2 *Performance assessment*

We used the smallest problem instances to find the best values for our search parameters as well as adjusting the size of the solution pool. We now assess the performance of the procedure with the larger problem instances. As shown in Table 4, Gurobi was able to solve 28 of the 36 instances of size of 65% of the original flight plans, namely all the sets for January and the "easy" and "medium" sets for May and September. Therefore, we are able to calculate optimally gaps and speed-up values when applying our procedure to these problem instances. Figure 20 (a) and (b) summarize the results.

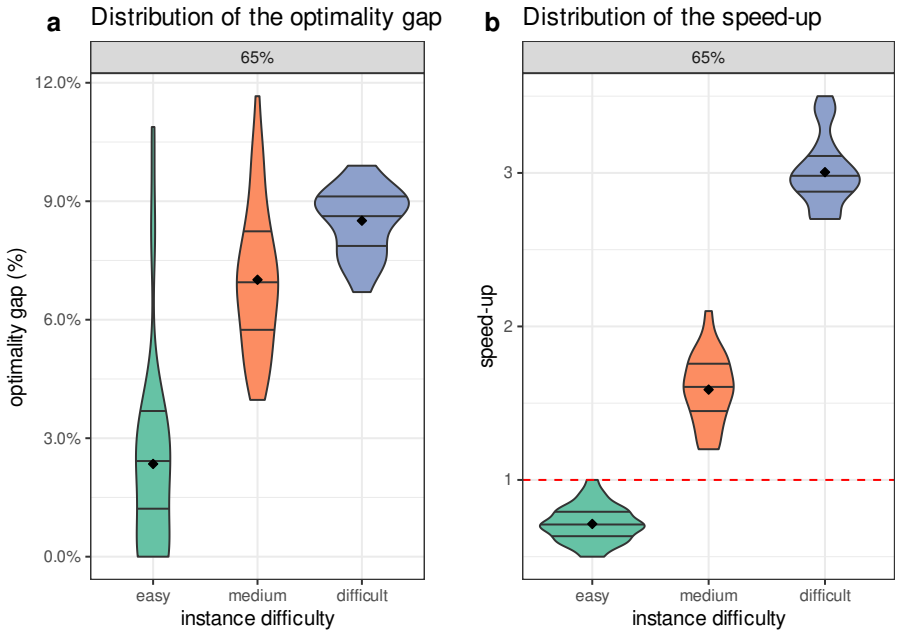


Figure 20: Results of the algorithm for instances of size 65% that exact methods could also solve.

The first figure shows that the proposed heuristic generates high-quality solutions for these instances. For the easy cases, the heuristic solutions are, on average, less than

3% away from optimality. For the medium cases, in more than 75% of the times the optimality gap is below 9% and the average is less than 7.5%. For the difficult cases, the optimality gap is below 10%, with an average of 8.5%. In terms of computational effort, our speed-up calculation shows that the heuristic was faster than the exact method in the medium and difficult cases, but not in the easy ones. It is interesting to point out that in the instances when the proposed heuristic is faster than the exact method, Gurobi is still solving the LP relaxation at the root node when the heuristic has already finished.

A possible conclusion from these results is that Gurobi should be the preferred method to solve the easy problems, since it can confirm optimality before our procedure finishes. However, this result depends on the cost structure. Recall that when we used randomly generated costs, Gurobi was only able to solve 24 out of the 108 test problems. A comparison of optimality gap and speed-up between our procedure and Gurobi for those 24 instances is shown in Figure 21 (a) and (b). We observe that, under this cost structure, the solutions achieved by our procedure are near-optimal, and they are reached faster than Gurobi (with a minimum speed-up of 1.5).

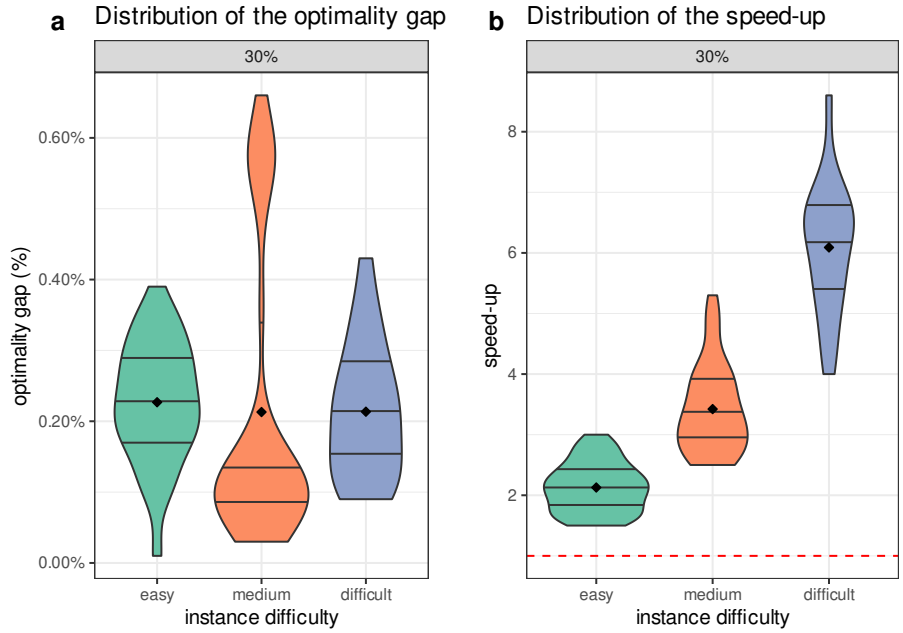


Figure 21: Results using a random cost structure.

This experiment shows that the performance of the exact method within Gurobi is sensitive to the cost structure. On the other hand, our heuristic-based approach exhibits a robust performance across cost structures. Furthermore, we observed that the

solutions that we found after the combination method have an optimality gap of 1%. Therefore, in cases where computational time is limited, it would seem sensible to skip the local search phase when tackling problems with this cost structure.

For the flight plans of May and September under the difficult scenario in the 65% set, and all the instances in the 100% set, Gurobi, due to memory limitations, was unable to even solve the LP relaxation of the problem. Therefore, we were not able to assess the quality of our solutions with the lower bound corresponding to the LP relaxation. We attempted to find solutions for these instances with LocalSolver ([93]), a general-purpose optimizer based on metaheuristic principles and methodologies, but this software also failed to handle them. In order to obtain lower bounds, we employed the Lagrangian Relaxation resulting from dualizing the capacity constraints as described in Subsection 3.3.3.2. Recall from Corollary 2 that the optimum of the Lagrangian dual problem is guaranteed to be equal to the LP relaxation of the IP formulation of the SRC-MSPP (model (3.5)-(3.8) on page 57). Based on the gaps reported in Table 3 and 4, the lower bounds resulting from the Lagrangian Relaxation are expected to be tight.

We attempted to solve the Lagrangian Relaxation using the subgradient and surrogate methods discussed at the end of Subsection 3.3.3.1. For the subgradient, the step size rule given by $\theta^j = \mu_j \frac{z^{\text{LR}^*} - z(\boldsymbol{\lambda}^j)}{\|\boldsymbol{\phi}^j\|^2}$ was employed. In preliminary experiments, none of the two methods exhibited a good behavior with the common parameters specified in literature, so we decided just to focus on the subgradient method, but using a dynamic step size rule. Concretely, we started with a big value of μ_j and reduced it after every 300 iterations. Using big μ_j values at the beginning of the subgradient method was established after observing that term $\|\boldsymbol{\phi}^j\|^2$ was too big, leading to small step sizes at each iteration and a bad global convergence. Big μ_j values solved that issue, but keeping them during the whole algorithm produced an erratic behavior and prevented from convergence, hence the reduction step after every 300 iterations. As an example, the next sequence of values was employed for μ_j when solving some of the instances: 300, 150, 70, 30 and 5. Regarding the stopping rule of the subgradient method, we stopped when one of the next conditions hold: i) The number of iterations was greater than $5,000$, ii) The distance between multipliers was less than 10^{-6}, or iii) An optimal solution (either primal or dual) was reached.

Despite the modifications done to the subgradient method, we still noticed the following difficulties:

1. The sequence of μ_j values employed depended on the difficulty of the problem to solve.

2. The algorithm required long solution times just to obtain lower bounds, i. e., no primal feasible solutions were produced at any moment.

These points make that the methods based on the Lagrangian Relaxation require further research to become a feasible option to address the problem, and not just to provide lower bounds. In this respect, Chapter 5 exposes some possible solutions to explore.

Respect to the results obtained, Figure 22 shows, for each instance and size group, the gap between all the solutions found throughout our experimentation and the Lagrangian bound. The dashed line is the mean gap of the heuristic solution with respect to the Lagrangian bound. The band around the mean represents the range of gaps obtained from the 5 runs of our procedure. Similarly, the solid line in the figure shows the gap of Gurobi's solutions (when available) with the Lagrangian bound.

The general observation from Figure 22 is that the deviation from the Lagrangian bounds increases with the size of the problem. This is true for both the heuristic and the optimal solutions. For the full flight plans, our procedure obtained, in the worst case, solutions with average gaps of 16%. However, in most cases the gap varied between 4% and 14.5%. Considering that these gaps are against a lower bound and not the optimal solution, these results are more than reasonable for the practical application that we studied.

For the sake of completeness, we show in Table 6 the results of the matheuristic algorithm for the cases that Gurobi could not solve due to memory limitations.

The first four columns of the table show the date of the flight plan, the percentage of flights chosen from the flight plan, the level of difficulty and the capacity reduction due to the storm. This is followed by $\bar{z}$, the solution's average objective function value in the 5 runs. Δ_z is the gap between the best and worst solution obtained in the 5 runs. It measures how much changes the solution between executions. Column $\bar{t}$ is the average wall time in seconds to obtain the solution, and column Δ_t is the ratio between the worst and best solution times obtained. As happens with Δ_z, this is a measure of the robustness of the algorithm.

Respect to the values of the table: i) Δ_z values are most of the times around 2%, meaning that there is not much variability in the quality of the solution between runs, ii) In the worst case, the average computational time is below 40 minutes, being less

than 25 minutes in 32 of the 44 instances. These times, which are within the valid range for ATFM, show the capability of the algorithm to tackle big and difficult instances of the problem. And iii) Δ_t shows that computational times between runs do not differ excessively. Only in 12 out of 44 instances the variation was larger than 15%.

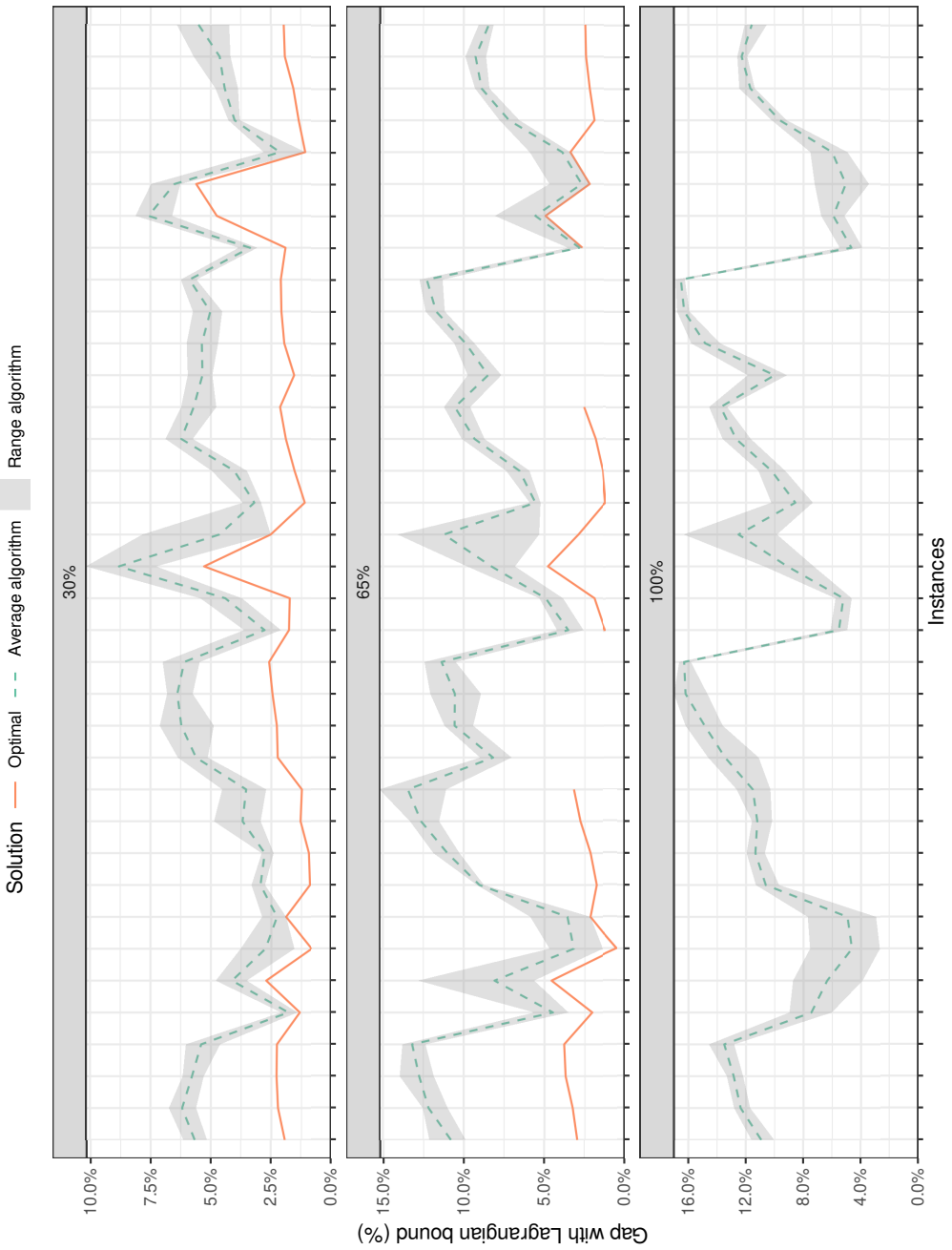


Figure 22: Deviation from Lagrangian lower bounds.

Table 6: Subset of instances solved only by the algorithm.

Flight plan	Size	Difficulty	Reduction	$\bar{z}$	Δ_z	$\bar{t}$	Δ_t
May	65%	difficult	10%	3855.34	1.74%	747.33	1.15
May	65%	difficult	20%	4165.98	2.88%	783.06	1.08
May	65%	difficult	30%	5069.83	1.62%	908.54	1.09
May	65%	difficult	40%	7479.85	1.82%	1207.91	1.06
Sep	65%	difficult	10%	4722.99	1.28%	752.46	1.14
Sep	65%	difficult	20%	5140.61	1.07%	777.71	1.16
Sep	65%	difficult	30%	6034.70	1.14%	959.48	1.13
Sep	65%	difficult	40%	8001.80	1.93%	1259.62	1.31
Jan	100%	easy	10%	47.23	4.62%	765.30	1.08
Jan	100%	easy	20%	50.27	4.72%	781.46	1.06
Jan	100%	easy	30%	94.05	4.66%	669.44	1.06
Jan	100%	easy	40%	156.08	2.78%	683.83	1.06
Jan	100%	medium	10%	2543.51	2.13%	951.27	1.18
Jan	100%	medium	20%	2552.13	1.27%	974.65	1.08
Jan	100%	medium	30%	2617.34	1.15%	955.93	1.15
Jan	100%	medium	40%	2683.86	1.47%	947.25	1.12
Jan	100%	difficult	10%	5418.48	1.55%	1794.73	1.16
Jan	100%	difficult	20%	5678.10	1.01%	1811.53	1.14
Jan	100%	difficult	30%	6241.71	1.00%	1978.41	1.29
Jan	100%	difficult	40%	7321.88	1.44%	2193.39	1.25
May	100%	easy	10%	165.86	5.10%	861.24	1.07
May	100%	easy	20%	561.57	3.21%	1087.71	1.16
May	100%	easy	30%	1637.88	1.07%	1689.4	1.08
May	100%	easy	40%	3827.27	1.12%	1233.41	1.08
May	100%	medium	10%	2236.06	1.10%	902.00	1.08
May	100%	medium	20%	2691.73	1.75%	942.26	1.04
May	100%	medium	30%	3841.26	1.71%	1041.81	1.05
May	100%	medium	40%	6365.82	2.69%	1254.65	1.05
May	100%	difficult	10%	4728.95	0.70%	1356.94	1.05
May	100%	difficult	20%	4913.83	2.02%	1390.09	1.11
May	100%	difficult	30%	5394.70	2.27%	1502.27	1.12
May	100%	difficult	40%	6734.75	3.16%	1639.46	1.15
Sep	100%	easy	10%	118.97	2.45%	841.09	1.08
Sep	100%	easy	20%	425.72	3.65%	1079.66	1.06
Sep	100%	easy	30%	1333.18	1.54%	1270.19	1.21
Sep	100%	easy	40%	3794.71	1.51%	1687.39	1.20
Sep	100%	medium	10%	3118.49	1.39%	952.18	1.05
Sep	100%	medium	20%	3454.52	0.61%	944.66	1.06
Sep	100%	medium	30%	4484.01	0.97%	1143.31	1.07
Sep	100%	medium	40%	7046.61	0.95%	1515.47	1.13
Sep	100%	difficult	10%	6264.73	0.56%	1421.54	1.09
Sep	100%	difficult	20%	6970.70	0.74%	1535.93	1.12
Sep	100%	difficult	30%	8678.90	1.77%	1718.51	1.22
Sep	100%	difficult	40%	11981.68	2.51%	2336.88	1.12

4.2.2.3 *Insight analysis of the matheurisitic*

The preceding results show that we are able to produce high-quality solutions in reasonable computational times, and that our procedure was able to find solutions to problems that a commercial solver could not handle. The purpose of the following analysis is to provide insights into the performance of our procedure.

Figure 23 shows the average percentage of time that our procedure employs in each phase, as a function of the problem size and the level of difficulty.

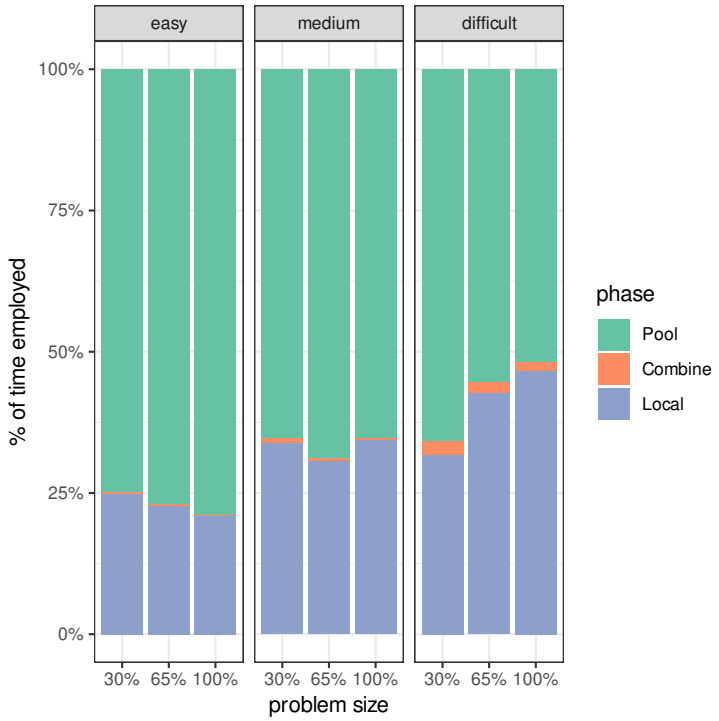


Figure 23: Percentage of solution time in each phase of the proposed solution procedure.

We observe that: 1) The solution pool generation represents the bottleneck, consuming consuming from 52% to 79% of the solution time; 2) Combining solutions consumes a negligible amount of time (2.5% in the worst case); and 3) The time required by local search is sensitive to the difficulty of the problem, ranging from 21% to 46% for the biggest and most difficult instances.

The good news of the solution pool generation being the procedure's bottleneck is that, as mentioned before, the parallelization of this phase linearly scales with the number of threads. This means that when generating a pool of 96 solutions, if instead of 8 threads we had 48, the time in this phase would have been $\frac{1}{6}$ of the time that we report.

In such situation, the procedure will most likely also outperform in time to the plain usage of the solver in the easy cases. Furthermore, if a larger number of threads was available, the procedure could be set up to generate more solutions, which in turn could improve the quality of the results while maintaining the time performance.

Figure 24 (a) and (b) shows the percentage of improvement in solution quality achieved after each phase of the procedure, as a function of the problem size and difficulty level.

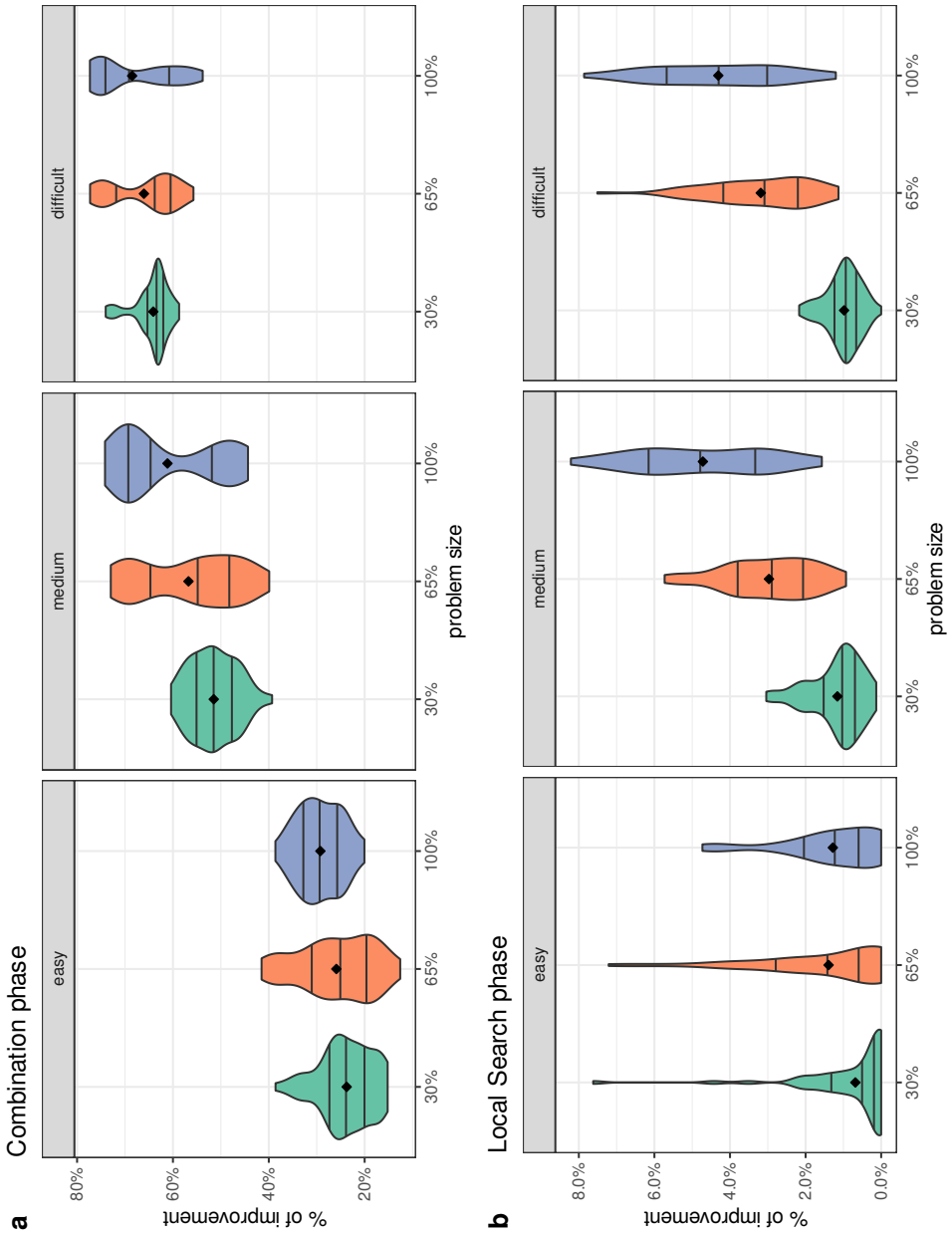


Figure 24: Improvement achieved after each phase of the proposed procedure.

Figure 24 (a) reveals the large improvement achieved by the solution combination phase, particularly as the difficulty of the problem increases. The large solution-quality improvement combined with the modest amount of computational time make this phase one of the key elements for the success of the solution method.

When compared to the solution combination phase, the local search achieves a significantly smaller improvement in solution quality. This is mainly due to the reduced opportunities for improvement after solutions are combined. Nonetheless, the improvement achieved by the local search justifies its computational requirements and therefore its inclusion as part of the solution process.

For the pool sizes employed, it was exposed before how the algorithm proposed achieved good optimality gaps. Table 7 and 8 show that these gaps can be further reduced by simply using larger pool sizes. Concretely, we employed pools of 200 solutions for the instances of size 30%, and 300 for instances of size 65%. In this case, the results in the tables correspond to a single run of the algorithm.

The first three columns of each table show the date of the flight plan, the level of difficulty and the capacity reduction due to the storm. The next one (z) shows the solution's objective function value. Gap stands for the optimality gap. Columns t_{Pool}, t_{Comb}, t_{LS} are, respectively, the time in seconds required by the algorithm in the phases of: Generating the pool of solutions, combining the solutions in the pool and local search. The last column, t, shows the total time in seconds required by the algorithm.

As can be seen, for the pool sizes employed in this part of the experimentation, all the gaps were below 7%, being only larger than 2% in 16 of the 64 instances. That is, the algorithm proposed is capable of finding near-optimal solutions when using a sufficiently large pool. Respect to the computational times, note that: 1) The pool generation, as expected, keeps being the bottleneck of the algorithm, and 2) Except for 7 instances, the computational time was less than 30 minutes. These times are valid for the ATFM application, but larger than the times required by Gurobi. Thus, larger pool sizes result in near-optimal solutions, but large pool sizes are only recommended when a computer with more than 8 threads is available.

To conclude the analysis of the proposed procedure, we point out that we designed the algorithmic elements to take advantage of the structure of the problem. In particular, the generation of the solution pool exploits a very specific characteristic of our problem, which turns out to also be true in the context of the resource constrained shortest path problem. In general, the least expensive arcs are also those that consume the most limited resources at a higher rate. The key is to generate a set of diverse solutions that

include the use of arcs with limited or nonexistent consumption of the most limited resources. Typically, these arcs are the most costly or else the trivial solution in which all origins and destinations are connected by their shortest paths would be feasible and therefore optimal. The controlled random element included in the generation of solutions results in a diversity of arcs in the solution pool that is then exploited by the combination method.

Table 7: Algorithm results for instances of size 30% when using a pool of 200 solutions.

Flight plan	Difficulty	Reduction	z	Gap	t_{Pool}	t_{Comb}	t_{LS}	t
Jan	easy	10%	54.08	0.26%	174.06	0.35	42.65	217.06
Jan	easy	20%	140.71	0.00%	234.53	0.38	42.09	276.99
Jan	easy	30%	346.72	0.56%	436.48	0.45	45.90	482.83
Jan	easy	40%	801.51	0.25%	616.50	0.71	48.54	665.74
Jan	medium	10%	1435.24	1.18%	247.95	2.10	51.22	301.26
Jan	medium	20%	1563.68	0.38%	264.04	2.36	53.83	320.23
Jan	medium	30%	1858.28	0.64%	340.51	2.55	56.09	399.15
Jan	medium	40%	2584.16	0.53%	384.88	4.11	58.67	447.67
Jan	difficult	10%	2353.34	0.91%	298.06	5.33	54.74	358.13
Jan	difficult	20%	2432.39	1.56%	294.46	5.86	57.95	358.26
Jan	difficult	30%	2529.71	0.94%	325.39	6.62	64.93	396.94
Jan	difficult	40%	2816.06	1.33%	334.72	9.42	78.39	422.54
May	easy	10%	68.70	0.00%	283.75	0.46	53.32	337.53
May	easy	20%	199.46	1.98%	363.43	0.52	53.92	417.87
May	easy	30%	578.22	0.19%	769.83	0.66	60.18	830.68
May	easy	40%	1457.59	0.08%	807.41	1.13	56.99	865.53
May	medium	10%	1409.47	1.48%	342.16	2.23	51.93	396.31
May	medium	20%	1581.44	1.39%	428.45	2.68	52.16	483.29
May	medium	30%	1977.48	1.10%	442.06	3.08	54.85	499.99
May	medium	40%	2775.31	1.10%	523.70	4.49	68.40	596.58
May	difficult	10%	2722.61	1.44%	629.78	9.24	80.27	719.29
May	difficult	20%	2812.17	1.99%	645.85	10.53	74.48	730.86
May	difficult	30%	3135.97	1.91%	640.23	16.98	76.44	733.65
May	difficult	40%	3801.92	0.72%	625.42	24.13	91.58	741.13
Sep	easy	10%	83.08	0.91%	409.54	0.45	54.54	464.53
Sep	easy	20%	152.74	0.00%	421.65	0.46	53.36	475.46
Sep	easy	30%	379.46	0.05%	581.61	0.53	46.93	629.08
Sep	easy	40%	895.02	0.30%	680.48	0.80	51.86	733.15
Sep	medium	10%	1016.61	1.30%	233.17	0.93	53.06	287.16
Sep	medium	20%	1117.07	1.08%	267.70	1.02	47.74	316.45
Sep	medium	30%	1305.57	1.40%	278.67	1.13	53.04	332.84
Sep	medium	40%	1650.21	0.89%	337.18	1.55	58.73	397.46
Sep	difficult	10%	2498.53	1.15%	510.83	7.06	65.01	582.90
Sep	difficult	20%	2655.36	1.36%	491.01	7.66	71.41	570.08
Sep	difficult	30%	2937.15	1.27%	556.60	11.71	76.86	645.16
Sep	difficult	40%	3531.24	1.55%	581.91	21.73	97.04	700.69

Table 8: Algorithm results for instances of size 65% when using a pool of 300 solutions.

Flight plan	Difficulty	Reduction	z	Gap	t_{Pool}	t_{Comb}	t_{LS}	t
Jan	easy	10%	63.94	0.00%	919.43	0.77	117.71	1037.91
Jan	easy	20%	137.28	0.00%	1068.11	0.80	114.78	1183.69
Jan	easy	30%	463.82	0.37%	1454.79	0.97	122.82	1578.58
Jan	easy	40%	1373.50	0.34%	1870.31	1.61	122.99	1994.91
Jan	medium	10%	2315.04	5.69%	1042.12	2.77	145.56	1190.45
Jan	medium	20%	2464.53	6.45%	1014.33	3.25	160.20	1177.78
Jan	medium	30%	2982.49	5.95%	1155.65	4.56	177.05	1337.26
Jan	medium	40%	4125.23	3.33%	1328.93	8.01	225.94	1562.88
Jan	difficult	10%	4080.30	4.04%	1280.86	14.49	241.28	1536.63
Jan	difficult	20%	4256.71	4.35%	1283.01	18.02	253.55	1554.58
Jan	difficult	30%	4623.09	4.01%	1372.58	21.12	280.21	1673.91
Jan	difficult	40%	5310.65	3.19%	1600.57	35.85	251.57	1887.99
May	easy	10%	114.83	0.88%	1090.78	1.00	133.28	1225.06
May	easy	20%	415.99	0.85%	1490.67	1.15	146.18	1637.99
May	easy	30%	1145.14	0.94%	2739.68	1.63	141.49	2882.80
May	easy	40%	2765.46	0.51%	2572.17	3.22	152.80	2728.19
May	medium	10%	2188.75	4.19%	1520.24	3.53	199.58	1723.35
May	medium	20%	2689.60	4.54%	1847.65	4.70	212.12	2064.47
May	medium	30%	3794.71	2.88%	2697.38	7.96	237.96	2943.30
May	medium	40%	5950.58	2.37%	3547.94	15.1	293.16	3856.20
Sep	easy	10%	41.06	0.00%	1455.26	0.92	124.48	1580.66
Sep	easy	20%	46.67	0.00%	1509.18	0.92	126.32	1636.42
Sep	easy	30%	71.64	0.00%	1537.06	0.91	126.15	1664.11
Sep	easy	40%	114.07	0.00%	1591.90	0.93	125.99	1718.82
Sep	medium	10%	2505.77	3.84%	1133.80	3.82	153.67	1291.29
Sep	medium	20%	2616.56	3.69%	1173.49	3.92	163.08	1340.49
Sep	medium	30%	2890.33	3.44%	1151.64	4.27	171.74	1327.65
Sep	medium	40%	3686.11	2.88%	1385.44	6.15	150.68	1542.27

4.2.3 *Flight Plans Modifications*

We conclude the chapter by showing the flight plans modifications obtained in one execution of the algorithm. The results for the instances of size 30%, 65% and 100% are shown in Table 9, 10 and 11, respectively.

The first three columns of each table show the date of the flight plan, the level of difficulty and the capacity reduction due to the storm. The next one ($\mathcal{F}_m$) shows the number of flights with at least one modification respect to their original flight plan. The next two columns show the number of flights which were assigned some ground delay ($\mathcal{F}_g$), distinguishing between those which also landed late ($\mathcal{F}_{g,l}$) and those which did not ($\mathcal{F}_{g,\bar{l}}$), i. e., which landed on time or earlier than scheduled. The next two columns

show the number of flights which changed their speed ($\mathcal{F}_{sc}$), distinguishing between delays ($\mathcal{F}_d$) and speed increases ($\mathcal{F}_i$). Finally, the last two columns show the number of flights arriving at destination in a different time than scheduled ($\mathcal{F}_{ad}$), distinguishing between late ($\mathcal{F}_l$) and early ($\mathcal{F}_e$) arrivals.

The following observations relate to the values in the tables:

1. The number of modified flights ($\mathcal{F}_m$) never exceeded the 17%, being less than 10% in 82 cases out of 108.

2. The number of modified flights is less than the sum of flights with ground delay ($\mathcal{F}_g$), speed change ($\mathcal{F}_{sc}$) and lading out of schedule ($\mathcal{F}_{ad}$). This is because some of the flights suffered from multiple modifications simultaneously.

3. No information about the alternative routes is shown because their usage was not required in the solution. Alternative routes have a high cost so conflicts are solved (when possible) just by means of time-schedule modifications, as occurs in daily operations.

4. Ground delays ($\mathcal{F}_g$) were more used than air delays ($\mathcal{F}_d$). This is customary in ATFM operations given that ground delays are safer than air delays.

5. Column $\mathcal{F}_{g,l}$ shows that most of the late departures resulted in a late landing.

6. Column $\mathcal{F}_{g,\bar{l}}$ shows how many times a late departure was compensated through a speed increase ($\mathcal{F}_i$). The rest of the times, speed increases were used to compensate delays, but without forcing to land on time. That is, it is preferable to land a little bit later than continuously be changing the speed.

7. Most of the speed changes ($\mathcal{F}_{sc}$) are speed increases ($\mathcal{F}_i$). These are to compensate as much as possible, any type of delay and reduce the late arrivals.

8. The number of late landings ($\mathcal{F}_l$) not explained by the ground ($\mathcal{F}_{g,l}$) and air ($\mathcal{F}_d$) delays are because of the propagation-delay effect that occurs with continued flights.

9. Most of the arrivals at destination out of schedule ($\mathcal{F}_{ad}$) are late landings ($\mathcal{F}_l$). Early landings ($\mathcal{F}_e$) are to reduce the number of continued flights suffering from the propagation-delay effect.

Table 9: Flight modifications in the solution of the algorithm for instances of size 30%.

				$\mathcal{F}_g$		$\mathcal{F}_{sc}$		$\mathcal{F}_{ad}$	
Flight plan	Difficulty	Reduction	$\mathcal{F}_m$	$\mathcal{F}_{g,l}$	$\mathcal{F}_{g,\bar{l}}$	$\mathcal{F}_d$	$\mathcal{F}_i$	$\mathcal{F}_l$	$\mathcal{F}_e$
Jan	easy	10%	16	8	3	1	6	9	1
Jan	easy	20%	38	16	10	1	18	19	4
Jan	easy	30%	81	36	14	2	37	48	8
Jan	easy	40%	172	72	24	7	74	106	18
Jan	medium	10%	429	151	81	5	188	251	43
Jan	medium	20%	467	156	76	6	202	279	55
Jan	medium	30%	515	179	86	8	219	306	58
Jan	medium	40%	639	222	85	9	265	398	75
Jan	difficult	10%	680	249	97	12	260	441	58
Jan	difficult	20%	691	250	104	10	279	433	68
Jan	difficult	30%	708	249	120	14	287	441	63
Jan	difficult	40%	736	258	115	22	307	452	79
May	easy	10%	21	8	6	0	8	13	0
May	easy	20%	45	15	13	0	24	23	2
May	easy	30%	114	32	32	0	66	54	17
May	easy	40%	241	68	54	3	136	128	31
May	medium	10%	399	142	77	12	187	225	37
May	medium	20%	442	147	103	16	215	232	45
May	medium	30%	515	172	100	14	246	286	57
May	medium	40%	630	212	123	16	306	357	73
May	difficult	10%	753	264	133	15	345	440	75
May	difficult	20%	755	274	127	14	337	450	85
May	difficult	30%	843	300	145	17	370	504	82
May	difficult	40%	948	355	145	15	419	580	94
Sep	easy	10%	22	5	6	1	9	12	2
Sep	easy	20%	31	6	13	1	18	12	3
Sep	easy	30%	83	21	26	2	44	39	5
Sep	easy	40%	164	41	35	1	86	87	23
Sep	medium	10%	271	87	80	2	147	130	25
Sep	medium	20%	291	98	79	1	153	149	27
Sep	medium	30%	330	109	82	2	167	173	41
Sep	medium	40%	388	128	92	3	188	217	41
Sep	difficult	10%	703	223	136	10	300	417	56
Sep	difficult	20%	714	223	141	10	311	421	62
Sep	difficult	30%	786	253	150	13	340	464	68
Sep	difficult	40%	854	283	148	11	380	518	87

Table 10: Flight modifications in the solution of the algorithm for instances of size 65%.

				$\mathcal{F}_g$		$\mathcal{F}_{sc}$		$\mathcal{F}_{ad}$	
Flight plan	Difficulty	Reduction	$\mathcal{F}_m$	$\mathcal{F}_{g,l}$	$\mathcal{F}_{g,\bar{l}}$	$\mathcal{F}_d$	$\mathcal{F}_i$	$\mathcal{F}_l$	$\mathcal{F}_e$
Jan	easy	10%	19	8	4	0	10	9	1
Jan	easy	20%	42	19	7	1	19	22	4
Jan	easy	30%	122	48	28	1	64	63	11
Jan	easy	40%	304	114	67	5	167	154	27
Jan	medium	10%	639	236	141	11	297	368	52
Jan	medium	20%	662	240	140	9	312	375	63
Jan	medium	30%	774	279	165	11	357	450	68
Jan	medium	40%	983	336	191	21	467	572	103
Jan	difficult	10%	1,108	395	206	16	508	661	108
Jan	difficult	20%	1,177	422	211	16	505	724	108
Jan	difficult	30%	1,213	428	223	15	554	727	121
Jan	difficult	40%	1,290	462	235	21	617	754	141
May	easy	10%	32	11	12	0	20	12	4
May	easy	20%	105	37	31	1	59	48	13
May	easy	30%	262	87	75	2	147	127	27
May	easy	40%	549	183	112	5	302	294	78
May	medium	10%	609	205	171	8	321	302	60
May	medium	20%	703	236	181	12	359	353	75
May	medium	30%	920	301	226	19	474	470	101
May	medium	40%	1,251	400	259	22	643	678	161
May	difficult	10%	1,098	368	218	23	494	622	101
May	difficult	20%	1,170	392	207	26	499	687	108
May	difficult	30%	1,306	435	231	21	585	758	146
May	difficult	40%	1,707	569	250	30	751	1,042	196
Sep	easy	10%	15	4	5	0	10	5	2
Sep	easy	20%	17	4	7	0	12	5	2
Sep	easy	30%	22	4	10	1	16	5	4
Sep	easy	40%	29	8	10	2	20	9	6
Sep	medium	10%	674	226	154	13	319	365	58
Sep	medium	20%	712	243	163	13	337	385	58
Sep	medium	30%	776	261	168	17	357	432	76
Sep	medium	40%	923	345	173	19	415	540	87
Sep	difficult	10%	1,250	438	233	16	548	739	117
Sep	difficult	20%	1,311	456	241	22	574	771	129
Sep	difficult	30%	1,488	503	270	20	665	891	145
Sep	difficult	40%	1,733	595	296	26	793	1,033	199

Table 11: Flight modifications in the solution of the algorithm for instances of size 100%.

				$\mathcal{F}_g$		$\mathcal{F}_{sc}$		$\mathcal{F}_{ad}$	
Flight plan	Difficulty	Reduction	$\mathcal{F}_m$	$\mathcal{F}_{g,l}$	$\mathcal{F}_{g,\bar{l}}$	$\mathcal{F}_d$	$\mathcal{F}_i$	$\mathcal{F}_l$	$\mathcal{F}_e$
Jan	easy	10%	14	4	6	0	10	4	3
Jan	easy	20%	15	6	5	0	9	6	3
Jan	easy	30%	28	10	9	0	18	10	6
Jan	easy	40%	45	16	15	0	27	18	9
Jan	medium	10%	710	233	186	27	366	342	68
Jan	medium	20%	729	241	188	26	376	355	68
Jan	medium	30%	735	244	196	24	391	346	73
Jan	medium	40%	759	248	195	30	385	370	75
Jan	difficult	10%	1,413	497	283	31	649	806	124
Jan	difficult	20%	1,452	519	294	34	689	813	127
Jan	difficult	30%	1,534	529	310	40	735	850	150
Jan	difficult	40%	1,677	579	318	42	797	959	169
May	easy	10%	43	14	8	1	22	21	6
May	easy	20%	127	43	30	1	70	62	15
May	easy	30%	350	120	79	4	186	181	41
May	easy	40%	722	235	141	10	387	395	99
May	medium	10%	666	219	172	4	333	344	54
May	medium	20%	758	248	188	7	378	397	63
May	medium	30%	969	327	217	11	478	529	88
May	medium	40%	1,337	431	283	15	710	712	162
May	difficult	10%	1,323	423	282	20	632	726	117
May	difficult	20%	1,355	433	290	17	655	743	119
May	difficult	30%	1,433	447	321	13	704	773	134
May	difficult	40%	1,598	495	339	24	802	855	164
Sep	easy	10%	29	7	13	0	21	8	5
Sep	easy	20%	103	34	30	1	59	44	12
Sep	easy	30%	288	95	55	0	137	162	36
Sep	easy	40%	696	226	100	3	343	390	113
Sep	medium	10%	850	299	188	12	412	461	79
Sep	medium	20%	908	315	198	17	438	496	81
Sep	medium	30%	1,121	395	214	13	530	633	108
Sep	medium	40%	1,479	506	243	14	704	846	185
Sep	difficult	10%	1,676	616	271	14	679	1,056	139
Sep	difficult	20%	1,827	671	283	14	739	1,162	165
Sep	difficult	30%	2,079	752	306	28	897	1,294	221
Sep	difficult	40%	2,543	930	368	36	1,125	1,586	295

5

CONCLUSIONS AND FUTURE RESEARCH

This thesis has addressed the problem of Air Traffic Flow Management (ATFM) from a mathematical optimization perspective. We now detail all the contributions and conclusions by chapters.

Chapter 2 introduces the reader to the problem of air traffic saturation and the mechanisms that exist to deal with it, in particular, that of ATFM. The chapter also contains a literature review of the most recent works that employ mathematical optimization to cope with the ATFM problem. For those works closer to the approach developed in the thesis, a deeper analysis is provided. The main conclusion obtained in this first part of the chapter was the necessity of a formulation that overcame some of the limitations of the current state of the art for the decisions involved at intermediate points of the flight route. In this respect, we introduce in the chapter two new and equivalent integer programming formulations for the ATFM problem. Considering the route of a flight as a sequence of arcs to fly, our formulations are based on incorporating into the decision variables definition, not only information about the arc employed, but also about the time leaving and reaching the corresponding tail and head nodes. On top of that, in the chapter we also present a novel procedure based on a graph of conflicts to detect, beforehand, routes that will be part of the optimal solution. As conclusions of these two methodological contributions, in the chapter we show how the integer programming models proposed provide:

i) A more realistic modelization for the decisions involved at intermediate points of the flight routes.

ii) The possibility of including non-linear costs while keeping the model linear.

iii) An easy way to model dynamic air sector configurations.

iv) Formulations where most of the constraints define facets of the polytope. Actually, the second formulation proposed is indeed a shortest path problem in multiple networks with limited shared resources. A fact that enlarges the range of solving strategies for the problem.

Respect to the graph of conflicts, despite being an interesting idea, it did not provide any computational time advantage when using it. Thus, its validity in this sense is limited. Nevertheless, as described in the future lines of research, a better coding of the algorithm employed to build and explore the graph could revert the situation.

Chapter 3 contains the following contributions: 1) Introducing a family of shortest path problems that, to the best of our knowledge, has not been previously studied in optimization literature: the Shared Resource Constrained Multi-Shortest Path Problem (SRC-MSPP). 2) Extending some of the results in Chapter 2 by demonstrating how the SRC-MSPP can be used to solve, in addition to ATFM problems, some types of project scheduling problems. And 3) Studying two solution methods for the SRC-MSPP, one based on matheuristics, and the other based on Lagrangian relaxations. Respect to the latter, two different relaxations are studied, one resulting from dualizing the capacity constraints in the SRC-MSPP integer programming model (3.5)-(3.8), and the other resulting from dualizing a copy of the decisions variables. The conclusions obtained in this chapter are:

i) The SRC-MSPP, as corroborated by the computational experience, has strong LP relaxation.

ii) Despite similarities with the Resource Constrained Shortest Path Problem, the SRC-MSPP has important particularities, e. g., the structure of the solutions.

iii) The SRC-MSPP permits to incorporate multiple features at the same (e. g., execution modes or alternative sequences of activities) when modeling project scheduling problems.

iv) The matheuristic is designed to take advantage of multi-core computers, and to exploit the structure of the problem where the least expensive arcs are also those that consume the most limited resources at a higher rate.

v) Both Lagrangian relaxations presented have interesting subproblem structures and theoretical properties.

Chapter 4 includes as main contributions: 1) The generation (and released for free disposal) of ATFM instances, 2) The empirical validation of the results obtained in the previous chapters of the thesis, and 3) Insights on how and why the proposed solution methods worked the way they did. The conclusions of this chapter are:

i) Commercial MIP solvers are capable of tackling instances of the ATFM problem up to a certain size in acceptable computational times.

ii) The matheuristic proposed is shown to: a) Be capable of solving all the instances proposed (regardless of size and difficulty) in times that are feasible for the ATFM problem, b) Obtain high-quality solutions for the pool sizes employed, c) Outperform in time to exact methods in most of the occasions, d) Exhibit a robust performance under different cost structures, e) Largely benefit from the parallel design, and f) Be able to improve the quality of any solution by simply using a larger pool size.

iii) The Lagrangian relaxations proposed require further research. The relaxation based on dualizing the capacity constraints of the SRC-MSPP, although useful to obtain lower bounds with which to compare the proposed matheuristic, showed some limitations that make it not suitable, for the moment, to become a competitive solution method for the problem. For the relaxation based on dualizing a copy of the decision variables, no conclusions were obtained because computational results are part of the future lines of research.

iv) The flight plan modifications obtained with the IP formulations proposed are coherent with practical ATFM operations.

After summarizing the conclusions and contributions made in the thesis, we identify the following topics for future research:

1. Further exploring the idea of the graph of conflicts. This includes two keypoints. The first one is re-coding the algorithm that builds and explores the graph so these operations can be done faster. In order words, that the time trade-off when using the graph of conflicts becomes beneficial. The second point is obtaining some other information of the graph such as its connectivity. This is oriented to find clusters of nodes connected to each other just by a few arcs. Thus, removing those few arcs (i. e., relaxing the associated capacity constraints in the problem), would lead to independent components (i. e., independent subproblems to be solved).

2. Keep testing the matheuristic proposed. Although the computational experience showed a good performance of the algorithm for different sizes and difficulties, conducting more tests will help to obtain more reliable conclusions. This line of research includes testing the algorithm with: i) The instances of Dal Sasso et al. [34], and ii) Other types of project scheduling problems where the algorithm can be applied.

3. Extending the results in Chapter 3 to tackle project scheduling problems with a general structure of predecessor and successor activities. As pointed out in the chapter, the SRC-MSPP can incorporate multiple features when dealing when project scheduling problems, but it is limited to projects with serial sequences of activities. Thus, to enlarge the range of applicability while maintaining the network structure, we think that the solution passes through working with generalized networks and minimum cost flow problems, instead of shortest paths. That way, most of the algorithm proposed would still be valid to solve other types of project scheduling problems.

4. Continue studying the Lagrangian relaxation as an independent solution method for the problem and not just to obtain lower bounds. This implies, on the one hand, obtaining computational results for the relaxation based on dualizing a copy of the decision variables (recall in Subsection 3.3.3.2). And, on the other hand, fixing the problems described in Subsection 4.2.2.2 for the relaxation resulting from dualizing the capacity constraints. For the latter, the options that we want to explore are:

a) Testing other updating rules for the Lagrangian multipliers. In the computational experience, two methods were tested, both based on obtaining subgradients and employing their squared norm to determine the step size with which to update the multipliers. As the norm values were always too big, these methods exhibited slow convergence rates, leading to long computational times. Thus, for the Lagrangian relaxation to become a feasible solution option, other rules that do not exhibit this problem have to be tested.

b) Developing a heuristic algorithm based on repairing infeasible solutions. To guarantee that when solving the Lagrangian relaxation we finish with a feasible solution for the original problem, we should be able to employ the (expected) infeasible solutions obtained at each step of the relaxation and transform them into feasible ones.

c) Incorporating to the relaxation the idea of the graph of conflicts. Particularly, that described before about finding clusters of nodes connected to each other just by a few arcs. That way, instead of relaxing all the capacity constraints, we could relax small subsets based on this information. This would make it more likely to obtain feasible solutions for the original problem.

5. Exploring the stochastic version of the SRC-MSPP. That is, the version in which different sources of uncertainty are considered. Respect to this line of research, our plans include:

   a) Incorporating uncertainty into the capacity of the resources and, when modeling scheduling problems such as ATFM, into the execution time of the activities. Notice the importance of these two facts in order to obtain more robust and realistic schedules.

      Our primary intention is to tackle the stochastic version of the problem via scenario trees. This motivates the following point.

   b) Extending the matheuristic proposed (or studying new algorithms) for the stochastic version of the problem. Concretely, we intend to explore its integration with the branch-and-fix coordination algorithm proposed in Alonso-Ayuso, Escudero, and Ortuño [9].

   c) Studying the behavior of the problem under different risk measures, e. g., Shortfall Probability, Value-at-risk, etc.

BIBLIOGRAPHY

[1] NR Achuthan and A Hardjawidjaja. "Project scheduling under time dependent costs–A branch and bound algorithm." In: *Annals of Operations Research* 108.1-4 (2001), pp. 55–74.

[2] Alba Agustín. "Mathematical optimization in air traffic flow management under uncertainty." PhD thesis. Department of Statistics and Operations Research, Universidad Rey Juan Carlos, Spain, 2011.

[3] Alba Agustín, Antonio Alonso-Ayuso, Laureano F Escudero, and Celeste Pizarro. "Mathematical optimization models for air traffic flow management: A review." In: *Studia Informatica Universalis* 8.2 (2010), pp. 141–184.

[4] Alba Agustín, Antonio Alonso-Ayuso, Laureano F Escudero, and Celeste Pizarro. "On air traffic flow management with rerouting. Part I: Deterministic case." In: *European Journal of Operational Research* 219.1 (2012), pp. 156–166.

[5] Alba Agustín, Antonio Alonso-Ayuso, Laureano F Escudero, Celeste Pizarro, et al. "On air traffic flow management with rerouting. Part II: Stochastic case." In: *European Journal of Operational Research* 219.1 (2012), pp. 167–177.

[6] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows: theory, algorithms, and applications*. English. Upper Saddle River (New Jersey): Prentice Hall, 1993. ISBN: 9780136175490;013617549X;

[7] Ravindra K Ahuja, James B Orlin, Stefano Pallottino, and Maria G Scutella. "Dynamic shortest paths minimizing travel times and costs." In: *Networks: An International Journal* 41.4 (2003), pp. 197–205.

[8] Ali Akgunduz, Brigitte Jaumard, and Golbarg Moeini. "Deconflicted Air-Traffic Planning With Speed-Dependent Fuel-Consumption Formulation." In: *IEEE Transactions on Intelligent Transportation Systems* (2017).

[9] Antonio Alonso-Ayuso, Laureano F Escudero, and M Teresa Ortuño. "BFC, a branch-and-fix coordination algorithmic framework for solving some types of stochastic pure and mixed 0–1 programs." In: *European Journal of Operational Research* 151.3 (2003), pp. 503–519.

[10] Yash P Aneja and Kunhiraman PK Nair. "Bicriteria transportation problem." In: *Management Science* 25.1 (1979), pp. 73–78.

[11] Hamsa Balakrishnan and Bala G Chandran. "Optimal large-scale air traffic flow management." In: *unpublished* (2014).

[12] Marc Baumgartner. "The organisation and operation of European airspace." In: *European Air Traffic Management: principles, practice and research*. Ed. by Adrew Cok. Ashgate Publishing, 2007, pp. 1–34.

[13] John E Beasley and Nicos Christofides. "An algorithm for the resource constrained shortest path problem." In: *Networks* 19.4 (1989), pp. 379–394.

[14] Richard Bellman. *Dynamic Programming*. 1st ed. Princeton, NJ, USA: Princeton University Press, 1957.

[15] Richard Bellman. "On a routing problem." In: *Quarterly of applied mathematics* 16.1 (1958), pp. 87–90.

[16] Dimitris Bertsimas and Sarah Stock Patterson. "The air traffic flow management problem with enroute capacities." In: *Operations research* 46.3 (1998), pp. 406–422.

[17] Dimitris Bertsimas and Sarah Stock Patterson. "The traffic flow management rerouting problem in air traffic control: A dynamic network flow approach." In: *Transportation Science* 34.3 (2000), pp. 239–255.

[18] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*. Vol. 6. Athena Scientific Belmont, MA, 1997.

[19] Dimitris Bertsimas and Robert Weismantel. *Optimization over integers*. Vol. 13. Dynamic Ideas Belmont, 2005.

[20] Dimitris Bertsimas, Guglielmo Lulli, and Amedeo Odoni. "An integer optimization approach to large-scale air traffic flow management." In: *Operations research* 59.1 (2011), pp. 211–227.

[21] Jean-Charles Billaut, Federico Della Croce, and Andrea Grosso. "A single machine scheduling problem with two-dimensional vector packing constraints." In: *European Journal of Operational Research* 243.1 (2015), pp. 75–81.

[22] Natashia L Boland and Martin WP Savelsbergh. "Perspectives on integer programming for time-dependent models." In: *Top* 27.2 (2019), pp. 147–173.

[23] Mikhail A Bragin, Peter B Luh, Joseph H Yan, Nanpeng Yu, and Gary A Stern. "Convergence of the surrogate Lagrangian relaxation method." In: *Journal of Optimization Theory and applications* 164.1 (2015), pp. 173–201.

[24] Pasquale Carotenuto, Stefano Giordani, and Salvatore Ricciardelli. "Finding minimum and equitable risk routes for hazmat shipments." In: *Computers & Operations Research* 34.5 (2007), pp. 1304–1327.

[25] Ismail Chabini. "Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time." In: *Transportation research record* 1645.1 (1998), pp. 170–175.

[26] Alain Chabrier. "Vehicle routing problem with elementary shortest path based column generation." In: *Computers & Operations Research* 33.10 (2006), pp. 2972–2990.

[27] Yu-Heng Chang, Senay Solak, John-Paul B Clarke, and Ellis L Johnson. "Models for single-sector stochastic air traffic flow management under reduced airspace capacity." In: *Journal of the Operational Research Society* 67.1 (2016), pp. 54–67.

[28] Athanasios P Chassiakos and Serafim P Sakellaropoulos. "Time-cost optimization of construction projects with generalized activity constraints." In: *Journal of Construction Engineering and Management* 131.10 (2005), pp. 1115–1124.

[29] Dan Chen, Minghua Hu, Honghai Zhang, Jianan Yin, and Ke Han. "A network based dynamic air traffic flow model for en route airspace system traffic flow optimization." In: *Transportation Research Part E: Logistics and Transportation Review* 106 (2017), pp. 1–19.

[30] Jing Chen, Long Chen, and D Sun. "Air traffic flow management under uncertainty using chance-constrained optimization." In: *Transportation Research Part B: Methodological* 102 (2017), pp. 124–141.

[31] Jun Chen, Yi Cao, and Dengfeng Sun. "Modeling, Optimization, and Operation of Large-Scale Air Traffic Flow Management on Spark." In: *Journal of Aerospace Information Systems* (2017), pp. 1–13.

[32] Giuseppe Confessore, Stefano Giordani, and Silvia Rismondo. "A market-based multi-agent system model for decentralized multi-project scheduling." In: *Annals of Operations Research* 150.1 (2007), pp. 115–135.

[33] Luca Corolli, Guglielmo Lulli, Lewis Ntaimo, and Saravanan Venkatachalam. "A two-stage stochastic integer programming model for air traffic flow management." In: *IMA Journal of Management Mathematics* 28.1 (2017), pp. 19–40.

[34] Veronica Dal Sasso, Franklin Djeumou Fomeni, Guglielmo Lulli, and Konstantinos G Zografos. "Incorporating Stakeholders' priorities and preferences in 4D trajectory optimization." In: *Transportation Research Part B: Methodological* 117 (2018), pp. 594–609.

[35] Veronica Dal Sasso, Franklin Djeumou Fomeni, Guglielmo Lulli, and Konstantinos G Zografos. "Planning efficient 4D trajectories in Air Traffic Flow Management." In: *European Journal of Operational Research* 276.2 (2019), pp. 676–687.

[36] Emilie Danna, Edward Rothberg, and Claude Le Pape. "Exploring relaxation induced neighborhoods to improve MIP solutions." In: *Mathematical Programming* 102.1 (2005), pp. 71–90.

[37] Federico Della Croce, Fabio Salassa, and Vincent T'kindt. "A hybrid heuristic approach for single machine scheduling with release times." In: *Computers & Operations Research* 45 (2014), pp. 7–11.

[38] Xudong Diao and Chun-Hsien Chen. "A sequence model for air traffic flow management rerouting problem." In: *Transportation Research Part E: Logistics and Transportation Review* 110 (2018), pp. 15–30.

[39] Edsger W Dijkstra. "A note on two problems in connexion with graphs." In: *Numerische mathematik* 1.1 (1959), pp. 269–271.

[40] Marco Dorigo. "Optimization, learning and natural algorithms." In: *PhD Thesis, Politecnico di Milano* (1992).

[41] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. "Ant system: optimization by a colony of cooperating agents." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1 (1996), pp. 29–41.

[42] Andreas Drexl, Ruediger Nissen, James H Patterson, and Frank Salewski. "Progen/πx–An instance generator for resource-constrained project scheduling problems with partially renewable resources and further extensions." In: *European Journal of Operational Research* 125.1 (2000), pp. 59–72.

[43] Irina Dumitrescu and Natashia Boland. "Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem." In: *Networks: An International Journal* 42.3 (2003), pp. 135–153.

[44] EUROCONTROL. *2017 Comparison of ATM-related performance: U.S. – Europe*. Accessed on December 17, 2020. 2017. URL: https://www.eurocontrol.int/sites/default/files/2019-05/us-europe-comparison-operational-performance-2017.pdf.

[45] EUROCONTROL. *Challenges of growth 2018*. Accessed on December 17, 2020. 2018. URL: https://www.eurocontrol.int/sites/default/files/content/documents/official-documents/reports/challenges-of-growth-2018.pdf.

[46] EUROCONTROL. *Annual Network Operations Report 2018*. Accessed on December 17, 2020. 2019. URL: https://www.eurocontrol.int/sites/default/files/2019-11/nm-annual-network-operations-report-2018-main-report.pdf.

[47] EUROCONTROL. *Performance Review Report: An Assessment of Air Traffic Management in Europe during the Calendar Year 2018*. Accessed on December 17, 2020. 2019. URL: https://www.eurocontrol.int/sites/default/files/2019-06/prr-2018.pdf.

[48] David Eppstein. "Finding the k shortest paths." In: *SIAM Journal on computing* 28.2 (1998), pp. 652–673.

[49] Luis Fanjul-Peyro, Federico Perea, and Rubén Ruiz. "Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources." In: *European Journal of Operational Research* 260.2 (2017), pp. 482–493.

[50] Martina Fischetti and Matteo Fischetti. "Matheuristics." In: *Handbook of Heuristics*. Ed. by Rafael Martí, Panos M. Pardalos, and Mauricio G. C. Resende. Cham: Springer International Publishing, 2018, pp. 121–153. ISBN: 978-3-319-07124-4. DOI: 10.1007/978-3-319-07124-4_14. URL: https://doi.org/10.1007/978-3-319-07124-4_14.

[51] Matteo Fischetti and Andrea Lodi. "Local branching." In: *Mathematical programming* 98.1-3 (2003), pp. 23–47.

[52] Marshall L Fisher. "The Lagrangian relaxation method for solving integer programming problems." In: *Management science* 27.1 (1981), pp. 1–18.

[53] David García-Heredia, Antonio Alonso-Ayuso, and Elisenda Molina. "A Combinatorial model to optimize air traffic flow management problems." In: *Computers & Operations Research* 112 (2019), p. 104768.

[54] David García-Heredia, Elisenda Molina, Manuel Laguna, and Antonio Alonso-Ayuso. *A solution method for the shared Resource Constrained Multi-Shortest Path Problem*. 2020. URL: https://e-archivo.uc3m.es/handle/10016/30793.

[55] Carlos García-Martínez, Francisco J. Rodriguez, and Manuel Lozano. "Genetic Algorithms." In: *Handbook of Heuristics*. Ed. by Rafael Martí, Panos M. Pardalos, and Mauricio G. C. Resende. Cham: Springer International Publishing, 2018, pp. 431–464. ISBN: 978-3-319-07124-4. DOI: 10.1007/978-3-319-07124-4_28. URL: https://doi.org/10.1007/978-3-319-07124-4_28.

[56] Renan Garcia. "Resource constrained shortest paths and extensions." PhD thesis. Georgia Institute of Technology, 2009.

[57] Stuart Geman and Donald Geman. "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images." In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1984), pp. 721–741.

[58] Arthur M Geoffrion. "Lagrangean relaxation for integer programming." In: *Approaches to integer programming*. Springer, 1974, pp. 82–114.

[59] Fred W Glover and Gary A Kochenberger. *Handbook of metaheuristics*. Vol. 57. Springer Science & Business Media, 2006.

[60] Fred W Glover and Manuel Laguna. *Tabu Search*. Springer Science & Business Media, 1998.

[61] Fred Glover. "Heuristics for integer programming using surrogate constraints." In: *Decision sciences* 8.1 (1977), pp. 156–166.

[62] Fred Glover. "Future paths for integer programming and links to ar tifi cial intelli g en ce." In: *Computers operations research* 13.5 (1986), pp. 533–549.

[63] Fred Glover. "Tabu search—part I." In: *ORSA Journal on computing* 1.3 (1989), pp. 190–206.

[64] Fred Glover. "Tabu search—part II." In: *ORSA Journal on computing* 2.1 (1990), pp. 4–32.

[65] Fred Glover. "A template for scatter search and path relinking." In: *Lecture notes in computer science* 1363 (1998), pp. 13–54.

[66] José Fernando Gonçalves, Jorge JM Mendes, and Maurício GC Resende. "A genetic algorithm for the resource constrained multi-project scheduling problem." In: *European Journal of Operational Research* 189.3 (2008), pp. 1171–1190.

[67] F Guerriero and R Musmanno. "Label correcting methods to solve multicriteria shortest path problems." In: *Journal of optimization theory and applications* 111.3 (2001), pp. 589–613.

[68] Monique Guignard. "Lagrangean relaxation." In: *Top* 11.2 (2003), pp. 151–200.

[69] Monique Guignard and Siwhan Kim. "Lagrangean decomposition: A model yielding stronger Lagrangean bounds." In: *Mathematical programming* 39.2 (1987), pp. 215–228.

[70] LLC Gurobi Optimization. *Gurobi Optimizer Reference Manual*. Accessed on December 17, 2020. 2020. URL: http://www.gurobi.com.

[71] Bruce Hajek. "Cooling schedules for optimal annealing." In: *Mathematics of operations research* 13.2 (1988), pp. 311–329.

[72] Gabriel Y Handler and Israel Zang. "A dual algorithm for the constrained shortest path problem." In: *Networks* 10.4 (1980), pp. 293–309.

[73] Pierre Hansen and Nenad Mladenović. "Variable Neighborhood Search." In: *Handbook of Heuristics*. Ed. by Rafael Martí, Panos M. Pardalos, and Mauricio G. C. Resende. Cham: Springer International Publishing, 2018, pp. 759–787. ISBN: 978-3-319-07124-4. DOI: 10.1007/978-3-319-07124-4_19. URL: https://doi.org/10.1007/978-3-319-07124-4_19.

[74] Sönke Hartmann and Dirk Briskorn. "A survey of variants and extensions of the resource-constrained project scheduling problem." In: *European Journal of operational research* 207.1 (2010), pp. 1–14.

[75] John Henry Holland et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[76] Markó Horváth and Tamás Kis. "Solving resource constrained shortest path problems with LP-based methods." In: *Computers & Operations Research* 73 (2016), pp. 150–164.

[77] Frank Hutter, Holger H Hoos, and Thomas Stützle. "Automatic algorithm configuration based on local search." In: *Aaai*. Vol. 7. 2007, pp. 1152–1157.

[78] IATA. *IATA ECONOMIC BRIEFING*. Accessed on December 17, 2020. 2013. URL: https://www.iata.org/en/iata-repository/publications/economic-reports/inefficiency-in-european-airspace/.

[79] IATA. *Annual Review 2016*. Accessed on December 17, 2020. 2016. URL: https://www.iata.org/about/Documents/iata-annual-review-2016.pdf.

[80] IATA. *Tackling the European infrastructure crisis*. Accessed on December 17, 2020. 2016. URL: http://airlines.iata.org/analysis/tackling-the-european-infrastructure-crisis.

[81] Nikola Ivanov, Fedja Netjasov, Radosav Jovanović, Stefano Starita, and Arne Strauss. "Air Traffic Flow Management slot allocation to minimize propagated delay and improve airport slot adherence." In: *Transportation Research Part A: Policy and Practice* 95 (2017), pp. 183–197.

[82] Sai Prashanth Josyula, Johanna Törnquist Krasemann, and Lars Lundberg. "A parallel algorithm for train rescheduling." In: *Transportation Research Part C: Emerging Technologies* 95 (2018), pp. 545 –569. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2018.07.003. URL: http://www.sciencedirect.com/science/article/pii/S0968090X18309410.

[83] Radosav Jovanović, Vojin Tošić, Mirjana Čangalović, and Milan Stanojević. "Anticipatory modulation of air navigation charges to balance the use of airspace network capacities." In: *Transportation Research Part A: Policy and Practice* 61 (2014), pp. 84–99.

[84] Carolin Kellenbrink and Stefan Helber. "Scheduling resource-constrained projects with a flexible project structure." In: *European Journal of Operational Research* 246.2 (2015), pp. 379–391.

[85] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. "Optimization by simulated annealing." In: *science* 220.4598 (1983), pp. 671–680.

[86] Robert Klein. "Project scheduling with time-varying resource constraints." In: *International Journal of Production Research* 38.16 (2000), pp. 3937–3952.

[87] Robert Klein and Armin Scholl. "Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling." In: *European Journal of Operational Research* 112.2 (1999), pp. 322–346.

[88] Rainer Kolisch and Sönke Hartmann. "Experimental investigation of heuristics for resource-constrained project scheduling: An update." In: *European journal of operational research* 174.1 (2006), pp. 23–37.

[89] Rainer Kolisch and Rema Padman. "An integrated survey of deterministic project scheduling." In: *Omega* 29.3 (2001), pp. 249–272.

[90] Doreen Krüger and Armin Scholl. "A heuristic solution framework for the resource constrained (multi-) project scheduling problem with sequence-dependent transfer times." In: *European Journal of Operational Research* 197.2 (2009), pp. 492–508.

[91] Jürgen Kuster, Dietmar Jannach, and Gerhard Friedrich. "Extending the RCPSP for modeling and solving disruption management problems." In: *Applied Intelligence* 31.3 (2009), p. 234.

[92] Manuel Laguna and Rafael Martí. *Scatter search: methodology and implementations in C*. Vol. 24. Springer Science & Business Media, 2012.

[93] LocalSolver. *LocalSolver Reference Manual*. Accessed on December 17, 2020. 2020. URL: https://www.localsolver.com/.

[94] Manuel López-Ibáñez, Thomas Stützle, and Marco Dorigo. "Ant Colony Optimization: A Component-Wise Overview." In: *Handbook of Heuristics*. Ed. by Rafael Martí, Panos M. Pardalos, and Mauricio G. C. Resende. Cham: Springer International Publishing, 2018, pp. 371–407. ISBN: 978-3-319-07124-4. DOI: 10.1007/978-3-319-07124-4_21. URL: https://doi.org/10.1007/978-3-319-07124-4_21.

[95] Leonardo Lozano and Andrés L Medaglia. "On an exact method for the constrained shortest path problem." In: *Computers & Operations Research* 40.1 (2013), pp. 378–384.

[96] Zhiqiang Lu, Yifei Ren, Lin Wang, and Hongwei Zhu. "A Resource Investment Problem based on Project Splitting with Time Windows for Aircraft Moving Assembly Line." In: *Computers & Industrial Engineering* (2019).

[97] Guglielmo Lulli and Amedeo Odoni. "The European air traffic flow management problem." In: *Transportation Science* 41.4 (2007), pp. 431–443.

[98] Rafael Martí, Panos M. Pardalos, and Mauricio G.C. Resende. *Handbook of heuristics*. Vol. 57. Springer, 2018.

[99] Nenad Mladenović and Pierre Hansen. "Variable neighborhood search." In: *Computers & operations research* 24.11 (1997), pp. 1097–1100.

[100] Alejandro Montoya, Christelle Guéret, Jorge E Mendoza, and Juan G Villegas. "A multi-space sampling heuristic for the green vehicle routing problem." In: *Transportation Research Part C: Emerging Technologies* 70 (2016), pp. 113–128.

[101] Alexander G Nikolaev and Sheldon H Jacobson. "Simulated annealing." In: *Handbook of metaheuristics*. Springer, 2010, pp. 1–39.

[102] OpenMP Architecture Review Board. *OpenMP Application Program Interface Version 4.5*. Accessed on December 17, 2020. 2015. URL: https://www.openmp.org/wp-content/uploads/openmp-4.5.pdf.

[103] Metin Ozgur and Aydan Cavcar. "0–1 integer programming model for procedural separation of aircraft by ground holding in ATFM." In: *Aerospace science and technology* 33.1 (2014), pp. 1–8.

[104] Michal Pióro, Áron Szentesi, János Harmatos, Alpár Jüttner, Piotr Gajowniczek, and Stanislaw Kozdrowski. "On open shortest path first related network optimisation problems." In: *Performance evaluation* 48.1-4 (2002), pp. 201–223.

[105] Andrea Raith and Matthias Ehrgott. "A comparison of solution strategies for biobjective shortest path problems." In: *Computers & Operations Research* 36.4 (2009), pp. 1299–1331.

[106] Rob Shone, Kevin Glazebrook, and Konstantinos G. Zografos. "Applications of stochastic modeling in air traffic management: Methods, challenges and opportunities for solving air traffic problems under uncertainty." In: *European Journal of Operational Research* (2020). ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2020.10.039. URL: http://www.sciencedirect.com/science/article/pii/S0377221720309164.

[107] Kenneth Sörensen and Fred W. Glover. "Metaheuristics." In: *Encyclopedia of Operations Research and Management Science*. Ed. by Saul I. Gass and Michael C. Fu. Boston, MA: Springer US, 2013, pp. 960–970. ISBN: 978-1-4419-1153-7. DOI: 10.1007/978-1-4419-1153-7_1167. URL: https://doi.org/10.1007/978-1-4419-1153-7_1167.

[108] Túlio AM Toffolo, Haroldo G Santos, Marco AM Carvalho, and Janniele A Soares. "An integer programming approach to the multimode resource-constrained multiproject scheduling problem." In: *Journal of Scheduling* 19.3 (2016), pp. 295–307.

[109] Johanna Törnquist. "Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms." In: *5th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS'05)*. Ed. by Leo G. Kroon and Rolf H. Möhring. Vol. 2. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik,

2006. ISBN: 978-3-939897-00-2. DOI: 10.4230/OASIcs.ATMOS.2005.659. URL: http://drops.dagstuhl.de/opus/volltexte/2006/659.

[110] Johanna Törnquist. "Design of an effective algorithm for fast response to the re-scheduling of railway traffic during disturbances." In: *Transportation Research Part C: Emerging Technologies* 20.1 (2012). Special issue on Optimization in Public Transport+ISTT2011, pp. 62 –78. ISSN: 0968-090X.

[111] US Department of Transportation Bureau of Transportation Statistics. *Airline On-Time Performance Data*. Accessed on December 17, 2020. URL: https://www.transtats.bts.gov/tables.asp?db_id=120&DB_Name=.

[112] US Department of Transportation Bureau of Transportation Statistics. *Aviation Support Tables*. Accessed on December 17, 2020. URL: https://www.transtats.bts.gov/tables.asp?DB_ID=595&DB_Name=&DB_Short_Name=.

[113] P Wei, Y Cao, and D Sun. "Total unimodularity and decomposition method for large-scale air traffic cell transmission model." In: *Transportation Research Part B: Methodological* 53 (2013), pp. 1–16.

[114] David H Wolpert and William G Macready. "No free lunch theorems for optimization." In: *IEEE transactions on evolutionary computation* 1.1 (1997), pp. 67–82.

[115] Mingming Xiao, Kaiquan Cai, and Hussein A Abbass. "Hybridized encoding for evolutionary multi-objective optimization of air traffic network flow: A case study on China." In: *Transportation Research Part E: Logistics and Transportation Review* 115 (2018), pp. 35–55.

[116] Jin Y Yen. "Finding the k shortest loopless paths in a network." In: *management Science* 17.11 (1971), pp. 712–716.

[117] Xing Zhao, Peter B Luh, and Jihua Wang. "Surrogate gradient algorithm for Lagrangian relaxation." In: *Journal of optimization Theory and Applications* 100.3 (1999), pp. 699–712.

[118] Xiao-long Zheng and Ling Wang. "A multi-agent optimization algorithm for resource constrained project scheduling problem." In: *Expert Systems with Applications* 42.15-16 (2015), pp. 6039–6049.