DISRUPTION RECOVERY IN COMMERCIAL AVIATION

Francisco de Lemos

First Supervisor: Dr. John Woodward
Second Supervisor: Dr. Jun Chen
Independent Assessor: Dr. Alexander Brownlee

School of Electronic Engineering and Computer Science

This dissertation is submitted for the degree of
Doctor of Philosophy

September 2022

In loving memory of my mother Graciete Martins Rodrigues
I will hold you in my heart until I can hold you again in heaven

Declaration

Francisco de Lemos
September 2022

Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council under the OR-Master project. I want to thank my Supervisory Team and Independent Assessor. First Supervisor Dr. John Woodward, for his scientific advice, extended support, and encouragement during the PhD. Second Supervisor Dr. Jun Chen for the support, the long, very productive brainstorms and guidance. Independent Assessor Dr. Alexander Brownlee for his remarks during the appraisals.

The first contribution included in this thesis could not have been achieved without the assistance, and support of professionally experienced individuals. On this note, I would like to extend my gratitude to Luís Costa Ramos (Aeronautical Engineer Technician), Ricardo Pato (First Officer at Transportes Aéreos Portugueses (Portuguese Air Transportations) (TAP)), António Francisco Caixeiro (Colonel at the Portuguese Air Force) and João Ferreira (Flight Commander at TAP). Their technical guidance helped me through extremely difficult times over the course of the analysis and the writing of this thesis and for that I sincerely thank them for their confidence in me.

I would also like to thank my colleague Dr. Mosab Bazargany for his technical help and scientific insight. Finally, I would like to express my gratitude to my longtime friend Dr. Pedro Machado, for his encouragement and outstanding review of my work.

Abstract

This thesis presents three major contributions for commercial aviation planning and disruption recovery in commercial aviation. The first contribution presented in this thesis consists of a flight planning model to calculate Block Time and Fuel (BTF) consumed for an aircraft model during the flight. The BTF model computes the ground distance between the origin and destination airports, derives the flight's cruise altitude, and by integrating two institutional data sets calculates the duration and the fuel consumed for the whole of taxi-out, take-off, climb, cruise, descent, approach, landing, and taxi-in phases. The model renders very good results for block time and consumed fuel however, it does not consider aircraft weight loss neither the influence of the wind. The second contribution of this thesis consists of a recovery procedure for disrupted aircraft rotations, the Constructive Heuristic for the Aircraft Recovery Problem (CHARP). The CHARP recovers the infeasible rotation combining a meta-heuristic that performs a pincer movement over the search space and Constraint Programming (CP). Additionally, the CHARP uses Constraint Propagation to reduce the size of the search therefore reducing computing. The initial experiments demonstrated that if Constraint Propagation was not used computing time would double. The recovery strategy included flight creation delays and cancellations however it did not include aircraft swap. The third contribution of this thesis combines the BTF model and the CHARP. Since the BTF model returns lower block time flights than those used by the CHARP this thesis investigates six disruption scenarios with shorter block time.

Acronyms

AAP Aircraft Assignment Problem. 87

ABC Artificial Bee Colony. 88

AC Arc-consistency. 63

ACO Ant Colony Optimisation. 87, 91

AEV Adaptive Evaluated Vector. 87

AIR DIST Air Distance. 49

ARP Aircraft Recovery Problem. 7–9, 12, 26, 31, 54–56, 67, 76, 78, 85, 87–90, 94, 97, 124

ASK Available Seat Kilometres. 4

ATC Air Traffic Control. 34

ATM Air Traffic Management. 12, 18, 123

BaB Branch and Bound. 7, 21, 91

BADA Base of Aircraft Data. 11, 12, 16–19, 31, 33, 35, 38, 43, 47, 54, 123, 124

BCN Barcelona Josep Tarradellas Airport. xiii, 41, 43–45

BD Benders Decomposition. 7

BTA Backtracking Algorithm. 63, 67, 74

BTF Block Time and Fuel. xiii, xv, xix, 8, 9, 33, 35–43, 46–54, 97–99, 122–125, 144

BTS Bureau of Transportation Statistics. 4, 6

CCD Climb, Cruise and Descent. 19, 30, 34, 54, 123

CDG Paris Charles de Gaulle Airport. xiii, 41, 43–45, 58–60

CHARP Constructive Heuristic for the Aircraft Recovery Problem. 8, 9, 55, 63, 64, 66, 70, 76, 78, 81, 85, 89, 90, 93–95, 97, 98, 111, 122, 124, 125

CI Cost Index. 2, 7, 30, 122

CNS/ATM Communication Navigation and Surveillance/Air Traffic Management. 17

CP Constraint Programming. ix, 8, 11, 19, 20, 26, 31, 55, 62, 69, 88–90, 93, 94, 97, 123, 124

CPr Constraint Propagation. 11, 31, 55, 63, 69, 78, 90, 97, 124

CPrA Constraint Propagation Algorithm. 69

CRP Crew Recovery Problem. 8

CSA Cuckoo Search Algorithm. 15, 48

CSP Constraint Satisfaction Problem. 7, 11, 55, 62, 63, 67, 124

DFS Depth First Search. 91

DM Disruption Management. 4

DMSM Direct Multiple Shooting Method. 12

DP Dynamic Programming. 13, 30

DSS Disruption Set Solver. 87, 91

ECON Economic Speed. 2

EEA European Environment Agency. 19, 33, 35, 37, 38, 41, 54, 123

ELDT Estimated Landing Time. 49

EMEP European Monitoring and Evaluation Programme. 19, 33, 35, 37, 38, 41, 54, 123

EPD Expected Propagated Delay. 7

ETA Estimated Time of Arrival. 49

ETD Estimated Time of Departure. 49

EU European Union. 37

FAP Fleet Assignment Problem. 1

FDA Flight Domain Algorithm. 66, 67

FDRs Flight Data Records. 15

FL Flight Level. 2, 16, 34, 37, 39–43, 47, 54

FOB Fuel On Board. 33

FVA Flight Verification Algorithm. 66

GA Genetic Algorithm. 13, 87

GaT Generate and Test. 62
GMT Greenwich Mean Time. 57, 59
GPS Global Positioning System. 30
GRASP Greedy Randomised Adaptive Search Procedure. 85, 87
GSA Greedy Simulated Annealing. 88

i4D Initial 4D. 18
IATA International Air Transport Association. 48, 49, 57, 58
ICAO International Civil Aviation Organization. xiii, 17, 36–38, 49

KPI Key Performance Indicators. 4, 6
KSC Kosice Airport. 18

LHA Lower Heuristic Algorithm. 63, 70, 72, 78, 81, 89
LHR London Heathrow Airport. 59–61
LNS Large Neighbourhood Search. 26, 28, 88
LOF Lines od Flights. 88
LP Linear Programming. 22, 88
LR Lagrangian Relaxation. 7, 22, 23
LW Landing Weight. 49

MILP Mixed Integer Linear Programming. 88
MOTO Multi-Objective Trajectory. 17
MRC Maximum Range Cruise. 2

NCE Nice Côte d'Azur Airport. 57–60
NFA New Flights Algorithm. 65
NP-Complete Nondeterministic Polynomial-Time Complete. 22

OR Operations Research. 19
ORY Orly Airport. 59

PBS Performance Based Navigation. 15
PD Propagated Delay. 4, 7
PLN Planned. 49
PMH Pincer Meta-heuristic. 89, 94, 97
PRP Passenger Recovery Problem. 8
PSO Particle Sworm Optimisation. 14
PTF Performance Table Files. 38, 40, 41, 43, 54

RFTO Random Flying Time. 88
RMSE Root Mean Square Error. xiii, 47, 52, 123, 136–138
ROADEF French Association for Operational Research and Decision Support. xiii, xvi, 33, 41, 46, 47, 52, 54, 56, 85, 136–138
ROCD Rate of Climb and Descent. 38, 39, 52, 54, 124
RPK Revenue Passenger Kilometres. 4
RTW Recovery Time Window. 55, 56, 60, 63, 67, 98–100, 106, 111, 117, 122
RVSM Reduced Vertical Separation Minima. 36

SA Simulated Annealing. 26, 85
SARP Stochastic Aircraft Recovery Problem. 26, 87, 88
SDP Soft Dynamic Programming. 13
SIN Singapore Changi Airport. 59, 60

TA Tail Assignment. 20
TAP Transportes Aéreos Portugueses (Portuguese Air Transportations). vii
TAS True Air Speed. 15, 38, 39, 41, 47, 49, 52, 125
TFA Taxi Flights Algorithm. 63
TOW Take-Off Weight. 18, 33, 49
TS Tabu Search. 26

UHA Upper Heuristic Algorithm. 63, 70, 72, 78, 81, 89
US United States. xiii, 3, 15, 16, 18, 29, 36

WTBAM Weighted Time-band Approximation Model. 88

List of figures

List of tables

Table of contents

Chapter 1

Introduction and Overview

1.1 Background on Commercial Aviation Management

Commercial aviation started in the early twenty century and through its course made a remarkable evolution. Aircraft became an efficient and safe means of transportation over long distances, and nowadays the airline industry transports millions of customers around the globe, on business and on vacation. This section provides a high-level overview of the procedures that airlines must implement and the challenges they face to provide their service.

Airline planning is among others one of its most challenging problems. The objective is to determine the optimal flight network that should be operated by the company in a certain time period in the future. The commercial flight must serve with reasonable and sufficient satisfaction the passenger, so that he will want to buy more. The flight planning stage involves researching the average demand between the locations to better manage the flights and achieve profitability for the company.

In addition to forecasting the number of passengers, it is also necessary to determine the fleet that best fits the demand. A fleet is a set of aircraft with different models, each of which suited for specific flight ranges. The next step consists of developing the flight schedules. The latter are mapped out several months in advance by the schedule planning teams. After which aircraft models are assigned for each flight. This procedure is known as the Fleet Assignment Problem (FAP) (Hane et al., 1995), (Bélanger et al., 2006), (Salazar-González, 2014).

When the schedule design and fleet assignment are determined, the flight network is decomposed into sub-networks to each fleet. The next step consists in determining the aircraft rotation. This part of the planning process specifies the route taken by each aircraft, given a set of flights that must be performed with a certain type of fleet, with pre-specified locations, duration and maintenance frequencies (Clarke et al., 1997).

Even though there are considerable number of resources allocated to develop accurate airline planning there are also many factors that can make the airline industry low profitable. Commercial aviation is characterised by high fixed operating and overhead costs. Considering the direct operating cost of a given flight, in most civil aviation flights, three types of costs are present: fuel costs, time-dependent costs, and fixed costs. Time-dependent costs include, among others, maintenance or flight crew related costs. Fixed costs are independent of the time or fuel consumption such as landing fees or aircraft ground handling. Therefore, not only fuel consumption but also time-related costs are considered when airlines try to minimise their total operating cost.

This relation is particularly important during the longest phase of the flight, where the aircraft flies at cruise speed. As shown in figure 1.1, fuel and time-dependent costs vary as a function of the flight cruise speed. Aircraft operators have to trade-off between the amount of fuel consumed and the time needed to fly a certain route.

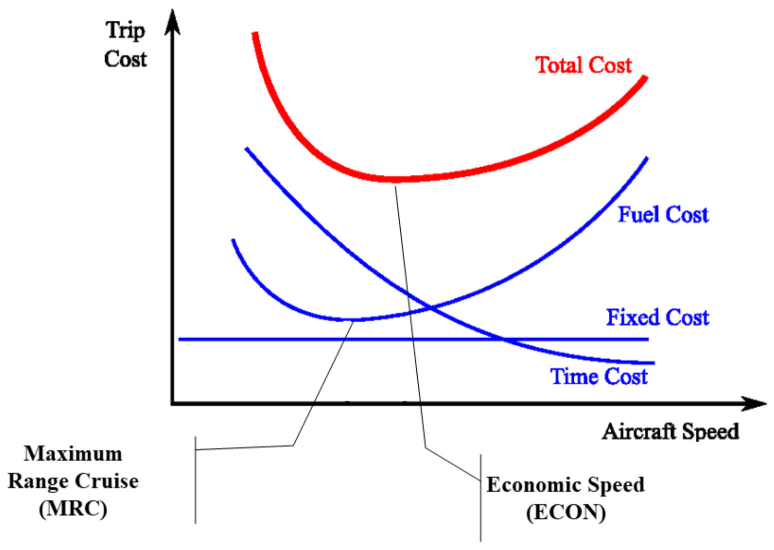


Fig. 1.1 Aircraft operating costs as a function of the cruise speed (source (Delgado and Prats, 2011)).

To comply with the airline policy regarding its operating costs, aircraft equipped with flight management systems use the Cost Index (CI) parameter when optimising their flight trajectories. The CI expresses the ratio between the cost of the flight time and the cost of fuel (Boeing, 2007).

Thus, a CI set to zero means that the cost of the fuel is infinitely higher than the cost of time and the aircraft will fly at the speed which minimises the fuel consumed per unit of distance flown: the Maximum Range Cruise (MRC), figure 1.1. The maximum value of the CI gives priority to the flight time, regardless of the fuel needed. In this case, and in accordance with the manufacturer safety margins, the aircraft will fly at the maximum operating speed. By choosing and introducing the CI in the flight management computer, the pilot is changing the ratio of cost between fuel and time and, therefore, is determining the speed that minimises the total cost. This speed is called the Economic Speed (ECON).

According to figure 1.1 flying below or above ECON will cause an increase of total costs. For low speed the fuel savings will not compensate the inevitable higher time costs and vice versa for higher speed. However, ECON depends significantly of the flight conditions (fuel cost curve is dependent on the gross weight of the aircraft, assigned Flight Level (FL), air temperature as well as wind conditions), hence the relation between speed and operating costs is more complex.

The objective of commercial aviation is to operate flights according to a schedule, in order to maximise their profits. Therefore, airlines generally create tight schedules in order to increase their profitability. These tight schedules, have a minimum slack between flights legs, because airlines rely on the assumption that the flight legs will be operated as planned. However, this process is in many occasions subject to deviations that can complicate the execution of the planned flight schedule.

1.2 Background on Commercial Aviation Disruption

A significant number of factors, such as mechanical failures, crew delays, problems with ground operations, industrial action, inclement weather, congestion at airports, cause disruptions such as flight delays or cancellations. A disruption is a situation that is likely to appear unexpectedly during the time an operation is executed in which its impact is large enough to urge planners to revise the original operation. In other words, disruption makes the initial flight schedule become inefficient in the sense that it cannot deliver a passenger's itinerary. Financial results in the aviation industry have been mixed. According to the (Airlines of America, 2022), between 1990 and 1993, the world industry lost a total of 20.3 billion dollars, consequence of the Gulf War and the ensuing economic recession. However, in the period between 1995 and 1999, industry profits reached about 35 billion. In the first decade of the twenty first century only in 2006, 2007 and 2010 the results were positive and between 2001 and 2005 the losses amounted to 41,5 billion dollars.

Events such as 9/11, the financial crisis of 2007-2010 and fuel price increase have led to the accumulation of losses. According to recent information, during 2015 the performance of commercial aviation was influenced by weather, mainly snowfall at the start of 2015, industrial action within airlines such as Lufthansa and Norwegian, as well as air traffic control disruptions mainly in France, gave origin to peaks in operational cancellations of planned flights. Likewise, similar causes have also resulted in delays in the US. Fig. 1.2 shows the leading causes of flight delays by the share of total delay minutes in the US from 2004 to 2019. The statistic shows the leading cause of flight delays by the share of total delay minutes in the U.S. from 2004 to 2019. In 2019, the leading cause of flight delays was late aircraft arrival which accounted for 39.7% of the total delay minutes.

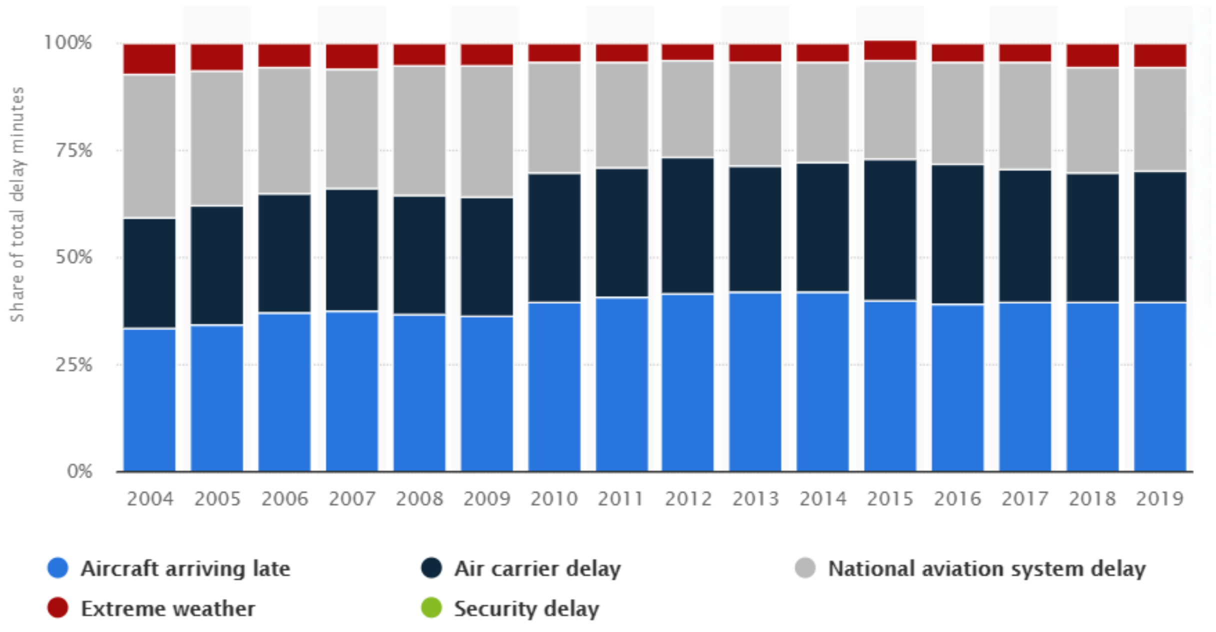


Fig. 1.2 Share of total minutes flights were delayed in the US from 2004 to 2019, by cause (source (Statista, 2021)).

More recently in the year 2020 the airline industry had 139.1 billion dollars in net losses (Airlines of America, 2022), due to the disruptive events related with the COVID-19 pandemic. Figure 1.3 depicts the economic performance for the years 2019 and 2020, grouped by geographical areas (IATA, 2020) (IATA, 2019). The suffix 'E' and 'F' in the column names stand for 'Expected' and 'Forecast'. Revenue Passenger Kilometres (RPK) is an airline industry metric calculated multiplying the number of paying passengers by the distance travelled. Available Seat Kilometres (ASK) is a measure of an airline's carrying capacity to generate revenue, taken from multiplying the available seats on any given aircraft by the number of kilometres flown on a given flight. The load factor is an indicator that measures the percentage of available seating capacity that is filled with passengers. Net post-tax profit per passenger, revenue, RPK, ASK and the load factor decreased between 2019 and 2020. Consequently, the break-even load factor had to increase to face the losses. As demonstrated in figure 1.3 the events related with the COVID-19 pandemic were unable to be forecast in 2019 (column 2020F).

In commercial aviation, the airline schedule represents the central element of planning (Grosche, 2009). If airline companies cannot deliver on time the passenger itinerary, they are liable to pay compensation for the resulting passenger inconvenience. This outcome has a negative financial impact on an airline. It thus becomes of the most importance to resume normal operations as quickly as possible. The process of monitoring and allocating resources near the day of operations to bring operations back on schedule as rapidly as possible, while incurring minimal costs is designated Disruption Management (DM) (Kohl et al., 2007).

In conclusion, in case of disruptions to the planned flight schedule, a flight schedule must be determined for a period of time designated by recovery period, and an effective solution must be found to minimise the resulting costs and the potential impacts to passengers.

1.3 Current Approaches on Disruption Recovery

This section introduces the relation cause effect between disruption in commercial aviation and recovery methods commonly used to resume normal activity.

(Leung, 2004), mentions that disruption management is a process that requires the ability to dynamically revise the original schedules to suit the newly changed operational environment.

Delay is a critical indicator to evaluate schedule performance. Generally, the delays can be specified into two parts, namely, the primary delay (also called non-propagated or independent delay) and Propagated Delay (PD). Thus, many studies focused on robust schedule via delay or PD minimisation.

(AhmadBeygi et al., 2008) investigates the relation between schedule planning and delay propagation. The data source used is the Bureau of Transportation Statistics (BTS) to measure the propagation of a single flight delay through the entire network. The modelling consists of a network of resources used during the flight (e.g. cockpit crew and aircraft). The work of (AhmadBeygi et al., 2008) uses a set of Key Performance Indicators (KPI) to measure the effects of a root delay in the network:

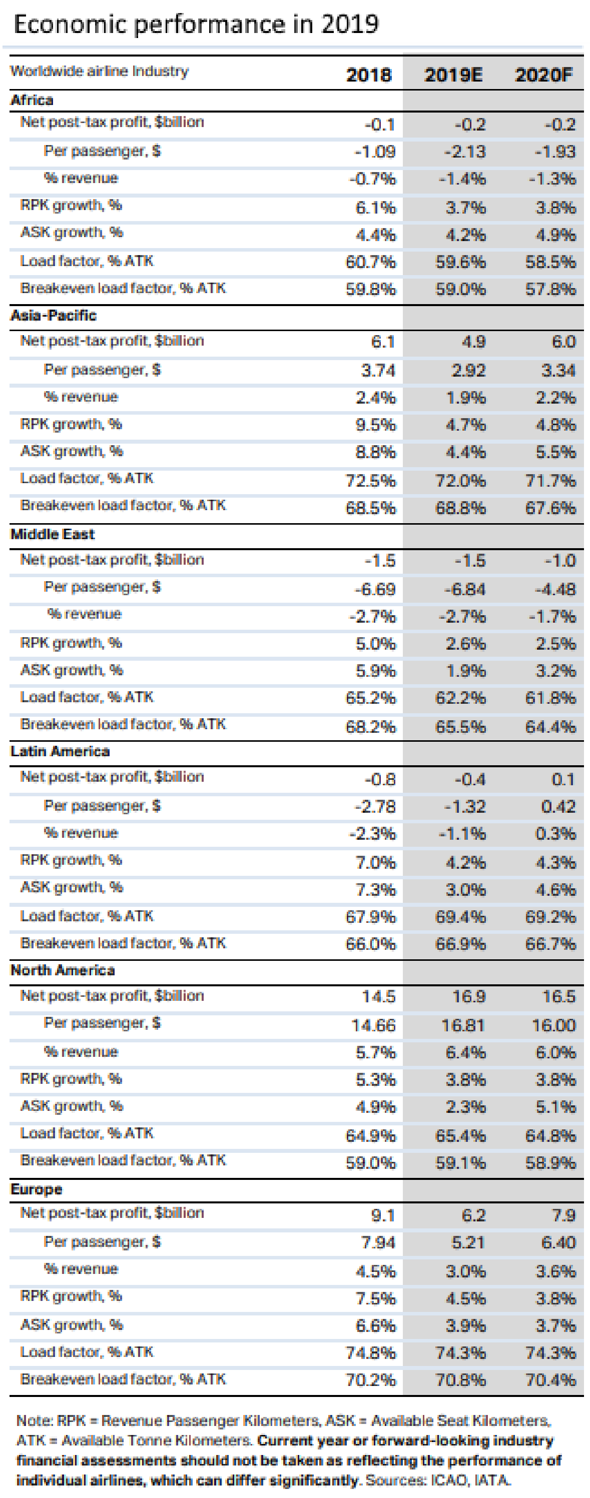
1. PD: The sum of the delays (in minutes) imposed on downstream flights by an initial root delay in a propagation tree.

2. Magnitude: The ratio of PD to root delay.

3. Severity: The number of disrupted flights, excluding the root flight itself.

4. Depth: The number of nodes in the longest path in a propagation tree, not counting the root delay.

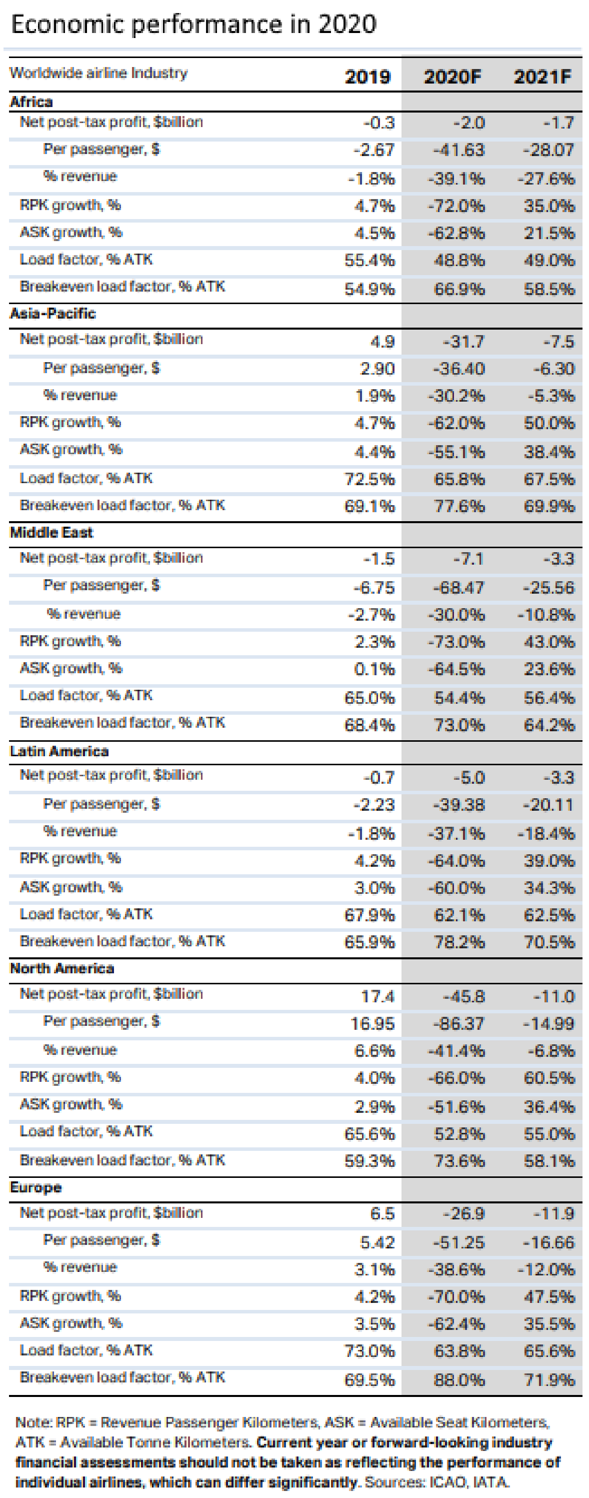
5. Depth ratio: The ratio of depth to severity in a propagation tree.



Economic performance in 2019

Worldwide airline Industry	2018	2019E	2020F
Africa			
Net post-tax profit, $billion	-0.1	-0.2	-0.2
Per passenger, $	-1.09	-2.13	-1.93
% revenue	-0.7%	-1.4%	-1.3%
RPK growth, %	6.1%	3.7%	3.8%
ASK growth, %	4.4%	4.2%	4.9%
Load factor, % ATK	60.7%	59.6%	58.5%
Breakeven load factor, % ATK	59.8%	59.0%	57.8%
Asia-Pacific			
Net post-tax profit, $billion	6.1	4.9	6.0
Per passenger, $	3.74	2.92	3.34
% revenue	2.4%	1.9%	2.2%
RPK growth, %	9.5%	4.7%	4.8%
ASK growth, %	8.8%	4.4%	5.5%
Load factor, % ATK	72.5%	72.0%	71.7%
Breakeven load factor, % ATK	68.5%	68.8%	67.6%
Middle East			
Net post-tax profit, $billion	-1.5	-1.5	-1.0
Per passenger, $	-6.69	-6.84	-4.48
% revenue	-2.7%	-2.7%	-1.7%
RPK growth, %	5.0%	2.6%	2.5%
ASK growth, %	5.9%	1.9%	3.2%
Load factor, % ATK	65.2%	62.2%	61.8%
Breakeven load factor, % ATK	68.2%	65.5%	64.4%
Latin America			
Net post-tax profit, $billion	-0.8	-0.4	0.1
Per passenger, $	-2.78	-1.32	0.42
% revenue	-2.3%	-1.1%	0.3%
RPK growth, %	7.0%	4.2%	4.3%
ASK growth, %	7.3%	3.0%	4.6%
Load factor, % ATK	67.9%	69.4%	69.2%
Breakeven load factor, % ATK	66.0%	66.9%	66.7%
North America			
Net post-tax profit, $billion	14.5	16.9	16.5
Per passenger, $	14.66	16.81	16.00
% revenue	5.7%	6.4%	6.0%
RPK growth, %	5.3%	3.8%	3.8%
ASK growth, %	4.9%	2.3%	5.1%
Load factor, % ATK	64.9%	65.4%	64.8%
Breakeven load factor, % ATK	59.0%	59.1%	58.9%
Europe			
Net post-tax profit, $billion	9.1	6.2	7.9
Per passenger, $	7.94	5.21	6.40
% revenue	4.5%	3.0%	3.6%
RPK growth, %	7.5%	4.5%	3.8%
ASK growth, %	6.6%	3.9%	3.7%
Load factor, % ATK	74.8%	74.3%	74.3%
Breakeven load factor, % ATK	70.2%	70.8%	70.4%

Note: RPK = Revenue Passenger Kilometers, ASK = Available Seat Kilometers, ATK = Available Tonne Kilometers. **Current year or forward-looking industry financial assessments should not be taken as reflecting the performance of individual airlines, which can differ significantly.** Sources: ICAO, IATA.



Economic performance in 2020

Worldwide airline Industry	2019	2020F	2021F
Africa			
Net post-tax profit, $billion	-0.3	-2.0	-1.7
Per passenger, $	-2.67	-41.63	-28.07
% revenue	-1.8%	-39.1%	-27.6%
RPK growth, %	4.7%	-72.0%	35.0%
ASK growth, %	4.5%	-62.8%	21.5%
Load factor, % ATK	55.4%	48.8%	49.0%
Breakeven load factor, % ATK	54.9%	66.9%	58.5%
Asia-Pacific			
Net post-tax profit, $billion	4.9	-31.7	-7.5
Per passenger, $	2.90	-36.40	-6.30
% revenue	1.9%	-30.2%	-5.3%
RPK growth, %	4.7%	-62.0%	50.0%
ASK growth, %	4.4%	-55.1%	38.4%
Load factor, % ATK	72.5%	65.8%	67.5%
Breakeven load factor, % ATK	69.1%	77.6%	69.9%
Middle East			
Net post-tax profit, $billion	-1.5	-7.1	-3.3
Per passenger, $	-6.75	-68.47	-25.56
% revenue	-2.7%	-30.0%	-10.8%
RPK growth, %	2.3%	-73.0%	43.0%
ASK growth, %	0.1%	-64.5%	23.6%
Load factor, % ATK	65.0%	54.4%	56.4%
Breakeven load factor, % ATK	68.4%	73.0%	64.2%
Latin America			
Net post-tax profit, $billion	-0.7	-5.0	-3.3
Per passenger, $	-2.23	-39.38	-20.11
% revenue	-1.8%	-37.1%	-18.4%
RPK growth, %	4.2%	-64.0%	39.0%
ASK growth, %	3.0%	-60.0%	34.3%
Load factor, % ATK	67.9%	62.1%	62.5%
Breakeven load factor, % ATK	65.9%	78.2%	70.5%
North America			
Net post-tax profit, $billion	17.4	-45.8	-11.0
Per passenger, $	16.95	-86.37	-14.99
% revenue	6.6%	-41.4%	-6.8%
RPK growth, %	4.0%	-66.0%	60.5%
ASK growth, %	2.9%	-51.6%	36.4%
Load factor, % ATK	65.6%	52.8%	55.0%
Breakeven load factor, % ATK	59.3%	73.6%	58.1%
Europe			
Net post-tax profit, $billion	6.5	-26.9	-11.9
Per passenger, $	5.42	-51.25	-16.66
% revenue	3.1%	-38.6%	-12.0%
RPK growth, %	4.2%	-70.0%	47.5%
ASK growth, %	3.5%	-62.4%	35.5%
Load factor, % ATK	73.0%	63.8%	65.6%
Breakeven load factor, % ATK	69.5%	88.0%	71.9%

Note: RPK = Revenue Passenger Kilometers, ASK = Available Seat Kilometers, ATK = Available Tonne Kilometers. **Current year or forward-looking industry financial assessments should not be taken as reflecting the performance of individual airlines, which can differ significantly.** Sources: ICAO, IATA.

Fig. 1.3 Economic performance of the airline industry for 2019 and 2020 (source (IATA, 2020) (IATA, 2019)).

6. Stay: The number of nodes (disrupted flights) in which the crew and the aircraft are the same as in the preceding node.

7. Crew-out: The number of nodes (disrupted flights) in which the crew is not the same as the preceding node, because the crew in the preceding node has ended their pairing.

8. Split: The number of nodes (disrupted flights) in which either the crew or the aircraft is not the same as the preceding flight, because these resources split to serve two different subsequent flights.

9. Split ratio: The ratio of split to severity in a propagation tree.

The case study used in (AhmadBeygi et al., 2008) involves creating delays that last between 15 to 180 minutes in increments of 15 minutes and constructing the propagation tree. The comparison using the KPI are made between hub-and-spoke carrier and point-to-point carrier. The authors conclude that in terms of delay propagation the consequences are more extended if the root flight delay is earlier, because there are more opportunities for delay. Delay can be eliminated by the end of the day, due to overnight break, but some flight delays do propagate overnight, because the crews have short overnight rest periods forcing them to delay their first flight the next day. The delay propagation has significant difference when factoring by departure time of the day, with severity, depth, and magnitude decreasing as the origin time of the root flight increases later into the day. Considering the latter, (AhmadBeygi et al., 2008) concluded that a disrupted flight early in the day will benefit more substantially from increased slack to absorb disruption than will a flight disrupted later in the day.

The work of (Gopalakrishnan et al., 2016) provides an approach to analyse a set of networks consisting of airports connected by flight delays in order to identify characteristic delay states and characteristic types of day that take into account both spatial and temporal patterns and connectivity. The state of the air transportation network is represented as a weighted directed graph in which the nodes represent the airports, and the weight of the edges consists of the delays experienced between the origin airport and the destination airport. The study uses a set of algorithms to determine clusters of traffic delay in the network, and to determine sequences of delays. These patterns of delay are evaluated using eigenvector centralities and the hub and authority scores of different nodes. The paper also covers the grouping of airports based on their closeness in terms of delay state. To model the problem, (Gopalakrishnan et al., 2016) first build a set of networks using the BTS as the data source. The data was collected for a period of two years, and the set of networks was built using the hourly delay as the weight for the edges. The total number of networks for the two year period is:

```
731  days × 24  hours = 17,544  networks
```

To reduce complexity in comparing networks the authors chose to study features that represent the nodes namely integrated in-degree and out-degree, eigenvector centralities, and the hub and authority scores of nodes to evaluate the similarity between nodes. Eigenvector centrality identifies which nodes have a wide-reaching influence within a given network. Hub and spoke authority scores reflect the connectivity and the propensity for delay to propagate into and out of a node in a network. This score is based on the links made to a specific node from other nodes, therefore the authors of this work chose the importance of a node in the network would be given by eigenvector centrality. The process used to compare two networks uses the Euclidean distance to measure similarities between their feature vectors. K-means and k-medoids algorithms were used to determine clusters of similar graphs. Gopalakrishnan et al. 2016 conclude that spatial and time series patterns obtained can be used to develop predictive models of air traffic network delays that can result in advancements in decision support for stakeholders and air traffic managers.

Most researchers choose string-based networks to model PD because delay propagates along the routes. (Liang et al., 2015) present an accurate computation of the Expected Propagated Delay (EPD) of a string. Because computing the EPD of a string could be time-consuming, they use a tight lower bound for estimating the EPD and propose a two-stage column generation method that makes use of the lower bound to speed up the solution process. Large test cases with more than 6000 flights per week can be solved within 3 hours.

Instead of minimising EPD, (Yan and Kung, 2018) sought to minimise the maximal possible PD. When flight leg delays lie in a pre-specified uncertainty set, using a robust optimisation approach, they propose an exact decomposition solution approach under a column-and-row generation framework. Their approach is reported to outperform the local approach provided by (Dunbar et al., 2014) by reducing both mean and extreme total PD.

As aforementioned, many studies on disruption recovery are based on the string-based network. However, because of the large number of possible connections, enumerating all feasible strings is impractical. The column generation framework has been proven efficient and effective for such problems with a huge number of strings. Other methods, such as Branch and Bound (BaB), Lagrangian Relaxation (LR), Benders Decomposition (BD), *etc.* are also proposed to solve different models.

Another important approach to mitigate the effects of disruption that has gained attention from the scientific community consists in changing cruise speed. In the work (Marla et al., 2017) the authors explore the trade-off between delays and fuel burn. This work makes use of the CI for a flight to capture both the flight time and fuel burn. In terms of the applied concept, it explores two alternatives, the first consisting in flight speed changes to reduce the travel time, and the second consisting in combining flight speed changes with holding strategies to preserve passenger connections. Using an airline disruption management simulator the authors evaluated, for each possible flight speed, the trade-off between the fuel cost versus passenger-related delay costs to the airline. The authors used JetPlan™, a flight planning tool developed by Jeppesen Commercial and Military Aviation™.

1.4 Aims and Objectives

Flight planning is the basis of airline scheduling and there are many publications that model specific parts of the flight e.g. ground and air movements. Since flight planning studies are focused in specific flight phases there are no scientific models that integrate each flight phase from gate to gate. Additionally, each of the data sets are difficult to obtain, modelling is complex, and most importantly and obviously companies do not share their core business knowledge base.

Modelling airline disruption even in the simplest form, is an arduous task, that requires a deep understanding of the data sets, parameters and decision variables that are involved in the process, namely those related to flight routes, aircraft specifications, airport capacity and passenger itineraries. Moreover, it is also important to have access to data that can be used to simulate the occurrence of disruptions such as flight delays, airport capacity decrease and aircraft availability shortage. Due to its complexity, from the perspective of computer science, finding solutions for this type of problem is quite challenging. Recover aircraft rotations is designated as the Aircraft Recovery Problem (ARP) and it implies finding solutions that comply with a set of constraints, namely flight continuity, transit or turn-round time, maintenance, and airport capacity. In its essence the ARP is a Constraint Satisfaction Problem (CSP). CSPs are commonly tackled using an approach that formulates the problem and systematically employs deductive reasoning to reduce the search space. This method is designated by CP.

Considering the nonexistence of a comprehensive flight planning model, and CP formulation for the ARP, exposed in the previous paragraphs one should question:

1. Could it be possible to develop a flight planning model that combines all phases of a flight? Calculate block time and the fuel consumed by an aircraft on a flight from the origin gate to the destination gate airport in a matter of seconds? Even though this modelling is to be made in a chaotic system such as Earth's atmosphere, one can actually combine statistical data and answer these questions.

2. Is it possible to create a disruption recovery procedure based on CP?

3. Moreover, in the presence of disruption, is it possible to mitigate or recover its effects by speeding up flights?

This PhD dissertation answers these questions by:

1. Developing the BTF model, designed for flight planning, in Chapter 3. The contributions are aggregating all the flight phases, calculating the time taken for a flight between the departure and the arrival gate, and calculating the fuel consumed by the aircraft.

2. Developing the Constructive Heuristic for the Aircraft Recovery Problem (CHARP) based in CP, in Chapter 4. The novelty of this contribution consists of an algorithm that can find feasible solutions using constraint propagation to reduce the search space and therefore reduce computing time (Vilain and Kautz, 1986), (Demassey et al., 2005).

3. Merging the BTF, which decreases the the initial block time, with the CHARP, in Chapter 5. The contribution is to determine the disruption scenarios for which smaller block times render lower cost recovery.

1.5 Summary of the Thesis

Chapter 2: Literature Review

Chapter 2 is divided in six sections. Section 2.1 provides a review of relevant work done regarding flight planning, in particular the methods and techniques to model flight trajectory and flight phases using institutional data. Section 2.2 presents examples of real-world situations solved efficiently using CP models, Section 2.3 reviews the methods to recover disrupted aircraft rotations, Section 2.4 extends the latter topic by including in the disruption model the Crew Recovery Problem (CRP) and the Passenger Recovery Problem (PRP). Section 2.5 reviews the publications that investigate the general benefits of flight speed changes to save fuel, manage flight time, and as a mean for flight disruption recovery. Section 2.6 summarises the literature review on these topics and areas of further research.

Chapter 3: The Block Time and Fuel Model

Chapter 3 is divided in five sections. Section 3.1 introduces the topic, Section 3.2 describes the flight phases, Section 3.3 describes the BTF model, Section 3.4 presents the results and finally Section 3.5 draws the conclusions and future work.

Chapter 4: The Constructive Heuristic for the Aircraft Recovery Problem

Chapter 4 introduces the ARP model and the method that is used to find solutions, Section 4.2 describes the ARP model, its parameters and formulation, Section 4.3 describes the constraint programming concepts that will be used to find solutions for the ARP, Section 4.4 provides a detailed overview of the algorithms and the workflow of the CHARP, Section 4.5 presents the computational results for different scenarios and compares them with published work. Section 4.6 makes an extensive comparison regarding the solution methods, disruptions, recovery actions and problem characteristics, between published literature on the ARP and the CHARP. Finally, Section 4.7 presents the conclusions and future work.

Chapter 5: The Impact of Smaller Block Times in Disruption Recovery

Chapter 5 presents the inclusion of the BTF results in the data set that is used by the CHARP. Section 5.2 compares block time differences between the BTF and the data set for westbound and eastbound flights. Section 5.3 presents the CHARP results using smaller block time flights for three distinct scenarios. Section 5.4 studies the effects of smaller block times for six different disruption scenarios. Finally, Section 5.5 presents the conclusions and future work.

Chapter 6: Contributions and Future Work

Chapter 6 summarises all models developed and the methods that were used to solve them, highlighting first and foremost the contributions of this thesis. Additionally, this section makes use of critical thinking to expose some of the limitations of the models proposed.

Chapter 2

Literature Review

This chapter is entirely devoted to analysing published scientific papers related with the topics of research of this thesis. Whilst doing this literature review, it should be clear that the use of the expression "the authors" addresses the specific paper under consideration.

Flight planning is the process that intends to describe a proposed aircraft flight. It involves safety-critical aspects such as fuel calculation, to ensure that the aircraft can safely to the destination, along an optimised trajectory. Section 2.1 makes an extensive overview of flight trajectory modelling namely in Subsection 2.1.1 studies fuel consumption, 4D trajectories, and optimisation methods. Flight planning aims at reducing fuel consumption, environmental impact and mitigate risks of aircraft collision or crashing. Subsection 2.1.2 reviews the studies that have evaluated the benefits of continuous or optimised profile descents and climbs. In general terms, these works are based on flight paths obtained from airspace control agencies. These flight paths and other flight information are then used in conjunction with an aircraft performance model, such as Eurocontrol's Base of Aircraft Data (BADA) (EUROCONTROL, 2014) to estimate fuel consumption and the flight altitude profile. BADA is an aircraft performance model which is based on the total energy model of the aircraft and can be considered as a reduced point-mass model and it also provides data sets for flight modelling based of the aircraft. By the end of Section 2.1 the reader should be familiarised with the flight phases, numerical methods, and algorithms to model flight trajectories, consumed fuel, and pollutant emissions during a flight.

Commercial aviation industry is quite diversified and heavily regulated. Several problems such as scheduling and sequencing are CSP and are typically solved using a form of search. CP is a powerful technique for solving these problems. CP was developed by the artificial intelligence community in the late 1980s (Jaffar and Lassez, 1987) and (Jaffar and Maher, 1994), (Van Hentenryck, 1989). Currently, it successfully solves many decision problems and optimisation problems. However, it is often inaccessible to users without expert knowledge in the area, excluding the wide-spread use of CP techniques. These techniques combine search algorithms with Constraint Propagation (CPr) to try to find good solutions quickly without testing each possible combination of variable assignments. Some approaches are systematic and guaranteed to produce an optimal solution while others give up optimality in an effort to find near-optimal solutions faster. Section 2.2 reviews the CP models used to solve commercial aviation related problems and by the end of it one should be clued in with its potential for solving complex problems in commercial aviation.

A considerable number of factors, such as mechanical failures, crew delays, problems with ground operations, industrial action, inclement weather, congestion at airports can cause disruptions such as flight delays or cancellations. As a result, the initial rotation becomes inefficient in the sense that flights must be delayed or cancelled. Since airline companies are liable to pay compensation for the resulting passenger inconvenience in these situations which in turns leads to a negative financial impact on an airline's profits. It thus becomes of the utmost importance to resume normal operations as quickly as possible. Section 2.3 presents some of the relevant literature in aircraft recovery classified according to the methods used to find solutions for the ARP. Even though the topic of this dissertation does not cover integrated recovery Section 2.4 reviews the topic since it also covers the methods to find solutions for the ARP.

Flight speed changes is the concept of adjusting the block time of a flight. For example, if a flight's departure is delayed by five minutes due to industrial action, the delay can be mitigated by a subsequent flight speed change decision that decreases the block time. On the other hand, if the Air Traffic Management (ATM) informs in advance that the aircraft will be on hold before landing the pilot can adjust the cruise speed to allow the aircraft to fly slower with the same or lower fuel consumption in the original flight. Therefore, flight times are increased, and the holding delay can be partially performed in the air, at no extra fuel cost for the operator. Section 2.5 verses about the benefits of flight speed changes on aircraft fuel consumption, and in the block time as a means of recovery of disrupted flights.

2.1 Flight Planning

This section outlines pertaining flight trajectory modelling and the use of BADA.

2.1.1 Flight Trajectory Modelling

In one of the initial studies that modelled the trajectory for fuel consumption optimisation, (Neuman and Kreindler, 1985) created an algorithm capable of calculating the minimum fuel consumption during the climb phase from 2,000 feet to 10,000 feet for long-haul flights (6 to 12 hours of flight). (Neuman and Kreindler, 1985) derived a system of dynamic equations considering small angles of attack and trajectory, coordinated turns, absence of atmospheric winds, and also assumed that the weight of the aircraft remained constant. Approach altitude is considered above 2,000 feet and climb, approach and landing speeds are set for a commercial jet airliner. The authors concluded that the trajectories combined with latero-directional and longitudinal flight were optimal in terms of fuel consumption when there was little variation in altitude. Finally, the authors concluded that the amount of fuel that can be optimised during the climb and descent phase in the terminal area of the flight is small compared to the fuel spent during the cruise phase, nevertheless it is important to have these costs in account in flight planning.

Moreover, (Grimm et al., 1986) sought to quantify the importance of the change in the aircraft weight of the aircraft due to fuel consumption by developing a model that took into account two different approaches: the first assuming the aircraft has constant weight, and the second considering weight decrease as fuel is consumed. The authors used the Direct Multiple Shooting Method (DMSM). This method, which is used for optimising boundary constraints only in the state domain, consists basically of dividing the range at which solutions are searched into several smaller ranges. Because it is quite iterative, DMSM becomes heavy for processing systems such as 4D path optimisation problems.

In order to design an optimal trajectory for a commercial aircraft, taking into account realistic constraints along the trajectory, (Betts and Cramer, 1995) applied the direct transcription method. The goal was to achieve results that met civil aviation safety standards. In order to interpolate the points in the dataset and guarantee smoothness at the data points, aircraft aerodynamics and propulsion data were processed using a cubic spline product tensor. Cubic spline interpolants are continuous in the zeroth through second derivatives and pass through all the data points, therefore becoming the choice to model the aircraft's equations of motion. This procedure was investigated for two different approaches through data interpolation and least squares taking into account the turning restrictions. This method may be fast due to the approach by cubic splines, but this is only an approximation and for a more accurate approach the need for continuous derivation can complicate the results.

Also, in the context of of 4D trajectories optimisation, Hagelauer and Mora-Camino (1998) conducted a study, in which they presented a method based on Dynamic Programming (DP), in the presence of various time constraints. A discrete formulation of the problem was proposed, and the optimisation problem was solved using a progressive DP framework. Processing time was decreased using neural networks to calculate the costs associated with each decision step in the search process. In this study, a two layer neural network has been found to be sufficient to provide good fuel flow approximations. In order to reduce the computation time of the simulated neural network, the classical sigmoid activation function has been replaced by 2.1:

$$f(x) = \frac{x}{|x|+1} \tag{2.1}$$

This method was called Soft Dynamic Programming (SDP). Comparing this method with the one, described in the previous paragraph, the authors concluded that the use of neural networks to calculate fuel consumption in each decision step reduced by 88.2% the time spent processing.

More recent work (Franco et al., 2010), based on (Neuman and Kreindler, 1985), consisted of analysing the cruise fuel minimisation problem for a fixed altitude and arrival time as a simple optimisation problem. The objective of this work was to verify the influence of cruising altitude in the calculation of optimal trajectories, calculating the minimum fuel required. Results were presented for a Boeing 767-300ER. The authors concluded that for higher altitude cruise levels, there is little influence on the Mach number variation with the aircraft weight. In fact, the highest fuel consumption optimisation rates are obtained at higher altitudes.

The flight phases in which an aircraft consumes more fuel are undoubtedly the climb and cruise. (Turgut and Rosen, 2012) studied the relationship between fuel consumption and altitude variation during the descent phase for a commercial transport aircraft. To find approximate solutions, the authors used a Genetic Algorithm (GA) composed of five modules, figure 2.1.

The first module includes features includes the population, iteration, crossover and mutation ratios and accuracy degree. Small intervals of variables are investigated for a wide range of data, which are obtained for real flight conditions from the flight data records. In the second module, the genomes are ranked to the values of the actual count of each genome. At the end of this task, the genomes that have a zero value of actual counts are replaced by genomes which have a maximum actual count. Modules three and four involve codes that perform the tasks of crossover and mutation. In the fifth module, the outputs of the first iteration are recorded in a data set. The output having the best solution in terms of objective function is also recorded on another data set with the values for variables along with the results from the iterations. Within the modules, auxiliary tasks are also conducted, such as routines for checking the proper performance of crossover and mutation and to prevent the loss of the best output from the population after the tasks of crossover and mutation. After analysing the results, the authors concluded that the optimum

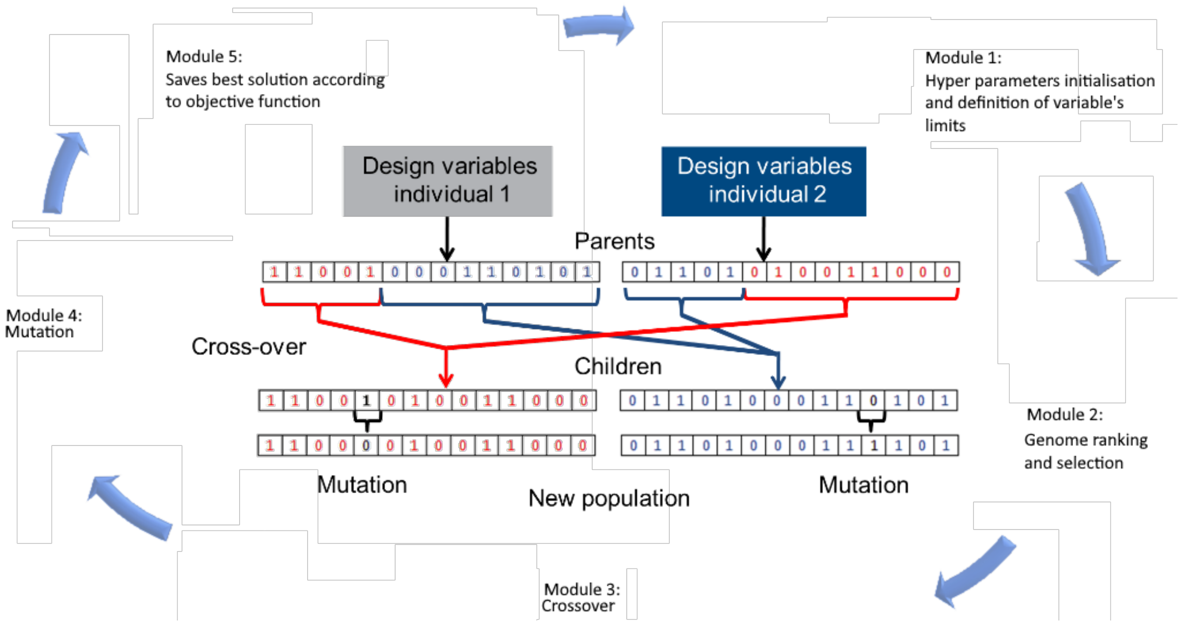

Module 5:
Saves best solution according to objective function
Module 1:
Hyper parameters initialisation and definition of variable's limits
Design variables individual 1
Design variables individual 2
Parents
Module 4:
Mutation
Cross-over
Children
Module 2:
Genome ranking and selection
Mutation
New population
Mutation
Module 3:
Crossover

Fig. 2.1 Genetic algorithm proposed by (Turgut and Rosen, 2012)

fuel consumption values occur when, during descent, the aircraft is kept as long as possible at higher altitudes.

The work of (Murrieta-Mendoza et al., 2017) optimises the fuel burn of the vertical profile of a commercial aircraft. The airspace was modelled under the form of a unidirectional graph. The selection of waypoints where to execute the changes in altitudes that provided the most economical flight cost in terms of fuel burn was determined using the Particle Sworm Optimisation (PSO) algorithm. To compute the flight cost the flight trajectory is divided in equidistant segments of 20 nautical miles. The authors claim that by doing so it is possible to achieve a good compromise between accuracy and computation time. At the beginning of each segment the model subtracts to the aircraft's gross weight the fuel burned required to travel the previous segment; the ground speed is computed considering the wind speed, wind direction and temperature; linear interpolations are executed to compute the fuel flow or the fuel burn and the horizontal travelled distance. Fuel burn was computed in two different ways depending on the cruise regime: steady altitude or change of altitude. The total fuel burn is calculated by aggregating all the segments taking into account, if necessary, the change of altitude cost. Aviation products from specialised numerical weather prediction predict the evolution of weather elements that can affect aircraft in flight, such as turbulence and icing (ice accumulation on aircraft), therefore the weather forecast was introduced in the model using Environment Canada. The trajectories provided by the algorithm developed in this paper were compared against simple geodesic trajectories to validate its optimisation potential, and against as flown trajectories. The authors claim that the results have showed that up to 6.5% of fuel burn can be saved comparing against simple trajectories, and up to 3.1% was optimised comparing against as flown trajectories.

In the work of (Hartjes et al., 2018) a tool is developed that optimises the trajectories of multiple airliners that seek to join in formation to minimise overall fuel consumption or direct operating cost. When in formation, a discount factor is applied to simulate reduction in the induced drag of the trailing aircraft. Using the developed tool, a case study has been conducted pertaining to the assembly of two-aircraft formation flights across the North-Atlantic, nevertheless information regarding aircraft distance between

them was not provided.
The authors mention that the results of the various numerical experiments show that formation flight can lead to significant reductions in fuel consumption compared to flying solo, even when the original trip times are maintained and that the performance and the characteristics of the flight formation mission - notably the location of rendezvous and splitting points - are affected when one aircraft seeking to join the formation suffers a departure delay.

A fuel flow rate model was developed in (Oruc and Baklacioglu, 2020) using flight altitude, True Air Speed (TAS) and fuel flow rate values obtained from B737-800 type passenger aircraft Flight Data Records (FDRs). In the model, fuel flow rate is achieved as a function of altitude and TAS. The fuel flow rate model uses a Cuckoo Search Algorithm (CSA) for the climbing phase of the flight. The CSA used in this study uses a combination of local and global search. The authors claim that this approach increases search richness and versatility while using Lévy flights makes the search area more efficient.

The European current political agenda includes two programmes to improve safety, environmental, and efficiency indicators, SESAR 2020 and Clean Sky 2, respectively. One of the ways to reduce the negative impact of air transport on the environment and improve its efficiency is to reduce fuel consumption and pollutants emissions resulting from fuel combustion. (Pawlak et al., 2021) focused their work in the cruise phase when aircraft flies at a constant altitude with a constant air speed. The result of the analysis was the development of methodology for fuel consumption and emission of main pollutants in cruise conditions. The model calculates fuel consumption for the thrust required for horizontal flight at cruising altitude. The procedure consists of four steps. The first consists in determining the performance parameters of the aircraft engines for a given altitude and cruising speed. The second calculates the emission indexes of CO, NO_x, HC, and CO_2 for the cruise phase of flight. The third calculates the emission intensity for the aforementioned pollutants. Finally, in the fourth step the model computes the fuel consumed. The research was carried out for two flight variants — a route from Rome to Athens and a route from Athens to Rome. These routes have between 7 to 8 daily flights departing from Rome and landing in Athens airport, and vice versa, (Flightaware, 2022). Each flight will emit between 69 to 115 Kg of CO_2 per passenger (Google, 2022). Therefore, modelling these route's trajectory will potentially drive the reduction of daily CO_2 emissions. In both cases, this work assumes identical meteorological conditions for this geographical area, however the direction of the wind on the aircraft was symmetric. To determine the shortest path the authors used Dijkstra's algorithm. The model uses a network where the distances between vertices are the graph edges. The weights of the graph edges were determined based on wind parameters (wind speed and direction). Each edge weight corresponded to the time needed to pass over a given edge length. The Boeing 737-300 aircraft was selected to conduct the research. During the research, different flight trajectories were determined to minimise pollutant emissions. The authors conclude that the trajectory optimisation has a significant impact on the reduction of pollutants emission in the jet engines exhausts. Emission depends directly on the engine's run time, and the shorter the run time (shorter flight duration), the lower the emission.

2.1.2 Flight Modelling Using BADA

In the work of (Melby and Mayer, 2008), the authors analysed trajectory data at 34 US airports in one day focusing on the continuity of vertical flight profiles at air terminals and the benefits that could be achieved by implementing vertical orientation Performance Based Navigation (PBS) procedures. For this, they applied a metric that considers the time in level flight during the descent or ascent to the trajectories. The time in level flight metric evaluated the time selected departure operations required to climb through 100 ft of altitude. Similarly, the metric evaluated the average time selected arrival operations required to descend through 100 ft of altitude. Fuel consumption estimates were made with the BADA based model. The results show that operators could save 380 million dollars annually and associated reductions in carbon dioxide emission gases of 850,000 metric tons with more efficient descent and climb profiles.

To estimate the benefits of continuous descents in congested airspace (Robinson III and Kamgarpour, 2010), developed a model that built trajectories from flight plans at 8 air terminals in the US over a period of thirty to sixty days (depending on the airport). The descent phase flight segments were identified, and two types of continuous descent trajectories were modelled. In the first, level flight segments were moved to higher altitudes and a distance-only constraint was applied to simulate non-congested airspace. In the second, the level flight segments were also moved to higher altitudes, but now with a time constraint, to simulate congested airspace. BADA was used in this work as well, and the results show that potential savings are sensitive to the size and diversity of traffic analysed (e.g. number of days, flights, aircraft mix, etc.). The authors mention that in general, the mean fuel savings per flight is between 20% to 80% greater than the median fuel savings for the same traffic sample. One cause of the disparity between the mean and median values is that fuel flow rates vary by aircraft type. For example, fuel flow rates for aircraft of the super-heavy and heavy weight-classes are 2 to 4 times larger than those of aircraft of the large weight-class, and more than eight times larger than those of regional and business jets. Thus, whereas aircraft of the super-heavy and heavy weight-class represent approximately 8% of all operations, they account for approximately 22% of the potential fuel savings.

In (Knorr et al., 2011), in addition to continuous descents, cruising speed reductions are exploited to absorb delays. This is encouraged by the fact that much of the extra fuel consumed is related to aircraft sequencing problems at the terminal. The authors' methodology was based on four principles: the first consisting of supporting the analysis of a large number of flights, without detailed wind or aircraft weight data. The second consisting of the use of surveillance data for position information. The third consisting of the use of BADA table for aircraft performance information, and the fourth consisting of the potential benefit expressed in terms of time and fuel. The results show that the potential for improvement at the terminals averages 3 minutes per flight or 100 kg of fuel. Reductions in cruising speed can save up to 30% of total extra fuel, compared to optimal trajectory.

Using similar techniques, (Howell and Dean, 2017) assesses the impact of the Federal Aviation Administration (FAA) initiatives on the US flight efficiency during the descent phase. For this purpose, flight path data was collected between the years 2010 and 2015. Potential fuel savings were calculated for each flight, identifying FL segments in the descent phase, and comparing the total fuel burned on each FL segment with the total fuel that would have been burned if all these FL segments were moved to the cruise phase. The calculation for potential time savings followed the same method. BADA was used for aircraft fuel consumption estimates. The results show that there has been a significant improvement in fuel efficiency, especially in places where optimized profile descents have been adopted.

The previous studies concentrate in the CCD phases, however aircraft surface movements in airports have also been the focus of research. (Khadilkar and Balakrishnan, 2012) built a model that, given the taxi trajectory (for example, from a surface surveillance system), can estimate the resultant fuel burn from observations of the aircraft's position, velocity and acceleration during taxiing. The authors used the flight data recorder (FDR) measurements of key aircraft parameters. FDR archives belonging to an international airline, from over 2300 flights in the year 2004, were used in this study and introduced into two linear models for estimation of the taxi-out fuel burn. The first model is based on an initial hypothesis consisting of the total fuel burn on the ground would be a function of the taxi time, number of stops and number of turns made by the aircraft. The second incorporates lessons learned from the first model namely, that other factors might be more important determinants of fuel burn. The authors removed the number of stops and the number of turns from the regression, and instead added the number of acceleration events as an independent variable. The logic behind this decision was that fuel flow rates were seen to increase for aggressive starts from standstill, as opposed to gradual ones. The parameters of both models were calculated using least-squares regression. According to the authors, taxi time is the main driver of fuel consumption. Their work presented an accurate estimate of the fuel burn index [1] and, proved that a good estimate of the fuel consumption of a surface trajectory can be obtained using just the taxi time.

In (Ravizza et al., 2013) the authors have developed a prediction model that combines both airport layout and historic taxi time information within a multiple linear regression analysis, identifying the most relevant factors affecting the variability of taxi times for both arrivals and departures. The two main applications for this research are for total taxi time prediction and for use in a ground movement decision support system. The authors use multiple linear regression to find a function which could more accurately predict the taxi times than existing methods and concluded that the average speed between the gate and runway (and between the runway and gate) was found to be highly correlated to the taxi distance, with higher speeds being expected for longer distances. Arrivals had higher taxi speeds than departures, due to departure queues at the runway, and the quantity of traffic at the airport was also found to have a significant impact upon the average taxi speed, as identified by several variables in the resulting model. During taxiing the authors also concluded that turning angle and the operating mode (which runways were in use) were also highly correlated to the average taxi speed. (Ravizza et al., 2014) uses the same explanatory variables and shows an extensive analysis of different regression approaches for predicting taxi times at airports to demonstrate the performance of each. Six different approaches were analysed in detail: multiple linear regression, least median squared linear regression, support vector regression, M5 model trees, Mamdani fuzzy rule-based systems and Takagi-Sugeno-Kang (TSK) fuzzy rule-based systems.

In (Chen et al., 2016) the authors present a new active routing (AR) framework to model fuel consumption, with the aim of providing a more realistic, cost-effective, and environmentally friendly surface movement. The paper focuses on optimal speed profile generation using a physics-based aircraft movement model. The two modelling approaches were based respectively in BADA and in the ICAO engine emissions database. The authors tested the model for Manchester International Airport and concluded that the results reveal an apparent trade-off between fuel burn and taxi times irrespective of fuel consumption modelling approaches.

The aim of (Gardi et al., 2016) is to review the current scientific knowledge regarding the optimisation of transport aircraft flight trajectories with respect to multiple and typically conflicting objectives arising from the inclusion of multiple environmental and operational criteria, and to deduce or infer all the useful notions for the development of algorithms that are specifically conceived for the implementation in novel Communication Navigation and Surveillance/Air Traffic Management (CNS/ATM) and Avionics (CNS+A) systems. The Multi-Objective Trajectory (MOTO) model proposed in this work consists of an optimisation process split between control inputs and state variables. From these two the authors

[1]The authors defined fuel burn index in $[kg/(s\sqrt{T_{amb}})]$

establish a network of relations between the aircraft's engine thrust, fuel consumption, dynamics models and the atmosphere. These relations return the inputs for the emissions, noise and contrail models, which will return the values for the output and state variables. The authors conclude that MOTO algorithms have a clear potential to enable real-time planning and re-planning of more environmentally efficient and economically viable flight routes by simultaneously addressing the dynamic nature of both weather and air traffic conditions.

There are several approaches regarding fuel burn estimations and in order to compare them the work of (Enea et al., 2017) summarises the collaboration between researchers from several globally recognised institutions to address the question of fidelity of fuel estimation. Interviews were conducted initially to categorise common elements that typical ATM studies share. An international team of fuel modellers was assembled and participated by running their models on a common set of inputs. The outputs generated by these models, were categorised using metrics on empirical trajectories and other operational data, including predicted fuel burn. The set of flights analysed for this study was recorded in June 2015 from various airports in the US. The sample included flights with variable length, different origin-destination pairs, and various aircraft types and for different day of the month of June 2015. Flights from 19 different days, 65 origin-destination pairs and 16 different aircraft types were selected. To present a meaningful cross-comparison of the fuel burn models evaluated, only the subset of common flights was used for this analysis. This subset represented the maximum number of flights with valid fuel and Take-Off Weight (TOW) predictions from all the models. There was wide variability in the observed fuel burn error across all models, the smallest median error was obtained with the Dali [2] BADA 3 run with -3.9% while the largest median error was observed for the AFEST [3] run without known TOW, with -13.1% the latter representing a deterioration from the AFEST run with known TOW that presented a median error of -9.5%, hence showing the impact of the initial TOW error. The authors concluded that even the highest fidelity model will significantly under-perform if low quality input data is provided, thus one of the results of this work is that fuel burn estimation models with different level of complexity, present different level of accuracy performance.

In order to model the descent and approach to the destination airport (Glaser-Opitz et al., 2020) propose a Landing System where this phase can be done more efficiently and safe using performance based on terrain reference navigation using own created terrain elevation database, based on radar altimeter measurements compared to the overflown terrain. The simulations were performed for a flight arriving at Kosice airport Kosice Airport (KSC), a Boeing 737-800 aircraft, and the descend trajectory was modelled with BADA performance model as a continuous descent approach from proposed merging point to the KSC runway. The descend procedure for this airport was designed in cooperation with professional pilots and all simulations were created for KSC as continuous descent approach; procedures, based on real world airline data in compliance with Initial 4D (i4D)[4] trajectory and proposed merging point. Based on mentioned models and simulations, the Landing System prototype was developed, with BADA model based trajectory prediction capability. The authors developed a client/server interface for testing and further research activities which enabled the Landing System prototype to communicate with flight simulator and with datalink communication simulator. Although, the authors described thoroughly the

[2]Dali consists of an aircraft trajectory modeling toolbox and is developed by Airservices

[3]AFEST is developed by the Modeling and Simulation Branch in the NextGen Office at the FAA William J. Hughes Technical Center

[4]The core characteristic of i4D is making sure that trajectories are always synchronised between air and ground, which enables more efficient handling and certainty of flight profiles.

methods used to derive the landing system, they did not make any explicit conclusions regarding its efficiency or safety.

On the overall the previous paragraphs describe several methods for modelling ground movements, or Climb, Cruise and Descent (CCD) phases of a flight. The methods range from solving differential equations to linear regression, and the models do not encompass the complete flight from the departure to arriving gate. Integrating each of the flight phases can be achieved as it shall be demonstrated in Chapter 3, by combining data from institutional sources e.g. European Monitoring and Evaluation Programme (EMEP)/European Environment Agency (EEA) and BADA, and using Newtonian Mechanics.

2.2 Constraint Programming Models and Applications

CP models are commonly solved using a tree-search algorithm similar to branch-and-bound. In CP every variable is initialised at the root of the search tree with a set of possible values, known as its domain, which is continually reduced by the constraints as the search progresses deeper into the tree. The search continues until either the domain of every variable contains exactly one value or until the domain of at least one variable is empty. In the first case, the values represent a solution, and in the second case, the empty domain represents a violation of a constraint. Central to the success of constraint programming are propagators, contracting functions removing values proven not to be in any solution of a given constraint (Bessière, 2011), (Allignol et al., 2012), (Schwartz, 2015).

Practical applications of CP are discussed, for example, in (Freuder and Wallace, 2000), (Rossi et al., 2006b), (Wallace, 2007). This section presents a review of commercial aviation related problems with time-dependent action costs, as well as CP approaches that have been used to solve them.

2.2.1 Constraint Programming Applications in the Commercial Aviation

CP has been an area of research since the late 1980s and is now a mature technology that has been successfully used for tackling a wide range of real-world complex problems, especially for scheduling, (Baptiste et al., 2001). For the aircraft sequencing problem on a single runway, (Fahle et al., 2003) compared different exact and heuristic methods including a CP model. For the same single runway problem, (Díaz and Mena, 2005) presented a CP implementation and pre-processing techniques. In this paper describes an application that solves the problem of aircraft sequencing in airports using a single runway. The model assumes that the air traffic controller must compute a landing (take off) time for each plane in the horizon or airport. The cost is associated with the difference between the preferred time (for landing or taking off) and the time assigned to it. There is also a minimum separation time between planes that must be respected to avoid accidents.

Two independently developed column generation methods, (Junker et al., 1999), (Yunes et al., 2000) solve the pricing sub-problem using CP techniques. Both approaches integrate mathematical programming and constraint satisfaction techniques, taking advantage of their particular abilities in modelling and solving specific parts of the crew scheduling problem. This method was applied to a wide range of applications. A survey is made by (Gualandi and Malucelli, 2009), in which the authors collect several applications and advances of the CP-based column generation framework, where the master sub-problem is solved by traditional Operations Research (OR) techniques, while the pricing sub-problem is solved by CP.

Within the area of short term airline operational planning, the Tail Assignment (TA) problem consists of assigning flight legs to individual identified aircraft while satisfying all operational constraints, and optimising some objective function. TA should be solved as part of both the short and the long term airline planning (Grönkvist, 2006), (Gabteni and Grönkvist, 2009). A hybrid column generation and constraint programming solution is proposed. The authors claim this approach can be used to quickly produce solutions for operations management, and also to produce near optimal solutions for long and mid term planning scenarios.

Airline crew assignment problems are large-scale optimisation that can be formulated as a constraint satisfaction problem. In the work of (Fahle et al., 2002) each airline regulation is encoded by one or several constraints. The authors introduce an additional constraint which encapsulates a shortest path algorithm for generating columns with negative reduced costs. This constraint reduces the search space significantly. The resulting domain reductions are propagated to the other constraints which additionally reduces the search space. The authors used data of a large European airline and claim the the model demonstrates the potential of the proposed approach.

Airport runway scheduling can be affected by inclement weather namely in winter during snowfall. Runways can be temporarily closed to clear them from snow, ice and slush. (Pohl et al., 2021) propose an integrated optimisation model to simultaneously plan snow removal for multiple runways and to assign runways and take-off and landing times to aircraft. The authors present a time-discrete binary model formulation using clique inequalities and an equivalent CP model. To solve the winter runway scheduling problem optimally, the authors use a start heuristic based on CP that generates a feasible initial start solution. A column generation scheme is used to identify all variables of the binary program which are required to solve it optimally. Finally, the algorithm uses a branch-and-bound procedure to the resulting binary program. Additionally, the authors propose a method to discretise the planning horizon, which enables improved solutions and allows an efficient balance between solution quality and model size. This algorithm was applied to realistic instances from Munich International Airport. An analysis of resulting model sizes proves the ability of the approach to significantly reduce the number of required variables and constraints of the time-discrete binary program. The authors claim that this method computes optimal schedules in a short amount of time and often outperforms a time-continuous formulation as well as a pure CP approach.

2.3 Aircraft Recovery Methods

Researchers on airline disruptions in schedules focused on aircraft rotation recovery. A reason could be that aircraft are the scarcest resources in an airline and also the rules that determine airline schedules are straight forward (Clausen et al., 2010). Aircraft recovery is typically modelled as a network problem. Like many network routing problems, the adopted models are usually arc-based or path-based. Subsections 2.3.1 to 2.3.5 categorise the problem solving approaches used in literature based on the network representation models that were developed.

2.3.1 Connection Networks

In the seminal work of (Teodorović and Guberinić, 1984), the authors developed a network model that minimises the total passenger delay on an airline network and solved the problem of finding new routing and scheduling plan to optimality using a BaB heuristic. The authors model passenger delays explicitly but they assume that all passenger itineraries contain only a single flight leg. Given the considerable number of routings for even modest sized problems, it is doubtful that their approach could be materialised for practical problems. On the other hand, since companies manage their operations using hub and spoke [5] flights, this study would also have limited scope of application.

In the connection network, presented in the work of (Abara, 1989) (figure 2.2) each airport has two timelines, namely, departure and arrival. Nodes represent the time points of the departure and arrival of legs. The connection network uses three types of arcs, which are leg arcs, connection arcs, and original/terminal arcs.

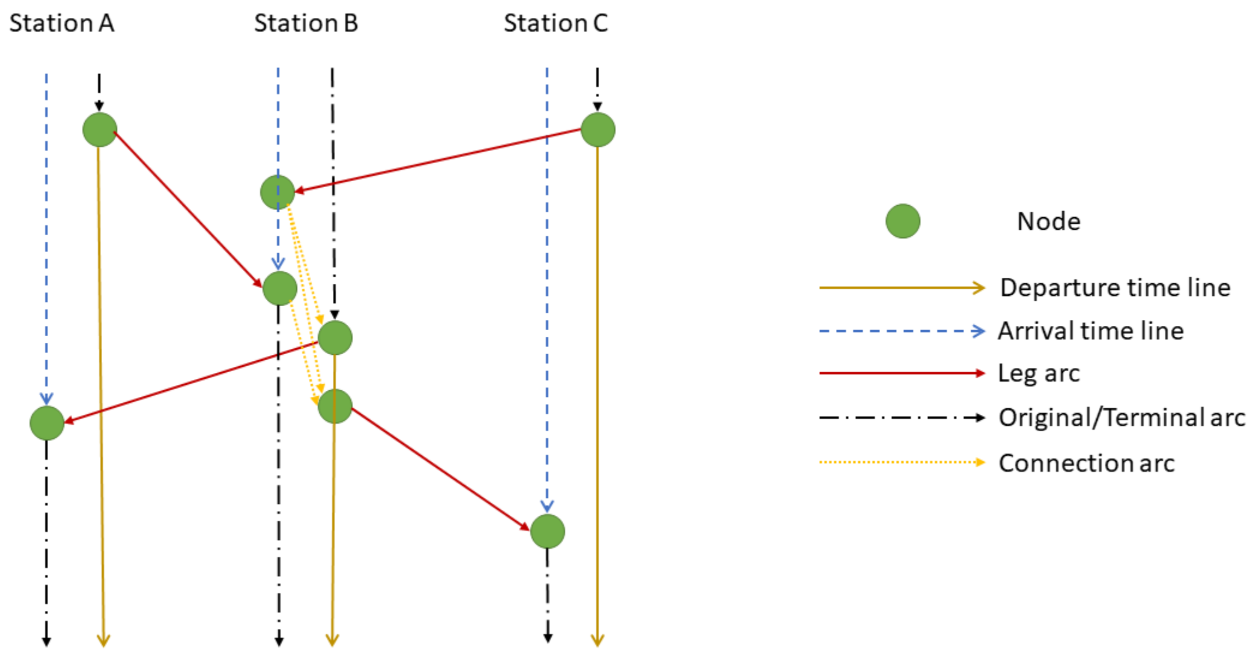


Fig. 2.2 Connection network proposed by (Abara, 1989)

The leg arcs represent different flights between airports. The connection arcs signify the possible aircraft connection between an arrival flight and a departure flight. The original arcs represent aircraft departing from the airport at the beginning of the day, and the terminal arcs represent aircraft arriving and remaining at the airport for the rest of the day. The objective of this formulation is to maximise the contributions of the flight legs over the costs of aircraft ownership, the cost of aircraft shortages (imbalances) and the cost of stations (airports). The constraints for this model enforce that each flight leg is assigned to exactly one aircraft type, only available aircraft are assigned, and that each aircraft type is assigned to the same number of flight legs arriving at the station as departing that station. (Rosenberger et al., 2003) presented an optimisation model for the aircraft schedule recovery problem, as a set-packing problem in which each leg is either in exactly one route or cancelled. In the procedure presented, aircraft recovery problem is solved for each fleet separately. The goal is to minimise the cost of flight leg cancellation and route re-assignment. Although set-packing problems are NP-Hard, the authors overcame this difficulty using an aircraft selection heuristic that allows to determine the subset of aircraft to reroute.

[5] A hub is a central airport that flights are routed through, and spokes are the routes that planes take out of the hub airport.

2.3.2 Time Line Networks

Linear Programming (LP) modelling is widely used to solve minimum cost network optimisation problems. Linear programming is a way of choosing interdependent activities, with inputs and outputs, so as to achieve an optimum in some dimension (e.g., profits or some index of welfare). The simplex method starts with a guess at a set of activities that are run in some measure. Then a set of prices are chosen to make the activities operate at zero profit. If none of the unchosen activities are profitable at these prices, then the initial set is optimum. If one is profitable, it is chosen, and one of the previously chosen ones is eliminated. The process is then repeated. In the end, the optimal set of activities will be obtained, (Dantzig, 2016). The simplex method is one of the most useful and efficient algorithms ever invented, and it is still the standard method employed on computers to solve optimisation problems. However, the simplex method is prone to get trapped into local maxima. The results of a simplex run depend on the starting conditions. In order to raise the chance of finding the global optimum, one should repeat several simplex runs with different starting conditions.

A great number of practical network problems are Nondeterministic Polynomial-Time Complete (NP-Complete) and therefore impossible to solve in a reasonable computing time. In such circumstances, it may be compelling to solve a simplified problem to obtain approximations or bounds on the initial hardest problem. Considering the optimisation problem where $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ and $S \subseteq \mathbb{R}^n$:

$$Minimise\ f(x) \tag{2.2}$$

$$Subject\ to\ x \in S \tag{2.3}$$

A relaxation of the above problem has the following form:

$$Minimise\ f_R(x) \tag{2.4}$$

$$Subject\ to\ x \in S_R \tag{2.5}$$

Where $f_R : \mathbb{R}^n \longrightarrow \mathbb{R}$ is such that $f_R(x) \leq f(x)$, $\forall x \in S$, $S \subseteq S_R$. The optimal solution f_R^* of the relaxation is a lower bound of the optimal solution of the initial problem. A large number of these problems have an underlying network structure. The goal of LR is to try to use the underlying network structure of these problems in order to use efficient algorithms, (Held and Karp, 1970), (Held and Karp, 1971), Geoffrion (1974). The LR is a method of decomposition: the constraints $S = S_1 \cup S_2$ of the problems are separated into two groups, namely the easy constraints S_1 and the hard constraints S_2, (Fisher, 2004). The hard constraints are then removed, i.e., $S_R = S_1$ and transferred into the objective function, i.e., f_R depends on f and S_2. Since S_R is a set of easy constraints, it will be possible to solve the relaxation problem. Moreover, the interest of the LR is that, in some cases, the optimal solution of the relaxed problem actually gives the optimal solution of the initial problem.

Time-line Networks base their implementation on the use of ground arcs, flight arcs and overnight arcs. Jarrah et al., 1993 present an overview of a decision support system for the aircraft recovery problem with the attempt of conceptualising the problem. The authors used a minimum cost network model, one delay model and one cancellation model and implemented an algorithm that solves the shortest path problem repeatedly to determine the necessary flows. Their decision support framework was to enable flight controllers decide when to cancel or delay flights. The possibility of swapping aircraft during the recovery period was considered where the swaps could involve spare aircraft or overnight layovers. The timeline network in this case had two node types (aircraft node and flight node) per station used to model the assignment of aircraft to flights. The research tested cases for both minor and major disruptions. The test scenarios were based on United Airlines' Boeing 737 fleet and a regional subdivision of America. One of the major drawbacks of this work consists of not having solutions that combine both flight delay and cancellation.

(Yan and Yang, 1996) developed a model that combined flight delays, cancellations, and ferrying that solved the perturbations of flight schedules using a time-space network flow model. Their model used simplex method to solve pure network flow problems and LR with sub-gradient methods solved the network flow problems with side constraints. Their paper presented four variations of the model, two of which are pure network flow problems while the other two are network flow problems with side constraints.

(Yan and ping Tu, 1997) extended the work of (Yan and Yang, 1996) focused on a single fleet, to accommodate multiple fleet. Their model was further extended to address cases of airport closures and multiple aircraft fleet substitutions for a network that consisted of 24 cities, 7 fleets and 273 flights with computation time below 30 minutes.

(Yan and Lin, 1997) developed a framework to help carriers handle schedule perturbation, due to temporary closure of airports, and resume their normal services as possible. This framework is based on a time-space network flow model, where the arcs in the network represent airborne flights, grounded flights, and overnight connections. The research was conducted over a set of scenarios, mathematically formulated either as pure network flow problems or network flow problems with side constraints, the former solved using simplex method and the latter using a LR-base algorithm, respectively. These models minimise the schedule-perturbed time after incidents so that carriers can resume their normal services as soon as possible. The salient features of the models consist in combining systematically flight cancellations, flight delays, the modification of multi-stop flights, the ferrying of idle aircraft and the swapping of aircraft. This procedure aims at adjusting effectively a schedule following incidents, so that a carrier can maintain its profitability. The authors confine the scope of this research to the operations of a single fleet as well as simplifying multi-leg flights by considering that the time block is from the beginning of the first leg to the end of the last leg.

(Cao and Kanafani, 1997a) and (Cao and Kanafani, 1997b) extended the work developed by Jarrah et al. (1993) evaluating solutions with flight cancellations and delays. The authors used a linear programming approximation algorithm based on a quadratic programming model that included both flight delays and cancellations. One of the major limitations of this work relates to the approach being limited to aircraft assigned to routes containing at most two flight legs. (Thengvall et al., 2000) solves the aircraft recovery problem for a single fleet over the entire flight schedule with an objective function that accounts for flight revenues, delays, cancellations, and an incentive to minimise deviations from the original schedule. The objective function consists in minimising total operating costs, but crew and passenger disruptions are not considered. The problem is modelled using a time-space network and solved as an integer linear program using an optimisation software. To represent the delays for a particular flight leg, a series of delay arcs are introduced to consider the available options for later flights. To ensure that only a single flight is chosen, a side constraint must be added requiring that the sum for all arcs representing the same flight is less or equal to one. The model was able to accommodate recovery periods of arbitrary length that begin and end at arbitrary periods of the day. In addition, a rounding heuristic is introduced to obtain feasible integer solutions from the linear programming relaxation of the mixed-integer programming formulation. As mentioned by the authors one weakness of the model is that it does not track individual passengers and thus does not consider passenger connections.

2.3.3 Time Band Networks

A time-band chart is defined as a chart containing two dimensions representing time intervals and airport stations respectively. (Thengvall et al., 2001) extended the work of (Arguello et al., 1997) and (Thengvall et al., 2000) by solving the multi-fleet aircraft schedule recovery, using multi-commodity network-type models during a hub closure. The model consisted of three cases, the first was a pure network with side constraints, the second a generalised network, and the last was a pure network with side constraints where the time horizon is discretised. The first two cases aimed at maximising the profit function while the other cases aimed at reducing the sum of delay and cancellation costs. The data spans 2 and 1/2 days and includes 332 active aircraft from 12 different fleets. Each fleet has from one to six sub-fleets for a total of 28 different types of aircraft. The schedule includes 2921 flights between 149 domestic and international locations.

2.3.4 Integer Programming

In their paper, (Andersson and Värbrand, 2004) use a mixed integer multi-commodity flow model with side constraints further reformulated into a set packing model with generalised upper bound constraints and using the Dantzig Wolfe decomposition. Disruptions are solved using cancellations, delays and aircraft swaps and the model ensures that the schedule returns to normal within a certain time. Computational tests and results show the capability of the model to provide quality optimised solutions in seconds and therefore, fit to be used as a dynamic decision support tool by airlines.

The model proposed by (Hu et al., 2011) established an integer programming model based on a time-band network and a passenger transiting network for the combined aircraft and passenger recovery problem. Solutions for single-fleet aircraft recovery were obtained by first solving the linear programming relaxation and then applying a rounding heuristic. A simulation entailing disturbance experiment included temporary airport closure. The hybrid method provided an efficient short-haul schedule recovery solution.

Constructing the network for each aircraft and solving the corresponding integer linear program can be very time-consuming. To achieve real-time performance (Vink et al., 2020) developed a selection algorithm comprising three stages. The number of selected aircraft involved in each stage is limited in order to speed up the solving process. If no solution is found by the selected aircraft, the set of candidate aircraft is expanded and moved to the next stage. Based on historic data from the Reporting Carrier On-Time Performance database[6], probability distributions of different disruption events (e.g. flight delay, aircraft unavailability) were derived. Using these distributions, 370 disruption events were randomly generated and solved. Each disruption event contains one or more disruptions. In total these disruption events contain 565 individual flight disruptions. For 96% of the 370 scenarios, the Selection Algorithm Solution (SA)[7] had the same cost as the Dynamic Global Solution (DG)[8]. In 88% of the 370 events,

[6]Published by the United States Department of Transportation

[7]The best solution found by selection algorithm presented in this work, after considering all the selections of aircraft candidate to solve the disruption events

[8]The solution when the entire fleet of candidate aircraft is used. Determining this solution takes too long to be suitable for real-time use. However, this solution corresponds to the global optimum solution to the disruption problem, as defined in this work. This is used as the reference solution in the analysis of the results.

the SA found the trivial solution[9]. There are a few outliers for which the solution found by SA is more expensive. In five of the 370 events the SA found a solution which was more than twice the cost found by the DG solution. Such cases are rare (1.35% of the cases) and they occur when the DG uses a large combination of aircraft that are not considered in the same iteration by the selection algorithm.

2.3.5 Heuristics in Aircraft Recovery

The research led by (Arguello et al., 1997) used a greedy randomised adaptive search procedure (GRASP) meta-heuristic to minimise flight cancellation and delay costs associated with a recovery aircraft routing, as a response to groundings and delays. The GRASP described in the study is adapted for use as a randomised neighbourhood search technique. In the procedure, the neighbours of an addressed solution are evaluated, and the most desirable are placed on a restructured candidate list. Neighbour generation operations are performed on pairs of aircraft routes. The authors introduced a series of constraints to enforce the model to share operational practices namely that, every flight in each aircraft route must depart from the airport where its previous flight arrived (balance constraint), a minimum turn-round time must be enforced between each flight arrival and subsequent departure, the recovery period extends to the end of the current day, aircraft must be positioned at the end of the recovery period in a specific airport so that the flight schedule can be resumed next day, airport departures are restricted in the curfew period and aircraft with planned maintenance will not have their original route altered. The approach was tested on data supplied by *Continental Airlines* and the results proved that the GRASP could produce in most cases optimal or near-optimal solutions.

(Løve et al., 2001) defined the Dedicated Aircraft Recovery Problem (DARP) as *given an original flight schedule and one or more disruptions, the DARP consists of changing the flight assignments of the aircraft to produce a feasible and more preferable revised flight schedule*. The authors compared the results obtained using various versions of a heuristic, namely the Iterated Local Search (ILS) with Variable Neighbourhood Search (VNS) incorporated and, a Steepest Ascent Local Search (SALS) along with a Repeated Steepest Ascent Local Search (RSALS). The latter performs entirely like SALS but is repeated for different initial conditions. According to the authors, ILS algorithms were able to find solutions within the first 10 seconds, which proved to be robust. After 24-hour test runs, significantly better solutions were not found. As for the SALS the authors mention that the algorithm quickly finds a local optimum and that once reached it is not easy to escape from. The latter proved not to be a drawback when the local optimum is close to the value of the global optimum. However, the further the distance to the best solution ever found, the worse the objective values get. This correlation provides a strong indication that the solution space has a profile with a plateau, like that illustrated in figure 2.3. The authors conclude that many local optima are close to the global optimum since the SALS was able to increase the solutions' revenue when revising the flight schedules of all problem instances.

Since the results achieved were very auspicious, British Airways (BA), provided some of their flight data for realistic testing of the RSALS heuristic. After a few simplifications, 10 BA flight schedules were extracted, which had an average of 80 active aircraft, 44 airports, and 340 flights. Each flight schedule was disrupted and afterwards the RSALS heuristic was used to improve the flight schedules by delaying flights, swapping aircraft, and cancelling flights. The model proved to deliver well recovered flight schedules, by

[9]The solution which is found taking only the disrupted aircraft in consideration. This solution resembles the immediate solution that a controller would find during operations and it is the first solution obtained by the selection algorithm. It is found in about two to three seconds, because the resulting problem involves few aircraft and is therefore small.

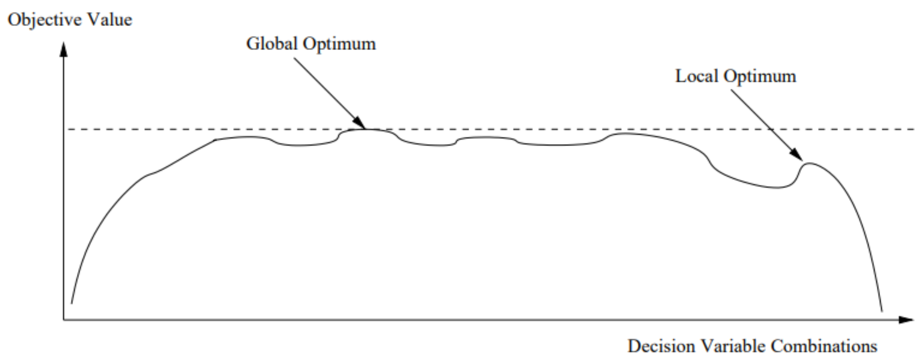


Fig. 2.3 Connection network proposed by (Løve et al., 2001)

solving a SALS heuristic using the subjacent network representation of the problem. The authors conclude that further advancements should be performed to include crew scheduling and passenger itineraries. As for the initial claim made by the authors that the time for a tool to find solutions for such problem, should be less than three minutes, (Andersson, 2006) refers that this benchmark varies depending on the envelopment of the problem, and also on quality of the solutions delivered. In (Andersson, 2006) Tabu Search (TS) and Simulated Annealing (SA) methods were used to solve the ARP. TS is a metaheuristic local search method used for mathematical optimisation. Local search methods have the tendency to be stuck in suboptimal regions. TS enhances the performance of these techniques by prohibiting already visited solutions or others through user-provided rules. After evaluating the quality of the solutions and the robustness of the method TS proved to be the preferable solution strategy.

2.3.6 Constraint Programming

This section finalises introducing the application of CP in the ARP. The work of (Guimarans et al., 2013) aims at minimising the losses, caused by the uncertainty external factors, in aircraft schedules, Stochastic Aircraft Recovery Problem (SARP). This work uses CP and simulation to solve the SARP. The method solves the problem through the rescheduling of the flight plan using delays and swaps. The main objective is to restore as much as possible the original flight schedule, minimising the total delay. This method is not tested using real data scenarios and the authors mention that it may be extended to tackle more complex variants of the problem that can include crew scheduling. The formulation proposed by (Guimarans et al., 2015) for the ARP is based on the CP. The authors based the optimisation approach on the Large Neighbourhood Search (LNS) metaheuristic, combined with simulation at different stages in order to ensure solutions' robustness. The method is tested on a set of instances with different characteristics, including some instances originating from real data provided by a Spanish airline.

2.4 Integrated Airline Recovery

When scheduling flights, passenger travel demand is clearly a key consideration for commercial airlines. The efficient flow of passengers is critical, especially when regular operations are affected by disruptions, however in practice, passenger recovery is observed as the last stage of the sequential recovery approach. Due to the high dependency of passenger schedules on aircraft and crew schedules, passenger recovery is either integrated with aircraft recovery or with both aircraft and crew recovery. This section describes different approaches in literature to solve the integrated recovery problem. Subsection 2.4.1 focuses on integrated recovery of aircraft and passengers while subsection 2.4.2 covers relevant literature in integrated aircraft, crew and passenger recovery.

2.4.1 Integrated Aircraft and Passenger Recovery

In the work of (Zhang and Hansen, 2008) the authors present a model for a hub-and-spoke network that uses various modes of transportation to accommodate passengers whose travel plans have been perturbed. This strategy, referred to as real-time intermodal substitution (RTIMS), used mathematical programming to help airlines decide how to delay, cancel or substitute flights with buses. A numerical scenario for a four-hour recovery period consisting of 40 flights and 736 passengers was evaluated. After substituting transport, results showed a massive decrease in cost since the number of disrupted passengers dropped. However, their model rescheduled only flights that would arrive or depart from the affected airports, and it did not consider the downstream effect of the delayed or cancelled flights.

The mathematical model proposed by (Jafari and Zegordi, 2011), simultaneously recovered airline schedules by recovering disrupted aircraft and passengers. The authors proposed a formulation of recovery by considering flight re-timing, aircraft swapping, ferry fight, reserved aircraft and flight cancellations. The overall objective of the model was to reduce costs associated with flight cancellation, aircraft recovery, and passenger disruption related costs. A data set consisting of two disruption scenarios was used to evaluate the model. The data set contained 13 aircraft divided into 2 fleet, 100 flights, 19 airports and 2236 passengers with 8 itineraries and 55 connections. The authors concluded that the recovery procedure will, not only, directly affect the passengers on that particular flight, but it may also, indirectly, affect the passengers on the next flight in the route for the aircraft in question, and claim their model gives a solution for this situation. Another important conclusion made by the authors is that using the aircraft rotation instead of flights helps to limit scope of disruption and is useful to recover the schedule efficiently.

The model proposed by (Bisaillon et al., 2011) consists of a very efficient large neighbourhood (LNS) search containing three phases, construction, repair and improvement. Figure 2.4 demonstrates a summary of the three phases the model presented.

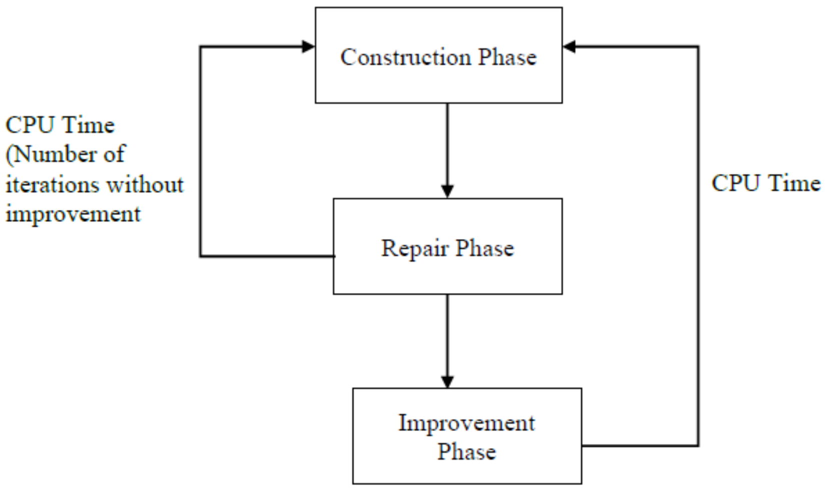


Fig. 2.4 LNS model proposed by (Bisaillon et al., 2011)

The construction phase consists of constructive heuristic that tries to create feasible solutions, by removing flight sequences until all constraints, with the possible exception of airport capacity constraints, are satisfied. The repair phase makes use of a heuristic that proceeds in three steps. First, by delaying flights to respect all airport capacity constraints. Second, by re-inserting flights that were removed during the construction phase, between two successive flights whose time interval is long enough to accommodate them. The authors argument that this approach yields expressive cost improvements in few iterations. Third, by accommodating passengers whose itineraries have been cancelled by repeatedly solving shortest path problems for a network of flights. The source node represents the origin airport of the itinerary at the departure time and the sink node represents the destination airport of the itinerary at the arrival time. This process attempts to assign to the path the largest number of passengers satisfying the aircraft seating capacity and is iterated if new passenger can be accommodated. Although the solutions obtained are feasible, they possibly are sub-optimal since many itineraries may have been cancelled. The improvement phase consists of a procedure that attempts to extend the repair phase by delaying some flights in the hope of accommodating additional passengers. Just like in the repair phase passengers are re-assigned by repeatedly solving shortest path problems. The global process is iterated introducing diversification in the construction phase, by randomly sorting the aircraft to treat them in a different order each time the construction phase is performed. Finally, whenever improved cost is found, the corresponding solution replaces the current one. On the overall, the algorithm executes a very large number of simple and fast actions. By doing so, not only it finds quickly feasible solutions, but also does not rely on any knowledge of the current flight network.

Considering more flights, fleets, and nodes will increase exponentially the complexity of the integrated aircraft and passenger recovery problem, leading to intractable models for exact methods. The development of math-heuristics has proven reduce the computational time in such complex models. (Mansi et al., 2012) presented a model that combines a math-heuristic with an oscillation heuristic to improve the solutions obtained. The math-heuristic consists of mixed integer programs that intends to maximise the number of aircraft that satisfy the maintenance constraint and the number of passengers that arrive at their final destination while minimising the total delay. However, this procedure may not render a feasible solution, meaning that at least one aircraft will not be able to reach the airport assigned for maintenance on time without cancelling flights. To make the solution feasible, a repair heuristic is applied to try to satisfy hard maintenance constraints, and also to maximise the number of passengers arriving at destination. After generating a feasible solution, the algorithm makes improvements alternating between constructive and destructive phases, by adding or removing some parts of the aircraft routes.

The LNS proposed by (Bisaillon et al., 2011), was later on improved by (Sinclair et al., 2014) where the authors included additional steps in each phase to make it more time efficient and cost effective. These improvements offered a better understanding of the relation between the cost of delay and the cost of cancelling a flight. Figure 2.5 illustrates the improvements add to the LNS proposed by (Bisaillon et al., 2011).

More recently, the work of (Sun et al., 2021) extended the work of (Zhang and Hansen, 2008) by implementing the proposed research approach to a real-time intermodal network. The model modified the traditional time-band network used for representing the airline recovery problem so that many redundant flight arcs and infeasible recovery flight arcs can be eliminated. This procedure reduces the size of the mathematical model and feasible solution space without compromising optimality.In relation to passenger recovery, the authors proposed a method for generating passenger candidate itineraries for reassigning

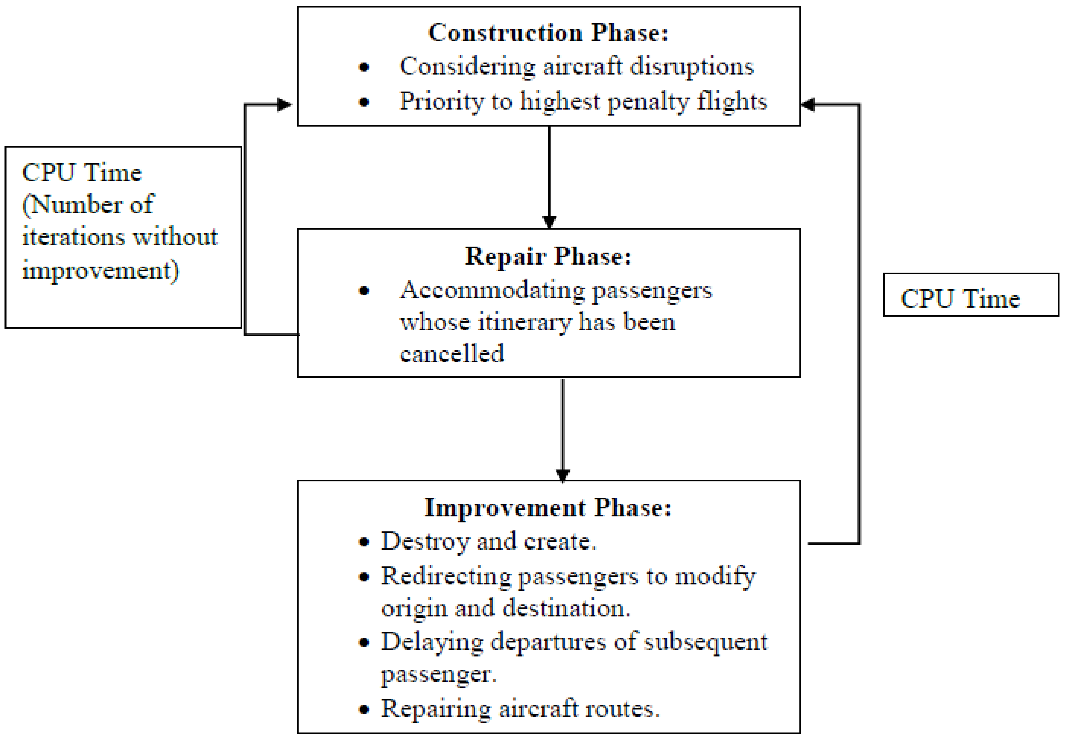

Construction Phase:
• Considering aircraft disruptions
• Priority to highest penalty flights
CPU Time (Number of iterations without improvement)
Repair Phase:
• Accommodating passengers whose itinerary has been cancelled
CPU Time
Improvement Phase:
• Destroy and create.
• Redirecting passengers to modify origin and destination.
• Delaying departures of subsequent passenger.
• Repairing aircraft routes.

Fig. 2.5 Improved LNS heuristic for an integrated aircraft and passenger recovery problem (source (Sinclair et al., 2014))

disrupted passengers. The authors expanded the time-band network into an intermodal network with both air and ground transportation modes and compared the disruption management performance with and without ground transportation modes.

2.4.2 Integrated Aircraft, Crew and Passenger Recovery

In (Bratu and Barnhart, 2006) the authors focus on passenger recovery while incorporating rules and regulations on aircraft and crew. They propose models for integrated recovery, using a flight schedule network representing flight legs with flight arcs. Several copies of the flight arcs are made to account for each possible departure time decision within the window of feasible departures for a specific flight. The objective either minimises the sum of operating costs and disrupted passenger costs, or the sum of operating costs and total passenger delay costs. To test the models the authors developed an Airline Control Center Simulator, to simulate domestic operations of a major US airline. The airline's data consisted of 83,869 passengers on 9,925 different passengers' itineraries per day, operated by 302 aircraft divided into 4 fleets, 74 airports and 3 hubs. For all scenarios solutions are generated that resulted in reductions in passenger delay and disruptions.

2.5 Modelling Flight Speed Changes and Consumed Fuel

The concept of speed change has already been studied in several publication whether to minimise potential conflicts by reducing flight speed and absorb holding delays, or to speed up the flight to recover from flight delays.

If it is possible to reduce flight speed, aircraft can leave the airport earlier, therefore helping the management of congestion at a surface level in the airport of origin. Another advantage will be the psychological effect on passengers, who will spend less time waiting at the airport of origin or even inside the aircraft while waiting at the departure gate. The work of (Delgado and Prats, 2012) proposes an en route speed reduction to complement current ground delay practices aimed at absorbing part of the air traffic flow management delays. Instead of performing the ground delays at the airport of origin the authors demonstrate that by flying slower, flight times are increased and can absorb holding delays at the destination airport, with no extra fuel cost for the airline.

Accurate estimation of the fuel consumed during aircraft operation is key for determining the fuel load, reducing the airline operating cost, and mitigating environmental impacts. In the work of (Patrón et al., 2015) the authors describe an algorithm to be implemented in an flight management system to create optimal flight trajectories and reduce fuel burn by analysing CCD phases. A complete wind model is used to calculate a more accurate assessment of the aircraft fuel burn, as well as to analyse the influence of the winds during a flight. The algorithm's objective is to obtain the maximum reduction in the flight cost, but it did not consider air traffic management constraints. The flights simulated to verify the algorithm's optimisation capabilities at reducing fuel consumption were performed using real flight information obtained from FlightAware™. The CI was set to zero, indicating that the only parameter to consider was the fuel consumed and not the flight time. The tests simulated flights from Lisbon to Toronto and London to Toronto. The results from the tests performed show an average flight cost reduction of 5.92%, and an average flight time reduction of 2.57%.

To study the correlation between flight time and fuel consumption and cost index selection on flight management systems, the work of (Wickramasinghe, 2015) proposes an optimisation model that introduces a performance index through DP. The performance index is built by considering the minimum cost with the trade-off between fuel consumption and flight time. The objective of the model consists in creating fuel-optimal-only trajectories and fuel optimal trajectories with arrival time constraints, as a function of the CI. The study was conducted in Japan and a series of flight data was measured by a commercial Global Positioning System (GPS) receiver, to perform trajectory optimisation and discuss the influence of operational procedures in the current system towards the selection of CI setting. The model simulates scenarios for 4D trajectory optimised for both aircraft's lateral and vertical profiles and 3D trajectory optimisation for the aircraft's vertical profile with a fixed trajectory. The authors claim that the airline flight procedures were time-oriented and airline companies select a CI which could be considered as not optimal in means of fuel saving strategies. Several optimal flights tend to select the flight path almost identical to Y20 RNAV[10] route, although the large variations in jet stream winds position over Japan's airspace suggest that, air route setting should consider seasonal wind conditions to increase the efficiency in flight operations. The authors claim that there is room for improvement in descent operations for fuel saving and that with newer aircraft, capable of performing at high altitudes, selecting higher cruising altitudes to avoid strong headwinds could reduce fuel consumption.

Although flight time management is a recovery strategy to deal with the disruptions, in practice its benefit has been limited because it does not consider network wide integrated effects, (Aktürk et al., 2014). Instead of simply delaying flight the work of (Aktürk et al., 2014) propose a flight rescheduling model

[10]Area navigation (RNAV) is a method of navigation that permits aircraft operation on any desired flight path within the coverage of ground- or space-based navigation aids, or within the limits of the capability of self-contained aids, or a combination of these

that includes adjusting cruise stage speed on a set of affected and unaffected flights as well as swapping aircraft optimally. The authors claim that their work successfully integrates cruise speed control action in the recovery model. In addition to the additional fuel cost of speeding up flights, the authors manage to integrate environmental costs and constraints. The authors report that cruise speed control can provide significant cost savings. One of the major contributions of (Aktürk et al., 2014) is enabling the use of a realistic fuel cost function based on the fuel flow model developed by BADA. This approach focuses only aircraft schedules however, disruptions can have severe effects on existing aircraft rotations, crew scheduling, and passenger itineraries. To overcome this shortcoming (Arikan et al., 2016), developed a flight network based representation for the integrated airline recovery problem that includes the flow of each aircraft, crew member, and passenger. This model allows common recovery decisions such as departure delays, aircraft/crew rerouting and passenger re-accommodation. Additionally, the authors implemented flight time management as a result of aircraft cruise speed changes. The authors in their conclusions claim that speeding up flights may be ticket cancellations, and flight cancellations. beneficial to help mitigate delays and preserve passenger connections in cases of disruptions. However, speeding up a flight increases fuel consumption, therefore an additional fuel cost is incurred. Since the relation between fuel consumption and aircraft speed is non-linear the formulation uses second order cone programming. With this representation the problem is solved with a commercial mixed integer programming solver, within reasonable computing time.

2.6 Literature Review Summary

The literature review versed about plethora of topics regarding commercial aviation. The main focus concerns flight planning models, CP and disruption recovery algorithms. Commercial aviation modelling is highly data driven, most of which is proprietary, hence becoming extremely difficult to come over. BADA is a very comprehensive data source and proves be quite useful for modelling ground movements and flight planning.

CP has proved to be an efficient approach to tackle management problems in commercial aviation such as tail assignment, crew scheduling, airport runway management and is also gaining room in ARPs. However, the CP papers that were reviewed do not provide details regarding CPr or backtracking techniques to reduce search space or tackle infeasible solutions.

In relation to the ARP the literature review covered an extensive set of recovery methods and strategies such as departure holding flight cancellation, aircraft rerouting, aircraft ferrying, and the use of spare aircraft. It was also verified that time management, namely the introduction of flight delays, proves to be an alternative for disruption recovery however, the latter often involves propagating delays in downstream flights.

The integrated recovery problem is highly complex and solution strategies can include crew rerouting, crew deadheading, passenger ticket cancellation, passenger reallocation. A real-time solution requirement for this problem is challenging when dealing with large networks.

Finally, It is possible to conclude from the literature review that aircraft speed can have a practical effect in the entire block time of a flight. Time controllability can be used as strategy for disruption recovery in commercial aviation. Nonetheless, the specific conditions and scenarios are not addressed in detail. One should ask what is the point of speeding up a flight if there is no airport arrival capacity, to allow the airport to land in the desired time slot.

Chapter 3

The Block Time and Fuel Model

3.1 Introduction

Block time consists of the total amount of time a flight takes, from pushing back from the departure gate, to arriving at the destination gate. This chapter defines the various phases of the flight plan and matches them to the data provided by EMEP / EEA and BADA. For each of the flight phases the model calculates the time they take, ground distance covered and consumed fuel. Finally, the model is validated using real airline data, and exploratory data analysis. To the best of our knowledge, we are the first to provide such computational results from departure to arrival gates for an extensive list of flights.

The remainder of this chapter is organised as follows. In Section 3.2 we will review the most important concepts regarding the flight phases. Section 3.3 makes the integration of the EMEP/EEA emissions data set with BADA aircraft performance data tables to derive the BTF model and deliver reliable block time, and consumed fuel. Section 3.4 describes minutely the results, and compares them against the ROADEF 2009 Challenge data set, the published literature and with real flight plans. Finally, Section 3.5, performs the appraisal of the results, derives the conclusions and foresees future work.

3.2 Flight Phases

This section defines the flight phases that make part of a flight. The phases have different rate of fuel consumed during, since the aircraft's engine thrust is also different for each of them. The BTF model considers that the flight begins at the time the aircraft is ready to move with the purpose of flight and continues until such time it comes to rest at the end of the flight and the primary propulsion system is shut down.

Each aircraft has a flight plan consisting of a document that can include, among others, information regarding the:

- **Aircraft** such as Fuel On Board (FOB), TOW.
- **Flight** such as origin and destination airports, flight type, expected departure and arrival time, cruising speed, maximum expected altitude, chosen route and an alternate route.
- **Weather** such as the forecast during the flight.

All this information will help the cockpit crew to pilot the aircraft safely during the flight phases. Block time encompasses all the phases of the flight and typically, a flight has the following phases:

- **Taxi-out** is the controlled movement of an aircraft on the ground, under its own power, between its parking area and the point of the runway from which its taking-off operations will occur.
- **Take-off** is the phase of flight in which an aircraft moves from the runway to flying in the air.
- **Climb** is the phase of flight during which the aircraft ascents to a predetermined cruising altitude after take-off. Although a single climb phase is typical, multiple step climb phases may also occur.
- **Cruise** occurs between the climb and descent phases and is usually the longest part of a journey. It ends as the aircraft approaches its destination and the descent phase of the flight commences in preparation for landing. During the cruise phase, because of operational or Air Traffic Control (ATC) reasons, aircraft may climb or descend from one FL to a higher or lower FL. During very long flights, aircraft are able to fly higher as the weight of the fuel aboard decreases. Usually, pilots ask ATC to allow them to fly at the optimum FL for the aircraft they are operating. This optimum FL is dependent on, for example, the type of aircraft, its operating weight and the length of the flight. ATC generally accepts this request if it does not jeopardise safety. For most commercial passenger aircraft, the cruise phase of a flight consumes the majority of the fuel.
- **Descent** is the phase of flight during which the aircraft decreases its altitude in preparation for landing and is the opposite of the climb phase. Similarly to the climb, descent can be continuous or stepped as a consequence of operational or ATC decisions; continuous descent is the most fuel-efficient option.
- **Final approach** is the last leg of an aircraft's approach to landing, when the aircraft is in line with the runway and descending for landing.
- **Landing** is the part of a flight when an aircraft returns to the ground up to the point at which taxi-in starts.
- **Taxi-in** is the movement of an aircraft on the ground, under its own power, which occurs from the point that the aircraft turns off the landing runway (after returning to normal taxi speed) to the point at which it parks on the ground and shuts down its engines.

The vertical profile of the flight, also known as the aircraft altitude profile, consists of three main phases: climb, cruise and descent (CCD). Commercial aircraft, regardless of the route they are taking, usually cruise at an altitude between 30,000 feet and 41,000 feet (FL300 and FL410 respectively) above sea level. There are several reasons for choosing altitudes in this range such as better fuel economy and passenger comfort. Within this altitude range, the aircraft encounters less resistance to travel, which makes the engine's thrust lower with increasing altitude, hence saving even more fuel. It would be logical to imagine that if aircraft were to fly even higher, the economy of consumption would be greater. The problem is that from a certain altitude, which varies with each aircraft model, they reach the so-called service ceiling (around 41,000 feet of altitude). The definition of the service ceiling is the height above sea level at which an aircraft is unable to climb at a rate greater than 100 feet per minute. With altitude, the atmosphere gets less dense, with fewer oxygen molecules per volume of air, therefore the aircraft's engine will produce less and less power. A service ceiling is not really an absolute limit on the altitude that a particular design can achieve, but one at which the aircraft begins to run out of climb capability.

3.3 The Block Time Fuel model

This section demonstrates how to model the block time and fuel consumed, using Newtonian Mechanics and the public available data sets EMEP/EEA and BADA. The BTF model comprises the following three subsections:

- **Subsection 3.3.1** demonstrates how to calculate the ground distance and bearing between an origin and destination. The BTF model uses these two results to define the cruising altitude.
- **Subsection 3.3.2** demonstrates, given the aircraft model, how to use the EMEP/EEA data set to determine the duration, fuel flow and fuel consumed for taxi-out, take-off, approach, landing and taxi-in phases.
- **Subsection 3.3.3** describes how to use BADA's aircraft performance data table to determine for each time instant of a flight the fuel flow and fuel consumed for CCD phases.

3.3.1 Calculating Ground Distance and Defining Cruise Altitude

To calculate the ground distance between the origin and the destination airport it is necessary to have beforehand:

ϕ_o latitude for the origin airport
λ_o longitude for the origin airport
ϕ_d latitude for the destination airport
λ_d longitude for the destination airport

The latitude difference is:

$$\Delta\phi = \phi_d - \phi_o \tag{3.1}$$

and the longitude difference is:

$$\Delta\lambda = \lambda_d - \lambda_o \tag{3.2}$$

using the auxiliary calculation:

$$a = sin^2\left(\frac{\Delta\phi}{2}\right) + cos(\phi_o) \times cos(\phi_d) \times sin^2\left(\frac{\Delta\lambda}{2}\right) \tag{3.3}$$

Assuming that Earth is perfect sphere with a radius R of 6378.137 kilometres, it is possible to determine the physical distance d between the origin and destination airports using the Haversine formula in 3.4:

$$d = 2 \times R \times atan\left(\frac{\sqrt{a}}{\sqrt{1-a}}\right) \tag{3.4}$$

To avoid collisions between aircraft travelling in opposite directions it is necessary to impose a vertical separation. The ICAO[1] semi-circular rule defines the available flight levels in the conventional airspace and also in the Reduced Vertical Separation Minima (RVSM) airspace when applicable between FL290 and FL410. The default worldwide semi-circular rule, can be observed in figure 3.1 and is applied according the aircraft's magnetic bearing: if this value is between 0° and 179° (eastbound flights), FL or altitude must be odd FL310, FL330, FL350, etc, whereas if the aircraft has a magnetic bearing between 180° and 359° (westbound flights), the FL must be even FL320, FL340, FL360, etc.

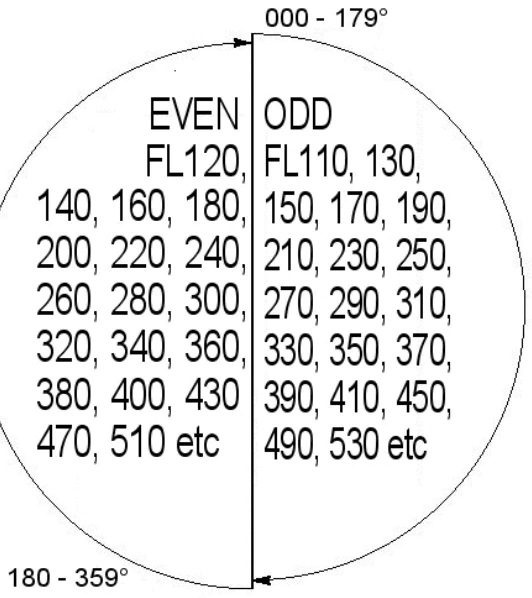


Fig. 3.1 ICAO Cruising Levels (RVSM). Source: ICAO

The cruise altitude (flight level) depends on the ground distance (Pagoni and Psaraki-Kalouptsidi, 2017), table 3.1. (Pagoni and Psaraki-Kalouptsidi, 2017) noticed that short-haul flights present significant deviations in cruise altitude even for shorter distances. On the other hand, longer flights tend to have a more homogeneous flight performance. Their study adopts a distance increment of 250 statute miles [2] (sm) to present the data.

The BTF model uses an extended flight distance profile. The one used by (Pagoni and Psaraki-Kalouptsidi, 2017), from 500 sm to 2500 sm, covers a wide spectrum of US domestic flight distances. Indeed, those flights comprise 87.4% of total passenger miles in 2012.

[1] ICAO is a United Nation's specialised agency, established by states in 1944 to manage the administration and governance of the Convention on International Civil Aviation (Chicago Convention).

[2] The US statute mile, also called a survey mile, measures 1609.3472 meters, a difference of 3.2 millimetres (1/8 inch) per mile. This is due to a usage of the equation of a survey foot equalling 1,200/3,937 meters rather than 30.48 centimetres.

Table 3.1 Flight level for westbound and eastbound flights during cruise phase

	Westbound flights	Eastbound flights
Distance range [sm]	FL	FL
310 - 500	340	330
500 -750	360	350
750 - 2500	380	370

3.3.2 Modelling Taxi-out, Take-off, Approach, Landing and Taxi-in Phases

For taxi-out, take-off, approach, landing and taxi-in phases the BTF model uses the data set provided by the EMEP/ EEA air pollutant emission inventory guidebook 2016 1.A.3.a Aviation – Annex 5 – LTO emissions calculator 2016. The fuel burnt and emission data provided in this set are for supporting the European Union (EU) and the member states of the EEA in the maintenance and provision of European and national emission inventories. Fuel burn and emission data in this spreadsheet are modelled estimates and not absolute values. Where only one type of engine is associated with a particular aircraft type, it is the most common type of engine (as seen in Europe), or the best equivalent type of engine, for that aircraft type. Where several types of engines are associated with a particular aircraft type, the most-common type of engine is highlighted.

The first part of our modelling will use the EMEP/EEA data set and consists of determining the duration and fuel flow for the taxi-out and take-off phases, based on the type of the aircraft, origin airport and year. The mass of fuel burnt M_{out} during the taxi-out phase is calculated using equation 3.5:

$$M_{out} = T_{out} \times E_7 \times N_e \tag{3.5}$$

T_{out} is the taxi-out time [s], E_7 is the rate of fuel burn [kg/s/engine] during taxi-out and N_e is the number of engines of the particular aircraft performing the flight. EMEP/EEA assumes that during the taxi-out phase the engine thrust is set to 7% following the ICAO convention. Similarly, the mass of fuel burnt M_{off} during take-off is calculated using equation 3.6:

$$M_{off} = T_{off} \times E_{100} \times N_e \tag{3.6}$$

where T_{off} is the take-off time [s], E_{100} is the rate of fuel burn [kg/s/engine] during take-off. In this phase EMEP/EEA assumes that during the take-off phase the engine thrust is set to 100% following the ICAO convention.

The mass of fuel burnt M_{al} during the approach and landing phase is calculated using equation 3.7:

$$M_{al} = T_{al} \times E_{30} \times N_e \tag{3.7}$$

T_{al} is the approach and landing time [s], E_{30} is the rate of fuel burn [kg/s/engine] during approach and landing. In this phase EMEP/EEA assumes that during the approach and landing phase the engine thrust is set to 30% following the ICAO convention.
The mass of fuel burnt M_{in} during the taxi-in phase is calculated using equation 3.8:

$$M_{in} = T_{in} \times E_7 \times N_e \tag{3.8}$$

T_{in} is the taxi-in time [s], E_7 is the rate of fuel burn [kg/s/engine] during taxi-in and N_e is the number of engines of the particular aircraft performing the flight. EMEP/EEA assumes that during the taxi-in phase the engine thrust is set to 7% following the ICAO convention.

3.3.3 Modelling Climb Descent and Cruise Phases

To model the CCD phases the BTF model uses the BADA Performance Table Files (PTF) for each specific aircraft. In table 3.2 we present the performance data structure within the file.

Table 3.2 BADA performance data structure

Column name	Units
Flight level	[FL]
Cruise TAS (nominal mass)	[knots]
Cruise fuel consumption (low mass)	[kg/min]
Cruise fuel consumption (nominal mass)	[kg/min]
Cruise fuel consumption (high mass)	[kg/min]
Climb TAS (nominal mass)	[knots]
Rate of climb with reduced power (low mass)	[ft/min]
Rate of climb with reduced power (nominal mass)	[ft/min]
Rate of climb with reduced power (high mass)	[ft/min]
Climb fuel consumption (nominal mass)	[kg/min]
Descent TAS (nominal mass)	[knots]
Rate of descent (nominal mass)	[ft/min]
Descent fuel consumption (nominal mass)	[kg/min]

In order to model the the CCD phases the BTF model uses Newton's equations of motion:

$$v = a \times t + v_0 \tag{3.9}$$

$$r = r_0 + v_0 \times t + \frac{1}{2} \times a \times t^2 \tag{3.10}$$

$$r = r_0 + \frac{1}{2} \times (v + v_0) \times t \tag{3.11}$$

$$v^2 = v_0^2 + 2 \times a \times (r - r_0) \tag{3.12}$$

$$r = r_0 + v \times t - \frac{1}{2} \times a \times t^2 \tag{3.13}$$

Where r_0 is the initial position, r the final position, v_0 the initial velocity, v is the final velocity, a the acceleration and t is the time interval.
During the climb phase, the Rate of Climb and Descent (ROCD) and fuel flow, the BTF uses the nominal mass level. The reason that underlies this assumption is to prevent the fact that for high mass level there are several aircraft models that do not have any climb data available. During the climb phase, the model will interpolate between pairs of flight levels (current FL_c and next FL_n respectively), the pairs of values

for true airspeed (TAS_c, TAS_n), fuel flow (f_{cc}, f_{cn}), rate of climb and descent (R_c, R_n).
Figure 3.2 depicts the interpolation process during the climb phase between FL0 to FL5 and FL5 to FL10.

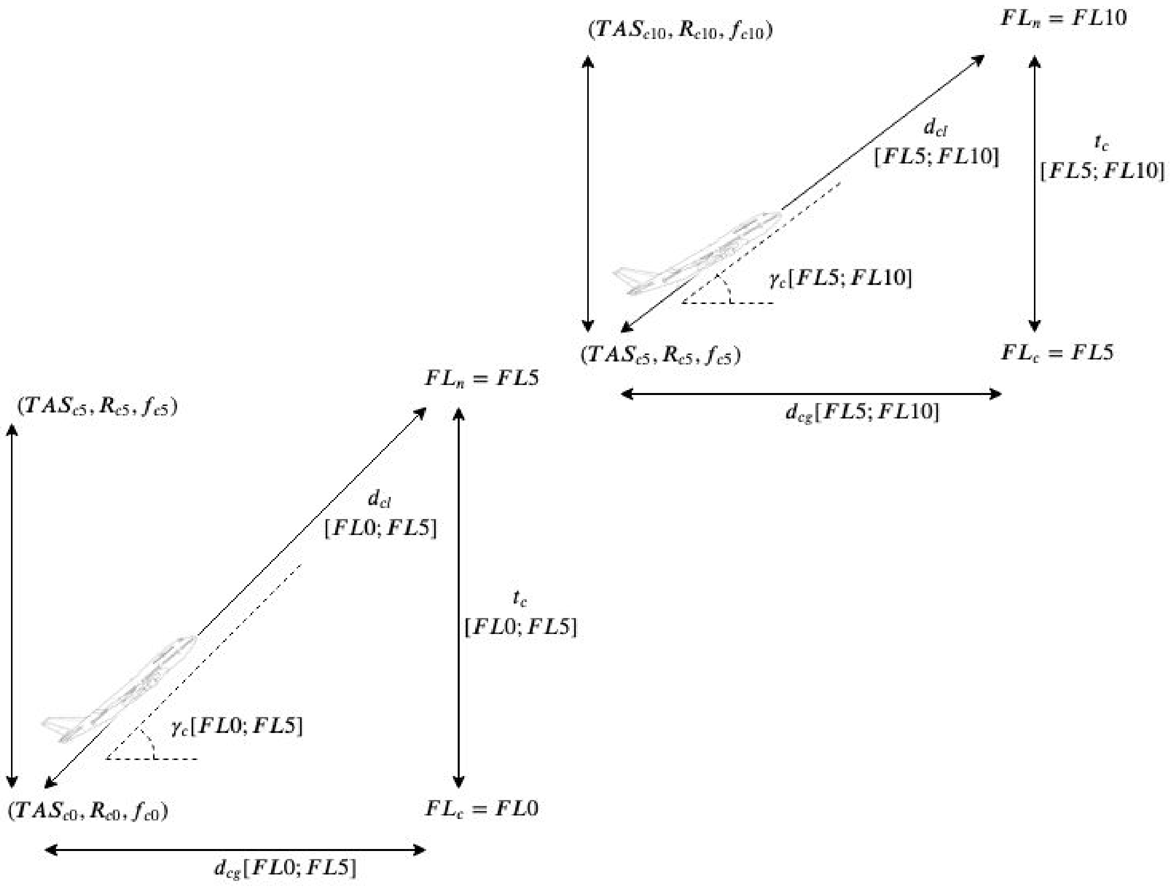


Fig. 3.2 Climb between flight levels FL0 to FL5 and FL5 to FL10

Since the ROCD consists of the variation of altitude with time, the BTF model considers it to be the aircraft's vertical speed component. Since the ROCD changes with altitude, it is necessary to compute its rate of change with time. The latter consists of the vertical component of the aircraft's acceleration a_{cv} and it is calculated by interpolating between the current (FL_c) and next (FL_n) flight level, the rate of climb and descent R_c and R_n, respectively. Based in equation 3.12 the interpolation equation is expressed as follows:

$$a_{cv} = \frac{1}{2} \times \frac{R_n^2 - R_c^2}{FL_n - FL_c} \tag{3.14}$$

After determining the aircraft's vertical acceleration the BTF model computes, based in equation 3.9, the time t_c the aircraft took to travel from FL_c to FL_n :

$$t_c = \frac{R_n - R_c}{a_{cv}} \tag{3.15}$$

The longitudinal component of the aircraft's acceleration a_{cl} is determined, based on equation 3.9, measuring the variation of TAS between the current and the next FL, TAS_c and TAS_n respectively:

$$a_{cl} = \frac{TAS_n - TAS_c}{t_c} \tag{3.16}$$

The longitudinal distance d_{cl} that the aircraft flew during the ascent between the current and next FL, can be calculated based on equation 3.11, and thus be obtained by:

$$d_{cl} = TAS_c \times t_c + \frac{1}{2} \times a_{cl} \times t_c^2 \tag{3.17}$$

The aircraft's trajectory angle γ_c is obtained by:

$$\gamma_c = arcsin\left(\frac{FL_n - FL_c}{d_{cl}}\right) \tag{3.18}$$

The ground distance d_{cg} is obtained projecting the longitudinal distance d_{cl} on the horizontal plane:

$$d_{cg} = d_{cl} \times cos(\gamma_c) \tag{3.19}$$

Finally, for the climb phase, the BTF model derives the amount of fuel consumed using a similar approach. It first calculates the variation of the fuel flow F_{cf} during the time interval t_c, according to:

$$F_{cf} = \frac{f_{cn} - fcc}{t_c} \tag{3.20}$$

where f_{cn} and f_{cc} are the fuel flow for the next and current flight levels. The total amount of fuel consumed C_c during the time interval t_c is obtained using:

$$C_c = f_{cc} \times t_c + \frac{1}{2} \times F_{cf} \times t_c^2 \tag{3.21}$$

During the climb phase, the interpolation procedure will terminate when the aircraft reaches the cruise FL. For some aircraft models there are no data points for the specific cruise FL, for instance for an A320 that will cruise at FL340 the PTF only has data points (TAS_c, R_c and f_c) for FL330 and FL350. Figure 3.3 depicts the interpolation between FL330 and FL340.

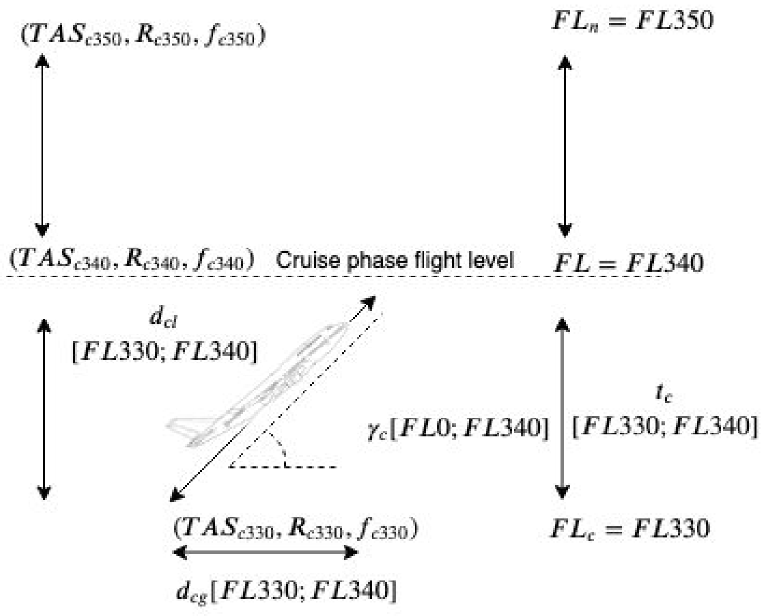

$(TAS_{c350}, R_{c350}, f_{c350})$
$FL_n = FL350$
$(TAS_{c340}, R_{c340}, f_{c340})$
Cruise phase flight level
$FL = FL340$
d_{cl} $[FL330; FL340]$
$\gamma_c[FL0; FL340]$
t_c $[FL330; FL340]$
$(TAS_{c330}, R_{c330}, f_{c330})$
$FL_c = FL330$
$d_{cg}[FL330; FL340]$

Fig. 3.3 Climb between flight levels FL330 to FL340

In the next part the BTF model will calculate for the descent phase the values of the vertical acceleration a_{dv}, time of descent t_d, longitudinal acceleration a_{dl}, longitudinal distance d_{dl}, aircraft's trajectory angle γ_d, ground distance d_{dg}, fuel flow variation F_{df} and total amount of consumed fuel C_d. The descent phase is symmetrical to the climb phase, thus in the descent phase the BTF model will interpolate between flight levels, starting at cruise altitude until 3000 feet (FL30), since from there on the BTF model will be using EMEP/EEA to compute the approach, landing and taxi-in phases. The reason for this procedure derives from the fact the EMEP/EEA data set aggregates in a single phase approach and landing.

Finally, for the cruise phase, the BTF model initially calculates the ground distance d_g that it is needed to be covered using:

$$d_g = d - (d_{cg} + d_{dg}) \tag{3.22}$$

To calculate the time t_{cr} that it takes to fly the ground distance the BTF model assumes that its value is the same as the one the aircraft will travel. The value for the TAS can be obtained directly from the PTF if the cruise FL is present, otherwise the BTF model interpolates between the two pairs, lower and upper FL and lower and upper TAS. Thus the cruise time t_{cr} is obtained by:

$$t_{cr} = \frac{d_g}{TAS} \tag{3.23}$$

Similarly, the value for the fuel flow (Φ) during the cruise phase can be obtained directly from the PTF if the cruise FL is present, otherwise the BTF model interpolates between the two pairs, lower and upper FL and lower and upper Φ. Thus, the consumed fuel F during the cruise phase is obtained by:

$$F = \Phi \times t_{cr} \tag{3.24}$$

With the final equation 3.24 the BTF model achieves a complete integration of all phases of a flight. The next section calculates for each of them the start and end time, consumed fuel, start and end altitude and ground distance.

3.4 Computational Results

In our numerical experiments, the BTF model computed for a discrete set of time instants the values for fuel flow, altitude, and ground distance. Table 3.3 describes the flight block, grouped in seven phases, for the flight between CDG in Paris and BCN, operated by a Airbus A320 aircraft.

For clarity purposes, the next subsections describe the previous flight in detail, namely in Subsection 3.4.1, the variation through time of the fuel flow. Since it was possible to obtain flight data from FlightAware™ Flight Track Log (AFR1148, 2019b), Subsection 3.4.2 compares the results for the altitude profile and in Subsection 3.4.3 ground distance versus the time. Subsection 3.4.4 presents the exploratory data analysis of the results using the distance data set from the ROADEF 2009 Challenge. Subsection 3.4.5 compares the results for the BTF model with those presented in the literature review. Finally, Subsection 3.4.6 compares the BTF results with those used by the flight planning software Lido™.

Table 3.3 Flight block

Phase	Start time [min]	End time [min]	Consumed fuel [Kg]	Start altitude [FL]	End altitude [FL]	Ground distance [Km]
Taxi-out	00:00	17:34	219.3	0	0	0
Take-off	17:34	18:16	95.1	0	0	0
Climb	18:16	40:20	1634.9	0	360	271.1
Cruise	40:20	68:43	1016.0	360	360	391.5
Descent	68:43	84:41	120.8	360	30	177.4
Approach and landing	84:41	88:41	149.8	30	0	19.4
Taxi-in	88:41	93:48	63.9	0	0	0
		Total	3299.6			859.4

3.4.1 Fuel flow vs. Time

Figure 3.4 depicts in detail, the pattern of each of the seven phases. From the initial instant to 17:34 it is possible to observe a flat fuel flow for the taxi-out phase, after which there is a sharp increase for 42 seconds corresponding to the take-off phase. After the aircraft takes off the fuel flow will start to decrease for 22 minutes and 6 seconds, until the aircraft reaches cruising altitude. When the aircraft reaches cruise altitude the fuel flow will first decrease instantly and then will become a flat line for 28 minutes and 23 seconds until the aircraft starts the descent phase. When the aircraft starts the descent phase the fuel flow again decreases abruptly and increases very slowly for the next 15 minutes and 58 seconds until the aircraft starts the approach and landing phase. During the approach and landing phase the fuel flow will increase instantaneously and maintain flat rate for 4 minutes. Finally, after landing, the aircraft will initiate the taxi-in phase. This phase will have a duration of 5 minutes and 7 seconds and the fuel flow will be constant. The fuel flow presented in the our model not only is consistent to the one presented by (Alam et al., 2009), but it also extends the work of these authors by including taxi-out, take-off and taxi-in phases.

3.4.2 Altitude vs. Time

Figure 3.5 compares the results for the altitude profile for the BTF model and the values retrieved from FlightAware™ Flight Track Log (AFR1148, 2019b). FlightAware™ provides the information for each flight; the aircraft's location and speed sampling occur at approximately each minute, providing an extensive flight profile. The end time value retrieved from FlightAware™ flight (AFR1148, 2019a), for the taxi-out phase is 11 minutes however there is no data regarding the duration for the take-off. It is possible to verify that the BTF model, takes 6 minutes and 34 seconds longer for the taxi-out phase.
As for the climb phase, the FlightAware™ flight ends 6 minutes and 55 seconds earlier and in terms of duration takes 22 minutes and 30 seconds whereas the BTF takes 22 minutes and 6 seconds. The FL configured for the BTF model to end the climb phase (and thus start the cruise phase) is FL360, however the FlightAware™ flight ends the climb phase at FL341.
In respect to the cruise phase, as already mentioned, the FlightAware™ FL is lower than the BTF model. Added to the latter the duration for FlightAware™ is 36 minutes and 36 seconds whereas the BTF model takes 28 minutes and 23 seconds. In conclusion, the cruise phase finishes 1 minute and 21 seconds earlier

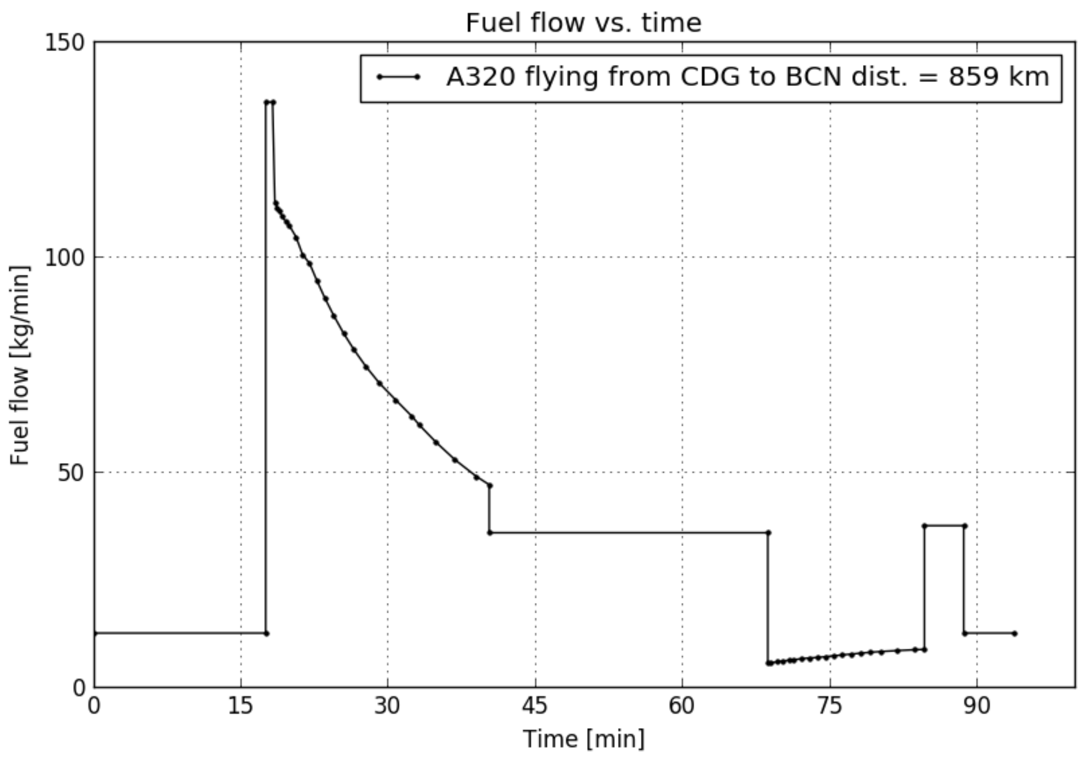


Fig. 3.4 A320 aircraft fuel flow vs. time during the flight CDG-BCN

than FlightAware™ .

As for the descent phase our model starts earlier and it is important to refer that since it is an exact representation of BADA's PTF file it will not fit perfectly the usual descent profile depicted by FlightAware™ data. In the descent phase, aircraft are piloted to glide as much as possible to save fuel, hence the descent phase for FlightAware™ shows clearly changes in the slope. It is also possible to observe from FlightAware™ data that, in the time interval from 89 to 94 minutes, the altitude remains constant at FL 57. This can be explained as an holding period for which the aircraft had to wait until it was given authorisation by the air traffic controllers to complete the descent phase and land safely.

For time comparison purposes the BTF model aggregates the descent and approach and landing phases. The duration for the BTF model is 12 minutes and 18 seconds less than FlightAware™ . As a consequence, the FlightAware™ flight lands 13 minutes and 5 seconds later that the BTF model.

Finally for the taxi-in phase, the BTF model takes 5 minutes and 7 seconds whereas FlightAware™ takes 3 minutes and 12 seconds.

On the overall the block time difference between the FlightAware™ flight and the BTF model is 11 minutes and 11 seconds.

3.4.3 Ground Distance vs. Time

In terms of ground distance covered, in figure 3.6 it is possible to observe, that it varies linearly with time soon after the aircraft take-off until the end of the descent phase at FL30. However, after taking off the aircraft in the BTF model flies faster than FlightAware™'s and overruns it at a distance of 400 kilometres and at instant 47 minutes.

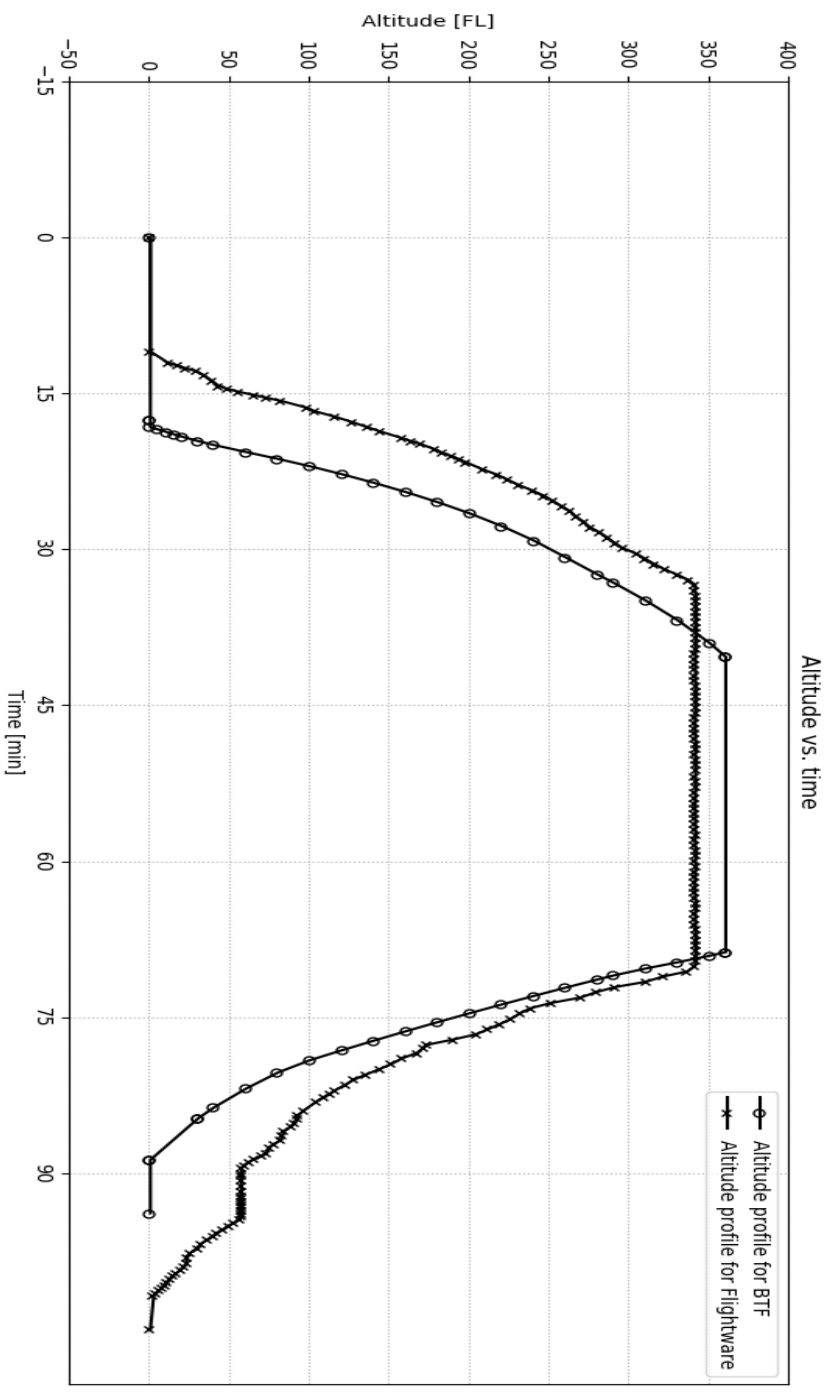


Fig. 3.5 A320 altitude profile during the flight CDG-BCN

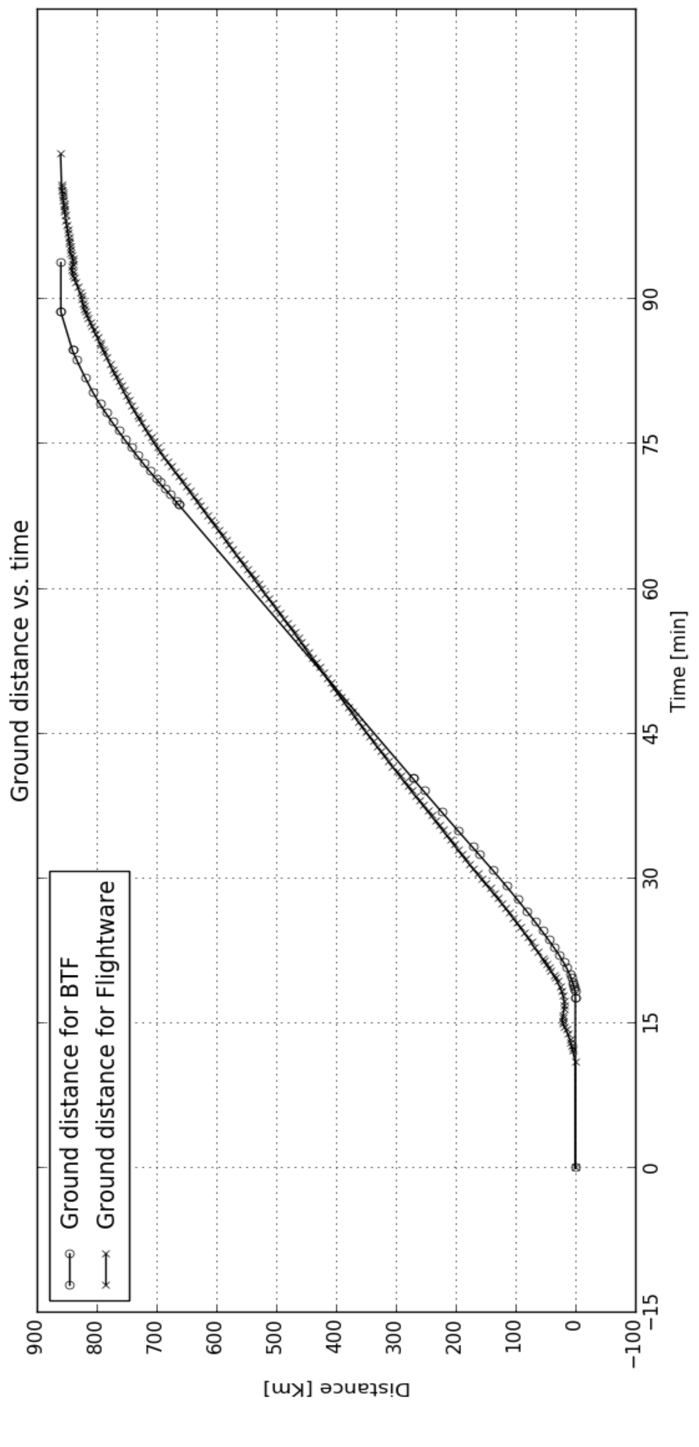


Fig. 3.6 A320 ground distance vs. time during the flight CDG-BCN

3.4.4 Benchmarking the BTF Model Results Against ROADEF 2009 Challenge

Every two years the French Society of Operational Research and Decision Making releases the ROADEF Challenge, which consists on a competition to solve a complex optimisation problem that occurs in industry. In the ROADEF 2009 Challenge there is a step-wise simplification of a model, for disruption management in commercial aviation that aims at finding recovery planning of flights, aircraft assignments and passengers (including flight leg cancellation) on a given maximal horizon, so that a sum of penalties corresponding to various costs or discomforts is minimised.
The ROADEF Challenge 2009 provides a data set with the duration of flights between an origin and a destination airport; however, these values do not have in consideration the aircraft model.
This section compares the BTF model results for block time with those supplied by the ROADEF 2009 Challenge (see table in Appendix A.1 for the extended result set). The sample has a total of 60 distinct flight tuples consisting of origin airport, aircraft model and destination airport and retrieved the real flight block time distribution from (Flightaware, 2019) between the 7^{th} and the 23^{rd} of July 2019.

The comparison methods that are used consist of percentile versus distance and root mean square error versus distance. Figure 3.7 illustrates the block time percentile of the BTF model and ROADEF versus the ground distance. This graph gives the percentage of block time retrieved from (Flightaware, 2019) that are below the block time values of the BTF model or the ROADEF 2009 Challenge. The number of percentile values less than 100 presented in the BTF model exceeds those presented by ROADEF 2009 Challenge, which leads us to conclude that our results for block time can be used as a lower bound for simulation or validation procedures.

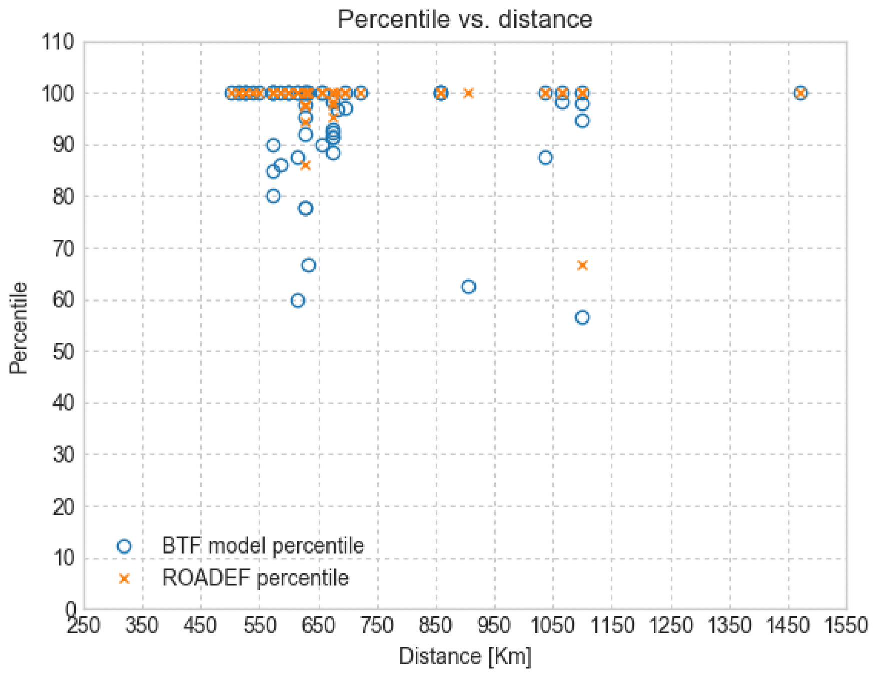


Fig. 3.7 Block time percentile for the BTF model and ROADEF, benchmarked to FlightAware™ vs. distance

Figure 3.8 illustrates RMSE for the BTF model and ROADEF 2009 Challenge versus the ground distance. The RMSE for each flight's block time is obtained using equation 3.25:

$$rmse_{if} = \sqrt{\frac{\sum_{n=1}^{k}(t_n - t_{if})^2}{k}}, i = 1,2, f = 1...60 \tag{3.25}$$

where k is the sample size for each flight f, t_n is the block time retrieved from (Flightaware, 2019) and t is the block time value either from the BTF model ($i = 1$) or the ROADEF 2009 Challenge ($i = 2$).

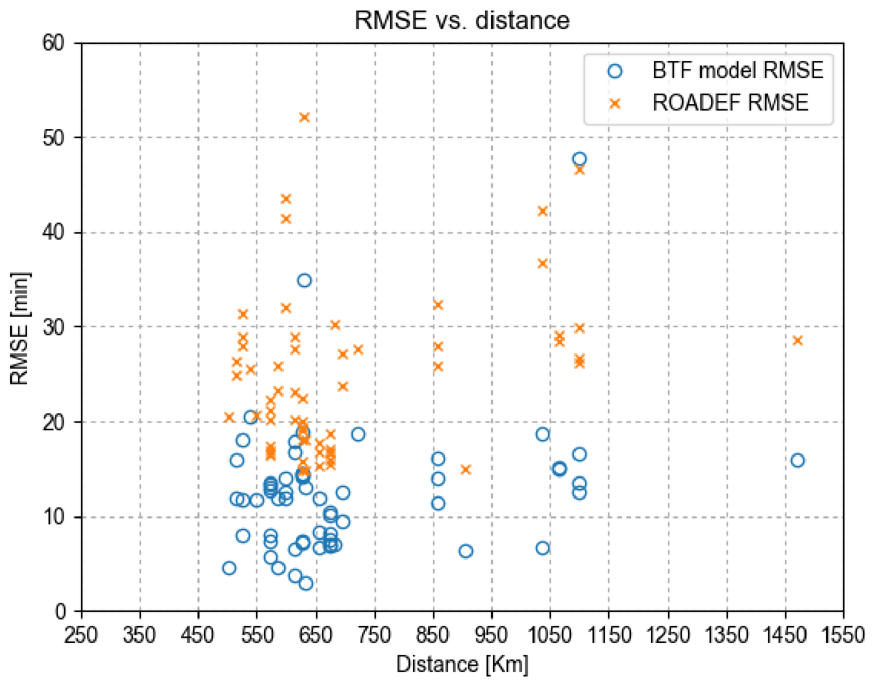


Fig. 3.8 Block time RMSE for the BTF model and ROADEF, benchmarked to FlightAware™ vs. distance

The RMSE for the BTF model and the ROADEF 2009 Challenge, confirms that the percentile results, in the sense that the BTF model presented a larger set of lower values than the one presented by ROADEF.

As for the results regarding consumed fuel it is not possible to make the full comparison between the BTF model and the ROADEF 2009 Challenge since it does not have this data.

3.4.5 Comparing the BTF Model Results and Literature Review

This section compares the results obtained using the BTF model and those published in the literature review. The BTF calculates the fuel consumed by an aircraft considering each flight phase encompassed in the flight, from the origin to the destination airport gates. The cruise FL is derived from the physical distance between the origin and destination airports.

The work of (Oruc and Baklacioglu, 2020) consists in determining a fuel flow function for the climb phase, for a Boeing 737-800. In respect to the latter, the BTF model uses the fuel flow data provided by BADA, in the performance table for the Boeing 737-800. The authors modelled the fuel function for five flights, and made the respective error analysis. Since it is not possible to have access to the TAS values the authors used to derive the fuel function, one cannot make the exact comparison between the values used for the BTF and those obtained using the CSA or the real ones. However, since the authors published the graphs of fuel flow versus altitude, one can make a qualitative comparison by superimposing the values used in the BTF model. Using this method one can observed that, with the exception of flight five, the values used in the BTF model are in close accordance with those derived by the CSA and the real ones, as demonstrated in Appendix A.2 .

Table 3.4 compares the results of (Murrieta-Mendoza et al., 2017) with those obtained using the BTF model.

Table 3.4 PSO and BTF results

	Consumed fuel			Relative difference	
Flight	Geodesic [Kg]	Optimal [Kg]	BTF [Kg]	Diff. Geodesic [%]	Diff. Optimal [%]
Montreal (YUL) Paris (CDG)	28,076	26,976	31,929	13.7%	18.4%
Toronto (YYZ) London(LDH)	28,846	27,633	32,931	14.2%	19.2%
Montreal (YUL) Vienna (VNN)	31,154	29,727	37,250	19.6%	25.3%

In the work of (Murrieta-Mendoza et al., 2017) there is no explicit reference to which phases, or aircraft models considered when modelling fuel burn using the geodesic or the optimal trajectory. The assumption is that the authors considered only CCD phases and, regarding the aircraft models our assumption is based on our research of FlightAware™ for the most common aircraft models used on the same flights namely the Boeing 787-900 for the flights from Montreal to Paris and Toronto to London. Regarding the latter, the authors, refer to the flight from Toronto to London using the International Air Transport Association (IATA) airport codes YYZ and LDH respectively. After checking IATA codes it was not possible to find out any airport in London with such IATA code, hence it was assumed London Heathrow (LHR) airport. As for the flight from Montreal to Vienna, since it was not possible to find any direct flights, it was considered the aggregated consumed fuel from two legs, the first from Toronto to Amsterdam using a Airbus 330-200, and the second one from Amsterdam to Vienna using a Boeing 737-800. As for the results it is possible to observe relative differences that range from 13.0% to 25.3% which can be accounted mainly for the fact that neither the aircraft models, neither the flight levels for the cruise phase are provided in their work.
In the work of (Hartjes et al., 2018) the authors modelled the flights, from London to Atlanta and from Madrid to New York, using a Boeing 747-400. The comparison between the results of the BTF model and those presented in (Hartjes et al., 2018) requires the same aircraft model and the precise airport locations for each of the flights, as shown in table 3.5.

Table 3.5 Solo flight, formation flight and and BTF results

	Solo flight			Formation flight			BTF model		
	Consumed fuel [Kg]	Time [h]	Distance [Km]	Consumed fuel [Kg]	Time [h]	Distance [Km]	Consumed fuel [Kg]	Time [h]	Distance [Km]
London (LHR) Atlanta (ATL)	79,093	7.92	6,760	75,530	8.09	6,825	77,051	8.18	6,768
Madrid (MAD) New York (JFK)	66,239	6.80	5,760	67,238	6.90	5,835	65,989	6.98	5,768

In the work of (Hartjes et al., 2018) during the cruise phase, the fight level increases with time and TAS decreases with time. The BTF model assumes that the cruise altitude is FL380 and that the TAS has a constant value 451 Kts. Although these distinctions exist, one can see that the relative differences, presented in table 3.6, between the results of work of (Hartjes et al., 2018) and the BTF model are minimal. The latter observation leads to the conclusion that, the more information, the better the fit between real values, for solo flights and, the BTF model. Finally, the consistent pattern, even though negligible, is that the BTF time is always greater than the ones presented for solo or formation flights.

Table 3.6 Solo flight, formation flight and and BTF relative differences

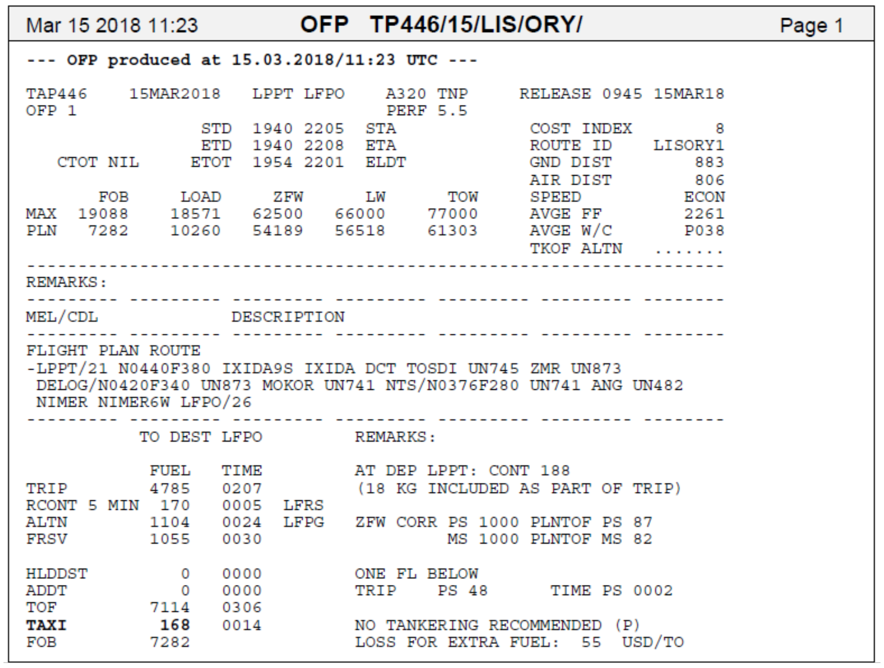
	BTF vs. Solo flight			BTF vs. Formation flight		
	Consumed fuel [Kg]	Time [h]	Distance [Km]	Consumed fuel [Kg]	Time [h]	Distance [Km]
London (LHR) Atlanta (ATL)	-1.3%	3.3%	0.1%	3.3%	1.1%	-0.8%
Madrid (MAD) New York (JFK)	-0.9%	2.6%	0.1%	-0.6%	1.2%	-1.1%

3.4.6 Comparing the BTF Model Results with Lido™

Lido™ consists of a software solution from Lufthansa Systems™ that offers extensive automation and optimisation functions for flight planning and control processes. Lido™'s flight plans are very extensive documents with information regarding the aircraft, the route and the weather.

This section compares the BTF model results for block time and consumed fuel, against Lido™'s flight plans. Figure 3.9 consists of the front page of the flight plan TP446/15/LIS/ORY. According to this flight plan the aircraft departs from Lisbon's Humberto Delgado Airport (IATA code: LIS, ICAO code: LPPT) and arrives in Paris' Orly Airport (IATA code: ORY, ICAO code: LFPO). It is assumed that the flight's block time that is going to be used in our comparisons is the difference between the Estimated Time of Departure (ETD) at 19:40 and the Estimated Time of Arrival (ETA) at 22:08. The fuel consumed during the flight is the difference between the Planned (PLN) TOW and the PLN Landing Weight (LW), 61,303 and 56,518 Kg respectively, plus the fuel mass consumed for 14 minutes of taxi-out (TAXI), 168 Kg. The flight plan does not provide the mass of fuel consumed during the taxi-in. The time the flight plan takes to do taxi-in is the difference between ETA and the Estimated Landing Time (ELDT), 3 minutes. Since the flight plan does not provide the mass of fuel during taxi-in, it will not be considered in the comparisons. Finally, instead of using the physical distance between origin and destination airports, the distance between the airports used in the BTF model is the Air Distance (AIR DIST) 806 nautical miles. This is justified by the fact that the AIR DIST is the distance through the air, hence based on the aircraft speed through the air. If the aircraft has a tailwind, that means the air mass is moving along with the aircraft, thus decreasing the actual distance through air to get from the origin to the destination airport. As a corollary a tailwind effectively decreases the flight plan air distance while a headwind increases it. In conclusion by assuming this assumption it is possible to account for the effect of the wind.

Appendix A.3 provides the remaining Lido™'s flight plan extracts for the flights that are being compared with the BTF model. Table 3.7 presents the comparison between the results obtained for block time and consumed fuel. It is possible to observe significant differences



```
Mar 15 2018 11:23            OFP   TP446/15/LIS/ORY/                    Page 1
--- OFP produced at 15.03.2018/11:23 UTC ---

TAP446    15MAR2018   LPPT LFPO    A320 TNP     RELEASE 0945 15MAR18
OFP 1                              PERF 5.5
                 STD  1940 2205  STA             COST INDEX        8
                 ETD  1940 2208  ETA             ROUTE ID    LISORY1
   CTOT NIL     ETOT  1954 2201  ELDT            GND DIST        883
                                                 AIR DIST        806
        FOB      LOAD     ZFW       LW      TOW  SPEED          ECON
MAX  19088     18571   62500    66000    77000   AVGE FF        2261
PLN   7282     10260   54189    56518    61303   AVGE W/C       P038
                                                 TKOF ALTN   .......
--------------------------------------------------------------------
REMARKS:
--------- --------- --------- --------- --------- --------- --------
MEL/CDL              DESCRIPTION
--------- --------- --------- --------- --------- --------- --------
FLIGHT PLAN ROUTE
-LPPT/21 N0440F380 IXIDA9S IXIDA DCT TOSDI UN745 ZMR UN873
 DELOG/N0420F340 UN873 MOKOR UN741 NTS/N0376F280 UN741 ANG UN482
 NIMER NIMER6W LFPO/26
--------- --------- --------- --------- --------- --------- --------
           TO DEST LFPO         REMARKS:

            FUEL   TIME         AT DEP LPPT: CONT 188
TRIP        4785   0207         (18 KG INCLUDED AS PART OF TRIP)
RCONT 5 MIN  170   0005  LFRS
ALTN        1104   0024  LFPG   ZFW CORR PS 1000 PLNTOF PS 87
FRSV        1055   0030                  MS 1000 PLNTOF MS 82

HLDDST         0   0000         ONE FL BELOW
ADDT           0   0000         TRIP     PS 48      TIME PS 0002
TOF         7114   0306
TAXI         168   0014         NO TANKERING RECOMMENDED (P)
FOB         7282                LOSS FOR EXTRA FUEL:  55  USD/TO
```

Fig. 3.9 Lido flight plan TP446 for the Lisbon, Orly flight

Table 3.7 Comparing block time and consumed fuel between the BTF model and Lido™'a flight plans

	BTF model		Lido flight plans		Relative difference	
Flight	Block time [min.]	Consumed fuel [Kg]	Block time [min.]	Consumed fuel [Kg]	Block time	Consumed fuel
TP757/15/CPH/LIS	237	8651	233	8629	1.69%	0.25%
TP946/20/LIS/GVA	158	5467	167	5929	-5.70%	-8.45%
TP672/17/LIS/AMS	168	6110	177	6230	-5.36%	-1.96%
TP586/17/LIS/CGN	168	6183	176	6134	-4.76%	0.79%
TP446/15/LIS/ORY	135	4958	148	4953	-9.63%	0.10%

To better understand the results in table 3.7, figure 3.10 plots the block time relative differences vs. BTF block time, figure plots 3.11 the block time relative differences vs. air distance, figure plots 3.12 consumed fuel relative differences vs. BTF block time and, figure 3.13 plots consumed fuel relative differences vs. air distance. The respective linear regression was added and the goodness-of-fit measure for linear regression value R^2 is calculated. It is possible to verify a significant goodness-of-fit for the linear regression for block time relative difference vs. BTF block time and, block time relative difference vs. air distance. However, for consumed fuel relative difference vs. BTF block time and, consumed fuel relative difference vs. air distance, the R^2 value is very low, hence the linear regression does not explain the variation in the consumed fuel relative difference.

In conclusion, the observed relative differences for consumed fuel cannot be entirely accounted by block time or air distance. Factors such as meteorology play an important role in flight planning and aircraft trajectory optimisation (Cheung et al., 2015), (Lindner et al., 2020). Weather conditions for all altitudes such as wind's direction and speed, cloud cover, visibility, and precipitation influence block time and fuel consumed during a flight. Additionally, fog, snow, ice, and crosswinds mean that air traffic controllers have to increase the gap between planes that are landing, reducing the number of aircraft that an airport can manage. The same weather can make it slower and more difficult for the aircraft to taxi

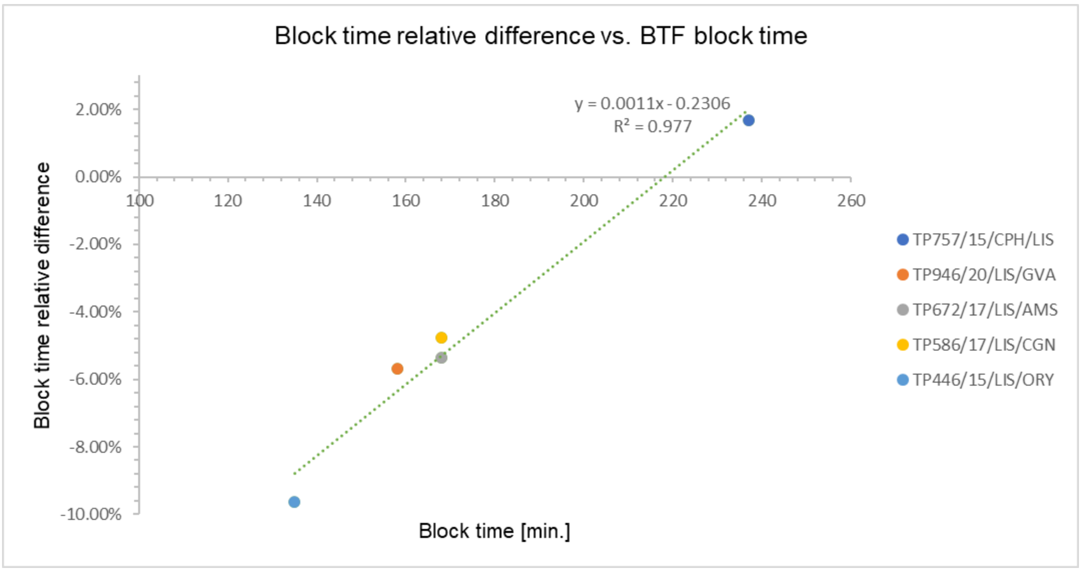


Fig. 3.10 Block time relative difference vs. BTF block time

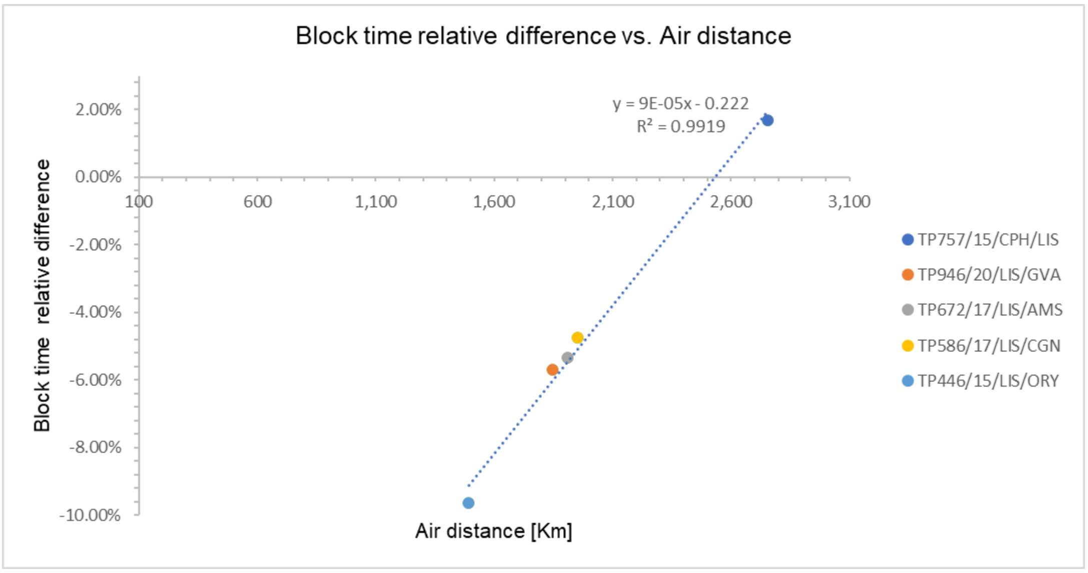


Fig. 3.11 Block time relative difference vs. Air distance

between the runway and the terminal building. Therefore, weather conditions and uncertain weather forecasts might induce the necessity to re-optimize ground movements and the trajectory during the flight. However, the re-optimisation leads to a complexity increase in the flight planning model, which must be balanced with the benefit of the re-optimisation. From this follows the option for not accounting for weather conditions in the BTF model, since it would require access to real-time weather forecast and a substantial increase of computing time. Another important factor that should be accounted for in flight planning is the aircraft's weight. Weight is a force that acts on the aircraft and is always directed toward the centre of the Earth. The magnitude of the weight depends on the mass of all the aircraft parts, plus the amount of fuel, plus any payload on board (people, luggage, freight, etc.). During the flight as fuel is consumed the aircraft's weight decreases, however the BTF model assumes that the weight of the aircraft during the flight does not change, therefore the modelling uses the same mass level during the entire flight. Including the aircraft weight change in the modelling, would mean the possibility of mass level change

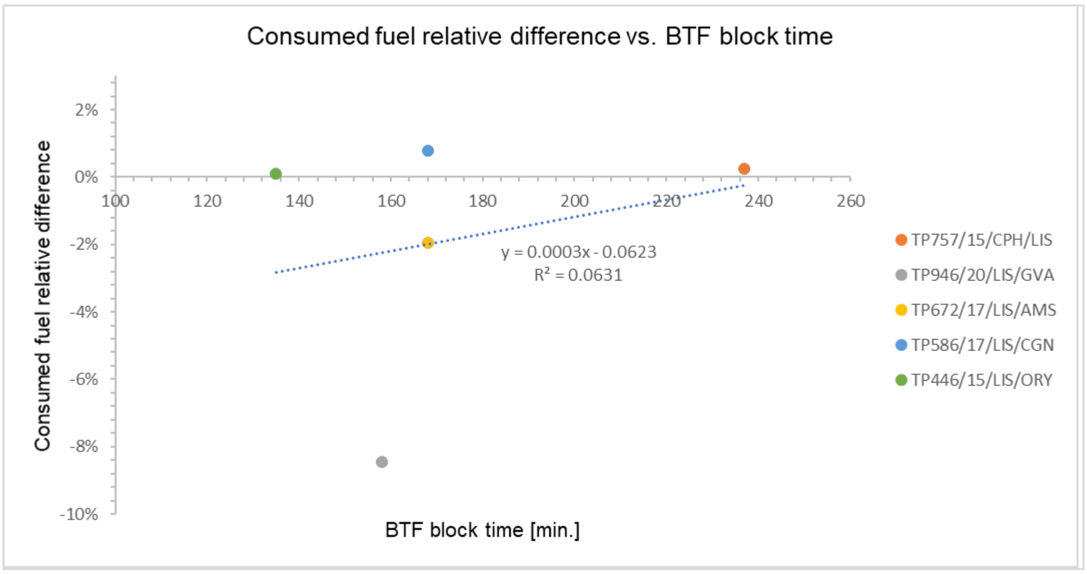


Fig. 3.12 Consumed fuel relative difference vs. BTF block time

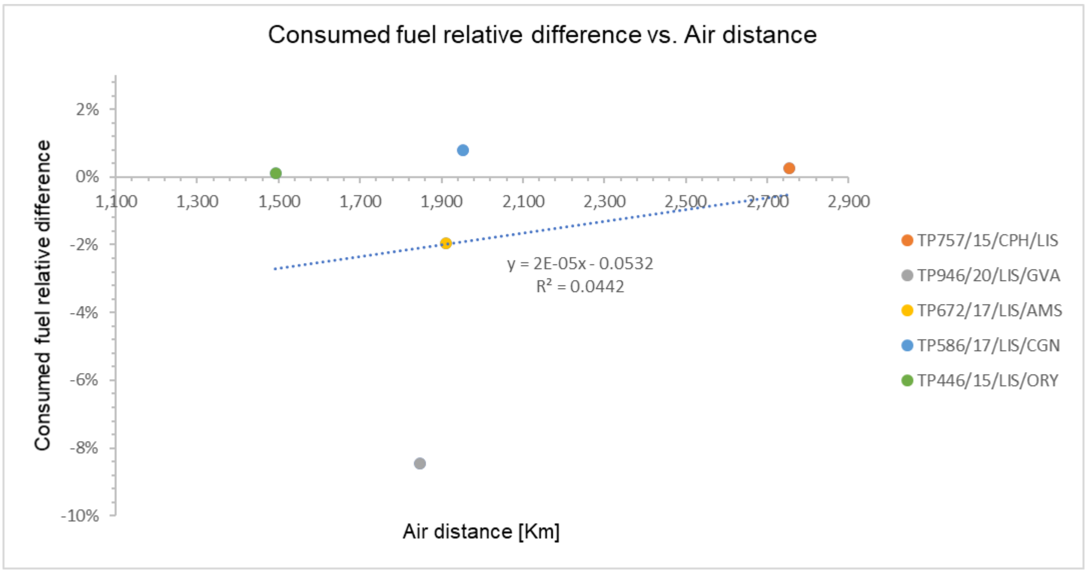


Fig. 3.13 Consumed fuel relative difference vs. Air distance

during the flight, which would result in changing the data points for ROCD, TAS and fuel flow used by the BTF. The complexity of the algorithm would become extremely high since the mass level change could happen at any point of the trajectory. The former requires a structural change in the algorithm, which can only be considered in the scope of future work.

To further compare the BTF and Lido™ values for block time, figure 3.14 presents the percentile for each of the flights when compared with the distribution retrieved by FlightAware™.

Similarly, figure 3.15 illustrates the RMSE of the BTF model and Lido™ for block time when compared with FlightAware™, versus the ground distance.

The complete set of results are presented are presented in table A.3. As for the plots, it is possible to observe that they are in line with those obtained for ROADEF in figures 3.7 and 3.8. Again it possible to observe lower percentile and lower RMSE for the BTF.

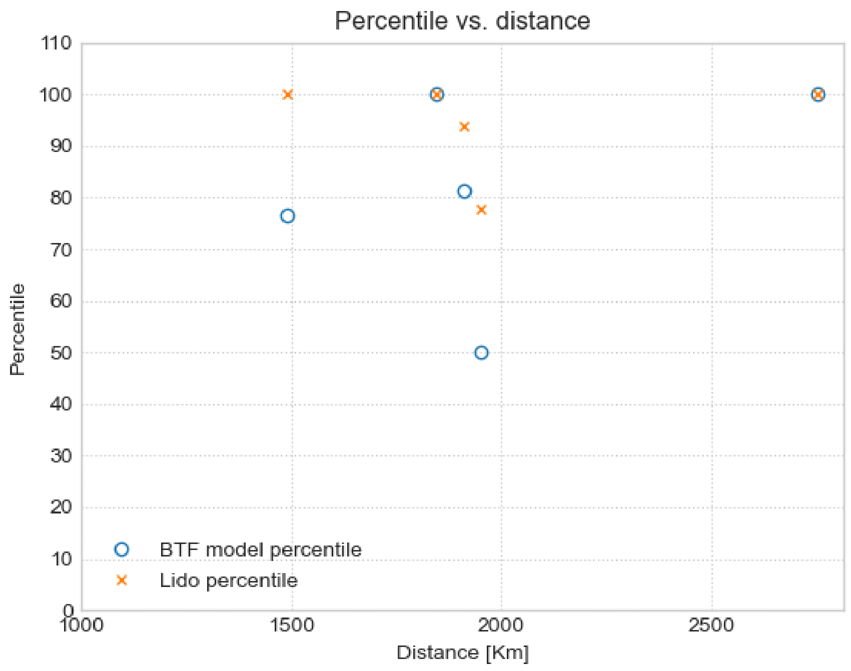


Fig. 3.14 Block time percentile for the BTF model and Lido ™ vs. distance

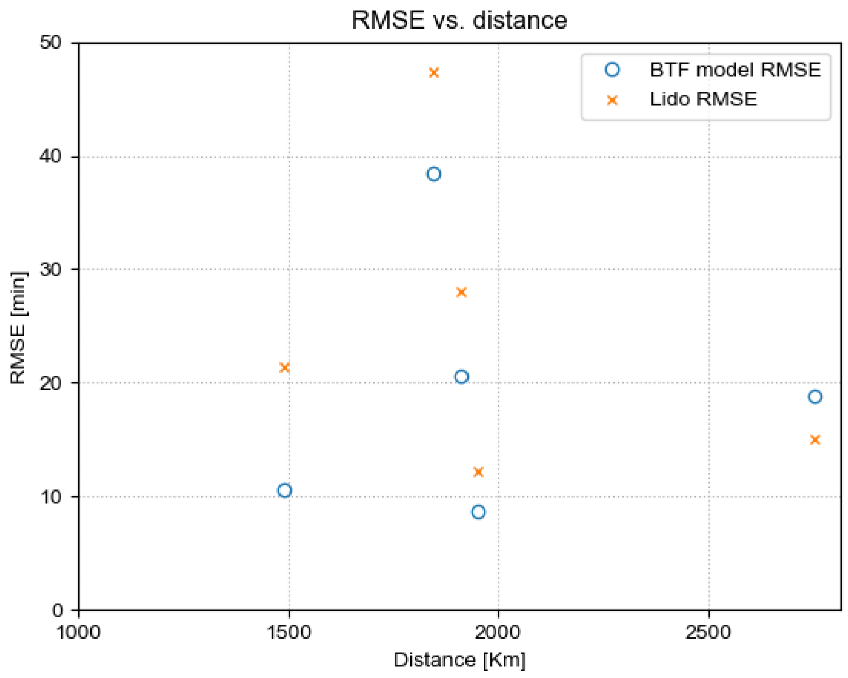


Fig. 3.15 Block time RMSE for the BTF model and Lido ™ vs. distance

3.5 Conclusion and Future Work

This chapter models flights integrating each of its composing phases starting from the departure gate and ending at the arrival gate. The BTF model integrates the ground movements using the data from

EMEP/EEA emissions data set with BADA aircraft performance data tables. To model the flight when the aircraft is airborne the BTF model uses Newtonian Mechanic. It is assumed the Earth as a perfect sphere and to calculate the distance between the origin and destination airports the Haversine formula is used.

As a general rule, the aircraft reaches its FL, then its cruising speed, in that order. From the moment the FL is reached, the excess thrust (compared to the drag) would accelerate the aircraft, however the BTF uses constant cruise speed. Additionally, due to the loss of mass (due to fuel burn) the lighter aircraft would tend to climb, however the BTF does not consider this and, during the cruise phase the altitude does not vary. Similarly, the BTF does not account for the change in the mass level and only uses the nominal mass level ROCD and fuel flow. As for the descent it is assumed that it is continuous in the sense that trajectory does not have periods in which it is flat. In reality in this phase there are periods in which pilots will correct the descent trajectory making it a stable flat line for short periods.

Although the ROADEF 2009 Challenge provides a complete data set to model airline disruption, it lacks data regarding the operational characteristics for the aircraft, namely the amount of fuel consumed during a flight. In order to overcome this shortage, this thesis uses the values obtained from literature review, and it was possible to concluded that the BTF model results have a good fit for time and consumed fuel for the CCD phases, provided that the inputs for origin and destination airports, and also the aircraft model are known. To extend the validation of the BTF results, block time and consumed fuel comparisons are made with Lido™ flight plans. It is possible to conclude that the BTF shows a good fit and that the differences in the block time can be accounted for. As for the observed differences in consumed fuel further investigation should be made in future work to reflect the effects of mass level changes during the flight.

It is possible to conclude that aggregating EMEP/EEA data with BADA PTF data files provides a simple and fast approach to calculate block time and consumed fuel. Hence, it is possible to confirm the conclusion of (Alam et al., 2009) that, BADA fuel flow tables are a good approach when computational cost is a factor. Using this method one was able to extend the work of (Alam et al., 2009), (Murrieta-Mendoza et al., 2017) and (Hartjes et al., 2018) not only in terms of the number flights evaluated but also for all the flight phases.

Finally it was also possible to conclude that the BTF model can calculate block times lower than those used in the ROADEF 2009 data set but still on 90^{th} percentile when compared with real ones obtained using Fligtaware™. In Chapter 5 the values used in the ROADEF Challenge 2009 for block time, will be replaced with those obtained from the BTF results in order to study the impact of smaller block times in the ARP.

Chapter 4

The Constructive Heuristic for the Aircraft Recovery Problem

4.1 Introduction

The financial objective of airline companies is maximising their profits while operating flights according to a schedule. Therefore, airlines create tight schedules to increase their profitability. These tight schedules, have a minimum slack between flight legs, because they rely on the assumption that the flight legs will be operated to the planned rotation. A rotation consists in assigning an aircraft to a flight schedule while complying with operational constraints such as, aircraft maintenance, flight continuity, turn-round time and airport capacity for departure and arrival.

In case of disruptions, it is necessary to recover the rotation during the period of time designated by the Recovery Time Window (RTW). During the RTW it is necessary to find a feasible rotation that mitigates the impact of disruptions. This process is designated by disruption recovery.

Disruption recovery is a real-time practice that requires finding a fast solution when irregularities occur. Often, disruptions take place during operations (i.e., only in few cases, it is possible to know a disruption in-advance), and a provisional plan must be provided quickly. Thus, it is necessary to construct efficient algorithms to have feasible solutions in minutes, moreover because when disruptions occur, they can affect more than just the directly disrupted flight and can send a shockwave of disruptions through the network.

When building a feasible solution for a disrupted aircraft rotation it is mandatory to satisfy the model's constraints, therefore a disruption recovery procedure is a CSP. CP is a method for solving CSPs. CP uses consistency techniques (so-called constraint propagation) that can effectively reduce the search space and early identify inconsistencies, along the search, by deduction, (Hooker and van Hoeve, 2017).

The contribution of this chapter consists of modelling the ARP as a CSP and implement the recovery for large data instances in a reduced computing time using CP and CPr.

The structure of this chapter is as follows, Section 4.2 introduces the ARP model, Section 4.3 describes the CP concepts that are in the base of the CHARP, Section 4.4 presents all the algorithms of the CHARP, Section 4.5 presents the computational results for the CHARP under two scenarios, Section 4.6 presents the comparison between the CHARP and published work, and finally the conclusions obtained and future work are presented in Section 4.7.

4.2 Aircraft Recovery Problem Model

The ARP can be defined as the problem of modifying the aircraft rotation to compensate the presence of disruptions during operations that make the rotation infeasible. The strategy to overcome the disruptions, can only be implemented during the RTW. The ARP model presented in this section follows the ROADEF 2009 Challenge and uses its data set (see Appendix B.1) to run the experiments. This data set has thirty two data instances divided into three sets A, B and X. In respect to the dimension of the problem, the smaller instances have 608 flights whereas the biggest have 2,178 flights distributed for RTWs spanning 1 to 3 days. These flights are operated by a proportional number of aircraft, for the smaller instances 85 and for the biggest 255. As for the number of airports, it varies from 35 to 168 and finally the number of itineraries is distributed between 1,943 for the smaller instances and 28,308 for the biggest. Each of the data instances has scenario combining disruptions for flights, aircraft and airport capacity. In the following subsections we will describe each of the data instances' sets that are being used to model the ARP.

4.2.1 Nomenclature for the ARP

The purpose of this subsection is to introduce the sets, parameters and iterators that oft the ARP, tables 4.1, 4.2 and 4.3 respectively.

Table 4.1 ARP sets

Sets	Description
A	Airport set
Δ	Block time set
P	Aircraft set
ϕ	Flight set
ρ	Rotation set
$\mathscr{D}$	Flight disruption set
$\mathscr{B}$	Aircraft disruption set
$\mathscr{R}$	Airport disruption set

Table 4.2 ARP sets' iterators

Iterator	Description
a	Airport iterator
h	Time window iterator
δ	Block time iterator
p	Aircraft iterator
i	Flight leg iterator
d	Flight disruption iterator
b	Aircraft disruption set

4.2.2 Airport Departure and Arrival Capacity

The key infrastructure in commercial aviation consists of airports. In this problem, they have also an essential role since their capacity consists of a hard constraint that limits the maximum number of arrivals and departures per hour. The airports form a set A. For each $a \in A$ and for each window $h \subseteq RTW$, the

Table 4.3 ARP parameters

Parameter	Description
c_a^{lh}	Maximum number of airport arrivals
c_a^{jh}	Maximum number of airport departures
$\delta_{a_1 a_2}$	Block time between airports a_1 and a_2
O_p	Origin airport for aircraft p
$\rho_p^o(i)$	Origin airport fro the $i^t h$ flight leg for aircraft p
$\rho_p^f(i)$	Destination airport for the $i^t h$ flight leg for aircraft p
$\rho_p^d(i)$	Scheduled departure time for the $i^t h$ flight leg for aircraft p
$\rho_p^a(i)$	Scheduled departure time for the $i^t h$ flight leg for aircraft p
t	Delay [minute]
s	Starting time of an aircraft disruption
e	End time of an aircraft disruption
z	Affected airport activity
c	Airport new capacity
H	Day time [hour]

value c_a^{lh} is the maximum number of arrivals that can occur during the time window h at airport a and c_a^{jh} the maximum number of departures. It is necessary to discretise these hourly capacities to incorporate them within the model's formulation. For example, the recovery time window between $[12{:}00, 16:00[$ comprises four unit time windows, and as an illustration, if the departure capacity of an airport between $[14:00, 15:00[$ is equal to 3, this indicates there can be at most three flights departing form this airport between 14:00 and 14:59.

The A set encodes for each element the departure and arrival airport capacities, which correspond to a maximum number of operations allowed per one-hour interval, for a typical day. These thresholds depend upon the time of day (peak time, normal time, night time, possible curfew). Each airport is coded according to IATA codes, followed by a series of quadruples specifying the capacities associated with each time period. Capacities are non-negative integers, and the time periods are characterised by two entries of type "time" corresponding to the start time and end time of the period *e.g.*:

```
NCE 0 0 00:00 03:00 5 0 03:00 07:00 20 20 07:00 19:00 5 10 19:00 21:00 0 0 21:00 00:00
```

The above example describes a typical day (in Greenwich Mean Time (GMT)) for Nice Côte d'Azur Airport (NCE):

- Neither departures nor arrivals between 21:00 and 3:00.
- Maximum 5 departures per one-hour interval [H, H + 1[between 3:00 and 7:00 (and no arrivals).
- Maximum 20 departures and 20 arrivals per one-hour interval [H, H + 1[between 7:00 and 19:00.
- Maximum 5 departures and 10 arrivals per one-hour interval [H, H + 1[between 19:00 and 21:00.

For simplicity, some real constraints are not taken into account in the problem (e. g. the maximum number of aircraft in the airport surface).

4.2.3 Block Time Between Airports

The Δ set encodes the block time for each airport pair, as well as the flight type. More exactly, for each airport pair $\langle a_1, a_2 \rangle \in A, a_1 \neq a_2$, the block time between a_1 and a_2 is $\delta_{a_1 a_2} \in \Delta$. Note that, for a given airport pair, the flight times may depend on the direction of the flight. Each element $\delta \in \Delta$ has the three-letter IATA codes of the origin and destination airports, the flight time and the flight type *e.g.*:

```
CDG NCE 95 D
NCE CDG 95 D
```

The above example provides the block time between NCE and CDG (95 minutes in both directions) and specifies that the flight is domestic (D).

4.2.4 Aircraft specification

Let P denote the set of aircraft, where each aircraft $p \in P$ has the following set of operational characteristics:

- **Aircraft type**: it defines the family, model, and configuration; subsets of aircraft with common characteristics are grouped within families (e. g., A318, A319, A320, and A321 in the Airbus Small family). Operational characteristics are common to all aircraft of a given model: turn-round time, transit time, range, and set of possible configurations. The configuration provides the number of seats for each cabin class, economic, business and first.
- **Transit time**: corresponds to the minimum time between the arrival and departure of multi-leg flight operated by the same aircraft. The advantage of this configuration is that it allows for a reduction in the time necessary to prepare the aircraft for the second leg.
- **Turn-round time**: defines the minimum idle time between two different consecutive flights operated by the same aircraft.
- **Scheduled maintenance**: an aircraft needs to undergo scheduled maintenance on a specified airport during a period of time, in which it is unavailable for flight duty, and a maximum allowable flying range before the maintenance is due.

Each aircraft $p \in P$ encodes the following characteristics e.g.:

```
A320#1 A320 AirbusSmall 0/20/150 480 1500.0 30 30 CDG CDG-10/01/08-14:00-10/01/08-20:00-900
A320#2 A320 AirbusSmall 10/30/110 480 1500.0 30 30 NCE NULL
TranspCom#1 TranspCom TranspCom -1/-1/-1 60 0.0 5 5 CDG NULL
TranspCom#2 TranspCom TranspCom -1/-1/-1 60 0.0 10 10 ORY NULL
```

The above example provides the characteristics of the first two Airbus A320 in the fleet. Note that they have several common characteristics, but their configurations are different. The first one is located in CDG, and must undergo maintenance there on 10/01/08 between 14:00 and 20:00 (it cannot fly more than 15 hours between two consecutive maintenance actions). The second one is located in Nice and has no planned maintenance. The remaining two consist of surface transportation vehicles 1 and 2. Both of them belong to the family TranspCom, have infinite seating capacities, and operating costs of zero. Vehicle 1 is located at CDG in the beginning of the period, whereas vehicle 2 is in Orly Airport (ORY).They are not accountable in terms of airport capacity.

4.2.5 Flights and Rotations

The flight set ϕ provides information regarding the flights operated by the airline. For each flight, the following data is provided: unique identification number (strictly positive integer), origin and destination airports, departure and arrival times, and the flight number of the preceding leg (strictly positive in the case of multi-leg flights, 0 otherwise) *e.g.*:

```
1 NCE CDG 14:00 15:35 0
2 SIN LHR 15:20 05:30+1 0
3 LHR CDG 06:30 07:45 2
4 ORY CDG 08:30 09:00 0
```

The above example describes flight number 1, leaving from NCE at 14:00 GMT and arriving at CDG at 15:35 GMT. It also describes the multi-leg flight from Singapore Changi Airport (SIN) to CDG via London Heathrow Airport (LHR), composed of flights numbered 2 and 3. The first leg (intercontinental) leaves from SIN at 15:20 GMT and arrives at LHR at 5:30 GMT the following day; the second flight (continental) departs LHR at 6:30 GMT and arrives in CDG at 7:45 GMT. The last line describes flight number 4 from ORY to CDG, corresponding to a surface transportation "flight".

The ρ set describes rotations for all aircraft throughout the planning period. Each flight is uniquely defined by a flight number (strictly positive integer) and a departure date. The aircraft operating the flight is also provided. The lines are grouped by aircraft and sorted chronologically for each aircraft *e.g.*:

```
2 20/01/08 B747#5
3 21/01/08 B747#5
2 21/01/08 A340#2
3 22/01/08 A340#2
4 21/01/08 TranspCom#2
```

The above example describes the rotations of the Boeing 747 number 5, the Airbus A340 number 2, and the surface vehicle number 4 throughout the recovery period, which is from 20/01/08 to 21/01/08. These rotations consist of flights number 2 and 3 (multi-leg flights from SIN to CDG) on 20/01 and 21/01, and of the surface trip from ORY to CDG on 21/01, respectively.

The result of the match between the ϕ set with the ρ, over the flight number, returns the aircraft fleet rotation for the planning horizon. An aircraft p performs a rotation which is a sequence ρ_p of flight legs starting from an origin airport O_p. The i^{th} flight leg in the aircraft rotation $\rho_p(i)$ is defined as the direct flight connecting an origin $\rho_p^o(i)$ to a destination airport $\rho_p^f(i)$ without any stop in between. Each of these flights is also characterised by a scheduled departure time $\rho_p^d(i)$ and an arrival time $\rho_p^a(i)$. A rotation can be extended for several days and the recovery procedure can be made only made during the RTW.

4.2.6 Flight Disruptions

Flight disruptions, consists of multiple flight delays or cancellations. The set of flight delays $\mathscr{D}$ is such that each delay $d \in \mathscr{D}$ can be defined by a triplet $\langle p,i,t\rangle \in P \times \mathbb{N}^+ \times \mathbb{N}^+$ with p the affected aircraft, i the index of the affected leg in ρ_p, and t the delay in minutes. The set of flight cancellations is defined by a couple $\langle p,i\rangle \in P \times \mathbb{N}^+$ with p the affected aircraft and i the index of the affected leg in ρ_p. The $\mathscr{D}$ set provides the disruptions happening to flights operated by the considered airline, namely delays and cancellations. Each impacted flight is uniquely identified by a flight number and a departure date. The information regarding the flight disruption conists of the length of the delay in case of delay, and -1 in case of cancellation *e.g.*:

```
2 20/01/08 45
1 21/01/08 -1
```

The above example specifies a delay of 45 minutes on flight number 2 (SIN - LHR) on 20/01/08 and the cancellation of flight number 1 (NCE - CDG) on 21/01/08.

4.2.7 Aircraft Disruptions

Aircraft mechanical failures or the absolute need of maintenance are the reasons for aircraft disruptions. The set of aircraft disruptions $\mathscr{B}$ is such that each aircraft disruption $b \in \mathscr{B}$ forms a triplet $\langle p,s,e\rangle \in P \times \mathbb{N}^+ \times \mathbb{N}^+$ with p the affected aircraft, s the start of the aircraft's period of unavailability and e the end time of the aircraft's unavailability period. The $\mathscr{B}$ set provides the periods of unavailability of aircraft. Each element contains the aircraft identification number of the unavailable aircraft, the date and time of the beginning of the period of unavailability, and the date and time of the end of the period of unavailability *e.g.*:

```
A320#1 20/01/08 04:00 20/01/08 20:00
```

The above example mentions that the Airbus A320 number 1 will not be available between 4:00 and 20:00 on 20/01/08.

4.2.8 Airport Disruptions

Airport disruptions results in capacity reduction in the number of departures or arrivals per hour, caused by inclement weather or industrial action. The set of airport capacity reductions $\mathscr{R}$ is such that each reduction $r \in \mathscr{R}$ is defined by a quadruplet $\langle a,h,z,c\rangle \in A \times RTW \times \{j,l\} \times \mathbb{N}^+$ with a the affected airport, h the time window during which the capacity reduction occurs, z the affected activity (departure or arrival), and c the new capacity. The $\mathscr{R}$ set provides the periods of temporary reductions in airport capacities. Each element contains the three-letter code of the airport where the reduction occurs, the date and time of the start of the period of reduction, the date and time of the end of the period of reduction, and the applicable departure and arrival capacities during the period of reduction. The capacities are non-negative integers *e.g.*:

```
LHR 20/01/08 04:00 20/01/08 10:00 0 2
```

The above example corresponds to a scenario of dense fog in LHR: no departures and only two arrivals are allowed per one hour interval [H, H + 1[from 4:00 to 10:00.

4.2.9 Aircraft Constraints

Since this model is a simplification of the real problem, neither crew scheduling nor passenger re-accommodation will be considered. The crew scheduling problem is an NP-hard optimisation problem that must be solved under numerous constraints (Deveci and Demirel, 2016). Due to the difficulty of solving the airline crew scheduling problem, it is generally divided into two sub-problems consisting of the crew pairing problem (CPP) and the crew rostering problem (CRP). However, ideally it should be one integrated problem and model. As for the passenger re-accommodation problem, it can be formulated as follows: given a recovered flight and crew schedule and a set of disrupted passenger itineraries, re-assign to each disrupted itinerary the (recovered) flights necessary (given seat availability) to accommodate passengers from their current position to their destination while minimising cost. Add crew scheduling and passenger re-accommodation would increase the complexity of the model, therefore increasing computing time. Additionally, solutions that satisfy aircraft, crew, and passenger recovery would become harder to find and substantially more costly.

Rotation continuity

A rotation ρ_p starting at the origin airport $\rho_p^o(1) = O_p$, must be connected. In expression 4.1 for all flight legs in the interval from the first flight to the open end of the rotation size, the arrival airport of the current flight equals the departing airport of the next flight:

$$\forall i \in [1, |\rho_p|[, \; \rho_p^f(i) = \rho_p^o(i+1) \tag{4.1}$$

$|\rho_p|$ being the number of flights in aircraft p rotation

Turn-round Time

The aircraft p must respect the turn-round duration t_{rp} between the consecutive legs:

$$\forall i \in [1, |\rho_p|[, \; \rho_p^a(i) + t_{rp} \leq \rho_p^d(i+1) \tag{4.2}$$

The same applies for transit time between multi leg flights.

Maintenance

The maintenance constraints are hard constraints. For a subset $P_m \subset P$, each aircraft $p \in P_m$ must undergo maintenance in a certain airport during a period of time.

4.3 Constraint Programming Concepts and Applications

A constraint is simply a logical relation among several unknowns (or variables), each taking a value in a given domain. A constraint thus restricts the possible values that variables can take, it represents some partial information about the variables of interest. IBM, 2022 defines CP as the method of optimising a function subject to logical, arithmetic, or functional constraints over discrete variables or interval variables. The basic idea in constraint programming is that the user states the constraints and a general purpose constraint solver is used to solve them. Constraints are just relations, and a CSP states which relations should hold among the given decision variables.

A CSP $\mathscr{P}$ is defined by a triple $\mathscr{P} = \langle \mathscr{X}, \mathscr{D}, \mathscr{C} \rangle$ where $\mathscr{X}$ is an n-tuple of variables $\mathscr{X} = \langle x_1, x_2, ..., x_n \rangle$ $\mathscr{D}$ is a corresponding n-tuple of domains $\mathscr{D} = \langle D_1, D_2, ..., D_n \rangle$ such that $x_i \in D_i$, C is a t-tuple of constraints $\mathscr{C} = \langle C_1, C_2, ..., C_t \rangle$. A constraint C_j is a pair $\langle R_{S_j}, S_j \rangle$ where R_{S_j} is a relation on the variables in $S_i = scope(C_i)$. In other words, R_i is a subset of the Cartesian product of the domains of the variables in S_i (Rossi et al., 2006a).

A solution to the CSP $\mathscr{P}$ is an n-tuple $A = \langle a_1, a_2, ..., a_n \rangle$ where $a_i \in D_i$ and each C_j is satisfied in that R_{S_j} holds on the projection of A onto the scope S_j . In a given task one may be required to find the set of all solutions, $sol(\mathscr{P})$, to determine if that set is non-empty or just to find any solution, if one exists. If the set of solutions is empty the CSP is unsatisfiable.

We will consider the algorithms for solving CSPs under two broad categories: inference and search, and various combinations of those two approaches. If the domains D_i are all finite, then the finite search space for putative solutions is $\Omega = \bowtie_i D_i$ (where $\bowtie$ is the join operator of relational algebra).

In inference techniques, local constraint propagation can eliminate large subspaces from Ω on the grounds that they must be devoid of solutions. Search systematically explores Ω, often eliminating subspaces with a single failure. The success of both strategies hinges on the simple fact that a CSP is conjunctive: to solve it, all of the constraints must be satisfied so that a local failure on a subset of variables rules out all putative solutions with the same projection onto those variables.

Many algorithms for solving CSPs search systematically through the possible assignments of values to variables. Such algorithms are guaranteed to find a solution, if one exists, or to prove that the problem is insoluble. Thus, the systematic search algorithms are sound and complete [1]. The main disadvantage of these algorithms is that they take a very long time to do so. As opposed, an incomplete method for solving a general constraint satisfaction problem is one that does not provide the guarantee that it will eventually either report a satisfying assignment or declare that the given formula is unsatisfiable. In practice, most such methods are biased towards the satisfiable side: they are typically run with a pre-set resource limit, after which they either produce a valid solution or report failure; they never declare the formula to be unsatisfiable.

Generate and Test (GaT) method originates from the mathematical approach to solving combinatorial problems. It is a typical representative of algorithms that search the space of complete assignments. First, the GaT algorithm generates some complete assignment of variables and then it tests whether this assignment satisfies all the constraints. The GaT algorithm search systematically the space of complete assignments, *i.e.*, it explores each possible combination of the variable assignments. The number of combinations considered by this method is equal to the size of the Cartesian product of all the variable domains.

[1] A proof system is sound if everything that is provable is in fact true, and complete if everything that is true has a proof.

When a variable is instantiated a value from its domain is picked and assigned to it. Then constraint checks are performed to make sure that the new instantiation is compatible with all the instantiations made so far. If all the completed constraints are satisfied this variable has been instantiated successfully and we can move on to instantiate the next variable. If it violates certain constraints, then an alternative value when available is picked. If all the variables have a value noting that all the assignments are consistent the problem is solved. If at any stage no value can be assigned to a variable without violating a constraint backtracking occurs. In general, that means the most recent instantiated variable has its instantiation revised and a new value if available is assigned to it. However, as it will be demonstrated in Section 4.4.8, in the case of the CHARP it is necessary a search procedure to find the particular variable assignment that needs to be revised.

CPr is aimed at transforming a CSP into an equivalent problem that is hopefully easier to solve. CPr works by reducing the domain size of the variables in such a manner that no feasible solutions are ruled out. The CHARP uses repeatedly CPr to satisfy constraints such as airport landing and arrival capacity and, flight turn-round time, to reduce the size of the search space. By dramatically reducing the size of the search space, CPr methods accelerates the search procedure Grönkvist (2006).

CSPs have associated constraint graphs where the nodes represent variables and the edges binary constraints. The simplest degree of consistency that can be enforced on a CSP is node consistency which concerns only the unary constraints. A CSP is node consistent if and only if for all variables all values in its domain satisfy the unary constraints on that variable. If a CSP is not node consistent then there exists a certain variable x_i and a certain value a in its domain, such that value a does not satisfy the unary constraints on variable x_i. This means the instantiation of x_i to $d \in D_i$ always results in an immediate failure. In other words, value d is redundant and will not be in any solution tuples. In Section 4.4.4 we make sure that every domain value satisfies airport departure and arrive capacity.

A stronger degree of consistency is Arc-consistency (AC). AC concerns the binary constraints in a CSP and considers binary constraints between one pair of variables at a time. Arc (x_i,x_j) is arc consistent if and only if for every value a in the current domain of x_i there exists some value b in the domain of x_j such that $x_i = a$ and $x_j = b$ are permitted by the binary constraint between x_i and x_j. In this thesis the rotation's flights are represented by (flight) arcs. Subsection 4.2.9 introduced the turn-round constraint 4.2 and Subsection 4.4.5 describes the method used to enforce arc consistency and reduce the search space.

4.4 The Constructive Heuristic for the Aircraft Recovery Problem

Given the original aircraft rotation and a set of disruptions the CHARP creates a new combination of aircraft routes during the RTW that minimises the two tiered objective, first number of flight cancellations and second the amount of delay necessary to comply with the operational constraints. The next subsections describe in detail the objective in Subsection 4.4.1, the algorithms that compose the CHARP, namely in Subsection 4.4.2 the algorithm that creates the flights in the presence of flight or aircraft disruption, in Subsection 4.4.3 the algorithm that checks the rotation's feasibility, in Subsection 4.4.4 the algorithm that computes the flight domains and intervals, in Subsection 4.4.5 the algorithm that propagates constraints and enforces AC, in Subsection 4.4.6 the Lower Heuristic Algorithm (LHA), in Subsection 4.4.7 the Upper Heuristic Algorithm (UHA), in Subsection 4.4.8 the Backtracking Algorithm (BTA), in Subsection 4.4.9 the Taxi Flights Algorithm (TFA), and finally in Subsection 4.4.10 the CHARP algorithm.

4.4.1 Objective

Some of the recovery strategies that will be used consist of cancelling flights and delaying them. the cost of flight cancellation is greater than the cost of flight delay, hence the objective of the CHARP is two tiered, the first priority is to minimise the number of cancellations, expression 4.3, and the second priority is to minimise the difference of minutes of delay for each recovered aircraft rotation, expression 4.4.

$$Min \sum_{p=1}^{|P|} \sum_{i=1}^{|\rho'_p|} \rho'^d_p(i)[\rho'^d_p(i) = -1] \tag{4.3}$$

$$Min \sum_{p=1}^{|P|} \sum_{i=1}^{|\rho'_p|} \rho'^d_p(i) - \rho^d_p(i) \tag{4.4}$$

From the algorithmic standpoint the CHARP implements solutions as vectors whose elements are either the option to cancel a flight, encoded as -1, or the amount of delay encoded as a non-negative integer. Hence the best solution consists of the vector with the same size as the rotation's part that needs to be recovered. Each element of this vector is initialised with the cancellation value -1, afterwards Algorithm 1 guides, in each iteration, the search procedure towards the objectives 4.3 and 4.4. The inputs are the best solution *bestSol* and the new solution *row*. *bestSol* encodes the amount of cancellation, minutes of delay and the best solution. Every time a new solution is to be tested the algorithm calculates the number of cancellations, the amount of delay, and encodes them into a new solution *newSol*, lines 1 to 3. Since the amount of cancellation is a negative number, if the amount of cancellation for the new solution is lower than the best solution the algorithm does not accept the new solution and returns *null*, lines 4 and 5. If the amount of cancellation for the new solution is equal to the amount of cancellation for the best solution and, the amount of delay for the new solution is greater than the best solution, the algorithm rejects the new solution and returns *null*, lines 6 and 7. Finally, if the above conditions are not respected the algorithm returns the new solution, line 8.

Algorithm 1: Objective function

```
  Input: $bestSol, row$
  Output: newSol
1 $noCancel \leftarrow \sum_i row(i)$, if $row(i) = -1$ /* The sum is a negative number */
2 $totalDelay \leftarrow \sum_i row(i)$, if $row(i) \neq -1$
3 $newSol \leftarrow \{noCancel, totalDelay, row\}$
  /* bestSol[0] has the amount of cancellation */
4 if $newSol[0] < bestSol[0]$ then
5     return $null$
  /* bestSol[1] has the amount of delay */
6 if $(newSol[0] = bestSol[0]) \wedge (newSol[1] \geq bestSol[1])$ then
7     return $null$
8 return $newSol$
```

4.4.2 New Flights Algorithm

Before starting the New Flights Algorithm (NFA) (algorithm 2), the main algorithm checks the rotation for cancelled flights or aircraft breakdown periods that also lead to flight cancellation. If these disruptions exist, the main algorithm adds empty flight slots ρ_p^n, to the aircraft rotation ρ_p.

Algorithm 2 describes the creation of new flights in the case of aircraft disruption. The algorithm receives as inputs the aircraft rotation, the block time set, the maximum flight number M, the aircraft breakdown period, and the end time of the recovery window (RTW_e). The algorithm will return the aircraft rotation with the new flight and the updated maximum flight number. This algorithm starts by initialising two sub-rotations containing available flight slots, the cancelled flights and the starting time (that corresponds to the end time of the aircraft disruption B_{pe}) from which new flights can be created (lines 1 to 3). Algorithm 2 will afterwards loop through the new flight slots and cancelled flights and calculate the departure time and the flight time (line 6). If the departure time added to the flight time of the new flight overshoots the end time of the RTW_e, the loop breaks and the algorithm returns the rotation with the new flights and the number of the last flight, otherwise the flight number is incremented, and the new flight slot is updated (lines 11 to 15). Finally, the algorithm checks if the new flight's arrival time added with the transit time overshoots the end time of the recovery time window. If true, the algorithm breaks the loop and returns the rotation with the new flights and the number of the last flight.

For flight cancellation, an identical algorithm can be derived from algorithm 2 by simply allocating the new flight slots to those that were cancelled.

Algorithm 2: New Flights Algorithm

```
   Input: $\rho_p, \Delta, M, \mathscr{B}_p, RTW_e$
   Output: $\rho_p, M$
1  $\rho_p^n \leftarrow$ Subset of available new flights in $\rho_p$
2  $\rho_p^c \leftarrow$ Subset of cancelled flights in $\rho_p$
3  $start \leftarrow \mathscr{B}_{pe}$ /* End time of aircraft disruption */
4  for $\rho_p^n(i), \rho_p^c(i)$ in $\rho_p^n, \rho_p^c$ do
5      if $i \neq 0$ then
6          $start \leftarrow \rho_p^{na}(i-1) + t_{rp}$
7      $\delta_{of} \leftarrow \Delta(\rho_p^{co}(i), \rho_p^{cf}(i))$
       /* Check if the departure is outside the RTW */
8      if $start + \delta_{of} > RTW_e$ then
9          break
10     $M \leftarrow M + 1$
11     $\rho_p^n(i) = M$ /* Assign the new flight number */
12     $\rho_p^{nd}(i) \leftarrow start$ /* Departure time */
13     $\rho_p^{no}(i) \leftarrow \rho_p^{co}(i)$ /* Origin airport */
14     $\rho_p^{na}(i) \leftarrow start + \delta_{of}$ /* Landing time */
15     $\rho_p^{nf}(i) \leftarrow \rho_p^{cf}(i)$ /* Destination airport */
16     if $\rho_p^{na}(i) + t_{rp} > RTW_e$ then
17         break
18 return $\rho_p, M$
```

4.4.3 Feasibility Verification Algorithm

The CHARP's next step uses the Flight Verification Algorithm (FVA) (algorithm 3) to traverse the rotation and check if the following constraints are being respected:

- Flight schedule continuity.
- Transit or turn-round time between consecutive flights or transit time between multi leg flights.
- Departure and arrival airport capacity.
- Aircraft arrive on time for maintenance.

The algorithm receives as inputs the aircraft rotation and the airport set. The algorithm will verify the infeasibilities in the rotation regarding, continuity, transit time, airport departure and arrival capacity, and maintenance (lines 1 to 5). In line 6 the algorithm concatenates all the infeasibility sets in a single set. If the latter is not empty it returns the rotation and the index of the first infeasibility, otherwise if the rotation is feasible the algorithm returns an empty set and -1. Consequently, the rotation will be added to the recovery solution, the departure and arrival airport capacity are updated and the algorithm loops to the next aircraft.

Algorithm 3: Feasibility Verification Algorithm

```
   Input: $\rho_p, A$
   Output: $\rho_p, index$
1  $inf1 \leftarrow continuityFunc(\rho_p)$
2  $inf2 \leftarrow ttFunc(\rho_p)$
3  $inf3 \leftarrow depFunc(\rho_p, A)$
4  $inf4 \leftarrow arrFunc(\rho_p, A)$
5  $inf5 \leftarrow maintFunc(\rho_p)$
6  $inf \leftarrow inf1 \cup inf2 \cup inf3 \cup inf4 \cup inf5$ /* All infeasibility indices */
7  if $inf \neq \{\}$ then
8      $index \leftarrow min(inf)$ /* Index of the first infeasible flight in the rotation */
9      return $\rho_p, index$
10 else
11     return $\{\}, -1$
```

4.4.4 Flight Domain Algorithm

The Flight Domain Algorithm (FDA) (algorithm 4) returns the domains for each of the flights, starting at the first infeasible flight until the end of the rotation, the latter designated henceforth the second sub-rotation (ρ_p^+). The FDA searches for each flight, in the second sub-rotation, the domain where it is possible to depart and land without breaching airport departure and arrival capacity constraints. This search is done incrementally, and it will allow delaying the rotation's flights. Added to the latter, and except for those flights that are already subject to a disruptive delay, the algorithm also adds the option to cancel the flight as a delay valued -1.

To implement this procedure, algorithm 4 receives as inputs the second sub-rotation, the airport capacity, the maximum amount of delay, and the delay increment. After initialising the variables to record the rotation's flight domains, singletons, flight intervals, size of the search space, and cancellations (lines 1 to 5) the algorithm loops through the rotation to compute and retrieve their respective values (lines 6 to 25).

The flight domain for the i^{th} leg is initialised in line 7 and if it has been disrupted, or is outside the RTW, its domain assumes a single value of zero thus becoming a singleton (line 9). The function *checkSingletonFunc*, adds zero to the flight domain, checks if there are flight assignments that make the singleton infeasible, and if so adds it to the singleton set. The singleton set will be used by the BTA (algorithm 8), to remove aircraft rotations that make the singleton an infeasible assignment. In line 10 the algorithm adds, the flight's departure time less the turn-round time, and the arrival time to the *intervals* set. In line 11 FDA continues to the next flight in the rotation. If the flight is neither disrupted nor departs outside the RTW, then the algorithm adds the cancellation option (lines 12 to 14). Afterwards it will loop incrementally through the range of delay values starting at zero and add any of them that comply with airport departure and arrival capacity (lines 16 to 22). In lines 23 to 25 algorithm 4 adds the domain to the flight domain set, the intervals to the flight intervals set, and updates the number of combinations. Finally in line 27 algorithm 4 returns the sub-rotation's flight domain and flight intervals.

Each flight domain is coded in a dictionary, where the flight legs correspond to the keys of 4.5. The latter is coded by combining the flight number and flight date. The values consist of each flight leg domain, coded as vectors with the option to cancel flights (-1) and the time delays that respect airport departure and arrival capacity.

$$\begin{aligned} flightDomain = \{'784501/03/08' &: [-1,0,600,660,720,780,840,900], \\ '784002/03/08' &: [-1,0,60,120,180,240,300,360,420,480,600,660,840], \\ '785302/03/08' &: [-1,0,60,120,180,240,300,360,420], \\ '785402/03/08' &: [-1,0,60,240], \\ '788302/03/08' &: [-1,0]\} \end{aligned} \tag{4.5}$$

Consistent with the CSP formulation in Section 4.3, the ARP variables are the flight legs departure time and the domains are the vectors with the delays and option to cancel flights (-1).

Flight intervals in 4.6 is a similar representation of the flight domains, however the former includes information for the second sub-rotation's flight departure, turn-round and arrival time. In equation 4.6, there are several negative intervals, which represent the cancellation intervals for each of the flights.

$$\begin{aligned} intervals = \{'784501/03/08' &: [[-1,-1],[1115,1260],[1715,1860], \\ &[1775,1920],[1835,1980],[1895,2040],[1955,2100],[2015,2160]], \\ '784002/03/08' &: [[-2,-2],[1740,1915],[1800,1975],[1860,2035], \\ &[1920,2095],[1980,2155],[2040,2215],[2100,2275],[2160,2335], \\ &[2220,2395],[2340,2515],[2400,2575],[2580,2755]], \\ '785302/03/08' &: [[-3,-3],[2155,2310],[2215,2370],[2275,2430], \\ &[2335,2490],[2395,2550],[2455,2610],[2515,2670],[2575,2730]], \\ '785402/03/08' &: [[-4,-4],[2320,2495],[2380,2555],[2560,2735]], \\ '788302/03/08' &: [[-5,-5],[2550,2690]]\} \end{aligned} \tag{4.6}$$

Algorithm 4: Flight Domain Algorithm

```
Input: $\rho_p^+, A, maxDelay, delayIncrement$
Output: $flightDomain, singletonList, totalCombos, intervals$
1  $flightDomain \leftarrow \{\}$
2  $singletonList \leftarrow \{\}$
3  $intervals \leftarrow \{\}$
4  $totalCombos \leftarrow 1$
5  $cancel \leftarrow 0$
6  for $\rho_p^+(i)$ in $\rho_p^+$ do
7      $domain(i) \leftarrow \{\}$
       /* Check if the flight is fixed */
8      if $(\rho_p^+(i) \cap \mathscr{D} \neq \{\}) \vee (\rho_p^{+d}(i) < RTW_s) \vee (\rho_p^{+d}(i) > RTW_e)$ then
9          $checkSingletonFunc(\rho_p^+(i), A, singletonList, domain(i))$
10         $intervals[\rho_p^+(i)] \leftarrow intervals[\rho_p^+(i)] \cup \{\rho_p^{+d}(i) - t_{rp}, \rho_p^{+a}(i)\}$
11         continue
12     $domain(i) \leftarrow domain \cup \{-1\}$ /* Add the cancellation to the domain */
13     $cancel \leftarrow cancel - 1$
14     $interval \leftarrow \{\{cancel, cancel\}\}$ /* Create a new cancellation interval */
       /* Loop through the time window of delay */
15     for $delay$ in $range(0, maxDelay, delayIncrement)$ do
16         $\rho_p^{-d}(i) \leftarrow \rho_p^{+d}(i)$
17         $\rho_p^{-a}(i) \leftarrow \rho_p^{+a}(i)$
18         $\rho_p^{-d}(i) \leftarrow \rho_p^{-d}(i) + delay$
19         $\rho_p^{-a}(i) \leftarrow \rho_p^{-a}(i) + delay$
           /* Add the delay to the domain if there is airport capacity for the flight */
20         if $(depFunc(\rho_p^-(i), A) = \{\}) \wedge (arrFunc(\rho_p^-(i), A) = \{\})$ then
21             $domain(i) \leftarrow domain(i) \cup delay$
22             $interval(i) \leftarrow interval(i) \cup \{\rho_p^{-d}(i) - t_{rp}, \rho_p^{-a}(i)\}$
23     $flightDomain[\rho_p^+(i)] \leftarrow domain(i)$ /* Add the delay domain to the flight domain set */
24     $intervals[\rho_p^+(i)] \leftarrow interval(i)$ /* Add the interval domain to the intervals set */
25     $totalCombos \leftarrow totalCombos \times |domain(i)|$
26 return $flightDomain, singletonList, totalCombos, intervals$
```

4.4.5 Arc-consistency and Constraint Propagation

After algorithm 4 returns the flight domains, it is possible to compute the size of the search space, by multiplying the size of each flight domain. The search space consists of the matrix that results from the Cartesian product between the flight domain vectors. Each column represents a flight, and each row has the values for delays or cancellations, such as matrix 4.7:

$$\begin{array}{c}\text{Flight domain}\\ \text{search space}\end{array} = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 0 \\ -1 & -1 & -1 & 0 & -1 \\ \dots & & & & \\ 900 & 840 & 420 & 60 & 0 \\ 900 & 840 & 420 & 240 & -1 \\ 900 & 840 & 420 & 240 & 0 \end{bmatrix} \tag{4.7}$$

Each row of 4.7 represents a solution. During the experiments, it was verified that the size of the search spaces that results from the computation of the Cartesian product of the flight domains can have an order of magnitude up to 10^{25}. The latter value is not tractable in reasonable computing time. Additionally, the delay added to the flight, in each row of the search space, will render new departure and arrival times, some of which overlapping. Hence the intervals search space in 4.8 is a better representation to understand how to perform CPr.

$$\begin{array}{c}\text{Intervals}\\ \text{search space}\end{array} = \begin{bmatrix} [-1,-1] & [-2,-2] & [-3,-3] & [-4,-4] & [-5,-5] \\ [-1,-1] & [-2,-2] & [-3,-3] & [-4,-4] & [2550,2690] \\ [-1,-1] & [-2,-2] & [-3,-3] & [2320,2495] & [-5,-5] \\ \dots & & & & \\ [2015,2160] & [2580,2755] & [2575,2730] & [2380,2555] & [2550,2690] \\ [2015,2160] & [2580,2755] & [2575,2730] & [2560,2735] & [-5,-5] \\ [2015,2160] & [2580,2755] & [2575,2730] & [2560,2735] & [2550,2690] \end{bmatrix} \tag{4.8}$$

The Constraint Propagation Algorithm (CPrA) (algorithm 5), receives as input the *intervals* values in the form of the matrix 4.9.

$$\begin{array}{c}\text{Intervals}\\ \text{values}\end{array} = \begin{bmatrix} [[-1,-1],[1115,1260],[1715,1860],[1775,1920], \\ [1835,1980],[1895,2040],[1955,2100],[2015,2160]], \\ \dots \\ [[-4,-4],[2320,2495],[2380,2555],[2560,2735]], \\ [[-5,-5],[2550,2690]] \end{bmatrix} \tag{4.9}$$

Algorithm 5 uses *cpmpy* and *cpmpy_hakank* libraries to flat the sets and perform CP. The first step consists of flattening the *intervalsValues* matrix into a *cmp_array* (lines 1 to 5). The algorithm afterwards retrieves the maximum and minimum value in the flatten set (lines 6 and 7), and the size of *intervalsValues*, (line 8). Each of *intervalsValues* vector's size is recorded in *lens* (lines 9 to 11). The CP model is initialised in line 12, same as the array of interval variables x in line 13. Considering equation

4.6 $x[i]$ is the selected interval ($i^{th} \in 0..4$), each with a maximum size of 13. Similarly, it is possible to recognise from 4.6 that each set of values has specific lengths, hence in lines 14 and 15 the algorithm adds to the model length constraints for each $x[i]$ interval variable. In lines 16 and 17 interval variables $starts/ends$ are initialised. These variables define the start/end of the i^{th} interval for each $starts[i]/ends[i]$ interval variable. The algorithm afterwards adds the values of the selected intervals (lines 18 to 20) to the model using the $Element$ object. In lines 21 to 24 the algorithm adds to the model the constraint that ensures that the i^{th} selected interval does not overlap with the rest of the intervals (the j^{th} interval). Finally, in lines 25 and 26 the algorithm solves the model and returns the reduced intervals search space.

4.4.6 Lower Heuristic Algorithm

As referred subsection 4.4.5 the search space can have orders of magnitude up to 10^{25}. To tackle these situations the CHARP uses two algorithms, the LHA and the UHA. The LHA finds solutions for search spaces whose order of magnitude is below the lower bound. On the other hand, if the order of magnitude of the size search space is greater than the upper bound the CHARP uses the UHA.

The lower bound is initialised by default at 4×10^4. The LHA loops through every row of the reduced intervals search space in order to find the optimal solutions that minimise the number of cancelled flights and the total amount of delay. Algorithm 6 receives as inputs the original rotation, the index of the first infeasibility and the reduced intervals search space. The algorithm will recover the infeasible rotation and will output the recovered one. It starts by initialising the best solution and reconverting the intervals search space into flight domains search space. Henceforward it will loop through the search space to find its best value. In lines 4 and 5 algorithm 6 sums the cancelled flights and the total amount of delay for each row. In line 6 it assigns them to the new solution. The algorithm compares the new solution with the best one and if the new one is not better it will iterate (lines 7 to 11). Based on the values coded in *row*, in line 12 the algorithm creates the new rotation by cancelling or delaying flights in the original rotation ρ_p. Afterwards if any constraint is violated the LHA continues (lines 13 and 14), else the new feasible solution updates the best solution (lines 15). After traversing the entire search space, the algorithm updates the original aircraft rotation with the optimal solution and returns it (lines 16 and 17).

4.4.7 Upper Heuristic Algorithm

The main purpose of the UHA is to recover the aircraft rotation part (infeasible second sub-rotation), starting from the index of the first infeasibility, when the search space is not tractable in a reasonable computing time. To cover the intractable search space the UHA upper bound is initialised by default at 3×10^{12}. The UHA decomposes the infeasible second sub-rotation into partial sub-rotations whose search space size is lower than the lower bound defined for the LHA. The UHA loops through every partial sub-rotation until it finds a feasible solution for the entire second sub-rotation. Although this procedure may not return an optimal solution, it can find feasible solutions in a reasonable computing time. Algorithm 7 receives as inputs the infeasible rotation, the index of the first infeasibility, and the flight domains. The reason for using flight domains is related to the fact that the computing time proved to be less than using reduced intervals, because this method is relatively expensive for very compact interval sets. The recovered sub-rotation and the lower index of the partial sub-rotation are initialised,

Algorithm 5: Constraint Propagation Algorithm

```
Input: $intervalsValues$
Output: $reducedIntervals$
1  $intervalsFlatten \leftarrow \{\}$
   /* Flat all the intervals into an array */
2  for $interval$ in $intervalsValues$ do
3      $intervalsFlatten \cup cmp\_array(flatten\_lists(interval))$
4  $intervalsFlatten \leftarrow cpm\_array(intervalsFlatten)$
   /* We need all values to create the domains of the selected interval values */
5  $allValues \leftarrow flatten\_lists(intervalsFlatten)$
6  $maxVal \leftarrow max(allValues)$ /* Max. value in the intervals */
7  $minVal \leftarrow min(allValues)$ /* Min. value in the intervals */
8  $n \leftarrow |intervalsValues|$ /* Interval's size */
9  $lens \leftarrow \{\}$
   /* Get the size of each interval */
10 for $interval$ in $intervalsValues$ do
11     $lens \cup \{interval\}$
12 $model \leftarrow Model()$ /* Instantiate the model */
   /* x[i] is the selected interval for the i'th interval list */
13 $x \leftarrow intvar(0, max(lens), shape \leftarrow n, name \leftarrow "x")$
   /* Reduce the domain (the possible values) of each interval list since they have
      different lengths */
14 for $i$ in $range(n)$ do
15     $model \cup \{x[i] < lens[i]\}$
   /* starts[i] is the start value of the i'th selected interval */
16 $starts \leftarrow intvar(minVal, maxVal, shape \leftarrow n, name \leftarrow "starts")$
   /* ends[i] is the end value of the i'th selected interval */
17 $ends \leftarrow intvar(minVal, maxVal, shape \leftarrow n, name \leftarrow "ends")$
   /* Main constraints: */
   /* Pick exactly one of the intervals from each intervals list */
   /* Ensure that there are no overlaps between any of the selected intervals */
18 for $i$ in $range(n)$ do
       /* Use Element to obtain the start and end values of the selected interval */
19     $model \cup \{starts[i] = Element(intervalsFlatten[i], x[i] \times 2 + 0),$
20     $ends[i] = Element(intervalsFlatten[i], x[i] \times 2 + 1)]\}$
   /* Ensure that the i'th selected interval doesn't overlap w the rest of the intervals
      (the j'th interval) */
21 for $i$ in $range(n)$ do
22     for $j$ in $range(i+1, n)$ do
           /* Ensure that the start value of one interval is not inside the other interval */
23         $model \cup \{\neg((starts[i] > starts[j]) \wedge (starts[i] < ends[j])),$
24         $\neg((starts[j] > starts[i]) \wedge (starts[j] < ends[i]))]\}$
25 $reducedIntervals \leftarrow ortools\_wrapper2(model, x, get\_solution)$ /* Get the solutions */
26 return $reducedIntervals$
```

Algorithm 6: Lower Heuristic Algorithm

```
Input: $\rho_p, index, reducedIntervals$
Output: $\rho'_p$
1  $bestSol \leftarrow \{\}$
   /* Convert intervals in delays                                    */
2  $flightDomains \leftarrow convert2FlightDomainsFunc(reducedIntervals)$
3  for $row$ in $flightDomains$ do
4      $noCancel \leftarrow \sum_i row(i)$, if $row(i) = -1$ /* Get the amount of cancellations */
5      $totalDelay \leftarrow \sum_i row(i)$, if $row(i) \neq -1$ /* Get the amount of delay */
6      $newSol \leftarrow \{noCancel, totalDelay, row\}$
       /* Compare the new solution w/ the best solution               */
7      if $bestSol \neq \{\}$ then
8          if $newSol[0] < bestSol[0]$ then
9              continue
10         if $(newSol[0] = bestSol[0]) \wedge (newSol[1] \geq bestSol[1])$ then
11             continue
12     $\rho'_p \leftarrow solRotFunc(\rho_p, index, row)$ /* Assign the new temporary solution */
       /* Check if the new temporary solution is feasible             */
13     if $allContraints(\rho'_p) \neq \{\}$ then
14         continue
15     $bestSol \leftarrow newSol$
16 $\rho'_p \leftarrow solRotFunc(\rho_p, (bestSol[2]))$ /* Assign the new best solution */
17 return $\rho'_p$
```

the latter with the value of the index of the first infeasibility. In line 3 the algorithm uses the function *upperIndexFunc* computes the upper index for the partial sub-rotation that will be recovered and extracts the corresponding partial flight domains. In line 4 the algorithm computes the search space and in line 5 reduces the flight domains for the partial sub-rotation that is to be recovered. From lines 6 to 28 the algorithm will loop through every partial sub-rotation, recovering each one of them. The algorithm finds the best solution to recover each partial sub-rotation using the same method as the LHA (lines 8 to 20). In line 21 the partial sub-rotation part between the lower and the upper index is updated with the best solution. If the upper index equals the size of the recovered rotation this means the recovery procedure is terminated and it returns the recovered rotation (lines 22 and 23) otherwise, the algorithm assigns the previous upper index to the current lower index, computes the new upper index, new partial flight domains, the search space and updates the remaining flight domains (lines 24 to 28).

It is important to notice that in order to optimise the looping over the search space, the LHA and UHA compare the current solution with the new one and if the latter is not better, they will not proceed to test feasibility, thus saving significant computation time. As for the overarching algorithm, in every iteration it loops through the aircraft list and based on the size of the search space decides which heuristic will find the solution to recover the infeasible rotations while minimising the number of cancelled flights and the total amount of delay. However, if the search space size is above the lower bound or below the upper bound, the infeasible rotation will not be recovered. To overcome this situation the algorithm iterates the aircraft loop using the list of aircraft left with infeasible rotations, increments the lower bound, and decrements the upper bound. This procedure results in a pincer movement that will entrap the entire

Algorithm 7: Upper Heuristic Algorithm

```
Input: $\rho_p, index, flightDomains$
Output: $\rho'_p$
1  $\rho'_p \leftarrow \rho_p$
2  $lIndex \leftarrow index$
3  $uIndex, partialFlightDomains \leftarrow upperIndexFunc(lIndex, \rho'_p, flightDomains)$
4  $matrix \leftarrow productFunc(partialFlightDomains.values())$ /* Assign the partial sub-rotation */
5  $removeFlightDomainsFunc(flightDomains, \rho'_p(i) \forall i \in [uIndex, |\rho_p|])$
6  while $True$ do
7      $bestSol \leftarrow \{\}$
       /* Recover the partial sub-rotation */
8      for $row$ in $matrix$ do
9          $noCancel \leftarrow \sum_i row(i)$, if $row(i) = -1$;
10         $totalDelay \leftarrow \sum_i row(i)$, if $row(i) \neq -1$
11         $newSol \leftarrow [noCancel, totalDelay, row]$
12         if $bestSol \neq \{\}$ then
13             if $newSol[0] < bestSol[0]$ then
14                 continue
15             if $(newSol[0] = bestSol[0]) \wedge (newSol[1] \geq bestSol[1])$ then
16                 continue
17         $newPartialRotationFunc(row, \rho'_p(i) \forall i \in [lIndex : uIndex])$
18         if $allContraints(\rho'_p) \neq \{\}$ then
19             continue
20         $bestSol \leftarrow newSol$
21     $newPartialRotationFunc(bestSol[2], \rho'_p(i) \forall i \in [lIndex : uIndex])$
22     if $uIndex = |\rho'_p|$ then
23         return $\rho'_p$
24     else
           /* Get the new partial sub-rotation */
25         $lIndex \leftarrow uIndex$ /* Assign the upper index to the lower index */
26         $uIndex, partialFlightDomains \leftarrow upperIndexFunc(lIndex, \rho'_p, flightDomains)$
27         $matrix \leftarrow product(partialFlightDomains.values())$ /* Assign the new partial sub-rotation */
28         $removeFlightDomainsFunc(flightDomains, \rho'_p(i) \forall i \in [uIndex, |\rho'_p|])$
```

search space, making sure that every infeasible sub-rotation is recovered, figure 4.1.

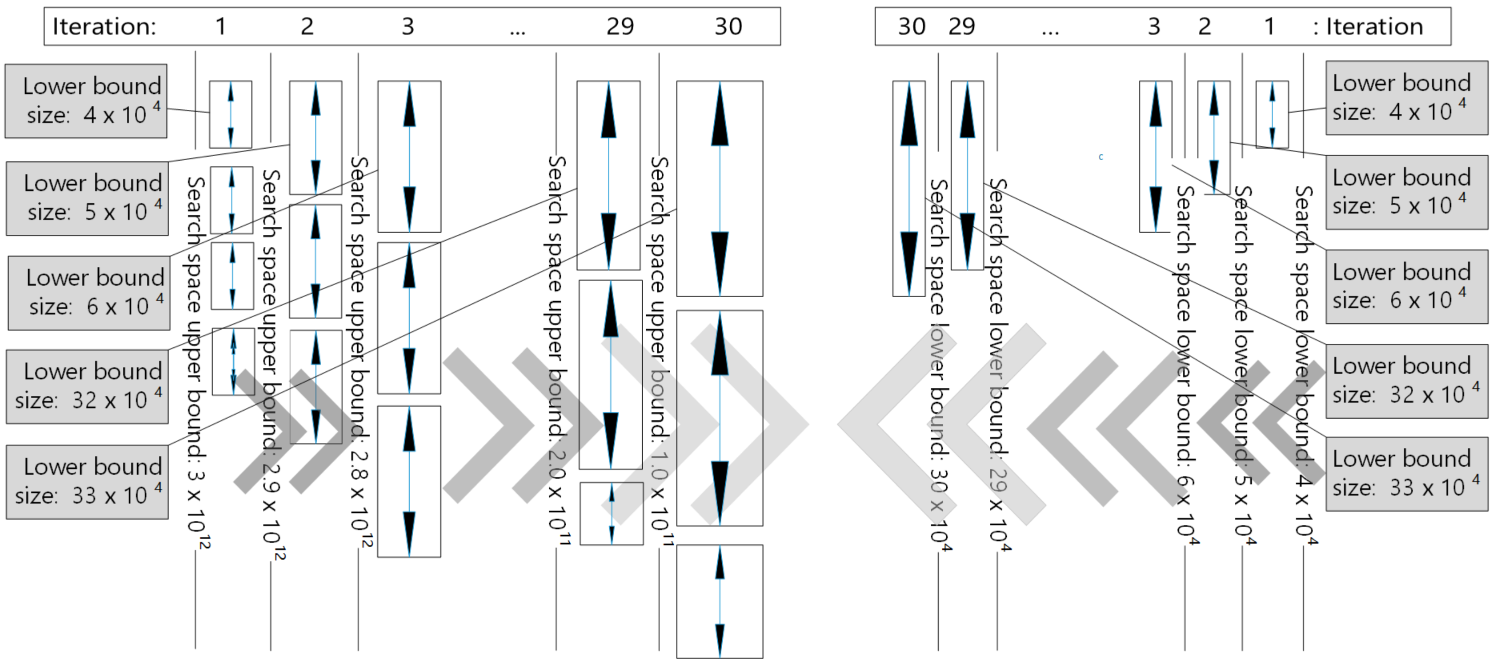

Iteration: 1 2 3 ... 29 30
30 29 ... 3 2 1 : Iteration
Lower bound size: 4 x 10 4
Lower bound size: 5 x 10 4
Lower bound size: 6 x 10 4
Lower bound size: 32 x 10 4
Lower bound size: 33 x 10 4
Search space upper bound: 3 x 10^12
Search space upper bound: 2.9 x 10^12
Search space upper bound: 2.8 x 10^12
Search space upper bound: 2.0 x 10^11
Search space upper bound: 1.0 x 10^11
Search space lower bound: 30 x 10^4
Search space lower bound: 29 x 10^4
Search space lower bound: 6 x 10^4
Search space lower bound: 5 x 10^4
Search space lower bound: 4 x 10^4

Fig. 4.1 Pincer Meta-heuristic (PMH)

4.4.8 Backtracking Algorithm

In CSP it is common to find variables that have domain size 1, such variables are designated by singletons. Since both heuristics are based on constraint satisfaction, the rotations that are known in advance having variables with domain size 1, are handled first. In the case of rotations with scheduled maintenance, the algorithm treats them as a flight without turn round time and with the same origin and destination. Thus, the first elements of the aircraft list have scheduled maintenance. The flights that are disrupted by delays are designated as fixed flights and they too cannot be moved. In this case, the domain is a singleton consisting of value $\{0\}$ and if this value is infeasible because there is no available departure and/or arrival airport capacity, the BTA removes the rotation of another aircraft that can release the necessary airport capacity.

Algorithm 8 receives as inputs the singleton set, the airport capacity, the aircraft set whose rotations are feasible, the ARP current solution, the aircraft rotation with the singleton(s), and the index of the first infeasibility. The algorithm will return the ARP solution without the rotation that will allow the singleton to become feasible and the updated aircraft solution set (line 20). Algorithm 8 will loop while the singleton set is not empty (line 1). Afterwards it verifies if the first singleton's infeasibility is on the departure and/or in the arrival airport capacity (lines 2 and 11). Depending on the situation the algorithm computes the airport time slot for the departure/arrival (lines 3, 4, and lines 12, 13) and based on the origin/destination airport (lines 5 and 14), searches the flights and aircraft to cancel (lines 6, 7 and lines 15, 16). The algorithm will then remove the aircraft from the solution list, and the respective rotation from the ARP solution (lines 8, 9 and lines 17, 18). Finally, algorithm 4 checks if there are any more infeasible singletons (lines 10 to 19).

Algorithm 8: Backtracking Algorithm

```
   Input: $singletonList, A, aircraftSolList, solutionARP, \rho_p, index$
   Output: $solutionARP, aircraftSolList$
   /* Loop as long as there are singletons to be removed                     */
1  while $singletonList \neq \{\}$ do
       /* Check if the airport departure capacity causes the singleton        */
2      if $singletonList(0) =' dep'$ then
           /* Assign the index of the time slot                               */
3          $startInt \leftarrow 60 \times int(singleton^d(0)/60)$
4          $endInt \leftarrow startInt + 60$
5          $origin \leftarrow singleton^o(0)$ /* Assign the origin airport        */
6          $flight2Cancel \leftarrow solutionARP[(origin, startInt, endInt)]$
7          $airc2Cancel \leftarrow updateMulti(flight2Cancel, A, solutionARP])$
8          $aircraftSolList.pop(airc2Cancel)$ /* Remove the aircraft from the solution */
9          $solutionARP.pop(airc2Cancel)$ /* Remove the rotation from solution */
10         $flightRanges, singletonList, totalCombos \leftarrow domainFlights(\rho_p(i) \forall i \in$
             $[index, |\rho_p|], A, index)$
       /* Check if the airport arrival capacity causes the singleton          */
11     if $singleton(0) =' arr'$ then
           /* Assign the index of the time slot                               */
12         $startInt \leftarrow 60 \times int(singleton^a(0)/60)$
13         $endInt \leftarrow startInt + 60$
14         $destination \leftarrow singleton^f(0)$ /* Assign the destination airport */
15         $flight2Cancel \leftarrow solutionARP[(destination, startInt, endInt)]$
16         $airc2Cancel \leftarrow updateMulti(flight2Cancel, A, solutionARP)$
17         $aircraftSolList.pop(airc2Cancel)$ /* Remove the aircraft from the solution */
18         $solutionARP.pop(airc2Cancel)$ /* Remove the rotation from solution */
19         $flightRanges, singletonList, totalCombos \leftarrow domainFlights(\rho_p(i) \forall i \in$
             $[index, |\rho_p|], A, index)$
20 return $solutionARP, aircraftSolList$
```

4.4.9 Taxi Flights Algorithm

After finding a feasible solution the algorithm verifies if continuity is breached between the origin airport and the first leg of the recovered rotation. Algorithm 9 receives as inputs the set of aircraft disruptions, the set of distances, the origin airport, the recovered rotation, and the maximum flight number. As for the output, it returns the updated recovered rotation and the maximum flight number. Algorithm 9 will either create a taxi flight to connect the origin airport and the recovered rotation or cancel flights in the recovered rotation until the continuity infeasibility is removed.

In line 1 algorithm 9 retrieves the time slots where there is available departure capacity in the aircraft origin airport. Afterwards the algorithm will loop through the flight legs of the recovered rotation and if their origin is the same as the origin airport it returns the updated recovered rotation and the maximum flight number (lines 2 to 4). In line 6 the algorithm computes the distance from the origin airport and the origin of the flight in the recovered rotation. In line 7 the algorithm extracts the upper slots from the origin slots by subtracting the flight's departure time in the recovered rotation, the distance, and the turn round time. In line 8 the upper slots and the aircraft breakdown period are subtracted from the origin slots in order to retrieve the lower slots from where the aircraft can depart. If there are no available departure slots the algorithm cancels the flight in the recovered rotation and continues to the next flight (lines 9 to 11). If there are available departure slots at the origin, the algorithm tries to find destination slots with available airport arrival capacity. To achieve the latter the algorithm finds the destination slots with available capacity, the upper destination slots (lines 12 and 13). By subtracting the latter from the former and the aircraft breakdown period, it determines the lower destination slots (line 14). If there are no lower destination slots, the algorithm cancels the flight in the recovered rotation and continues to the next flight. Since A is a dictionary, it is necessary to extract the destination intervals and initialise the destination index i and the time offset from which the flight departs and arrives in feasible airport slots (lines 18 to 22). The algorithm will then loop through the origin slots and in line 24 it will initialise the object obj with the starting and end time of the original lower slots plus the distance, and in line 25 determines the index of intersection in the destination slot and the offset. If the index i is different from -1 the algorithm will create the taxi flight and add it to the recovered rotation (lines 26 to 34).

4.4.10 The CHARP Algorithm

The CHARP algorithm receives as inputs the sets for the aircraft, flight and aircraft disruptions, flights, rotations, airports, lower bound, upper bound, lower increment and upper increment, algorithm 10. After implementing its logic the CHARP algorithm returns the solution with the recovered rotations. After receiving its inputs algorithm 10 initialises the aircraft set by removing the $TranspCom$ since they are not accounted for airport capacity, line 1. The algorithm afterwards initialises the ARP's solution, line 2. From line 3 to 39 algorithm 10 loops until all aircraft rotations are recovered. In line 4 the algorithm loops through the aircraft set that have aircraft whose rotation (line 5) may not be recovered. If there are flight or aircraft disruption in the rotation the algorithm adds new flights (line 6 and 7). In line 8 the algorithm checks if the rotation is infeasible or not. In the best case scenario, the aircraft rotation is feasible, hence the algorithm adds it to the ARP's solution. Additionally, the algorithm also adds the aircraft p to the aircraft solution set and updates the airport capacity (lines 35 to 37). On the other hand, if the aircraft rotation ρ_p is infeasible, the algorithm tries to recover it. Using algorithm 4, in line 10, the algorithm

Algorithm 9: Taxi Flights Algorithms

```
Input: $\mathscr{B}, \Delta, O_p, \rho'_p, A, M$
Output: $\rho'_p, M$
$originSlots \leftarrow A(O_p)$ if $c_a^{jh} > noDep$ /* Origin airport time slots w/ departure capacity */
for $\rho'_p(i)$ in $\rho'_p$ do
    if $\rho{\prime}_p^o(i) = O_p$ /* Continuity is restored */
    then
        return $\rho'_p$, M
    $\delta_{of} \leftarrow \Delta(O_p, \rho_p^{\prime o}(i))$ /* Calculate the block time from the origin to the destination
        airport */
    $originSlotsUpper \leftarrow originSlots$ if $endInt > \rho_p^{\prime d}(i) - \delta_{of} - t_{rp}$
    $originSlotsLower \leftarrow originSlots - originSlotsUpper - [\mathscr{B}_p^s, \mathscr{B}_p^e]$ /* Airport time slots w/
        departure capacity */
    if $originSlotsLower = \{\}$ then
        cancel($\rho'_p(i)$) /* Cancel the flight in the recovered rotation */
        continue
    $destinationSlots \leftarrow A(\rho_p^{\prime o})$ if $c_a^{lh} > noArr$
    $destinationSlotsUpper \leftarrow destinationSlots$ if $endInt > \rho_p^{\prime d}(i) - t_{rp}$
    $destinationSlotsLower \leftarrow destinationSlots - destinationSlotsUpper - [\mathscr{B}_p^s, \mathscr{B}_p^e]$ /* Airport
        time slots w/ arrival capacity */
    if $destinationSlotsLower = \{\}$ then
        cancel($\rho'_p(i)$) /* Cancel the flight in the recovered rotation */
        continue
    $destIntervals \leftarrow \{\}$
    $i \leftarrow -1$
    $offset \leftarrow -1$
    for $x \in destinationSlotsLower$ do
        $destIntervals \leftarrow destIntervals \cup [x^s, x^e]$
    /* Loop through origin airport w/ available capacity */
    for $os(i)$ in $originSlotsLower$ do
        $obj \leftarrow interval(os^s, os^e) + \delta_{of}$
        $i, offset \leftarrow obj.findIntersection(destIntervals)$ /* offset fits the flight between
            the origin and the destination airport */
        /* Check if the block time is inside a departure and a time slot */
        if $i \neq -1$ then
            /* Create taxi flight */
            $taxiFlight^o \leftarrow O_p$ /* Taxi flight origin airport */
            $taxiFlight^d \leftarrow os[startInt] + offset$ /* Taxi flight departure time */
            $taxiFlight^f = \rho_p^{\prime o}(i)$ /* Taxi flight destination airport */
            $taxiFlight^a = taxiFlight^d + \delta_{of}$ /* Taxi flight arrival time */
            $taxiFlight[flightNumber] = M$ /* Taxi flight number */
            $M \leftarrow M + 1$ /* Increment flight number */
            $\rho'_p \leftarrow \rho^r prime_p + taxiFlight$ /* Add taxi flight to the rotation */
            return $\rho'_p, M$
```

computes the flight domains, the singleton set, the size of the search space and the flight intervals. If the size of the search space is bigger than the upper bound (line 11), or lower than lower bound the algorithm recovers the infeasible rotation. Otherwise, the algorithm 10 loops to the next aircraft. For both paths of recovery the algorithm always backtracks if there are singletons (lines 12 to 14 and lines 22 to 24), adds taxi flights if necessary (lines 16 and 17, and lines 27 and 28), and finally adds the recovered rotation to the ARP's solution, the aircraft p to the aircraft solution set and updates the airport capacity (lines 18 to 20 and lines 29 to 31). The difference between the paths of recovery consists in using flight domains for the UHA (line 15), or using the reduce intervals for the LHA (lines 25 and 26).

4.5 Computational Results

This section presents the CHARP results for the recovery cost, computing time, and the influence of adding new and taxi flights for the overall thirty two data instances.

It is important referring that the results presented were obtained after a long research procedure. The initial set of experiments were done in an Intel® Xeon Gold CPU @ 2.3GHz box with 72 CPU(s), 128 GiB of RAM box and did not included CPr (algorithm 5). Computing time for the worst case scenarios was in the order of magnitude 10^3. The recovery procedure was also tried using a genetic algorithm, instead of using the UHA (algorithm 7). The former proved to be inappropriate for search spaces of order of magnitude 10^{25}. Not only the solution took a long time to improve, but the improvement also proved to be very weak.

The results presented hereafter are for the CHARP, algorithm 10, and were obtained running in parallel the 32 data instances through the maximum delay time window in 4.10. Each batch of the 32 data instances was processed in a specific CPU and during processing resource usage was monitored, figure 4.2.

$$Time\,window = [720, 780, 840, 900, 960, 1020, 1080, 1140, 1200, 1260] \tag{4.10}$$

Each of the values of 4.10 consist of the *maxDelay* parameter that is used in algorithm 4 to compute the flight domain.

Apart from computing time the results presented for cost, amount of delay, cancelled flights, new flights and taxi flights are the aggregated value of each data instance recovery. The computing time is collected when the last data instance finishes the recovery.

The remainder of this section consists of, Subsection 4.5.1 describing the default scenario for which it is studied the impact of adding new flights and taxi flights. Since the heuristic performs a pincer movement Subsection 4.5.2 presents the results for different speeds by changing the decremental and incremental steps.

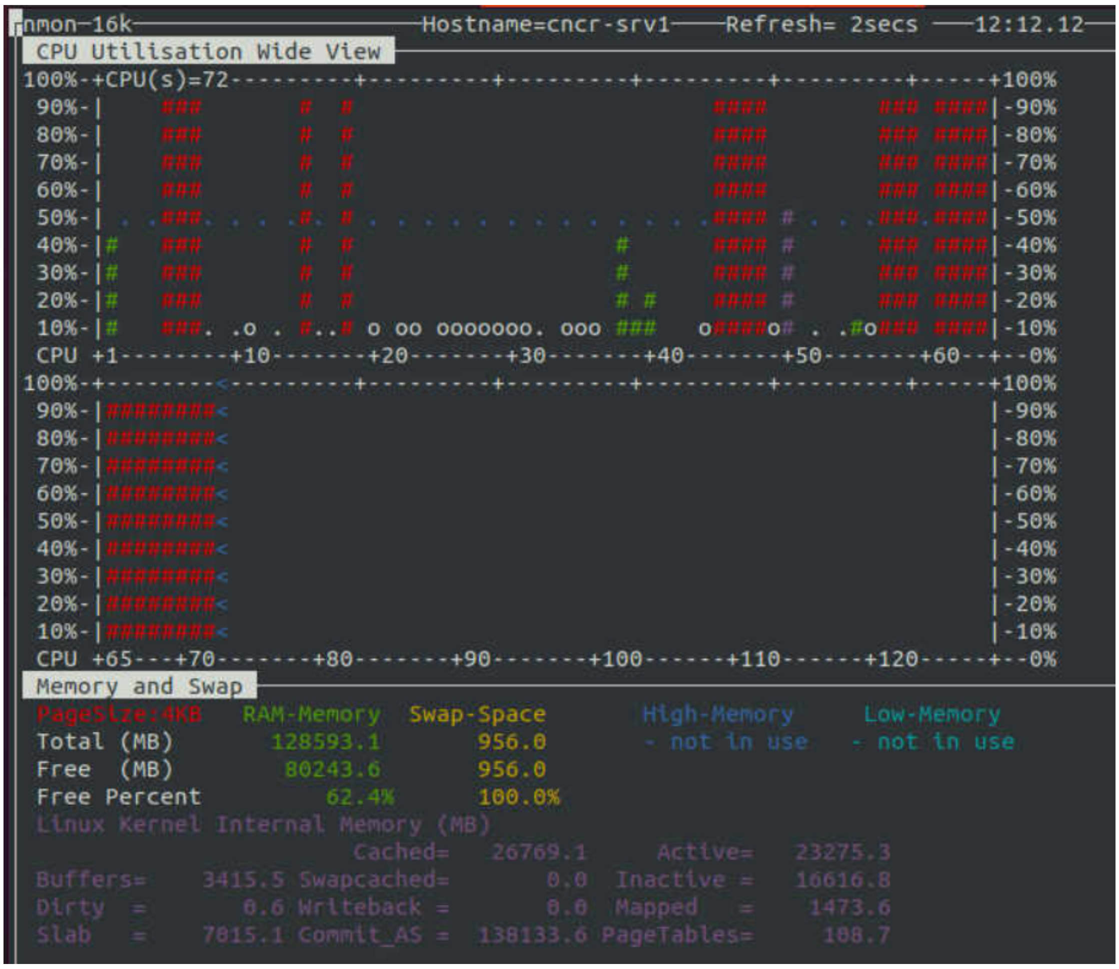
4.5.1 Default scenario

The default scenario is defined in table 4.4:

Figure 4.3, represents the results for "Flights added" which considers both new flights and taxi flights to the CHARP, "No flights added" where the CHARP has no new or taxi flights added. "Only new flights added" and "Only taxi flight added", are self explanatory.

Algorithm 10: CHARP Algorithm

```
Input: $P, \mathscr{D}, \mathscr{B}, F, \rho, A, \beta_l, \beta_u, \varepsilon_l, \varepsilon_u$
Output: $solutionARP$
$aircraftList \leftarrow P \setminus \{TranspCom\}$
$solutionARP \leftarrow \{\}$
while $|aircraftSolList| \neq |aircraftList|$ do
    for $p$ in $aircraftList$ do
        $\rho_p \leftarrow F \cap \rho$
        if $(\mathscr{D}(\rho_p) \neq \{\}) \vee (\mathscr{B}(\rho_p) \neq \{\})$ then
            $\rho_p, M \leftarrow Algorithm\,2$ /* Add new flights to replace cancelled */
        $\rho_p, index \leftarrow Algorithm\,3$ /* Check rotation feasibility */
        if $index \neq -1$ then
            /* Get flight domains, singleton list, search space size, and intervals */
            $flightDomain, singletonList, totalCombos, intervals \leftarrow Algorithm\,4$
            if $totalCombos > \beta_u$ /* Check if the search space is above the upper bound */
             then
                while $|singletonList| > 0$ /* Remove singletons */
                 do
                    $solutionARP, aircraftSolList \leftarrow Algorithm\,8$
                    $flightDomain, singletonList, totalCombos, intervals \leftarrow Algorithm\,3$
                $\rho\prime_p \leftarrow Algorithm\,7$ /* Recover the rotation w/ the UHA */
                if $O_p \neq \rho'_p(0)$ /* Check continuity */
                 then
                    $\rho'_p, M \leftarrow Algorithm\,9$ /* Create taxi flight */
                $solutionARP \cup \rho'_p$ /* Add the recovered rotation to the solution */
                $aircraftSolList \cup p$ /* Add the aircraft solution list */
                $updateAirportCapacityFunc(A)$ /* Update airport capacity */
            else if $totalCombos < \beta_l$ /* Check if the search space is bellow the lower bound
               */
             then
                while $|singletonList| > 0$ /* Remove singletons */
                 do
                    $solutionARP, aircraftSolList \leftarrow Algorithm\,8$
                    $flightDomain, singletonList, totalCombos, intervals \leftarrow Algorithm\,3$
                $reducedIntervals \leftarrow Algorithm\,5$ /* Constraint propagation algorithm */
                $\rho'_p \leftarrow Algorithm\,6$ /* Recover the rotation w/ LHA */
                if $O_p \neq \rho'_p(0)$ /* Check continuity */
                 then
                    $\rho'_p, M \leftarrow Algorithm\,9$ /* Create taxi flight */
                $solutionARP \cup \rho'_p$ /* Add the recovered rotation to the solution */
                $aircraftSolList \cup p$ /* Add the aircraft solution list */
                $updateAirportCapacityFunc(A)$ /* Update airport capacity */
            else
                continue
        else
            $solutionARP \cup \rho_p$ /* Add the rotation to the solution */
            $aircraftSolList \cup p$ /* Add the aircraft to the solution list */
            $updateAirportCapacityFunc(A)$ /* Update airport capacity */
    $\beta_u \leftarrow \beta_u - \varepsilon_u$ /* Decrement the upper bound */
    $\beta_l \leftarrow \beta_l + \varepsilon_l$ /* Increment the lower bound */
return $solutionARP$
```



```
┌nmon─16k───────────────────Hostname=cncr-srv1───Refresh= 2secs ───12:12.12──
 CPU Utilisation Wide View
100%-+CPU(s)=72---------+---------+---------+---------+---------+-----+100%
 90%-|    ###       #  #                          ####        ### ####|-90%
 80%-|    ###       #  #                          ####        ### ####|-80%
 70%-|    ###       #  #                          ####        ### ####|-70%
 60%-|    ###       #  #                          ####        ### ####|-60%
 50%-| . .###. . . .#. # . . . . . . . . . . . . .#### # . . .###.####|-50%
 40%-|#   ###       #  #                   #      #### #      ### ####|-40%
 30%-|#   ###       #  #                   #      #### #      ### ####|-30%
 20%-|#   ###       #  #                   # #    #### #      ### ####|-20%
 10%-|#   ###. .o . #..# o oo ooooooo. ooo ###   o####o# . .#o### ####|-10%
 CPU +1--------+10-------+20-------+30-------+40-------+50-------+60--+--0%
100%-+--------<---------+---------+---------+---------+---------+-----+100%
 90%-|########<                                                       |-90%
 80%-|########<                                                       |-80%
 70%-|########<                                                       |-70%
 60%-|########<                                                       |-60%
 50%-|########<                                                       |-50%
 40%-|########<                                                       |-40%
 30%-|########<                                                       |-30%
 20%-|########<                                                       |-20%
 10%-|########<                                                       |-10%
 CPU +65---+70-------+80-------+90-------+100------+110------+120-----+--0%
 Memory and Swap ────────────────────────────────────────────────────────────
 PageSize:4KB   RAM-Memory  Swap-Space       High-Memory     Low-Memory
 Total (MB)       128593.1       956.0       - not in use    - not in use
 Free  (MB)        80243.6       956.0
 Free Percent        62.4%      100.0%
 Linux Kernel Internal Memory (MB)
                        Cached=   26769.1     Active=   23275.3
 Buffers=    3415.5 Swapcached=       0.0  Inactive =   16616.8
 Dirty  =       0.6 Writeback =       0.0  Mapped   =    1473.6
 Slab   =    7815.1 Commit_AS =  138133.6 PageTables=     108.7
```

Fig. 4.2 Resources usage while running the CHARP

Table 4.4 Default Scenario

delayIncrement [min]	60
β_u	3.0×10^{12}
ε_u	1.0×10^{11}
β_l	4.0×10^{4}
ε_l	1.0×10^{4}

It is possible to observe that the best aggregate cost is always lower for the heuristic that has flights added (new flights and taxi flights). This result was expected since adding new flights or taxi flights consists of adding new arcs which in turn can increase the flow of the flight network.

It is possible to observe the influence of taxi flights in computing time, which for the best case (time window 840 minutes (14 hours)) is 220.9 seconds. The latter can be explained on a series of cascading effects: adding taxi flights will decrease the available airport capacity, which in turn will decrease the size of the domains hence resulting in a smaller size search space which needs less time to be traversed.

It is observable that extending the time window increases the total amount of delay returned by the CHARP. This results from the fact that there are more time slots available to delay the flight. As for the total number of cancellations, it is lowest for the heuristic that has new flights and taxi flights added. The tendency of this results is consistent with the aggregate cost. Finally, the total number of new flights is in the order of magnitude of 125 and the total number of taxi flights is in the order of magnitude of 200. Both values do not change substantially with the size of the time window.

Regarding the addition of new flights, the best results are obtained for the time windows of 780 minutes (13 hours) for "Flights added", and for the 900 minutes (15 hours) for "Only new flights", as shown in figure 4.4. The maximum number of taxi flights, 209, occurs for the 1140 minutes time window.

Figure 4.4 draws the Pareto front and it is possible to verify that the solution's best result for aggregate cost do not coincide with computing time. The best result for aggregate cost, 6.235×10^8, is for the category "Flights added" and for the time window of 900 minutes (15 hours) of maximum delay. On the other hand, the best result for computing time, 220.9 seconds, is achieved for the category "Only taxi flights added" and for the time window of 840 minutes (14 hours) of maximum delay. One can conclude that the aggregate cost is in conflict with computing time. Looking back, one can conclude that this observation results from the added complexity of adding new flights.

4.5.2 Impact of the Pincer Speed

The CHARP uses the UHA and the LHA to perform a pincer movement over the search space. This closing movement can be performed at different speeds. Since Subsection 4.5.1 demonstrated that the category "Flights added" has the best results for aggregate cost, this subsection investigates if its computing time can be improved.

To study the impact of the pincer speed we define the set of increment and decrement values in table 4.5.

Table 4.5 Pincer speeds

	ε_l	ε_u
Speed 0.5	0.5×10^4	0.5×10^{11}
Speed 1	1.0×10^4	1.0×10^{11}
Speed 2	2.0×10^4	2.0×10^{11}
Speed 3	3.0×10^4	3.0×10^{11}
Speed 4	4.0×10^4	4.0×10^{11}
Speed 5	5.0×10^4	5.0×10^{11}
Speed 6	6.0×10^4	6.0×10^{11}

The ε_l and ε_u values are used to increment and decrement the lower and the upper bound in each iteration of the CHARP.

In figure 4.5 it is possible to observe that the minimum aggregate cost is improved for speed 5 in the 900 minutes (15 hours) time window, taking 236 seconds. Once again it does not coincide with the best results for computing Time. The latter was obtained for speed 2, 205.9 seconds.

For the total amount of delay, it increases with the time window and that the different speeds observe the same pattern of behaviour. In relation to the total number of flight cancellations, the values have a sharp decrease from 720 to 780 minutes (12 to 13 hours), after which they stabilise between a minimum of 2076 at speed 1 for the 960 minutes (16 hours) time window. As for the number of new flights, they increase steadily with the size of the time window. Finally, in relation to the number of taxi flights we observe again a sharp decrease in value for the time window of 900 minutes (15 hours), the remaining values distribute themselves between 196 at speed 3 for the 1260 minutes (21 hours) time window and 226 at speed 3 for the 1020 (17 hours) minutes time window.

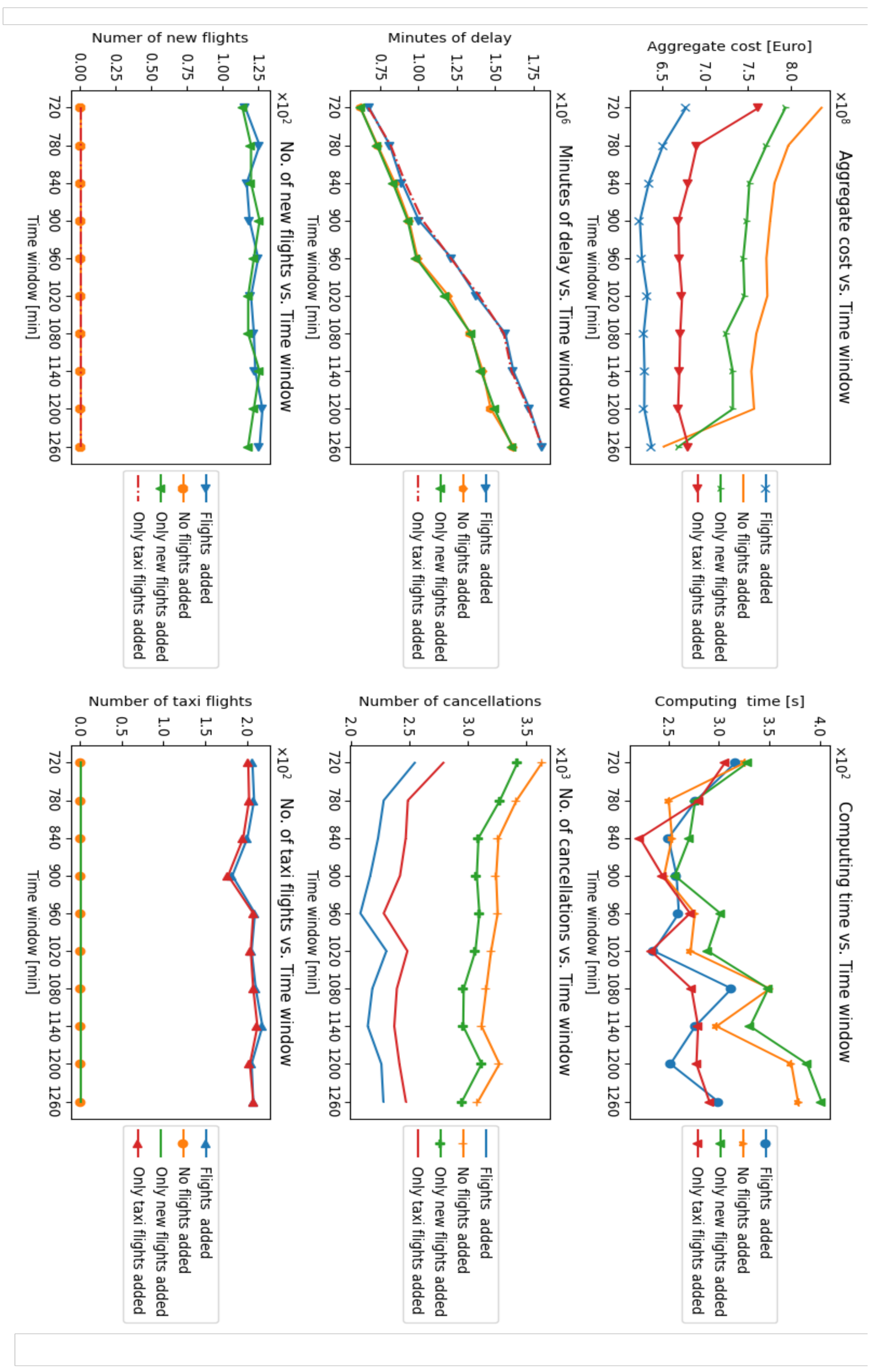


Fig. 4.3 Default scenario for the CHARP

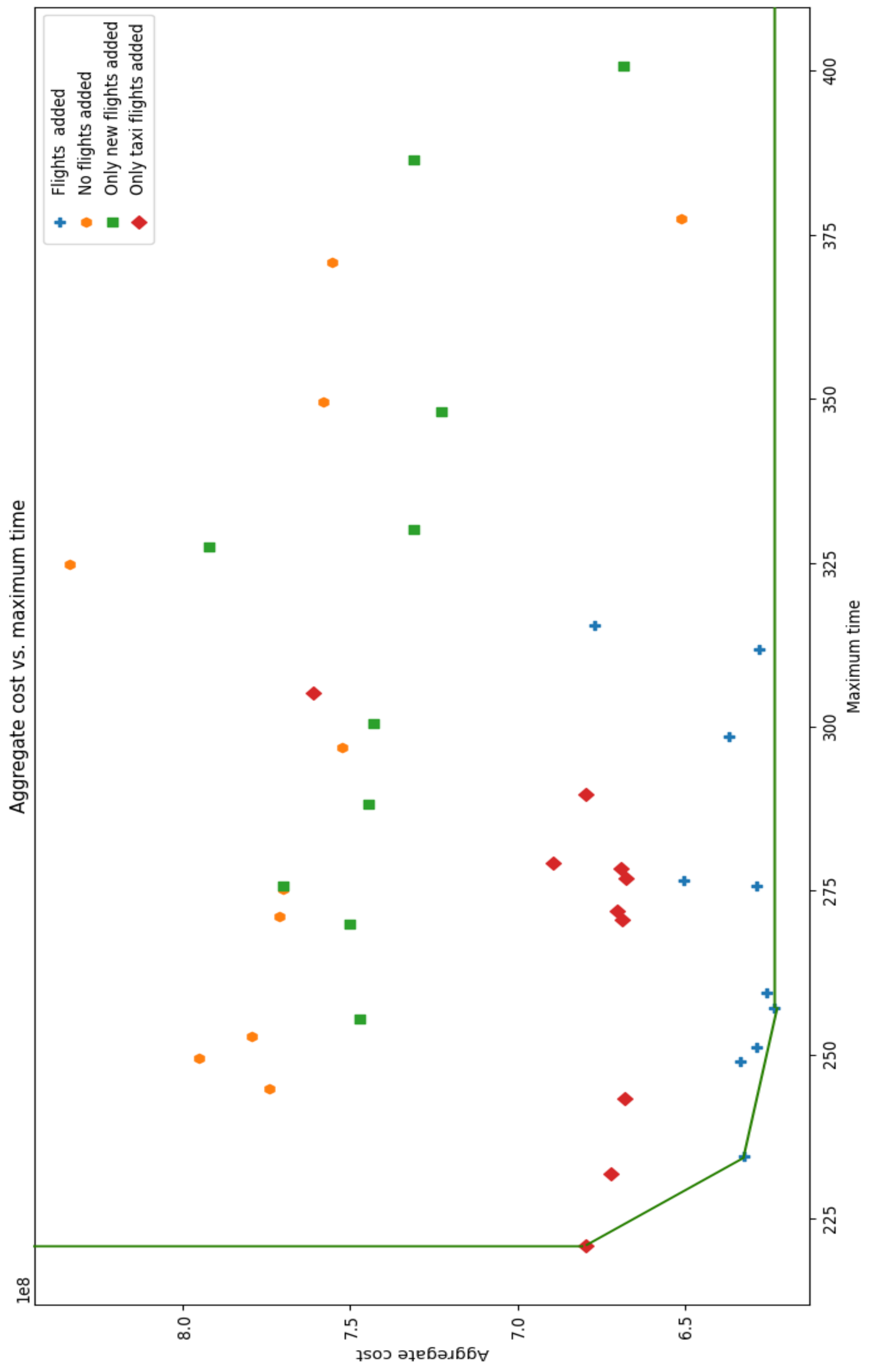


Fig. 4.4 Pareto front for the CHARP's default scenario

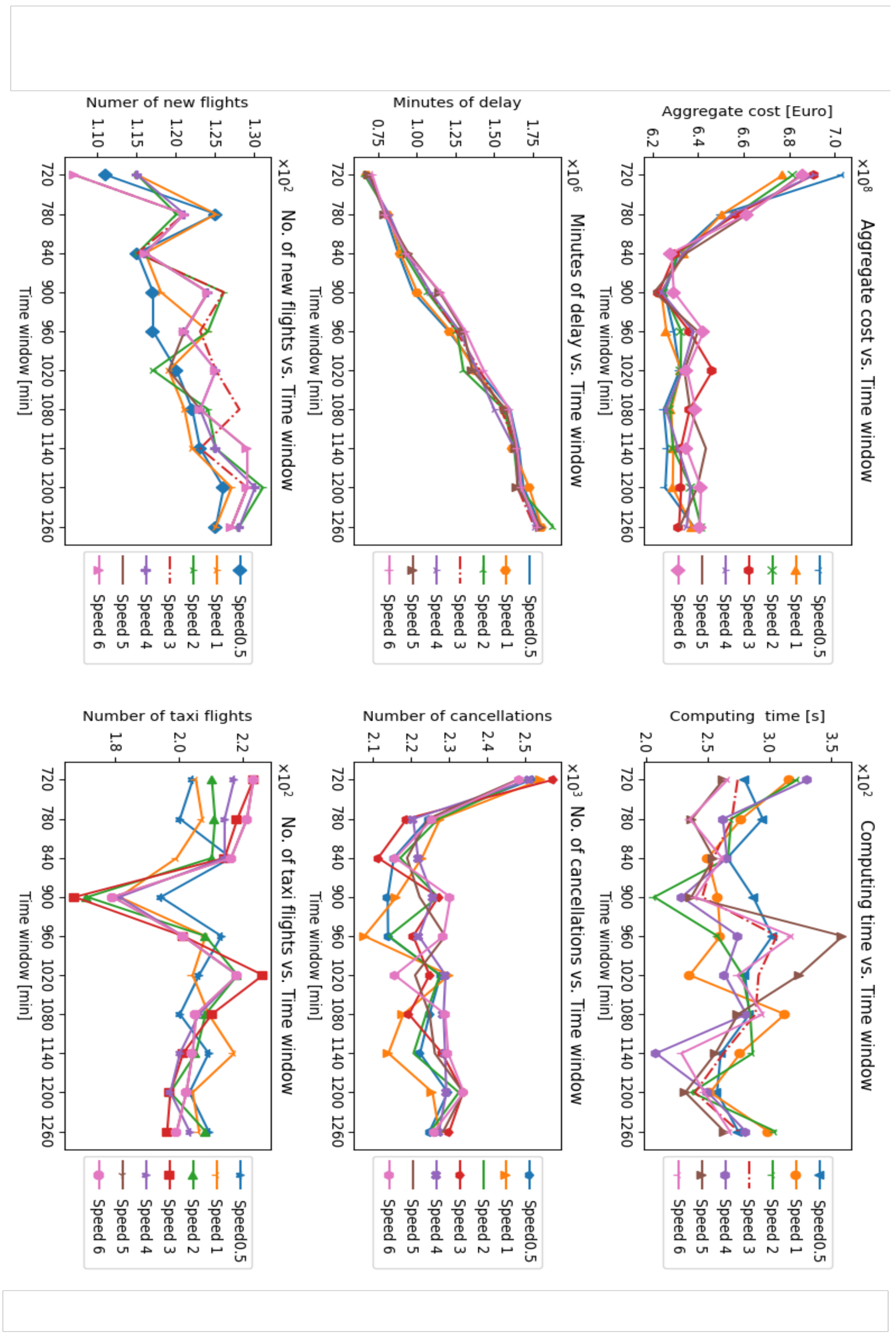


Fig. 4.5 Impact of the Pincer Speed

To have a better understanding of the solution quality in figure 4.6 draws the Pareto front considering the aggregate cost versus the computing time for the different speeds. With speed 5 it is possible to confirm the improvement for both aggregate cost and computing time. Speed 2 has the lowest value for computing time, 205.9 seconds.

4.6 Comparing the CHARP with the ARP Published Work

This section makes an extensive comparison between the published work from 2009 to 2020 and the CHARP. Subsection 4.6.1 compares the modelling used to solve the ARP, Subsection 4.6.2 compares the disruptions, Subsection 4.6.3 compares recovery actions, and finally Subsection 4.6.4 the problem characteristics.

Tables 4.6, 4.7, 4.8 and 4.9 use as legend the following abbreviations:

```
'EX': Exact method
'MH': Meta-heuristic
'HH': Hyper-heuristic
'CP': Constraint programming
'O': Optimisation
'Y': Included or mentioned
'N': Not included nor considered
'X': Not mentioned or not relevant
'F': Airport flow restriction
'C': Airport closure
'G': Generated data
'RL': Real-life data
'CPU': Computation time in seconds
```

4.6.1 ARP Solution Methods

(Qiang et al., 2009) developed a greedy simulated annealing algorithm, combining characteristics of Greedy Randomised Adaptive Search Procedure (GRASP) and SA. The combination of heuristics improves the efficiency of the neighbourhood selection and decreases the probability of falling into local optimal solution. The objective of the model is to minimise the total passenger delay time. However, the objective function does not consider all cost incurred by irregular operations *e.g.* the cost of ferrying and fleet substitution is not taken into account. To obtain the cost of ferrying (taxi flights), and creating new flights, this thesis uses the cost function supplied by ROADEF 2009 Challenge. We were able to confirm that it is beneficial both financially and also in terms of computing time.

(Eggenberg and Salani, 2009) presented a modelling framework to solve aircraft recovery by allowing consideration of operational constraints within the Column Generation (CG) scheme. This column generation algorithm solved aircraft recovery problem considering maintenance planning. A time-band recovery network was constructed for each aircraft to incorporate maintenance constraints by introducing a maintenance arc.

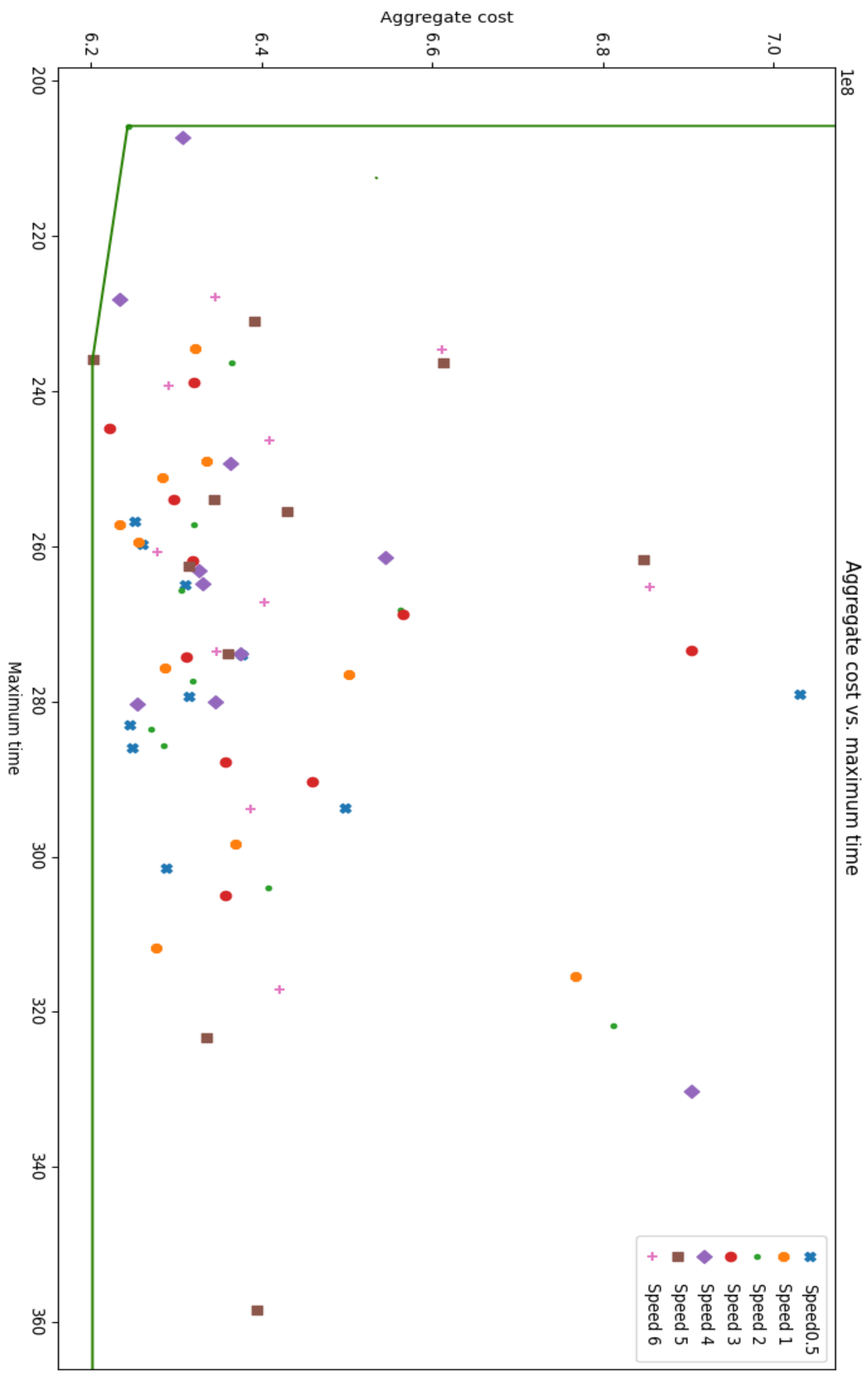


Fig. 4.6 The Pareto front for the Pincer Heuristic speed change

(Liu et al., 2010) presented a hybrid heuristic that combined an Adaptive Evaluated Vector (AEV) and an inequality-based multi-objective GA formulation that was used to search for Pareto solutions to the daily short-haul recovery problems. The AEV was used to guide the search and the GA was to provide the multi-objective solution. Although considering aircraft swap and re-timing options, the model does not consider flight cancellations as a recovery method, table 4.8.

(Wu and Le, 2012) developed a model based on flight strings instead of individual flights. They transform these strings into a time–space model that considers maintenance constraints and regulations. The model is solved with a heuristic that was developed by the authors called the Iterative Tree Growing with Node Combination.

(Xiuli and Yanchi, 2012) used a hybrid heuristic that combined a GRASP with Ant Colony Optimisation (ACO). Compared to the original GRASP algorithm, it provides a high global optimisation capability.

(Le et al., 2013) transformed the aircraft recovery problem into a vehicle routing problem with time window modelling. The formulation considers aircraft recovery and passenger delivery. In the model, aircraft are vehicles, passengers are commodities and airports are nodes. Each aircraft rotation is considered a route.

(Arias et al., 2013) combined constraint programming with a simulation approach to solve the SARP. The goals of the model are to restore the original flight schedule as much as possible, minimising the total flight delay and the number of cancelled flights. The recovery actions used consist of the rescheduling of the flight plan using delays, swaps, and cancellations, table 4.8. The robustness of the solutions is assessed by comparing the standard deviation from the simulation results with the variation of the probability distribution that was used for generating the stochastic delays and the expected propagation.

(Aktürk et al., 2014) were the first to successfully integrate cruise speed control to deal with the ARP. The authors consider the option of speeding up flights to reduce delays, at the cost of higher fuel costs. Due to the non-linearity of fuel burn in cruise speed, the authors use a conic quadratic optimisation approach to solve the problem with minimisation of recovery related costs like aircraft swap, fuel consumption, and passenger delay. CO_2 emission cost was integrated next to the additional fuel cost of speeding up flights. It is stated in the paper that significant cost savings can be achieved with cruise speed control, making it a suitable recovery approach to include in aircraft recovery studies.

(Vos et al., 2015) established a dynamic framework, named Disruption Set Solver (DSS) for the aircraft schedule recovery. The framework handles disruptions as they happen and builds on the solutions of previous disruptions i.e. the recovery problem is solved as disruptions happen, involving the solutions of new disruption but also considering the decision of the incumbent solution. The framework relies on the combined usage of an efficient aircraft selection algorithm and a linear programming model which can track the status of individual aircraft on parallel time–space networks.

(Sousa et al., 2015) presented a ACO. The proposed algorithm combines the Aircraft Assignment Problem (AAP) with the ARP and aims to minimise the operational cost and re-schedules flights dynamically by using a rolling time window. When the algorithm first runs it receives information about which routes are needed to assign aircraft, as well as where and how many aircraft are available. Therefore, its first objective is to create a valid aircraft assignment such that all flights are feasible while minimising the cost. On the other hand, the algorithm keeps running in order to adapt its schedule in case of a sudden disruption, where the original solution is modified so that all flights remain feasible with the lowest cost raise.

(Zhu et al., 2015) propose a two-stage stochastic recovery model to deal with the ARP. The first stage is a resource assignment model to minimise delay and cancellation cost. The second stage re-times the aircraft routings obtained in the first stage, with the objective of minimising the expected cost on the resource strategy of the first stage plan due to uncertainty of aircraft recovery time. The authors use a stochastic algorithm framework combining Greedy Simulated Annealing (GSA) and a simple re-timing strategy. Based on different scenarios of restoration time, the second stage model can be decoupled in several linear models.

(Xu et al., 2015) presented a time-band approximation model to compute delay cost considering a random time around the planned flying time. The Random Flying Time (RFTO) model is formulated as a Mixed Integer Linear Programming (MILP) and solved using a commercial LP solver.

(Guimarans et al., 2015) described a methodology for the SARP, that considers the stochastic nature of air transportation systems. The methodology is based on the LNS metaheuristic, combined with a simulation run at different stages to ensure robustness. A CP formulation is developed to solve the deterministic ARP. The work does not consider cancellations in our approach. In these situations, connecting flights are generally cancelled so the aircraft may be assigned to a later flight from the same airport. In other cases, aircraft may be ferried, i.e. flying without passengers, to the destination airport in order to restore the original schedule.

(Xu and Han, 2016) presents the Weighted Time-band Approximation Model (WTBAM) incorporating simplex group cycles for flight operations recovery. The objective of the WTBAM is to minimise both the delay costs and the cancellation costs through implementation of a weighted threshold.

(Hu et al., 2017) presented a solution approach for solving a multi-objective recovery problem by combining $\varepsilon - constraints$ and neighbourhood search methods. The $\varepsilon - constraints$ method is in charge of seeking the Pareto front for the multi-objective ARP and the neighbourhood search algorithm is responsible for improving the locally feasible solutions of the ARP in each iteration of the $\varepsilon - constraints$ method. The problem includes three conflicting objectives, the first objective minimises the total deviation from the original flight schedule, the second minimises the maximum flight delay time, and the third objective minimises the number of aircraft swapped.

(Zhang, 2017) use feasible Lines od Flights (LOF) as the basic variables in the model, where LOF are defined as a sequence of flights flown by one aircraft within one day. A two-stage heuristic is presented to reduce the number of included LOF, thereby reducing the run-time. In the first stage, LOF are scored and selected based on the number of swaps (less is better) and the number of flight legs included in the LOF (more is better). In the second stage, flow balance constraints for the aircraft were aggregated by creating constraints for each airport only.

(Khaled et al., 2018) proposed a multi-objective integer linear programming problem for the tail assignment problem which minimises the operating cost and the deviation from the original solution.

(Šarčević et al., 2018) described a methodology where the Artificial Bee Colony (ABC) algorithm was applied to the aircraft disruption problem.

(Zhao and Chen, 2018) presented a weight-table heuristic algorithm for the aircraft recovery problem.

The stochastic recovery problem was also proposed by (Lee et al., 2020). The authors propose an innovative reactive and proactive approach to solve the ARP problem. By forecasting systematic delays at hub airports, their study optimises recovery actions that respond to both realised disruptions and anticipated future disruptions. The authors combine a stochastic queuing model to capture airport congestion, with a commercial flight planning tool, and with a dynamic integer programming solution to model the disruption recovery. A solution based on a look-ahead approximation and sample average approximation is proposed to solve the modelling framework.

Table 4.6 Comparison of ARP solution methods with the CHARP

Bibliographic reference and CHARP	Network	Type	Solution approach
(Qiang et al., 2009)	Flight strings	MH + HH	GRASP and simulated annealing algorithm
(Eggenberg et al., 2010)	Time band	HH	Dynamic programming with column generation
(Liu et al., 2010)	Connection	MH	Hybrid multiobjective genetic algorithm
(Wu and Le, 2012)	Time-space	MH	Iterative tree growing with node combination method
(Xiuli and Yanchi, 2012)	X	MH	GRASP combined with Ant colony
(Le et al., 2013)	X	MH	Time Window Modelling and Genetic Algorithm
(Arias et al., 2013)	X	O	Constraint programming with simulation
(Aktürk et al., 2014)	Time-space	EX	Conic quadratic mixed integer programming
(Vos et al., 2015)	Time-space	MH + EX	Aircraft Selection Heuristic with MILP
(Sousa et al., 2015)	Connection	MH	Dynamic Aircraft Scheduling with Ant Colony Optimization
(Zhu et al., 2015)	X	MH	Stochastic Greedy Simulated Annealing algorithm
(Xu et al., 2015)	Time-band	EX	Time-band approximation with MILP
(Guimarans et al., 2015)	X	O	Constraint programming with LNS and simulation
(Xu and Han, 2016)	Time-band	MH	Weighted time-band approximation with MILP
(Hu et al., 2017)	Connection	MH	Neighborhood search algorithm with - constraints
(Zhang, 2017)	Connection	MH	Two stage heuristic for LOF reduction
(Khaled et al., 2018)	Time-space	MH	Multiobjective LP with e-constaint for Pareto frontier N
(Šarčević et al., 2018)	X	MH	Artificial Bee Colony algorithm implemented in MASDIMA
(Zhao and Chen, 2018)	Time-spce	MH	Weight-table based heuristic algorithm
(Lee et al., 2020)	Tim-space	O	Dynamic stochastic integer programming framework Y
(CHARP, 2022)	Time-space	CP + MH	Constraint programming with meta-heuristic

The CHARP finds solutions for the ARP using CP. The first objective is to minimise the number of cancellations and the second, the amount of delay. Following the guidelines of CP, the model gives priority to the recovery of infeasible rotations with smaller search space or singletons. The CHARP decomposes the ARP iterating through the aircraft fleet. Each infeasible aircraft rotation is recovered traversing the search space, obtained from the Cartesian product of the flight domains. In each iteration the is covered by the Pincer Meta-heuristic (PMH). The latter uses the LHA or the UHA. Some of the methods cited in this section consist of meta-heuristics that mimic biologic events or genetic improvement. The CHARP was tested against these methods however, due to the size of the search space, the recovery was very slow. The search space played a major role in the choice of CP in particular in the use of CPr. To reduce the search space the CHARP uses CPr, by removing infeasible solutions, (Hooker et al., 1999). From this viewpoint, CP operates on the set of possible solutions, narrows it, and ultimately reduces computing time needed to traverse the search space. In situations where the infeasible rotation cannot be recovered, because airports do not have capacity, the CHARP backtracks, by removing from the solution an aircraft rotation that can increase airport capacity. Both backtracking and CPr were not not mentioned in the references cited in this section, therefore one can admit novelty in the usage of CP in the CHARP.

4.6.2 ARP Disruptions Scenarios

The ARP models are a limited representation of reality, hence some assumptions are needed to model the disruption problem. A common assumption, followed by almost all papers, is that crew is always available to perform the flights in the recovered schedule, (Sousa et al., 2015), (Vos et al., 2015). Another common simplification is the exclusion of airport capacity constraints or slot availability (Liu et al., 2010), (Arias et al., 2013). The CHARP assumes that the crew is always available however it considers airport capacity constraints and disruptions. The majority of the papers cited in this section do not present explicitly the disruption scenarios however, for the remainder the most common is aircraft breakdown, followed by flight delay, airport capacity shortage, and flight cancellation, table 4.7. For instance, the disruptions included in (Zhang, 2017) are airport closures and aircraft unavailability due to unplanned maintenance. Zhao and Chen, 2018, only consider disruptions from airport closures due to bad weather conditions. The CHARP not only considers all the aforementioned disruption scenarios singly, but also combinations of two of them, flight and aircraft, flight and airport, and aircraft and airport.

4.6.3 ARP Recovery Actions

It is possible to observe from table 4.7, that there are many papers do not provide explicit information regarding the recovery actions, namely for the usage of reserve aircraft, ferry aircraft and cruise speed. In general, the more recovery actions the bigger the search space becomes. The latter impacts on computing time, hence many papers choose not to include flight creation. In the case of the CHARP, flight creation decreases computing time. It is also observable that the most common and explicit recovery procedure is flight cancellation, followed by flight delay and aircraft swap.

4.6.4 ARP Characteristics

(Liu et al., 2010) tested the model on a daily flight schedule of a Taiwanese airline with 7 aircraft (single fleet) during a 1 hour airport closure, impacting 39 flights. The authors mention the heuristic presents results in 3.6 minutes on average (7.5 minutes maximum).

(Wu and Le, 2012) is tested on a data set from China Airlines consisting of 170 flights, 5 fleets, 35 aircraft, and 51 airports. The computing time is not reported.

(Xiuli and Yanchi, 2012) state that the model was tested on a multi-fleet network with 50 aircraft and more than 5 aircraft types. This work does not present the number of flights in the case study nor the computation time.

Table 4.7 ARP disruptions

Bibliographic reference and CHARP	Flight delay	Flight cancellation	Aircraft breakdown	Airport capacity shortage
(Qiang et al., 2009)	X	X	Y	Y
(Eggenberg et al., 2010)	X	X	Y	Y
(Liu et al., 2010)	X	X	X	Y
(Wu and Le, 2012)	X	X	Y	X
(Xiuli and Yanchi, 2012)	X	X	X	X
(Le et al., 2013)	X	X	Y	X
(Arias et al., 2013)	Y	X	X	X
(Aktürk et al., 2014)	Y	X	X	X
(Vos et al., 2015)	Y	X	Y	X
(Sousa et al., 2015)	X	X	Y	X
(Zhu et al., 2015)	X	X	Y	X
(Xu et al., 2015)	X	X	Y	X
(Guimarans et al., 2015)	Y	X	X	X
(Xu and Han, 2016)	X	X	Y	X
(Hu et al., 2017)	X	X	Y	X
(Zhang, 2017)	X	X	Y	Y
(Khaled et al., 2018)	X	X	Y	Y
(Šarčević et al., 2018)	Y	X	Y	Y
(Zhao and Chen, 2018)	X	X	X	Y
(Lee et al., 2020)	X	Y	Y	Y
(CHARP, 2022)	Y	Y	Y	Y

The proposed model by (Arias et al., 2013) is tested with real data from a commercial airline with a total of 51 flights, 13 airports, and 11 aircraft. The proposed model can match the optimal solution in 14 cases out of 20. According to the authors, the results suggest that the inherent uncertainty of the ARP makes it a suitable candidate for combining simulation and optimisation methods

The framework used by (Vos et al., 2015) is applied to a set of real disruptive days in the operation of Kenya Airways. In 93.3% of the times, the DSS found solutions within 10 minutes. Furthermore, the authors showed that the solution costs are underestimated when computed using a static approach.

The work of (Sousa et al., 2015) consist of two different experiments, both using real data from a commercial airline. On a problem with 100 flights, the ACO outperforms (non-truncated) BaB and Depth First Search (DFS) in terms of solution quality, although it takes 40% more time on average. This trade-off between time and quality will probably be an issue to take into account when applying this approach in larger data sets.

Table 4.8 ARP recovery actions

Bibliographic reference and CHARP	Flight Delay	Flight cancellation	Create flight	Aircraft swap	Reserve aircraft	Ferry aircraft	Cruise Speed Control
(Qiang et al., 2009)	Y	Y	N	Y	X	X	X
(Eggenberg et al., 2010)	Y	Y	N	Y	X	X	X
(Liu et al., 2010)	Y	Y	N	Y	X	X	X
(Wu and Le, 2012)	X	Y	Y	N	Y	X	X
(Xiuli and Yanchi, 2012)	X	Y	Y	N	Y	X	X
(Le et al., 2013)	Y	N	N	Y	X	X	X
(Arias et al., 2013)	Y	Y	Y	Y	N	N	N
(Aktürk et al., 2014)	Y	N	N	Y	X	X	Y
(Vos et al., 2015)	Y	Y	N	Y	X	X	X
(Sousa et al., 2015)	Y	Y	N	Y	X	X	X
(Zhu et al., 2015)	Y	Y	N	Y	X	X	X
(Xu et al., 2015)	Y	Y	N	N	X	X	X
(Guimarans et al., 2015)	Y	N	N	Y	X	X	X
(Xu and Han, 2016)	N	Y	N	X	X	X	X
(Hu et al., 2017)	Y	Y	N	Y	X	X	X
(Zhang, 2017)	Y	Y	N	Y	X	X	X
(Khaled et al., 2018)	N	Y	N	Y	X	X	X
(Šarčević et al., 2018)	Y	Y	N	Y	X	X	X
(Zhao and Chen, 2018)	Y	Y	N	Y	X	X	X
(Lee et al., 2020)	X	Y	Y	N	Y	X	Y
(CHARP, 2022)	Y	Y	Y	N	N	Y	N

With data on the actual flying time and the planned flying time from 400 flights in a day of Sichuan Airlines, (Xu et al., 2015) create a uniform probability density function which predicts the flying time of flights. The model is tested on a network of generated data with 3 aircraft and 11 flights.

The proposed methodology by (Guimarans et al., 2015) was tested on several instances with different characteristics, some of which were obtained from real data provided by a Spanish airline.

(Xu and Han, 2016) uses actual data from Air China show that the weighted time-band approximation model is feasible. The authors mention also that the results of stochastic experiments using actual data from Sichuan Airlines show that the flight disruption and computation time are controlled by the airline operations control centre, which aims to achieve a balance between the flight disruption scope and depth, computation time, and recovery value.

The methodology is tested on real-world empirical data for a Boeing 737 fleet consisting of 104 aircraft from a major Chinese airline covering 410 flights. The computation times range between 12 and 20 min, depending on the disruption instance.

The approach in (Zhang, 2017) is tested on five real life test scenarios. The largest instance included 44 aircraft and 638 flights; the computation time was 150 seconds.

The model proposed in (Khaled et al., 2018) computes solutions in less than 30 seconds for the adapted test case involving 111 flights and 10 aircraft.

The system in (Šarčević et al., 2018) is tested on a month worth of real airline data, however, dimensions of the case study and required runtime are not given.

(Zhao and Chen, 2018) depicts a single case study consisting of 6 aircraft and 31 flights. The computation times are not presented.

The data instances tested using the CHARP includes aircraft maintenance, a planning horizon of 1 to 3 days and number of flights between 608 and 2178. The testing is performed over 32 data instances each of which has its specific aircraft fleet and flights. It is possible to verify that it comprises biggest number of aircraft and flights. However, the use of CP reduces significantly computing time. The data used is publicly available and it has been tested extensively.

Table 4.9 ARP problem characteristics

Bibliographic reference and CHARP	Multi-Fleet	Maintenance	Data	Aircraft	Fleets	Flights	Computing time [sec.]
(Qiang et al., 2009)	X	N	N	50	1	200	< 120
(Eggenberg et al., 2010)	Y	Y	RL	100	1	760	63
(Liu et al., 2010)	N	N	RL	7	1	72	81 to 450
(Wu and Le, 2012)	Y	Y	RL	35	5	170	X
(Xiuli and Yanchi, 2012)	Y	Y	RL	50	5	X	X
(Le et al., 2013)	Y	N	RL	6	3	30	< 98
(Arias et al., 2013)	X	N	RL	11	X	51	X
(Aktürk et al., 2014)	Y	N	RL	60	6	207	202
(Vos et al., 2015)	X	Y	RL	43	1	X	< 600
(Sousa et al., 2015)	N	N	RL	72	1	100	18
(Zhu et al., 2015)	N	N	RL	6	1	23	< 900
(Xu et al., 2015)	N	N	RL + G	3	1	11	< 1
(Guimarans et al., 2015)	N	N	RL	40	1	163	< 226
(Xu and Han, 2016)	X	N	RL	60	X	254	< 1006
(Hu et al., 2017)	Y	N	RL	104	1	410	1200
(Zhang, 2017)	N	Y	RL	44	1	638	150
(Khaled et al., 2018)	N	Y	RL	10	1	111	< 30
(Šarčević et al., 2018)	X	N	X	X	X	X	X
(Zhao and Chen, 2018)	X	N	RL	6	X	32	X
(Lee et al., 2020)	Y	N	RL	X	3	852	< 300
(CHARP, 2022)	Y	Y	Y	618	8	2178	< 206

To compare the performance of the different bibliographic references and the CHARP, table 4.10 presents the ratio between computing time per number of flights and aircraft. Similarly, figure 4.7 presents the same results using the logarithmic scale (in the vertical axis) for the same ratio.

4.7 Conclusion and future work

ARP is a practical problem that needs to be solved during operations, therefore, the efficiency of the method in terms of computational time is a very important characteristic. The CHARP loads the respective

Table 4.10 Performance comparison

Bibliographic reference and CHARP	Computing time per flight per aircraft
(CHARP, 2022)	1.53E-04
(Eggenberg et al.,2010)	8.29E-04
(Sousa et al.,2015)	2.50E-03
(Zhang,2017)	5.34E-03
(Qiang et al.,2009)	1.20E-02
(Aktürk et al.,2014)	1.63E-02
(Khaled et al.,2018)	2.70E-02
(Hu et al.,2017)	2.81E-02
(Xu et al.,2015)	3.03E-02
(Guimarans et al.,2015)	3.47E-02
(Xu and Han,2016)	6.60E-02
(Le et al.,2013)	5.44E-01
(Zhu et al.,2015)	6.52E+00

data, introduces the disruptions and allocates flight slots to replace the cancelled flights. It afterwards recovers infeasible rotations splitting them in two. The recovery procedure consists of delaying, cancelling and creating new flights. It is done using the upper or the lower heuristic depending on the size of the search space. When necessary and if possible, the recovery procedure creates taxi flights to connect the aircraft's origin airport with the first flight of the recovered rotation. In the impossibility of the latter, the algorithm will cancel the flights in the recovered rotation until the continuity constraint is satisfied. The data instances vary in size comprising a number of flights ranging from 608 to 2178, aircraft from 85 to 618 and airports from 35 to 168. Each of the data instances has a disruption scenario that can include one or two types of disruptions. Flights can be delayed or cancelled, aircraft can have a breakdown period and airports can have their capacity for departure and arrival shorted. The experiments were run simultaneously and starting from the base scenario it was demonstrated that computing time can be improved by changing the incremental and decremental values of the CHARP's pincer movement. This chapter demonstrated the use of a novel heuristic based in CP, to run thirty two data instances simultaneously, and we were able to find solutions in 206 seconds of computing time for the biggest data instance. This result opens perspectives to include crew and passenger recovery. Additionally, it also allows the study of the impact of block time changes in the ARP.

The main conclusion that can be drawn from the comparison with the published work in Section 4.6 is that the airline disruption management problem is still a growing field of research. In this chapter we proved that CP combined with the PMH can be used successfully to tackle the ARP, in a reasonable computing time, overcoming situations where the problem is not tractable. The upper heuristic is initially used when the search space is bigger than 10^{12}. One could be tempted to propose the use of a quantum algorithm for such an intractable problem from the standpoint of classical computing. In the previous decades, the world has witnessed a number of major breakthroughs in quantum algorithms that provably exceed the finest classical algorithms. However, it is currently unknown if quantum algorithms can leverage computing power over classical computing, and if so, how much, or how to create quantum algorithms that achieve such benefits. Heuristic algorithms, have shown to be effective empirically, but have not been mathematically demonstrated to outperform other approaches, that are used to solve many of the most difficult computational real world problems. Although quantum heuristic algorithms have been proposed, empirical testing won't be achieved until quantum computation hardware is mature, (Biswas et al., 2017).

Several practical and methodological challenges can be identified and stimulate future research. Chapter 5 follows recent developments that include flight time variability (Arıkan et al., 2017), (Wen et al., 2020) to recover airline disruption. The CHARP will be extended by integrating the results, for block time and fuel obtained by (de Lemos and Woodward, 2021) since the latter proved to be reliable and the block time is lower than the one used in the ROADEF 2009 data set.

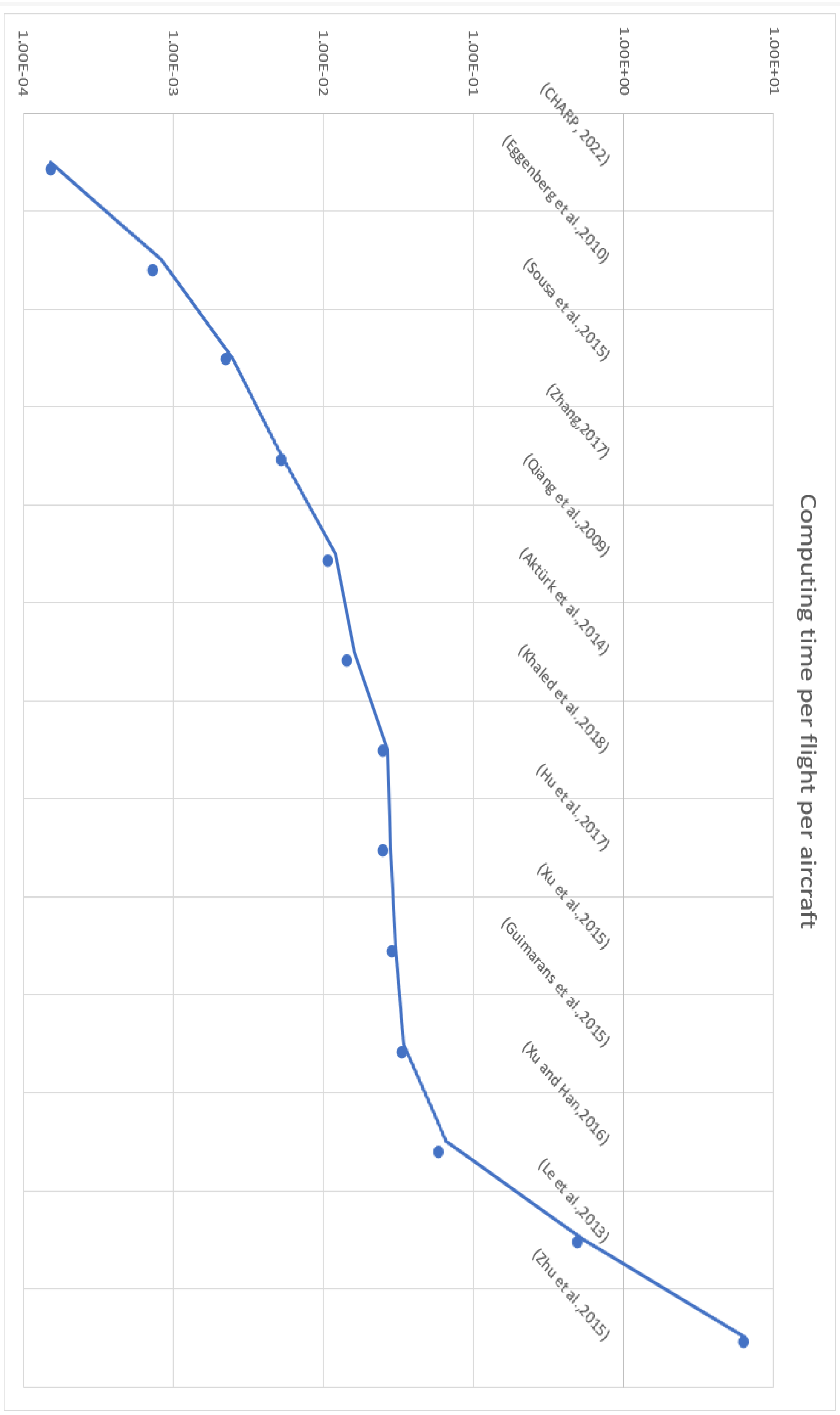


Fig. 4.7 Performance comparison

Chapter 5

The Impact of Smaller Block Times in Disruption Recovery

5.1 Introduction

While working on this thesis' goals, Chapter 3 presented the BTF flight planning model, that given two airports - origin and destination airport - and an aircraft model to fly between them, calculates the distance between the two airports, derives the flight profile and computes the block time and consumed fuel. The BTF model proved to compute reliable results for the block time and consumed fuel for a specific aircraft model. When comparing its results for block time with those of the ROADEF 2009 Challenge, they proved to be smaller. The comparison with the block time obtained from Flightaware™ demonstrated that the BTF values for the block time cluster in the 90^{th} percentile. Added to the latter comparing the BTF values with those used in Lido™ flight plans also proved the values for block time and consumed fuel obtained from the BTF are quite accurate.

Chapter 4 describes a novel algorithm to tackle the ARP, the CHARP. The latter consists of a CP model combined with he PMH to find solutions for the ARP in a reasonable computing time. The CHARP uses flight delays or cancellations and creates new flights to recover from disruption. To reduce the search space the CHARP uses CPr, the latter renders a substantial reduction in computing time. Comparing the CHARP's with other models demonstrated that it outperforms them in terms of computing time per flight and aircraft.

During the research process, it was also noted that one of the disruption recovery strategies that lately caught the attention of researchers consists in speeding up flights. Implicitly one expects that speeding up flights will reduce block time. Since the BTF computes block time values lower than those in the data sets it is natural to try disruption recovery using these values. Hence, this chapter studies the effect of smaller block times in the CHARP. In particular, it researches for which disruption scenarios, a lower block time is a viable alternative to recover disruption in commercial aviation. The experiments to study the effects of smaller block times were made in an Intel® Xeon Gold CPU @ 2.3GHz box with 72 CPU(s), 128 GiB of RAM box.

The remaining sections of this chapter have the following content, Section 5.2 compares the block time differences with the ROADEF 2009 Challenge data set for westbound and eastbound flights, Section 5.3 presents the effects of smaller block time in the CHARP, Section 5.4 compares for six different disruptions types the results for the default scenario defined in table 4.4 and the smaller block time, and finally Section 5.5 presents the conclusions and future work.

5.2 Block Time Comparison

While developing the BTF model, we considered the difference between westbound and eastbound flight, figure 3.1. It is also also known that flights with longer ground distance have higher cruise flight level. Table 5.1 presents the parametric analysis of block time differences with respect to the cruise flight level. From table 5.1 the values for maximum, minimum, average and standard deviation increase with the cruise flight level. This pattern is also observed for westbound flights, table 5.2. To understand qualitatively how the block time differences vary with flight level, for eastbound and westbound flights, ground distance is also added to the plot, figures 5.1 and 5.2.

Table 5.1 Parametric analysis of block time differences for west bound flights

FL	Max. [min.]	Min.[min.]	Average [min.]	Stand. dev. [min.]
340	40	1	12.4	6.8
360	46	1	14.7	6.0
380	69	2	20.6	12.2

Table 5.2 Parametric analysis of block time differences for east bound flights

FL	Max. [min.]	Min. [min.]	Average [min.]	Stand. dev. [min.]
330	35	1	11.6	6.5
350	28	1	12.0	5.2
370	85	1	21.4	15.5

The overall parametric analysis, for east and west bound flights is presented in table 5.3. Since this chapter aims at understanding the effects of block time differences for all flights are also presented.

Table 5.3 Parametric analysis of block time differences for all flight types

Flights	Max. [min.]	Min.[min.]	Average [min.]	Stand. dev. [min.]
All east bound	85	1	13.7	9.7
All west bound	69	1	14.6	8.6
All flights	85	1	14.2	9.1

5.3 Effects of Smaller Block Time in the CHARP

This section uses the smaller block times computed by the BTF model and introduces them in the aircraft rotations inside the RTW. The CHARP will then recover the data set using the parameters defined in Section 4.5.1, table 4.4. The following subsections present the CHARP graphical results using the smaller block times for all flights in 5.3.1, only disrupted flights in 5.3.2, only non-disrupted flights 5.3.3 and finally Subsection 5.3.4 presents a quantitative comparison between the previous scenarios and the default scenario.

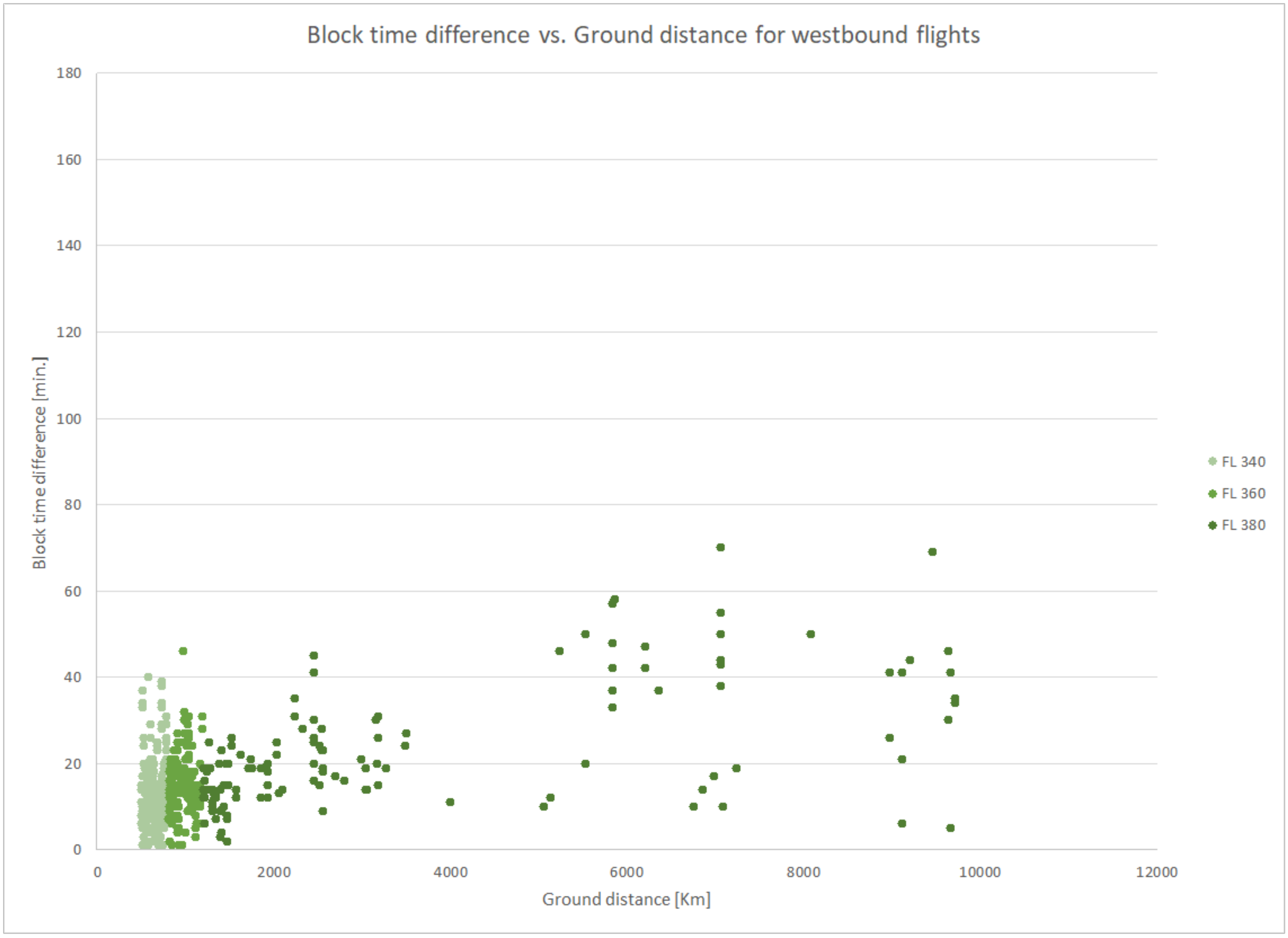


Fig. 5.1 Block time difference vs. ground distance for westbound flights

5.3.1 Smaller Block Time for All Flights Inside the RTW

In our first attempt to understand the impact of a smaller block time the model recreated the rotations replacing the flights inside the RTW with the smaller block times calculated using the BTF model. The results in figure 5.3 compared with default scenario in figure 4.3 show that the best results for aggregate cost are for the "Flights added" scenario. It is also possible to observe that the aggregate cost and computing did not improve. The minutes of delay show the same tendency however with the current scenario their value has increased. Similar results can be observed with the number of cancellations. The number and the tendency of new flights created is quite similar to the default scenario. The number of taxi flights is bigger and notably, it increases for the 900 minutes time window, as opposed to the default scenario.

5.3.2 Smaller Block Time for Disrupted Flights Inside the RTW

In the previous section it was possible to conclude that the recovery, with a smaller block time for all flights inside the RTW did not improve the results. This section presents the graphical results upon the introduction of new block times only for disrupted flights inside the RTW. Apart from computing time,

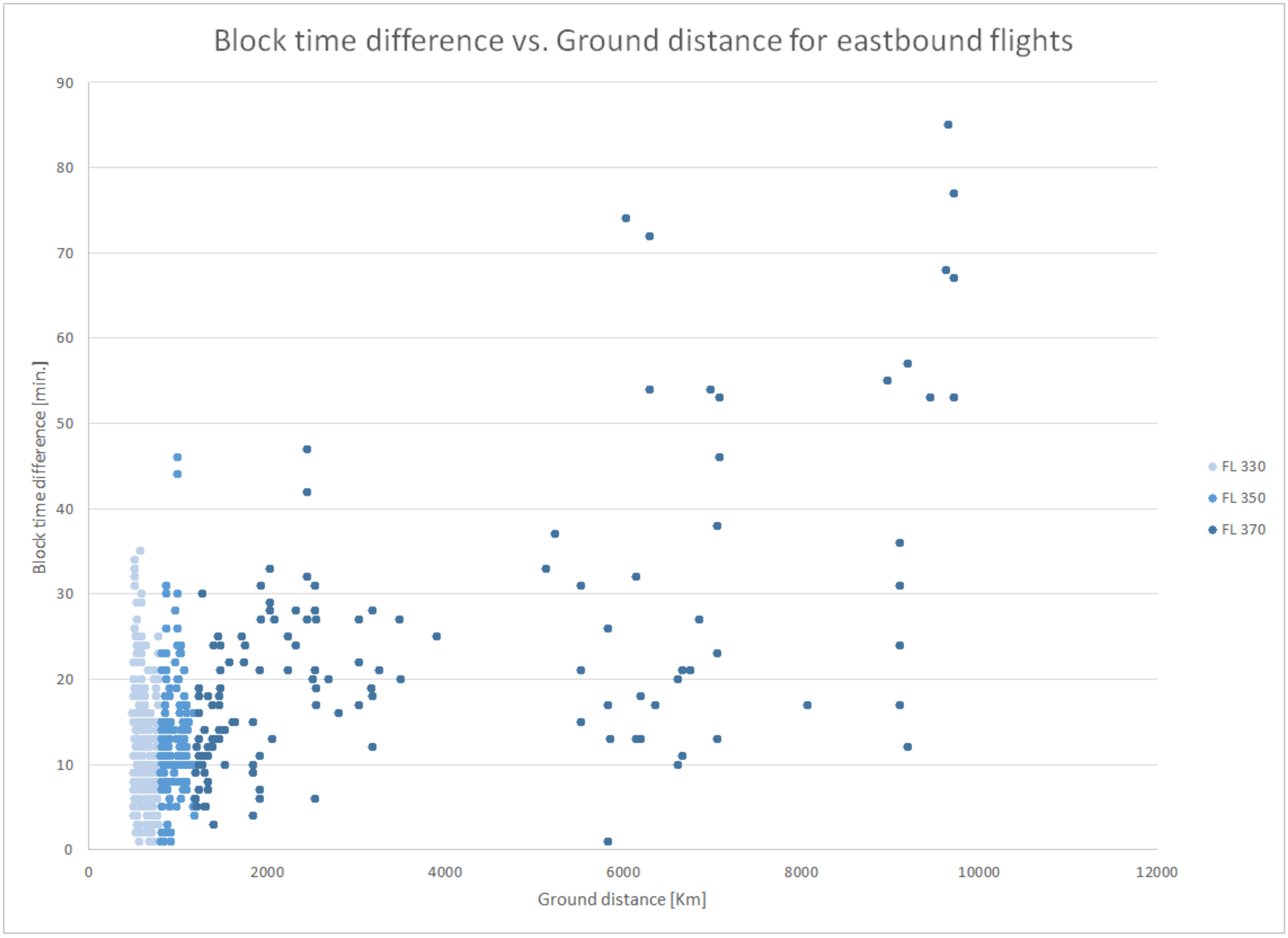


Fig. 5.2 Block time difference vs. ground distance for eastbound flights

the results in figure 5.4 are equal to the default scenario. The computing time as aforementioned is greater than the default scenario.

5.3.3 Smaller Block Time for Non-disrupted Flights Inside the RTW

In the previous section it was possible to conclude that it was not possible to improve the results. The only remaining option to consider, consists of updating the block time for non-disrupted flights inside the RTW. The results for this scenario presented in figure 5.5 are similar to those presented in figure 5.3.

5.3.4 Smaller Block Time Quantitative Result Comparison

This section compares the results between the default scenario defined in Section 4.5.1, table 4.4, and scenarios presented in the previous three sections. In terms of aggregate cost, the best recovery strategy includes the creation of new flights and taxi flights, therefore all results henceforth presented are for the "Flights added" scenario. The first pattern that emerges from the results presented in tables 5.4, 5.5, 5.6, 5.7, 5.8 and 5.9 is that replacing flights with smaller block times render worse solutions. In each of

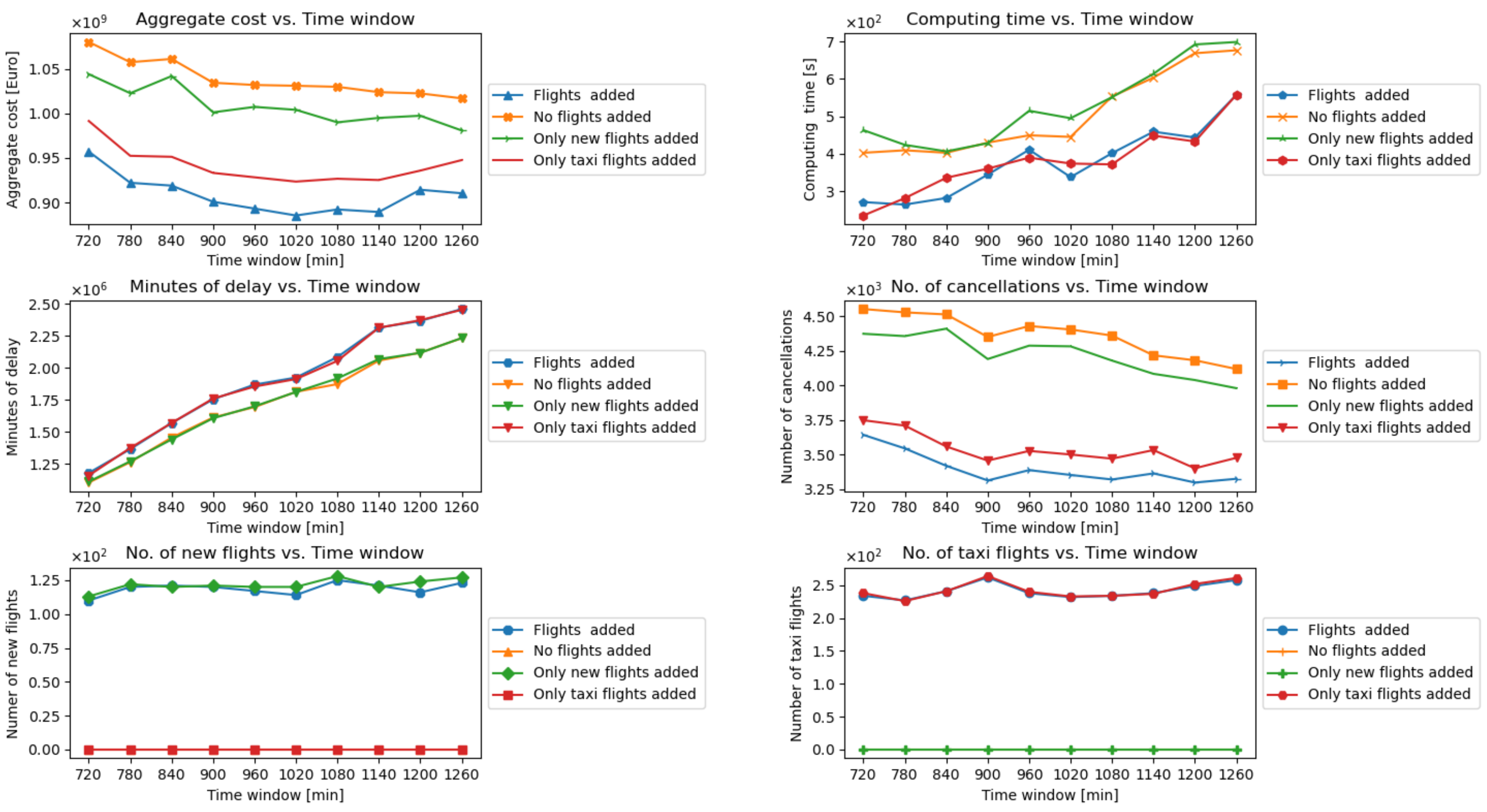


Fig. 5.3 Impact of smaller block time for all flights inside the RTW

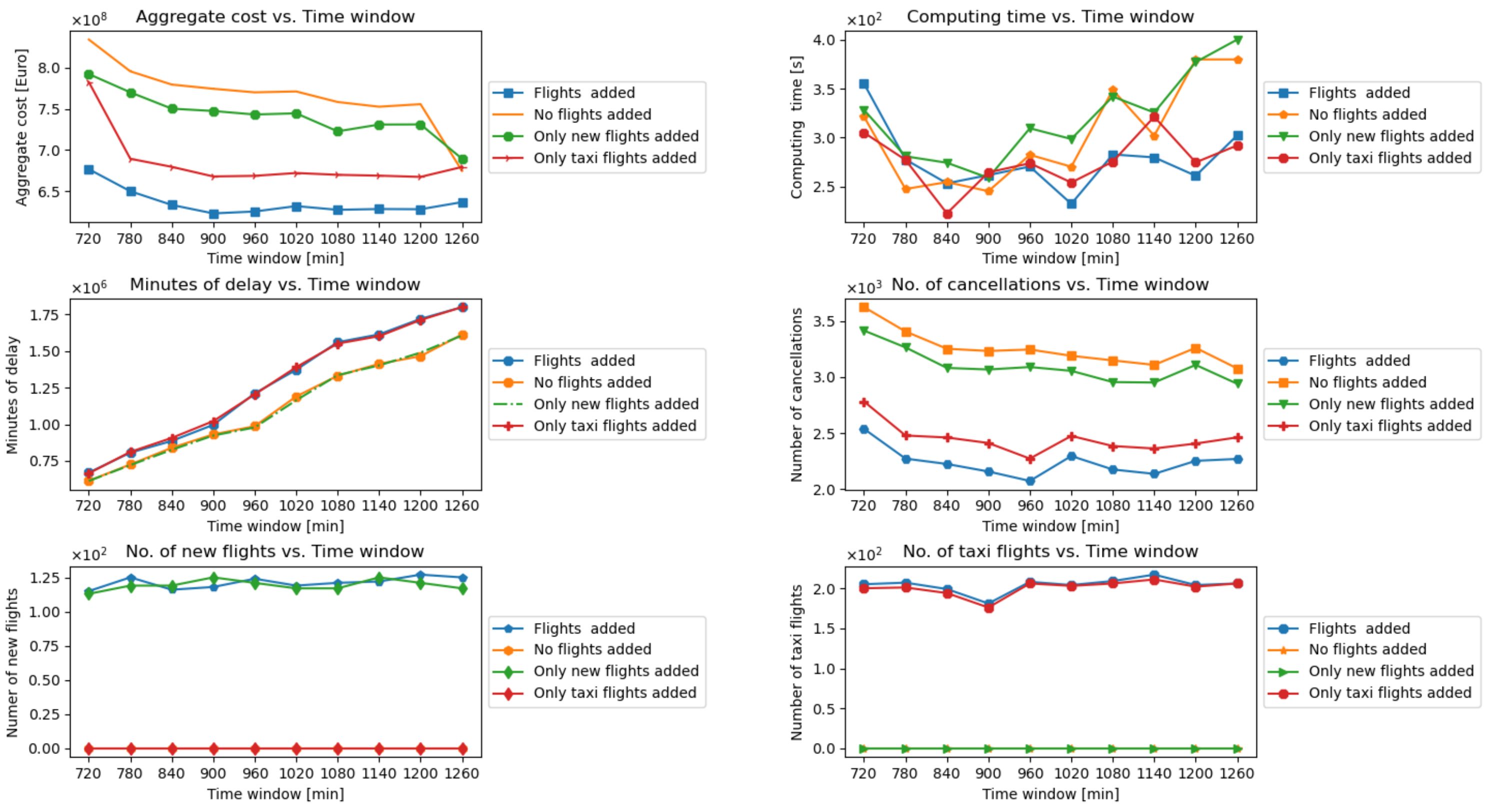


Fig. 5.4 Impact of smaller block time for flights inside the RTW

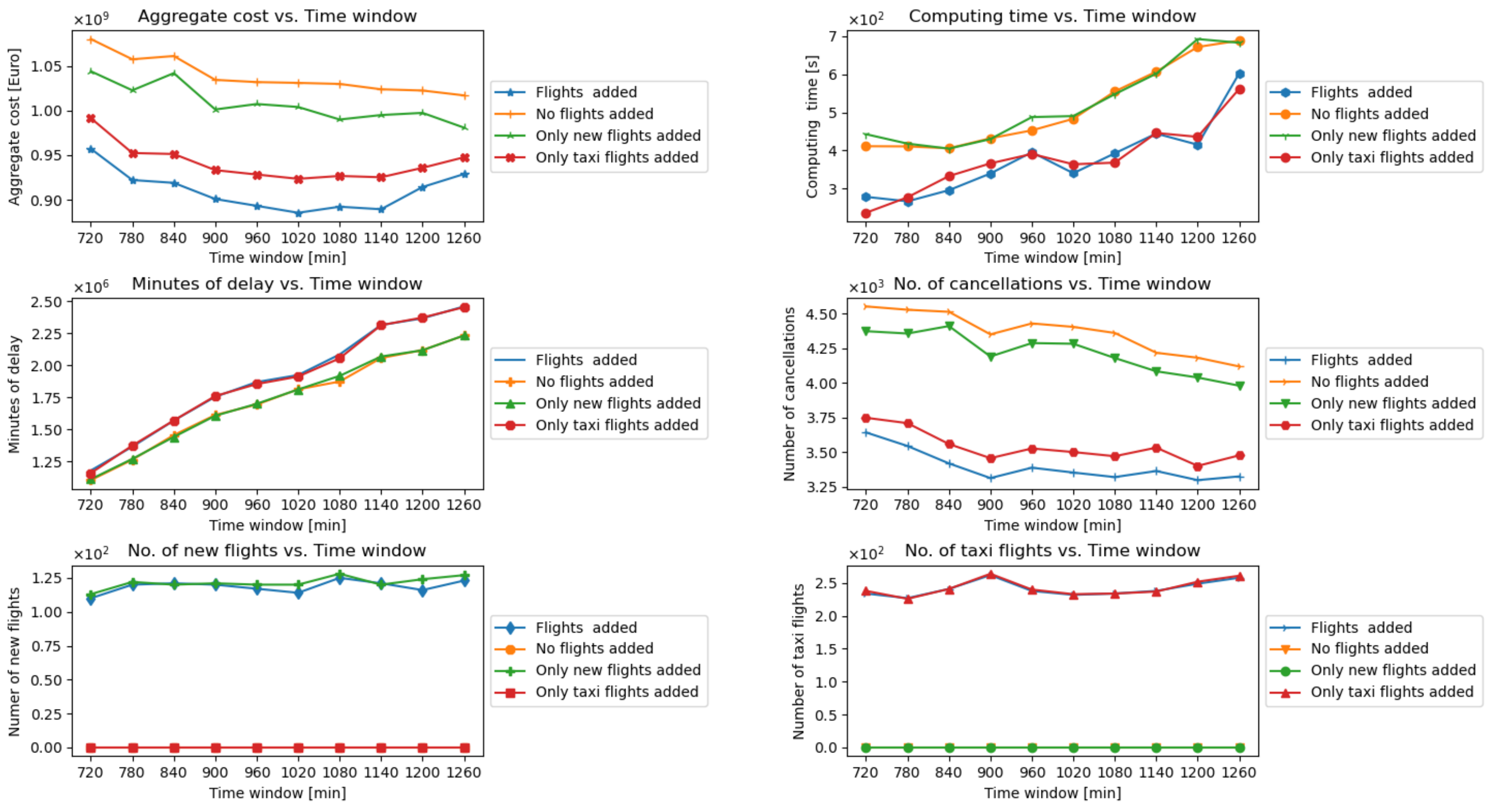


Fig. 5.5 Impact of smaller block time for non-disrupted flights inside the RTW

these tables, the cells highlighted in grey are the minimum value and those highlighted in green are the maximum value.

Except for computing time, the other pattern that is visible is that the results presented for the default scenario are equal to the results presented for smaller block times in disrupted files. Similarly, the results presented for smaller block times for all flights are equal to the results presented for smaller block times for non-disrupted flights. Therefore, the impact of smaller block times for disrupted flights is null.

Table 5.4 Aggregate cost comparison between the default scenario and smaller block time scenarios

	Default scenario	All flights		Disrupted flights		Non-disrupted flights	
Time window	Cost [10^8 €]	Cost [10^8 €]	Relative difference	Cost [10^8 €]	Relative difference	Cost [10^8 €]	Relative difference
720	6.77	9.57	41.42%	6.77	0.00%	9.57	41.42%
780	6.50	9.22	41.84%	6.50	0.00%	9.22	41.84%
840	6.34	9.19	45.05%	6.34	0.00%	9.19	45.05%
900	6.23	9.01	44.53%	6.23	0.00%	9.01	44.53%
960	6.26	8.93	42.78%	6.26	0.00%	8.93	42.78%
1020	6.32	8.85	40.08%	6.32	0.00%	8.85	40.08%
1080	6.28	8.92	42.16%	6.28	0.00%	8.92	42.16%
1140	6.29	8.89	41.49%	6.29	0.00%	8.89	41.49%
1200	6.28	9.14	45.51%	6.28	0.00%	9.14	45.51%
1260	6.37	9.10	42.93%	6.37	0.00%	9.29	45.83%

Table 5.5 Computing time comparison between the default scenario and smaller block time scenarios

	Default scenario	All flights		Disrupted flights		Non-disrupted flights	
Time window	Computing time	Computing time	Relative difference	Computing time	Relative difference	Computing time	Relative difference
720	315.41	270.96	-14.09%	355.09	12.58%	278.68	-11.65%
780	276.54	264.31	-4.42%	277.54	0.36%	266.90	-3.48%
840	249.05	281.85	13.17%	253.15	1.65%	296.09	18.89%
900	257.19	344.26	33.86%	261.62	1.72%	339.88	32.15%
960	259.45	410.83	58.35%	270.30	4.18%	395.66	52.50%
1020	234.53	337.68	43.98%	232.19	-1.00%	340.58	45.22%
1080	311.72	401.67	28.86%	282.74	-9.30%	392.78	26.00%
1140	275.66	459.03	66.52%	279.70	1.47%	444.55	61.26%
1200	251.15	443.47	76.58%	260.91	3.89%	415.35	65.38%
1260	298.43	557.34	86.76%	302.23	1.27%	602.20	101.79%

Table 5.6 Minutes of delay comparison between the default scenario and smaller block time scenarios

	Default scenario	All flights		Disrupted flights		Non-disrupted flights	
Time window	Min. of delay [10^6]	Min. of delay [10^6]	Relative difference	Min. of delay [10^6]	Relative difference	Min. of delay [10^6]	Relative difference
720	0.67	1.18	75.81%	0.67	0.00%	1.18	75.81%
780	0.81	1.36	69.22%	0.81	0.00%	1.36	69.22%
840	0.89	1.57	76.92%	0.89	0.00%	1.57	76.92%
900	1.00	1.75	76.02%	1.00	0.00%	1.75	76.02%
960	1.21	1.87	54.29%	1.21	0.00%	1.87	54.29%
1020	1.37	1.92	40.12%	1.37	0.00%	1.92	40.12%
1080	1.56	2.08	33.49%	1.56	0.00%	2.08	33.49%
1140	1.61	2.31	43.32%	1.61	0.00%	2.31	43.32%
1200	1.72	2.37	37.54%	1.72	0.00%	2.37	37.54%
1260	1.80	2.46	36.58%	1.80	0.00%	2.46	36.58%

Table 5.7 Number of cancellations comparison between the default scenario and smaller block time scenarios

	Default scenario	All flights		Disrupted flights		Non-disrupted flights	
Time window	Number of cancel. [10^3]	Number of cancel. [10^3]	Relative difference	Number of cancel. [10^3]	Relative difference	Number of cancel. [10^3]	Relative difference
720	2.54	3.64	43.48%	2.54	0.00%	3.64	43.48%
780	2.27	3.55	55.89%	2.27	0.00%	3.55	55.89%
840	2.23	3.42	53.59%	2.23	0.00%	3.42	53.59%
900	2.16	3.31	53.45%	2.16	0.00%	3.31	53.45%
960	2.08	3.39	63.28%	2.08	0.00%	3.39	63.28%
1020	2.30	3.35	45.97%	2.30	0.00%	3.35	45.97%
1080	2.18	3.32	52.50%	2.18	0.00%	3.32	52.50%
1140	2.14	3.36	57.34%	2.14	0.00%	3.36	57.34%
1200	2.25	3.30	46.32%	2.25	0.00%	3.30	46.32%
1260	2.27	3.33	46.41%	2.27	0.00%	3.33	46.41%

Table 5.8 Number of new flights comparison between the default scenario and smaller block time scenarios

	Default scenario	All flights		Disrupted flights		Non-disrupted flights	
Time window	Number of new flights	Number of new flights	Relative difference	Number of new flights	Relative difference	Number of new flights	Relative difference
720	115	110	-4.35%	115	0.00%	110	-4.35%
780	125	120	-4.00%	125	0.00%	120	-4.00%
840	116	121	4.31%	116	0.00%	121	4.31%
900	118	120	1.69%	118	0.00%	120	1.69%
960	124	117	-5.65%	124	0.00%	117	-5.65%
1020	119	114	-4.20%	119	0.00%	114	-4.20%
1080	121	125	3.31%	121	0.00%	125	3.31%
1140	122	121	-0.82%	122	0.00%	121	-0.82%
1200	127	116	-8.66%	127	0.00%	116	-8.66%
1260	125	123	-1.60%	125	0.00%	123	-1.60%

Table 5.9 Number of taxi flights comparison between the default scenario and smaller block time scenarios

	Default scenario	All flights		Disrupted flights		Non-disrupted flights	
Time window	Number of taxi flights	Number of taxi flights	Relative difference	Number of taxi flights	Relative difference	Number of taxi flights	Relative difference
720	205	234	14.15%	205	0.00%	234	14.15%
780	207	227	9.66%	207	0.00%	227	9.66%
840	199	241	21.11%	199	0.00%	241	21.11%
900	181	262	44.75%	181	0.00%	262	44.75%
960	208	238	14.42%	208	0.00%	238	14.42%
1020	204	232	13.73%	204	0.00%	232	13.73%
1080	209	234	11.96%	209	0.00%	234	11.96%
1140	217	238	9.68%	217	0.00%	238	9.68%
1200	204	249	22.06%	204	0.00%	249	22.06%
1260	206	258	25.24%	206	0.00%	258	25.24%

Table 5.10 summarises the average differences, between the default scenario and scenarios for smaller block times for all flights, only disrupted flights and non-disrupted flights inside the RTW. The average difference considers all time windows, and it is possible to confirm previous observations.

Table 5.10 Average differences between the default scenario and smaller block time scenarios

	All flights	Disrupted flights	Non-disrupted flights
Aggregate cost	42.78%	0.00%	43.07%
Computing time	38.96%	1.68%	38.81%
Minutes of delay	54.33%	0.00%	54.33%
Number of cancellations	51.82%	0.00%	51.82%
Number of new flights	-2.00%	0.00%	-2.00%
Number of taxi flights	18.68%	0.00%	18.68%

5.4 Effects of Smaller Block Time for Disruption Scenarios

This section aims to understand if there are any circumstances where having smaller block times render a better solution. The results are presented according to the types of disruption(s) that happen, flight disruption in Subsection 5.4.1, aircraft disruption in Subsection 5.4.2, airport disruption in Subsection 5.4.3, flight and aircraft disruption in Subsection 5.4.4, flight and airport disruption in Subsection 5.4.5 and finally aircraft and airport disruption in Subsection 5.4.6. To prevent the results from overlapping and become invisible, henceforth the latter will be presented using bar graphs. For each of the disruption scenarios an additional table is presented summarising the relative differences for the aggregate cost.

5.4.1 Flight Disruption

In figure 5.6 it is possible to observe that in face of flight disruption, smaller block times when compared with the default scenario, do not show any improvement for the aggregate cost, computing time, number of minutes of delay and number of cancellations. It is only possible to notice an exceedingly small improvement in the number of new flights, time window 1080. As for the number of taxi flights created it is possible to observe that for smaller block times it is triple the number for the default scenario, 6 to 2 respectively. Since the results figure 5.6 do not show any improvement, the next research step is to granulate the data to confirm this observation. Table 5.11, presents the aggregate cost and its relative difference. Finally yet importantly, it is possible to see evidence of improvement whilst having smaller block time, data instances A1, A2, A6, and A7. It is noticeable that these data instances, have a smaller number of flights and the RTW has the duration of one day, as opposed to the remaining B instances where the RTW spans for two days. It is also observable that extending the time window does not impact the improvement for the A data instances, whereas the opposite happens to the B data instances making the aggregate cost increase.

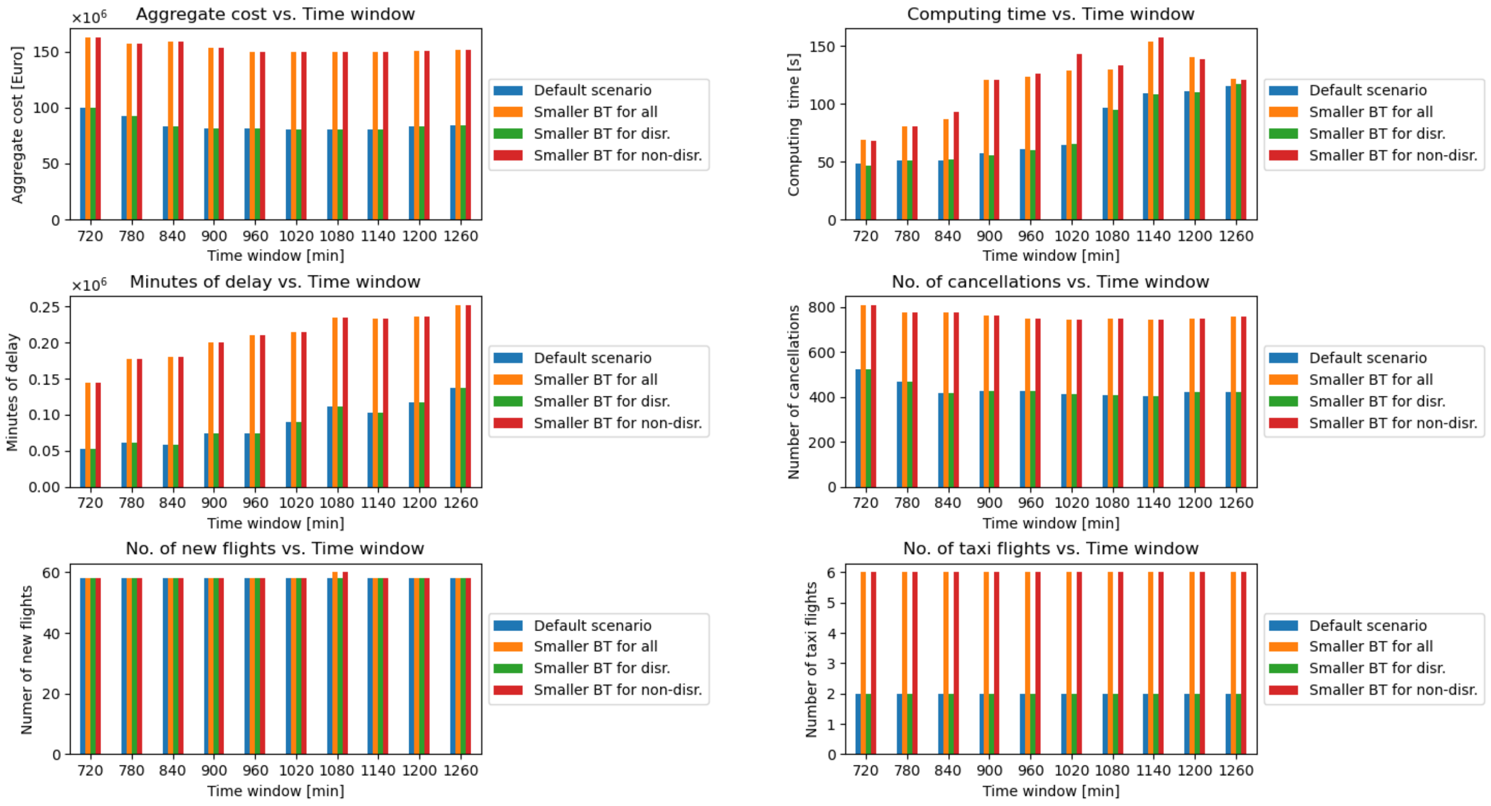


Fig. 5.6 Effects of smaller block time for flight disruption

Table 5.11 Aggregate cost relative differences of smaller block time for flight disruption

		Default scenario	All flights	
Time window	Data instance	Aggregate cost	Aggregate cost	Relative difference
720	A1	31,352.85	28,387.90	-10.44%
720	A2	45,984.10	45,237.95	-1.65%
720	A6	35,965.75	32,540.70	-10.53%
720	A7	57,620.75	56,661.05	-1.69%
720	B1	22,621,894.15	34,604,127.05	34.63%
720	B2	21,990,602.05	38,241,507.20	42.50%
720	B6	28,366,139.10	42,530,924.90	33.30%
720	B7	26,662,429.95	46,979,285.70	43.25%
720 Average relative difference				16.17%
780	A1	31,352.85	28,387.90	-10.44%
780	A2	45,984.10	45,237.95	-1.65%
780	A6	35,965.75	32,540.70	-10.53%
780	A7	57,620.75	56,661.05	-1.69%
780	B1	19,783,267.70	33,488,335.50	40.92%
780	B2	21,551,845.45	37,082,320.70	41.88%
780	B6	24,608,852.55	40,949,484.35	39.90%
780	B7	25,880,693.05	44,873,286.40	42.32%

Table 5.11 Aggregate cost relative differences of smaller block time for flight disruption

		Default scenario	All flights	
Time window	Data instance	Aggregate cost	Aggregate cost	Relative difference
780 Average relative difference				17.59%
840	A1	31,352.85	28,387.90	-10.44%
840	A2	45,984.10	45,237.95	-1.65%
840	A6	35,965.75	32,540.70	-10.53%
840	A7	57,620.75	56,661.05	-1.69%
840	B1	17,216,123.90	33,633,019.25	48.81%
840	B2	20,320,886.45	37,885,946.80	46.36%
840	B6	21,342,334.75	41,220,533.05	48.22%
840	B7	24,262,338.55	45,518,962.20	46.70%
840 Average relative difference				20.72%
900	A1	31,352.85	28,387.90	-10.44%
900	A2	45,984.10	45,237.95	-1.65%
900	A6	35,965.75	32,540.70	-10.53%
900	A7	57,620.75	56,661.05	-1.69%
900	B1	17,003,379.90	31,947,215.40	46.78%
900	B2	19,988,418.95	37,704,760.85	46.99%
900	B6	20,592,662.65	38,518,068.45	46.54%
900	B7	23,989,632.20	45,215,529.50	46.94%
900 Average relative difference				20.37%
960	A1	31,352.85	28,387.90	-10.44%
960	A2	45,984.10	45,237.95	-1.65%
960	A6	35,965.75	32,540.70	-10.53%
960	A7	57,620.75	56,661.05	-1.69%
960	B1	17,003,379.90	31,112,596.45	45.35%
960	B2	19,988,418.95	36,516,349.80	45.26%
960	B6	20,592,662.65	37,675,314.75	45.34%
960	B7	23,989,632.20	43,954,485.60	45.42%
960 Average relative difference				19.63%
1020	A1	31,352.85	28,387.90	-10.44%
1020	A2	45,984.10	45,237.95	-1.65%
1020	A6	35,965.75	32,540.70	-10.53%
1020	A7	57,620.75	56,661.05	-1.69%
1020	B1	16,582,880.50	31,858,155.55	47.95%
1020	B2	19,935,279.55	36,088,951.60	44.76%
1020	B6	19,678,203.40	38,174,834.50	48.45%
1020	B7	23,711,828.90	43,416,577.40	45.39%
1020 Average relative difference				20.28%
1080	A1	31,352.85	28,387.90	-10.44%
1080	A2	45,984.10	45,237.95	-1.65%

Table 5.11 Aggregate cost relative differences of smaller block time for flight disruption

		Default scenario	All flights	
Time window	Data instance	Aggregate cost	Aggregate cost	Relative difference
1080	A6	35,965.75	32,540.70	-10.53%
1080	A7	57,620.75	56,661.05	-1.69%
1080	B1	17,224,058.55	31,395,333.05	45.14%
1080	B2	19,782,494.50	36,560,202.25	45.89%
1080	B6	20,117,658.25	37,783,625.30	46.76%
1080	B7	23,463,628.70	43,788,666.20	46.42%
1080 Average relative difference				19.99%
1140	A1	31,352.85	28,387.90	-10.44%
1140	A2	45,984.10	45,237.95	-1.65%
1140	A6	35,965.75	32,540.70	-10.53%
1140	A7	57,620.75	56,661.05	-1.69%
1140	B1	16,976,934.05	31,934,688.85	46.84%
1140	B2	19,906,616.90	35,629,876.25	44.13%
1140	B6	19,857,512.25	38,748,783.35	48.75%
1140	B7	23,370,848.05	42,886,839.05	45.51%
1140 Average relative difference				20.11%
1200	A1	31,352.85	28,387.90	-10.44%
1200	A2	45,984.10	45,237.95	-1.65%
1200	A6	35,965.75	32,540.70	-10.53%
1200	A7	57,620.75	56,661.05	-1.69%
1200	B1	17,555,330.85	32,363,041.20	45.76%
1200	B2	20,660,593.20	36,055,745.05	42.70%
1200	B6	20,501,129.15	38,968,354.50	47.39%
1200	B7	24,231,551.05	43,387,724.10	44.15%
1200 Average relative difference				19.46%
1260	A1	31,352.85	28,387.90	-10.44%
1260	A2	45,984.10	45,237.95	-1.65%
1260	A6	35,965.75	32,540.70	-10.53%
1260	A7	57,620.75	56,661.05	-1.69%
1260	B1	17,792,420.85	32,746,668.85	45.67%
1260	B2	20,660,593.20	35,748,303.25	42.21%
1260	B6	20,797,398.95	39,374,131.70	47.18%
1260	B7	24,231,551.05	43,258,326.35	43.98%
1260 Average relative difference				19.34%

5.4.2 Aircraft Disruption

In figure 5.7 it is possible to observe that in face of aircraft disruption, smaller block times when compared with the default scenario, do not show any improvement for the aggregate cost, computing time, number of minutes of delay, number of cancellations and number of new flights created. However, the scenario with smaller block times, as opposed to the default scenario, creates taxi flights. Additionally, for the aircraft disruption scenario, the main difference between the smaller block time scenario and the default scenario is the minutes of delay necessary for recovery. The order of magnitude of the time windows greater than 780 is 10^2 . In terms of the relative difference for the aggregate cost, in table 5.12 it is not possible to observe any improvement in the X2 and X4 data instances. The grand average relative difference is 43.43%.

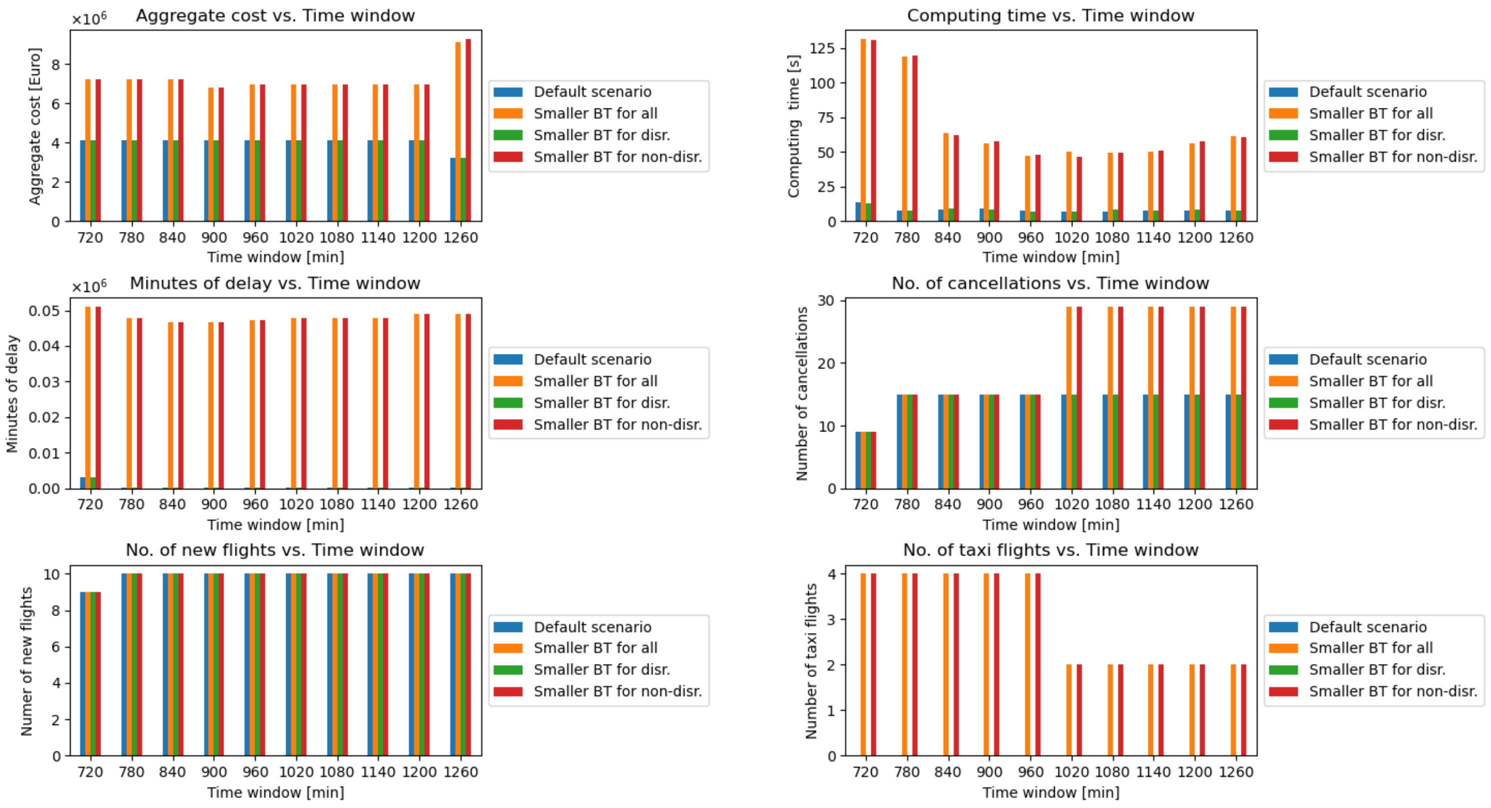


Fig. 5.7 Effects of smaller block time for aircraft disruption

Table 5.12 Aggregate cost relative differences of smaller block time for aircraft disruption

		Default scenario	All flights	
Time window	Data instance	Aggregate cost	Aggregate cost	Relative difference
720	X2	1,960,194.50	3,358,459.15	41.60%
720	X4	2,168,904.95	3,866,983.75	43.91%
720 Average relative difference				42.76%
780	X2	1,960,194.50	3,340,808.15	41.30%
780	X4	2,168,904.95	3,857,586.30	43.78%
780 Average relative difference				42.54%

Table 5.12 Aggregate cost relative differences of smaller block time for aircraft disruption

		Default scenario	All flights	
Time window	Data instance	Aggregate cost	Aggregate cost	Relative difference
840	X2	1,960,194.50	3,340,808.15	41.30%
840	X4	2,168,904.95	3,857,586.30	43.78%
840 Average relative difference				42.54%
900	X2	1,960,194.50	3,114,814.25	37.04%
900	X4	2,168,904.95	3,664,037.05	40.81%
900 Average relative difference				38.92%
960	X2	1,960,194.50	3,262,530.40	39.89%
960	X4	2,168,904.95	3,689,560.10	41.22%
960 Average relative difference				40.55%
1020	X2	1,960,194.50	3,262,530.40	39.89%
1020	X4	2,168,904.95	3,689,560.10	41.22%
1020 Average relative difference				40.55%
1080	X2	1,960,194.50	3,262,530.40	39.89%
1080	X4	2,168,904.95	3,687,975.80	41.19%
1080 Average relative difference				40.54%
1140	X2	1,960,194.50	3,282,076.45	40.25%
1140	X4	2,168,904.95	3,687,975.80	41.19%
1140 Average relative difference				40.72%
1200	X2	1,960,194.50	3,282,076.45	40.25%
1200	X4	2,168,904.95	3,687,975.80	41.19%
1200 Average relative difference				40.72%
1260	X2	1,521,838.45	4,103,283.70	62.91%
1260	X4	1,706,760.70	5,014,345.15	65.96%

5.4.3 Airport Disruption

In figure 5.8, it is possible to observe that in face of airport disruption, once again, smaller block times when compared with the default scenario, do not show any improvement for the aggregate cost, computing time, number of minutes of delay, number of cancellations and number of new flights created. The way the CHARP works it does not create flights if there are none that were initially cancelled by flight or aircraft disruption. Nonetheless, why should they be created if, due to airport capacity shortage, there are no time slots available? In an opposite direction it is possible to observe that, unlike the flight and aircraft disruption scenarios, the CHARP creates a big number of taxi flights. This observation results from the fact that RTW has an early start, and although many flights get cancelled due to the decrease in airport capacity, the CHARP, re-connects the initial airport with the recovered rotation. It is also observable that the size of the time window plays an important role because it allows to extend the minutes of delay.

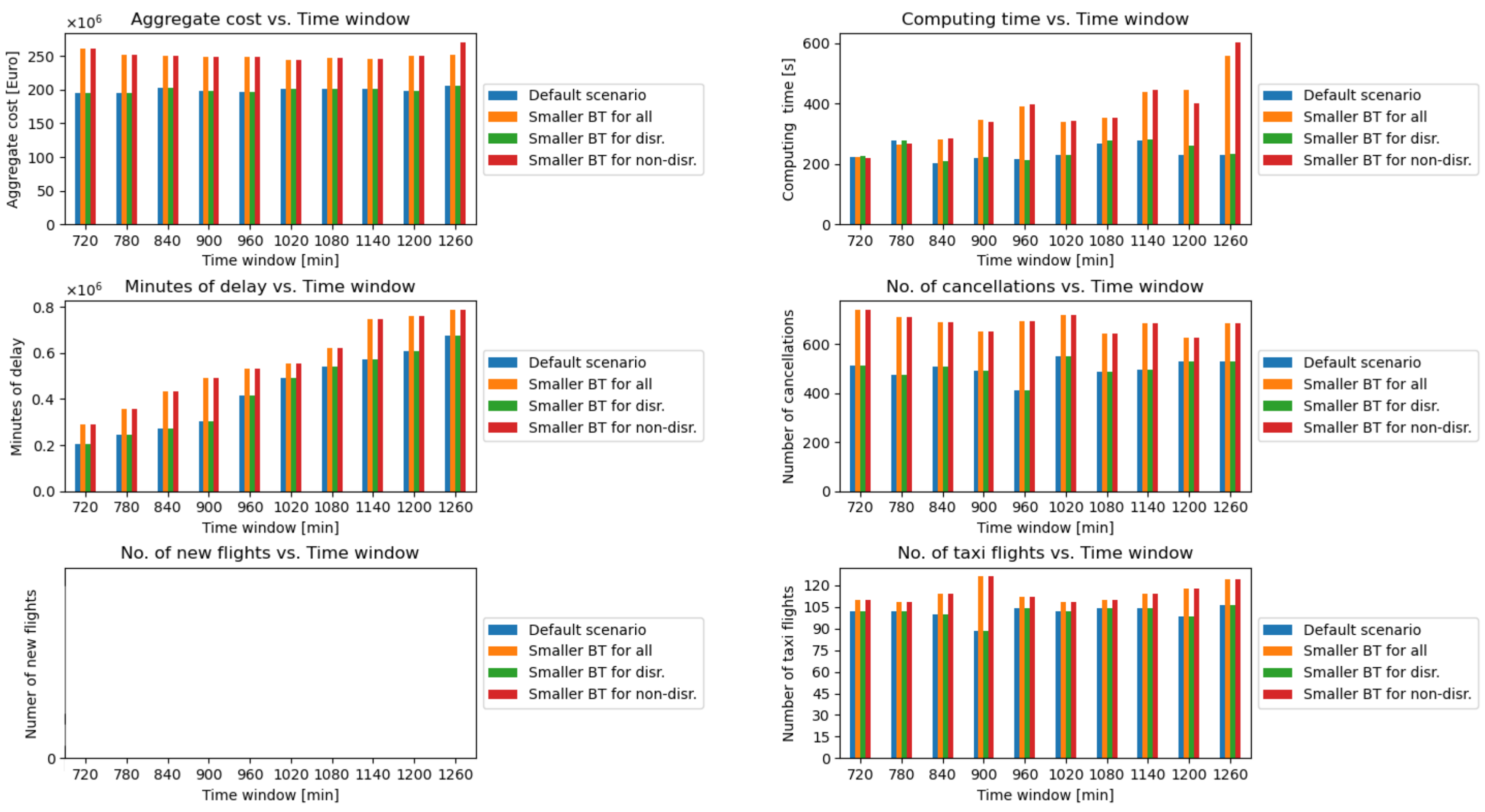


Fig. 5.8 Effects of smaller block time for airport disruption

As for the relative differences in the aggregate cost, even though there is no improvement, it is observable that the average is quite similar to the result obtained for the flight disruption scenario.

Table 5.13 Aggregate cost relative differences of smaller block time for airport disruption

		Default scenario	All flights	
Time window	Data instance	Aggregate cost	Aggregate cost	Relative difference
720	A5	14,690,167.60	16,819,556.75	12.66%
720	A10	17,430,567.25	20,383,437.40	14.49%
720	B5	72,910,445.75	100,414,460.45	27.39%
720	B10	90,838,407.65	124,227,533.20	26.88%
720 Average relative difference				20.35%
780	A5	13,292,798.90	15,278,189.75	12.99%
780	A10	15,963,866.70	19,206,976.70	16.89%
780	B5	74,421,160.15	97,698,257.55	23.83%
780	B10	92,314,065.60	120,175,342.30	23.18%
780 Average relative difference				19.22%
840	A5	15,035,562.80	15,668,232.75	4.04%
840	A10	18,013,321.65	19,215,855.40	6.26%
840	B5	75,573,470.05	97,229,602.45	22.27%
840	B10	93,684,190.90	119,264,216.30	21.45%
840 Average relative difference				13.50%

Table 5.13 Aggregate cost relative differences of smaller block time for airport disruption

		Default scenario	All flights	
Time window	Data instance	Aggregate cost	Aggregate cost	Relative difference
900	A5	13,193,238.25	14,109,892.50	6.50%
900	A10	15,688,235.40	17,296,453.80	9.30%
900	B5	75,337,021.05	98,176,639.30	23.26%
900	B10	93,817,153.20	120,035,355.35	21.84%
900 Average relative difference				15.23%
960	A5	11,996,263.95	13,668,927.45	12.24%
960	A10	13,905,589.90	17,027,685.00	18.34%
960	B5	76,342,060.95	98,053,396.70	22.14%
960	B10	95,024,218.10	119,839,478.55	20.71%
960 Average relative difference				18.36%
1020	A5	13,373,381.00	14,108,576.55	5.21%
1020	A10	15,934,541.90	17,652,190.35	9.73%
1020	B5	77,083,154.55	95,602,772.05	19.37%
1020	B10	94,593,672.20	116,623,881.60	18.89%
1020 Average relative difference				13.30%
1080	A5	12,264,907.45	13,682,095.70	10.36%
1080	A10	15,561,766.60	16,616,151.20	6.35%
1080	B5	77,468,292.30	97,654,744.20	20.67%
1080	B10	96,014,102.30	119,654,576.15	19.76%
1080 Average relative difference				14.28%
1140	A5	12,173,298.80	13,038,527.00	6.64%
1140	A10	14,822,046.40	15,914,844.25	6.87%
1140	B5	77,935,952.30	97,301,366.05	19.90%
1140	B10	96,745,517.35	119,831,090.20	19.27%
1140 Average relative difference				13.17%
1200	A5	12,525,933.85	13,123,189.40	4.55%
1200	A10	14,672,673.05	15,825,720.65	7.29%
1200	B5	76,664,519.60	99,056,206.80	22.61%
1200	B10	94,912,417.45	122,144,704.45	22.30%
1200 Average relative difference				14.18%
1260	A5	13,165,899.70	12,966,643.40	-1.54%
1260	A10	15,556,048.15	15,670,646.05	0.73%
1260	B5	78,823,794.80	100,004,299.10	21.18%
1260	B10	97,821,206.70	123,711,316.50	20.93%
1260 Average relative difference				10.33%

5.4.4 Flight and Aircraft Disruption

In figure 5.9, it is possible to observe the previous pattern of no improvement for the aggregate cost, computing time, number of minutes of delay and number of cancellations. Additionally, with the exceptions of the 780 and 840 time windows the number of new flights created for the smaller block time scenario, the remaining time windows do not show any improvement. Unlike the default scenario, the smaller block time scenario creates taxi flights.

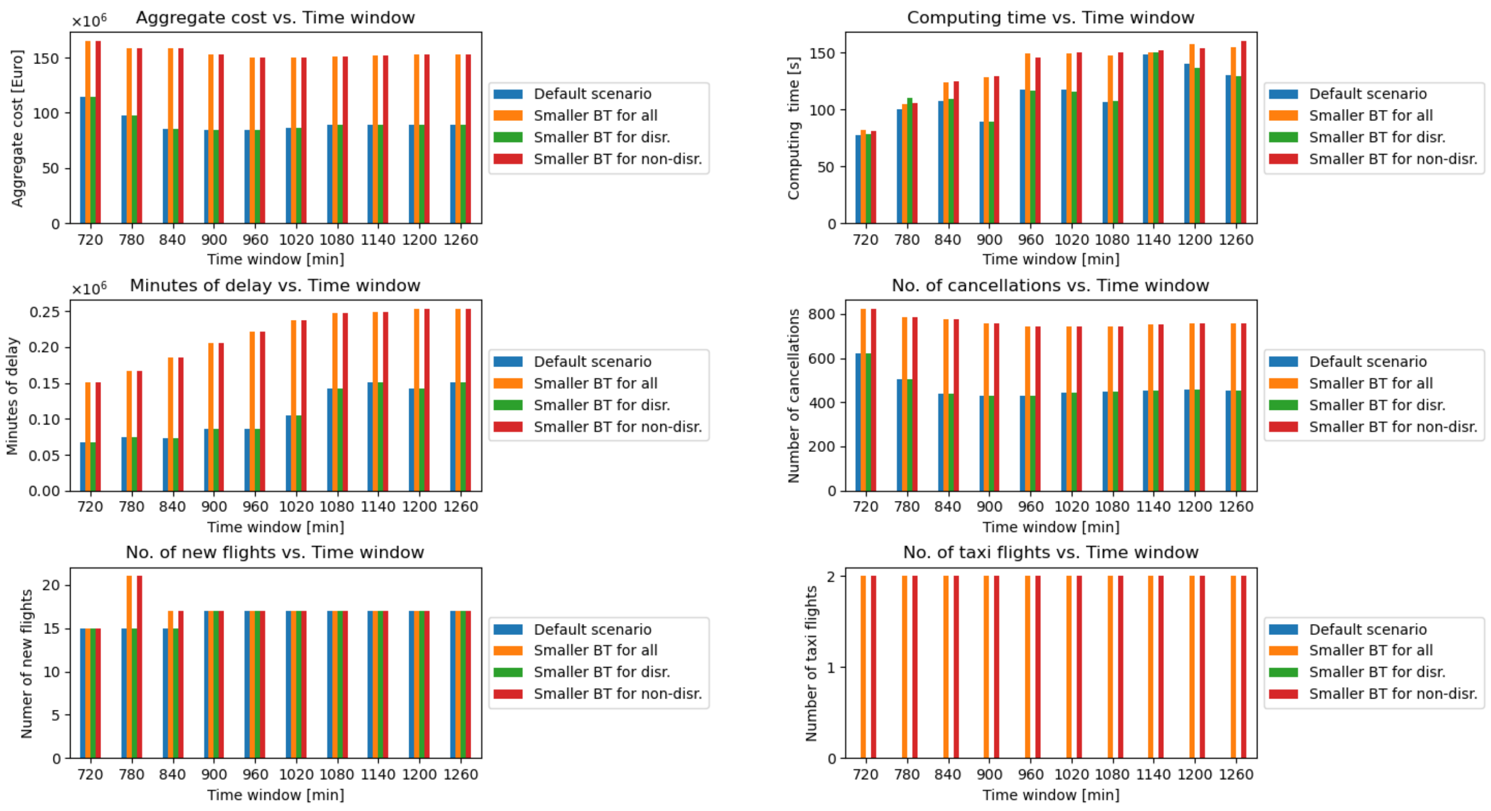


Fig. 5.9 Effects of smaller block time for flight and aircraft disruption

As for the relative difference in the aggregate cost, it is possible to observe in table 5.14 that for some of the values are quite similar having values lower than 1%, while others are greater than 40%.

Table 5.14 Aggregate cost relative differences of smaller block time for flight and aircraft disruption

		Default scenario	All flights	
Time window	Data instance	Aggregate cost	Aggregate cost	Relative difference
720	A3	531,738.20	541,115.00	1.73%
720	A8	758,704.45	773,527.85	1.92%
720	B3	23,286,201.60	35,343,794.35	34.12%
720	B8	29,151,411.80	42,948,791.90	32.13%
720	XA1	1,031,607.95	1,040,984.75	0.90%
720	XA3	1,479,332.15	1,494,155.55	0.99%
720	XB1	24,941,586.90	36,761,481.30	32.15%
720	XB3	32,887,911.55	46,234,619.50	28.87%

Table 5.14 Aggregate cost relative differences of smaller block time for flight and aircraft disruption

		Default scenario	All flights	
Time window	Data instance	Aggregate cost	Aggregate cost	Relative difference
720 Average relative difference				16.60%
780	A3	534,261.20	536,959.75	0.50%
780	A8	763,195.45	766,615.70	0.45%
780	B3	19,964,072.65	33,804,938.60	40.94%
780	B8	24,707,993.95	41,432,969.30	40.37%
780	XA1	1,034,130.95	1,036,829.50	0.26%
780	XA3	1,483,823.15	1,487,243.40	0.23%
780	XB1	21,900,819.65	35,639,932.70	38.55%
780	XB3	27,099,023.10	44,125,800.20	38.59%
780 Average relative difference				19.99%
840	A3	540,810.20	543,619.40	0.52%
840	A8	769,363.45	773,008.70	0.47%
840	B3	17,062,641.65	33,875,372.20	49.63%
840	B8	20,958,015.95	41,286,222.55	49.24%
840	XA1	1,040,679.95	1,043,489.15	0.27%
840	XA3	1,489,991.15	1,493,636.40	0.24%
840	XB1	18,988,754.75	35,690,198.80	46.80%
840	XB3	24,407,161.90	43,778,373.20	44.25%
840 Average relative difference				23.93%
900	A3	540,810.20	543,619.40	0.52%
900	A8	769,363.45	773,008.70	0.47%
900	B3	16,875,549.05	32,694,235.45	48.38%
900	B8	20,313,346.15	39,722,769.80	48.86%
900	XA1	1,040,679.95	1,043,489.15	0.27%
900	XA3	1,489,991.15	1,493,636.40	0.24%
900	XB1	18,863,732.05	33,811,441.70	44.21%
900	XB3	24,541,802.45	42,797,119.50	42.66%
900 Average relative difference				23.20%
960	A3	540,810.20	543,619.40	0.52%
960	A8	769,363.45	773,008.70	0.47%
960	B3	16,873,008.95	32,103,448.15	47.44%
960	B8	20,308,031.35	38,829,065.50	47.70%
960	XA1	1,040,679.95	1,043,489.15	0.27%
960	XA3	1,489,991.15	1,493,636.40	0.24%
960	XB1	18,837,569.15	32,953,562.00	42.84%
960	XB3	24,541,802.45	42,082,409.35	41.68%
960 Average relative difference				22.64%
1020	A3	540,810.20	543,619.40	0.52%
1020	A8	769,363.45	773,008.70	0.47%

Table 5.14 Aggregate cost relative differences of smaller block time for flight and aircraft disruption

		Default scenario	All flights	
Time window	Data instance	Aggregate cost	Aggregate cost	Relative difference
1020	B3	17,678,534.15	32,404,863.60	45.44%
1020	B8	20,929,492.20	38,743,029.80	45.98%
1020	XA1	1,040,679.95	1,043,489.15	0.27%
1020	XA3	1,489,991.15	1,493,636.40	0.24%
1020	XB1	19,713,518.75	32,480,889.15	39.31%
1020	XB3	24,508,113.90	42,547,592.50	42.40%
1020 Average relative difference				21.83%
1080	A3	540,810.20	543,619.40	0.52%
1080	A8	769,363.45	773,008.70	0.47%
1080	B3	18,481,652.10	32,339,201.05	42.85%
1080	B8	21,477,335.65	38,755,353.20	44.58%
1080	XA1	1,040,679.95	1,043,489.15	0.27%
1080	XA3	1,489,991.15	1,493,636.40	0.24%
1080	XB1	20,434,861.90	33,135,636.90	38.33%
1080	XB3	24,746,768.95	42,626,127.65	41.94%
1080 Average relative difference				21.15%
1140	A3	540,810.20	543,619.40	0.52%
1140	A8	769,363.45	773,008.70	0.47%
1140	B3	18,286,579.70	32,592,913.95	43.89%
1140	B8	21,270,807.65	39,454,283.75	46.09%
1140	XA1	1,040,679.95	1,043,489.15	0.27%
1140	XA3	1,489,991.15	1,493,636.40	0.24%
1140	XB1	20,313,176.70	33,442,610.15	39.26%
1140	XB3	25,403,670.05	42,778,619.25	40.62%
1140 Average relative difference				21.42%
1200	A3	540,810.20	543,619.40	0.52%
1200	A8	769,363.45	773,008.70	0.47%
1200	B3	18,392,900.60	32,872,615.40	44.05%
1200	B8	21,606,507.25	39,533,524.50	45.35%
1200	XA1	1,040,679.95	1,043,489.15	0.27%
1200	XA3	1,489,991.15	1,493,636.40	0.24%
1200	XB1	20,152,075.20	34,484,967.05	41.56%
1200	XB3	25,526,286.45	42,470,592.30	39.90%
1200 Average relative difference				21.54%
1260	A3	540,810.20	543,619.40	0.52%
1260	A8	769,363.45	773,008.70	0.47%
1260	B3	18,392,900.60	32,872,615.40	44.05%
1260	B8	21,606,507.25	39,533,524.50	45.35%
1260	XA1	1,040,679.95	1,043,489.15	0.27%

Table 5.14 Aggregate cost relative differences of smaller block time for flight and aircraft disruption

		Default scenario	All flights	
Time window	Data instance	Aggregate cost	Aggregate cost	Relative difference
1260	XA3	1,489,991.15	1,493,636.40	0.24%
1260	XB1	19,960,976.30	34,484,967.05	42.11%
1260	XB3	25,064,203.65	42,474,093.30	40.99%
1260 Average relative difference				21.75%

5.4.5 Flight and Airport Disruption

In figure 5.9, it is possible to observe the previous pattern for flight and aircraft disruption. However, for the flight and airport disruption, there is no creation of new flights. As for taxi flights both disruptions, flight and aircraft, flight and airport, create the same amount.

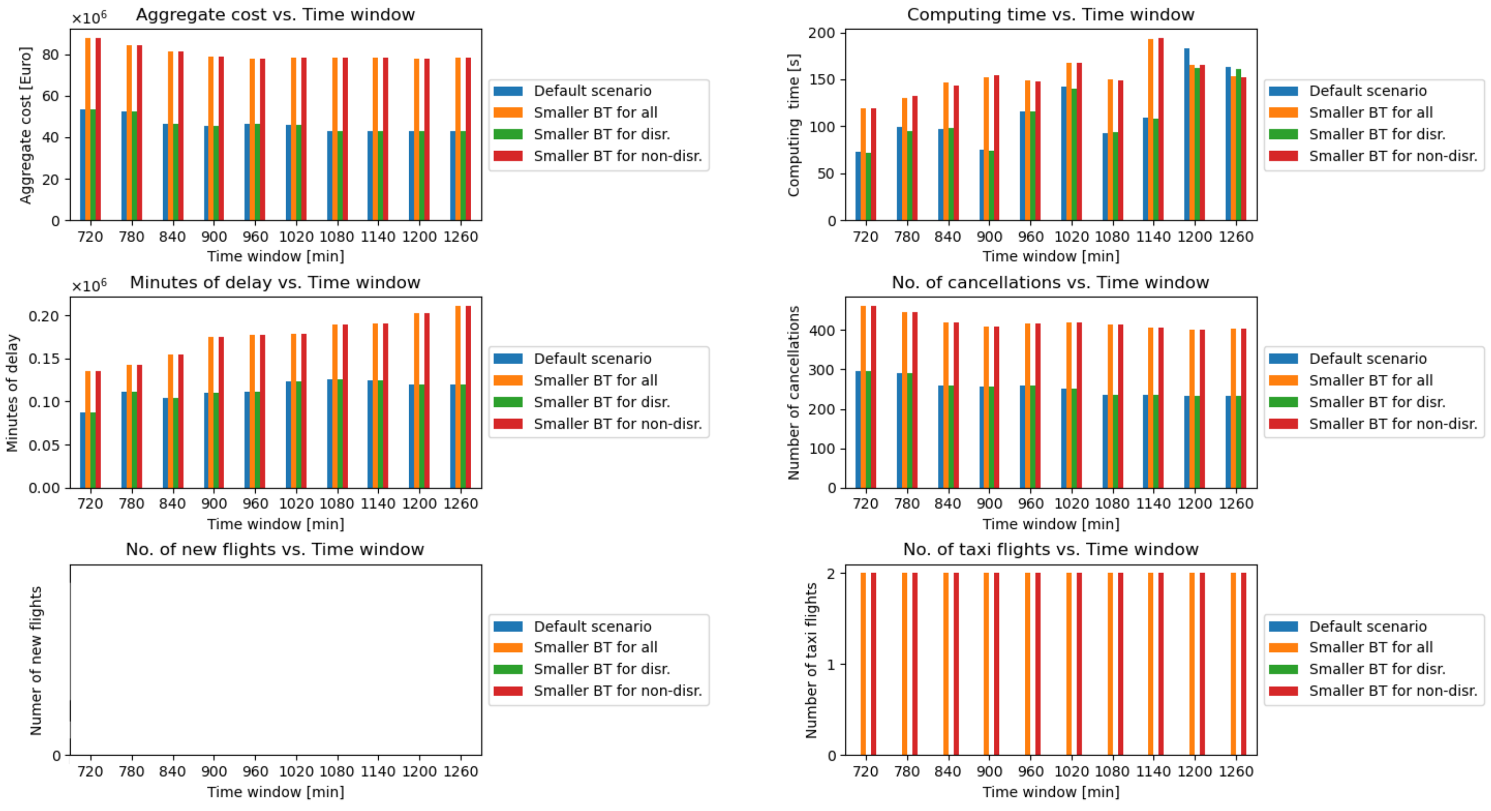


Fig. 5.10 Effects of smaller block time for flight and airport disruption

In table 5.16 it is possible to observe a substantial improvement in the aggregate cost for the A1 and A9 instances. This improvement tends to increase with the size of the time window and stabilises for 960 minutes onward. However, in the opposite direction, it is possible to observe an increase in the aggregate cost relative difference for the B4 and B9 instances, the latter having a bigger number of aircraft and flights and their RTW extends for two days. Lastly, the grand average relative difference for the aggregate cost is negative.

Table 5.15 Aggregate cost relative differences of smaller block time for flight and airport disruption

		Default scenario	All flights	
Time window	Data instance	Aggregate cost	Aggregate cost	Relative difference
720	A4	441,085.65	415,272.30	-6.22%
720	A9	506,093.80	469,780.75	-7.73%
720	B4	23,286,878.25	39,167,481.35	40.55%
720	B9	29,278,539.50	47,897,637.85	38.87%
720 Average relative difference				16.37%
780	A4	446,080.65	222,232.30	-100.73%
780	A9	514,448.80	278,950.40	-84.42%
780	B4	22,983,934.40	37,546,809.65	38.79%
780	B9	28,624,255.65	46,312,585.45	38.19%
780 Average relative difference				-27.04%
840	A4	446,080.65	222,134.35	-100.82%
840	A9	514,448.80	280,273.40	-83.55%
840	B4	20,337,460.50	36,535,033.05	44.33%
840	B9	25,306,862.75	44,631,284.30	43.30%
840 Average relative difference				-24.18%
900	A4	451,516.65	222,134.35	-103.26%
900	A9	521,735.80	280,273.40	-86.15%
900	B4	20,282,190.40	35,279,667.65	42.51%
900	B9	24,421,705.85	43,239,592.95	43.52%
900 Average relative difference				-25.85%
900	A4	457,195.65	222,134.35	-105.82%
900	A9	526,784.80	280,273.40	-87.95%
900	B4	20,502,079.35	35,030,036.55	41.47%
900	B9	24,786,779.25	42,566,603.65	41.77%
900 Average relative difference				-27.63%
1020	A4	457,195.65	222,134.35	-105.82%
1020	A9	526,784.80	280,273.40	-87.95%
1020	B4	20,380,647.75	35,050,635.40	41.85%
1020	B9	24,446,483.80	42,722,467.70	42.78%
1020 Average relative difference				-27.29%
1020	A4	457,195.65	222,134.35	-105.82%
1020	A9	526,784.80	280,273.40	-87.95%
1020	B4	19,118,226.45	35,297,341.50	45.84%
1020	B9	22,735,433.05	42,531,879.20	46.54%
1020 Average relative difference				-25.35%
1140	A4	457,195.65	222,134.35	-105.82%
1140	A9	526,784.80	280,273.40	-87.95%
1140	B4	19,291,054.95	35,208,102.30	45.21%
1140	B9	22,936,471.85	42,904,318.15	46.54%

Table 5.15 Aggregate cost relative differences of smaller block time for flight and airport disruption

		Default scenario	All flights	
Time window	Data instance	Aggregate cost	Aggregate cost	Relative difference
1140 Average relative difference				-25.51%
1200	A4	457,195.65	222,134.35	-105.82%
1200	A9	526,784.80	280,273.40	-87.95%
1200	B4	19,179,376.65	35,178,144.25	45.48%
1200	B9	22,878,868.35	42,496,090.70	46.16%
1200 Average relative difference				-25.53%
1260	A4	457,195.65	222,134.35	-105.82%
1260	A9	526,784.80	280,273.40	-87.95%
1260	B4	19,179,376.65	35,347,476.20	45.74%
1260	B9	22,878,868.35	42,725,864.50	46.45%
1260 Average relative difference				-25.40%

5.4.6 Aircraft and Airport disruption

Having reached the final subsection it is possible, once again, to observe in figure 5.11, the same patterns. There is a trend of increasing minutes of delay whilst increasing the time window. As for taxi flights both disruptions, airport, aircraft and airport, create the same amount.

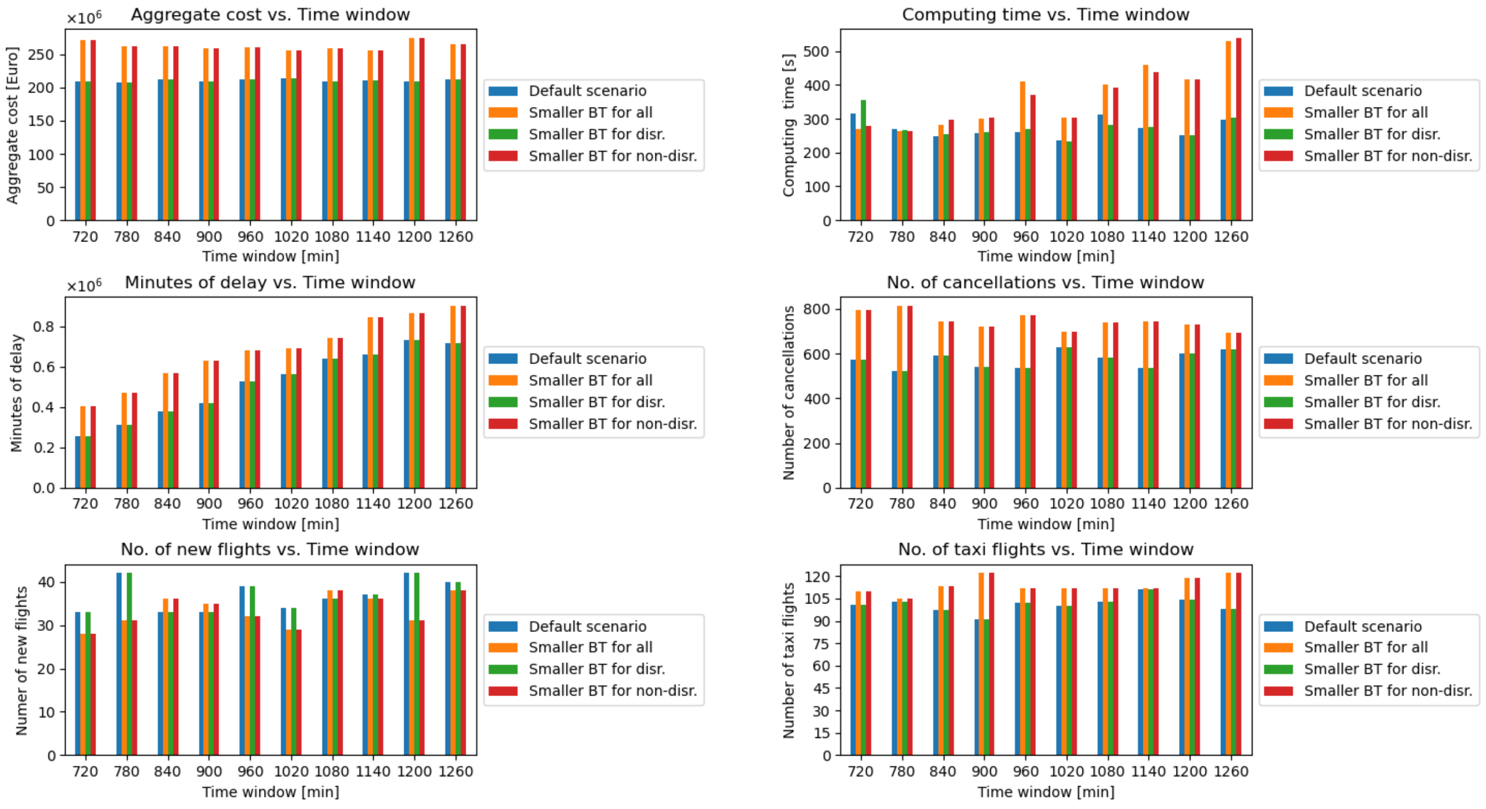


Fig. 5.11 Effects of smaller block time for aircraft and airport disruption

In terms of improvement, it is possible to observe from table 5.16 improvement happens for the XA2 and X3 data instances in the 720 and 1260 time windows. Other than this there is no improvement for the remaining data instances.

Table 5.16 Aggregate cost relative differences of smaller block time for aircraft and airport disruption

		Default scenario	All flights	
Time window	Data instance	Aggregate cost	Aggregate cost	Relative difference
720	X1	2,550,004.50	4,080,152.05	37.50%
720	X3	3,214,415.70	3,815,975.80	15.76%
720	XA2	17,847,730.55	17,456,133.35	-2.24%
720	XA4	20,900,892.60	21,212,335.05	1.46%
720	XB2	72,783,526.25	100,990,299.05	27.93%
720	XB4	92,121,412.40	124,927,990.05	26.26%
720 Average relative difference				17.78%
780	X1	2,542,654.50	3,902,640.30	34.85%
780	X3	2,706,850.90	3,144,737.85	13.92%
780	XA2	15,407,902.35	16,072,256.85	4.13%
780	XA4	18,372,614.90	20,783,376.95	11.60%
780	XB2	76,420,706.30	97,827,953.60	21.88%
780	XB4	92,525,663.30	121,114,774.45	23.60%
780 Average relative difference				18.33%
840	X1	2,554,408.50	3,817,111.55	33.08%
840	X3	2,696,107.90	3,144,737.85	14.27%
840	XA2	16,009,015.35	16,400,173.05	2.39%
840	XA4	19,225,213.30	19,840,723.60	3.10%
840	XB2	75,986,401.90	98,009,127.30	22.47%
840	XB4	95,464,035.35	120,621,377.75	20.86%
840 Average relative difference				16.03%
900	X1	2,554,408.50	3,809,067.95	32.94%
900	X3	2,702,032.90	3,168,134.85	14.71%
900	XA2	15,084,847.00	15,123,021.00	0.25%
900	XA4	17,543,948.65	18,578,127.10	5.57%
900	XB2	75,752,479.20	98,968,982.70	23.46%
900	XB4	95,638,338.55	119,386,205.50	19.89%
900 Average relative difference				16.14%
960	X1	2,539,858.50	3,804,153.95	33.23%
960	X3	2,706,601.90	3,166,256.85	14.52%
960	XA2	14,725,586.80	15,278,221.05	3.62%
960	XA4	17,664,267.50	18,877,820.60	6.43%
960	XB2	77,646,446.30	98,404,578.05	21.09%
960	XB4	96,498,068.00	120,829,182.10	20.14%
960 Average relative difference				16.50%
1020	X1	2,547,361.50	3,811,656.95	33.17%

Table 5.16 Aggregate cost relative differences of smaller block time for aircraft and airport disruption

		Default scenario	All flights	
Time window	Data instance	Aggregate cost	Aggregate cost	Relative difference
1020	X3	2,690,044.90	4,261,966.05	36.88%
1020	XA2	14,377,296.55	14,417,920.20	0.28%
1020	XA4	17,241,635.25	18,041,630.30	4.43%
1020	XB2	80,323,908.55	96,669,121.55	16.91%
1020	XB4	97,282,323.85	119,325,900.10	18.47%
1020 Average relative difference				18.36%
1080	X1	2,547,361.50	3,489,358.65	27.00%
1080	X3	2,691,943.90	4,275,459.75	37.04%
1080	XA2	13,337,800.00	14,672,371.65	9.10%
1080	XA4	16,642,069.95	17,741,346.45	6.20%
1080	XB2	76,883,295.95	98,334,513.70	21.81%
1080	XB4	97,522,717.30	120,434,206.85	19.02%
1080 Average relative difference				20.03%
1140	X1	2,547,493.50	3,508,904.70	27.40%
1140	X3	2,693,494.90	4,252,149.75	36.66%
1140	XA2	12,271,738.55	13,793,214.75	11.03%
1140	XA4	14,783,507.45	16,658,393.55	11.25%
1140	XB2	78,790,039.55	96,458,551.95	18.32%
1140	XB4	99,102,529.75	121,579,624.45	18.49%
1140 Average relative difference				20.52%
1200	X1	2,547,493.50	3,508,904.70	27.40%
1200	X3	2,705,779.90	3,949,310.25	31.49%
1200	XA2	13,430,461.85	14,301,864.55	6.09%
1200	XA4	15,987,588.95	17,216,739.30	7.14%
1200	XB2	76,468,628.65	99,734,763.85	23.33%
1200	XB4	98,626,459.00	136,176,968.05	27.57%
1200 Average relative difference				20.50%
1260	X1	2,359,490.25	3,508,904.70	32.76%
1260	X3	4,110,622.20	3,949,310.25	-4.08%
1260	XA2	13,696,970.60	14,201,588.25	3.55%
1260	XA4	16,356,905.10	16,900,456.35	3.22%
1260	XB2	78,522,145.95	100,528,698.80	21.89%
1260	XB4	97,744,790.50	126,771,805.55	22.90%
1260 Average relative difference				13.37%

5.5 Conclusions and Future Work

This chapter analyses the effects of replacing the original block times with smaller ones computed by the BTF model presented in Chapter 3. The analysis of results obtained by the BTF for the block time showed they were in agreement with the real ones, obtained from Flightaware™. Additionally, it also demonstrated that consumed fuel had no significant differences when compared with the planned values obtained from Lido™. Therefore, no additional cost was introduced, upon replacement.

The disrupted rotations were subsequently recovered using the CHARP defined in Chapter 4. The analysis presented in Section 5.4 compared the default scenario (Section 4.5.1, table 4.4) with the introduction of smaller block times for all flights, disrupted flights and non-disrupted flights, during the RTW. The introduction of smaller block times aimed at understanding if these would reduce the cost of recovery, therefore proving to be a suitable strategy. It is possible to conclude that, inside the RTW there are no differences between the default scenario and the disrupted flights with smaller block times scenario. Similarly, there are no differences between the scenarios having all or only non-disrupted flights, with smaller block times. Most strikingly, it was not possible to detect any improvement using smaller block.

In principle this result was not expected, one is to expect that if a flight is delayed, speeding it would mitigate the effects of this disruption. However, there is more than one scenario of disruption to be considered, therefore Section 5.4 analysis six different disruption scenarios. It was possible to demonstrate that, for flight, flight and airport, disruption scenarios, reducing block times can render a decrease in the recovery cost. The reduction in the recovery aggregate cost was also demonstrated, on a small scale for the aircraft and airport disruption scenario. In relation to published work, the results presented in this chapter not only confirm the conclusions obtained in (Aktürk et al., 2014), but also extend the scope of the research in terms of the nature of the disruption. It is important to mention that speeding up flights may not be a suitable strategy in the face of airport capacity shortages. This chapter presents an extended result set in terms of number of aircraft and flights, table 4.9, and also presents results classified according to 6 disruption scenarios.

Although the BTF modelling does not include CI changes it is also possible to confirm (Marla et al., 2017). Unlike the latter, the flight modelling performed in this thesis was done without the use of commercial software.

The final conclusion is that reducing block time to achieve recovery with lower cost is highly dependent on the nature of the disruption, the size of the RTW, and the time window of delay.

In terms of future work, it would be worth considering changing the order in which new flights are created, making sure that they are allocated to airports with available capacity. Added to this feature, swapping aircraft is a very common recovery feature that could be included in the recovery strategy.

Chapter 6

Contributions and Future Work

This chapter summarises the contributions of this dissertation, a flight planning model, a novel disruption recovery model based in CP and a model for disruption recovery using smaller block times, Section 6.1. Additionally, since any model has its shortcomings, Section 6.2 presents a set of topics that can improve and extend the models presented.

6.1 Contributions

The first step that was taken during research was to understand the envelopment regarding commercial aviation. However, due to the size and complexity of this industry, this was narrowed down to flight planning. The literature review demonstrated that research is mostly focused on specific phases of the flight, and to our best knowledge there are no flight planning models published that encompass all the flight phases for a specific aircraft model. On this note, this thesis gathered the data for ground movements, air phases and aircraft performance to create the BTF model. Using EMEP/EEA historical data from pollutant emissions the BTF model is able to find how long it takes for an aircraft to taxi-out, take-off, approach, land and taxi-in. Added to the latter by using BADA aircraft performance tables, and Newtonian mechanics it is possible to model CCD phases and calculate ground distance, flight times and consumed fuel. The results for block time obtained from the BTF model were benchmarked against the real ones obtained from Flightaware™, using percentile and RMSE. This comparison was also made against published work and Lido™flight plans. It was possible to conclude that the block time result sets computed by the BTF were quite accurate. In terms of consumed fuel, it was also possible to confirm the accuracy of the BTF model results, by comparing them with published literature and Lido™. Other than being an accurate model the BTF model, also proved to be very efficient in terms of computing time.

On occasions, flights cannot be operated according to the original flight plan. There is a series of factors that affect a flight to the point that it can become delayed or cancelled. Events such as aircraft malfunction, inclement weather and industrial action can cause disruptions in the aircraft rotation, making it infeasible. Hence, faced with disruption, ATM systems are fundamental to provide recovery solutions that can mitigate disruption costs in a reasonable computing time. Disruption recovery in commercial aviation encompasses three steps, first the aircraft rotations, second the crew roster and third passenger itineraries. ATMs follow this order of recovery and every time a feasible solution cannot be found they backtrack to the previous step. From the standpoint of computer science, disruption recovery in commercial aviation consists of a NP-hard combinatorial problem. Due to the aforementioned complexity this thesis focus solely on the recovery of aircraft rotations, commonly known as the ARP. Since the latter is a scheduling problem, it can be modelled as a CSP. The approach that is most suited to solve CSP is CP. According to Google™:

"CP is based on feasibility (finding a feasible solution) rather than optimisation (finding an optimal solution) and focuses on the constraints and variables rather than the objective function. In fact, a CP problem may not even have an objective function — the goal may simply be to narrow down a very large set of possible solutions to a more manageable subset by adding constraints to the problem."

Chapter 4, defines the ARP model and introduces CP concepts, such as CPr and backtracking. The former two are introduced into the CHARP. The CHARP consists of a set of nine algorithms that upon finding infeasible rotations, recovers them by creating new flights, adding delays, cancellations and creating taxi flights. To perform these actions the CHARP finds the flight domains and computes the search space. If the search space size is greater than the upper bound, the upper heuristic tries to perform the recovery. On the other hand, if the search space size is below the lower bound the procedure using CPr, reduces the size of the search space even further. Afterwards, the lower heuristic algorithm tries to perform the recovery. Whenever recovery is not possible the CHARP performs backtracking and removes a recovered rotation from the solution. Additionally, if the recovered rotation breaks continuity from the original airport, the CHARP reconnects it by creating taxi flights. The CHARP handles a spectrum of search spaces with sizes from 10^1 to 10^{25} using a meta-heuristic that performs a pincer movement, combining the upper and the lower heuristic. By reducing the size of the search space, the CPr algorithm also reduces computing time. Comparisons with published literature demonstrate that the CHARP can handle six distinct disruption scenarios, and the biggest number of flights, aircraft and airports. The computing time also proved to be suited for industrial applications.

In recent years disruption recovery research has been focusing on cruise speed changes. Published work is promising however it does not get into detail regarding the disruption scenarios, the planning horizon or the aircraft model. By integrating the BTF with the CHARP Chapter 5 includes all these factors. In Section 5.4 flight, flight and airport, disruption scenarios, demonstrated that reducing block times can render a decrease in the recovery cost for some data instances. Hence, the results show that it is possible to reduce the cost of recovery by reducing block time however, this conclusion is highly dependent on the problem's nature.

Finally, it is important to mention that the BTF model is published and cited in (Montlaur et al., 2021). The CHARP was presented in the OR63 Conference on the 14^{th} of September under the category "Applications in Aviation".

6.2 Future Work

The BTF assumed that the Earth is a perfect sphere, while in fact the Earth is not a perfect sphere but rather closer to an ellipsoid. However, even an ellipsoid does not adequately describe the Earth's unique and ever-changing shape. Our planet is "pudgier" at the equator than at the poles by about 21 Km. This is due to the centrifugal force created by the earth's constant rotation. During the flight, the BTF model also assumes the aircraft does not change its weight. This assumption impacts modelling because the BTF uses the constant nominal mass level defined in the BADA aircraft performance tables. In fact, the aircraft can start at a high mass level and afterwards change to nominal due to the loss of weight. This change has an impact, no less than, on the ROCD, fuel consumption rate, engine thrust, cruising altitude, and

TAS. Due to the jet stream, the BTF distinguishes eastbound from westbound flights however, it does not incorporate the daily weather conditions. The weather and in particular the wind have an important effect on the aircraft speed. Ultimately inclement weather can have a strong impact on flight planning to the point of causing flight cancellation.

In order to mitigate the effects of disruption, this thesis developed and presented the CHARP. This heuristic uses a set of steps to perform recovery, which do not include aircraft swap as a recovery strategy. The latter can become part of future work and in addition, the CHARP can be extended to include an improvement phase that can introduce additional flights.

The observations provided in in Chapters 3, 4 and 5 leads to conclude that this thesis opens many possibilities for disruption recovery. The introduction of smaller block times can on occasions reduce the recovery cost, therefore it would be interesting to understand in which situations flights should be sped up. This procedure would require the introduction of itineraries in the objective function and due to the added complexity, it would be important to consider the introduction of a hyper-heuristic to guide the solution.

At the moment the author of this thesis is writing the paper "Computing Time Benefits of Constraint Propagation". Overall the work presented in this thesis has major contributions and can be extended with further investigation.

References

Jeph Abara. Applying integer linear programming to the fleet assignment problem. *Interfaces*, 19(4): 20–28, 1989.

Flightaware Flight AFR1148. Flight AFR1148, 2019a. URL https://uk.flightaware.com/live/flight/AFR1148/history/20191023/0520Z/LFPG/LEBL.

Flightaware Flight Track Log AFR1148. Flight Track Log - AFR1148, 2019b. URL https://uk.flightaware.com/live/flight/AFR1148/history/20191023/0520Z/LFPG/LEBL/tracklog.

Shervin AhmadBeygi, Amy Cohn, Yihan Guan, and Peter Belobaba. Analysis of the potential for delay propagation in passenger airline networks. *Journal of Air Transport Management*, 14(5):221 – 236, 2008. ISSN 0969-6997. doi: https://doi.org/10.1016/j.jairtraman.2008.04.010. URL http://www.sciencedirect.com/science/article/pii/S0969699708000550.

Airlines of America. Annual financial results: World airlines. 2022. URL https://www.airlines.org/dataset/annual-results-world-airlines.

M. Selim Aktürk, Alper Atamtürk, and Sinan Gürel. Aircraft rescheduling with cruise speed control. *Operations Research*, 62(4):829–845, aug 2014. doi: 10.1287/opre.2014.1279.

S Alam, J Tang, C J Lokan, and H A Abbass. An assessment of BADA fuel flow methodologies for in-trail procedure evaluation. *Defence & Security Applications Research Centre, University of New South Wales, Australian Defence Force Academy, Canberra, Australia*, 2009.

Cyril Allignol, Nicolas Barnier, Pierre Flener, and Justin Pearson. Constraint programming for air traffic management: a survey. *The Knowledge Engineering Review*, 27(3):361–392, jul 2012. doi: 10.1017/s0269888912000215.

T. Andersson and P. Värbrand. The flight perturbation problem. *Transportation Planning and Technology*, 27(2):91–117, apr 2004. doi: 10.1080/0308106042000218195.

Tobias Andersson. Solving the flight perturbation problem with meta heuristics. *Journal of Heuristics*, 12 (1-2):37–53, January 2006. ISSN 1381-1231.

Michael F. Arguello, Jonathan F. Bard, and Gang Yu. A grasp for aircraft routing in response to groundings and delays. *Journal of Combinatorial Optimization*, 1(3):211–228, 1997.

Pol Arias, Miguel Mujica Mota, Daniel Guimarans, and Geert Boosten. A methodology combining optimization and simulation for real applications of the stochastic aircraft recovery problem. In *2013 8th EUROSIM Congress on Modelling and Simulation*. IEEE, sep 2013. doi: 10.1109/eurosim.2013.55.

Ugur Arikan, Sinan Gurel, and M. Selim Akturk. Integrated aircraft and passenger recovery with cruise time controllability. *Annals of Operations Research*, 236(2):295–317, Jan 2016. ISSN 1572-9338. doi: 10.1007/s10479-013-1424-2. URL https://doi.org/10.1007/s10479-013-1424-2.

Uğur Arıkan, Sinan Gürel, and M. Selim Aktürk. Flight network-based approach for integrated airline recovery with cruise speed control. *Transportation Science*, 51(4):1259–1287, nov 2017. doi: 10.1287/trsc.2016.0716.

Philippe Baptiste, Claude Le Pape, and Wim Nuijten. *Constraint-based scheduling: applying constraint programming to scheduling problems*, volume 39. Springer Science & Business Media, 2001.

Nicolas Bélanger, Guy Desaulniers, François Soumis, and Jacques Desrosiers. Periodic airline fleet assignment with time windows, spacing constraints, and time dependent revenues. *European Journal of Operational Research*, 175(3):1754–1766, dec 2006. doi: 10.1016/j.ejor.2004.04.051.

Christian Bessière. *Basic CP Theory: Consistency And Propagation (Advanced)*. 01 2011. doi: 10.1002/9780470400531.eorms0087.

John T Betts and Evin J Cramer. Application of direct transcription to commercial aircraft trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 18(1):151–159, 1995.

Serge Bisaillon, Jean-François Cordeau, Gilbert Laporte, and Federico Pasin. A large neighbourhood search heuristic for the aircraft and passenger recovery problem. *4OR: A Quarterly Journal of Operations Research*, 9(2):139–157, 2011.

Rupak Biswas, Zhang Jiang, Kostya Kechezhi, Sergey Knysh, Salvatore Mandrà, Bryan O'Gorman, Alejandro Perdomo-Ortiz, Andre Petukhov, John Realpe-Gómez, Eleanor Rieffel, Davide Venturelli, Fedir Vasko, and Zhihui Wang. A nasa perspective on quantum computing: Opportunities and challenges. *Parallel Computing*, 64:81–98, 2017. ISSN 0167-8191. doi: https://doi.org/10.1016/j.parco.2016.11.002. URL https://www.sciencedirect.com/science/article/pii/S0167819116301326. High-End Computing for Next-Generation Scientific Discovery.

Boeing. Fuel conservation strategies:cost index explained. *Aero Magazine*, 2007. URL https://www.boeing.com/commercial/aeromagazine/articles/qtr_2_07/article_05_1.html.

Stephane Bratu and Cynthia Barnhart. Flight operations recovery: New approaches considering passenger recovery. *J. of Scheduling*, 9(3):279–298, June 2006. ISSN 1094-6136.

Jia-Ming Cao and Adib Kanafani. Real-time decision support for integration of airline flight cancellations and delays part i: mathematical formulation. *Transportation Planning and Technology*, 20(3):183–199, 1997a.

Jia-Ming Cao and Adib Kanafani. Real-time decision support for integration of airline flight cancellations and delays part ii: algorithm and computational experiments. *Transportation Planning and Technology*, 20(3):201–217, 1997b.

Jun Chen, Michal Weiszer, Giorgio Locatelli, Stefan Ravizza, Jason A Atkin, Paul Stewart, and Edmund K Burke. Toward a more realistic, cost-effective, and greener ground movement through active routing: A multiobjective shortest path approach. *IEEE Transactions on Intelligent Transportation Systems*, 17(12):3524–3540, 2016. doi: 10.1109/tits.2016.2587619.

Jacob Cheung, Alan Hally, Jaap Heijstek, Adri Marsman, and Jean-Louis Brenguier. Recommendations on trajectory selection in flight planning based on weather uncertainty. *Proc. 5th SESAR Innovation Days (SID2015), Bologna, Italy*, pages 1–8, 2015.

Lloyd Clarke, Ellis Johnson, George Nemhauser, and Zhongxi Zhu. *Annals of Operations Research*, 69:33–46, 1997. doi: 10.1023/a:1018945415148.

Jens Clausen, Allan Larsen, Jesper Larsen, and Natalia J. Rezanova. Disruption management in the airline industry-concepts, models and methods. *Comput. Oper. Res.*, 37(5):809–821, May 2010. ISSN 0305-0548.

George Dantzig. Linear programming and extensions. In *Linear programming and extensions*. Princeton university press, 2016.

F. de Lemos and J. Woodward. Calculating block time and consumed fuel for an aircraft model. *The Aeronautical Journal*, 125(1287):847–878, mar 2021. doi: 10.1017/aer.2020.137.

Luis Delgado and Xavier Prats. Simulation of airborne atfm delay and delay recovery by cruise speed reduction. 09 2011.

Luis Delgado and Xavier Prats. En route speed reduction concept for absorbing air traffic flow management delays. 49(1):214–224, jan 2012. doi: 10.2514/1.c031484.

Sophie Demassey, Christian Artigues, and Philippe Michelon. Constraint-propagation-based cutting planes: An application to the resource-constrained project scheduling problem. *INFORMS Journal on computing*, 17(1):52–65, 2005.

Muhammet Deveci and Nihan Cetin Demirel. A hybrid genetic algorithm for airline crew pairing optimization. *Economic and Social Development: Book of Proceedings*, page 118, 2016.

Juan Francisco Díaz and Javier Andrés Mena. Solving the aircraft sequencing problem using concurrent constraint programming. In *Multiparadigm Programming in Mozart/Oz*, pages 292–304. Springer Berlin Heidelberg, 2005. doi: 10.1007/978-3-540-31845-3_24.

Michelle Dunbar, Gary Froyland, and Cheng-Lung Wu. An integrated scenario-based approach for robust aircraft routing, crew pairing and re-timing. *Computers & Operations Research*, 45:68–86, may 2014. doi: 10.1016/j.cor.2013.12.003.

Niklaus Eggenberg and Matteo Salani. Challenge roadef 2009. airline disruption recovery roadef challenge 2009. *ROADEF 2009*, page 403, 2009.

Niklaus Eggenberg, Matteo Salani, and Michel Bierlaire. Constraint-specific recovery network for solving airline recovery problems. *Computers & Operations Research*, 37(6):1014–1026, jun 2010. doi: 10.1016/j.cor.2009.08.006.

Gabriele Enea, Hartmut Fricke, Mike Paglione, Jesper Bronsvoort, Australia Melbourne Airservices, Almira Australia, Ramadani, Christian Seiss, and Judith Rosenow. Fuel burn estimation modeling for atm benchmark applications perspectives from an international collaboration. 07 2017.

EUROCONTROL. User Manual for the Base of Aircraft Data (BADA) Revision 3.14, 2014. URL http://www.eurocontrol.int/sites/default/files/field{_}tabs/content/documents/sesar/user-manual-bada-3-12.pdf.

Torsten Fahle, Ulrich Junker, Stefan E Karisch, Niklas Kohl, Meinolf Sellmann, and Bo Vaaben. Constraint programming based column generation for crew assignment. *Journal of Heuristics*, 8(1):59–81, 2002.

Torsten Fahle, Rainer Feldmann, Silvia Götz, Sven Grothklags, and Burkhard Monien. The aircraft sequencing problem. In *Lecture Notes in Computer Science*, pages 152–166. Springer Berlin Heidelberg, 2003. doi: 10.1007/3-540-36477-3_11.

Marshall L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 50(12):1861–1871, 2004. ISSN 00251909, 15265501. URL http://www.jstor.org/stable/30046157.

Flightaware. Flight Track Log, 2019. URL https://uk.flightaware.com/.

Flightaware. Flight Track Log, 2022. URL https://uk.flightaware.com/live/findflight?origin=LIRF&destination=LGAV.

Antonio Franco, Damián Rivas, and Alfonso Valenzuela. Minimum-fuel cruise at constant altitude with fixed arrival time. *Journal of guidance, control, and dynamics*, 33(1):280–285, 2010.

G. Freuder and M. Wallace. Constraint technology and the commercial world [interview]. *IEEE Intelligent Systems and their Applications*, 15(1):20–23, jan 2000. doi: 10.1109/mis.2000.820324.

Sami Gabteni and Mattias Grönkvist. Combining column generation and constraint programming to solve the tail assignment problem. *Annals of Operations Research*, 171(1):61–76, 2009.

Alessandro Gardi, Roberto Sabatini, and Subramanian Ramasamy. Multi-objective optimisation of aircraft flight trajectories in the ATM and avionics context. *Progress in Aerospace Sciences*, 83:1–36, may 2016. doi: 10.1016/j.paerosci.2015.11.006.

A. M. Geoffrion. *Lagrangean relaxation for integer programming*, pages 82–114. Springer Berlin Heidelberg, Berlin, Heidelberg, 1974. ISBN 978-3-642-00740-8. doi: 10.1007/BFb0120690. URL https://doi.org/10.1007/BFb0120690.

Henrich Glaser-Opitz, Ján Labun, Kristína Budajová, and Leonard Glaser-Opitz. DESCENT TRAJECTORY MODELLING FOR THE LANDING SYSTEM PROTOTYPE. *Transport*, 35(2):133–142, apr 2020. doi: 10.3846/transport.2020.12231.

Google. Rome to Athens, 2022. URL https://www.google.com/travel/flights/search?tfs=CBwQAhopagwIAhIIL20vMDZjNjISCjIwMjItMDktMjVyCwgCEgcvbS8wbjJ6KABwAYIBCwj___________8BQAFIAZgBAg&tfu=EgYIARABGAA&hl=pt-PT&gl=uk&curr=GBP.

Karthik Gopalakrishnan, Hamsa Balakrishnan, and Richard Jordan. Clusters and communities in air traffic delay networks. *2016 American Control Conference (ACC)*, pages 3782–3788, 2016.

W Grimm, K H Well, and H J Oberle. Periodic control for minimum-fuel aircraft trajectories. *Journal of Guidance, Control, and Dynamics*, 9(2):169–174, 1986.

Mattias Grönkvist. Accelerating column generation for aircraft scheduling using constraint propagation. *Computers & Operations Research*, 33(10):2918–2934, 2006.

T. Grosche. Computational intelligence in integrated airline scheduling. 2009.

Mattias Grönkvist. Accelerating column generation for aircraft scheduling using constraint propagation. *Computers & Operations Research*, 33(10):2918–2934, oct 2006. doi: 10.1016/j.cor.2005.01.017.

Stefano Gualandi and Federico Malucelli. Constraint programming-based column generation. *4OR*, 7(2): 113–137, 2009.

Daniel Guimarans, Pol Arias, and Miguel Mujica Mota. A new methodology to solve the stochastic aircraft recovery problem using optimization and simulation. 07 2013.

Daniel Guimarans, Pol Arias, and Miguel Mujica Mota. *Large Neighbourhood Search and Simulation for Disruption Management in the Airline Industry*, pages 169–201. 04 2015. ISBN 978-3-319-15032-1. doi: 10.1007/978-3-319-15033-8_6.

Patrick Hagelauer and Felix Mora-Camino. A soft dynamic programming approach for on-line aircraft 4D-trajectory optimization. *European Journal of Operational Research*, 107(1):87–95, 1998.

Christopher A. Hane, Cynthia Barnhart, Ellis L. Johnson, Roy E. Marsten, George L. Nemhauser, and Gabriele Sigismondi. The fleet assignment problem: Solving a large-scale integer program. *Mathematical Programming*, 70(1-3):211–232, oct 1995. doi: 10.1007/bf01585938.

Sander Hartjes, Marco E. G. van Hellenberg Hubar, and Hendrikus G. Visser. Multiple-phase trajectory optimization for formation flight in civil aviation. *CEAS Aeronautical Journal*, 10(2):453–462, sep 2018. doi: 10.1007/s13272-018-0329-9.

Michael Held and Richard M Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162, 1970.

Michael Held and Richard M Karp. The traveling-salesman problem and minimum spanning trees: Part ii. *Mathematical programming*, 1(1):6–25, 1971.

J. N. Hooker and W.-J. van Hoeve. Constraint programming and operations research. *Constraints*, 23(2): 172–195, dec 2017. doi: 10.1007/s10601-017-9280-3.

John N Hooker, Greger Ottosson, Erlendur S Thorsteinsson, and Hak-Jin Kim. On integrating constraint propagation and linear programming for combinatorial optimization. In *AAAI/IAAI*, pages 136–141. Citeseer, 1999.

Dan Howell and Rob Dean. Have Descents really become more Efficient? In *USA/Europe Air Traffic Management R&D Seminar, Seattle, Washington, USA*, 2017.

Yuzhen Hu, Xu Baoguang, Mingang Gao, and Hong Chi. The study on the aircraft recovery with consideration of passenger transiting. 11 2011.

Yuzhen Hu, Hong Liao, Song Zhang, and Yan Song. Multiple objective solution approaches for aircraft rerouting under the disruption of multi-aircraft. *Expert Systems with Applications*, 83:283–299, oct 2017. doi: 10.1016/j.eswa.2017.04.031.

IATA. Airline industry economic performance - december 2019 - report. 2019. URL https://www.iata.org/en/iata-repository/publications/economic-reports/airline-industry-economic-performance---november-2020---report/.

IATA. Airline industry economic performance - november 2020 - report. 2020. URL https://www.iata.org/en/iata-repository/publications/economic-reports/airline-industry-economic-performance---december-2019---report/.

IBM. What is constraint programming?, 2022. URL https://www.ibm.com/docs/en/icos/20.1.0?topic=programming-what-is-constraint.

Niloofar Jafari and Seyed Hessameddin Zegordi. Simultaneous recovery model for aircraft and passengers. *Journal of the Franklin Institute*, 348(7):1638 – 1655, 2011. ISSN 0016-0032.

Joxan Jaffar and J-L Lassez. Constraint logic programming. In *Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 111–119, 1987.

Joxan Jaffar and Michael J. Maher. Constraint logic programming: a survey. *The Journal of Logic Programming*, 19-20:503–581, 1994. ISSN 0743-1066. doi: https://doi.org/10.1016/0743-1066(94)90033-7. URL https://www.sciencedirect.com/science/article/pii/0743106694900337. Special Issue: Ten Years of Logic Programming.

Ahmad I. Z. Jarrah, Gang Yu, Nirup Krishnamurthy, and Ananda Rakshit. A decision support framework for airline flight cancellations and delays. *Transportation Science*, 27(3):266–280, 1993. ISSN 00411655, 15265447.

Ulrich Junker, Stefan E. Karisch, Niklas Kohl, Bo Vaaben, Torsten Fahle, and Meinolf Sellmann. A framework for constraint programming based column generation. In *Principles and Practice of Constraint Programming – CP'99*, pages 261–274. Springer Berlin Heidelberg, 1999. doi: 10.1007/978-3-540-48085-3_19.

Harshad Khadilkar and Hamsa Balakrishnan. Estimation of aircraft taxi fuel burn using flight data recorder archives. *Transportation Research Part D: Transport and Environment*, 17(7):532–537, 2012.

Oumaima Khaled, Michel Minoux, Vincent Mousseau, Stéphane Michel, and Xavier Ceugniet. A multi-criteria repair/recovery framework for the tail assignment problem in airlines. *Journal of Air Transport Management*, 68:137–151, may 2018. doi: 10.1016/j.jairtraman.2017.10.002.

Dave Knorr, Xing Chen, Marc Rose, John Gulding, Philippe Enaud, and Holger Hegendoerfer. Estimating ATM efficiency pools in the descent phase of flight. In *9th USA/Europe air traffic management research and development seminar (ATM2011)*, volume 5, pages 16–23, 2011.

Niklas Kohl, Allan Larsen, Jesper Larsen, Alex Ross, and Sergey Tiourine. Airline disruption management—perspectives, experiences and outlook. *Journal of Air Transport Management*, 13(3):149 – 162, 2007. ISSN 0969-6997.

Meilong Le, Jinmin Gao, and Chenxu Zhan. Solving the airline recovery problem based on vehicle routing problem with time window modeling and genetic algorithm. In *2013 Ninth International Conference on Natural Computation (ICNC)*. IEEE, jul 2013. doi: 10.1109/icnc.2013.6818089.

Jane Lee, Lavanya Marla, and Alexandre Jacquillat. Dynamic disruption management in airline networks under airport operating uncertainty. *Transportation Science*, 54(4):973–997, 2020.

J. Leung. Handbook of scheduling: Algorithms, models, and performance analysis. 2004.

Zhe Liang, Yuan Feng, Xiaoning Zhang, Tao Wu, and Wanpracha Art Chaovalitwongse. Robust weekly aircraft maintenance routing problem and the extension to the tail assignment problem. *Transportation Research Part B: Methodological*, 78:238–259, aug 2015. doi: 10.1016/j.trb.2015.03.013.

Martin Lindner, Judith Rosenow, Thomas Zeh, and Hartmut Fricke. In-flight aircraft trajectory optimization within corridors defined by ensemble weather forecasts. *Aerospace*, 7(10):144, 2020.

Tung-Kuan Liu, Chiu-Hung Chen, and Jyh-Horng Chou. Optimization of short-haul aircraft schedule recovery problems using a hybrid multiobjective genetic algorithm. *Expert Systems with Applications*, 37(3):2307–2315, mar 2010. doi: 10.1016/j.eswa.2009.07.068.

M. Løve, K. R. Sørensen, J. Larsen, and J. Clausen. Using heuristics to solve the dedicated aircraft recovery problem. 2001.

Raïd Mansi, Saïd Hanafi, Christophe Wilbaut, and François Clautiaux. Disruptions in the airline industry: math-heuristics for re-assigning aircraft and passengers simultaneously. *European Journal of Industrial Engineering 10*, 6(6):690–712, 2012.

Lavanya Marla, Bo Vaaben, and Cynthia Barnhart. Integrated disruption management and flight planning to trade off delays and fuel burn. *Transportation Science*, 51(1):88–111, February 2017. ISSN 1526-5447. doi: 10.1287/trsc.2015.0609. URL https://doi.org/10.1287/trsc.2015.0609.

Paul Melby and Ralf Mayer. Benefit potential of continuous climb and descent operations. In *The 26th Congress of ICAS and 8th AIAA ATIO*, page 8920, 2008.

Adeline Montlaur, Luis Delgado, and César Trapote-Barreira. Analytical models for co2 emissions and travel time for short-to-medium-haul flights considering available seats. *Sustainability*, 13(18), 2021. ISSN 2071-1050. doi: 10.3390/su131810401. URL https://www.mdpi.com/2071-1050/13/18/10401.

Alejandro Murrieta-Mendoza, Hugo Ruiz, and Ruxandra Mihaela Botez. Vertical reference flight trajectory optimization with the particle swarm optimisation. In *Modelling, Identification and Control*. ACTAPRESS, 2017. doi: 10.2316/p.2017.848-032.

Frank Neuman and Eliezer Kreindler. Minimum-fuel, three-dimensional flight paths for jet transports. *Journal of Guidance, Control, and Dynamics*, 8(5):650–657, 1985.

Ridvan Oruc and Tolga Baklacioglu. Modelling of fuel flow-rate of commercial aircraft for the climbing flight using cuckoo search algorithm. *Aircraft Engineering and Aerospace Technology*, 92(3):495–501, feb 2020. doi: 10.1108/aeat-10-2019-0202.

Ioanna Pagoni and Voula Psaraki-Kalouptsidi. Calculation of aircraft fuel consumption and co2 emissions based on path profile estimation by clustering and registration. *Transportation Research Part D: Transport and Environment*, 54:172 – 190, 2017. ISSN 1361-9209. doi: https://doi.org/10.1016/j.trd.2017.05.006. URL http://www.sciencedirect.com/science/article/pii/S1361920916305533.

Roberto S. Félix Patrón, Yolène Berrou, and Ruxandra M. Botez. New methods of optimization of the flight profiles for performance database-modeled aircraft. 229:1853–1867, 2015. ISSN 0954-4100. doi: 10.1177/0954410014561772.

Małgorzata Pawlak, Michał Kuźniar, and Andrzej R. Majka. Determination of the flight trajectory in terms of emission and fuel consumption minimization. page 095441002110122, apr 2021. doi: 10.1177/09544100211012265.

Maximilian Pohl, Christian Artigues, and Rainer Kolisch. Solving the time-discrete winter runway scheduling problem: A column generation and constraint programming approach. *European Journal of Operational Research*, aug 2021. doi: 10.1016/j.ejor.2021.08.028.

Gao Qiang, Tang Xiao-wei, and Zhu Jin-fu. Research on greedy simulated annealing algorithm for irregular flight schedule recovery model. In *2009 IEEE International Conference on Grey Systems and Intelligent Services (GSIS 2009)*. IEEE, nov 2009. doi: 10.1109/gsis.2009.5408145.

Stefan Ravizza, Jason A D Atkin, Marloes H Maathuis, and Edmund K Burke. A combined statistical approach and ground movement model for improving taxi time estimations at airports. *Journal of the Operational Research Society*, 64(9):1347–1360, 2013.

Stefan Ravizza, Jun Chen, Jason A D Atkin, Paul Stewart, and Edmund K Burke. Aircraft taxi time prediction: comparisons and insights. *Applied Soft Computing*, 14:397–406, 2014.

John Robinson III and Maryam Kamgarpour. Benefits of continuous descent operations in high-density terminal airspace considering scheduling constraints. In *10th AIAA aviation technology, integration, and operations (ATIO) conference*, page 9115, 2010.

Jay M. Rosenberger, Ellis L. Johnson, and George L. Nemhauser. Rerouting aircraft for airline recovery. *Transportation Science*, 37(4):408–421, 2003.

Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006a. ISBN 0444527265.

Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006b.

Juan-José Salazar-González. Approaches to solve the fleet-assignment, aircraft-routing, crew-pairing and crew-rostering problems of a regional carrier. *Omega*, 43:71–82, mar 2014. doi: 10.1016/j.omega.2013.06.006.

Tanja Šarčević, Ana Paula Rocha, and Antonio J. M. Castro. Artificial bee colony algorithm for solving the flight disruption problem. In *Highlights of Practical Applications of Agents, Multi-Agent Systems, and Complexity: The PAAMS Collection*, pages 72–81. Springer International Publishing, 2018. doi: 10.1007/978-3-319-94779-2_7.

Peter J. Schwartz. Constraint optimization literature review. 2015.

Karine Sinclair, Jean-François Cordeau, and Gilbert Laporte. Improvements to a large neighborhood search heuristic for an integrated aircraft and passenger recovery problem. *European Journal of Operational Research*, 233(1):234 – 245, 2014. ISSN 0377-2217.

Henrique Sousa, Ricardo Teixeira, Henrique Lopes Cardoso, and Eugénio Oliveira. Airline disruption management - dynamic aircraft scheduling with ant colony optimization. In *Proceedings of the International Conference on Agents and Artificial Intelligence*. SCITEPRESS - Science and and Technology Publications, 2015. doi: 10.5220/0005205303980405.

Statista. Share of total minutes flights were delayed in the united states from 2004 to 2019, by cause. Technical report, 2021. URL https://www.statista.com/statistics/481333/leading-causes-of-flight-delay-in-the-us/.

Fang Sun, Hong Liu, and Yu Zhang. Integrated aircraft and passenger recovery with enhancements in modeling, solution algorithm, and intermodalism. *IEEE Transactions on Intelligent Transportation Systems*, PP:1–16, 07 2021. doi: 10.1109/TITS.2021.3090329.

Dušan Teodorović and Slobodan Guberinić. Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operational Research*, 15(2):178 – 182, 1984. ISSN 0377-2217.

Benjamin G. Thengvall, Jonathan F Bard, and Gang Yu. Balancing user preferences for aircraft schedule recovery during irregular operations. *IIE Transactions*, 32(3):181–193, 2000.

Benjamin G Thengvall, Gang Yu, and Jonathan F Bard. Multiple fleet aircraft schedule recovery following hub closures. *Transportation Research Part A: Policy and Practice*, 35(4):289 – 308, 2001. ISSN 0965-8564.

Enis T Turgut and Marc A Rosen. Relationship between fuel consumption and altitude for commercial aircraft during descent: preliminary assessment with a genetic algorithm. *Aerospace Science and Technology*, 17(1):65–73, 2012.

Pascal Van Hentenryck. *Constraint satisfaction in logic programming*. MIT press, 1989.

Marc B Vilain and Henry A Kautz. Constraint propagation algorithms for temporal reasoning. In *Aaai*, volume 86, pages 377–382, 1986.

J. Vink, B.F. Santos, W.J.C. Verhagen, I. Medeiros, and R. Filho. Dynamic aircraft recovery problem - an operational decision support framework. *Computers & Operations Research*, 117:104892, may 2020. doi: 10.1016/j.cor.2020.104892.

Hans-Wieger M. Vos, Bruno F. Santos, and Thomas Omondi. Aircraft schedule recovery problem – a dynamic modeling framework for daily operations. *Transportation Research Procedia*, 10:931–940, 2015. doi: 10.1016/j.trpro.2015.09.047.

Mark Wallace. Constraint programming - the paradigm to watch. 2007.

Xin Wen, Hoi-Lam Ma, Sai-Ho Chung, and Waqar Ahmed Khan. Robust airline crew scheduling with flight flying time variability. *Transportation Research Part E: Logistics and Transportation Review*, 144:102132, dec 2020. doi: 10.1016/j.tre.2020.102132.

Navinda Wickramasinghe. Correlation between flight time and fuel consumption in airliner flight plan with trajectory optimization. 01 2015.

Congcong Wu and Meilong Le. A new approach to solve aircraft recovery problem. 2012.

Zhao Xiuli and Guo Yanchi. Study on GRAPS-ACO algorithm for irregular flight rescheduling. In *2012 International Conference on Computer Science and Service System*. IEEE, aug 2012. doi: 10.1109/csss.2012.74.

Haiwen Xu and Songchen Han. Weighted time-band approximation model for flight operations recovery considering simplex group cycle approaches in china. *Mathematical Problems in Engineering*, 2016: 1–17, 2016. doi: 10.1155/2016/3201490.

Haiwen Xu, Songchen Han, Yong Zhang, and Jianguang Li. The time-band approximation model on flight operations recovery model considering random flight flying time in china. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, oct 2015. doi: 10.1109/smc.2015.131.

Chiwei Yan and Jerry Kung. Robust aircraft routing. *Transportation Science*, 52(1):118–133, jan 2018. doi: 10.1287/trsc.2015.0657.

Shangyao Yan and Chung-Gee Lin. Airline scheduling for the temporary closure of airports. *Transportation Science*, 31(1):72–82, 1997.

Shangyao Yan and Yu ping Tu. Multifleet routing and multistop flight scheduling for schedule perturbation. *European Journal of Operational Research*, 103(1):155–169, nov 1997. doi: 10.1016/s0377-2217(96) 00260-3.

Shangyao Yan and Dah-Hwei Yang. A decision support framework for handling schedule perturbation. *Transportation Research Part B: Methodological*, 30(6):405–419, dec 1996. doi: 10.1016/0191-2615(96)00013-6.

Tallys H Yunes, Arnaldo V Moura, and Cid C De Souza. Solving very large crew scheduling problems to optimality. In *Proceedings of the 2000 ACM symposium on Applied computing-Volume 1*, pages 446–451, 2000.

Cheng Zhang. Two-stage heuristic algorithm for aircraft recovery problem. *Discrete Dynamics in Nature and Society*, 2017:1–12, 2017. doi: 10.1155/2017/9575719.

Yu Zhang and Mark Hansen. Real-time intermodal substitution: Strategy for airline recovery from schedule perturbation and for mitigation of airport congestion. *Transportation Research Record*, 2052: 90–99, 12 2008. doi: 10.3141/2052-11.

Tong Zhao and Xiong Chen. A weight-table based heuristic algorithm for aircraft recovery problem. In *2018 37th Chinese Control Conference (CCC)*. IEEE, jul 2018. doi: 10.23919/chicc.2018.8484150.

Bo Zhu, Jin fu Zhu, and Qiang Gao. A stochastic programming approach on aircraft recovery problem. *Mathematical Problems in Engineering*, 2015:1–9, 2015. doi: 10.1155/2015/680609.

Appendix A

BTF Extended Results and Data Sets

A.1 Flight data for the BTF model, ROADEF and FlightAware ™

Table A.1 Block time comparison between the BTF model, ROADEF and Flightaware

Origin air-port	Aircraft model	Destination airport	Ground dis-tance [Km]	BTF model block time [min]	BTF model per-centile	BTF model RMSE [min]	ROADEF block time [min]	ROADEF per-centile	ROADEF RMSE [min]	Sample size	Sample average flight time [min]
FCO	A320	CDG	1101	116	94.7	16.6	130	100.0	29.8	19	101.0
BIQ	A319	ORY	658	70	90.0	6.7	80	100.0	15.3	10	65.0
ORY	A320	MRS	627	70	97.7	14.5	75	97.7	19.3	44	56.0
CDG	A320	VIE	1036	106	100.0	18.6	130	100.0	42.3	55	88.0
ORY	A320	MPL	585	66	100.0	11.9	80	100.0	25.8	9	54.0
TLS	A321	ORY	572	64	90.0	7.3	80	100.0	22.3	75	58.0
MPL	A319	CDG	614	71	60.0	3.7	90	100.0	20.1	5	70.0
BCN	A320	CDG	859	98	100.0	11.4	115	100.0	28.0	60	87.0
ORY	A320	TLS	572	66	100.0	13.5	70	100.0	17.4	62	52.0
LIS	A321	CDG	1471	137	100.0	15.9	150	100.0	28.5	27	121.0
NCE	A321	ORY	675	74	92.4	7.1	85	98.3	17.1	59	68.0
BOD	A321	CDG	527	65	100.0	8.0	85	100.0	28.0	1	57.0
FCO	A319	CDG	1101	116	100.0	12.5	130	100.0	26.2	6	104.0
ORY	A321	MRS	627	70	77.8	14.3	75	86.1	18.1	18	60.0
MRS	A319	ORY	627	70	92.1	14.5	80	97.4	20.0	38	65.0
NCE	A319	ORY	675	74	93.0	7.5	85	95.3	16.5	43	69.0
ORY	A318	MRS	627	70	100.0	14.2	75	100.0	19.1	18	56.0
ORY	A319	MRS	627	70	95.2	18.9	75	97.6	22.4	42	57.0
ORY	A320	NCE	675	73	98.3	10.1	80	100.0	16.7	118	63.0

Table A.1 Block time comparison between the BTF model, ROADEF and Flightaware

Origin air-port	Aircraft model	Destination airport	Ground dis-tance [Km]	BTF model block time [min]	BTF model per-centile	BTF model RMSE [min]	ROADEF block time [min]	ROADEF per-centile	ROADEF RMSE [min]	Sample size	Sample average flight time [min]
TLS	A319	ORY	572	64	80.2	5.7	80	100.0	20.2	43	60.0
NCE	A320	ORY	675	74	88.5	10.4	85	97.5	18.8	122	68.0
MRS	A318	ORY	627	71	100.0	7.2	80	100.0	15.7	20	64.0
TLS	A318	ORY	572	65	85.0	8.0	80	100.0	21.1	20	59.0
CDG	A320	BES	515	66	100.0	15.9	75	100.0	24.8	20	50.0
PUF	A319	ORY	632	67	66.7	3.0	80	100.0	14.9	3	65.0
MUC	A319	CDG	682	81	96.9	6.9	105	100.0	30.3	32	74.0
CDG	A320	MAD	1065	111	98.4	14.9	125	100.0	28.4	63	97.0
CDG	A319	FCO	1101	114	56.7	47.7	130	66.7	46.6	30	125.0
MXP	A320	CDG	598	75	100.0	12.5	95	100.0	32.0	23	63.0
CDG	A320	BCN	859	93	100.0	14.0	105	100.0	25.8	5	79.0
ORY	A321	TLS	572	66	100.0	13.0	70	100.0	16.9	75	53.0
VIE	A319	CDG	1036	103	87.5	6.6	135	100.0	36.8	16	98.0
CDG	A319	MXP	598	75	100.0	11.8	105	100.0	41.5	9	63.0
CDG	A320	MXP	598	75	100.0	14.0	105	100.0	43.5	24	61.0
BCN	A320	LYS	549	71	100.0	11.8	80	100.0	20.6	27	59.0
FCO	A319	CTA	539	70	100.0	20.5	75	100.0	25.5	14	49.0
ORY	A319	TLS	572	66	100.0	12.6	70	100.0	16.6	45	53.0
DUS	A319	LYS	631	72	100.0	34.9	90	100.0	52.1	31	39.0
BES	A320	CDG	515	65	100.0	11.8	80	100.0	26.3	20	54.0
ORY	A319	NCE	675	72	91.4	8.1	80	100.0	15.5	35	65.0

Table A.1 Block time comparison between the BTF model, ROADEF and Flightaware

Origin airport	Aircraft model	Destination airport	Ground distance [Km]	BTF model block time [min]	BTF model percentile	BTF model RMSE [min]	ROADEF block time [min]	ROADEF percentile	ROADEF RMSE [min]	Sample size	Sample average flight time [min]
ORY	A320	BIQ	658	70	100.0	11.8	75	100.0	16.8	20	58.0
LYS	A320	FCO	722	81	100.0	18.7	90	100.0	27.7	9	62.0
CDG	A320	MPL	614	74	100.0	16.8	85	100.0	27.7	29	57.0
MPL	A319	ORY	585	65	86.1	4.6	85	100.0	23.2	18	62.0
ORY	A318	TLS	572	67	100.0	13.4	70	100.0	16.4	19	53.0
MAD	A318	CDG	1065	111	100.0	15.1	125	100.0	29.1	2	96.0
ORY	A319	PUF	632	70	100.0	13.0	75	100.0	18.0	3	57.0
MRS	A321	ORY	627	70	77.8	7.4	80	94.4	14.8	18	66.0
MPL	A320	CDG	614	72	87.5	6.5	90	100.0	23.0	28	67.0
BES	A319	ORY	501	59	100.0	4.5	75	100.0	20.5	2	54.0
CDG	A320	NCE	695	80	100.0	12.5	95	100.0	27.2	69	68.0
BOD	A319	CDG	527	65	100.0	11.7	85	100.0	31.4	18	53.0
NCE	A318	ORY	675	75	91.3	6.9	85	100.0	16.0	23	69.0
CDG	A321	BOD	527	69	100.0	18.0	80	100.0	29.0	1	51.0
NCE	A320	CDG	695	80	97.1	9.4	95	100.0	23.8	68	71.0
FCO	A321	CDG	1101	116	98.1	13.4	130	100.0	26.7	53	104.0
AJA	A321	ORY	907	87	62.5	6.3	100	100.0	14.9	8	86.0
BCN	A321	CDG	859	98	100.0	16.2	115	100.0	32.3	53	83.0
BIQ	A320	ORY	658	70	100.0	8.3	80	100.0	17.7	19	62.0
CDG	A318	MPL	614	74	100.0	17.9	85	100.0	28.9	13	56.0

A.2 Fuel flow for the climb phase for the B737-800

Table A.2 BTF data for fuel flow vs. altitude for the climb phase for the B737-800

Fuel flow [100,000 Kg/h]	Altitude [1,000,000 ft]
0.010	0.06666
0.012	0.06420
0.014	0.06174
0.016	0.05928
0.018	0.05676
0.020	0.05430
0.022	0.05178
0.024	0.04926
0.026	0.04674
0.028	0.04422
0.029	0.04290
0.031	0.03996

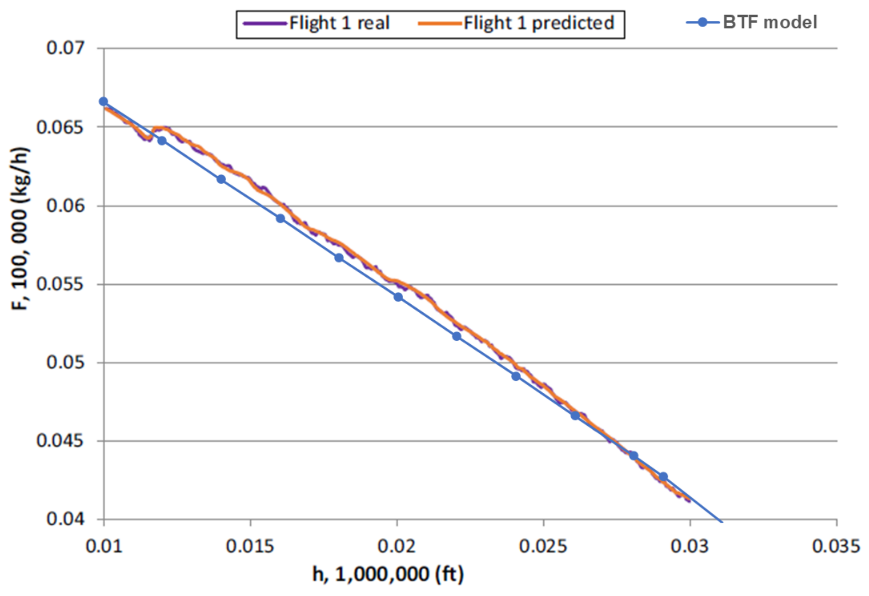


Fig. A.1 Fuel flow vs. altitude for flight 1

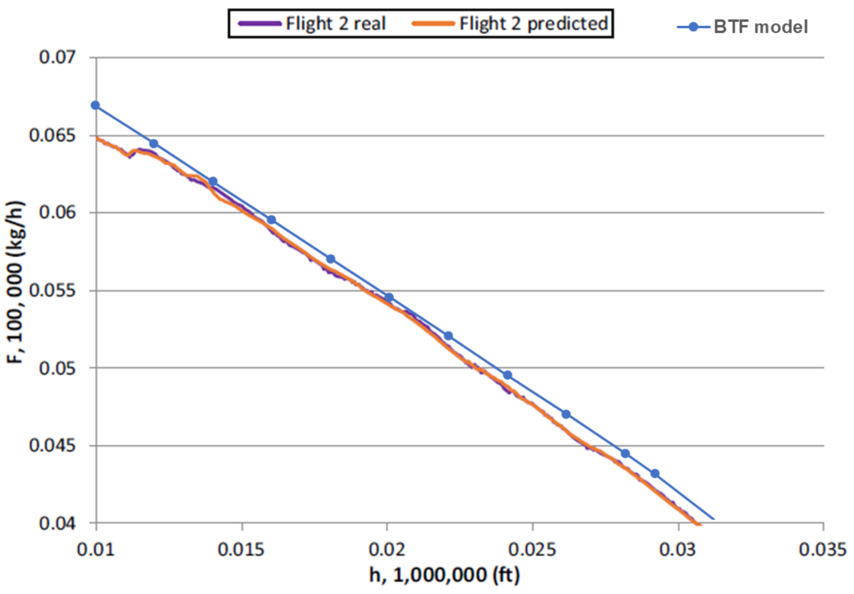


Fig. A.2 Fuel flow vs. altitude for flight 2

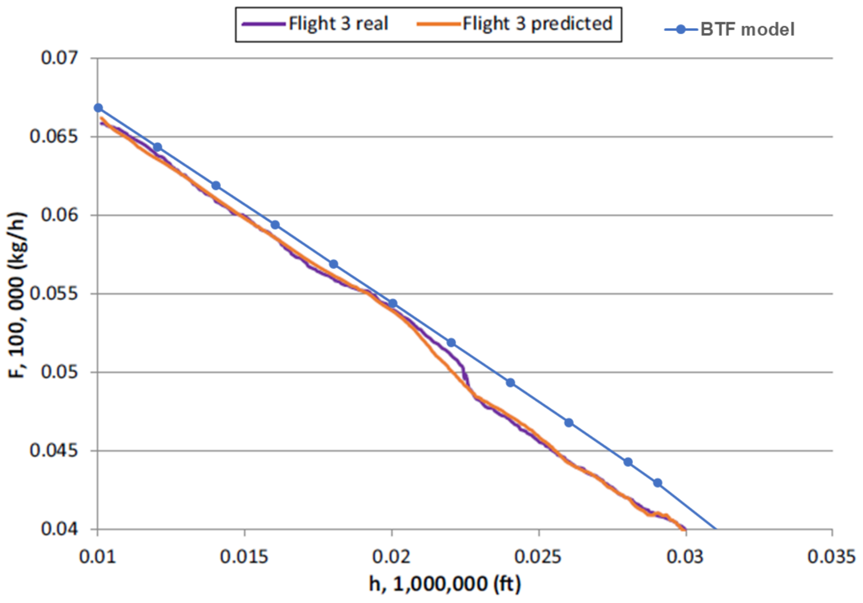


Fig. A.3 Fuel flow vs. altitude for flight 3

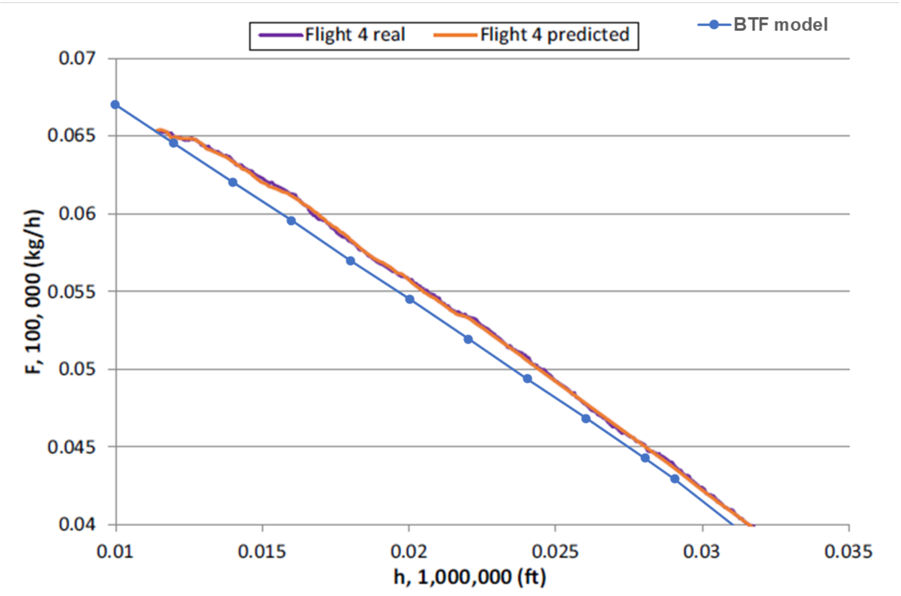


Fig. A.4 Fuel flow vs. altitude for flight 4

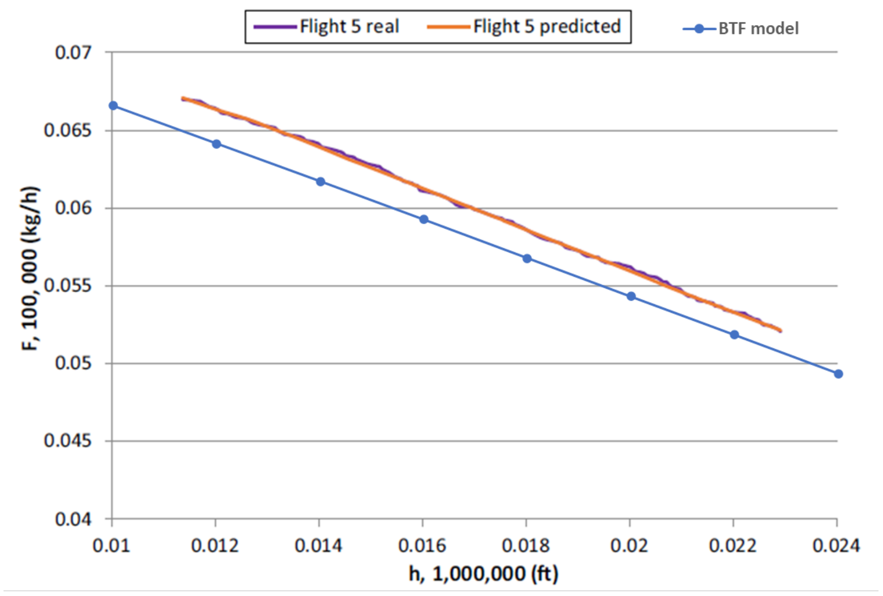
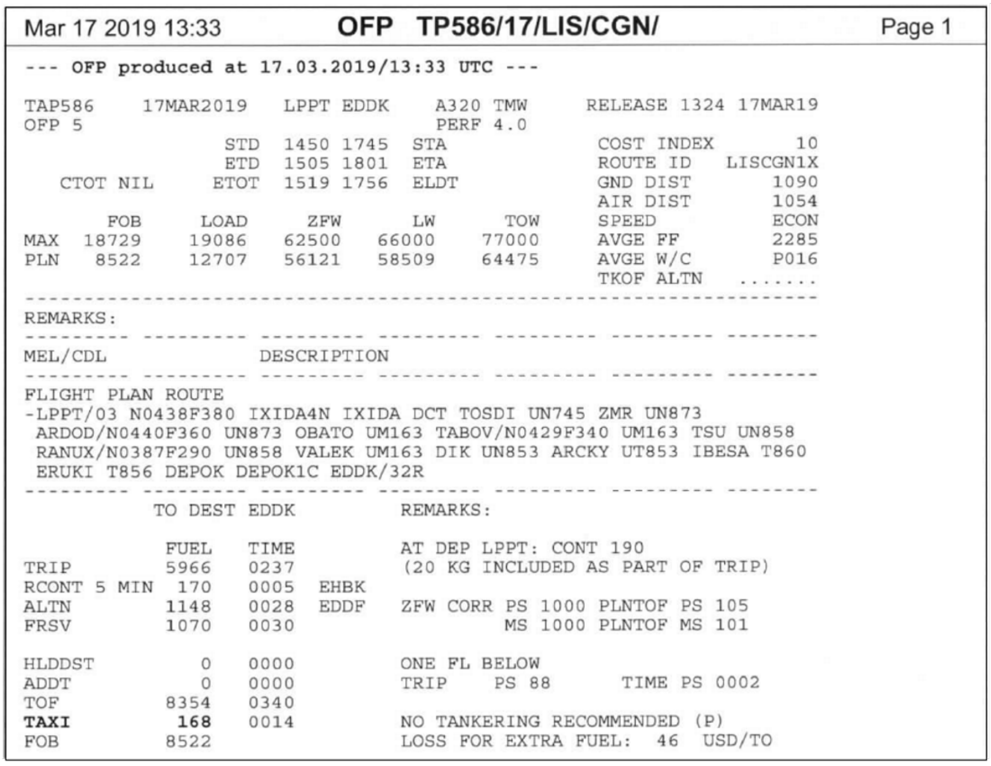


Fig. A.5 Fuel flow vs. altitude for flight 5

A.3 Lido™ flight plan extracts



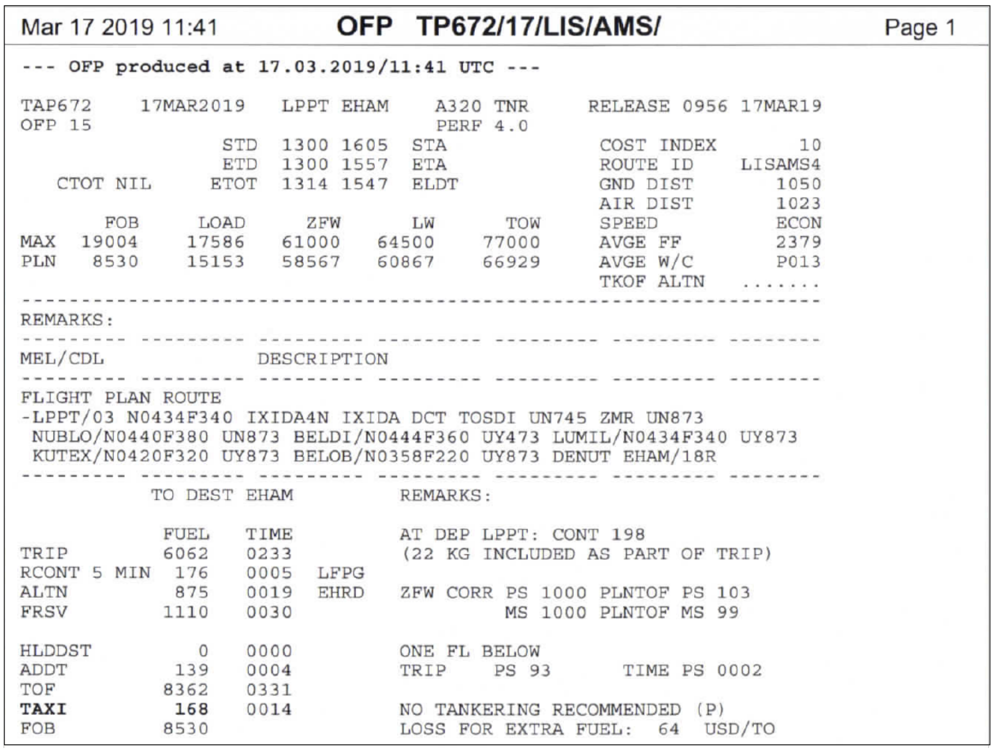
```
Mar 17 2019 13:33            OFP  TP586/17/LIS/CGN/                  Page 1
--- OFP produced at 17.03.2019/13:33 UTC ---

TAP586     17MAR2019    LPPT EDDK    A320 TMW     RELEASE 1324 17MAR19
OFP 5                                PERF 4.0
                 STD  1450 1745  STA               COST INDEX       10
                 ETD  1505 1801  ETA               ROUTE ID   LISCGN1X
   CTOT NIL     ETOT  1519 1756  ELDT              GND DIST       1090
                                                   AIR DIST       1054
       FOB      LOAD      ZFW       LW       TOW   SPEED          ECON
MAX  18729     19086    62500    66000     77000   AVGE FF        2285
PLN   8522     12707    56121    58509     64475   AVGE W/C       P016
                                                   TKOF ALTN   .......
--------------------------------------------------------------------
REMARKS:
--------- --------- --------- --------- --------- --------- --------
MEL/CDL            DESCRIPTION
--------- --------- --------- --------- --------- --------- --------
FLIGHT PLAN ROUTE
-LPPT/03 N0438F380 IXIDA4N IXIDA DCT TOSDI UN745 ZMR UN873
 ARDOD/N0440F360 UN873 OBATO UM163 TABOV/N0429F340 UM163 TSU UN858
 RANUX/N0387F290 UN858 VALEK UM163 DIK UN853 ARCKY UT853 IBESA T860
 ERUKI T856 DEPOK DEPOK1C EDDK/32R
--------- --------- --------- --------- --------- --------- --------
           TO DEST EDDK           REMARKS:

            FUEL   TIME           AT DEP LPPT: CONT 190
TRIP        5966   0237           (20 KG INCLUDED AS PART OF TRIP)
RCONT 5 MIN  170   0005   EHBK
ALTN        1148   0028   EDDF    ZFW CORR PS 1000 PLNTOF PS 105
FRSV        1070   0030                    MS 1000 PLNTOF MS 101

HLDDST         0   0000           ONE FL BELOW
ADDT           0   0000           TRIP     PS 88      TIME PS 0002
TOF         8354   0340
TAXI         168   0014           NO TANKERING RECOMMENDED (P)
FOB         8522                  LOSS FOR EXTRA FUEL:  46  USD/TO
```

Fig. A.6 Lido flight plan TP586 for the Lisbon, Cologne flight



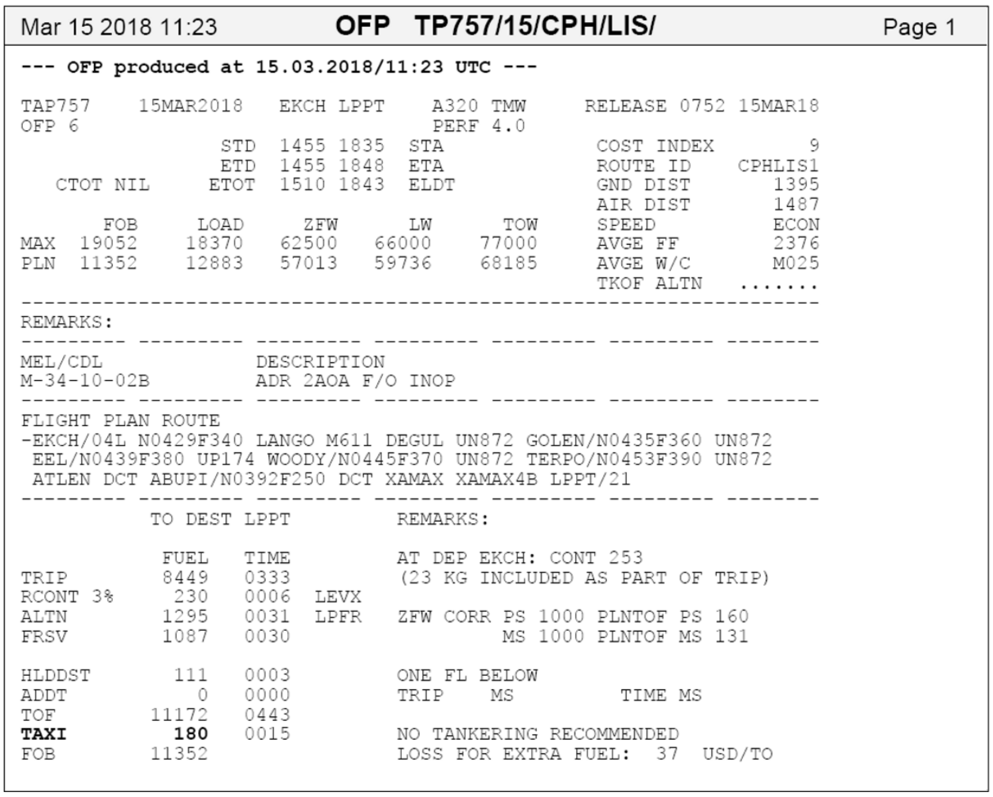
```
Mar 17 2019 11:41            OFP  TP672/17/LIS/AMS/                  Page 1
--- OFP produced at 17.03.2019/11:41 UTC ---

TAP672     17MAR2019    LPPT EHAM    A320 TNR     RELEASE 0956 17MAR19
OFP 15                               PERF 4.0
                 STD  1300 1605  STA               COST INDEX       10
                 ETD  1300 1557  ETA               ROUTE ID    LISAMS4
   CTOT NIL     ETOT  1314 1547  ELDT              GND DIST       1050
                                                   AIR DIST       1023
       FOB      LOAD      ZFW       LW       TOW   SPEED          ECON
MAX  19004     17586    61000    64500     77000   AVGE FF        2379
PLN   8530     15153    58567    60867     66929   AVGE W/C       P013
                                                   TKOF ALTN   .......
--------------------------------------------------------------------
REMARKS:
--------- --------- --------- --------- --------- --------- --------
MEL/CDL            DESCRIPTION
--------- --------- --------- --------- --------- --------- --------
FLIGHT PLAN ROUTE
-LPPT/03 N0434F340 IXIDA4N IXIDA DCT TOSDI UN745 ZMR UN873
 NUBLO/N0440F380 UN873 BELDI/N0444F360 UY473 LUMIL/N0434F340 UY873
 KUTEX/N0420F320 UY873 BELOB/N0358F220 UY873 DENUT EHAM/18R
--------- --------- --------- --------- --------- --------- --------
           TO DEST EHAM           REMARKS:

            FUEL   TIME           AT DEP LPPT: CONT 198
TRIP        6062   0233           (22 KG INCLUDED AS PART OF TRIP)
RCONT 5 MIN  176   0005   LFPG
ALTN         875   0019   EHRD    ZFW CORR PS 1000 PLNTOF PS 103
FRSV        1110   0030                    MS 1000 PLNTOF MS 99

HLDDST         0   0000           ONE FL BELOW
ADDT         139   0004           TRIP     PS 93      TIME PS 0002
TOF         8362   0331
TAXI         168   0014           NO TANKERING RECOMMENDED (P)
FOB         8530                  LOSS FOR EXTRA FUEL:  64  USD/TO
```

Fig. A.7 Lido flight plan TP672 for the Lisbon, Amsterdam flight



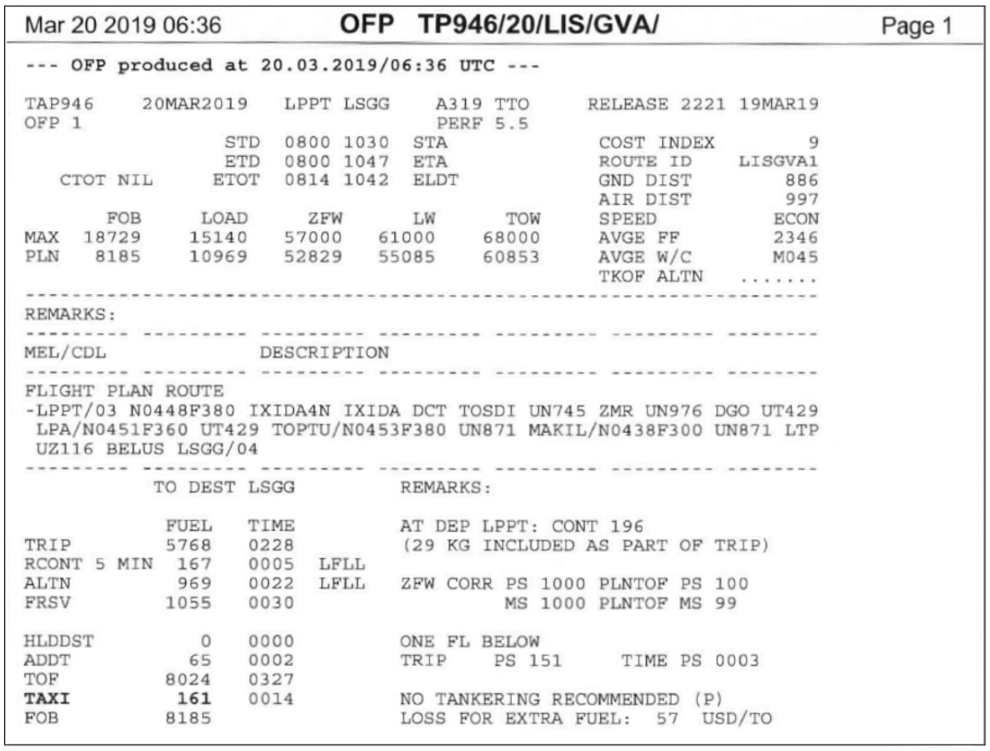
```
Mar 15 2018 11:23          OFP  TP757/15/CPH/LIS/              Page 1
--- OFP produced at 15.03.2018/11:23 UTC ---

TAP757    15MAR2018   EKCH LPPT    A320 TMW     RELEASE 0752 15MAR18
OFP 6                              PERF 4.0
                 STD  1455 1835  STA             COST INDEX        9
                 ETD  1455 1848  ETA             ROUTE ID    CPHLIS1
   CTOT NIL     ETOT  1510 1843  ELDT            GND DIST       1395
                                                 AIR DIST       1487
        FOB      LOAD      ZFW       LW      TOW  SPEED          ECON
MAX  19052    18370    62500    66000    77000   AVGE FF        2376
PLN  11352    12883    57013    59736    68185   AVGE W/C       M025
                                                 TKOF ALTN   .......
--------------------------------------------------------------------
REMARKS:
--------- --------- --------- --------- --------- --------- --------
MEL/CDL             DESCRIPTION
M-34-10-02B         ADR 2AOA F/O INOP
--------- --------- --------- --------- --------- --------- --------
FLIGHT PLAN ROUTE
-EKCH/04L N0429F340 LANGO M611 DEGUL UN872 GOLEN/N0435F360 UN872
 EEL/N0439F380 UP174 WOODY/N0445F370 UN872 TERPO/N0453F390 UN872
 ATLEN DCT ABUPI/N0392F250 DCT XAMAX XAMAX4B LPPT/21
--------- --------- --------- --------- --------- --------- --------
           TO DEST LPPT         REMARKS:

            FUEL   TIME         AT DEP EKCH: CONT 253
TRIP        8449   0333         (23 KG INCLUDED AS PART OF TRIP)
RCONT 3%     230   0006  LEVX
ALTN        1295   0031  LPFR   ZFW CORR PS 1000 PLNTOF PS 160
FRSV        1087   0030                  MS 1000 PLNTOF MS 131

HLDDST       111   0003         ONE FL BELOW
ADDT           0   0000         TRIP    MS        TIME MS
TOF        11172   0443
TAXI         180   0015         NO TANKERING RECOMMENDED
FOB        11352                LOSS FOR EXTRA FUEL:  37  USD/TO
```

Fig. A.8 Lido flight plan TP757 for the Copenhagen, Lisbon flight



```
Mar 20 2019 06:36          OFP  TP946/20/LIS/GVA/              Page 1
--- OFP produced at 20.03.2019/06:36 UTC ---

TAP946    20MAR2019   LPPT LSGG    A319 TTO     RELEASE 2221 19MAR19
OFP 1                              PERF 5.5
                 STD  0800 1030  STA             COST INDEX        9
                 ETD  0800 1047  ETA             ROUTE ID    LISGVA1
   CTOT NIL     ETOT  0814 1042  ELDT            GND DIST        886
                                                 AIR DIST        997
        FOB      LOAD      ZFW       LW      TOW  SPEED          ECON
MAX  18729    15140    57000    61000    68000   AVGE FF        2346
PLN   8185    10969    52829    55085    60853   AVGE W/C       M045
                                                 TKOF ALTN   .......
--------------------------------------------------------------------
REMARKS:
--------- --------- --------- --------- --------- --------- --------
MEL/CDL             DESCRIPTION
--------- --------- --------- --------- --------- --------- --------
FLIGHT PLAN ROUTE
-LPPT/03 N0448F380 IXIDA4N IXIDA DCT TOSDI UN745 ZMR UN976 DGO UT429
 LPA/N0451F360 UT429 TOPTU/N0453F380 UN871 MAKIL/N0438F300 UN871 LTP
 UZ116 BELUS LSGG/04
--------- --------- --------- --------- --------- --------- --------
           TO DEST LSGG         REMARKS:

            FUEL   TIME         AT DEP LPPT: CONT 196
TRIP        5768   0228         (29 KG INCLUDED AS PART OF TRIP)
RCONT 5 MIN  167   0005  LFLL
ALTN         969   0022  LFLL   ZFW CORR PS 1000 PLNTOF PS 100
FRSV        1055   0030                  MS 1000 PLNTOF MS 99

HLDDST         0   0000         ONE FL BELOW
ADDT          65   0002         TRIP    PS 151     TIME PS 0003
TOF         8024   0327
TAXI         161   0014         NO TANKERING RECOMMENDED (P)
FOB         8185                LOSS FOR EXTRA FUEL:  57  USD/TO
```

Fig. A.9 Lido flight plan TP946 for the Lisbon, Geneva flight

A.4 Flight data for the BTF model, Lido ™ and FlightAware ™

Table A.3 Comparing the BTF model results with Lido™'a flight plans

Origin airport	Aircraft model	Destination airport	BTF block time [min.]	Air distance [Km]	BTF model consumed fuel [Kg]	BTF model percentile	BTF model RMSE	Lido planned block time [min.]	Lido percentile	Lido RMSE	Sample size	Sample average flight time[min.]
LIS	A320	AMS	168	1911	6110	81.3	20.6	177	93.8	28.0	16	152
LIS	A320	CGN	168	1952	6183	50.0	8.7	176	77.8	12.2	9	167
LIS	A319	GVA	158	1846	5467	100.0	38.5	167	100.0	47.3	9	120
CPH	A320	LIS	237	2753	8651	100.0	18.9	233	100.0	15.1	3	219
LIS	A320	ORY	135	1492	4958	76.7	10.6	148	100.0	21.4	30	128

Appendix B

Constructive Heuristic for the Aircraft Recovery Problem Data Sets

B.1 ROADEF 2009 Challenge Data Sets

Table B.1 Instance set A characteristics

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
No. of flights	608	608	608	608	608	608	608	608	608	608
No. of aircraft	85	85	85	85	85	85	85	85	85	85
No. of airports	35	35	35	35	35	35	35	35	35	35
No. of itineraries	1,943	1,943	1,943	1,943	3,959	1,872	1,872	1,872	1,872	3,773
No. of flight disruptions	63	107	83	41	0	63	107	83	41	0
No. of aircraft disruptions	0	0	1	0	0	0	0	1	0	0
No. of airport disruptions	0	0	0	2	35	0	0	0	2	35
RTW [days]	1	1	1	1	2	1	1	1	1	2

Table B.2 Instance set B characteristics

	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
No. of flights	1,422	1,422	1,422	1,422	1,422	1,422	1,422	1,422	1,422	1,422
No. of aircraft	255	255	255	255	255	255	255	255	255	255
No. of airports	44	44	44	44	44	44	44	44	44	44
No. of itineraries	11,214	11,214	11,214	11,214	11,214	11,565	11,565	11,565	11,565	11,565
No. of flight disruptions	229	254	228	229	0	229	254	228	229	0
No. of aircraft disruptions	0	0	1	0	0	0	0	1	0	0
No. of airport disruptions	0	0	0	1	2	0	0	0	1	2
RTW [days]	2	2	2	2	2	2	2	2	2	2

Table B.3 Instance set X characteristics

	X1	X2	X3	X4	XA1	XA2	XA3	XA4	XB1	XB2	XB3	XB4
No. of flights	2,178	2,178	2,178	2,178	608	608	608	608	1,422	1,422	1,422	1,422
No. of aircraft	618	618	618	618	85	85	85	85	255	255	255	255
No. of airports	168	168	168	168	35	35	35	35	44	44	44	44
No. of itineraries	28,308	28,308	29,151	29,151	1,943	3,959	1,872	3,773	11,214	11,214	11,565	11,565
No. of flight disruptions	0	0	0	0	82	0	82	0	228	0	227	0
No. of aircraft disruptions	1	1	1	1	3	3	3	3	3	1	4	3
No. of airport disruptions	1	0	1	0	0	35	0	35	0	2	0	2
RTW [days]	3	3	3	3	2	2	2	2	2	2	2	2