Aircraft Disruption Management

Increasing Performance with Machine Learning Predictions

L.K. Hassan

Aircraft Disruption Management

Increasing Performance with Machine Learning Predictions

by

L.K. Hassan

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on 30th of April 2019 at 14:00.

Student number:	4165683	
Project duration:	February 12, 2018 – April 30, 2019	
Thesis committee:	Dr. ir. W.J.C. Verhagen,	TU Delft, chair
	Dr. ir. B.F. Santos,	TU Delft, supervisor
	Dr. D. Zarouchas,	TU Delft
	ir. H. Ritsema,	ORTEC, supervisor
	ir. J. Vink,	ORTEC

TU Delft

Abstract

Airlines experience schedule disruptions on a daily basis. Poor weather conditions, unscheduled aircraft maintenance and congested air spaces are just a few of the causes that prevent airlines from operating their flight schedules as planned. In the third quarter of 2017 over 20% of all scheduled flights in Europe suffered from delays. Operation Research based decision support systems (DSS) help airlines with their disruption management processes and provide suggestions for recovery options.

For large hub-and-spoke carriers, with an extensive network and a large number of aircraft and flights, the computation time required to find the optimal recovery solution after a disruption increases rapidly. As airlines require fast recovery solutions when disruptions occur, there is an ongoing trade-off between computation time and system sophistication. In the majority of disruption cases, no or a limited number of undisrupted aircraft are required to find the optimal recovery solution. By selecting a limited number of aircraft, flights and airports used to find a recovery solution, the computation time is reduced exponentially. The challenge is determining which aircraft and flights should be selected.

This research aims to develop a decision support system for the schedule and aircraft recovery process that is able to present a feasible solution to a disruption in less than 120 seconds. An aircraft recovery model will be developed based on the integer linear programming model that was created by Vink et al. (2019) and Vos et al. (2015). Crew and passenger recovery are not considered. To recover disruptions, the optimization model can delay and cancel flights as well as perform tails swaps, where the flights from two aircraft are switched. The novelty of the work is that machine learning is used to predict which undisrupted aircraft will help recover a disruption. Based on those predictions a sub-network selection algorithm will select the subset of aircraft to be included in the optimization instead of the entire aircraft fleet.

The performance of the system is tested on a case study for the domestic hub-and-spoke network of Delta Airlines. The dataset for the study consists of 2200 daily flights, 147 airports and 827 aircraft in 8 aircraft families. The results of the system are compared with the optimal solution, where no aircraft selections were made. The case study shows that the system is able to make an aircraft selection where 50% of the fleet is discarded, while still finding the optimal solution in 98.9% of the 556 disruptions tested. Furthermore, the system reduces the computation time by 45%, resulting in an average time of 48 seconds. For the disruptions, the computation time varied between 9 and 180 seconds.

Preface

This report is the final deliverable for my Master of Science graduation thesis on aircraft disruption management. The thesis project was performed in collaboration with the Air Transport and Operations section of the faculty of Aerospace Engineering of the Delft University of Technology and ORTEC. Over the course of the past year, I have worked on a problem that controllers in Air Operations Control Centers face on a daily basis and is an active field of research. Although my road to completing this work was a bumpy one, I have thoroughly enjoyed working on the topic. Every challenge I faced was an opportunity for personal and intellectual growth, and I am thankful for having been able to experience such a challenging topic. The completion of this work would not have been possible without certain people, whom I would like to thank.

In the first place, I would like to thank my thesis supervisors Bruno and Hendriena for our constructive discussions throughout the project. On several occasions you were able to shed some realism on my plans, thereby allowing me to reach my goals.

Jeroen, continuing your work was a daunting task. I am incredibly grateful for your guidance, feedback and day to day coaching throughout this project. Whenever I was facing set backs you were there to help me out. Even though it was utterly frustrating to see you come up with a solution in 5 minutes, I am sure I would not have been able to deliver the same quality of work without you. Thanks for keeping me sane!

Furthermore, I would like to thank Wouter, Daniel, Mor and Ronald at ORTEC for their content and implementation suggestions. Moreover, a big thanks to all my colleagues at ORTEC who were keen on helping me with the challenges I faced and for providing a supportive work environment. Pieterbas, Arnoud, Guido, Frank, Eline, Denise, Jan, Steffie, Annelies, Jasper, Astrid, Illy, Mireille, Bas, Jannigje, Eveline and Sandra, Thank you!

Additionally, I would like to thank my parents for their continuing mental and financial support throughout my studies which allowed me to develop personally and professionally.

Lotfy Hassan
Delft, April 2019

Acronyms

AAP	Aircraft Assignment Problem
ACO	Ant Colony Optimization
AOCC	Airline Operations Control Center
APD	Average Passenger Delay
ARP	Aircraft Recovery Problem
ATC	Air Traffic Control
B&B	Branch & Bound
B&P	Branch & Price
BTS	Bureau of Transportation Statistics (US)
CPM	Connecting Passenger Matrix
CRP	Crew Recovery Problem
CV	Constraint Violations
DFS	Depth First Search
DOC	Direct Operating Cost
DPC	Delayed Passenger Count
DPM	Delayed Passenger Minutes
DSS	Decision Support System
DV	Decision Variable
FAA	Federal Aviation Administration (US)
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
IATA	International Air Transport Association
KPI	Key Performance Indicator
LNS	Large Neighborhood Search
LP	Linear Programming
MAS	Multi-agent System
MCP	Missed Connection Passengers
MIP	Mixed Integer Programming
NAS	National Air System
OF	Objective Function
OTP	On-time Performance
STA	Scheduled Time of Arrival
STD	Scheduled Time of Departure
TAT	Turn-Around-Time
TFO	Time Found Out
TU Delft	Delft University of Technology
XF	Cancelled Flights
XP	Cancelled Passengers

Contents

List of Figures

List of Tables

1 Introduction

Poor weather conditions, congestion at hubs, and aircraft mechanical problems are just a few of the causes that prevent airlines from operating their flight schedules as planned. As a result departure/arrival delays, flight cancellations, and even airport closures can occur. These irregularities in operations are called disruptions. Disruptions are very common in the airline industry, greatly impacting the realized operational performance. To mitigate the effect of these disruptions, intervention by the airline is necessary to repair flight schedules, aircraft schedules, crew schedules and passenger itineraries. Consequently, any disruption results in a significant increase to an airline's operational costs, e.g.additional crew overtime, increased fuel usage and passenger re-accommodation cost.

According to statistics from EUROCONTROL[1], in the third quarter of 2017 almost 24.0% of all scheduled flights in Europe suffered from delays which equals around 6500 delayed flights per day in Europe. Ball et al. (2010) shows that in 2007, the total delay cost in the airline industry in the United States (US) was \$32.9 billion from which \$8.3 billion was of increased expenses for fuel, crew and maintenance. Because of the significant associated costs, the use of efficient and accurate recovery processes is of great importance to the airline industry.

According to Clausen et al. (2010), airlines have two ways to mitigate the effect of disruptions. The first way is to create *robust flight schedules*, where the goal is to make flight schedules and aircraft rotations less sensitive to disruptions, for example by building in more slack time between flights to absorb delays. The second way is to invest in *disruption management* processes, where the goal is to make decisions, after a disruption occurs, to recover scheduled operations as soon as possible. These decisions may include flight delays and/or cancellations. Given the cost of disruptions, improving disruption management processes is key.

The complete airline recovery process is a very large and complex problem that is commonly broken into a number of sequential stages. These stages are broadly categorized as schedule, aircraft, crew, and passenger recovery. Sequential optimization approaches do not fully capture the inter-dependencies between aircraft, crew and passengers and therefor usually result in sub-optimal recovery solutions. Both from a mathematical and computational perspective, the integration of all recovery stages (schedule, aircraft, crew and passengers) is a difficult task. With the increase in computing power in recent years, more and more researchers are integrating two or all stages of the recovery process. Especially with larger networks (e.g. the network of worldwide hub-and-spoke carriers), these integrated models take a long time to solve (generally over 20 minutes). Disruption managers in Airline Operations Control Centers (AOCC) cannot wait 20 minutes for a solution to a disruption,

[1]http://www.eurocontrol.int/articles/coda-publications

since the state of the network will have changed when the solution is provided. According to Vink (2016), from an AOCC perspective, solutions need to be provided within 120 seconds after a disruption.

In this research the potential benefits of problem size reductions during aircraft recovery within a real airline flight schedule are investigated. From the performed literature review, it was found that machine learning based problem size reductions have not been addressed and few researchers verified their model on real world airline schedules.

The objective of this research is to develop a decision support system for AOCCs that is able to present a feasible solution to a disruption in less than 120 seconds while minimizing the cost of the disruption. The novelty of the work is that machine learning is used to select a sub-network of resources that will be included in the optimization instead of the entire network.

Chapter 2 addresses the literature review that was performed for this research. In the end of that chapter the gap in the body of knowledge is identified. Chapter 3 elaborates on the framework that was used for the research, including the proposed methodology and hypotheses. The framework for the decision support system is presented in Chapter 4. It should be noted that the framework is largely similar to the framework presented by Vink et al. (2019). The mathematical formulation of the optimization model, again largely similar to the formulation presented by Vink et al. (2019), is discussed in Chapter 5. Chapter 6 will give an introduction into machine learning and will discuss the choices that were made to construct the aircraft classifier used to reduce the problem size. Verification and validation of the decision support system is presented in Chapter 7. A case study, performed on a dataset for Delta Airlines, is discussed in Chapter 8 and the conclusions and recommendations are given in Chapter 9.

This research is a continuation of the work carried out by Vos et al. (2015) and Vink et al. (2019). This research adds the following to the decision support system (DSS) and literature:

- The machine learning classifier as presented in Chapter 6. Moreover, a quantification of the potential benefits of problem size reductions using machine learning techniques in terms of computation time and solution quality is presented in Chapter 8.
- The extension to the Connecting Passenger Matrix as discussed in Section 6.2.4.
- The generated schedule, aircraft and disruption datasets for Delta Airlines as presented in Chapter 8, which allows for extensive verification and validation of the DSS.

This work is carried out as an Masters thesis for the Air Transport and Operations section at the Aerospace Engineering faculty of the Delft University of Technology (TU Delft).

2 Literature Review

Chapter 1 introduced the topic of this research. This chapter will elaborate on the literature review that was performed to gain better insight in the state of the art.

Section 2.1 will discuss common network representations used to model airline operations. Section 2.2 will review the literature on aircraft recovery. Finally, Section 2.3 concludes the chapter with a discussion on the gap in the body of knowledge and the position of this work within the body of knowledge.

It should be noted that Hassan et al. (2019) have written a literature review paper on airline disruption management. This chapter is a summary of that paper.

2.1. Network Representations

To translate real-world scheduling, routing and planning problems to mathematical models, network representations are commonly used. In the airline recovery context, networks are usually build for the duration of a recovery window, beginning at the time of disruption and ending at the time the schedule is proposed to be recovered. Source nodes represent the current positions of aircraft while sink nodes represent the possible positions at the end of the recovery window. The schedules in the recovery window are repaired while the scheduled flights outside of the recovery window remain unchanged. Three representations are commonly used in literature regarding airline planning and recovery management: connection networks, time-band networks and time-space networks. This section will briefly present all network representations, while a more extensive explanation is given by Clausen et al. (2010).

2.1.1. Connection network

A connection network is an activity-on-node network, where schedule information is used to generate the nodes in the network. All flight legs are represented by nodes in the network, which are defined by the origin airport, destination airport, departure time and date and arrival time and date. Directed edges (arcs) represent the aircraft connection between flight legs. Additionally, source and sink nodes represent the possible origin and destination airports of the aircraft in a fleet at the beginning and end of a day of operations. Nodes i and j are connected by an arc (i,j) if and only if it is feasible to fly flight leg j after flight leg i with the same aircraft with respect to turn-around-times (TAT) and airport. A feasible day of operations for an aircraft from origin to destination airport is represented in the connection network by a path. One of the drawbacks of connection networks is that time information is not presented. As a result it is unclear from the network what the location is of a specific aircraft at a certain time. Connection networks are extensively used in literature, for example by: Teodorović and Guberinić (1984), Maher (2015) and Wu et al. (2017c).

2.1.2. Time-Band network

Arguello (1998) presented the time-band network representation for airline disruptions, which is an activity-on-edge network. Nodes in the network represent the time and location (airport) of arriving or departing flights. There are two different sets of nodes: station-time nodes and station-sink nodes. Activities at an airport are aggregated in discrete time intervals, called time bands, and represented by the station-time nodes. The time coordinate of the station-time nodes correspond to the availability time of the first available aircraft in that time band. The end of the recovery period at each airport is represented by the station-sink nodes. Each station-time node is connected to the station-sink node of that location. For a flight from airport A to B, departure nodes are created at airport A for each aircraft that becomes available for that flight. Arcs are created from each airport A node for which there is an aircraft available that can fly the given flight within the recovery period. These arcs will end in the corresponding airport B node at the time that aircraft becomes available at airport B. Due to the aggregation of notes the size of the time-band network is reduced. However, this is achieved at the cost of time accuracy. Examples of papers that use the time-band network representation are: Bard et al. (2001), Eggenberg et al. (2010) and Hu et al. (2015).

2.1.3. Time-Space network

Time-space, also referred to as time-line, networks are similar to time-band networks since nodes correspond to a specific location and time. Unlike time-band networks, the nodes are not aggregated in a time-band. Arrivals and departures of aircraft are represented by nodes on the time-line of a specific airport. The time-space representation is an activity-on-edge network, where arcs between nodes represent the activities of aircraft. Arcs between nodes of different airports correspond to feasible flight legs while arcs between nodes of the same airport correspond to grounded aircraft. Examples of applications of time-space networks can be found in: Sinclair et al. (2014), Arikan et al. (2017) and Marla et al. (2017).

2.2. Review of Aircraft Disruption Management

The complete airline recovery process is a very large and complex problem that is commonly broken down into a number of sequential stages. These stages are broadly categorized as schedule, aircraft, crew, and passenger recovery, also defining clear boundaries for research in this area. Schedule and aircraft recovery is commonly solved at once. Since Clausen et al. (2010) have published an excellent literature review of the work until 2009, this literature review focuses on reviewing the airline disruption literature between 2009-2018, classifying them by solution methodology, i.e. exact optimization methods, (meta-)heuristics or multi-agent systems.

Exact optimization methods, such as branch & bound and dynamic programming, guarantee finding the global optimal solution. With most optimization problems, exact methods are the method of choice. With NP-hard problems, such as the airline recovery problem, the situation is different since the computation time grows exponentially with problem size and exact methods become intractable. Even medium-sized problems use extensive computation time to solve, which makes them unfit for operational use. To overcome these problems, (meta-)heuristics can be used. These methods are commonly applied to solve computationally intractable combinatorial optimization problems to a near-optimum. The effectiveness and quality of solutions depend on the heuristic's ability to adapt to a particular problem, exploit the problem structure and avoid getting stuck in local optima. A Multi-Agent System (MAS) is a software system composed of multiple interacting intelligent agents. Here, intelligence may be algorithmic search, reinforcement learning or procedural approaches among others. MAS typically refers to software agents, but could equally well be humans. In the context of airline disruption management, MAS usually represent the Operational Control Center of the airline.

Figure 2.1 shows the number of publications per recovery type per year in the field of airline disruption management. This chart provides some interesting insights: (1) the interest for solving airline disruption management problems is increasing, (2) roughly 50% of papers have been published after the last literature review paper (Clausen et al. (2010)), and (3) since then, there is an increase in the number of publications that integrate two or more resources in the recovery process.

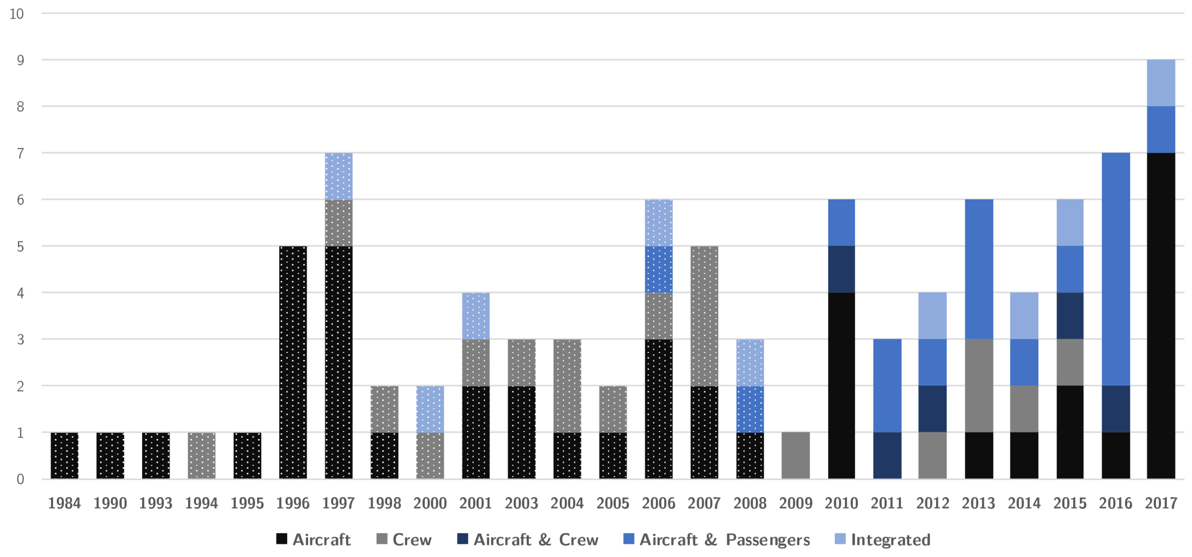


Figure 2.1: Number of publications per recovery type per year. Papers that have been reviewed in previous publications are shown as patterned.

Section 2.2.1 will present the literature focused exclusively on aircraft (and schedule) recovery. There are several publications that integrate two or more stages of the recovery process. Section 2.2.2 discusses papers that integrated aircraft and passenger recovery while Section 2.2.3 discusses papers covering integrated aircraft and crew recovery. Literature that integrates the full recovery process, that is schedule, aircraft, crew and passenger, is presented in Section 2.2.4.

2.2.1. Aircraft Recovery

Before 2009, the majority of publications focused on aircraft recovery, in part because (1) aircraft are the most constraining and expensive resource and (2) aircraft recovery is a smaller and simpler problem than crew recovery. Despite this, aircraft recovery is still an active research subject, where the efforts have been focused on increasing complexity to better represent real-world networks and decreasing the computation time. In the following subsections the literature on aircraft recovery after 2009 will be discussed and classified.

Initial efforts

Teodorović and Guberinić (1984) were the first authors that discussed the minimization of passenger delays in the aftermath of schedule perturbations. Their methodology utilizes branch & bound methods and is based on the assumption that the airline operates only one aircraft type. Furthermore, maintenance constraints are ignored and the model was tested on a network of 8 flights operated by 3 aircraft. Passengers are explicitly modeled, but they assume that all itineraries contain only a single flight leg. Teodorović and Stojković (1990) extended this work by considering airport curfews. A dynamic programming based approach is used where the goal was to minimize the total number of cancelled flights and the total passenger delay. The model was tested on a network of 14 aircraft and 80 flights. Crew constraints are included in the new work of the same authors, Teodorović and Stojković (1995). This new paper presents a heuristic based on the First In, First Out (FIFO) principle and a dynamic programming based sequential approach. The model determines aircraft and crew rotations while minimizing the total number of cancelled flights. The model was tested on 240 generated instances. Four to five different disturbances were arbitrarily generated for each of the 240 numerical instances, so that the developed models were tested on over 1,000 different situations.

Exact Optimization methods

Eggenberg et al. (2010) extended the work of Bard et al. (2001) and presented a column generation algorithm where a time-band network model is used to show the aircraft routes. Each unit (an aircraft, crew member or passenger) is associated with a specific recovery network and the model considers unit-specific constraints. The column-generation algorithm ensures global feasibility according to the structural constraints of the problem. The usual multi-commodity approach struggles with considering unit-specific constraints, which the authors overcome with the proposed solution. While the presented results seems promising and the majority of instances solve within 100 seconds, the authors report that for the most computationally expensive case the run time exceeds 1 hour. Furthermore it should be noted that the case instances are only tested with a single fleet type.

Aktürk et al. (2014) were the first to successfully integrate cruise speed control to deal with the Aircraft Recovery Problem (ARP). Due to the non-linearity of fuel burn in cruise speed, the authors use a conic quadratic optimization approach to solve the problem with minimization of recovery related costs like tail swap cost, increased fuel consumption and passenger delay cost. Environmental cost and constraints were 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.

Whereas other researchers validated models with a static disruption scenario, Vos et al. (2015) established a dynamic framework, named Disruption Set Solver for the aircraft schedule recovery. Instead of having full knowledge of all disruptions upfront, the framework handles disruptions as they happen and builds on the solutions of previous disruptions. The framework relies on the combined usage of an efficient aircraft selection algorithm and a linear-programming model which is able to track the status of individual aircraft on parallel time-space networks. The framework is applied to a set of real disruptions in the operations of Kenya Airways. In 93.3% of the cases the system found solutions within 10 minutes. Furthermore, the authors showed that the disruption costs are underestimated when using a static approach.

Given the inherent uncertainty of the aircraft recovery problem, several authors presented (partially) stochastic approaches. Arias et al. (2015) combined constraint programming with a simulation approach to solve the Stochastic Aircraft Recovery Problem. The goals of the model is to restore the original flight schedule as much as possible, minimizing the total flight delay and the number of cancelled flights. 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. The proposed model is tested with real data from a commercial airline with a total of 51 flights, 13 airports and 11 aircraft. The proposed model is able to 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 optimization methods.

Xu et al. (2016) presented a time-band approximation model with approximate delay cost considering a stochastic flying time. With data on the actual flying time and the planned flying time from 400 flights in a day of Sichuan Airlines, the authors 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. Xu and Han (2016) extended the work by presenting the weighted time-band approximation model which incorporates a simplex group cycle approach. Here the model is tested on data from China Airlines.

Wu et al. (2017c) were the first to adopt the iterative fixed-point method for integer programming (presented by Dang and Ye (2015)) for the construction of feasible flight routes. Two methods are presented to divide the solution space into independent segments and solve them with distributed computation. Since the segments are independent, the calculation of integer points can proceed parallel on each processor. Two division methods were proposed. The first method attempts to divide the solution space in segments that contain roughly equal integer points. The second method, specially proposed for long haul problems, takes the original flight routes as initial points. The algorithm is compared to the solutions of CPLEX CP Optimizer. In the majority of cases, the number of partial feasible flight lines, which have to be calculated for finding an optimized airplane schedule, is much fewer compared with the number needed by CPLEX. This makes the method a promising alternative to further develop in the future. Wu et al. (2017b) extended the work by considering multiple fleets, while Wu et al. (2017a) focused on disruptions caused by airport closures.

(Meta-)heuristics
Gao et al. (2009) developed a greedy simulated annealing algorithm, combining characteristics of Greedy Randomized Adaptive Search Procedure (GRASP) and Simulated Annealing. The combination of heuristics improves the efficiency of the neighborhood selection and decreases the probability of falling into a local optimum. The objective of the model is to minimize the total passenger delay time, which consists of delayed time in delayed and cancelled passengers. One drawback of the model is that the objective function does not take into account all cost incurred by irregular operations e.g. the cost of ferrying and fleet substitution is not taken in to account.

Liu et al. (2010) presented a hybrid heuristic which combined an adaptive evaluated vector (AEV) and an inequality-based multi-objective genetic algorithm (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. The model does not consider flight cancellations as a recovery method. The presented model is tested on a daily plan of a Taiwanese airline with 7 aircraft (single fleet) during a 1 hour airport closure, impacting 39 flights. The heuristic presents results in 3.6 minutes on average (7.5 minutes max). Due to the computation time, this model cannot be used in an operational setting for larger problem sizes.

A hybrid heuristic was also used by Xiuli and Yanchi (2012), who combined a Greedy Random Adaptive Search Procedure (GRASP) with Ant Colony Optimization (ACO). Compared to the original GRASP algorithm, it provides a high global optimization capability. The authors state that the model was tested on a multi-fleet network with 50 aircraft and more than 5 aircraft types. However, no results are presented.

Wu and Le (2012) developed a model based on flight strings instead of individual flights. They transform these strings to a time-space model which 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. The model is tested on a dataset from China Airlines consisting of 170 flights, 5 fleets, 35 aircraft and 51 airports. Although the solution method is able to obtain solutions that are near optimal ($\leq 1\%$ optimality gap), no computation times are given.

Zhu et al. (2015) presented a two-stage stochastic recovery model to deal with the ARP. The first stage is a resource assignment model with the objective of minimizing delay and cancellation cost. The second stage re-times the aircraft routings obtained in the first stage, with the objective of minimizing 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 as several linear models.

Another research using ACO was presented by Sousa et al. (2015). The proposed algorithm combines the Aircraft Assignment Problem (AAP) with the ARP and aims to minimize the operational cost and (re-)scheduled flights dynamically by using a rolling time window. Two different experiments, both using real data from a commercial airline, were conducted. On a problem with 100 flights, the ACO outperforms (non-truncated) branch & bound and Depth First Search (DFS) in terms of solution quality, although it takes 40% more time on average.

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

Zhang (2017) proposed to use feasible lines of flights (LOF) as the basic variables in the model, where LOFs 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 LOFs, thereby reducing the run-time. In the first stage, LOFs are scored and selected based on the amount of swaps (less is better) and the number of flights legs included in the LOF (more is better). In the second stage, flow balance constraints for the aircraft are aggregated by creating constraints for each airport only. The disruptions included in the model are airport closures and aircraft unavailability due to unplanned maintenance. The approach is tested on five real-life test scenarios. The largest instance included 44 aircraft and 638 flights, where the solution was computed in 150 seconds.

From the literature on aircraft recovery it can be seen that there is a trade-off between computation time, problem size and problem complexity. Since AOCCs require solutions within approximately 120 seconds, some of the presented solutions, e.g. Arias et al. (2015), Hu et al. (2017), Vos et al. (2015), cannot be implemented in practice. The methods proposed by Gao et al. (2009), Eggenberg et al. (2010) and Sousa et al. (2015), for example, manage to obtain reasonable computation times. Unfortunately, they only consider a single fleet, which does not represent the reality at most airlines. Most other papers do not consider all recovery options common at airlines or do not take maintenance and/or airport constraints into account, thereby simplifying the problem. Xiuli and Yanchi (2012) consider all recovery options, maintenance constraints and includes multiple fleets. However, it does not present the number of flights in the case study nor the computation times. In the majority of papers, the delay cost are calculated by using constants to express the average delay cost per minute. Similarly a constant parameter is used to express the average cancellation cost of a flight. This approach usually underestimates the cost, due to the non-linear relation between goodwill loss and the amount of delay , as discussed by Arikan et al. (2017).

2.2.2. Aircraft and Passenger Recovery

In the last decade there has been a trend towards integrating more than one resource in recovery models. Sequential optimization approaches do not fully capture the inter-dependencies between aircraft, crew and passengers and, therefore, usually result in sub-optimal recovery solutions. The papers in this section attempt to overcome these downsides by simultaneously solving the aircraft and passenger recovery.

Exact Optimization methods

Jafari and Zegordi (2010) presented an assignment model for solving the aircraft recovery problem and reassigning disrupted passengers simultaneously, using a rolling horizon time framework. The objective is to minimize the sum of aircraft assignment costs, delay costs, cancellation costs, and disrupted passenger costs. The proposed approach utilizes a wide range of recovery actions. The model used aircraft rotations and passenger itineraries instead of flights. The study did not consider maintenance constraints. Jafari and Zegordi (2011) extended the work. Due to the high complexity of the algorithm, the method was only tested on disruptions with 13 aircraft of 2 fleet types. The authors do not demonstrate that the method is computational efficient, nor do they show that the model is able to deal with disruptions that reflect operations of a larger airline. Zegordi and Jafari (2010) solved the same problem with an ACO heuristic. The experiments show that the ACO algorithm is able to build a revised schedule in less than 26 seconds. According to the authors, the method was implemented at an airline. The algorithm does not consider scenarios where aircraft from different flight rotations recover each other, thereby limiting the solution space.

Hu et al. (2015) presented an integrated integer programming model based on an approximate reduced time-band network and a passenger transiting relationship. The authors extend their earlier work to model multi-fleet aircraft routing. The objective is to minimize the total cost associated with the reassignment of aircraft and passengers to flights. One assumption the authors make is that all passenger itineraries are comprised of a single flight leg. A feasibility study is conducted to find the conditions under which aircraft and passenger recovery is possible. The authors test the model on 10 scenarios with real data of a Chinese airline with over 180 aircraft, 113 fleets and over 620 flights. All scenarios take less than 172 seconds to solve where the LP relaxation results in an maximum optimality gap of 8.53%.

Using a mixed integer non-linear programming model, Arıkan et al. (2016) modelled the aircraft recovery problem and the passenger recovery problem. The authors employ several recovery actions such as re-timing departures, cancelling passenger itineraries and flight planning (cruise speed control). The goal of the model was to minimize the passenger related costs and fuel costs. Due to the non-linearity of the cost associated with fuel consumption, a LP model is no longer applicable. However, the authors reformulate the non-linear model as a conic quadratic mixed integer programming model, similar to Aktürk et al. (2014), that can be solved efficiently. The authors used a time-space network representation to model the aircraft and passenger itineraries. The paper shows the impact of cruise speed control on the airline disruption problem and the ability to reduce cost, showing that cruise speed control is a feasible recovery technique. In a later paper (Arikan et al. (2017)), the authors mentioned that the proposed formulation is not flexible, since it cannot be extended (easily) with other entity types, such as aircraft crew and passengers, and recovery actions. In the same paper the authors propose a more generalized network structure, which will be discussed in Section 2.2.4.

Recently, Marla et al. (2017) extended the traditional recovery actions such as aircraft swaps, flight cancellations and passenger re-bookings with flight planning. The same time-space network representation from Bratu and Barnhart (2006) is utilized. Departure time decisions are incorporated by creating copies of flight arcs at different departure time alternatives. The cruise speed control alternatives are incorporated by generating a second set of flight copies for different cruise speed alternatives for each departure time alternative. This approach requires a discretization of the cruise speed options and increases the size of the generated network. In the paper, the authors propose an approximation model that deals with larger airline networks. The model is steered away from solutions that would result in passenger disruptions, by explicitly assigning cost to delayed flight that carry connecting passengers. A case study was performed on data from an European airline with about 250 daily flights in a hub-and-spoke network. The computation time is limited to 120 seconds. The proposed model is compared to a baseline sequential recovery model, several variants of an aircraft-centric model with flight planning and several variants of an passenger-centric model without flight planning. Based on the airlines historical data, 60 scenarios are considered. Based on the results the authors conclude that their enhanced recovery models reduce total costs and passenger-related delay costs for the airline, compared to existing approaches.

(Meta-)heuristics
In 2009, the French Operational Research and Decision Support Society (ROADEF) organized an OR challenge regarding disruption management for commercial aviation, which was proposed by Amadeus[1]. This challenge resulted in several publications. Bisaillon et al. (2011) formulated a large neighborhood search (LNS) heuristic that combined fleet assignment, aircraft routing and passenger assignment. The heuristic cycles through three phases: construction, repair and improvement. These phases destroy and repair parts of the solution in an iterative manner. The model constructs aircraft routes and passenger itineraries for the recovery period with the goal of minimizing operating cost and impact on passenger. The first two phases produce the initial solution, while taking into account the operational and functional constraints. The third phase considers large schedule changes and tries to improve the solution while maintaining feasibility. This work won the ROADEF 2009 challenge. Sinclair et al. (2014) improved the work of Bisaillon et al. (2011) by making changes in each of the three phases, in order to find better final solutions. In the construction phase, the aircraft that caused the highest cost when cancelled were prioritized. In the repair phase, the focus was on re-booking passengers with disrupted itineraries as well as covering flights that were cancelled in the construction phase with spare aircraft. In the improvement phase the authors attempt to accommodate disrupted passengers by delaying flights. The improved model was tested on the ROADEF 2009 dataset. The algorithm found 17 best solutions for 22 instances in five minutes and 21 best solutions in 10 minutes. Sinclair et al. (2016) extended on the work in Sinclair et al. (2014) and Bisaillon et al. (2011) by presenting a post-optimization column generation heuristic that reduces the model size to improve solutions within reasonable run-times. By defining dual variables after solving the LP relaxation, the reduces cost of the variables are calculated. The variables with negative reduces cost are considered when re-solving the LP problem. The model was tested on the ROADEF 2009 Challenge dataset and found best known solutions to all scenarios. The authors suggest future research should focus on implementing a rolling-time horizon with the column-generating algorithm.

Mansi et al. (2012) proposed a heuristic based on exact methods and an oscillation strategy. In the first phase, the heuristic solves a relaxation of the problem in order to find a feasible solution for aircraft and passengers close to the initial schedule. If no feasible solution is obtained, a dynamic programming algorithm refines the alternatives and generates a feasible solution. In the second phase, the oscillation strategy alternatively destroys and constructs aircraft routes and passenger itineraries and assigns them to aircraft and passengers simultaneously. This work received the second prize in the challenge.

Jozefowiez et al. (2013) presented a three-phase heuristic. In the first phase, the disruptions are integrated in the schedule. Each disruption is solved by a separate algorithm, flight legs are removed and passenger itineraries are cancelled in order to return a feasible solution. The second phase attempts to re-assign disrupted passengers with the same origin and destination to itineraries, using a shortest path problem. In the third phase, new flight legs are added to the schedule in an attempt to recover the remaining disrupted passengers. Passengers are grouped by itinerary and based on the size of the group a prioritization is made. This work was also one of the finalists of the ROADEF 2009 Challenge. Although it did not perform as well as Bisaillon et al. (2011), the algorithm did not keep iterating the full 10 minutes, but reached a feasible solution for all cases in less than 4 minutes.

Le et al. (2013) transformed the aircraft and passenger 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. The model only considers aircraft ferrying and departure delays as recovery options, while in reality more options are available. The problem is solved with a genetic algorithm which is tested on a small network from a regional Chinese airline. For three different disruption scenarios the GA solved within 100 seconds.

[1]`http://www.roadef.org/challenge/2009/en/`

Zhang et al. (2016) developed a three-stage sequential heuristic framework to solve the integrated aircraft and passenger recovery problem. In the first stage, the flight schedules and aircraft rotations are recovered. The next two steps iteratively solve the flight rescheduling problem and the passenger recovery problem. A time-space network representation is used together with a mixed-integer programming formulation of the model. The proposed algorithm is tested based on the same data sets used by the ROADEF 2009 challenge. The algorithm is able to beat the finalists of the challenge on all datasets.

Hu et al. (2016) proposed a mathematical model based on the flight connection network and the passenger reassignment relationship. To solve the problem, a heuristic based on a Greedy Randomized Adaptive Search Procedure (GRASP) is adopted. The heuristic is tested through experiments based on generated and real datasets. For all test instances, a solution was found within 100 seconds. The authors compare the results of the heuristic to a sequential solution approach and show that their heuristic is able to find higher quality solutions. However, the solution costs are not compared to a global optimum, so the (near-)optimality of solutions is not presented.

From the literature review on aircraft and passenger recovery it can be concluded that the ACO approach by Zegordi and Jafari (2010) seems promising for practical implementation, since it considers all relevant recovery options and maintenance constraints while still managing to solve a real life case in 26 seconds with ≤1% optimality gap. It is unknown how the computation times scale with problem size. Another promising paper is Hu et al. (2015), which, as discussed above, proposed an approximation approach that is able to solve several real-life instances in under 172 seconds with an optimality gap ≤9%. Unfortunately, the authors do not currently take maintenance constraints into consideration. A more recent work by the same author(s) is Hu et al. (2016), which was able to solve several disruption instances in less than 100 seconds. The authors mention that the results are near-optimal, however the global optimal solution is not given for the instances. Most other papers either do not include all recovery options that are common in real operations or take a long time to solve.

2.2.3. Aircraft and Crew Recovery

The previous section presented the literature on the simultaneous aircraft and passenger recovery. In this section, publications on simultaneous aircraft and crew recovery will be discussed. To the best of the authors knowledge, there are no pre-2009 papers that present solutions on the combined aircraft and crew recovery. Aguiar et al. (2013) was the first to suggest a solution method for this problem, as will be discussed below.

Exact Optimization methods

Maher (2016) proposed a column-and-row generation framework that extends existing branch & price (B&P) models and reduces the problem size. The model employs departure delays and cancellations as recovery techniques. The proposed model is compared to a column generation model. On average, the column-and-row generation model had a 27% lower run-time. The authors tested the model on both a point-to-point and a hub-and-spoke network with 262 and 442 flights respectively.

(Meta-)heuristics

Aguiar et al. (2013) used and compared several different meta-heuristics such as hill-climbing, simulated annealing and genetic algorithm to solve the sequential aircraft and crew recovery problem. For the aircraft recovery, a multi-objective approach that optimized delays and other cost associated with aircraft was developed. Hill-climbing, Simulated Annealing and Genetic algorithm were used to solve the ARP. Genetic algorithm outperformed the other heuristics, although all heuristics performed well. The solution of the ARP serves as the input for the crew connecting problem. To solve the CCP, hill-climbing and simulated annealing algorithms were developed and tested on data from TAP Portugal.

For the crew connecting problem, the simulated annealing algorithm performed best in terms of crew cost. None of the results are compared with the global optimum, so although feasible solutions are given, the quality of those solutions cannot be determined.

Zhang et al. (2015) proposed a two-stage heuristic for the sequential aircraft and crew recovery problem. In the first stage, the aircraft recovery with partial crew considerations model is built. This model is based on the traditional multi-commodity network model for the aircraft schedule recovery problem. In the second stage, the crew schedule recovery with partial aircraft consideration model is build. The authors propose a new multi-commodity model for the crew schedule recovery. The two stages are run iteratively until no improvement is found. The proposed algorithm is compared to the integrated model of Abdelghany et al. (2008) and a sequential algorithm. The proposed algorithm is able to improve the solutions of the other two approaches in literature for all scenarios. Although the algorithm has a higher run-time, it never exceeds 72 seconds.

2.2.4. Integrated Recovery

Both from a mathematical and computational perspective, the integration of all recovery stages (fleet, aircraft, crew and passengers) is a difficult task. To the best of the authors knowledge, the first proposal of a truly integrated approach is the PhD Thesis of Lettovsky (1997), where the author formulated the 'Airline Integrated Recovery' problem which consists of: aircraft routing, crew assignment and passenger flow. The thesis presents a linear mixed-integer mathematical problem that maximizes total profit to the airline while capturing availability of the aforementioned resources. A decomposition scheme is presented where the 'Schedule Recovery Model' master problem controls the three sub-problems known as the 'Aircraft recovery model', 'Crew recovery model' and 'Passenger flow model'. The solution is derived by applying Benders' decomposition. An important limitation is that the model only considers flight crew and not cabin crew. This subsection will present the fully integrated recovery papers between 2009-2018.

Multi-Agent Systems

Castro et al. (2014) presented a 'Multi-Agent System for Disruption Management' (MASDIMA) and a related work analysis and comparison with MASDIMA. The proposed MAS is capable of autonomously monitoring the operations of the airline and deciding whether an events requires action or not. The MAS is adaptive to the environment and includes learning capabilities. Furthermore, the MASDIMA allows for human-in-the-loop inclusion, which improves user acceptance of the solutions by reacting and learning from the user preferences. According to the authors, the main advantages of their approach are: generates integrated (i.e. that included all parts of the problem) and more balanced solutions (in terms of the objectives of each part).

Exact Optimization methods

Arikan et al. (2017) developed a new flight network representation for the integrated recovery problem, based on the flow of each entity (aircraft, crew and passenger) through the network. With the proposed flight network, the problem size is kept within limits so that real-time solutions can be provided since it does not require discretization of departure times and cruise speed decisions. Similar to Aktürk et al. (2014), the authors implemented aircraft cruise speed control and proposed a conic quadratic mixed integer programming formulation. The model explicitly models passengers, thereby evaluating the passenger delay costs more realistically. The authors test the model on a network of a major U.S. airline. The effect of the the pre-processing methods, the cruise speed control, the passenger delay function, the severity of the disruptions and the length of the recovery horizon on the optimality gap and run-time are evaluated.

(Meta-)heuristics
Petersen et al. (2012) presented an integrated optimization approach that resembled the one used by Lettovsky (1997), where they distinguish between four sequential phases: schedule recovery, aircraft rotations, crew assignment and passenger assignment. In the first phase, the schedule is repaired by flying, cancelling, delaying or diverting flights. Then, in the second phase, aircraft are assigned to the new schedule. Third, the crew is assigned to the aircraft rotations. In the last phase the passenger recovery ensures that all passengers arrive at their final destination. The authors tested the model with data from a regional US carrier that operates a hub-and-spoke network with 800 daily flights. The results of the proposed integrated model are compared to the results of a sequential approach. Where the proposed approach always finds a feasible solution, the sequential model only finds a feasible solution in 75% of the cases. The results show that the costs of the integrated approach are always equal or lower than the cost of the sequential approach. The computation time of the proposed solution ranges between 20-30 minutes. Currently, the network is rebuild with every disruption. The authors note that, by building the network in advance, the computation time could be reduced.

Maher (2015) presented a column and row generation approach to solve the integrated recovery model. The framework is based on general column generation and Bender's decomposition, which improves the run-time and quality of the solution. Using the Big M method, costs are assigned to the objective function when disrupted passengers are not assigned to a flight that recovers the itinerary. By using the Big M method, infeasibilities due to conflicting constraints are prevented, while as much passengers as possible are recovered. Due to the integration of passengers, the run-time increases. Solution times range between 500-2700 seconds depending on the scenario.

Following the literature review presented in this section, it can be concluded that Maher (2015) does not include swapping as an recovery technique and the model formulation does not allow for multiple fleet types. The model formulations of Petersen et al. (2012), Castro et al. (2014) and Arikan et al. (2017) are fit for use in an AOCC. Unfortunately, the computation times for the given case studies are too long for operational implementation

2.3. Conclusion on Literature Review

From the literature review and the overview of publications in Figure 2.1 it can be concluded that airline disruption management is an active field of research with an increasing interest in integration of several resources in the recovery process.

With an increase in computing power and knowledge gained through research, it is likely that larger more complex problems will be developed in future. These models may include larger flight schedules, fleets or longer time windows. Furthermore, additional constraints related to maintenance, airport operations or recovery options are likely to be included in future work. To capture the full complexity of airline operations, researchers are integrating more resources, such as crew and passengers, into the disruption management systems. Moreover, recently, stochastic elements are included to capture the uncertainty of decision making related to the future state of the network.

In the current body of knowledge, few researchers have been able to create systems that accurately model reality and are computationally efficient enough to use in AOCCs. With the projected increase in model size and complexity, more research has to be done on increasing computational performance. This research aims to quantify the potential benefits of problem size reductions using machine learning techniques in terms of computation time and solution quality. Considering that, for most disruptions, the flight schedule of less than 10% of the aircraft in the network are required to change, it would be beneficial to be able to classify and identify aircraft based on their probability of aiding in the recovery solution.

3 Research Framework

The previous chapter presented the literature review and the conclusion regarding the current state of the art and the position of this work in the body of knowledge. This chapter will present the research scope, main research question, sub questions and research objectives.

3.1. Research Scope

Airline disruption management focuses on the operational phase of airline operations. Figure 3.1 shows the airline planning framework. Operations control focuses on the optimization of the day-to-day operations. Given the position of operations control in the framework, most data and decisions, such as fleet composition, flight schedule, aircraft assignment and crew assignment, are already fixed. Furthermore, as tickets are already sold in this phase the (re-)optimization focuses solely on cost minimization and not revenue maximization.

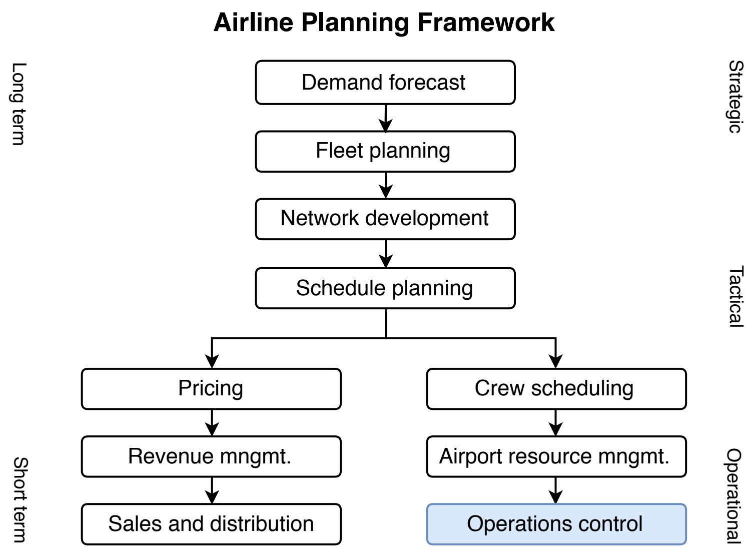


Figure 3.1: Airline planning framework, focus of this research is highlighted. Adapted from Barnhart (2003)

The disruptions that will be taken into account will be in one of three categories: (single) aircraft unavailability, flight delay or airport unavailability. Most of the common disruption causes can be classified as either of those categories.

Within the broader field of airline disruption management, this research focuses on the aircraft recovery problem (ARP). This research builds on the work of Vink et al. (2019), that was focused on including maintenance constraints and connecting passenger cost in the ARP. This research aims to extend the previous model by developing a classification algorithm that predicts which subset of the aircraft in the network will aid in recovering the schedule after a disruption. The research question for the MSc thesis can be formulated as follows:

Research question
How can machine learning techniques be incorporated into the aircraft recovery problem, and how does this integration help the recovery solution in terms of solution cost and computation time?

3.2. Research Objectives and Hypotheses

The research objective is formulated by incorporating the goals from an academic and industry perspective. The academic objective is focused on solving the aircraft recovery problem while considering the novelty of the machine learning methods. Based on discussions with industry experts, Vink et al. (2019) stated that the industry objective is focused on minimizing the cost following disruptions and obtaining a feasible solution within 120 seconds.

Research objective
Decrease the solution time of the aircraft airline recovery problem to less than 120 seconds while minimizing the cost of disruptions by developing a machine learning based selection algorithm that selects a sub-network to include in the optimization problem.

The research objective can be segmented into the following sub-goals:

1. Develop an efficient (near-)exact optimization model for aircraft recovery with all relevant constraints and recovery techniques.
2. Pinpoint the inputs and outputs that are necessary for both the machine learning algorithm and the exact optimization model.
3. Develop a machine learning algorithm which predicts, for each aircraft, the probability that including that aircraft in the exact optimization model will positively impact the feasibility and solution cost.
4. Train the machine learning model on a real airline schedule such that following the recommendations of the algorithm will result in a feasible, near-optimal recovery solution within 120 seconds.
5. Verify and validate the model.

Based on the literature review and the research questions, two hypotheses are formulated. This research will investigate whether or not these hypotheses are correct.

The first hypothesis is related to the classifier and sub-network selection algorithm that are to be developed. The primary objective of this thesis is to evaluate the performance of a machine-learning based classifier and its effect on the solution quality.

Hypothesis one
Based on real schedule, disruptions, network and fleet data of an airline, a machine learning model can be trained to select a sub-network that will result in a feasible recovery solution.

If the first hypothesis is proven, the effect on the computation time and solution quality are to be evaluated. Because the problem size is reduced, it is hypothesized that the computation time will decrease, however the effect on the solution quality is to be determined.

Hypothesis two
Making a sub-network selection based on machine learning predictions will decrease the computation time of the optimization model while not significantly deteriorating the recovery solution.

The proposed method to test these hypotheses is discussed in the following section.

3.3. Methodology

To test the hypotheses presented in the previous section, a decision support system will be developed that presents a recovery solution given the current schedule and the disruptions. An aircraft recovery model will be developed based on the integer linear programming model that was created by Vink et al. (2019). The optimization model and the (near-)optimal solution to each disruption will be used to train a machine learning model.

A case study with real airline data will be performed to test the newly developed decision support system. The first hypothesis will be tested by having the machine learning algorithm suggest a sub-network based on the current schedule and disruptions. The optimization engine will use this sub-network to find a solution to the disruptions. If these sub-network suggestions result in feasible recovery solutions, the hypothesis will be proven.

If hypothesis one is proven, the second hypothesis can be tested. The computation time and solution of the developed decision support system will be compared with the recovery solution found by using the (sub-)fleet in the optimization. The hypothesis will be proven correct if the system can predict sub-networks that reduce the combined computation time of the machine learning algorithm and optimization engine to less than 120 seconds.

In short, an experiment will be conducted. Specifically, a computer simulation will be designed to imitate the aircraft recovery process. While this will result in a high level of internal validity, it will not result in external validity. Some degree of external validity can be achieved by performing the case study.

Given the research objective, the combined computation time of the machine learning algorithm and the optimizer should be less than 120 seconds in an operational setting. While most researchers test their models on desktops or laptops, this is far from an operational setting. Most AOCC's have dedicated servers for scheduling tasks and optimization. Therefore, the computational set-up that will be used to run and time the program will be a Microsoft Azure Standard F16s V2 server that is optimized for mathematical optimization. This server has 16 vCPUs and 32 GB of memory.

4 Model Framework

In the previous chapter the thesis scope, objective and proposed methodology have been presented. In this chapter the framework for the decision support system is explained. This work is inspired and based on the work done by Vink et al. (2019). The model framework developed in that research is modified and extended with the classifier. In the system, four phases can be identified. In the pre-processing phase, the input data and disruptions are loaded and user-specified settings are defined. From the data, a feature space is generated and send to the classifier. The data and settings are send to the disruption solver. The classifier scores every aircraft in the network based on its features and sends a subset suggestion to the Disruption Solver. The disruption solver filters the schedule, network subset and disruptions, based on the aircraft family and solves for the disruptions. The solution is send to the post-processing phase where it is transformed into a format that is readable by the decision maker. A flowchart of the model framework is presented in Figure 4.1.

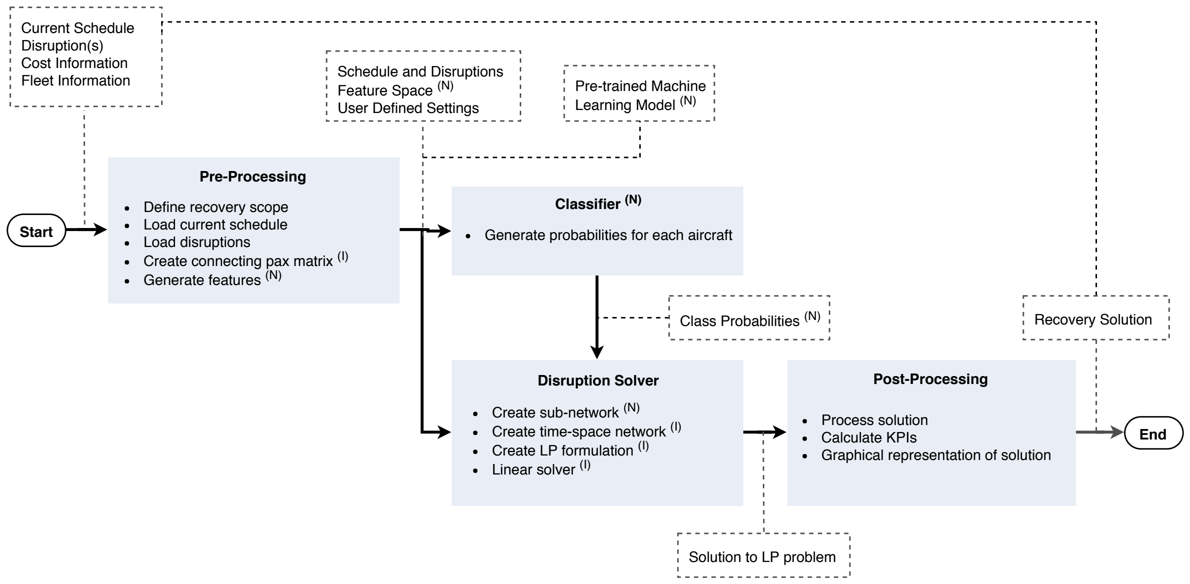


Figure 4.1: Flowchart of Model. Adapted from Vink et al. (2019). (N) Indicates new blocks developed for this research. (I) Indicates blocks developed by Vink et al. (2019), that were improved.

In the following sections, the steps taken in the four phases of the model will be explained in detail. In Section 4.1 the pre-processing phase is further explained. Section 4.2 discusses the classifier and Section 4.3 discusses the disruption solver. Finally, Section 4.4 elaborates on the post-processing phase.

4.1. Pre-processing

The pre-processing phase can be broken down into several parts: loading data sets, defining settings and feature space generation. A schematic representation of the pre-processing phase is given in Figure 4.2.

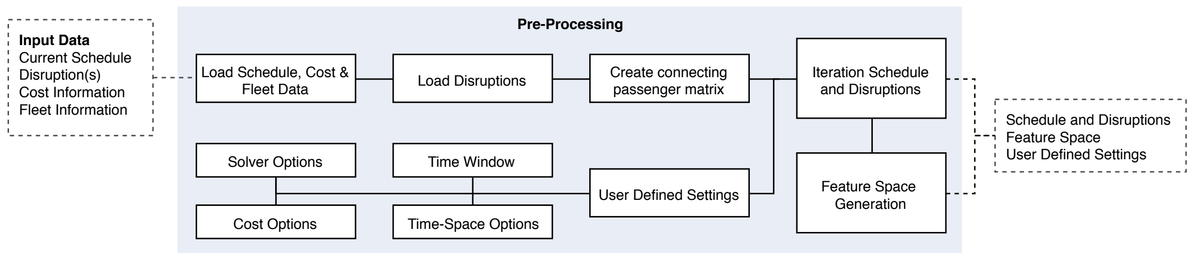


Figure 4.2: Flowchart of Pre-processing

Loading data sets

The decision support system requires several types of input data. A brief overview of the different types of input data is given here, while an more extensive overview can be found in Appendix B.

- **Fleet Information**
  - Aircraft (Tail Numbers) in the fleet
  - Direct Operating Cost (DOC) per aircraft type
  - Turn-around-Time (TAT) per aircraft type
  - Range per aircraft type
  - Passenger capacity in Economy and Business per aircraft type
- **Schedule Information**
  - Flight Schedule
  - Distances between all airports in the schedule
  - Connecting passenger itineraries
  - Minimum required time for passenger to transfer flights for all airports
- **Delay cost Information**
  - Cost of delay per minute of delay

Loading disruptions

Three different disruption types can be provided to the aircraft recovery model. For each disruption the following information is required:

- Disruption type: Flight delay, Aircraft Unavailability or Airport Closure
- Affected Flight, Aircraft or Airport
- Delay in minutes
- Time Found Out (TFO): the time at which the disruption becomes known at the AOCC

In case of a flight delay, the Scheduled Time of Departure (STD) of the flight is delayed. Since changing the cruise speed of the aircraft during the flight is not one of the options for recovery, the Scheduled Time of Arrival (STA) is delayed as much as the STD. An aircraft unavailability defines the time for which an aircraft is grounded and cannot be operated, for example due to unplanned maintenance after a bird strike. Finally, airport unavailability disruptions can be used to model airport closures, for example due to bad weather conditions.

Connecting passenger matrix

The research done by Vink et al. (2019) aimed to include the effects of passengers who miss their connection in the ARP. Based on the work done by Bratu and Barnhart (2006), Vink et al. (2019) decided to implement a *one-sided connecting passenger recovery*, where only the STA of the inbound flight will be changed and not the STD of the outbound flight.

In the approach described by Vink et al. (2019), *”connecting passengers are grouped in sets of passengers with the same itinerary. For all these groups, the effect of delay of the inbound flight is determined.* The determination of the cost of missed connections is illustrated in Figure 4.3, which depicts a time-space network (further elaborated on in Section 4.3). *In the example Flight 1 from BTV to DCA carries passengers that transfer at DCA to their final destination PHL. The scheduled connecting flight for these passengers is Flight 2. This flight departs 40 minutes after the arrival of Flight 1. With a minimum connection time of 30 minutes at DCA, a delay of Flight 1 of more than 10 minutes will cause the connecting passengers to miss their flight. If the connecting passengers miss their scheduled connection, there is an alternative flight that departs 80 minutes later, Flight 4. As a result, if Flight 1 is delayed by more than 10 minutes, the delay for the connecting passengers is 80 minutes (at their destination). Should Flight 1 be delayed by 100 minutes or more, then Flight 4 is missed as well. This is the last flight of the day, so if this flight is missed the maximum delay is assigned.”*

”Using the presented approach, all flights that have connecting passengers are evaluated. For these flights, for every time step of delay it is determined what the delay is at the end-destination of the connecting passenger. This results in the Connecting Passenger Matrix (CPM), which contains the delay cost for all connecting passengers. The CPM is a matrix of size $F \times T$ were F is the number of flights in the schedule, and T the number of delay steps the airline wishes to consider.”

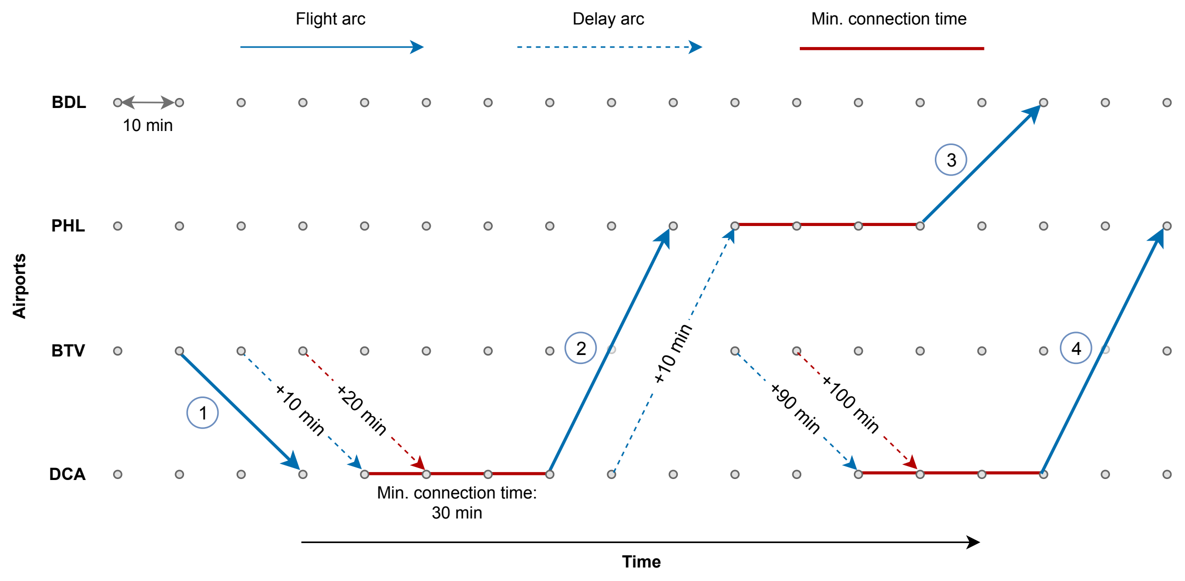


Figure 4.3: Connecting flights example. Passengers on Flight 1 need to connect to PHL at DCA. Flight 2 is the scheduled connection, Flight 4 is an alternative connection. Some passengers need to connect from Flight 2 to Flight 3.

The approach considers itineraries with a maximum of two flight legs. Itineraries with more than two flight legs, can be broken down into several two-flight itineraries. This limitation is necessary since the additional delay cost of one flight is calculated with respect to the departure of another flight and the connecting passengers between those flights.

This research extends the work by Vink et al. (2019) by allowing changes in the STD of the outbound flights if the inbound flight is disrupted. Continuing with the previous example, if Flight 1 is delayed by 20 minutes, the connecting passengers will miss Flight 2. However, it can be seen in the schedule that Flight 2 can be delayed by 10 minutes without any problems. Even if there are passengers that are transferring from Flight 2 to Flight 3 at PHL, these passengers will make their connection if Flight 2 is delayed by 10 minutes. This delay will ensure that the connecting passengers from Flight 1 will make their connection to Flight 2. It should be noted that this option is currently only available for the outbound connecting flights of a disrupted inbound flight. In the example, Flight 1 is disrupted and delayed by 20 minutes. There are connecting passengers on Flight 1, so the decision support system will incentivize the delay of the outbound connecting flights of those passengers (Flight 2). Subsequent delay options will not be taken into account, i.e. delay Flight 3 if passengers from Flight 2 were to miss their connection. Considering all subsequent delay options, would exponentially increase the problem size and run-time, therefore this research limits the added functionality to the outbound flights following an disrupted inbound flight.

As was the case by the approach proposed in Vink et al. (2019), *"it could occur that more passengers are rebooked to a flight than the number of seats still available on that flight. The reason for this limitation is that all flights with connecting passengers are assessed individually and the delay of one aircraft is independent of the delay of another. In such a case the cost of the missed connection are under-estimated by the recovery model. However, after a recovery solution has been found, such a situation can be identified so that the decision maker is made aware that additional action is required to re-accommodate passengers. For the purpose of the aircraft recovery model with which this research is concerned, the implications of this limitation are small. The goal of the proposed modelling approach is to penalize delaying flights that cause missed connections, which is still done in this example of conflicting connecting flights."*

User defined settings

There are several settings and options that change the behaviour of the decision support system. The settings can be divided into three categories: *airline preferences*, *cost factors* and *optimizer settings*. Each of the settings will be discussed in this section.

Airline preferences

Vink et al. (2019) defined several user settings which determine the boundaries within which a recovery solution is found. The system uses a *time window*, which defines the range of time that is examined during the recovery. All flights that arrive or depart within the time window are considered during optimization. By using a time window the problem size is reduced which decreases computation time. Moreover, the time window defines an end time, after which flights will no longer be affected by the disruption. This limits the effect of the disruption. The concept of a time window is illustrated in Figure 4.4.

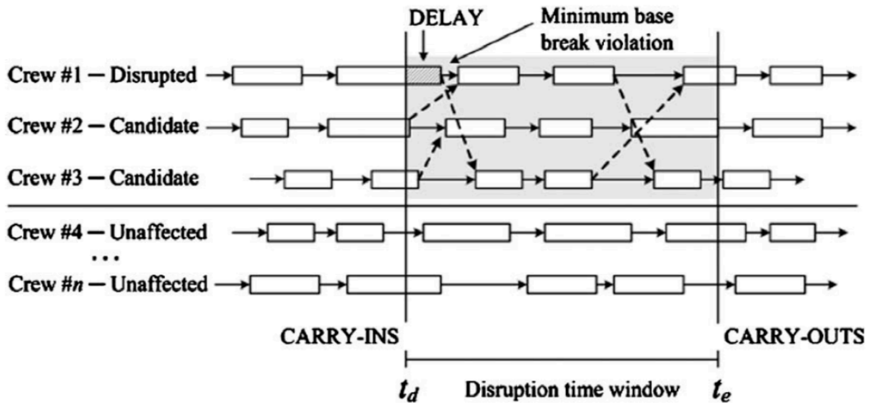


Figure 4.4: Example for crew recovery illustrating the concept of a time window for recovery. Adopted from Clausen et al. (2010)

The length of the time window is dependent on the type of flights under consideration. Since long-haul flights have longer flights times, less of these flights will be included in one day of operations. As a result, the time window can be longer without increasing the problem size and computation time. There are three options regarding the time window setting in the decision support system. The first is a fixed time window length starting at the first disruption and ending TW_{length} hours later. The second option is to always have the time window end at a fixed time, for example 06:00. If a disruption happens in the beginning of the day, the time window will be longer than if a disruption happens later in the day. Airlines may choose this option if they which to start each day with a clean slate.

Since a time window is employed, flights beyond the time window are not included in the model. The solution provided by the model could change flights in such a way that flights beyond the time window cannot be flown. To prevent this, constraints are added to the model, as discussed in Chapter 5, which ensure that a sufficient number of aircraft are available at all airports at the end of the time window. This ensures that all flights beyond the time window can be operated as scheduled. The airline can choose how aircraft are constrained. Firstly, a *specific aircraft* can be constrained. For example, if Aircraft N234DL is scheduled to end at ATL by the end of the time window, the model will ensure that this specific aircraft ends at ATL. Alternatively, the *aircraft type* can be constrained. In this case, the model will ensure that an aircraft of the same type as aircraft N234DL will end at ATL. This ensures that all booked passengers can be accommodated and that crew is available to execute the flight.

The decision support system discretizes time to limit the required computation time. The *time step* setting defines the interval of this discretization. With a time step of 10 minutes, all times in the model (e.g. departure and arrival times) are rounded to the nearest 10 minutes. The time step parameter should be chosen such that it balances the level of detail in the recovery solution with the computation time required. The granularity of the time step is discussed in Section 7.2.

One of the recovery options available to the system is to delay flights in the schedule. However, different airlines may have different preferences regarding the maximum delay. The *maximum delay* parameter defines by how many hours a flight can be delayed. A longer maximum delay leads to a larger model, while a shorter maximum delay could lead to more flight cancellations.

In addition to the parameters defined by Vink et al. (2019), this research includes the *tail swap time limit* setting. There are several ground processes surrounding aircraft turn-around. Ground operations need to be scheduled, passengers need to be notified of their gate, etc. Therefore, swapping tails in the last minute before departure is not feasible. With this parameter, airlines can fix a time block before departure in which an aircraft is not eligible for swapping.

Cost factors

The implemented linear solver has the objective to minimize cost, therefore several aspects of the recovery problem need to be expressed in monetary units. Section 5.1 explains the objective function of the solver and a summary of the cost settings is given in Table 4.1.

Depending on local legislation, airlines could be faced with additional cost when cancelling flights. The *cancellation fee* should be set to accurately represent these cost. The higher the cancellation fee, the more the model will attempt to prevent cancellations. If, to the airline, a business passenger is more valuable than an economy passenger, the *business multiplier* can be set as a multiple of the cost for economy passengers.

Thengvall et al. (2000) noted that recovery solutions should affect the smallest number of aircraft as possible. There exists a trade-off between the delay that can be prevented by a tail swap, and the effect of changing aircraft routings. The *schedule penalty cost* can be used as a threshold up to which a tail swap is not performed. Ideally, this cost factor should represent the actual cost associated with a tail swap, e.g. changing crew routings and ground operations. Alternatively, the cost factor can be based on the delay time up to which a tail swap should not be performed. For example, if an airline has the policy to only perform tail swaps for flights that are delayed by more than one hour, the penalty should be set on the average cost of delaying a flight for 60 minutes. Either way, this cost factor should not be set to 0, since that could result in random tail swaps.

As will be discussed in Section 5.2, slack variables are added to several constraints. These variables ensure feasibility of the model by assigning penalties if the constraint is violated. The magnitude of the penalty is equal to the *Big M constraint violation* cost factor. The Big M method, which is also used by Maher (2015), is further explained in Chapter 5.

Table 4.1: Cost factor user settings

Cost factor	Description
Cancellation fee	Additional cost per passenger in case a flight is cancelled.
Business multiplier	For business passengers, the delay and cancellation cost is calculated by multiplying the delay for an economy passenger by this factor.
Schedule penalty	Penalty assigned if, in the recovery solution, another aircraft than scheduled is used.
Big M constraint violation	If constraints are violated in the recovery solution, this penalty is assigned. See Chapter 5

Optimizer settings

There are several parameters that change the behaviour of the implemented linear solver, IBM ILOG CPLEX Optimization Studio 12.8.00 through the Python 2.7 API. These parameters will influence the computational performance and solution quality of the solver.

The linear solver attempts to solve the problem to an optimum solution. Optimality is relative to whatever tolerance and criteria the user has set. One of these tolerances is the *Mixed Integer Programming (MIP) gap tolerance*. The default value indicates to CPLEX to stop when an integer feasible solution has been proved to be within 0.01% of optimality. On difficult problems, where finding a proven optimum may lead to high computation time, the AOCC controller might choose a larger MIP gap to allow early termination.

The linear solver knows different phases. During pre-processing, CPLEX applies a *presolver and aggregator* several times to reduce the problem size of the MIP. This strengthens the root relaxation and decreases the problem size before it is passed to the optimizer. It may occur that, after a while, the presolver and aggregator take more time to reduce the problem than it would have taken the optimizer to solver the larger problem. To mitigate this, the controller can set a limit on the number of presolve passes made during pre-processing.

CPLEX offers several different *LP optimizers* for linear programming problems, e.g. Primal Simplex, Dual Simplex, Barrier and Concurrent, which starts the previous optimizers on different CPU cores. Depending on the problem characteristics, an optimizer may be better suited to solve the problem. The CPLEX Knowledge Base[1] offers extensive documentation on the different optimizers and the problems they are suited for. For the purpose of this thesis the Concurrent option was used, since this resulted in the lowest average solve time.

[1]https://www.ibm.com/support/knowledgecenter/en

Feature generation

After the data has been loaded, the Connecting Passenger Matrix has been created and the User-settings have been defined, the flight schedule for the iteration can be created. From this iteration schedule and the disruption(s) for the iteration, a feature space needs to be created to feed into the classifier. Chapter 6 will elaborate on the concepts of machine learning and features, however this section will describe the feature space layout.

Table 4.2 shows the layout of the feature space that needs to be created. C_m denotes the candidate aircraft for the disrupted aircraft D_p. Each C_mD_p combination forms a row in the feature space. The columns are reserved for the features F_n that describe the C_mD_p combination. The values are denoted $v_{m,p,n}$ and can be categorical, e.g. aircraft type, or numerical data, e.g. passenger capacity. Table 4.3 shows the layout of the feature space after classification. The last column in the feature space shows the labels $y_{m,p}$. These are not generated in this phase, but are the result of the classifier. These values denote the probability of candidate aircraft C_m having a positive impact on the recovery solution for disruption of aircraft D_p.

Table 4.2: Layout of the Feature Space before classification

	F_1	F_2	$\dots$	F_n
C_1D_1	$v_{1,1,1}$	$v_{1,1,2}$	$\cdots$	$v_{1,1,n}$
C_2C_1	$v_{2,1,1}$	$v_{2,1,2}$	$\cdots$	$v_{2,1,n}$
C_3C_1	$v_{3,1,1}$	$v_{3,1,2}$	$\cdots$	$v_{3,1,n}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
C_1C_2	$v_{1,2,1}$	$v_{1,2,2}$	$\cdots$	$v_{1,2,n}$
C_2C_2	$v_{2,2,1}$	$v_{2,2,2}$	$\cdots$	$v_{2,2,n}$
$\vdots$		$\vdots$	$\ddots$	$\vdots$
C_mD_p	$v_{m,p,1}$	$v_{m,p,2}$	$\cdots$	$v_{m,p,n}$

Table 4.3: Layout of the Feature Space after classification

	F_1	F_2	$\dots$	F_n	L
C_1D_1	$v_{1,1,1}$	$v_{1,1,2}$	$\cdots$	$v_{1,1,n}$	$y_{1,1}$
C_2C_1	$v_{2,1,1}$	$v_{2,1,2}$	$\cdots$	$v_{2,1,n}$	$y_{2,1}$
C_3C_1	$v_{3,1,1}$	$v_{3,1,2}$	$\cdots$	$v_{3,1,n}$	$y_{3,1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\dots$	$\vdots$
C_1C_2	$v_{1,2,1}$	$v_{1,2,2}$	$\cdots$	$v_{1,2,n}$	$y_{1,2}$
C_2C_2	$v_{2,2,1}$	$v_{2,2,2}$	$\cdots$	$v_{2,2,n}$	$y_{2,2}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\dots$	$\vdots$
C_mD_p	$v_{m,p,1}$	$v_{m,p,2}$	$\cdots$	$v_{m,p,n}$	$y_{m,p}$

4.2. Classifier

The created feature space is passed on to the machine learning classifier which will calculate class probabilities for all the candidate aircraft in the network of the airline using a pre-trained model. A schematic representation of the classifier phase is given in Figure 4.5.

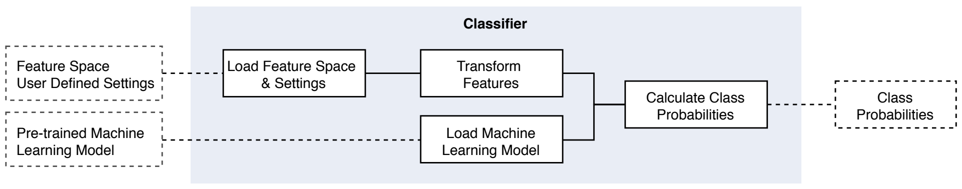

Classifier
Feature Space User Defined Settings
Pre-trained Machine Learning Model
Load Feature Space & Settings
Transform Features
Load Machine Learning Model
Calculate Class Probabilities
Class Probabilities

Figure 4.5: Flowchart of Classifier

Transform features

The feature space generated by the pre-processing phase is not always usable by the machine learning model. Most machine learning algorithms require all features to be in numerical form opposed to categorical form. To transform all features in the correct format, one-hot encoding will be used, which will be discussed in Chapter 6. Furthermore, missing values will be replaced by an appropriate number. Section 6.2.4 will further elaborate on the pre-processing of data for machine learning.

Calculate class probabilities

The transformed feature space will be processed by a pre-trained machine learning model to generate class probabilities. For the purpose of the decision support system the classifier will produce probabilities per candidate aircraft to indicate the probability that the aircraft will have a positive impact on solving the disruption and thus the solution quality. The quality of the probabilities will highly depend on the quality and predictive power of the machine learning model. Chapter 6 will discuss the machine learning framework that was used to train the classification model.

4.3. Disruption Solver

After the classifier has assigned class probabilities to all the candidate aircraft in the network, the ARP is solved by the disruption solver. In this section the functionality of the disruption solver is described. A schematic representation of the disruption solver phase is given in Figure 4.6.

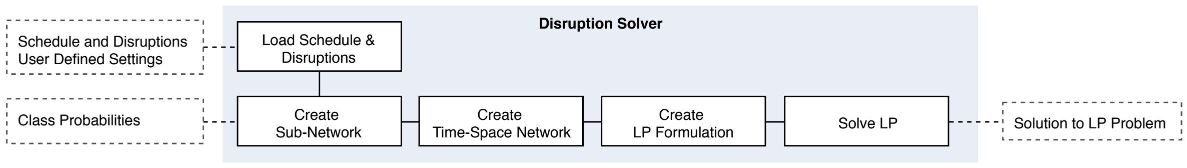


Figure 4.6: Flowchart of Disruption Solver

Three important steps can be distinguished in the disruption solver. First, a sub-network is created based on the class probabilities from the classifier. Second, the problem is translated into a time-space representation and LP formulation. Third, the linear solver attempts to find a solution to the problem. When a feasible solution is found, the solution is send to post-processing. Since the optimization model uses Big M variables, further elaborated upon in Chapter 5, the model never actually becomes infeasible.

Sub-network selection

Based on the class probabilities produced by the classifier, a sub-network needs to be created. The goal of this step is to reduce the problem size in order to increase computational performance of the decision support system. Therefore, the sub-network needs to consist of the lowest number of aircraft while maintaining feasibility and without significantly deteriorating the solution quality. There are several strategies for selecting the aircraft in the sub-network:

- **Top X% aircraft** This strategy selects the X% aircraft with the highest probability of helping solve the solution, e.g. top 15% of aircraft).
- **All aircraft above threshold** This strategy selects all aircraft that have a probability above a threshold X. With this strategy it could be hard to control the problem size, since it is hard to predict the number of aircraft in sub-network.

For each of these strategies, the value of X needs to be tested, which will be discussed in the sensitivity analysis in Section 8.3.

Time-space network

As can be determined from an overview of aircraft recovery methods discussed by Clausen et al. (2010) and Hassan et al. (2019), the time-space network is the most used network representation for such problem. As explained by Vink et al. (2019), *"a series of parallel time-space networks is required to be able to distinguish between individual aircraft. As a result, a separate time-space network is created for each aircraft in the sub-network."* An example of such a time-space network is given in Figure 4.7. In this figure, each node represents an airport at a point in time. Nodes are created for all airports in the network and at every time step t_s in the time window. Different *arcs* are used to model aircraft movements as time advances. *Flight arcs* represent flights from one airport to another. The origin node of the arc represents the origin airport at the departure time and the terminating node represents the destination airport and the arrival time. *Delay arcs* are used to model delayed flights. Delay arcs are created for each delay time step t. *Ground arcs* represent the ground time spend at an airport, where the aircraft stays on the same airport as time progresses.

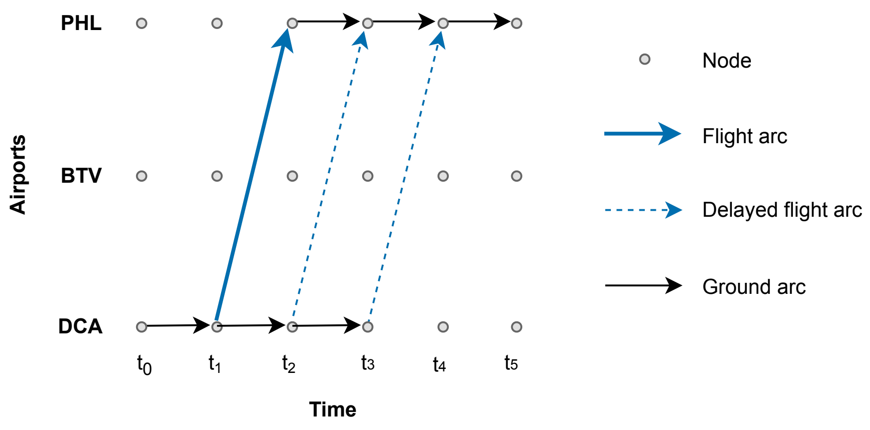


Figure 4.7: Ground, flight and delay arcs in time-space network.

LP formulation and linear solver

Based on the time-space network, the objective function, decision variables and constraints will be formulated for the CPLEX linear solver. The formulation will be elaborated on in Chapter 5. The linear solver will use this LP formulation and the user defined settings to attempt to find a feasible solution to the problem. On a high level, the linear solver will go through the following phases:

1. **Pre-processing** During this phase the goal is to improve the formulation provided to CPLEX and to reduce the problem size. This is achieved by identifying in-feasibility and/or redundancy in the model. Furthermore, using probing techniques, binary variables will be fixed and logical implications will be checked.

2. **Solve the root relaxation problem** After the re-formulation of the problem, the linear solver will remove all integrality restrictions, resulting in the linear-programming relaxation of the MIP. Unlike the original problem formulation, the relaxed LP problem can be solved in polynomial time. The solution is used to gain information about the original mixed-integer problem.

3. **Branch-and-Bound** Usually, the solution of the LP relaxation does not satisfy the integrality restrictions. During the Branch-and-Bound (B&B) phase, the linear solver picks one relaxed variable and restricts it to be integer, while its LP relaxation value is fractional. For example, variable x has a LP relaxation value of 8.7. The B&B algorithm will impose the restrictions $x \leq 8.0$ and $x \geq 9.0$. This creates two new sub-problems. This process is repeated until a feasible solution has been found or all variables have been exhausted.

4.4. Post-processing

A schematic representation of the post-processing phase is given in Figure 4.8. This phase is largely unchanged from the phase proposed by Vink et al. (2019).

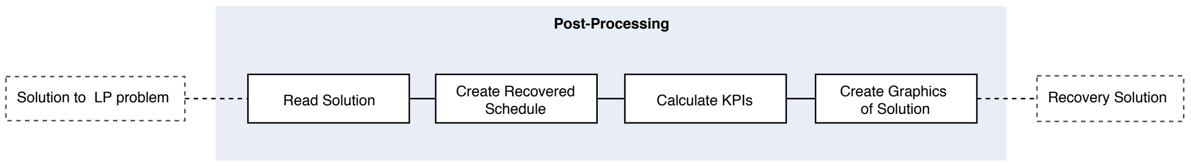


Figure 4.8: Flowchart of Post-processing

In the post-processing phase the LP solution found by the system is processed to an understandable suggested recovery solution and flight schedule. Key Performance Indicators (KPIs) summarize the recovery solution and allow an AOCC operator to determine the quality of the solution at a glance.

Vink et al. (2019) developed the KPIs presented in Table 4.4. The On-time performance (OTP) KPI describes the punctuality of airlines and is widely used in the airline industry. The Delayed Passenger Count (DPC) and Average Passenger Delay (APD) quantify the severity of delays. Flight cancellations disrupt (connecting) passenger itineraries. The number of Missed Connection Passengers (MCP), Cancelled Passengers (XP) and Cancelled Flights (XF) serve as an indicator for disrupted passengers. The number of Constraint Violations (CV) represent the constraints in the optimization model that could not be satisfied. This indicates that flight schedule after the window cannot be flown as scheduled, e.g. due to an aircraft missing at an airport at the end of the time window. The cost of the disruption KPI summarizes overall solution quality.

Table 4.4: Key Performance Indicators for recovery solutions

KPI	**Description**	**Unit**
OTP	% of flights operated with <15 minutes of delay	[%]
DPC	Number of delayed passengers	[#]
APD	Average delay per delayed passenger in minutes	[min/pax]
XP	Number of passengers on cancelled flights	[#]
XF	Number of cancelled flights	[#]
MCP	Number of passengers that missed a connection	[#]
CV	Number of constraints violated in recovery solution	[#]
Cost	Cost of the disruption	[$]

The last step of the post-processing phase is to automatically update the current schedule with the recovery solution. This ensures that the recovery solution is taken into account when a new disruption needs to be solved.

5 Optimization Model

Using the inputs discussed in Section 4.3, the optimization model attempts to find the optimal (minimum cost) solution to the aircraft recovery problem. This chapter will elaborate on the model used to find this optimal solution. The mathematical formulation of the optimization model is also summarized in Appendix D.

The following sets, indices and parameters will be used by the solver and will be used throughout this chapter:

Sets

F	set of flights
A	set of airports
E	set of aircraft types
P	set of aircraft
P(e)	set of aircraft *p* of type *e*
N	set of all nodes = $N_O \cup N_I \cup N_G \cup N_S$
N_O	set of origin nodes
N_I	set of intermediate nodes
N_S	set of sink nodes
T	set of delay steps
S	set of slack variables

Indices

i	flight index
t	delay time index
a	airport index
p	aircraft index
e	aircraft type index
n	node index
j	artificial variable index

Parameters

β_{bus}	Cost multiplier for business passengers
C_{canx}	Additional hard cost per passenger in case of cancellation
$C_{conn_{i,t}}$	Delay cost for connecting passengers on flight i, for delay time step t
$C_{D_{i,t}}$	Cost of delay for flight i, for delay time step t
C_{DS_t}	Soft cost of delay for delay time step t
C_{DH_t}	Hard cost of delay for delay time step t
$C_{OP_{p,i}}$	Operating cost of flight i with aircraft p
C_{C_i}	Cancellation cost of flight i
C_{G_n}	Cost of operating ground arc originating from node n
$C_{C_{SCH}}$	Penalty for operating a different aircraft than scheduled
C_{DOC_p}	Direct operating cost of aircraft p, per block hour
dist_i	Distance of flight i
orig_i	Origin airport of flight i
dest_i	Destination airport of flight i
STA_i	Scheduled Time of Arrival of flight i
STD_i	Scheduled Time of Departure of flight i

FT_i	Flight time of flight i, in hours
PaxY_i	Economy passengers on flight i
PaxJ_i	Business passengers on flight i
range_p	Range of aircraft p
seatsY_p	Economy seats on aircraft p
seatsJ_p	Business seats on aircraft p
h_n^e	Number of aircraft belonging to fleet e that should terminate at sink node $n \in \mathrm{N}_S$
TW	Number of time steps in time window
M	Big M cost factor
T_{swap}	Tail swap limit in minutes

5.1. Objective Function

The objective value function of the optimization model is given in Equation 5.1. The objective of the model is to minimize the sum of the following cost:

- Flights operated as scheduled - direct operating costs (DOC) only
- Flights operated with a delay - DOC and cost of delay
- Cancelled flights - cancellation cost
- Ground arc cost - cost of operating a certain ground arc
- Schedule consistency cost - additional cost for operating a different aircraft
- Big M cost - costs that indicate infeasibility of the problem

$$\begin{aligned} min \sum_{p\in P}\sum_{i\in F} \delta_{F_{p,i}} \cdot C_{OP_{p,i}} + \sum_{p\in P}\sum_{i\in F}\sum_{t\in T} \delta_{D_{p,i,t}} \cdot (C_{OP_{p,i}} + C_{D_{i,t}}) + \sum_{i\in F} \delta_{C_i} \cdot C_{C_i} \\ + \sum_{p\in P}\sum_{n\in N} \delta_{G_{p,n}} \cdot C_{G_n} + \sum_{i\in F} \delta_{F'_i} \cdot C_{C_{SCH}} + \sum_{j\in S} s_j \cdot M \end{aligned} \tag{5.1}$$

5.1.1. Decision variables

All decisions the model can make are represented by decision variables (DVs), shown below. All decision variables are binary, unless stated otherwise. The decision variables are denoted by a δ, or, in case of the slack variable, by a s.

$\delta_{F_{p,i}}$	= 1, if flight arc *i* is flown by aircraft *p* without delay
$\delta_{D_{p,i,t}}$	= 1, if flight arc *i* is flown by aircraft *p* with delay time step *t*
δ_{C_i}	= 1, if flight *i* is cancelled
$\delta_{G_{p,n}}$	= 1, if aircraft *p* uses ground arc originating from node *n*
$\delta_{F'_i}$	= 0, if flight *i* is flown by same aircraft *p* as scheduled
s_j	= 1, if the problem is infeasible, one or more slack variables are part of the basic solution

The $\delta_{F_{p,i}}$ define whether a flight i is flown by aircraft p without delay. For all flight-aircraft combinations, a decision variable is created. The cost for the decision variable is defined by the DOC of the aircraft and the flight duration.

The $\delta_{D_{p,i,t}}$ variables define whether a flight i is flown by aircraft p with delay t. Similar to the previous decision variable, a DV is created for all flight-aircraft-delay combination, where the delay is discretized to time steps, e.g. 10 minutes. Therefore, if the maximum allowed delay is 8 hours, 48 decision variables are created for all flight-aircraft combinations. The cost associated with these DVs equal the direct operating cost and a delay cost factor that depends on the delay duration.

Flight cancellations are defined by the δ_{C_i} variables. The cost of cancelling a specific flight are associated with the variable for that flight. Finally, the $\delta_{G_{p,n}}$ defined whether an aircraft uses the ground arc that originates in node n.

If the model assigns a different aircraft to a flight than originally scheduled, it is indicated by the $\delta_{F'_i}$ variables. If required by the airline, this may impose a penalty (cost).

The s_j slack variables are artificial variables needed for the Big M method as discussed by Hillier and Lieberman (2015). By assigning a high cost to these decision variables (M for Million), the model will only make these variables part of the solution if and only if the problem is otherwise infeasible. Since not all constraints may cause infeasibilities, Section 5.2 elaborates on which constraints contain the slack variables and the applicability to that constraint.

5.1.2. Cost factors

The different cost factors shown in the objective function will be discussed in this section.

The cost of flying a flight with an aircraft, $C_{OP_{p,i}}$, equals the direct operating cost of the aircraft (per minute) DOC_p multiplied with the flight time FT_i.

$$C_{OP_{p,i}} = C_{DOC_p} \cdot FT_i \tag{5.2}$$

The cost of flying a delayed flight equals the sum of the cost of flying the aircraft with the assigned aircraft as given in Equation 5.2, and the delay cost $C_{D_{i,t}}$. This delay cost depends on the number of economy and business passengers booked on the flight, and the delay time step. β_{bus} is the business multiplier that is used, since business passengers are more expensive to delay. C_{DS_t} equals the soft cost and C_{DH_t} equals the hard cost for the delay. If the flight contains passengers that are connecting to another flight at the destination, and the delay causes them to miss their transfer, an additional cost factor $C_{conn_{i,t}}$ is assigned as well. This cost factor is obtained from the Connecting Passenger Matrix (CPM).

$$C_{D_{i,t}} = C_{conn_{i,t}} + (C_{DS_t} + C_{DH_t}) \cdot (PaxY_i + \beta_{bus} \cdot PaxJ_i) \tag{5.3}$$

When a flight is cancelled, the maximum soft and hard delay cost will be incurred. Furthermore, a cancellation penalty is included, which represents the additional cost related with an cancellation, e.g. hotel accommodation.

$$C_{C_i} = (C_{DS_{max}} + C_{DH_{max}} + C_{canx}) \cdot (PaxY_i + \beta_{bus} \cdot PaxJ_i) \tag{5.4}$$

It should be noted that the current implementation of the cancellation cost may cause cancellations of flights with a low passenger load factor, since it may happen that the cost of cancelling those flights is lower than the actual direct operating cost of the flight. Even though airlines are pushing for high load factor flights, it is assumed that all scheduled flights should be flown, since other considerations could play are role. To avoid these cancellations, it should be ensured that the cost of cancelling a flight is always higher than the cost of operating that flight.

Ground fees imposed by airports, e.g. for parking, can be included in the model through the cost of operating a ground arc, C_{G_n}.

Since airlines have a preference to minimize the number of deviations from the original schedule, a penalty $C_{C_{SCH}}$, is incurred when the original scheduled aircraft is not assigned to a flight. By doing so, the model will prefer assigning the scheduled aircraft. This penalty should be chosen such that it equals the cost of a tail swap.

The cost associated with the artificial variables is denoted M, and usually equals a large cost (millions). This cost is included in the objective value if all other solutions lead to infeasibilties. The magnitude of M should be large enough to ensure the model will not make it part of the solution in any other case than to prevent infeasibilities.

5.2. Constraints

The objective function of the previous section is subjected to multiple constraints, which will be discussed next. The constraints can be divided into several groups: *time-space continuity constraints*, *airline constraints* and *disruption constraints*.

5.2.1. Time-space continuity constraints

The constraints in this section are necessary when working with a time-space network. Even when using a time-space network for another problem type, these constraints will be necessary in some form.

Flight coverage
A trivial low-cost solution to the aircraft recovery problem would be to ground all aircraft. To prevent this, the flight coverage constraint, presented in Equation 5.5, ensures that all flights in the schedule are either flown as scheduled, delayed or cancelled. This constraint ensures that the lowest cost solution is to operate all flights as scheduled, although this may not always be possible due to disruptions.

$$\sum_{p \in P} \left[\delta_{F_{p,i}} + \sum_{t \in T} \delta_{D_{p,i,t}} \right] + \delta_{C_i} = 1 \qquad \forall i \in F \tag{5.5}$$

Node continuity
In the time-space network, aircraft are constrained by following a path between nodes. The node continuity constraints ensure that all aircraft entering a node through flight or ground arcs also leave the node. Put differently, all aircraft that arrive at an airport must either stay on the airport or depart from the airport. The constraint in Equation 5.6 covers most of the nodes in the network. However, the nodes at the start at the time window are covered by the constraint in Equation 5.7.

$$\left[\delta_{G_{p,n-1}} + \sum_{i \in F_{in}} \delta_{F_{p,i}} + \sum_{i \in F_{in}, t \in T} \delta_{D_{p,i,t}} \right] - \left[\delta_{G_{p,n}} + \sum_{i \in F_{out}} \delta_{F_{p,i}} + \sum_{i \in F_{out}, t \in T} \delta_{D_{p,i,t}} \right] = 0 \qquad \forall p \in P, n \in N_I \tag{5.6}$$

$$\delta_{G_{p,n}} + \sum_{i \in F_{out}} \delta_{F_{p,i}} + \sum_{i \in F_{out}, t \in T} \delta_{D_{p,i,t}} = 1 \qquad \forall p \in P, n = \text{scheduled } N_O \text{ of } p \tag{5.7}$$

Like the nodes at the beginning of the time window, the nodes at the end of the time window, the sink nodes, are covered by a different constraint. Since only the aircraft and flights within the time window can be changed, by the end of the time window the flight schedule should be operated as scheduled. Airlines have the option to either fix specific aircraft by the end of the time window, e.g. if an aircraft was scheduled to be located at ATL at the end of the time window, that specific aircraft will be constrained to be there, this constraint is given in Equation 5.8. Alternatively, airlines can choose to fix the aircraft type instead of the specific aircraft. This option gives more flexibility to the model, since tail swaps within aircraft types are allowed. The formulation for this constraint is given in 5.9.

$$\delta_{G_{p,n-1}} + \sum_{i \in F_{in}} \delta_{F_{p,i}} + \sum_{i \in F_{in}, t \in T} \delta_{D_{p,i,t}} + s_j \geq 1 \qquad \forall p \in P, n = \text{scheduled } N_S \text{ of } p \tag{5.8}$$

$$\sum_{p \in P(e)} \left[\delta_{G_{p,n-1}} + \sum_{i \in F_{in}} \delta_{F_{p,i}} + \sum_{i \in F_{in}, t \in T} \delta_{D_{p,i,t}} \right] + s_j \geq h_n^e \qquad \forall e \in E, n \in N_S \tag{5.9}$$

Both sink node continuity constraints, Equation 5.8 and Equation 5.9, contain a slack variable s_j, since given a disruption, these constrains may not be satisfied. If the last flight of the day is cancelled, that aircraft will no longer reach its planned airport within the time window. Without these slack variables, the problem would be infeasible, however with this problem, the model results in a high objective value, which can be shown to the AOCC controller. The slack variables are not binary, but integer, since it could occur that multiple aircraft that were planned to end at an airport will not be able to reach the airport. For each of those aircraft M cost will be incurred.

Since the number of aircraft within the time window will not change, if an aircraft does not reach its intended end airport, there must me a surplus of aircraft at another airport. Therefor, the sink-node constraints are of the "equal or greater than" type, to allow more aircraft at the airport than planned.

5.2.2. Airline constraints

The following constraints are specific to the aircraft recovery problem and/or the preferences of airlines.

Aircraft seat capacity

Tail swaps should only be allowed if the new aircraft can transport the number of economy and business passengers on the flight. Equation 5.10 ensures that the new aircraft has the capacity to transport the passengers.

$$\delta_{F_{p,i}} + \sum_{t \in T} \delta_{D_{p,i,t}} = 0 \qquad \forall p, i \text{ where } (\text{seatsY}_p < \text{PaxY}_i \wedge \text{seatsJ}_p < \text{PaxJ}_i) \tag{5.10}$$

Aircraft range

Similar to the previous constraint, Equation 5.11 ensures that the new aircraft's range is sufficient to fly the distance of the flight leg.

$$\delta_{F_{p,i}} + \sum_{t \in T} \delta_{D_{p,i,t}} = 0 \qquad \forall p, i \text{ where } (\text{range}_p < \text{dist}_i) \tag{5.11}$$

Original schedule penalty

If a different aircraft than originally planned is scheduled on a flight in the recovery solution, Equation 5.12 ensures that $\delta_{F'_i} = 1$ and $C_{C_{SCH}}$ is included it the objective value.

$$\delta_{F_{p,i}} + \sum_{t \in T} \delta_{D_{p,i,t}} - \delta_{F'_i} = 1 \qquad \forall i \in F, p = \text{aircraft scheduled for } i \tag{5.12}$$

Tail swap time limit constraint

Due to turn-around processes planning and gate allocation schedules, aircraft cannot be tail swapped T_{swap} minutes before departure. Equation 5.13 ensures that aircraft will not be assigned to a different flight T_{swap} minutes before their scheduled time of departure (STD).

$$\delta_{F_{p,i}} + \sum_{t \in T} \delta_{D_{p,i,t}} = 0 \qquad \forall p \in P, i \text{ where } STD_i - T_{now} < T_{swap} \text{ and i} \neq \text{flight for p} \tag{5.13}$$

5.2.3. Disruption constraints

The constraints discussed so far are related to regular airline operations. If disruptions occur, these are added to the model in the form of constraints as well. This section will elaborate on the disruption constraints.

Flight delay
If flight i is delayed, the decision variable associated with flying that flight as scheduled,$\delta_{F_{p,i}}$, is set to zero for all aircraft. Furthermore, all delay arcs associated with a departure time earlier than the delay, $\delta_{D_{p,i,t}}$, are set to zero for all aircraft. The constraint is shown in Equation 5.14.

$$\sum_{p\in P} \delta_{F_{p,i}} + \delta_{D_{p,i,t}} = 0 \qquad \forall t \in T \leq delay, i = \text{delayed flight} \tag{5.14}$$

Flight cancellation
If flight i is cancelled, decision variable δ_{C_i} is set to one. Equation 5.15 works with the flight coverage constraint, Equation 5.5, to ensure all flight and delay arcs for this flight are set to zero.

$$\delta_{C_i} = 1, \qquad i = \text{cancelled flight} \tag{5.15}$$

Aircraft unavailability
Equation 5.16 ensures that, if aircraft p is unavailable for a certain time, all decision variables for the flight and delayed flight arcs of that aircraft are set to zero.

$$\begin{gathered}\delta_{F_{p,i}} + \sum_{i\in F} \delta_{D_{p,i,t}} = 0 \\ \forall i \in F \text{ where } (\mathrm{t}_{start} \leq \mathrm{STD}_i \leq \mathrm{t}_{end} \cup \mathrm{t}_{start} \leq \mathrm{STA}_i \leq \mathrm{t}_{end}), \\ \forall t \in T \text{ where } (\mathrm{t}_{start} \leq \mathrm{STD}_i + t \leq \mathrm{t}_{end} \cup \mathrm{t}_{end} \leq \mathrm{STA}_i + t \leq \mathrm{t}_{end})\end{gathered} \tag{5.16}$$

Airport unavailability
Similar to the previous constraint, Equation 5.17 ensures that, if an airport is unavailable for a certain period of time, all flight and delay arcs that are set to arrive or depart from the airport within that time, are set to zero.

$$\begin{gathered}\sum_{p\in P} (\delta_{F_{p,i}} + \sum_{i\in F} \delta_{D_{p,i,t}}) = 0 \\ \forall i \in F \text{ (where } (\mathrm{t}_{start} \leq \mathrm{STD}_i \leq \mathrm{t}_{end} \cup \mathrm{t}_{start} \leq \mathrm{STA}_i \leq \mathrm{t}_{end})) \cap (\mathrm{orig}_i \cup \mathrm{dest}_i) = \mathrm{a} \\ \forall t \in T \text{ (where } (\mathrm{t}_{start} \leq \mathrm{STD}_i + t \leq \mathrm{t}_{end} \cup \mathrm{t}_{end} \leq \mathrm{STA}_i + t \leq \mathrm{t}_{end})) \cap (\mathrm{orig}_i \cup \mathrm{dest}_i) = \mathrm{a} \\ \text{where a = unavailable airport}\end{gathered} \tag{5.17}$$

5.3. Assumptions and Implications

The aircraft recovery model proposed by Vink et al. (2019) and the ARP proposed by this research are a simplification of reality, hence some aspects of airline operations are not considered or simplified. An overview of these assumptions and their implications are given in Table 5.2. Most of the assumptions are based on the previous work by Vink et al. (2019). The validity of these assumptions is discussed in Section 7.2.

For most short to medium-haul flights, the cruise speed differences are negligible for different aircraft types. Therefore, it is assumed that the flight duration is not dependent on the scheduled aircraft. In reality, the arrival time of a flights may change by 10-20 minutes. According to Vink et al. (2019), *"This can have a small effect on the results, because the earlier or later arrival time also has an effect on subsequent flights scheduled for the aircraft."* Another benefit of this assumption, besides simplification of the model, is that only the schedule penalty parameter affects tail swaps. With differences in flight times, the model may choose to perform a tail swap to recovery some of the delay. As a result, the swapping behaviour can no longer be controlled using a single parameter.

Arıkan et al. (2016) showed that implementing a trade-off between accelerated fuel burn and reduced flight time is not straightforward due to the non-linear relationship between fuel consumption and cruise speed. Therefore, the option to recover some delay time by flying at a higher cost index is not available to the model. On long-haul flights, flying at a higher cost index could significantly decrease flight times. This could have a large effect on the cost of a disruption, especially if connections which are otherwise missed are possible again. Vink et al. (2019) presented recommendations concerning the inclusion of this aspect in the recovery model, which are repeated in Chapter 9

Explicit crew recovery is considered to be beyond the scope of this research. Therefore it is assumed that crew is always available. Recommendations about explicitly including crew in the recovery model are presented in Chapter 9.

Airport capacity and slot availability information was not available for this research, and it is assumed that flights are not constrained by reduced flow rates or airport close times. Constraining illegal flight arcs due to limited airport capacity or slot availability is straightforward. The current model could suggest a recovery solution where a flight is delayed to a time with no slot availability. This could have a large effect, since in reality this flight would be cancelled.

Section 4.1 discussed the time discretization used to reduce computation time by reducing the number of nodes in the time-space network. However, as a result arrival, departure and delay times are rounded to the nearest time step. This could lead to differences between the flight schedule as presented by the system and actual operations.

Due to limited information availability, it is assumed that the Turn-Around-Time (TAT) per aircraft type is a constant. In reality, TAT may be reduced by 5-10 minutes by assigning additional ground staff, especially on hub airports. To make this decision, a trade-off would have to be made between the additional cost for ground staff and the saved disruption cost. Therefore, ground staff cost and availability would have to be known. This aspect of the ARP is beyond the scope of this research.

For this research no information was available on aircraft maintenance schedules and resource availability. As a result, it cannot be reliably determined which maintenance tasks need to be scheduled for each aircraft. This assumption can potentially have large effects on the results. The recovery model currently does not schedule maintenance tasks, while in reality aircraft are legally required to have periodic maintenance. By not taking maintenance into account, the decision support system is less constrained than reality and the solution cost will be underestimated.

Due to limited data availability, only the results of aircraft disruptions are available. Section 8.1 elaborates on the dataset generation. Due to this limitation in disruption data, delays are not updated. For example, at time t_1 a delay disruption of 30 minutes could be known to the AOCC and at any time $> t_1$ the AOCC might find out that the flight will actually be delayed 50 minutes. Since these disruption updates are not available, only the final disruption is taken into account. The implication of this simplification is that there will be less changes to the schedule. Furthermore, in reality it may occur that due to previous disruption recovery solutions, certain recovery options become unavailable for following disruptions.

Changes in the Scheduled Time of Arrival (STA) due to weather conditions, except for disruptions, are assumed to have no effect on the flight schedule. In reality, especially on long-haul flights, having head or tail wind could significantly impact flight times. Because weather information and the effect on flights is not readily available, the decision support system currently does not take this effect into account. Furthermore, periodically determining the effect weather has on the flights in the network is computationally expensive.

Table 5.2: Assumptions and corresponding implications

	Assumption	**Implication**	**Expected effect on results**
		Assumptions adapted from Vink et al. (2019)	
1	Equal flight time for all aircraft types	The flight duration that was scheduled for a flight remains the same, even if in the recovery solution it is flown by an aircraft with a higher cruise speed.	Minimal
2	No cruise speed changes to recapture time during flight	A trade-off between delay cost and accelerated fuel burn is not made in the model. Shorter than scheduled flight times as a result of change in cruise speed are not considered.	Minimal
3	Crew is always available	Limits on flying times for crews, or the availability of crews at airports is not considered. It is assumed that there is always a crew available to perform the flights in the recovered schedule.	Large
4	Airport capacity and slots are always available	The capacity at airports and slot availability are not considered in the model. It is assumed that a flight can always be scheduled to depart or arrive at an airport at a later time than scheduled.	Large
5	Time discretization still provides sufficient detail for airline operations	Times (e.g. STD or STA) are rounded to nearest time step. Delays smaller than the time step are not considered for recovery.	Minimal
6	TAT is constant per aircraft type	Delays cannot be mitigated as a result of time recapture during TAT. The choice to assign more ground staff for shorter TAT cannot be made in the recovery model.	Minimal
		New Assumptions	
7	Maintenance is not considered	(Scheduled) maintenance tasks are not constraining the model. Unscheduled maintenance is considered as an aircraft unavailability.	Medium
8	Delays are not updated	Only initial available information about delays and delay duration is taken into account. Delay information is not updated if it becomes available.	Medium

6 Machine Learning Classifier

Section 4.2 presented a high level overview of the classifier and its place in the decision support system framework. This chapter will elaborate on machine learning based classification in general and the systematic process undertaken to create the pre-trained machine learning classifier that is used in the DSS framework.

Section 6.1 will briefly introduce machine learning and binary classification. Section 6.2 will discuss the framework and process that was followed to create the classifier. Finally, Section 6.3 will present the final classifier parameters and model analyses.

6.1. Introduction to Machine Learning and Binary Classification

Machine learning techniques are often used to create the decision function for classification. The machine learning domain can be divided into one of three categories: supervised learning, unsupervised learning and reinforcement learning.

In *supervised learning*, a set of fully labeled training data is available. Fully labeled means that each instance in the dataset comes with the label or category it belongs to. So, a labeled dataset of pictures of fruit, would tell the model which photos depict apples, bananas and oranges. When presented with a new photo, the model attempts to predict the correct label based on what it has learned from the training dataset. Hence, supervised learning is well suited for problems where a fully labeled dataset is available. However, clean and perfectly labeled data is not always available.

When a fully labeled dataset is not available, researchers may still use machine learning techniques. In *Unsupervised learning*, a model is given a training dataset without a specific label or preferred outcome. The model's task is to find structure in the data, which can be done in different ways, e.g. clustering, association, anomaly detection. Because labeled data is unavailable with unsupervised learning, it is hard to measure the performance of an algorithm.

Finally, *reinforcement learning* algorithms have agents that attempt to discover the optimal way to reach a particular goal. The agents take actions or decisions, and when these actions lead towards the set goal, the agent receives a reward. The overall aim of the model is to earn the biggest final reward. In its pursuit of reward, the model balances exploration of new tactics/decisions with exploitation, learnings from past experiences. Reinforcement learning is an iterative process where the model improves with more rounds of feedback. This class of machine learning is particularly useful for teaching robots autonomous decision making, e.g. self-driving vehicles. Especially when each iteration is computationally expensive, reinforcement learning may take a long time before producing promising results.

Since fully labeled data is available for this research and optimization is computationally expensive, the remainder of this research and thesis will focus on supervised learning.
With supervised learning, the interest lies in finding the relationship between the independent variables X and the dependent variables Y. The goal of binary or binomial classification is to separate the elements of a given dataset $\chi = (x_i, y_i)_{i=1}^{i=N}$, where x_i are features and $y_i \in 0, 1$ the corresponding binary class labels, into two groups (predicting which group each one belongs to) using a decision function. In probabilistic terms, classification is computing the posterior $p(y|x)$ and performing an operation like argmax (e.g. classifying instances by $argmax_{y\in 0,1} p(y|x)$). A typical plot of χ is given in Figure 6.1.

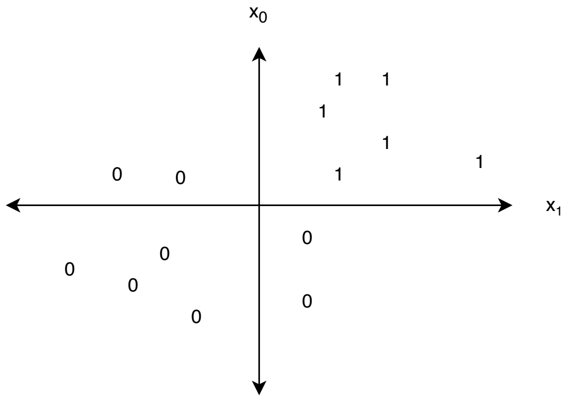


Figure 6.1: A typical plot of a data set χ with each instance $(x_i, y_i) \in X$ plotted at its features and labeled by its class.

6.1.1. The Bias-Variance Trade-off

With supervised learning, a relationship between feature(s) and labels is assumed and estimated. If the assumption is true, there must be a model f which exactly describes the relationship. In practice, this model f is often unknown and an attempt will be made to estimate it with model $\hat{f}$. Model $\hat{f}$ is obtained by training on a particular training dataset. If a different training set is used, it is likely that model $\hat{f}$ will be different as well. The amount by which $\hat{f}$ varies with different training sets is called the *variance*.

Besides often being unknown, the relationship between features and labels is complex for most applications. Assumptions and simplifications give *bias* to model $\hat{f}$. The more erroneous the simplifications and assumptions with respect to the true relationship, the higher the bias.

There exists a myriad of supervised learning algorithms with the aim of finding an $\hat{f}$. For all algorithms, the expected error on an unseen instance X was derived by Hastie et al. (2001) and will be:

$$\begin{aligned} E[(y - \hat{f}(x))^2] &= (Bias[\hat{f}(x)])^2 + Var[\hat{f}(x)] + \sigma^2 \quad \text{with,} \\ Bias[\hat{f}(x)] &= E[\hat{f}(x)] - f(x) \\ Var[\hat{f}(x)] &= E[\hat{f}(x)^2] - E[\hat{f}(x)]^2 \end{aligned} \tag{6.1}$$

For an expected error, it is clear that there is a trade-off between bias and variance. A choice has to be made between lower complexity models, with low variance but high bias, and higher complexity models, with low bias and higher variance.

6.1.2. Learning Curves and Bayes error rate

The previous section presented the basic concepts of bias and variance. This section will discuss how these concepts can be used to determine model performance and, if necessary, how to improve classification.

Given the explanation of bias and variance, it should be clear to the reader that the performance of the model on the training data cannot be generalized to unseen data. Hence, in practice available data is often split in a training data set and a test data set. The performance of a model trained on the training data set will be tested on the test data set, which should given the practitioner a better understanding of the model performance. Thus, there are two performance scores to monitor: one for the test data set and one for the training data set. The evolution of these two scores plotted against the size of the training set (in terms of instances), are called *learning curves*. Learning curves show how model performance changes as the size of the training data set increases.

Figure 6.2 shows an example of learning curves for two different classification models. The model on the left indicates a low variance, since the model performs similarly on the training and validation data set. Furthermore, the model shows a final performance of around 85%, meaning that 85% of instances were classified correctly, indicating a moderate bias. Adding more training instances here is unlikely to result in better performance, however a more complex model may improve performance. The model on the right indicates a high variance, since there exists a substantial gap between the performance of the model on the training and validation dataset. Adding more training instances here is likely to reduce the variance.

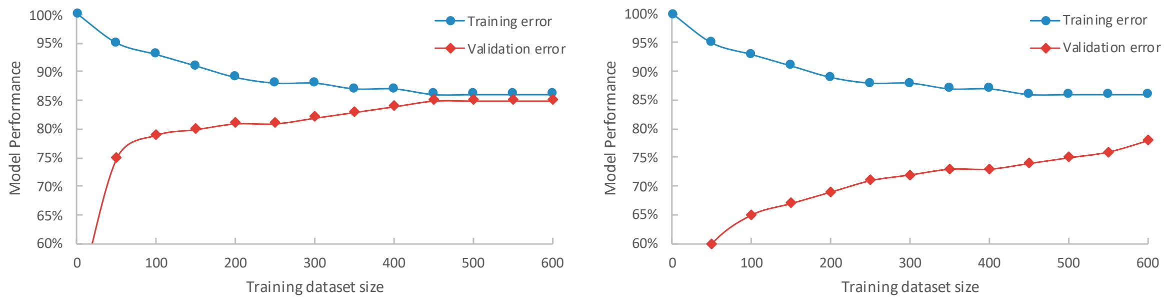


Figure 6.2: Example learning curves for two different classification algorithms. Left: Model with low variance and moderate bias. Right: Model with high variance and moderate bias. Adapted from: www.dataquest.io/blog/learning-curves-machine-learning/

In the example given above, the final model performance is around 85%. Attempting to reduce this error with a more complex algorithm is a valid follow up, based on the learning curves. As mentioned by Hastie et al. (2001), the reader should be warned that in practice a perfect model does not exist, due to the *irreducible or Bayes error rate*. In most real-world situations, the relationship between the features and labels is not perfect, for example due to the fact that not all features X that influence Y are captured in the data set. For many situations, the features may also contain measurement errors. For classification problems, the Bayes error rate gives an upper bound for error metrics that describe how good a model is.

6.2. Classifier Framework

The previous section presented a brief introduction to machine learning. This section will elaborate on the framework that was used to construct the machine learning classifier for the final DSS. A flowchart of the machine learning process is presented in Figure 6.3. A more detailed flowchart of the "Feature Engineering", "Hyperparameter Tuning" and "Model Validation" steps is shown in Figure 6.4. These steps are described in Sections 6.2.4, 6.2.5 and 6.2.6, respectively.

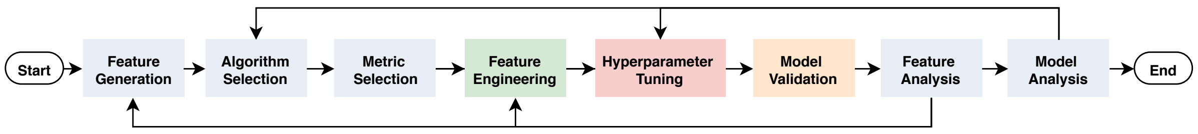


Figure 6.3: Machine Learning Framework

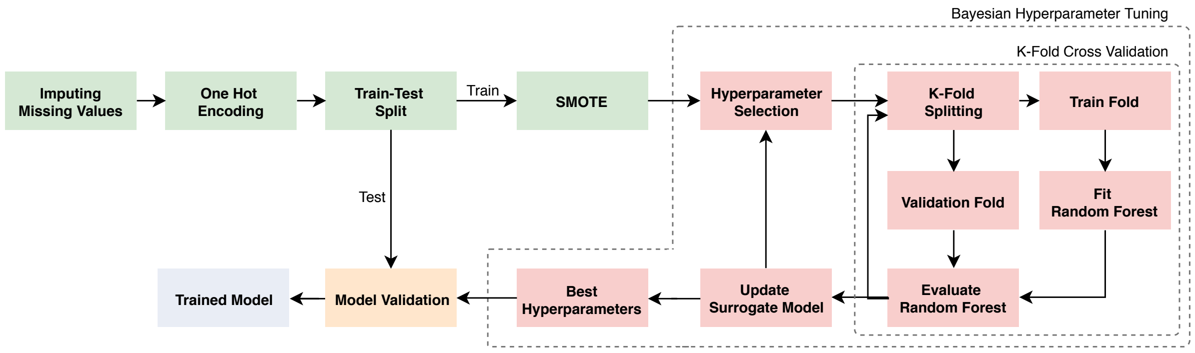


Figure 6.4: Flowchart for "Feature Engineering", "Hyperparameter Tuning" and "Model Validation" steps

6.2.1. Feature Generation

The first step in the machine learning framework is data gathering, i.e. feature generation. The quality and quantity of the data will largely determine the quality of the final machine learning algorithm. The aim of the classifier is to predict the probability that a candidate aircraft will help recover a disrupted flight or disrupted aircraft. Therefore, the features need to provide information on the candidate and disrupted aircraft and the schedule of these aircraft. Section 4.1 already presented the layout of the feature space. The features were generated based on domain knowledge and can be divided in the following categories:

- Candidate aircraft related features
- Disrupted flight/aircraft related features
- Features combining candidate aircraft and disrupted flight/aircraft characteristics
- Schedule related features

Table E.1 in Appendix E presents a detailed description of all features that were generated. These features were used to train a machine learning classifier, as discussed in the following sections. Based on the results of this classifier, feature analyses was performed as discussed in Section 6.2.7. Based on the analysis several changes were made to the feature space, which will be discussed in Section 6.2.7.

The feature space was generated by solving 1000 iterations of disruptions to a global optimum. The 1000 iterations covered about 6 days of disruptions in January. The aircraft that helped solve that disruption could be determined from the recovery solution, the disruption information and the schedule before the disruption. Figure 6.5 shows the framework that was used to generate the training set that contained the features.

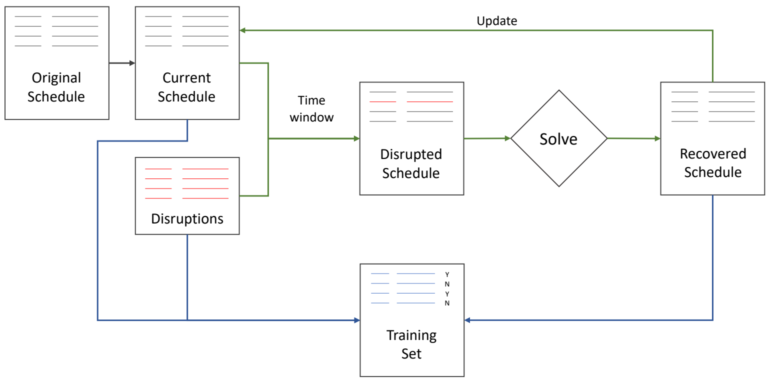


Figure 6.5: Framework used to generate the feature space for training the machine learning classifier

A single iteration can contain several disruptions, which will be solved at once. This complicates identifying which candidate aircraft helped solve which disruption. The following logic was used to relate candidate(s) to the correct disruptions:

- If the schedule of the candidate aircraft was changed, it helped solve one of the disruptions:
    1. If a candidate aircraft was tail swapped with one of the disrupted aircraft, the candidate was identified as helping solve that disruption. If more than one tail swap occurred, this logic was applied recursively on all tail swaps. For example, if disrupted flight DL220 was tail swapped from tail N9580 to tail N541DL, and flight DL988 was swapped from N541DL to N252S, both N541DL and N2525 are identified as helping solve the disruption.
    2. If the candidate was not tail swapped: for all flights of the candidate aircraft the origin and destination were checked. If any of the flights departed or arrived at the origin or destination airport of the disrupted flight/aircraft, the candidate was identified as helping solve that disruption.
- If the schedule of the candidate aircraft was not changed, it did not help solve any of the disruptions.

It should be noted that the described logic may not be sophisticated enough to correctly identify all candidate-disruption combinations. The labels in the feature space are expected to have some noise, which may impact the performance of the classifier. The results of the case study in Chapter 8 will determine to what extend the performance of the trained classifier is satisfactory.

6.2.2. Problem Characteristics and Algorithm Selection

The previous section described the framework that was used to create the feature space and the features that were created. This section will discuss the characteristics of the feature space and the logic that was used to select a machine learning algorithm. Nowadays, supervised classification is prevalent in research and industry, and a vast number of techniques have been developed. The most common of these techniques belong to one of four families: tree based algorithms, neural networks, statistics based algorithms and support vector machines.

Based on the feature dataset that is generated a few observations can be made:

- The instances in the feature dataset are labeled. Hence, supervised learning is possible and preferred.
- The feature dataset is dense, e.g. most features do not contain missing values or zeros.
- The dataset consists of both categorical (e.g. aircraft type) and numerical (e.g. direct operating cost) features.
- The features are not on the same scale. For example, the direct operating cost are in the thousands while the passenger load factor of the candidate aircraft is expressed between 0-1.
- The dataset is highly imbalanced. Only 0.25% of the instances belong to the ”True” class (i.e. candidate aircraft that helped), while the majority belongs to the ”False” class (i.e. candidate aircraft that did not help).

Olson et al. (2018) bench marked 13 state-of-the-art commonly used machine learning algorithms on a set of 165 publicly available classification problems. Each algorithm is fully hyperparameter optimized. Figure 6.6 shows the average ranking of the machine learning algorithms over all datasets. The paper demonstrates the strength of state-of-the-art, tree-based ensemble algorithms.

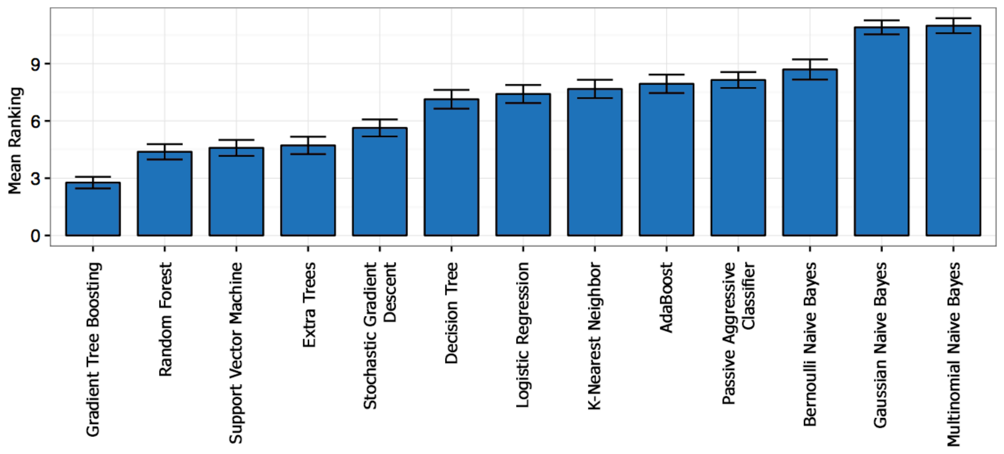


Figure 6.6: Average ranking of the algorithms over all datasets (lower is better). Error bars indicate the 95% confidence interval. Adapted from Olson et al. (2018).

Based on the results of Olson et al. (2018), the Gradient Tree Boosting, Random Forest and Support Vector Machine algorithms are expected to perform best for the aircraft classification. Table 6.1 presents a qualitative comparison of different supervised learning algorithms. The reader is referred to Kotsiantis et al. (2006) for a more thorough discussion on supervised classification algorithms.

Table 6.1: Qualitative comparison of classification techniques. Adapted from Kotsiantis et al. (2006)

	Decision Trees	Random Forest	Boosted Trees	Neural Networks	Naïve Bayes	kNN	SVM
Speed of learning w.r.t. no. of features and instances	***	**	**	*	****	****	*
Speed of classification	****	****	****	****	****	*	****
Tolerance to missing values	***	**	**	*	***	*	**
Tolerance to redundant features	**	**	**	**	*	**	***
Dealing with danger of over-fitting	**	***	***	*	***	***	**
Interpretability	****	**	**	*	****	***	*

Table 6.1 shows that Support Vector Machines (SVM) perform worse than Tree based algorithms regarding the speed of learning and the interpretability, while not scoring significantly better on other characteristics. Comparing the Random Forest and Boosted Trees algorithms, no distinction can be made based on the table. However, Boosted Trees are generated sequentially while Random Forest are generated in parallel. Therefore it is expected that Random Forests are generally slightly faster. Moreover, based on discussions with data scientists at ORTEC, it was found that the performance of Boosted Trees are generally more sensitive to the hyperparameters than Random Forests. In other words, the performance of Boosted Tree algorithms is very dependent on identifying the optimal hyperparameters, while Random Forests are less sensitive.

Based on this qualitative comparison, the Random Forest algorithm is chosen as the best starting point for the machine learning classifier. However, as mentioned by Olson et al. (2018), it should be noted that algorithm performance is strongly dependent on the problem specifics. The algorithm choice will be evaluated in Section 6.3.

6.2.3. Evaluation Metric Selection

The correct choice of evaluation metric plays a critical role during algorithm training, since it defines the goal for the algorithm. Hence, aligning the evaluation metric with the overall goal of classification is important to obtain a valid classifier. According to Hossin and Sulaiman (2015), threshold and ranking metrics are the most common metrics used by researchers.

For binary classification problems, the threshold metrics to identify the best solution during training can be defined based on the confusion plot shown in Figure 6.7. In this plot, TP (true positive) and TN (true negative) denote the number of positive and negative instances that were correctly classified. Meanwhile, FP (false positive) and FN (false negative) denote the number of misclassifed negative and positive instances, respectively. Table 6.2 presents several common threshold metrics, based on the confusion plot, to evaluate the performance of classifiers.

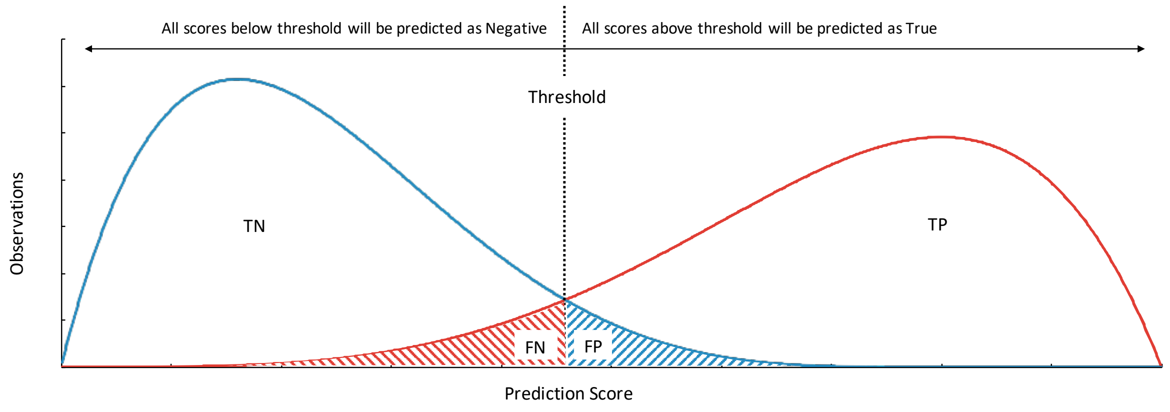


Figure 6.7: Confusion plot

By default, most binary or multi-class classifiers use *accuracy* or the *error rate* as the evaluation metric. Ranawana and Palade (2006) demonstrated that the simplicity of accuracy leads to sub-optimal performance when dealing with imbalanced class distributions. For example, when dealing with a dataset where 3% of the instances belong to class A and 97% of instances belong to class B, a classifier that classifies all instances as B will have a 97% accuracy score.

Table 6.2: Threshold metrics for classification evaluations. Adapted from Hossin and Sulaiman (2015)

Metric	**Formula**	**Evaluation Focus**
Accuracy (acc)	$\frac{TP+TN}{TP+TN+FP+FN}$	The accuracy metric measures the ratio of correct predictions over the total number of instances evaluated.
Error Rate (err)	$\frac{FP+FN}{TP+TN+FP+FN}$	The error rate measures the ratio of incorrect predictions over the total number of instances evaluated.
Specificity (sp)	$\frac{TN}{TN+FP}$	This metric is used to measure the fraction of negative predictions that are correctly classified.
Precision (p)	$\frac{TP}{TP+FP}$	Precision is used to measure the positive predictions that are correctly predicted from the total predictions in a positive class.
Recall (r)	$\frac{TP}{TP+FN}$	Recall is used to measure the fraction of positive predictions that are correctly classified

When choosing a threshold evaluation metric, it is important to think about what type of error should be minimized. For example, with a cancer detection problem, where out of 100 people only 5 people have cancer, the false negatives should be minimized. All people with cancer should be diagnosed as such, while it is fine to misdiagnose some healthy people, since further examination will declare them healthy. In contrast, with spam e-mail detection, false positives should be minimized. Here, important non-spam e-mails should not be classified as spam, while spam e-mails that are classified as not spam, can be manually deleted.

Precision gives information about the classifier's performance with respect to false positives, while *recall* provides information about the performance with respect to false negatives. However, it should be noted that when recall is chosen as the evaluation metric, this could lead to an increase in false positives. The inverse is true when choosing precision.

The *Area Under the Curve - Receiver Operating Characteristics (AUC-ROC)* is one of the most popular ranking metrics, opposed to threshold metrics. A Receiver Operating Characteristics curve is plotting True Positives Rate (recall) versus False Positives Rate (FP/(FP+TN)). The area under the curve indicates the overall ranking performance of a classifier. Figure 6.8 shows typical ROC curves for two algorithms. An optimal classifier would reach the top left corner in the AUC-ROC curve, where the true positive rate equals 1 and the false positive rate equals 0. Jin Huang and Ling (2005) showed that the AUC metric outperformed the accuracy metric for classifier performance evaluation and for discriminating the optimal solution during training. Although the performance of AUC-ROC was excellent, the computational cost of AUC-ROC is high compared to threshold metrics.

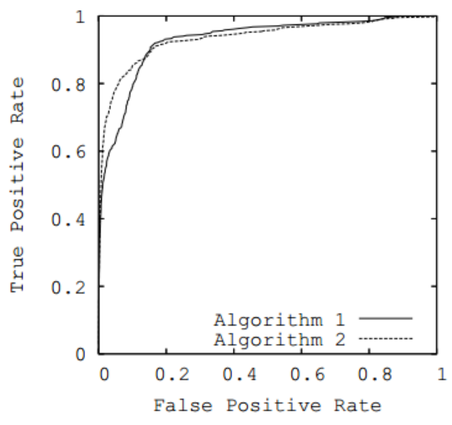


Figure 6.8: Example of typical ROC curves. Source: www.chioka.in (2014)

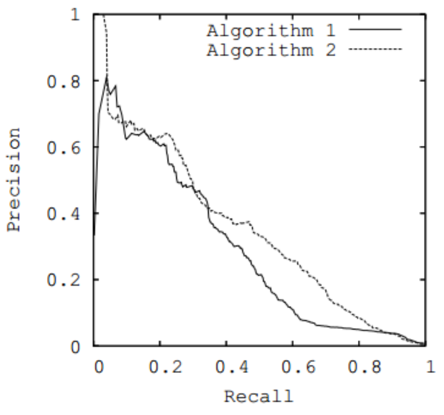


Figure 6.9: Example of typical Precision-Recall curves. Source: www.chioka.in (2014)

The *Area Under the Curve - Precision Recall (AUC-PR)* is another popular ranking metric, similar to the AUC-ROC. A PR curve is plotting Precision versus Recall. Since precision and recall do not take true negatives (TN) into account, the AUC-PR metric can be used in class imbalance problems where there is a 'False' majority class. Figure 6.9 shows the AUC-PR curves for two algorithms. An optimal classifier would reach the top right corner in the AUC-PR curve, where the recall and precision are both equal to 1. Since the AUC-PR metric, like the AUC-ROC metric, is computationally expensive to compute, the Average Precision (AP) metric can be used instead. The AP summarizes the AUC-PR as the weighted mean of precision achieved as each threshold, with the increase in recall from the previous threshold used as the weight. The formula for the AP is shown in Equation 6.2, where P_n and R_n are the precision and recall at the n-th threshold.

$$AP = \sum_n (R_n - R_{n-1})P_n \tag{6.2}$$

The AUC-ROC does not apply to this research since, due to the class imbalance, the dis-proportionally large number of true negatives (TN) will result in a low False Positive Rate across the board. The same is true for the AUC-PR and Average Precision (AP), where the large number of false positives (FP) will result in a low precision.

For the purpose of this research, the classification output will be used to construct a sub-network selection. Section 4.3 discussed several strategies for determining which aircraft to include in the sub-network. e.g. all aircraft with a probability above X. Moreover, given the imbalance of the ARP dataset, where only 0.25% belongs to the minority class, recall and specificity are important metrics to use when evaluating the quality of the classifier.

By replacing the precision with specificity in the AUC-PR and AP measures, the *Area Under the Curve - Specificity Recall* and *Average Specificity* are created for this research. Specificity measures the rate of correct predicted candidate aircraft that do not help with the recovery solution and recall measures the rate of the correct predicted candidate aircraft that do help with the recovery solution. Both specificity and recall should thus be maximized.

6.2.4. Data Pre-processing

Given that machine learning algorithms learn from data, data gathering and data quality are crucial for obtaining good results. This section discusses the data (pre-)processing steps that were taken before training the random forest classifier.

Dropping Columns

During feature generation several columns were added to the feature dataset that should not be used when training the classifier. These columns were primarily added for trace-ability and verification purposes, e.g. the iteration number, candidate aircraft tail number, disrupted aircraft tail number and the disrupted flight number. Furthermore, a column was added that verified that the candidate and disrupted aircraft belong to the same family. After it was verified that this was indeed the case for the entire dataset, the column was removed, since it did not hold any predictive value.

Moreover, after the random forest was trained and validated, feature analysis revealed that several columns were highly correlated and some features had a negative impact on the performance of the random forest. These features were also removed from the feature dataset in following iterations. This will be discussed further in Section 6.2.7.

Imputing Missing Values

One common problem in real world situations is that of erroneous data, where the gathered data is incomplete, noisy and/or inconsistent. In the generated features presented in Table E.1 in Appendix E, several columns have missing values. The random forest classifier presented by Breiman et al. (1984) and the implementation of Scikit-learn that was used for this thesis do not handle missing values, hence they need to be imputed beforehand. One obvious way of dealing with missing data is to delete those instances, however by doing so valuable information for training will be lost from the other features. Imputing missing values often leads to better results. There are several strategies for imputing values. It is the practitioners task to choose the best method based on the data type. For numerical data, the mean, mode or median of the column can be used to impute missing values. Another common method is to replace missing values with a constant numerical value far outside the normal range for the column, e.g. 9999. For categorical data, the missing data can be replaced with the most frequent category value or can be given its own category. Finally, for both numerical and categorical data, a separate prediction algorithm can be used to predict the missing values. For the purpose of this thesis, the missing values were imputed with a numerical value far outside the normal range of the feature. This way the instances with missing values can be kept, while it should be straightforward for the classifier to filter out these outliers. Table 6.3 shows the features that contain missing values and the values that were used to impute the missing values.

Table 6.3: Imputed values for feature columns with missing values

Feature	Value Range	Reason for NaN	Imputed value
c_econ_lf_std	0-1	Single flight on flight string, so no std. dev.	0
c_buss_lf_std	0-1	Single flight on flight string, so no std. dev.	0
same_airport_min_before	0-inf	Candidate not on same airport as disrupted	-999
same_airport_min_after	0-inf	Candidate not on same airport as disrupted	-999

One-Hot Encoding
The feature space consists of both numerical and categorical data. Since there exists an imbalance between the minority and majority class, oversampling techniques are employed to correct this imbalance, as will be discussed later in this section. The algorithm that was used for re-sampling the instances only handles numerical data, hence the categorical data needed to be converted to numerical data. Furthermore, the Scikit-Learn 0.20.1 implementation of the random forest classifier that was used for this research does not accept categorical features. The most straightforward conversion technique is *label encoding* where the categories are simply replaced by integers, e.g. '737' becomes 1, 'A320' becomes 2, etc...However, integer values have a natural ordered relationship between each other and many machine learning algorithms will assume that a 5 is better than a 2. For most categorical variables, where no ordinal relationship exists, label encoding is not advanced enough and will likely result in poor performance.

For the categorical features in the dataset for this thesis, no ordinal relationship exists, and *one-hot encoding* was used to convert the categorical data to numerical data. The idea behind one-hot encoding is to apply binarization to the categorical data, where the label encoded variables are removed and a new binary variable is added for all unique integer values. Table 6.4 shows the difference between label encoding and one-hot encoding.

Table 6.4: Difference between label encoding and one-hot encoding for converting categorical data to numerical data

Label encoding		
Aircraft Type	**Categorical No.**	**Range**
A320-200	1	3000
737-700	2	2930
MD-88	3	2045

One-Hot encoding			
Is_A320-200	**Is_737-700**	**Is_MD-88**	**Range**
1	0	0	3000
0	1	0	2930
0	0	1	2045

Splitting Data
The dataset available after one hot encoding is divided into a train and test set for validation purposes. The classifier is trained on the train set and validated on the test set. The train-test split is performed before training the classifier, since instances from the test set may not be used during training of the model, to ensure objective validation after training the model. Typically, the test set comprises 15%-30% of the total available dataset. For this thesis a 75%/25% Train/Test split was used.

Class Imbalance Corrections
The instances in the feature dataset where an aircraft helped with a disruption is approximately 0.25% of the total instances. This can be explained in part by the fact that for most of the disruptions, no other aircraft is needed for the optimal recovery solution. For example, most disruptions cause a delay $\leq$ 30 minutes. For the majority of these disruptions, the best option would be to delay the flights on the flight string of the disrupted aircraft. Furthermore, when additional aircraft are required to aid with the recovery of the disruption, more often than not the number of aircraft will be small compared to the size of the (sub-)fleet. In literature, e.g. Weiss (2013), a data set where the number of observations

of one class are significantly smaller than the observations of the other class, is called an imbalanced dataset. This imbalance can lead to an underestimation of the probability of belonging to the minority class.

Common approaches for handling class imbalance can be divided into *data level methods* and *algorithmic level methods*. The basic idea of data level class balancing methods is to re-sample the dataset to create a more equal class distribution. This can be done by randomly replicating occurrences of the minority class, which is called random over-sampling. Inversely, the dataset can be balanced by randomly deleting occurrences of the majority class. Both methods have their flaws. Over-sampling increases computation time, since the dataset becomes larger. Moreover, it increases the likelihood over overfitting, since rules may be constructed that seem accurate but in reality only cover a single replicated instance. Under-sampling risks discarding useful information which could be important to the learning process. Moreover, the selected subset of instances could be biased and therefore not representative of the entire class. Luckily, there exists a more intelligent way to over-sample, where instances are generated using some interpolation technique. Chawla et al. (2002) presented the Synthetic Minority Over-sampling Technique (SMOTE), which is a popular method for advanced over-sampling. The idea is to create new minority class instances by interpolating between multiple minority class instances that lie together. By using SMOTE, overfitting is reduced and the minority class decision boundaries are spread further into the majority class space.

One popular algorithmic level method for handling class imbalance is *changing the probability threshold* between the classes. The output of a classification problem is usually a set of class probabilities which describe the probability of an instance belonging to a class. For binary classification, the threshold is 50% by default. All probabilities > 50% would be classified as 1 or True and all probabilities < 50% as 0 or False. Changing this threshold affects the output and performance of the classification model.

Another algorithmic level approach is called *cost-sensitive learning*. By default, all misclassification is treated equally, without rewarding identification of of the minority class. Cost-sensitive learning defines a matrix that specifies the cost of misclassifying an instance of class A as class B. Using these cost the misclassification of the minority class can be penalized more heavily than misclassification of the majority class. A common set-up is to have the cost equal to the inverse of the proportion minority to majority class. This increases the penalization as the class size decreases.

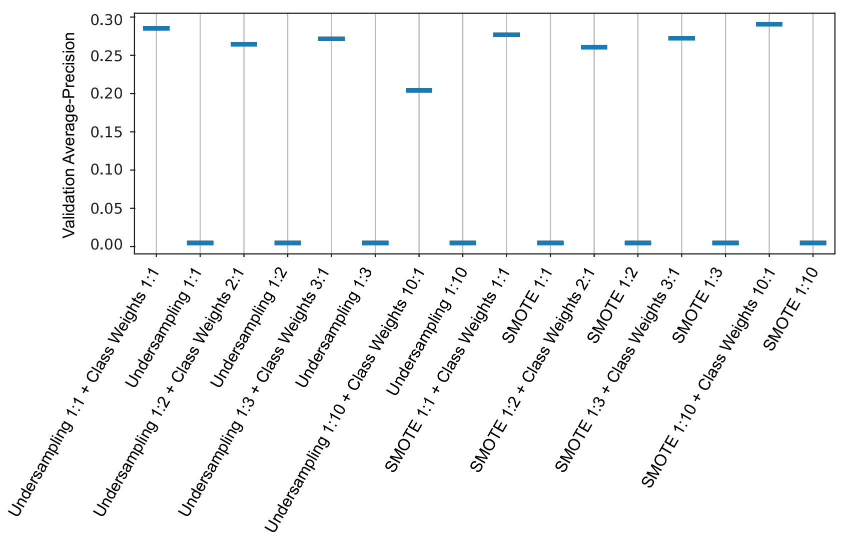


Figure 6.10: AP scores for different class balancing strategies

For this research several combinations of the mentioned re-sampling techniques and cost-sensitive learning were tested to identify the impact of each method and the (combination of) method(s) best suited for the problem. Figure 6.10 shows the results of these technique and parameter combinations. It should be noted that this was not an exhaustive search of all possible parameter combinations, nonetheless it can be seen that including cost-sensitive learning had the largest impact on the validation scores. Furthermore, SMOTE scored more consistent than under sampling. Therefore, the combination of SMOTE and Cost-sensitive learning was used when training the random forest classifier.

After the classifier was trained, the confusion matrix for different probability thresholds was obtained to determine the recall and specificity at each threshold. The results will be presented in Section 6.3.

6.2.5. Hyperparameter Optimization

The random forest classifier algorithm has specific hyperparameters that govern how the algorithm works, such as the maximum depth of the trees or the number of trees. Unlike the model parameters, such as the split points, hyperparameters are defined before training. The goal of hyperparameter optimization is to find the set of hyperparameters that, given a training dataset, will return the best possible performance on the test dataset, as measured by the evaluation metric.

Since evaluating the objective function of an algorithm to find the validation score can be expensive, trying all possible combinations of the hyperparameters is often impossible. For each possible combination of hyperparameters, the model has to be trained, predictions will have to be made and the score needs to be computed. With a large number of hyperparameters, some of which are continuous, and complex models, this soon becomes intractable.

The simplest way of performing hyperparameter optimization is by exhaustively searching through a manually set discrete subset of the hyperparameter space, a so called *grid search*. Although simple, grid search suffers from the ’curse of dimensionality’, where the computation time grows exponentially with the number of hyperparameters and the number of options per parameter. Because the hyperparameter space is discretized, grid search does not guarantee finding the optimal set of hyperparameters. *Random search* replaces the exhaustive searching by selecting random combinations of hyperparameter values. Bergstra and Yoshua (2012) showed that random search out performs grid search.

Both grid search and random search make uninformed decisions when choosing the next hyperparameters to evaluate, and as a result, often spend significant time evaluating poor performing hyperparameter combinations.

To optimize the hyperparameters of the random forest classifier for this research, *Bayesian optimization* was used. The Bayesian Optimization algorithm builds a probabilistic surrogate model that approximates the objective function $f(x)$ of the machine learning model. Typically, a Gaussian Process (GP) is used as the surrogate model. By sequentially evaluating hyperparameter combinations, based on the surrogate model and then updating the model, Bayesian optimization, gathers information with each iteration which leads to the location of the optimum. The use of all available information from previous iterations of $f(x)$ instead of relying on the local gradient and/or Hessian approximations is what makes Bayesian optimization powerful. The resulting procedure can find the minimum of complex non-convex functions with relative ease, i.e. without extensively evaluating $f(x)$, which is typically expensive.

An acquisition function is used to determine the next hyperparameter combination to evaluate. A popular choice is the Expected Improvement, which is the expected improvement in the value of $f(x)$ over the best value of $\hat{f}$ yet seen. This can be written as:

$$EI(x) = \mathbb{E}(max(f(x) - \hat{f}, 0)) \quad \text{where } \hat{f} \text{ is the maximum value of f seen so far} \tag{6.3}$$

This acquisition function tries to balance exploration, e.g. hyperparameter combination for which the outcome is uncertain, with exploitation, e.g. combinations that are close to the current best guess. Snoek et al. (2012) showed that Bayesian Optimization is able to obtain better results compared to both grid and random search, in fewer evaluations.

Figure 6.4, in the beginning of this chapter, shows the process of Bayesian Hyperparameter Tuning. Every iteration, a set of hyperparameters is chosen based on the acquisition function. For every iteration, *K-fold cross validation* is used to evaluate the performance of the random forest with the hyperparameters. An example of 5 fold cross validation where the training set is divided into five equal-sized mutually exclusive subsets is given in Figure 6.11. For each subset, the classifier is trained on the union of the other sets. The performance of the classifier over all subsets is then averaged. By performing K-fold cross validation the training set is kept larger and the sampling bias in the training set is decreased.

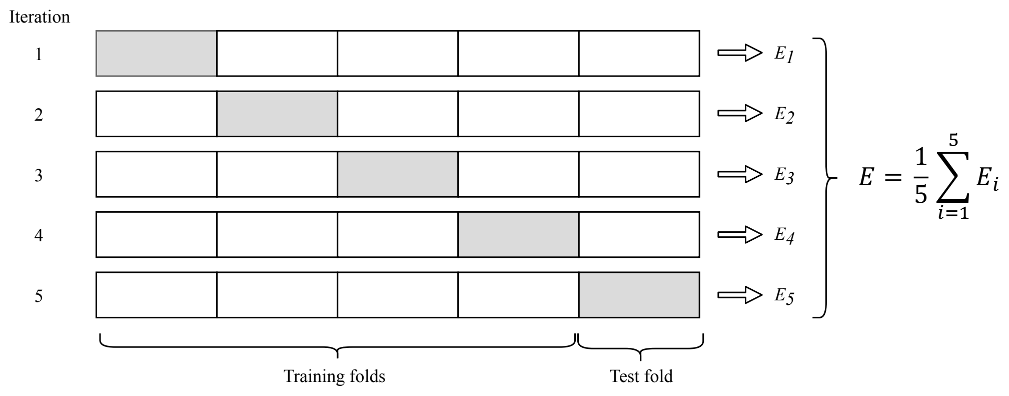


Figure 6.11: Example of K-fold cross validation where K=5

One of the hyperparameters that needs to be set for the random forest classifier is the number of trees in the forest. In general, as many trees as possible should be used, since more trees does not negatively impact the predictions of the classifier. However, a forest with more trees does take longer to train and takes longer to generate predictions. Furthermore, the incremental performance improvement decreases with a larger number of trees. To determine the optimal number of trees for the random forest classifier, the performance of different sized forests was compared. Figure 6.12 shows the impact of changing the number of trees. All other hyperparameters were kept constant. The analysis was performed three times, where the random state of every classifier was changed, to ensure the resulting optimal number of trees holds across different random forests. From the graph, it can be seen that the performance of the classifiers levels off around 200 trees. Therefore, the number of trees in the random forest classifier was fixed on 200.

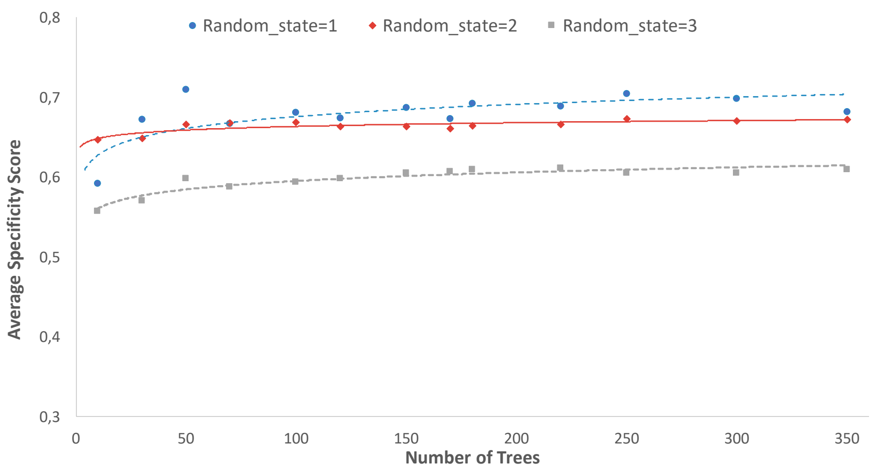


Figure 6.12: Impact of number of trees in random forest on model performance

After the number of trees in the forest was determined, the random forest classifier was trained on the 75% Train Split for 100 iterations with 5-fold cross validation using Bayesian Optimization. The following hyperparameters were tuned:

- **max_depth: between 1-50**
  The maximum depth of each tree in the forest. Deeper trees capture more splits and more information about the data. However, deeper trees also have a tendency to over fit on the training data, which decreases their ability to generalize to new data.
- **min_samples_split: between 10-100**
  Describes the minimum number of instances required to split an internal node in the tree. A lower number of instances could result in overfitting or a tree capturing noise, a higher number could result in underfitting.
- **min_samples_leaf: between 10-150**
  A leaf is an end node in a tree. This parameter determines the minimum number of instances that should be contained in one leaf. A low number could result in the tree capturing noise, a higher number could result in underfitting.
- **max_features: between 1-all**
  Random forests consider only a random subset of features per tree. This parameter determines the maximum number of features to include per tree. The power of Random Forests is that they do not consider all features in every tree, however more features allow for better splits in a tree.

Section 6.3 presents the optimal hyperparameters that were found after tuning. Section 8.3.3 discusses the sensitivity of the classifier's performance to the hyperparameters. After hyperparameter optimization, the optimal hyperparameters are used to construct a random forest classifier. This model needs to be validated on the 25% Test Split to check the performance of the classifier on new data. This validation is discussed in the next section.

6.2.6. Classifier Validation

The detailed framework in Figure 6.4 shows a train/test split where the training data is used to optimize the hyperparameters of the classifier, as discussed in the previous section. The test split is used to validate the trained classifier. Since the classifier was not trained on this dataset, the validation score measures how well the classifier is able to generalize to new data.

The average specificity score of the classifier on the test split equals 0.7885. The score shows that there is some predictive value in the feature space, however the performance can probably be improved. If improvement is indeed possible, this can be achieved by increasing the quality of the feature space (i.e. feature engineering), more elaborate hyperparameter optimization or by using a more complex machine learning algorithm. The following section will discuss the feature analyses, which aims to identify potential areas for improvement in the generated feature space.

6.2.7. Feature Analyses

During feature generation, 46 features were created and added to the feature space. Most likely, not all of those features are equally important for the final algorithm performance. Unnecessary features decrease model speed and interpretability. Moreover, keeping unnecessary features could decrease generalization performance of the final random forest classifier. This section describes several analyses that were performed to determine feature importance and to discover relationships. Section 6.2.8 will discuss how these analyses were used to improve the feature space.

Feature Importance

The Scikit-Learn 0.20.1 implementation of the random forest classifier computes the *Gini importance*, as described by Breiman et al. (1984), automatically. This method determines the feature importance based on the number of times the feature is used in the random forest and the level at which the feature is used, which is a measure of the discriminatory power of the feature. However, Strobl et al. (2007) benchmarked several feature importance measures and concluded that *“the variable importance measures of Breiman’s original Random Forest method ... are not reliable in situations where potential predictor variables vary in their scale of measurement or their number of categories.”* The paper showed that the model-agnostic *permutation feature importance*, did not suffer from the same bias and performs better on problems where the features vary in scale and/or their number of categories. Using this method the difference in model performance is measured when one of the features is permuted or randomized. If the performance decreases, the feature is important, since the model relied on the feature for classification. If the feature in unimportant, permuting its values should not impact the model performance. Figure 6.13 shows the permutation importance of the features in the dataset.

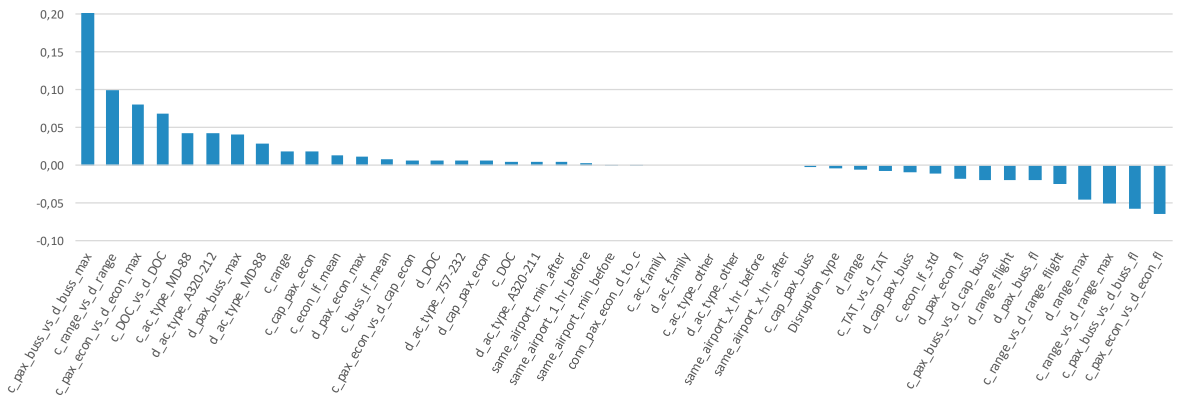


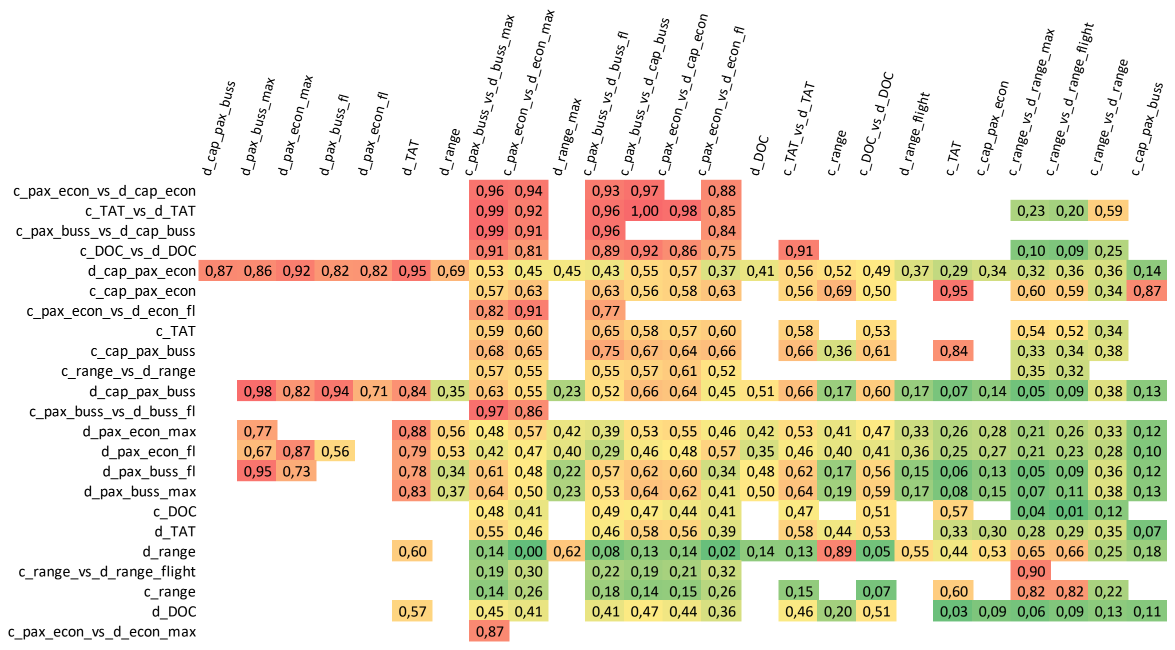
Figure 6.13: Permutation feature importance of features in the dataset

According to Figure 6.13, some features actually have a negative impact on the classifier’s performance. The following can be deduced from the figure:

- Most features related to the candidate or disrupted aircraft type have zero importance. Only if the candidate is of type MD-88 or if the disrupted aircraft is of the type MD-88, A320-212, A320-211 or 757-232 does this have any significance for the predictions. This could be due to specific differences in the characteristics of the aircraft types within a family.
- The combination features that describe whether the candidate can take over the disrupted flight are most important, i.e c_pax_buss_vs_d_buss_max, c_pax_econ_vs_d_econ_max and c_range_vs_d_range.
- Interestingly, the combination features c_pax_buss_vs_d_buss_fl and c_pax_econ_vs_d_econ_fl have the most negative impact on the performance of the classifier. This may be because, if two aircraft are tail switched the candidate aircraft should be able to take over the entire flight string and not only the one disrupted flight.

Feature Correlations

Since many of the features that are generated for this research are linear combinations of each other, there is a high probability that the features in the dataset are correlated. Since a lower number of features results in faster training, faster predictions and more interpretable models, it is useful to determine which features are correlated. If two features are highly correlated, one of those features can be removed without loosing much information. Figure 6.14 shows the feature correlation matrix for the top 25 correlated features. The features regarding the aircraft types and families are excluded to ensure readability. Appendix F shows the top 100 feature correlations.



	d_cap_pax_buss	d_pax_buss_max	d_pax_econ_max	d_pax_buss_fl	d_pax_econ_fl	d_TAT	d_range	c_pax_buss_vs_d_buss_max	c_pax_econ_vs_d_econ_max	d_range_max	c_pax_buss_vs_d_buss_fl	c_pax_buss_vs_d_cap_buss	c_pax_econ_vs_d_cap_econ	c_pax_econ_vs_d_econ_fl	d_DOC	c_TAT_vs_d_TAT	c_range	c_DOC_vs_d_DOC	d_range_flight	c_TAT	c_cap_pax_econ	c_range_vs_d_range_max	c_range_vs_d_range_flight	c_range_vs_d_range	c_cap_pax_buss
c_pax_econ_vs_d_cap_econ								0,96	0,94		0,93	0,97		0,88											
c_TAT_vs_d_TAT								0,99	0,92		0,96	1,00	0,98	0,85								0,23	0,20	0,59	
c_pax_buss_vs_d_cap_buss								0,99	0,91		0,96			0,84											
c_DOC_vs_d_DOC								0,91	0,81		0,89	0,92	0,86	0,75		0,91						0,10	0,09	0,25	
d_cap_pax_econ	0,87	0,86	0,92	0,82	0,82	0,95	0,69	0,53	0,45	0,45	0,43	0,55	0,57	0,37	0,41	0,56	0,52	0,49	0,37	0,29	0,34	0,32	0,36	0,36	0,14
c_cap_pax_econ								0,57	0,63		0,63	0,56	0,58	0,63		0,56	0,69	0,50		0,95		0,60	0,59	0,34	0,87
c_pax_econ_vs_d_econ_fl								0,82	0,91		0,77														
c_TAT								0,59	0,60		0,65	0,58	0,57	0,60		0,58		0,53				0,54	0,52	0,34	
c_cap_pax_buss								0,68	0,65		0,75	0,67	0,64	0,66		0,66	0,36	0,61		0,84		0,33	0,34	0,38	
c_range_vs_d_range								0,57	0,55		0,55	0,57	0,61	0,52								0,35	0,32		
d_cap_pax_buss		0,98	0,82	0,94	0,71	0,84	0,35	0,63	0,55	0,23	0,52	0,66	0,64	0,45	0,51	0,66	0,17	0,60	0,17	0,07	0,14	0,05	0,09	0,38	0,13
c_pax_buss_vs_d_buss_fl								0,97	0,86																
d_pax_econ_max		0,77				0,88	0,56	0,48	0,57	0,42	0,39	0,53	0,55	0,46	0,42	0,53	0,41	0,47	0,33	0,26	0,28	0,21	0,26	0,33	0,12
d_pax_econ_fl		0,67	0,87	0,56		0,79	0,53	0,42	0,47	0,40	0,29	0,46	0,48	0,57	0,35	0,46	0,40	0,41	0,36	0,25	0,27	0,21	0,23	0,28	0,10
d_pax_buss_fl		0,95	0,73			0,78	0,34	0,61	0,48	0,22	0,57	0,62	0,60	0,34	0,48	0,62	0,17	0,56	0,15	0,06	0,13	0,05	0,09	0,36	0,12
d_pax_buss_max						0,83	0,37	0,64	0,50	0,23	0,53	0,64	0,62	0,41	0,50	0,64	0,19	0,59	0,17	0,08	0,15	0,07	0,11	0,38	0,13
c_DOC								0,48	0,41		0,49	0,47	0,44	0,41		0,47		0,51		0,57		0,04	0,01	0,12	
d_TAT								0,55	0,46		0,46	0,58	0,56	0,39		0,58	0,44	0,53		0,33	0,30	0,28	0,29	0,35	0,07
d_range						0,60		0,14	0,00	0,62	0,08	0,13	0,14	0,02	0,14	0,13	0,89	0,05	0,55	0,44	0,53	0,65	0,66	0,25	0,18
c_range_vs_d_range_flight								0,19	0,30		0,22	0,19	0,21	0,32								0,90			
c_range								0,14	0,26		0,18	0,14	0,15	0,26		0,15		0,07		0,60		0,82	0,82	0,22	
d_DOC						0,57		0,45	0,41		0,41	0,47	0,44	0,36		0,46	0,20	0,51		0,03	0,09	0,06	0,09	0,13	0,11
c_pax_econ_vs_d_econ_max								0,87																	

Figure 6.14: Feature correlation matrix for top 25 correlated features. The features regarding aircraft types and families are excluded.

From the feature correlation matrix and the top 100 feature correlations in the appendix the following can be deduced:

- Either the Disruption_type_AC_Unavailable or Disruption_type_Delay can be dropped since these are inversely correlated, e.g. a disruption is either caused by an aircraft unavailability or by a delay.
- The business and economy load factor are highly correlated. Since load factor information was not separately available for economy and business class, it was assumed that these are equal. This does not necessarily hold for a real airline schedule.
- The features related to the Turn-Around-Time, Direct Operating Cost and passenger capacities are highly correlated. This makes sense since larger aircraft can accommodate more passengers, have a higher TAT and a higher DOC.

Feature Histogram Analysis

Some features have a direct relation with the target variable, i.e. class. For all features a figure was created which shows the histograms of both classes. The histograms for all features can be found in Appendix G. Most of the histograms showed no clear relationship between the feature value and the class. Figures 6.15, 6.16 show the histograms of the same_airport_min_after_min and c_pax_econ_vs_d_econ_fl features, respectively. These histograms indicate a slight difference in the distribution of the two classes over the feature value. Figure 6.15 shows that candidate aircraft that arrive or depart quickly after the STD or STA of the disrupted aircraft/flight, have a higher probability of not helping solve the disruption. Candidates that are on the same airport between 400-600 minutes after the STD or STA of the disrupted aircraft/flight have a relatively high probability of helping solve the disruption. Figure 6.16 shows that candidate aircraft that do not have enough capacity in economy class for the number of passengers booked on the disrupted flight, i.e. a negative value, have a close to zero probability of helping solve the disruption.

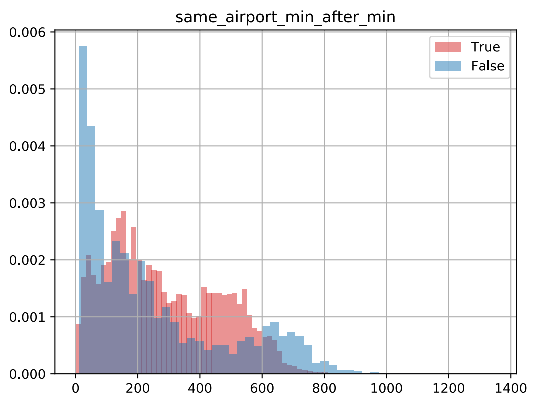


Figure 6.15: Histogram of feature: same_airport_min_after_min

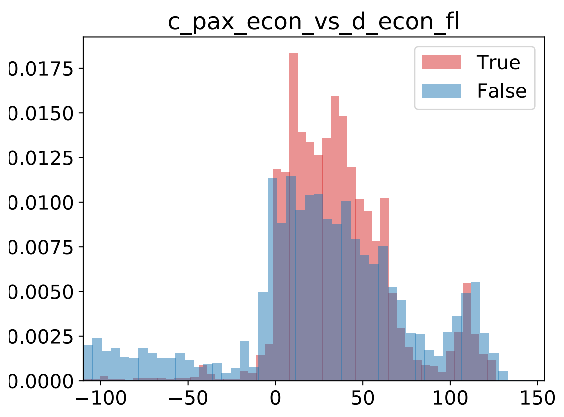


Figure 6.16: Histogram of feature: c_pax_econ_vs_d_econ_fl

6.2.8. Feature Engineering

The feature analyses resulted in a better insight into the importance and predictive value of the features. Based on the analyses, several changes were made to the feature space. These changes are discussed in this section.

The analyses showed that the features related to the schedule of the candidate aircraft with respect to the origin and destination airport of the disruption has some importance and shows some discriminatory power with respect to the classes. However, these features combined information from the origin and destination airport of the disruption. Table 6.5 shows the new features by which they have been replaced. These replacements features split the information for the origin and destination airport. Descriptions of all added features can be found in Table E.2 in Appendix E.

Table 6.5: Feature replacements after feature analyses

Initial Feature	Replacement Feature(s)
same_airport_min_before	d_origin_min_before_min
	d_dest_min_before_min
same_airport_min_after	d_dest_min_before_min
	d_orig_min_before_min
same_airport_1/2/3hr_before	d_orig_airport_1/2/3hr_before
	d_dest_airport_1/2/3hr_before
same_airport_1/2/3hr_before	d_orig_airport_1/2/3hr_before
	d_dest_airport_1/2/3hr_before

After analyzing the features, it was noticed that not all information related to the disruption was captured in the feature space. For that reason the following features were added:

- Disruption_cause: The cause of the disruption; Airline, NAS, Weather or Weather_Group
- Disruption_duration: The duration of the disruption in minutes

Although the features related to the schedule of the candidate aircraft captured whether the candidate was on the same origin or destination airport as the disrupted aircraft/flight, the feature space did not contain any information on the availability of the candidate. For this reason the following features were added:

- c_no_flights: The number of flights scheduled for the candidate aircraft in the time window

- c_flights_duration: The sum of the flight duration for the flights scheduled for the candidate aircraft in the time window
- c_ground_time_d_orig_airport: The total time the candidate aircraft spends on the ground at the origin airport of the disrupted aircraft/flight
- c_ground_time_d_dest_airport: The total time the candidate aircraft spends on the ground at the destination airport of the disrupted aircraft/flight

The feature analyses also suggested that several features could be removed from the feature space, either because they hold no predictive power or because they are highly correlated with another feature. The following features have been removed from the feature space:

- c_pax_buss_vs_d_buss_fl and c_pax_econ_vs_d_buss_fl, since they showed to have a negative impact on the classifier's performance.
- c_pax_buss_vs_d_buss and c_pax_econ_vs_d_econ, since they are highly correlated with c_pax_buss_vs_d_buss_max and c_pax_econ_vs_d_econ_max, respectively. Intuitively, relating the candidate's capacity to the actual maximum number of passengers on the flight string of the disrupted aircraft makes more sense than relating the candidate's capacity to the disrupted aircraft capacity.
- c_ac_family and d_ac_family, since they hold no predictive power. For all cases, the aircraft family of the candidate and disrupted aircraft are equal.
- c_buss_lf_mean and c_buss_lf_std, since they are highly correlated with the economy load factor features.

After the changes to the feature space were made, a new training dataset was generated. Using this new dataset another random forest classifier was hyperparameter optimized. The next section will discuss the characteristics and performance of the new random forest classifier.

6.3. Final Random Forest Classifier and Model Analysis

This section will present the characteristics of the random forest classifier that was created and used in the case study, discussed in Chapter 8. Furthermore, the performance of the classifier is discussed and the choice of using the random forest algorithm is evaluated.

The following optimal hyperparameters were found after 5-fold cross validation using Bayesian hyperparameter optimization with the new feature space:

- n_estimators: 200
- criterion = 'gini'
- class_weights: 0:1, 1:10
- max_features: 29
- max_depth: 33
- min_samples_leaf: 67
- min_samples_split: 67

The sensitivity of the classifier's performance to the hyperparameters is discussed in Section 8.3.3.

Section 6.1 of this chapter discussed the bias-variance trade-off and learning curves. Figure 6.17 shows the learning curves for the final random forest classifier. The curves indicate a low bias but high variance which indicates that the model probably overfitted to the training data and will perform worse on new data. Section 9.2 will present several recommendations to decrease the overfitting and increase the model performance.

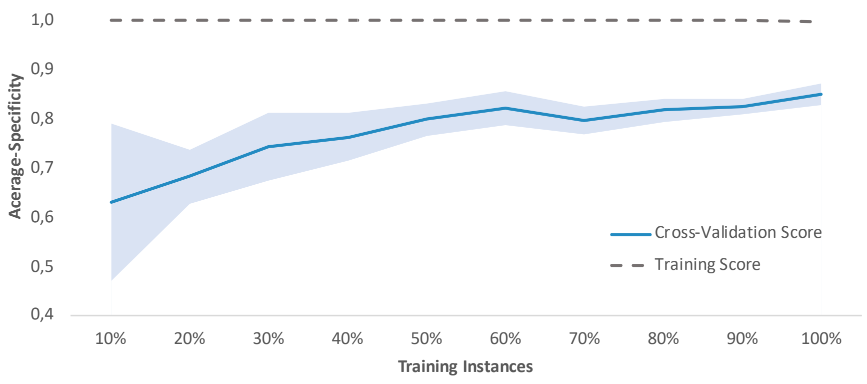


Figure 6.17: Learning curves for the final random forest classifier. The shaded area indicates one standard deviation.9

The final classifier has an average specificity score of 0.89, compared to 0.79 for the initial classifier. Figure 6.18 shows the Specificity-Recall curves for both classifiers. The 0.89 average specificity score of the random forest classifier implies both a high specificity and recall. This in turn means that, in most cases, the classifier is able to correctly predict the class of an aircraft, i.e. if the candidate aircraft will help solve the disruption. Figure 6.19 shows the recall and discard% at different probability thresholds. The discard% is the number of aircraft that are classified as not helping over the total number of aircraft. The chart shows that at a probability threshold of 0.26, the recall equals 95% and 50% of the aircraft are discarded. In other words, the random forest classifier can predict the aircraft that helped solve a disruption with 95% certainty while allowing 50% of the aircraft to be removed for optimization. This seems promising, but the case study will determine the overall performance of the classifier when implemented in the final decision support system.

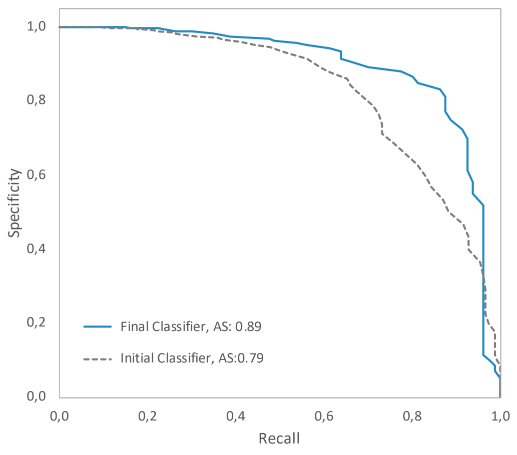


Figure 6.18: Specificity-Recall Curve for the initial and final random forest classifiers

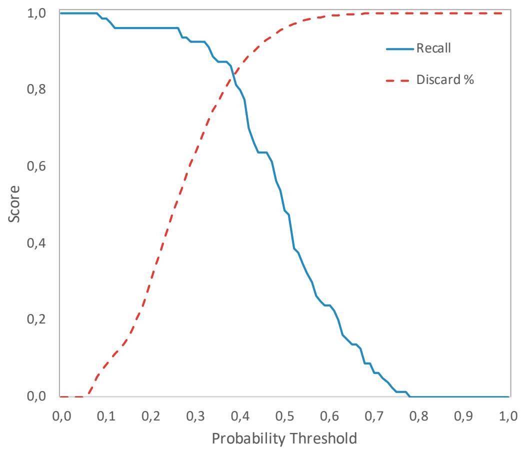


Figure 6.19: Recall and Discard% at different probability thresholds

In the beginning of this chapter, the algorithm selection section discussed different supervised learning algorithms and the choice for the random forest algorithm. The random forest algorithm was chosen based on the results of Olson et al. (2018), the qualitative comparison by Kotsiantis et al. (2006) and discussions with data scientists at ORTEC. Based on the performance of the final classifier, the algorithm choice seems justified. The performance difference of the initial and final random forest classifier shows the impact of feature analyses and feature engineering. Further improvements to the feature space and further investigation into the potential label noise are likely to result in a better classifier. The case study in Chapter 8 will determine how the classifier performs when implemented in the DSS with the sub-network selection algorithm. Moreover, the performance of the system on a different time period than the period used for training the classifier, will determine the classifier's ability to generalize.

7 Verification and Validation

This chapter will discuss the verification and validation of the decision support system (DSS). Section 7.1 will present the verification of the system with several examples for which the results are easily verified. From these examples it can be determined whether the model behaves as expected. Validation of the system is discussed in Section 7.2, which will summarize discussions with an expert from industry.

7.1. Verification

Verification aims to prove that the developed DSS is constructed properly and behaves as expected. The verification will be performed in two steps. Section 7.1.2 will discuss the verification of only the optimization model, while Section 7.1.3 will discuss the verification of the whole DSS. Throughout this section, verification will be discussed on the basis of several examples which can be verified by hand. For each example a time space representation will be shown and the objective value of the recovery solution will be verified by calculating the disruption cost by hand. To better understand the objective function, Section 7.1.1 will first present the different cost factors that are used in this objective function.

7.1.1. Cost Factors

Section 4.1 briefly discussed some of the different cost factors for the aircraft recovery model. This section will further elaborate on the different cost factors to ensure full understanding of the objective function, which will be be extensively discussed throughout this chapter to verify different examples. The soft cost used in the model, which represent the delay cost per passenger per minute of delay are given in Figure 7.1. These cost are equal to the soft cost used by Vink et al. (2019) and based on the research on delay cost by Cook et al. (2012). The hard cost correspond to the legal compensation passengers are entitled to under local legislation, and are shown in Figure 7.1 as well. When a flight is cancelled, the soft cost corresponding to 10 hours of delay and the hard cost corresponding to 8+ hours of delay are combined.

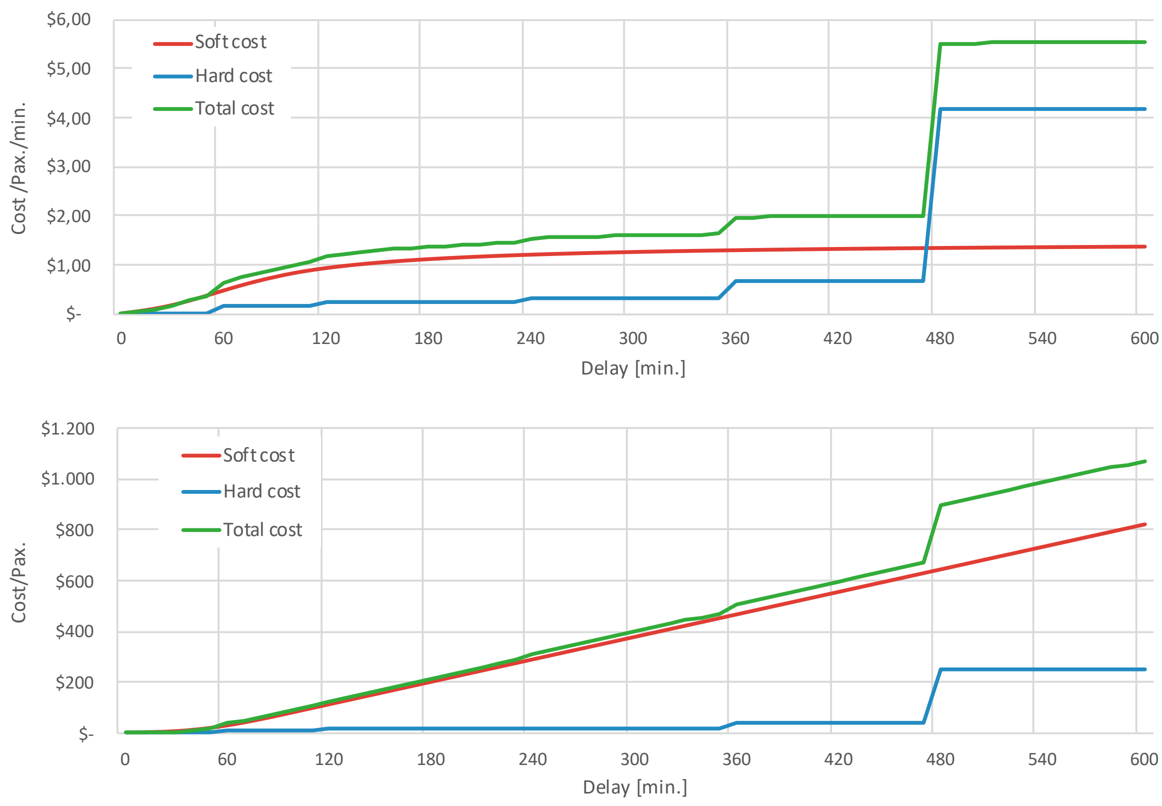


Figure 7.1: [Top] Delay cost (soft, hard, total) per minute per passenger. [Bottom] Delay cost (soft, hard, total) per passenger. Adapted from Vos et al. (2015)

7.1.2. Verification of Optimization Model

In this section the optimization model will be verified using several examples that can be verified by hand. Network tests will explain the effect of different disruptions and, where applicable, how these disruptions propagate through the network. Furthermore, an example regarding the modified connecting cost matrix will explain the effect of allowing connecting flight delays on the objective function. Additional verification scenarios are presented by Vink (2016), which used the same optimization model.

One aircraft, one delay

In this verification scenario one aircraft is scheduled to fly five flights in the time window. This flight schedule is shown in the top time-space graph in Figure 7.2. Flight 1, from PNS to MIA, is delayed 90 minutes. The optimal recovery solution is depicted in the bottom graph in the same figure.

As a result of the disruptions, all subsequent flights of the aircraft are delayed. Since there is some slack between flights 1-2 and 3-4 to absorb the delay, flight 5 is not delayed and flown as scheduled. The minimum turn-around-time of 30 minutes (for this aircraft) is respected between all flights.

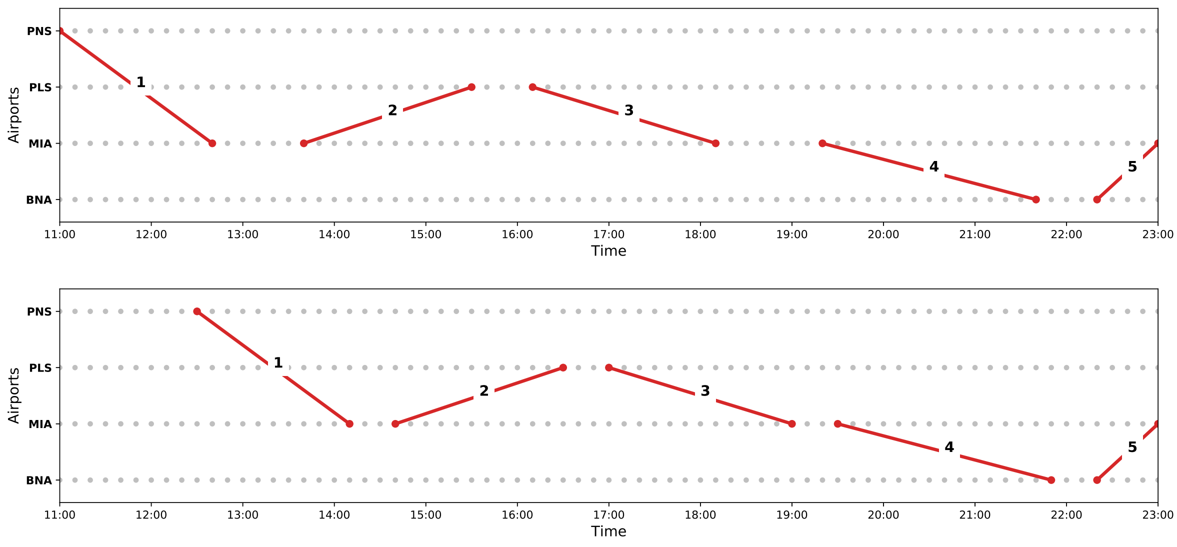


Figure 7.2: Optimal recovery solution given the delay of flight 1. [Top] Original schedule. [Bottom] Recovered flight schedule.

The optimizer found the solution within 2 seconds, with an objective function value of $29,463. Table 7.1 shows the cost breakdown for the recovery solution, which have been calculated by hand from the recovery solution. The cost of the solution is $29,466, which is $3 more than the solution found by the optimizer, this is due to rounding in the optimizer, where cost are rounded to the nearest integer. In the original schedule, no flights are delayed, hence the objective value of the optimizer consists of only the direct operating cost (DOC). The DOC for this aircraft type, Embraer 170 (E70), equals $29,90/min. The cost of the disruption, consisting of the soft cost and hard cost, equals $11,227. Since all flights are flown by the original aircraft, and the aircraft reaches it's final destination, no penalty cost are induced.

Table 7.1: Disruption cost breakdown for recovery solution shown in Figure 7.2

Flight	Duration [min]	Pax. Economy	Pax. Business	Delay [min]	Soft Cost	Hard Cost	DOC
1	100	60	8	90	$5.633	$840	$2.990
2	110	60	8	60	$2.356	$840	$3.289
3	120	60	8	50	$1.519	$1	$3.588
4	140	60	8	10	$38	$1	$4.186
5	140	60	8	0	-	-	$4.186
Total					**$9.545**	**$1.682**	**$18.239**

Two aircraft, one delay

The following verification scenario has the same aircraft as the previous scenario, with the same flight schedule. However, here, another aircraft has scheduled flights within the time window as well. Flight 1, from PNS to MIA, is delayed by 100 minutes. Figure 7.3 shows the original flight schedule in the top graph and the optimal recovered schedule in the bottom graph. It can be seen that, instead of delaying the flight string of aircraft 1, a tail swap between the two aircraft occurs on ORD. The minimum turn-around-time of 30 minutes (for both aircraft) is respected between all flights. Moreover, the tail swap limit of 180 minutes before scheduled departure is respected as well.

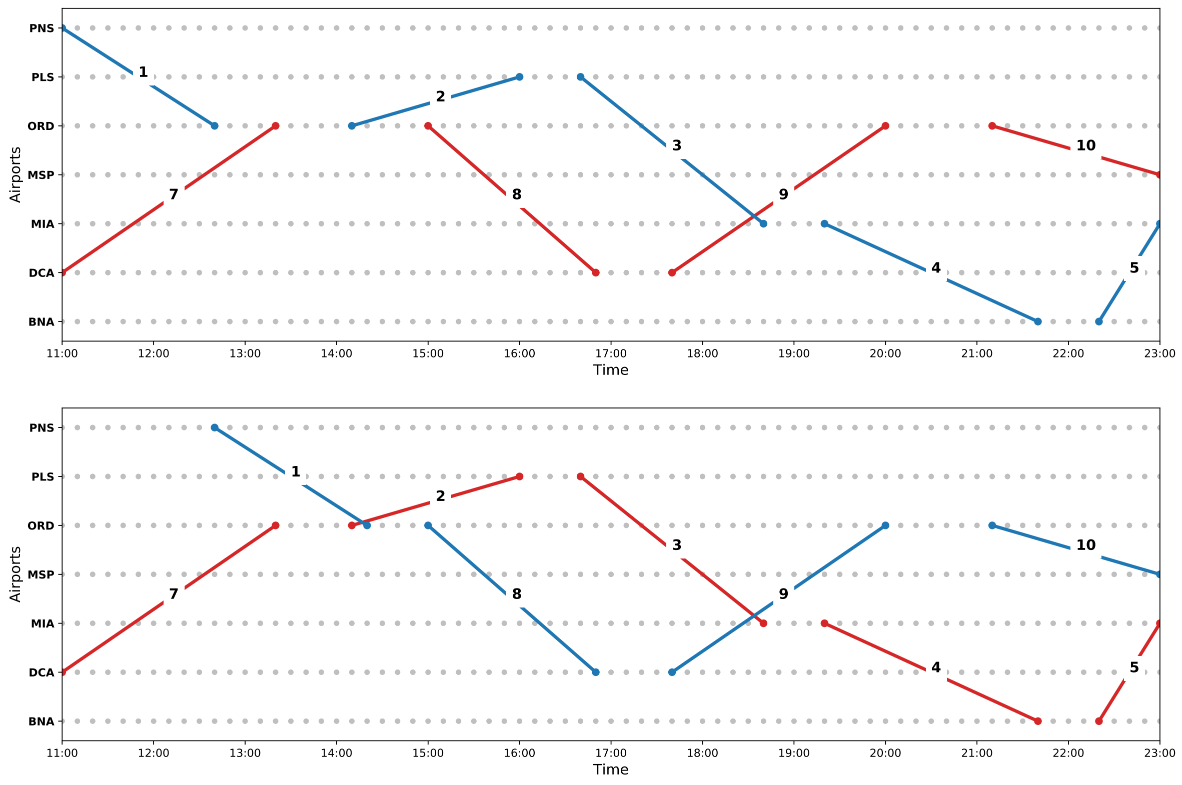


Figure 7.3: Optimal recovery solution given the delay of flight 1. [Top] Original schedule. [Bottom] Recovered flight schedule with tail swap.

The optimizer found the solution within 2 seconds, with an objective function value of $47,607. Table 7.2 shows the cost breakdown for the recovery solution, which have been calculated by hand from the solution. Both aircraft are the same type, Embraer 170 (E70), for which the direct operating cost equals $29,90/min. The cost of the solution is $47,607. The cost of the disruption, which is a sum of the soft cost, hard cost and penalty cost, equals $14,717. The penalty cost are induced on all flights that have been tail swapped and equals $1000 per swap in this scenario.

Table 7.2: Disruption cost breakdown for recovery solution shown in Figure 7.3

Flight	Duration [min]	Pax. Economy	Pax. Business	Delay [min]	Soft Cost	Hard Cost	Penalty Cost	DOC
1	100	60	8	100	$6.877	$840	-	$2.990
2	110	60	8	0	-	-	$1.000	$3.289
3	120	60	8	0	-	-	$1.000	$3.588
4	140	60	8	0	-	-	$1.000	$4.186
5	140	60	8	0	-	-	$1.000	$4.186
7	140	50	8	0	-	-	-	$4.186
8	110	60	8	0	-	-	$1.000	$3.289
9	140	60	8	0	-	-	$1.000	$4.186
10	100	60	8	0	-	-	$1.000	$2.990
Total					**$6.877**	**$840**	**$7.000**	**$32.890**

Connecting passengers - extension
Section 4.1 presented the Connecting Passenger Matrix developed by Vink et al. (2019) and the extension of this research where the STD of outbound connection flights is allowed if the inbound flight is disrupted and there are connecting passengers between the inbound and outbound flight. The verification of the original Connecting Passenger Matrix is presented by Vink et al. (2019). This verification scenario shows the impact of the extension, i.e. the impact of delaying connecting outbound flights on the objective function value.

In the original flight schedule the following flight connections are present:

Flight 1	Flight 2	Origin	Transfer	Destination	Economy Pax.	Business Pax.
4528	4670b	BTV	DCA	PHL	4	1
4670b	4606a	DCA	PHL	BDL	6	2

Figure 7.4 shows the time space graphs for four different situations:

- [Top] The original flight schedule.
- [Middle-Top] The optimal recovery solution without the connecting passenger matrix extension and where Flight ’4528’ is delayed 40 minutes. Here only the flight string of the disrupted aircraft is delayed. Flight 4670b is flown as scheduled and the connecting passengers from flight 4528 to flight 4670b will miss their connection.
- [Middle-Bottom] The optimal recovery solution with the connecting passenger matrix extension and where Flight ’4528’ is delayed 40 minutes. Here, next to the flight string of the disrupted aircraft, flight 4670b is delayed 30 minutes. This allows the connecting passengers from flight 4528 to make their connection to PHL. The connecting passengers from flight 4670b to 4606a will also make their connection. The disruption cost breakdown for this recovery solution is presented in Table 7.4.
- [Bottom] The optimal recovery solution with the connecting passenger matrix extension and where Flight ’4528’ is delayed 60 minutes. Here flight 4670b is not delayed, since the delay necessary to guarantee the connection for passengers on flight 4528 will also mean that passengers connecting from flight 4670b to 4606a will miss their connection.The disruption cost breakdown for this recovery solution is presented in Table 7.5.

In the first verification situation, which is shown in the Middle-Top graph in Figure 7.4, the optimizer found an objective function value of $24,385. The disruption cost breakdown for this recovery solution is presented in Table 7.3. All flights, except for flight 4606a, are flown by the same type, Embraer 170 (E70), for which the direct operating cost equals $29,90/min. Flight 4606a if flown by an Embraer 175 (E75) for which the direct operating cost equals $25,92/min. The cost of the solution is $24,387 and the cost of the disruption equals $7,921, which consists of the soft cost and the penalty cost. The penalty cost comes from the connecting passengers that miss their transfer because of the disruption. As previously mentioned, for connecting passengers that miss their transfer, the penalty cost will equal the maximum delay cost, if no alternative transfer exists. Since no alternative connection exists in this situation, the maximum soft cost of $822,58/pax and hard cost of $250 are induced per connecting passenger.

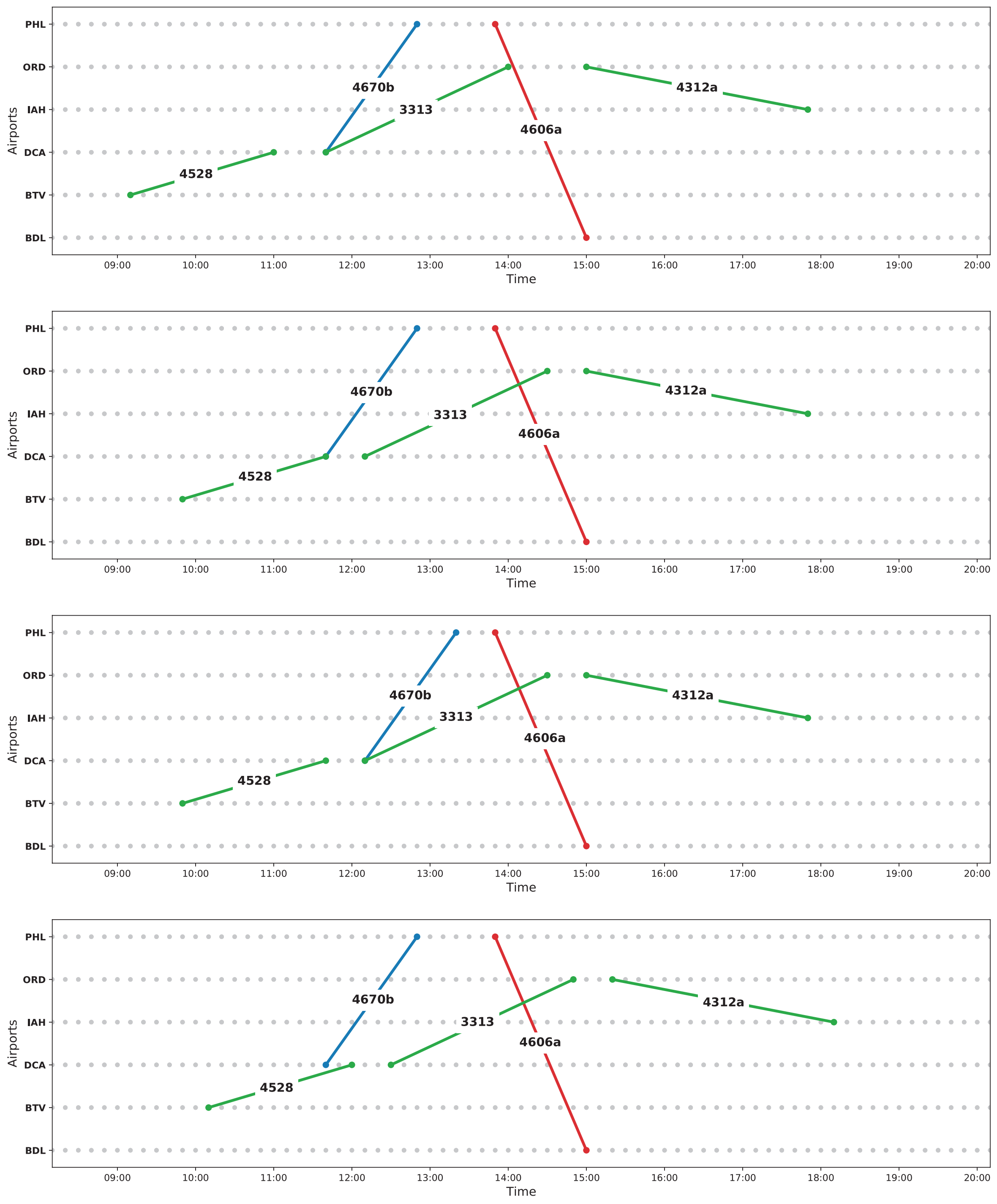


Figure 7.4: Optimal recovery solution given the delay of flight 4528. [Top] Original schedule. [Middle-Top] Recovered flight schedule without delaying outbound connecting flights. [Middle-Bottom] Recovered flight schedule with delaying outbound connecting flights. [Bottom] Recovered flight schedule with delaying outbound connecting flights, but where delay of flight 4528 is too long.

Table 7.3: Disruption cost breakdown for recovery solution shown in Figure 7.4 [Top] and [Middle-Top]

Flight	Duration [min]	Pax. Economy	Pax. Business	Delay [min]	Soft Cost	Hard Cost	Penalty Cost	DOC
4528	110	26	1	40	$229	-	$7.508	$3.289
3313	140	24	4	30	$185	-	-	$4.186
4312a	170	64	6	0	-	-	-	$5.083
4670b	70	23	2	0	-	-	-	$2.093
4606a	70	29	3	0	-	-	-	$1.814
Total					**$413**	**$0**	**$7.508**	**$16.465**

In the second verification situation, which is shown in the Middle-Bottom graph in Figure 7.4, the optimizer found an objective function value of $\$17,090$. The disruption cost breakdown for this recovery solution is presented in Table 7.4. The direct operating cost and aircraft types are equal to the first verification situation. The cost of the solution is $\$17,100$ and the cost of the disruption equals $635, which consists of the soft cost only. In this situation, the outbound connecting flight, 4670b, is delayed by 30 minutes, inducing some soft cost. However, since the connecting passengers can make their connection, the penalty cost is avoided. Hence, allowing outbound connecting flights to be delayed, result in lower disruption cost.

Table 7.4: Disruption cost breakdown for recovery solution shown in Figure 7.4 [Middle-Bottom]

Flight	Duration [min]	Pax. Economy	Pax. Business	Delay [min]	Soft Cost	Hard Cost	Penalty Cost	DOC
4528	110	26	1	40	$229	-	-	$3.289
3313	140	24	4	30	$185	-	-	$4.186
4312a	170	64	6	0	-	-	-	$5.083
4670b	70	23	2	30	$149	-	-	$2.093
4606a	70	29	3	0	-	-	-	$1.814
Total					**$635**	**$0**	**$0**	**$16.465**

For the final verification situation, which is shown in the Bottom graph in Figure 7.4, the optimizer found an objective function value of $\$24,624$. The disruption cost breakdown for this recovery solution is presented in Table 7.4. The direct operating cost and aircraft types are equal to the previous verification situations. The cost of the solution is $\$24,624$ and the cost of the disruption equals $\$9,159$, which consists of the soft cost, hard cost and penalty cost. In this situation outbound connecting flight delays are allowed. However, since the inbound connecting flight, 4528, is delayed by 60 minutes, flight 4670b would need to be delayed by 50 minutes. This would cause the connecting passengers from flight 4670b to 4606a to miss their transfer. Since there are more connecting passengers on that transfer, the cost of those passengers missing their transfer is higher. Therefore, flight 4670b is not delayed and the penalty cost is induced.

Table 7.5: Disruption cost breakdown for recovery solution shown in Figure 7.4 [Bottom]

Flight	Duration [min]	Pax. Economy	Pax. Business	Delay [min]	Soft Cost	Hard Cost	Penalty Cost	DOC
4528	110	26	1	60	$617	$220	$7.508	$3.289
3313	140	24	4	50	$651	-	-	$4.186
4312a	170	64	6	20	$163	-	-	$5.083
4670b	70	23	2	-	-	-	-	$2.093
4606a	70	29	3	0	-	-	-	$1.814
Total					**$1.431**	**$220**	**$7.508**	**$16.465**

7.1.3. Verification of Decision Support System

The previous section presented the verification of the optimization model using several examples. This section will discuss the verification of the full decision support system (DSS), i.e. the optimization model combined with the machine learning classifier and the sub-network selection algorithm. The results of the full DSS are compared with the optimization model without classifier and sub-network selection algorithm to ensure the results are similar or equal.

The verification scenario consisted of 50 iterations in the disruption dataset, which captured 18 hours of operations. These iterations cover 79 disruptions of all types and causes. Out of these disruptions 8% were ”Aircraft unavailabilities” and 92% were ”Flight Delays”. The flight delay causes were distributed as follows: 60% Airline, 30% National Air System, 7% Weather Groups and 3% Weather. Chapter 8 will further elaborate on the definitions of these causes.

For the results of the ”Optimizer”, the disruptions were solved per aircraft family and no aircraft were discarded, hence giving the baseline results. For the ”Full DSS”, the disruptions were also solved per aircraft family. Here, the machine learning classifier was used to rank all candidate aircraft per disruption and the sub-network selection algorithm was used to select the top 30% ranking candidates.
Table 7.6 shows the average results over 50 iterations for the full DSS compared to the optimization model without classifier. The full results of the DSS verification are published online and can be found in Hassan (2019).

Table 7.6: Comparison of the averaged results of 50 iterations for the full DSS compared to the optimization model without classifier and sub-network selection algorithm.

	Time [s]	Aircraft	Cost	OTP	DPC [pax]	APD [min/pax]	XP	XF	MCP	CV
Optimizer	95	128	$14.930	99,4%	182	41	0	0	10	0
Full DSS	46	39	$14.930	99,4%	182	41	0	0	10	0

From the results it was found that the solution to 8 out of the 79 disruptions was non-trivial, i.e. other aircraft besides the disrupted aircraft were needed to find the optimal solution. In these non-trivial cases the Full DSS still found the optimal solution. Table 7.6 and the full results show that the DSS is able to find the same solution as the baseline optimizer for each of the 50 iterations. This shows that the classifier is able to correctly rank the candidate aircraft in the top 30% for these iterations. Section 8.3 will elaborate on the effect of changing the sub-network selection algorithm.

7.2. Validation

Validation of the decision support system (DSS) was performed by two experts from industry. Several results of the case study, discussed in Chapter 8, were discussed with Arjen Blom, former flight captain and Director of Operations Management at KLM Royal Dutch Airlines. Mr. Blom has over 15 years of experience with airline disruption management. Results of the case study were discussed to determine whether the recovery solutions suggested by the DSS are realistic. Furthermore, the extensions researched in this thesis and their added value were discussed. The conclusions of the discussion are:

- The recovery solutions found by the DSS for the Aircraft Recovery Problem are realistic and comparable to solutions found by AOCC controllers. However, the exclusion of Crew Recovery and Passenger Itinerary Recovery, limit the applicability of the DSS in reality, since the constraints imposed by those problems significantly impact the recovery solution. Improving on the recovery model proposed by Vink et al. (2019), it is appreciated that the minimum tail swap limit is respected in the recovery solutions.

- The extension to the connecting passenger matrix, where outbound connecting flights can be delayed, is of added value to the DSS. Given the sometimes large number of connecting passengers on a flight, it would be hard for an AOCC controller to investigate opportunities for outbound flight delay. Automating this process and presenting suggestions to the controller is therefore appreciated. Currently the DSS investigates opportunities of outbound flight delay even if only a single connecting passenger is on the inbound flight. In reality, outbound flights are only delayed if more than 30 connecting passengers are on the inbound flight. Therefore it is advised to include a threshold to the DSS.
- The way the recovery solution is presented by the DSS is not easy to comprehend. More effort should be made to intuitively show why certain aircraft are part of the recovery solution and how they fit in the solution.

Additional investigation is required to fully validate the DSS. Preferably, the DSS would run side by side in a real AOCC, where the DSS would generate recovery solutions in parallel to the current operations. The solutions of the current process would need to be compared to investigate the recovery solutions generated by the DSS.

Besides the results, the assumptions of Table 5.2 were discussed with Mr. Blom and Frank Charlier. Mr. Charlier works for ORTEC and has over 12 years of experience with airline operations at KLM. The goal of these discussions was to determine the validity of the assumptions and the applicability to real airline operations. The conclusions of these discussions are:

- [1 Equal flight time for all aircraft types: Valid] The impact of assigning a different aircraft to a flight on the flight duration is minimal. Especially on short to medium haul flights this difference is negligible. If an aircraft was to arrive earlier than scheduled, it is likely that it will be placed in a hold position or will need to wait on the airport until the reserved gate slot becomes available.
- [2 No cruise speed changes to recapture time during flight: Valid] The influence of cruise speed on the flight duration is limited. This will likely have an effect on flights over 5 hours, where it could have an impact of 15 minutes. For short to medium haul flights, changing the cruise speed will not make a noticeable difference. Moreover, given the case study for Delta Airlines, the US airspace is quite congested, which makes it difficult to change the cruise speed. Furthermore, the daily fuel price will influence the willingness of airlines to change the cruise speed, which is a complicated effect to capture in the DSS.
- [3 Crew is always available: Not Valid] Besides aircraft, crew is a very limited resource for airlines. Moreover crew legalities are complicated and severely limit the employability and flexibility of crew. In daily operations and disruption management, crew restrictions are likely to impact the available recovery options. Hub-and-spoke carriers, like Delta and KLM, have more flexibility and recovery options available on their hubs, since reserve crew is usually available. This assumption would be more valid for budget airlines like Ryanair and Easyjet, since they usually operate a single aircraft type.
- [4 Airport capacity and slots are always available: Semi Valid] Regulations regarding airport closures and operating flights after certain hours, are very airport dependent. In the US, most airports do not close down during the night and flights can be operated 24 hours a day. However, especially on larger airports (e.g. JFK or LAX), the number of flight movements is maximized. Hence, arriving after the scheduled slot is not always possible. Several airports in Europe and Asia do close down during the night, which should be taken into account when implementing the DSS for those regions.
- [5 Time discretization provides sufficient detail: Valid] ICAO uses 15 minute time steps and most flight times are published with time steps of 5 minutes. Especially on larger airports, aircraft can depart every 30 seconds. However, for disruption management

and flight rescheduling a time step of 10 minutes is deemed sufficient. Flight crew and air traffic control will be responsible for finding a more specific departure or arrival slot.

- [6 TAT is a constant per aircraft type: Semi Valid] AOCC controllers regularly change the number of ground personnel to speed up TAT after a delay. The decrease in TAT that can be achieved depends on the aircraft type and the resources available on the turn around airport. However, the TAT is usually decreased by no more than 10 minutes.
- [7 Maintenance is not considered: Not Valid] Maintenance requirements of aircraft are heavily regulated and for most checks the maintenance schedule is fixed. This maintenance schedule limits the recovery options available, since it decreases the flexibility of changing flight strings of aircraft.
- [8 Delays are not updated: Semi Valid] In real operations, delay updates are regularly available. However, once a flight has been cancelled or passengers have been informed of a delay, this will not be undone. Delay times can be increased when disruptions grow in severity, e.g. 20 minute delay to 40 minute delay, but the reverse almost never happens.

7.3. Concluding Remarks on Verification and Validation

This chapter presented the verification and validation of the recovery solutions found by the DSS. For verification, several small examples were tested for which the solution was verified by hand. Besides proving that the optimization model behaves properly, the examples highlight the decisions and trade-offs made by the DSS. One of the examples verified the extension of this research to the Connecting Passenger Matrix. This verification scenario showed the trade-off between delaying outbound connecting flights to ensure passengers can make their transfer and the consequences of that delay for the passengers on the outbound flight.

Validation of the DSS was performed by two experts from industry. In general, it was concluded that the model's results are valid and comparable to the solutions found by AOCC controllers. It was noted that the recovery solutions to the aircraft recovery cannot be interpreted without considering crew and passenger recovery, which were not part of the scope of this research. Next to the results, the assumptions were validated. While the majority of the assumptions are valid, it was concluded that the assumptions related to crew availability and maintenance were not valid and limit the validity of the model. Recommendations regarding the improvements of the model are presented in Chapter 9.

8 Case study: Delta Airlines

The previous chapter discussed the verification and validation of the decision support system. This chapter will present a case study, where the DSS is tested on the domestic network of Delta Airlines.

Delta Airlines is the second largest airline in the world and operates a worldwide hub-and-spoke network. The airline performs roughly 2400 domestic flights per day and serves 150 destinations in the US. Delta operates a fleet of over 800 aircraft and has the largest Boeing 717, Boeing 757, Boeing 767, McDonnell Douglas MD-88, and McDonnell Douglas MD-90 fleets in the world. The Airbus aircraft were added to the fleet after Delta merged with Northwest Airlines. Performing a case study on the network of Delta airlines is interesting because of the larger fleet size (roughly six times), the fleet composition (26 aircraft types instead of two) and the size of operations (roughly five times more daily flights) than the airline used in the case study performed by Vink et al. (2019).

In Section 8.1 the dataset generation for the case study is discussed. Section 8.2.2 presents the case study that was performed on the Delta Airlines dataset. In that section the results of the decision support system (DSS) will be compared with the selection heuristic developed by Vink et al. (2019). Section 8.3 will present the sensitivity analyses.

8.1. Delta Airlines Dataset Generation

Chapter 4.1 and Appendix B discuss the input data needed for the decision support system (DSS). All data was obtained from public sources and other researchers. The following sections will present how the data was obtained and manipulated to generate the required input data. An overview of the data processing flowchart is given in Appendix H.

8.1.1. Flight Schedule Information

Flight schedule information for Delta Airlines was downloaded from the 'Reporting Carrier On-Time Performance' Database from the United States Department of Transportation - Bureau of Transportation Statistics. (2018b). Besides flight information such as scheduled and actual departure and arrival times, the database includes information on canceled and diverted flights and the causes of delay and cancellation. The full flight schedule for Q1 2015 was downloaded as well as supporting look-up tables for the airline and airport codes. The following processing steps were performed sequentially:

1. Convert local scheduled departure and scheduled arrival times (hhmm) to Coordinated Universal Time (UTC) datetime format (yyyy-mm-dd hh:mm).
2. Correct the UTC times for Daylight Savings Time (DST)
3. Calculate the Flight Time as $STA - STD$ in UTC.

4. Repair timing errors. The downloaded flight data did not come together as a correct flight schedule. When looking at the flights per tail number, some flight strings were broken. This was primarily due to timing errors (e.g. Flight 2 of Aircraft A departs before Flight 1 of Aircraft A arrives). These errors were corrected by recalculating the departure (and arrival) times such that the schedule would adhere to minimal Turn-Around-Times.
5. Add missing flights. Not unlike the previous point, the downloaded flight schedule did not include all flights by an aircraft. The data only showed flights that carried passengers, hence ferry flights were not included. This resulted in broken flights strings for most aircraft. To overcome this, ferry flights were added where necessary.

After these processing steps, the schedule data consisted of full flight strings for each aircraft in the schedule. Next, the dataset was split into a flight schedule dataset and a disruption dataset. The disruption dataset was further processed, which will be discussed in Section 8.1.4. An overview of the information consisted in the flight schedule dataset is given in Table 8.1.

Table 8.1: Flight schedule dataset information overview

Property	**Description**
Unique ID	Unique 6 digit identification number to help identify the flight throughout the process and recovery process.
Flight Number	The Delta Airlines flight number, e.g. DL2336. The flight number is not unique.
Tail Number	The unique tail number of the aircraft scheduled on the flight, e.g. N958DN.
Origin Airport	The three letter IATA code of the origin airport of the flight
Destination Airport	The three letter IATA code of the destination airport of the flight
Scheduled Time of Departure (STD)	The Scheduled Time of Departure (STD) in Coordinated Universal Time (UTC) of the format yyyy-mm-dd hh:mm
Scheduled Time of Arrival (STA)	The Scheduled Time of Arrival (STA) in Coordinated Universal Time (UTC) of the format yyyy-mm-dd hh:mm
Delay	The delay of the flight in minutes. Default is 0. This number will be updated during the recovery process.
Cancelled	Boolean variable to identify if the flight has been cancelled. Default is False. This flag will be updated during the recovery process.
Passengers Economy Class	The number of passengers in economy class that will be on the flight. See Section 8.1.3
Passengers Business Class	The number of passengers in business class that will be on the flight. See Section 8.1.3
Load Factor	The passenger load factor of the flight as an percentage.

Flight Schedule Statistics

This section discusses some key statistics about the flight schedule dataset. In total there are 197 thousand Delta flights in Q1 2015, with an average of 2164 flights a day. The network consists of 147 airports, which includes 8 hubs. Hub-to-spoke flights make up 86% of the flights, 13% are from a hub to another hub, and finally 1% of flights is from an outbound airport to another outbound airport. That last category includes flights from what Delta calls ”Focus Cities”, which primarily cater the local market instead of connecting passengers. Figure 8.1 shows the histogram of the flight duration. The average flight duration is 2 hours and 30 minutes.

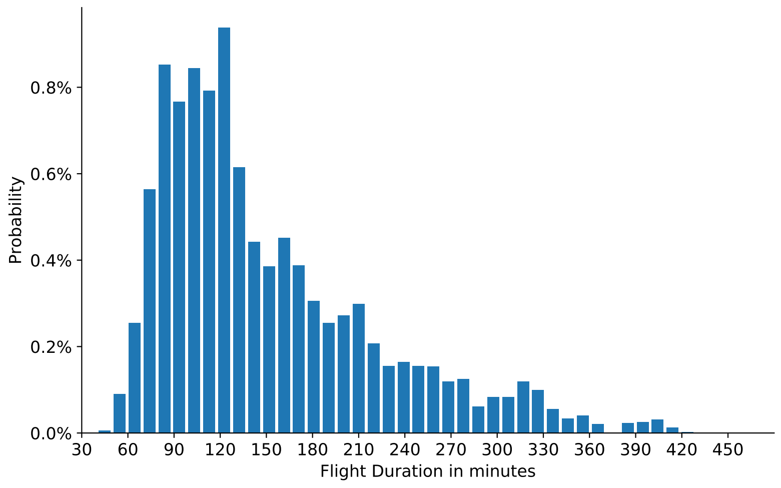


Figure 8.1: Histogram of the flight duration in minutes. Average flight duration is 2 hours and 30 minutes.

8.1.2. Fleet Information

The flight schedule dataset from the previous section included the tail numbers assigned to each flight, however aircraft information for these tail numbers was not included. Aircraft manufacturer and type information belonging to the unique tail numbers included in the dataset was obtained from the N-Number Database of the United States Federal Aviation Administration (2018).

For all unique aircraft types obtained from the FAA database, additional information was collected from the Fleet section on the Delta Website[1]. This information included the range and seat capacity per class. Delta Airlines knows three cabin classes: 'Economy Class', 'Delta Comfort+' and 'First Class'. For this research the seat capacity of 'Delta Comfort+' and 'Economy Class' were merged into 'Economy Class'. Information on the minimum Turn-Around-Time (TAT) required per aircraft type was obtained from discussions with Arjen Blom, former Captain and Director of Operations Management at KLM Royal Dutch Airlines.

The Direct Operating Cost (DOC) per aircraft type are estimated based on the Air Carrier Financial Database of the United States Department of Transportation - Bureau of Transportation Statistics. (2018a). This database includes information on the number of registered flying hours and operating expenses per aircraft, from which an estimate on the average DOC per flying hour can be derived.

Fleet Statistics

Table 8.2 shows the properties of the extracted fleet for Delta Airlines. The table shows a total fleet of 827 aircraft, from 8 aircraft families. The three largest families are: McDonnell Douglas (32%), Boeing 757 (17%) and Boeing 737 (17%).

[1]https://www.delta.com/us/en/aircraft/overview

Table 8.2: Delta Airlines fleet properties

Model	Family	Aircraft	Passenger Capacity		Range	DOC/Hour	TAT
			Bus.	Eco.	[Miles]	[$]	[min]
MD-88	MD	117	16	133	2045	$5.062	45
717-200	MD	85	32	78	1510	$4.571	30
737-832	737	73	16	144	2930	$4.463	45
757-232	757	70	20	179	4334	$4.886	50
767-332	767	58	30	221	3515	$5.679	60
MD-90-30	MD	65	16	142	1992	$5.087	45
A319-114	A320	57	12	120	2015	$4.980	40
737-932ER	737	50	20	160	2870	$4.525	45
A320-212	A320	42	16	144	3000	$4.604	40
757-251	757	31	20	179	4334	$4.886	50
A320-211	A320	27	16	144	3000	$4.604	40
A330-323	A330	21	34	259	5343	$6.571	45
767-432ER	767	20	40	206	6336	$6.377	70
757-351	757	16	24	210	3228	$5.086	50
747-451	747	14	48	328	7365	$10.026	85
A330-223	A330	11	34	200	6536	$6.118	45
737-732	737	10	12	112	2925	$4.572	45
757-2Q8	757	10	20	179	4334	$4.886	50
777-232LR	777	10	37	254	10375	$7.162	70
767-332ER	767	9	30	221	5980	$5.679	60
757-231	757	8	20	179	4334	$4.886	50
777-232	777	7	37	254	8542	$7.162	70
767-3P6	767	6	26	200	6221	$5.679	60
757-212	757	4	22	153	4334	$4.886	50
A330-302	A330	4	34	259	5343	$6.571	45
757-26D	757	1	20	179	4344	$4.886	50
767-324	767	1	30	221	3515	$5.679	60

8.1.3. Passenger Information

Barnhart et al. (2014) developed methodologies to model historical travel and delay for U.S. domestic passengers. Their dataset included information on the itineraries of (connecting passengers) and was made available for this research. From this dataset, information on the number of passengers on a certain origin-destination (O-D) pair, either on direct flights or via multiple flight legs connecting at other airports, could be determined. The following processing steps were performed sequentially:

1. Since the times in the flight schedule were corrected for the timing errors, the entries in the passenger dataset needed to be matched to the new STD and STA times.

2. The passenger dataset did not contain flight numbers, these had to be extracted from the flight schedule. For roughly 90% of the entries, matching flight numbers could be found based on the origin and destination airports and the departure and arrival times.

3. For the entries that could be matched, the number of O-D and connecting passengers per flight was determined. From this data the Passenger Load Factors (LF) per flight leg could be determined. This data was corrected for outliers (LF > 100%). The load factor distribution per flight leg was used to generate passenger numbers for the 10% entries that could not be matched in the previous step. This ensured realistic passenger numbers for all the flights in the flight schedule.

4. Only the total number of passengers per flight could be determined from the Barnhart et al. (2014) dataset. Based on the Business/Economy seat ratio a random number generator was used to determine the number of economy and business passengers per flight. The average probability that a passenger is a business passenger is 12,7% for the dataset.

After performing the processing steps, a dataset with all connecting passengers per itinerary was exported and the number of economy and business passengers per flight data was added to the flight schedule. An overview of the information consisted in the connecting passenger itinerary dataset is given in Table 8.3.

Table 8.3: Connecting passenger itinerary dataset overview

Property	**Description**
Flight Number 1	The Delta Airlines flight number of the first flight leg.
Flight Number 2	The Delta Airlines flight number of the second flight leg.
Origin Airport	The origin airport of the first flight leg.
Connecting Airport	The airport where the passengers transfer from flight leg one to flight leg two.
Destination Airport	The destination airport of the second flight leg.
STD Flight Leg 1	The Scheduled Time of Departure (STD) in Coordinated Universal Time of the first flight leg.
STA Flight Leg 1	The Scheduled Time of Arrival (STA) in Coordinated Universal Time of the first flight leg.
STD Flight Leg 2	The Scheduled Time of Departure (STD) in Coordinated Universal Time of the second flight leg.
STA Flight Leg 2	The Scheduled Time of Arrival (STA) in Coordinated Universal Time of the second flight leg.
No. of Economy Passengers	The number of economy passengers on the itinerary.
No. of Business Passengers	The number of business passengers on the itinerary.

Passenger Statistics

This section will present some key statistics about the passenger dataset. Figure 8.2 shows the *passenger load factor* or occupancy rate histogram for all Q1 2015 flights. The average passenger load factor is 75.9%, meaning that, on average, aircraft are 75.9% full. The peak at 100% indicates that roughly 10% of all Delta flights were full or overbooked. The 100% passenger load factor peak seems plausible, however the 'dip' at 95% does not seem realistic. A smoother increasing trend from 80-100% is expected to be realistic. This could be due to small errors in the received passenger data or due to truncation of overbooked flights. It is expected that this will not influence the research objective of this thesis negatively.

Out of all passengers that flew with Delta Airlines in Q1 2015, 26% were connecting passengers. Figure 8.4 shows the histogram of the number of connecting passengers per flight. On average, each flight transports 40 connecting economy and 6 connecting business passengers. Figure 8.3 shows the histogram of the number of passengers transferring per connection, e.g. the number of passengers transferring from inbound Flight A to outbound Flight B. Clearly, roughly 50% of connections are for a single passenger.

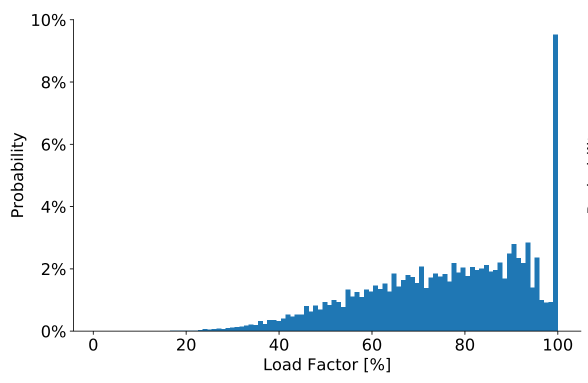


Figure 8.2: Passenger Load factor distribution
Mean = 75.9, std = 17.45

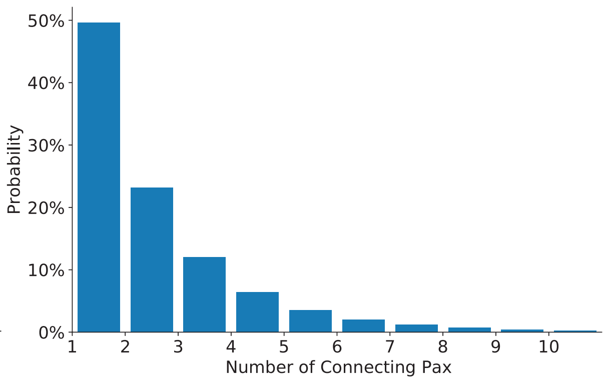


Figure 8.3: Histogram of the number of connecting passengers per connection

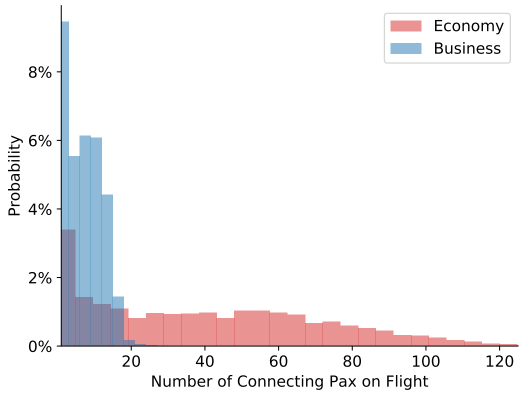


Figure 8.4: Histogram of the number of connecting passengers per flight

8.1.4. Disruption Information

This section discusses the processing steps that were taken to transform the delay dataset, obtained as described in Section 8.1.1, to a dataset that contains disruptions. For the purpose of this research only root causes were taken into account. For example, the delay dataset contained a category 'Late Aircraft Delay'. Since there must have been a previous disruption that caused the aircraft to be late, this category of delay was not taken into account. The same logic hold for the aircraft cancellations, since these were caused by another disruption or delay, e.g. an aircraft unavailability or weather delay.

The delay dataset only holds information on the duration of the delay in minutes and the cause(s) of delay. The delay cause categories are: 'Carrier Delay', 'Late Aircraft Delay', 'National Air System (NAS) Delay', 'Security Delay' and 'Weather Delay'. Definitions of these categories are presented in Appendix I. The dataset contains no information regarding the time at which the Delta AOCC was notified about the disruption, the Time Found Out (TFO), thus this time needs to be generated and added to the dataset. In order to obtain realistic disruptions, the following processing steps were taken:

1. To obtain aircraft unavailabilities, a logic had to be created to extract these disruptions from the delay dataset. After discussions with Mr. Blom, it was decided that 'Airline Delays' with a duration over 120 minutes would be reclassified as aircraft unavailabilities. This logic was based on the category description contained in the BTS database and the experience of Mr. Blom which led him to believe there would not be many other reasons for an airline delay to take over 2 hours.

2. The delay dataset contains data on weather delays, however only single flight delays are contained. In reality, weather may impact several flights at once. Especially with heavier weather conditions, the flow rate at an airport can be reduced or all aircraft can be grounded. To capture the 'group' effect of these heavier weather conditions, several entries in the delay dataset need to be clustered. This is done with the following logic:
   - Group flights with the same origin airport, with their STD's within 2 hours of each other and where $'NASDelay' +' WeatherDelay' \geq 50\% \cdot TotalDelayofFlight$ and where $TotalDelay \geq 15min$.
   - Divide Groups into subgroups with a duration of 3 hours. These subgroups will have the same Time Found Out (TFO), which is a simulated time that describes the time an AOCC controller is notified of the disruption.
3. Add all remaining single flight or single aircraft disruptions. If a delay is caused by multiple factors, such as 'NAS' and 'Airline", these causes will be added as separate disruptions.

An Airline Operations Control Center (AOCC) does not know when disruptions will happen, rather, they are notified of each disruption when it is discovered. For example, if an aircraft is hit by birds during landing, it may need some maintenance before continuing with the next flight. The AOCC controller will be notified by the flight crew after landing, which could be just 40 minutes before the next flight is scheduled to depart. To simulate this discovery of disruptions, each disruption (or weather subgroup) will be assigned a random generated Time Found Out (TFO), based on the delay cause. The following probability distributions, which are based on discussions with Mr. Blom, are used to generate these TFOs:

Beta probability density function:

$$f(x, \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}$$

National Air System Delays
with $\alpha = 5$ and $\beta = 2.5$
scaled to $t_{min} = 20$ and $t_{max} = 720$

Weather Delays
with $\alpha = 5$ and $\beta = 1.5$
scaled to $t_{min} = 30$ and $t_{max} = 720$

Security Delays
with $\alpha = 5$ and $\beta = 1.5$
scaled to $t_{min} = 10$ and $t_{max} = 120$

Uniform probability density function:

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } a \leq x \leq b \\ 0 & \text{for } a < x > b \end{cases}$$

Airline Delays
with $t_{min} = 10$ and $t_{max} = 120$

Aircraft Unavailabilities
with $t_{min} = 40$ and
$t_{max} = max(\text{time since STD previous flight}, 240)$

The TFO distributions show that NAS and Weather delays are usually known well in advance. Even though the distributions range between 20-720 minutes and 30-720 minutes, they are skewed towards the 720 minutes. Airline and Security delays are found out much closer to the Scheduled Time of Departure of the flight. Most Airline delays are caused by ground operations during turn-around, which typically starts 45-60 minutes before departure. Most Security delays cannot be forecasted and impact a flight during turn-around. Finally, disruptions caused by Aircraft Unavailabilties are typically reported to the AOCC after the departure of the previous flight and are known before turn-around operations start.

Disruption Statistics

This section will present some key statistics about the disruptions dataset. In total there are 29.4 thousand disruptions for Delta Airlines in Q1 2015 included in the dataset. Figure 8.5 shows the share of the different causes in terms of occurrence, while Figure 8.6 shows the share of the different causes in terms of delay minutes. Interesting to notice is that aircraft unavailabilities represent only 3% of the delays in occurrence but are responsible for 20% of delay in terms of duration. Detailed statistics on the disruptions per type and cause are given in Appendix J.

Figure 8.7 shows the histogram of the disruption duration in minutes. The average disruption has a duration is 37 minutes and a standard deviation of 65 minutes. Figure 8.8 shows the histogram of the number of disruptions per day. The average number of disruptions per day in Q1 2015 is 366, with a standard deviation of 219.

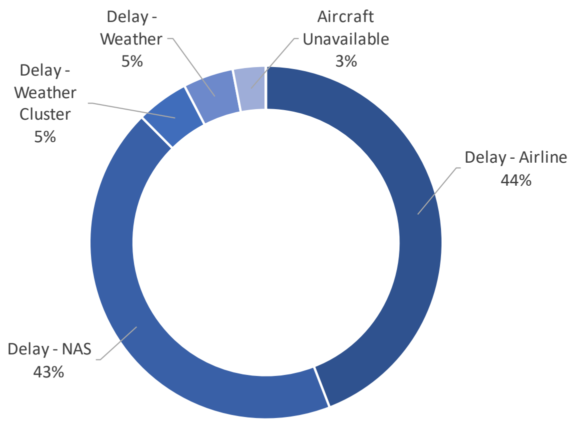


Figure 8.5: Disruption Cause Share in terms of Occurrence

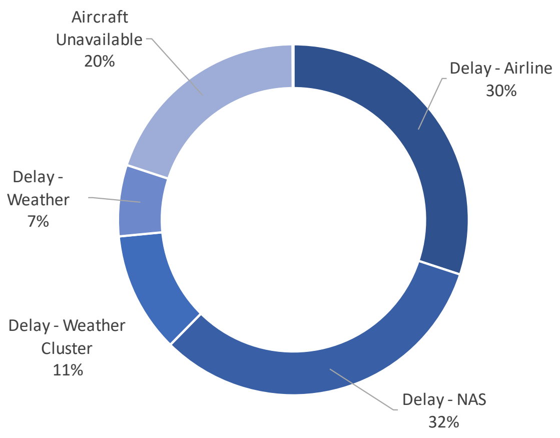


Figure 8.6: Disruption Cause Share in terms of Delay Minutes

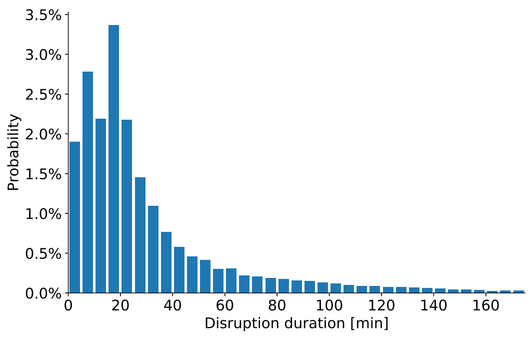


Figure 8.7: Disruption duration histogram.
Average duration: 37 minutes.

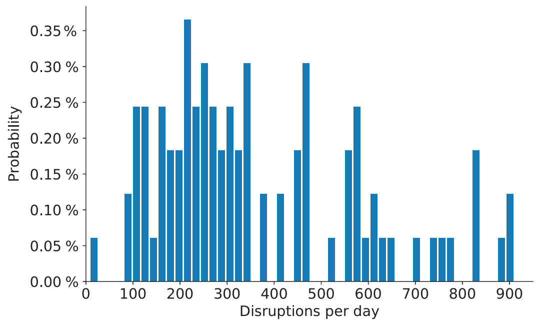


Figure 8.8: Disruptions per Day Histogram.
Average number of disruptions/day: 366

8.2. Case Study

The previous section presented how the Delta Airlines dataset was generated from public sources. From the dataset, a case study is formulated to evaluate the performance of the DSS. This section will elaborate on the case study iterations and the results.

8.2.1. Introduction and Case Study Overview

The case study consists of 200 iterations, which equals roughly 1 day of disruptions in January. An iteration is a set of disruptions which have the same Time Found Out (TFO). These disruptions are covered in the feature dataset that was used to train the random forest classifier in the DSS. The sensitivity analyses, presented in the next section, will evaluate the performance of the DSS on 100 iterations in March, which were not used to train the random forest classifier. The 200 iterations of the case study consist of 367 runs, where one run solves the disruptions for one aircraft family. For example, if iteration 1 has one delay in the MD fleet and two delays in the A320 fleet, iteration 1 will have two runs. These 367 runs cover 565 disruptions: 556 flight delays and 9 aircraft unavailabilities.

Figures 8.9, 8.10 and 8.11 summarize the iterations of the case study. Figure 8.9 shows the share of the different disruption causes in terms of occurrence. Figure 8.10 shows the histogram of the disruption duration in minutes. The average disruption has a duration of 31 minutes, which is slightly lower than the average disruption duration for Q1 2015. Figure 8.11 shows the number of disruptions per iteration, i.e. the number of disrupted flights or aircraft. Table 8.4 shows the parameter values for the optimizer, DSS and the heuristic of Vink et al. (2019) that were used for the case study.

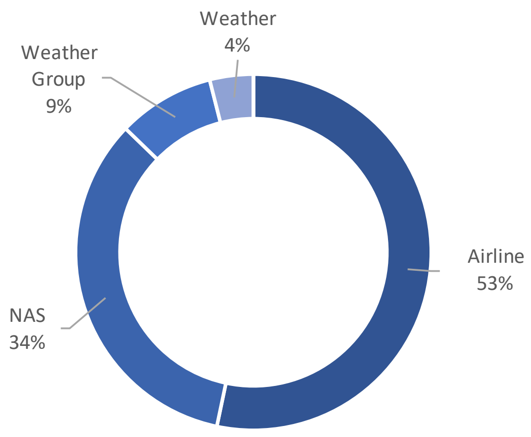


Figure 8.9: Case Study Disruption Cause Share in terms of Occurrence

Table 8.4: Parameters used for case study

Parameter	Value
TW length	12 hours
Time step	10 minutes
T_{swap}	3 hours
C_{canx}	\$250
$C_{C_{sch}}$	\$1.000
β_{buss}	3
Big M cost	\$1.000.000
Max Delay	8 hours
Fix exiting aircraft or type	Type
Sub-network selection	Top 50%
Heuristic test size k	10

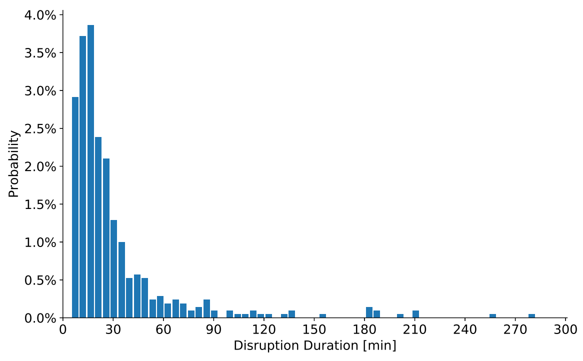


Figure 8.10: Case Study Disruption duration histogram. Average duration: 31 minutes.

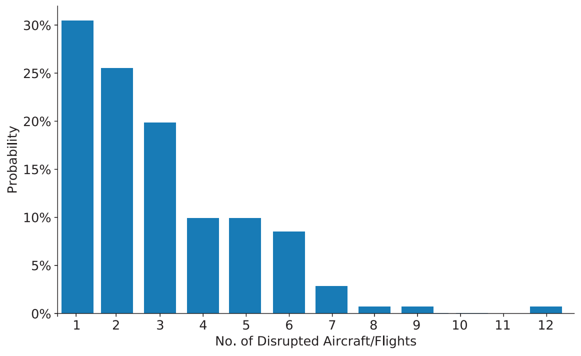


Figure 8.11: Case Study Disruptions per Iteration Histogram. Average number of disruptions/iteration: 3

To evaluate the performance of the DSS, four cases will be compared:

1. [Optimum] The optimization results where the disruption was solved per aircraft family without filtering any aircraft.
2. [Trivial] The trivial solution where only the disrupted aircraft are used to solve the disruption and no other aircraft are taken into account.
3. [Heuristic] The optimization results where the selection heuristic developed by Vink et al. (2019) creates several aircraft selections. Based on the number of undisrupted candidates C, the test size k and the number of disrupted aircraft d, the number of runs N required to process all candidate aircraft is determined. After the first run, which finds the trivial solution, N runs are performed, each with k+d aircraft. The k undisrupted aircraft are drawn from the list of candidate aircraft based on a sorting algorithm.
4. [Full DSS] The optimization results where a sub-network of aircraft is created with the random forest classifier and sub-network selection algorithm developed for this research.
5. [Full DSS-S] The optimization results where a sub-network of aircraft is created with the random forest classifier developed for this research. Based on the number of candidates in the sub-network C, the test size k and the number of disrupted aircraft d, N runs are performed, similar to the heuristic developed by Vink et al. (2019). Instead of a sorting algorithm, the probabilities from the random forest classifier are used to sort the candidate aircraft.

For the case study, the flight schedule was not updated with the recovery solution, i.e. for all runs the original flight schedule was used. The purpose of this research and of this case study is to determine the performance of the Full DSS against the Heuristic and the Optimum. Not updating the schedule ensures that the cases work with the same original schedule and hence gives a clear one to one comparison of the cases. If the schedule would be updated after each run, the schedules would diverge and the same run would start with a different schedule for the cases since the cases sometimes make different recovery decisions.

8.2.2. Case Study Results

Table 8.5 summarizes the results of the case study. The table shows averaged key statistics for the 367 runs. The full case study results are published online and can be found in Hassan (2019).

Table 8.5: Comparison of key statistics for the 367 runs of the case study

	Time [s] (Avg.)	**Δ Time** (Avg.)	**Aircraft** (Avg.)	**Cost** (Avg.)	**Δ Cost** (Avg.)	**OTP** (Avg.)
Optimizer	83	0%	140	$20.205	0%	99,33%
Trivial	32	-61%	2	$128.454	536%	99,30%
Heuristic	35	-58%	17	$24.096	19%	99,33%
Full DSS	48	-42%	71	$21.409	6%	99,34%
Full DSS-S	36	-57%	17	$23.907	18%	99,33%

	DPC (Sum)	**APD** (Avg.)	**XP** (Sum)	**XF** (Sum)	**MCP** (Sum)	**CV** (Sum)
Optimizer	92.542	29	459	4	4.285	0
Trivial	88.098	29	10.305	84	3.952	26
Heuristic	91.975	29	1506	12	4.396	0
Full DSS	91.746	28	897	8	4.285	0
Full DSS-S	91.686	28	1668	14	4.180	0

From Table 8.5 it can be concluded that the Heuristic, on average, presents a solution faster than the Full DSS. However, the Full DSS outperforms the Heuristic in terms of average cost and Key Performance Indicators (KPIs). More specifically:

- The Full DSS increases the average solution cost by 6% compared to an average cost increase of 19% for the Heuristic. Analysis of the full results show that the Full DSS is able to find the optimal solution in all but 4 of the 367 runs, i.e. the optimal solution is found in 98.9% of the runs. The Heuristic finds the optimal solution in all but 13 of the runs, or 96.5%.
- On average the Full DSS reduces the runtime by 42% compared to 58% for the Heuristic. Moreover, the maximum runtime found in the case study is 180 seconds for the Full DSS and 124 seconds for the Heuristic.
- In terms of delay KPIs, the largest difference is found when comparing the number of cancelled flights and cancelled passengers. The Heuristic tripled the number of cancelled flights and consequent passengers while the Full DSS doubled the cancelled flights, compared to the Optimum.
- The Full DSS-S case performs roughly equal to the Heuristic but worse than the Full DSS in terms of cost and delay KPIs. The Full DSS-S finds the optimal in all but 9 of the runs, this equals 97.5% of all runs.

Out of the 367 runs, 48 (13%) are non-trivial, i.e. other aircraft besides the disrupted aircraft are required to find the optimal solution. This shows that for the majority of disruptions, no other aircraft are needed to find the optimal solution. Table 8.6 shows a comparison of the key statistics for the non-trivial runs in the case study.

Table 8.6: Comparison of key statistics for the 48 non-trivial runs of the case study

	Time [s] (Avg.)	**Δ Time** (Avg.)	**Aircraft** (Avg.)	**Cost** (Avg.)	**Δ Cost** (Avg.)	**OTP** (Avg.)
Optimizer	124	0%	164	$59.624	0%	99,45%
Trivial	44	-65%	2	$887.282	1388%	99,19%
Heuristic	56	-55%	19	$89.356	50%	99,44%
Full DSS	68	-45%	83	$68.834	15%	99,46%
Full DSS-S	51	-59%	19	$87.927	47%	99,45%

	DPC (Sum)	**APD** (Avg.)	**XP** (Sum)	**XF** (Sum)	**MCP** (Sum)	**CV** (Sum)
Optimizer	16.667	41	0	0	1.362	0
Trivial	12.223	41	9.846	80	1.029	26
Heuristic	16.058	42	1.047	8	1.468	0
Full DSS	15.871	38	438	4	1.362	0
Full DSS-S	15.623	35	1.209	10	1.257	0

From Table 8.6 the following additional points are concluded for the non-trivial runs:

- The Full DSS increases the average solution cost by 15% compared to an average cost increase of 50% for the Heuristic. Analysis of the full results show that the Full DSS is able to find the optimal solution in 91.7% of all non-trivial runs. The Heuristic finds the optimal solution in 72.9% of the non-trivial runs.
- On average, the Full DSS reduces the runtime by 45% compared to 55% for the Heuristic. When comparing the non-trivial runs, the Heuristic finds the optimal solution in 56 seconds on average, while the Full DSS finds the optimal solution in 68 seconds on average.
- When comparing the delay KPIs it can be seen that the 4 flight cancellations in Table 8.5 come from trivial runs. The optimal recovery solution for the non-trivial runs require

no flight cancellations. The Heuristic cancels 8 flights while the Full DSS cancels 4 flights. Furthermore the Heuristic increases the number of passengers that miss their connecting flight.

- The Full DSS-S has slightly lower average cost for the non-trivial runs compared to the Heuristic. Even so, 2 additional flights (and corresponding passengers) are cancelled. The Full DSS-S finds the optimal solution in 81.3% of the non-trivial runs. However, for all but one case where the Heuristic did not find the optimal solution, the Full DSS-S found the same or a better solution than the Heuristic. The one case where the solution was not better, the cost was 0.12% higher. Furthermore, in seven out of the nine runs where the Full DSS-S did not find the optimal solution, the Heuristic also did not find the optimal. In the other two runs, both the Heuristic and the Full DSS were able to find the optimal.

Both the Heuristic and Full DSS-S usually find the best solution before all N runs are complete, therefore the time column in Table 8.5 shows the time it took to find the best solution and not the full runtime of the iteration. For the Optimum, Trivial and Full DSS, the time to the best solution is equal to the time of the full iteration. Figure 8.12 shows the boxplots of the time to the best solution for all cases and all iterations in the case study and Figure 8.13 shows the runtimes until the best solution for the non-trivial runs. For all runs the average time it takes to setup the iteration, e.g. load data, create the CPM, takes 32 seconds, which is the majority of the time for all cases except the Optimum. For the 48 non-trivial runs, the average iteration setup time equals 44 seconds, which is still the majority of the runtime for most cases. Chapter 9 will discuss how this setup runtime could potentially be reduced.

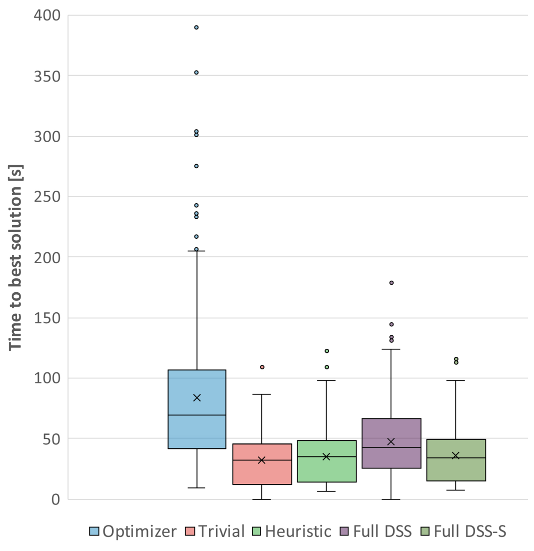


Figure 8.12: Boxplot of time to best solution for all runs

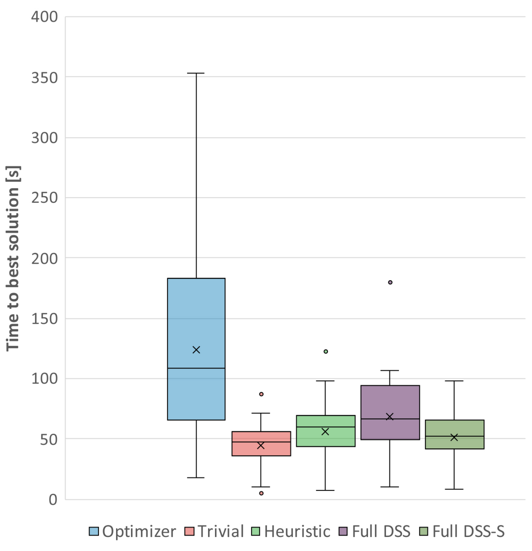


Figure 8.13: Boxplot of time to best solution for the 48 non-trivial runs

From the case study results it is clear that both the Heuristic and the Full DSS are able to find the optimal solution in the majority of runs. In all 200 iterations, the Full DSS found the same or a better solution than the Heuristic. The next section will highlight some of the runs where the Heuristic and/or the Full DSS were not able to find the optimal solution.

8.2.3. Highlighted Runs

This section will highlight some of the runs where the Full DSS or Heuristic was not able to find the optimal solution and discuss the cause. The highlighted runs cover all reasons why the Full DSS did not find the optimal solution.

Iteration 386 run McDonnell-Douglas

The original and recovered schedule for the aircraft involved in the recovery solution for this run are summarized in Table 8.7. All aircraft are of the same type. In this run two disruptions occurred within the McDonnell-Douglas family for iteration 386. The solution to the first disruption is trivial and will not be discussed, the other disruption was the following:

- Flight DL2299a: Delayed for 20 minutes, flown by aircraft N987AT

Table 8.7: Combined original and recovered flight schedule for aircraft involved in the recovery solution of iteration 386 run MD. Changes in schedule are shaded.

Flight No.	Original Tail No.	Recovery Tail No.	Orig.	Dest.	STD	STA	TAT	Original Delay	Recovery Delay
DL2225	N959AT	N959AT	PVD	ATL	17:40	20:30	30	0	0
DL1142b	N919AT	N919AT	ATL	HOU	19:00	21:30	30	0	0
DL1771a	N987AT	N987AT	MEM	ATL	20:00	21:10	30	0	0
DL2299a	N987AT	N959AT	ATL	HOU	21:40	00:00	30	0	20
DL1142a	N919AT	N959AT	HOU	ATL	22:10	00:10	30	0	160
DL2195b	N920AT	N920AT	MLB	ATL	23:00	00:40	30	0	0
DL2299b	N987AT	N919AT	HOU	ATL	00:30	02:20	30	0	0
DL1464	N919AT	N920AT	ATL	ORF	00:40	02:10	30	0	30
DL879	N987AT	N987AT	ATL	DAL	03:00	05:20	30	0	0

Aircraft N987AT is scheduled to fly flight DL2299a from ATL to HOU and flight DL2299b from HOU to ATL with minimal TAT in between. Flight DL2299b has 65 connecting economy and 16 connecting business passengers on board. Since the TAT between flights DL2299a and DL2299b is minimal, if flight DL2299a is delayed by 20 minutes, flight DL2299b would also be delayed by 20 minutes. Delaying flight DL2299b has a high cost, since 15 connecting passengers would miss their connecting flight. The tail swaps in the optimal recovery solution ensure that both flight DL2299a and DL2299b can be flown as scheduled. This in turn ensures that the connecting passengers on DL2299b can make their transfer. Through the tail swaps, the recovery solution transfers the delay to flights that have a lower number of connecting passengers which results in lower cost.

The Heuristic did not include aircraft N959AT and N919AT in the same selection and was only able to identify the trivial solution. The trivial solution is to delay flights DL2299a and DL2299b for 20 minutes. The Full DSS correctly included all necessary aircraft in the sub-network and was able to find the optimal solution. Table 8.8 shows the solution statistics and KPIs for this run.

Table 8.8: Statistics and KPIs for the recovery solutions found for iteration 386 run MD

	Time [s]	Aircraft	Cost	Δ Cost	OTP	DPC	APD	XP	XF	MCP	CV
Optimum	143	208	$110.801	0%	99,2%	480	58	0	0	70	0
Trivial	38	2	$125.441	13%	99,3%	530	22	0	0	105	0
Heuristic	40	22	$125.441	13%	99,3%	530	22	0	0	105	0
Full DSS	67	106	$110.801	0%	99,2%	480	58	0	0	70	0
Full DSS-S	50	22	$110.801	0%	99,2%	480	58	0	0	70	0

Iteration 482 run McDonnell-Douglas

The original and recovered schedule for the aircraft involved in the recovery solution for this run are summarized in Table 8.9. All aircraft are of type MD-88. In this run two disruptions occurred within the McDonnell-Douglas family for iteration 482. The solution to the first disruption is trivial and will not be discussed, the other disruption was the following:

- Flight DL845a: Delayed for 80 minutes, flown by aircraft N903DE

Table 8.9: Combined original and recovered flight schedule for aircraft involved in the recovery solution of iteration 482 run MD. Changes in schedule are shaded.

Flight No.	Original Tail No.	Recovery Tail No.	Orig.	Dest.	STD	STA	TAT	Original Delay	Recovery Delay
DL2146	N925DL	N925DL	JFK	PBI	13:20	16:20	45	0	0
DL616	N992DL	N992DL	MEM	ATL	13:20	14:40	45	0	0
DL845a	N903DE	N903DE	ATL	PBI	14:50	16:40	45	0	80
DL2644	N925DL	N903DE	PBI	JFK	17:10	19:50	45	0	100
DL1671	N911DL	N911DL	MIA	JFK	17:30	20:30	45	0	0
DL845b	N903DE	N925DL	PBI	ATL	17:30	19:20	45	0	0
DL478	N903DE	N992DL	ATL	JFK	20:10	22:30	45	0	0
DL2370	N925DL	N911DL	JFK	MCO	21:30	00:40	45	0	0
DL2510	N903DE	N903DE	JFK	ATL	23:20	02:00	45	0	0
DL2302	N911DL	N992DL	JFK	MIA	23:20	02:40	45	0	0
DL913	N925DL	N911DL	MCO	MEM	01:30	03:40	45	0	0

All flights on the flight string of aircraft N903DE, which includes flight DL845a as the first flight, are flown with minimal TAT in between, hence delaying flight DL845a means delaying all flights on the flight string by 80 minutes. Since that last flight on the flight string, DL2510, is crossing the time window, the trivial solution of delaying all flights on the flight string is not feasible. All flight swaps are necessary to ensure the schedule is feasible and the constraint of Equation 5.9 is satisfied, which constrains the number of aircraft per type required at the end of the time window at each airport.

The Full DSS did not include aircraft N911DL in the sub-network, since the airports on the flight string of N911DL do not equal the origin or destination airport of the disrupted flight. The flight string of the disrupted aircraft, N903DE, and the flight string of N911DL do cross at JFK later in the time window. Nonetheless, the Full DSS was able to find a different solution in which flights DL845a and DL845b are flown by N903DE and delayed by 80 minutes. Flights DL478 and DL2510 are tail swapped to another aircraft to ensure the constraint of Equation 5.9 is satisfied. Table 8.10 shows the solution statistics and KPIs for this run.

Table 8.10: Statistics and KPIs for the recovery solutions found for iteration 482 run MD

	Time [s]	Aircraft	Cost	Δ Cost	OTP	DPC	APD	XP	XF	MCP	CV
Optimum	185	220	$124.772	0%	99,7%	366	78	0	0	65	0
Trivial	49	2	$1.817.198	1356%	98,5%	117	50	551	4	62	0
Heuristic	70	22	$143.530	15.0%	98,7%	383	70	0	0	134	0
Full DSS	79	112	$143.530	15.0%	98,7%	383	70	0	0	134	0
Full DSS-S	40	22	$143.705	15.2%	98,7%	383	68	0	0	134	0

Iteration 492 run Airbus A320

The original and recovered schedule for the aircraft involved in the recovery solution for this run are summarized in Table 8.11. In this run two disruptions occurred within the Airbus A320 family for iteration 492. The solution to the first disruption is trivial and will not be discussed, the other disruption was the following:

- Flight DL2175: Delayed for 60 minutes, flown by aircraft N358NW

Table 8.11: Combined original and recovered flight schedule for aircraft involved in the recovery solution of iteration 492 run A320. Changes in schedule are shaded.

Flight No.	Original Tail No.	Recovery Tail No.	Orig.	Dest.	STD	STA	TAT	Original Delay	Recovery Delay
DL2175	N358NW	N358NW	LGA	RSW	16:30	19:50	40	0	60
DL2060	N340NB	N340NB	LGA	MCO	17:00	20:10	40	0	0
DL2181	N360NW	N360NW	LGA	MCO	19:00	22:10	40	0	0
DL1334	N339NW	N360NW	MCO	ATL	20:00	21:30	40	0	170
DL2604	N358NW	N358NW	RSW	LGA	20:40	23:30	40	0	50
DL2056	N340NB	N339NW	MCO	LGA	21:00	23:30	40	0	0
DL1776	N360NW	N340NB	MCO	LGA	23:00	01:30	40	0	0
DL1485	N358NW	N339NW	LGA	MCO	00:20	03:20	40	0	0
DL1147	N340NB	N358NW	LGA	ATL	01:00	03:40	40	0	0
DL906	N370NB	N360NW	ATL	LGA	01:40	03:50	40	0	0

The flight string of the disrupted aircraft, N358NW, has 20 minutes of slack time between the delayed flight and the end of the time window. Since the delay of flight DL2175 is 60 minutes, the trivial solution of delaying all flights on the flight string is not feasible. The last flight of N358NW, flight DL1485, is crossing the time window and needs to be flown as scheduled. To ensure this, the flight is tail swapped to N339NW at LGA. For N339NW to arrive at LGA, it itself needs to be tail swapped with aircraft N340NB at MCO. All aircraft are of type A320-212, except N340NB and N370NB, which are of type A319-114. The tail swaps on flights DL1776 and DL906 are necessary to satisfy the constraint of Equation 5.9.

The Full DSS did not include aircraft N339NW in the sub-network, since the airports on the flight string of N339NW do not equal the origin or destination airport of the disrupted flight. The flight string of the disrupted aircraft, N358NW, and the flight string of N339NW do cross at MCO later in the time window. The Full DSS only found the trivial solution, which is to cancel flights DL2175 and DL2604. Table 8.12 shows the solution statistics and KPIs for this run.

Table 8.12: Statistics and KPIs for the recovery solutions found for iteration 492 run A320

	Time [s]	Aircraft	Cost	Δ Cost	OTP	DPC	APD	XP	XF	MCP	CV
Optimum	106	104	$130.654	0%	99,1%	503	73	0	0	55	0
Trivial	52	2	$358.614	174%	98,4%	128	10	239	2	0	0
Heuristic	50	22	$358.614	174%	98,4%	128	10	239	2	0	0
Full DSS	61	54	$358.614	174%	98,4%	128	10	239	2	0	0
Full DSS-S	52	22	$358.614	174%	98,4%	128	10	239	2	0	0

Iteration 493 run McDonnell-Douglas

The original schedule for the aircraft involved in the recovery solution for this run is presented in Table 8.13. In this run two disruptions occurred within the McDonnell-Douglas family for iteration 493. The solution to the first disruption is trivial and will not be discussed, the other disruption was the following:

- Flight DL1891a: Delayed for 60 minutes, flown by aircraft N995DN

Table 8.13: Combined original and recovered flight schedule for aircraft involved in the recovery solution of iteration 493 run MD. Changes in schedule are shaded.

Flight No.	Original Tail No.	Recovery Tail No.	Orig.	Dest.	STD	STA	TAT	Original Delay	Recovery Delay
DL1067	N943DN	N943DN	MSP	TPA	13:00	16:20	45	0	0
DL1080	N955DN	N955DN	STL	ATL	13:00	14:40	45	0	0
FERRY71	N952DL	N952DL	DTW	MSP	15:40	17:50	45	0	0
DL1891a	N955DN	N955DN	ATL	TPA	15:40	17:00	45	0	60
DL1148	N943DN	N955DN	TPA	MSP	17:10	20:40	45	0	100
DL1891b	N955DN	N943DN	TPA	ATL	17:50	19:20	45	0	0
FERRY906	N929DN	N929DN	MIA	ATL	18:20	20:20	45	0	0
DL1800	N955DN	N929DN	ATL	BOS	21:20	23:50	45	0	0
DL959a	N943DN	N952DL	MSP	MCI	21:30	22:50	45	0	0
DL959b	N943DN	N952DL	MCI	MSP	23:40	01:10	45	0	0

The flight string of the disrupted aircraft, N995DN, has 10 minutes of slack time between the delayed flight and the end of the time window. Since the delay of flight DL1891a is 60 minutes, the trivial solution of delaying all flights on the flight string is not feasible. The last flight of N955DN, flight DL1800, is crossing the time window and needs to be flown as scheduled. To ensure this, the flight is tail swapped to N929DN at ATL. Flight DL1891b has 11 business and 74 economy connecting passengers. If the flights was to be flown by N955DN with 60 minutes of delay, 5 business and 16 economy passengers would miss their transfer. Therefore, flight DL1891b is tail swapped to N943DN at TPA. The original flight for N943DN, DL1148 is flown by N955DN with a delay of 100 minutes. This swap ensures that the connecting passengers can make their transfer. Flights DL959a and DL959b are tail swapped to N952DL, which arrived at MSP at 17:50 and does not have any more flights planned after that.

The Full DSS did not include aircraft N952DL in the sub-network, since the flight string of N952DL does not cross the flight sting of the disrupted aircraft. Nonetheless, the Full DSS was able to find a different solution in which 6 tail swaps with different aircraft ensure the connecting passengers can make their transfer. Table 8.14 shows the solution statistics and KPIs for this run.

Table 8.14: Statistics and KPIs for the recovery solutions found for iteration 493 run MD

	Time [s]	Aircraft	Cost	Δ Cost	OTP	DPC	APD	XP	XF	MCP	CV
Optimum	186	219	$71.248	0%	99,8%	520	48	0	0	18	0
Trivial	53	2	$474.810	566%	99,6%	285	60	259	2	65	0
Heuristic	69	22	$85.651	20%	99,8%	544	37	0	0	76	0
Full DSS	98	110	$82.319	16%	99,8%	520	48	0	0	18	0
Full DSS-S	63	22	$85.651	20%	99,8%	544	33	0	0	76	0

8.2.4. Impact of Delaying Outbound Connecting Flights

Section 4.1 discusses the extension to the Connecting Passenger Matrix (CPM), which allows changing the STD of the outbound connecting flights if the inbound flight is disrupted. This option was not enabled when performing the case study, since delaying the outbound connecting flights in combination with solving disruptions per aircraft family resulted in trace-ability issues. This behaviour will be explained in this section. Furthermore, the impact of the extension will be evaluated by comparing the averaged results over 100 iterations.

To explain the trace-ability issues, Table 8.15 shows two example connections. Aircraft N309US belongs to the Boeing 737 family, N371DA belongs to the Airbus 320 family and N900DE belongs to the McDonnell-Douglas family. When inbound flight DL2361 is delayed, the DSS will incentivize the delay of flights DL247 and DL1238 to ensure the connecting passengers can make their transfer on ATL. However, since aircraft N371DA and N900DE do not belong to the same family as the disrupted aircraft, N309US, these aircraft are not included in the run that solves the disruption for flight DL2361. Only in a later run, when an A320 or MD aircraft is disrupted, will these aircraft be included in the DSS and will the delay be included in the recovery solution. This behaviour was deemed undesirable for the case study where runs needed to be comparable and the cost need to be traceable and verify-able.

Table 8.15: Example connecting flights schedule

Flight 1	Tail No. 1	Flight 2	Tail No. 2	Orig.	Conn.	Dest.	Economy Pax.	Business Pax.
DL2361	N309US	DL427	N371DA	JAX	ATL	ORD	5	2
DL2361	N309US	DL1238	N900DE	JAX	ATL	HOU	8	3

The impact of the extension to the CPM will be evaluated by comparing the results of the DSS with and without the option enabled on iterations 0-100 of the disruption dataset, which covers about 32 hours of operations in January. Furthermore, the same parameters as presented in Table 8.4 will be used. Table 8.16 summarizes the results of the study. The averaged results should give an indication of the cost reduction that the extension effectuates. ”No Delay” refers to the results without delaying outbound connecting flights, while ”Delay” refers to the results where the outbound flights could be delayed to recovery passengers. In both cases, the random forest classifier and sub-network selection algorithm were not enabled, hence all aircraft were included. Besides the Missed Connecting Passengers (MCP) KPI, the table shows the Recovered Connecting Passengers (RCP). The RCP shows the number of connecting passengers that can make their transfer due to an outbound connecting flight being delayed.

Table 8.16: Impact of delaying outbound connecting flights on the KPIs

	Time [s]	**Cost**	**Δ Cost**	**OTP**	**DPC**	**APD**	**XP**	**XF**	**MCP**	**RCP**	**CV**
No Delay	79	$1.967.344	0%	99,45%	25.170	37	0	0	1.192	0	0
Delay	84	$1.126.036	-43%	99,32%	37.795	35	0	0	1.307	213	0

From Table 8.16 the following conclusions can be drawn:

- The overall disruption cost decreases by over 40% when allowing outbound connecting flights to be delayed in order to ensure passengers can make their connection.
- The number of delayed passengers (DPC) increases by 50%, since more flights are delayed and therefore more passengers are delayed.
- The average passenger delay (APD) decreases slightly, this indicates that the outbound connecting flights are usually delayed by less than 37 minutes.
- The passengers missing their connection (MCP) increases, which shows that the outbound connecting flights that are delayed have some connecting passengers as well. Some of these passengers might miss their connection. However, overall less passengers miss their connection since 213 passengers are recovered by delaying outbound flights.

8.3. Sensitivity Analyses

The core of this research is the addition of the random forest classifier and the sub-network selection algorithm to the DSS. The performance of the Final DSS is highly dependent on the predictive performance of the classifier and the aircraft selections. This section will investigate the sensitivity of the decision support system (DSS) to changes in the random forest classifier and the sub-network selection algorithm. Furthermore, the classifiers ability to generalize to unseen data will be assessed, i.e. how does the classifier perform on data it was not trained on. Vink (2016) presented a sensitivity analysis that discusses the relation between the size of the LP problem and the size of the inputs. Furthermore, the impact of the schedule penalty is discussed.

8.3.1. Sub-network Selection Algorithm

One of the novelties of this research is the sub-network selection algorithm that uses the class probabilities generated by the random forest classifier to create a sub-network of aircraft. For the case study in the previous section the sub-network strategy was set on 'Top 50%', so that, for every iteration the top 50% aircraft with the highest predicted probabilities of helping were selected to be included in the optimization. As discussed in Section 4.3 there are two strategies that can be employed to select the aircraft. This section will discuss the impact of the strategy on the recovery solution.

To determine the sensitivity of the solution to the selection strategy, several different strategies and parameter settings have been used to solve the 200 iterations of the case study. Table 8.17 summarizes the sensitivity analysis. The table shows the impact of different strategies and different parameter values on the runtime, cost and KPIs. One of the hypothesis of this research, discussed in Chapter 3, is that machine learning predictions will have a positive impact on the computation time while not significantly deteriorating the solution quality. Therefore, it is useful to determine the optimal strategy for the sub-network selection. It should be noted that the results of Table 8.17 are specific to the random forest classifier presented in Section 6.3. A different random forest classifier may require a different sub-network selection strategy and parameter value.

Table 8.17: Sensitivity of Recovery solution to changes in the sub-network selection strategy

	Time [s]	Cost	Aircraft	OTP	DPC	APD	XP	XF	MCP	CV
	(Avg.)	(Avg.)	(Avg.)	(Avg.)	(Sum)	(Avg.)	(Sum)	(Sum)	(Sum)	(Sum)
Full Fleet	83	$20.205	140	99,33%	92.542	28	459	4	4.285	0
Sub network selection strategy: Top X% of aircraft										
Top 60 %	83	$20.205	140	99,00%	92.573	28	459	4	4.286	0
Top 50%	48	$21.439	71	99,34%	91.746	28	897	8	4.285	0
Top 40%	43	$22.316	57	99,34%	91.540	28	1.189	10	4.285	0
Top 30%	39	$23.383	43	99,34%	90.425	28	1.477	12	4.187	0
Top 20%	36	$23.775	29	99,34%	90.944	28	1.477	12	4.344	0
Top 10%	33	$25.445	15	99,34%	90.610	28	2.008	16	4.241	0
Sub network selection strategy: All aircraft above threshold X										
Thresh 0.10	60	$20.255	91	99,33%	92.370	28	459	4	4.286	0
Thresh 0.20	42	$22.621	43	99,33%	91.741	28	1.201	10	4.193	0
Thresh 0.30	36	$29.043	22	99,33%	90.624	27	2.368	18	4.115	1
Thresh 0.40	34	$37.075	12	99,34%	89.492	28	2.982	23	4.181	3

The analysis shows that the most selective strategies, i.e. Top 10% and Threshold > 0.40, result in a deteriorated solution while not decreasing the computation time proportionately. In fact, in some of the iterations these strategies resulted in constraint violations (CV) and an increased number of cancelled flights (XF) and cancelled passengers (XF).

Figures 8.14 and 8.15 show the computation time and cost vs the number of aircraft in the sub-network selection, respectively. As expected, the computation time decreases and the cost increases with a lower number of aircraft. However, Figure 8.15 shows that the cost increases rapidly when there are less than 40 aircraft (on average) in the sub-network. It can be concluded that the optimal number of aircraft lays between 40-80, depending on the preferences of the airline. This corresponds to the strategies where either the top 40-60% of aircraft are selected or where all aircraft with a probability above threshold 0.10-0.20 are selected.

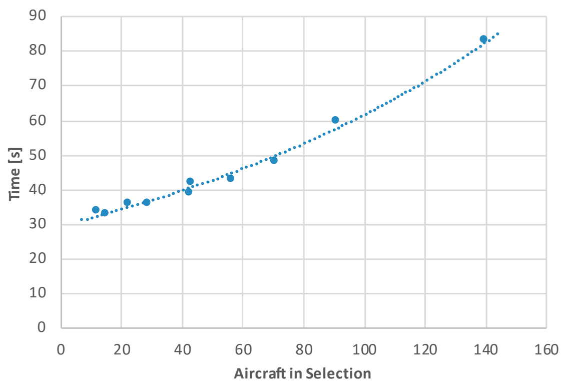


Figure 8.14: Computation time vs the number of aircraft in the sub-network selection

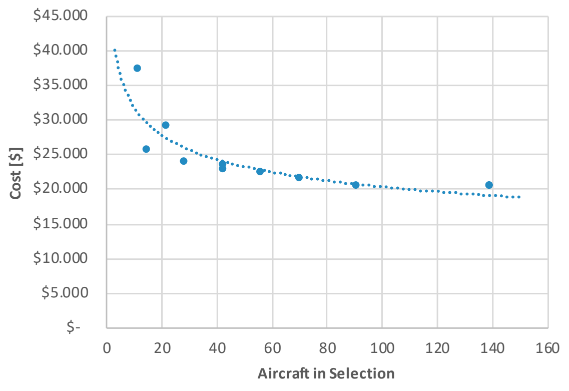


Figure 8.15: Cost vs the number of aircraft in the sub-network selection

Table 8.18 presents statistics about the computation time and number of selected aircraft in the sub-network for different strategies and parameter values. From the table it becomes clear that the Top X% strategies perform more consistent, or, in other words, with less variance. Thus, for operational use in an AOCC the Top X% strategies may be preferred.

Table 8.18: Performance statistics of sub-network selection strategies

	Computation Time			**Aircraft in Sub-network**		
	Mean	**Var**	**St.Dev**	**Mean**	**Var**	**St.Dev**
Sub-network selection strategy: Top X% of aircraft						
X = 60%	53,9	1124,3	33,5	84,2	1128,5	33,6
X = 50%	48,0	827,8	28,8	70,5	787,9	28,1
X = 40%	42,8	649,7	25,5	56,5	506,1	22,5
X = 30%	39,1	579,1	24,1	42,6	288,7	17,0
Sub-network selection strategy: All aircraft above threshold X						
X = 0.20	42,4	987,4	31,4	42,8	1788,6	42,3
X = 0.10	60,4	1995,4	44,7	91,0	3352,0	57,9

8.3.2. Random Forest Classifier Generalization

The airline industry is subject to seasonality, which impacts both the number of daily flights and the number of disruptions. In order for the random forest classifier and sub-network selection algorithm to be useful, they need to be able to accurately predict which aircraft will help solve a disruption throughout the year. The case study in the beginning of this chapter discussed the performance of the Full DSS with the classifier. This classifier was trained on training instances from January and assessed on disruptions in January. This section will determine the ability of the classifier to generalize by assessing the performance of the January classifier on disruptions in March. Figure 8.16 shows the number of disruptions in Q1 2015, where the highlighted bars indicate the days in January and March that were used in the case study. The Full DSS-Jan (January classifier) will be compared to the Optimum, Heuristic and the Full DSS with a classifier that was trained on March instances.

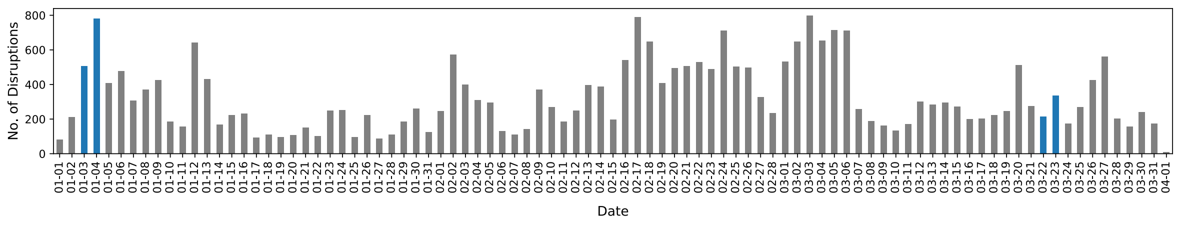


Figure 8.16: Number of disruptions per day in Q1 2015. Highlighted bars indicate days used for case study in January and March.

The study comprised of 100 iterations near the end of March, which consisted of 147 runs and 182 disruptions. Table 8.19 summarizes the results of the March disruptions.

Table 8.19: Comparison of averages key statistics for 100 iterations in March.

	Time [s] (Avg.)	Δ Time (Avg.)	Aircraft (Avg.)	Cost (Avg.)	Δ Cost (Avg.)	OTP (Avg.)
Optimum	123	0%	105	$28.608	0%	99,58%
Trivial	57	-54%	1	$197.873	592%	99,53%
Heuristic	61	-50%	8	$47.169	65%	99,58%
Full DSS-Jan	78	-37%	76	$39.183	37%	99,58%
Full DSS-Mar	75	-49%	76	$33.766	18%	99,58%

	DPC (Sum)	APD (Avg.)	XP (Sum)	XF (Sum)	MCP (Sum)	CV (Sum)
Optimum	35.419	38	289	2	1.910	0
Trivial	30.846	37	7.452	60	1.499	14
Heuristic	33.501	35	2.714	22	1.588	0
Full DSS-Jan	34.788	38	1.414	10	1.843	0
Full DSS-Mar	35.368	38	924	7	1.874	0

From Table 8.19 it can be concluded that the March classifier outperforms the January classifier on every KPI. Specifically on the cost, the Full DSS-Jan increases the cost by 37% while the Full DSS-Mar increases the cost by 18%. In the 147 runs, 35 (24%) are non-trivial. When comparing the non-trivial runs, the Full DSS-Mar finds the optimal solution in all but 2 runs, while the Full DSS-Jan finds the optimal solution in all but 5 runs. While it was to be expected that the Full DSS-Mar outperforms the Full DSS-Jan, this difference shows that the January classifier only has a moderate ability to generalize. The next chapter will discuss these results and recommendations regarding generalization performance.

8.3.3. Random Forest Classifier Hyperparameter Sensitivity

Chapter 6 discusses the framework that was used to obtain the random forest classifier. One of the crucial elements of the framework is the hyperparameter tuning. Section 6.3 presented the random forest classifier that was used for the case study, including the final hyperparameters obtained after training. This section will elaborate on the influence of the hyperparameters on the prediction performance, measured using the Average-Specificity (AS) metric. The same hyperparameters that were used for training will be used for the sensitivity analysis. For each hyperparameter the value was changed over a range while keeping the value of the other parameters equal to the optimal value found during training.

Figure 8.17 shows the results of the hyperparameter sensitivity analyses. The figures indicate that the performance of the random forest classifier is not very dependent on the hyperparameter values, as long as the value is larger than 5-10. The min_samples_leaf parameter shows the a slight trend upwards with an increasing parameter value for the test dataset and a negative trend for the train dataset. This makes sense, since a larger min_samples_leaf results in less overfitting. These results show that a parameter value of 150 instead of 67 would likely improve the performance of the classifier slightly.

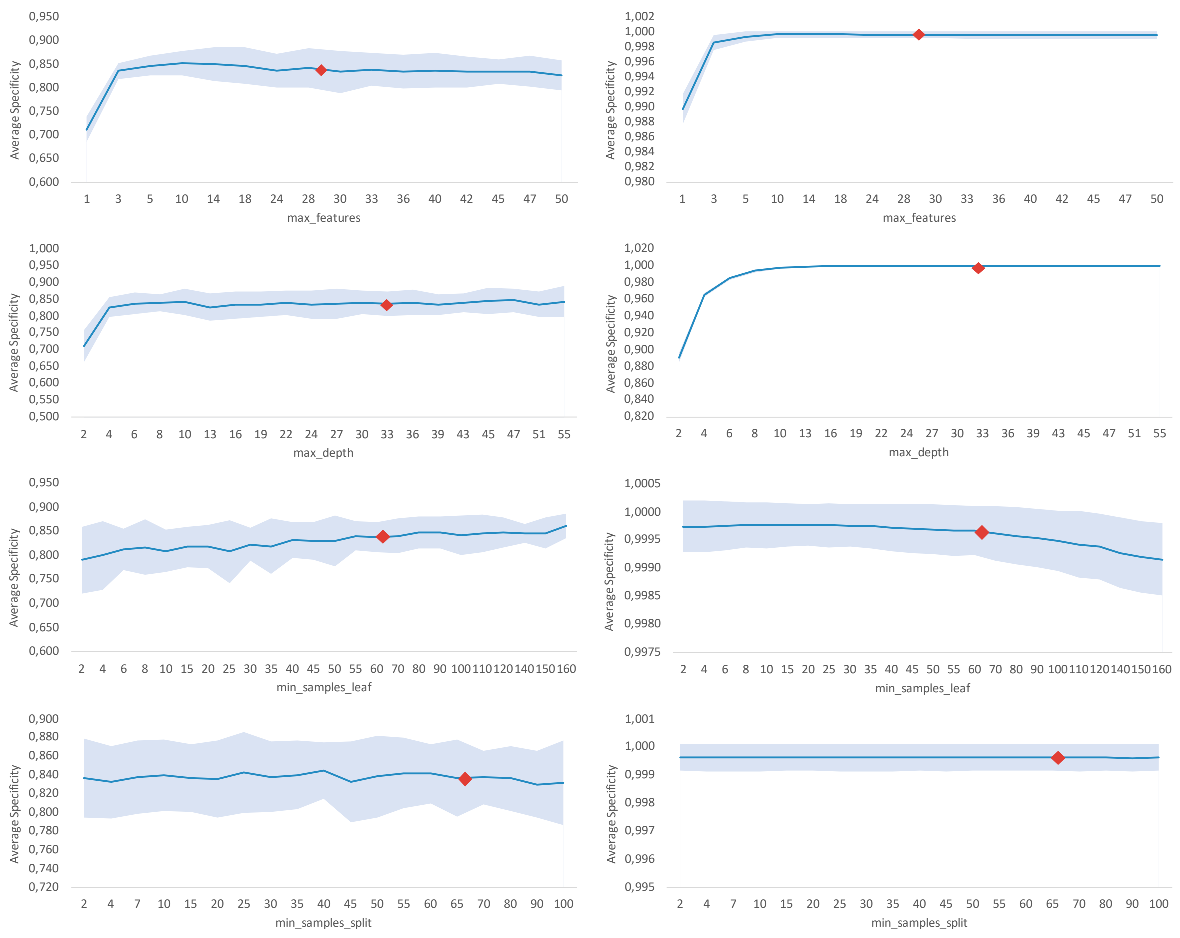


Figure 8.17: Sensitivity of random forest classifier to changes in hyperparameters in terms of Average Specificity. The left column shows the results on the test dataset, the right column shows the results on the train dataset. The shaded area indicates one standard deviation of the 5-fold cross validated results. The diamond indicates the optimal hyperparameter value found after training.

8.4. Concluding Remarks on Case Study

This chapter presented a case study where the performance of the Full DSS was compared with the optimal solution and the heuristic developed by Vink et al. (2019). The case study considered 200 iterations in January 2015 which covered 565 disruptions. Overall it was found that the Full DSS is able to find the same or a better solution than the heuristic for all disruptions. In 87% of all runs, the optimal solution equaled the trivial solution, where only the disrupted aircraft was involved in the recovery solution. The Full DSS and heuristic found the optimal solution in 91.7% and 72.9% of all non-trivial runs, respectively. Overall, the Full DSS increased the cost by 6.0% compared to 19.3% for the heuristic. Compared to the optimal solution, the Full DSS and the heuristic both decreased the time required to present the best solution. However, on average the Full DSS presented the best solution 13 seconds later than the heuristic.

Another case combined the classifier and sub-network selection algorithm of the Full DSS with an aircraft selection heuristic, similar to the one presented by Vink et al. (2019). Compared to the Full DSS, the benefit of this setup is the decrease in the required time to present the best solution. Nonetheless, the solution quality is lower since the pitfalls of the random forest classifier and the selection heuristic are combined. The classifier does not always identify the correct candidate aircraft required for the optimal recovery solution. Even if the correct candidate aircraft are included in the sub-network, the selection heuristic does not always include the correct combination of aircraft in the runs.

By further analyzing the 4 non-trivial cases in the case study where the Full DSS did not find the optimal solution, it was found that the random forest classifier has some difficulty identifying candidate aircraft required for tail swaps when the candidate is not on the same origin or destination airport as the disrupted flight. Moreover, the classifier does not correctly identify candidate aircraft if their flight string and the flight string of the disrupted aircraft do not cross.

The impact of the extension to the Connecting Passenger Matrix (CPM) was evaluated. For the 100 iterations that were tested the overall disruption cost decreased by over 40% when allowing outbound connecting flights to be delayed in order to ensure passengers can make their connection. However, since more flights were delayed, the number of delayed passengers increased by 50%.

A sensitivity analysis determined the relation between the selection strategy for the sub-network selection algorithm and the disruption cost and computation time. The relations show that, for the current classifier, the solution quality rapidly deteriorated when less than 40 aircraft (on average) are selected. This corresponds to the sub-network selection strategies where the Top 40-60% of aircraft are selected.

Another sensitivity analysis assessed the generalization performance of the random forest classifier by using the classifier to solve several disruptions in March. The analysis showed that the January trained classifier is moderately able to generalize to the March disruptions. A classifier trained on March data increased the cost by 18% on average, while the classifier trained on January data increased the cost by 37% on average.

Lastly, the hyperparameter sensitivity of the random forest classifier was determined by iteratively varying hyperparameters and determining the performance in terms of Average-Specificity. The results show that the performance of the classifier is not very dependent on the values of the tested hyperparameters as long as the value is larger than 10.

The next chapter of this report, Chapter 9, will present the conclusions of this research and recommendations for further research and improvements.

9 Conclusion and Recommendations

The previous chapter presented the results of the case study and the sensitivity analyses of the Full DSS, including classifier and sub-network selection algorithm. This chapter presents the conclusions of the research and recommendations for further improvement. The conclusions, based on the case study, are given in Section 9.1. Section 9.2 discusses the limitations of the system and recommendations for further improvement. Finally, in Section 9.3 the research question and hypotheses are evaluated.

9.1. Conclusions

Two categories of results are discussed. First, the results related to the inclusion of the random forest classifier and sub-network selection algorithm in the DSS are discussed. Second, the extensions and improvements to the modelling framework developed by Vink et al. (2019) are discussed.

9.1.1. Random Forest Classifier and Sub-network Selection

This research presented an approach for reducing the size of the aircraft recovery problem. Airlines require fast recovery solutions when disruptions happen. A random forest classifier was trained to predict the probability of undisrupted candidate aircraft being able to help solve a disruption. Features related to the characteristics of the candidate aircraft, disruption and flight schedule were developed and a random forest classifier was trained on 6 days worth of disruptions in January. This classifier was used to predict the probability that a candidate aircraft would have a positive impact on resolving a disruption. The classifier was combined with a sub-network selection algorithm to discard part of the aircraft in the fleet of the disrupted aircraft.

The developed decision support system (DSS) was compared with the optimal solution and the heuristic developed by Vink et al. (2019) in a case study on the network of Delta Airlines. The case study results show that the Full DSS is able to find the same or a better solution than the heuristic for all disruptions. Compared to the optimal solution, the Full DSS and the heuristic both decreased the time required to present the best solution. For the non-trivial runs, the Full DSS presented a solution in 68 seconds on average, which is 13 seconds later than the heuristic. Furthermore, for the non-trivial runs, the Full DSS increases the average solution cost by 15% compared to an average cost increase of 50% for the heuristic.

Further analyses revealed that random forest classifier has difficulty identifying candidate aircraft required for tail swaps when the candidate is not on the same origin or destination airport as the disrupted flight. Moreover, the classifier does not correctly identify candidate aircraft if their flight string and the flight string of the disrupted aircraft do not cross at all. This can be explained by the fact that there were no features in the feature space that identified these candidate aircraft and relation between their flight string and the flight string of the disrupted aircraft.

The generalization performance of the random forest classifier is a point of concern. A sensitivity analysis showed that the classifier, which was trained on January data, performed worse on disruptions in March. Furthermore, the learning curves of the classifier showed that the classifier overfitted to the training data. The next section will present recommendations to reduce the overfitting and increase the generalization performance of the classifier.

9.1.2. Improvements to Decision Support System (DSS)

This thesis continued the previous work by Vink et al. (2019) and Vos et al. (2015). The framework and implementation have been extended on several fronts:

- **Outbound connecting flight delay** - Vink et al. (2019) developed the Connecting Passenger Matrix (CPM) where the cost of missed connections where taken into account when considering delaying aircraft. However, delaying the outbound connecting flight was not included in the recovery options. This research extended the work by allowing the DSS to delay outbound connecting flights whenever possible to allow connecting passengers to make their transfer. The case study showed that, on average, the disruption cost decreased by 40% when allowing outbound connecting flights. However, the number of delayed passengers increased by 50%, since more flights are delayed. According to an industry expert, the addition made to the connecting passenger matrix, where outbound connecting flights can be delayed, is of added value to the DSS. Given the sometimes large number of connecting passengers on a flight, it would be hard for an AOCC controller to investigate opportunities for outbound flight delay. Automating this process and presenting suggestions to the controller is therefore appreciated.
- **Increased computational efficiency** - Working with a larger dataset revealed several implementation bottlenecks which increased the computational runtime. Several major bottlenecks have been been resolved which decreased the runtime by 60-70%. Recommendations regarding further improvements will be presented in Section 9.2.
- **Tail swap time limit** - Prior to this research the DSS allowed tail swaps close to the scheduled time of departure of an aircraft. In reality, swapping tails before departure is near impossible due to turn around operations and gate assignment schedules. This research included a tail swap constraint with a limit of 3 hours, although this can be set by the airline.

9.2. Recommendations

The previous section presented the conclusion to this research and extensions to the DSS. From the work, several key areas of further improvement have been identified. These recommendations for further improvements will be discussed in this section and can be divided in the following categories: *classifier*, *improvements to ARP*, *dataset*, *crew and passenger recovery* and *computational performance*.

9.2.1. Random Forest Classifier and Sub-network Selection

Although the case study showed that the random forest classifier was able to reduce the computation time of the ARP without significant solution deterioration, there are several recommendations that could lead to an improved classifier:

- **Increase generalization performance** - The learning curves of the final random forest classifier in Section 6.3 indicate that the classifier has overfitted to the training data. Furthermore, the sensitivity analysis regarding the generalization performance of the classifier showed that the classifier trained on January data did not generalize well to March data. There are several recommendations to decrease the overfitting and increase the generalization performance:

1. Decrease model complexity, by decreasing the max_depth, min_samples_split and/or min_samples_leaf hyperparameters
2. Gather more training data from diverse time periods, with more training data the classifier might be able to better predict specific situations. Gathering data from different time periods might increase the generalization performance.
3. Removing features, by removing irrelevant features the complexity of the model is reduced further. The next recommendation will further discuss feature engineering.

- **Further feature analyses and engineering** - The feature analysis, presented in Section 6.2.7, shows that the performance of the classifier was improved by adding new features and removing non-performing and/or highly correlated features. Further research into the features has the potential to improve the performance and robustness of the classifier. The case study showed that the DSS was unable to find the optimal solution in 4 of the 367 runs. In these runs a critical candidate aircraft was not correctly identified. In the current feature space contain features that identify candidate aircraft which are on the same origin or destination airport as the disrupted flight. Other features quantify their availability. In the 4 runs where the optimal solution was not found, the flight string of the critical candidate aircraft did not touch the origin or destination airport of the disrupted flight. Further research is needed to engineer features that will aid in the identification of those aircraft, for example by extending the current features to the entire flight string of the disrupted aircraft instead of the disrupted flight only.
- **Hyperparameter Optimization** - Bayesian optimization with 5-fold cross validation was used to find the optimal hyperparameters for the random forest classifier. By increasing the hyperparameter value ranges, increasing the number of iterations and using 10-fold CV, a better set of hyperparameters may be obtained.
- **Algorithm benchmark** - Section 6.2.2 discussed the machine learning algorithm selection and the choice for the random forest algorithm. However, the performance of an algorithm is highly dependent on the problem specifics. Although Olson et al. (2018) showed the strength of the random forest algorithm, a different algorithm may be better suited for the aircraft recovery problem. Well trained boosting algorithms, like XGBoost and LightGBM, are known to outperform random forests. Moreover, neural network are known to capture complex relationships and may be able to find relationships in the feature space that tree based algorithm are not able to find. It is recommended to benchmark different machine learning algorithms.

9.2.2. Aircraft Recovery

The current aircraft recovery formulation and implementation can be improved upon with several extensions:

- **Airport flow rate reductions and closures** - This research did not incorporate airport close times or reduced flow rates, for example a difference in daytime and nighttime operations. Therefore, the DSS is not limited by airport closures when considering flight delays. In reality, aircraft may not be allowed to land after certain hours, thereby limiting the delay options. Furthermore, airport flow rates (inbound and/or outbound) may be reduced due to weather conditions, this reduction is currently not incorporated in the research. Incorporating these limitations will require additional constraints which describe and limit the number of departures and arrivals per time period. These constraints will limit the number of flight slots for an airline, from which the DSS will be forced to choose the best recovery solution.
- **Connecting passengers rebookings** - As mentioned in Section 4.1, with the current implementation of the DSS and CPM, it could occur that more passengers are rebooked to a flight than the number of seats available. It is recommended to implement available seat updates after every iteration and to implement passenger spill constraints. This combination of features will allow for added realism and hence more realistic decisions by the DSS.

- **Connecting passengers and flight cancellations** - In the current implementation of the DSS, the maximum delay cost is assigned per passenger when a flight is cancelled. However, for connecting passengers an additional cost term is necessary to describe the itinerary recovery options in case of an cancellation. For example, cancelling the last flight of a day to a destination should be more expensive for connecting passengers then cancelling the first flight of a day.
- **Adding time window (TW) crossing flights flexibility** - As discussed in Section 4.1 the time window concept keeps the problem size manageable and limits the snowball effect of a disruption. Currently, all flights that cross the time window are constrained to fly as scheduled. This is done to ensure that the flight schedule is not disrupted after the end of the time window. However, this also constrains the recovery options available. There could be situations where the flight that crosses the time window can be delayed without problems, e.g. because the next scheduled flight for that aircraft departs much later. A rule of thumb could be created that evaluates the delay options and consequences of the flights that cross the time window and relaxes the constraint where possible.
- **Incorporating cost index (CI) decisions** - Marla et al. (2017) have shown that incorporating cost index decisions as a recovery technique can reduce the overall passenger delay and disruption cost for an airline. However, this will increase the size of the problem and hence increase computation time. Moreover, Arikan et al. (2017) have shown that there is no linear relation between fuel cost and cruise speed. Either the problem would have to be reformulated as an Mixed Integer Quadratic Programming (MIQP) problem or a discretization of the additional cost and flight time would need to be implemented. This could be achieved by creating additional flight and delay arc with lower flight times and higher cost due to additional fuel burn. These additional arcs (decision variables) will increase the problem size and hence computation time. The number of decision variables to add will equal the current number of flight and delay arcs times the discrete time steps of earlier arrival due to increased cruise speed, e.g. for 4 discrete earlier arrival options the number of flight and delay arcs would grow 400%. As shown by Vink et al. (2019), there exists a non-linear relationship between the number of decision variables and the runtime of the DSS, hence the runtime could increase by more than 400%. Clearly, a more intelligent way of adding these arcs is needed. Vink et al. (2019) suggested to only include these early arrival arcs for long haul flights, where the difference in arrival time is more significant than for short-haul flights.

9.2.3. Dataset

One of the contributions of this research is the generation of a realistic dataset for one of the largest hub-and-spoke carriers in the world. To the best of the author's knowledge, this is the largest and most realistic dataset available to the ATO section at the faculty of Aerospace Engineering. However, several opportunities for improvements still exist:

- **Maintenance Data** - Vink et al. (2019) extended the work by Vos et al. (2015) by adding (flexible) maintenance constraints to the DSS. Due to data unavailability, maintenance constraints were not included in this work. Since maintenance constraints limit the options available to the DSS and are a vital part of airline operations, it is recommended to obtain maintenance data and re-incorporate these constraints in further iterations of the DSS. It is critical to obtain real-world information to ensure validity of the results.
- **Disruptions dataset: disruption updates** - Since disruption development updates were not available, only the final disruption duration was evaluated in the DSS. In reality, AOCCs receive updates on weather conditions and disruptions on a regular basis and update these in their systems. For example, weather conditions may become more severe over the course of the day, which necessitates re-optimization of the disruption. If this data becomes available in future, it would be interesting to see how this would impact the performance of the DSS.

- **Disruptions dataset: aircraft unavailbilities** - Currently, aircraft unavailabilties are extracted with a simple rule-based technique, see Section 8.1. This rule is quite limited and lacks realism, since only aircraft unavailabilities of >120 minutes are extracted. In real world operations, aircraft could be unavailable for less then two hours. This rule should be improved in future work.
- **Schedule and Disruptions dataset** - The dataset used for the case study includes data on the first quarter of 2015. While this is already extensive, the data processing logic (see Appendix H can be applied to the other quarters of 2015 as well. This would create a dataset for the full year, allowing verification and validation in different months, seasons and peak periods. Given the seasonality of the airline industry, this could result in interesting findings and a more robust DSS.
- **Delay cost** - The current soft and hard cost are adapted from Vos et al. (2015), which based them on the research done by Cook et al. (2012). In reality, countries may have specific legislation, e.g. on passenger compensation in case of delays, which should be taken into account. Incorporating this (country specific) legislation will allow the DSS to make more realistic decisions.

9.2.4. Crew and Passenger Recovery

The aircraft recovery solution is only part of the solution required when a disruption occurs. Crew (flight and cabin) and passenger recovery were not considered in this research, but are vital for real life disruption management. However, including crew and passenger recovery will increase the computation time since the number of decision variables and constraints will increase. Since AOCCs prioritize fast solutions over perfect solutions, a sequential recovery strategy is advised over a global recovery strategy. Furthermore, techniques like column and row generation can improve the computational performance without significant deterioration of the solution quality. A (draft) crew recovery model formation is included in Appendix D and a partial implementation has been written for this research. However, it should be noted that this work is not verified nor validated.

9.2.5. Computational Performance

As mentioned in Section 9.1.2 the computational efficiency of the DSS has been improved. This section will discuss several options for further runtime reductions.

Implementation

Currently, the DSS is written in Python 2.7. While Python, like other interpreted languages, has many advantages, computational performance is not one of them for most applications. It is recommended to keep using Python for development due to the ease of development. However, when efficiency of execution considerations start to weight more heavily, e.g. when implementing the DSS in an operational setting, comparing Python, Java, C, Julia and other language implementations may prove to be beneficial. Furthermore, it could be investigated whether the current implementation of the DSS can be improved.

Parallel processing

Parallel computing allows to break down a problem into several similar subtasks that can be processed concurrently and whose results can be combined after completion. This allows for runtime reductions, since larger tasks can be split over multiple cores. There are several variants of parallel processing which could be implemented in the DSS:

- **Concurrent solving** - As can be seen in Section 8.2.2, the trivial solution is the optimal solution to the disruption in 87% of the runs. By starting two concurrent processing, one finding the trivial solution and one using the DSS, the time required to present a feasible solution can be reduced.
- **Concurrent computation** - The current implementation of the DSS consists of several steps where variables or data structures are computed per flight or aircraft in the selection. Runtime reductions can be achieved by performing these computations in

parallel instead of sequentially. Python has a mutex, the global interpreter lock (GIL), that protects access to objects. This makes parallel computing in Python a difficult task, which would also be an argument to rewrite the DSS in another language for real world implementation. Within Python, several packages, such as DASK and NUMBA can be leveraged to parallelize several computations.

9.3. Concluding Remarks on Research

This research investigated if machine learning predictions could be leveraged to reduce the computation time required to find a solution for the aircraft recovery problem. Based on the literature review and the research questions, two hypotheses were formulated for this thesis:

Hypothesis one
Based on real schedule, disruptions, network and fleet data of an airline, a machine learning model can be trained to select a sub-network that will result in a feasible recovery solution.

Hypothesis two
Making a sub-network selection based on machine learning predictions will decrease the computation time of the optimization model while not significantly deteriorating the recovery solution.

The case study supports both hypotheses. In the 565 disruptions that were tested, the decision support system (DSS) was able to find a feasible solution without constraint violations. Although only a limited case study was performed, the results are promising. Further research is required to further improve the performance of the system and the machine learning classifier. Nonetheless, hypothesis one is proven, since the case study proves that machine learning based predictions can be leveraged to find a feasible recovery solution.

Hypothesis two is related to both the computation time and the solution quality. First, the case study results shows that the sub-network selections consistently result in computation time reductions. On average the runtime is reduced by 42%. The computation times required for the disruptions of the case study ranged from 9 to 180 seconds, averaging around 48 seconds. In 98% of the runs, the best solution was found within 120 seconds. Second, on average the solution cost are increased by 6%. The DSS found the optimal solution in 98.91% of the runs and 91.7% of all non-trivial runs. These are strong indications that, by improving the classifier, hypothesis two is correct.

The research question for this thesis was formulated as:

Research question
How can machine learning techniques be incorporated into the aircraft recovery problem, and how does this integration help the recovery solution in terms of solution cost and computational time?

This research demonstrated that a machine learning classifier can be incorporated into the aircraft recovery problem. After a disruption occurs, all non-disrupted aircraft of the same aircraft family are scored by the classifier, resulting in a probability that the non-disrupted aircraft will help recover the disruption. Based on these probabilities, a sub-network selection algorithm selects the aircraft with the highest probability. This subset of aircraft is used to find a recovery solution to the disruption. The case study shows that this system is able to reduce the computation time by over 43% while increasing the solution cost by 6% on average.

Bibliography

Khaled F. Abdelghany, Ahmed F. Abdelghany, and Goutham Ekollu. An integrated decision support tool for airlines schedule recovery during irregular operations. *European Journal of Operational Research*, 185(2):825–848, 2008. ISSN 03772217. doi: 10.1016/j.ejor.2006.12.045.

Bruno Aguiar, Jose Torres, and António J M Castro. Operational Problems Recovery in Airlines - A Specialized Methodologies Approach. In *Progress in Artificial Intelligence*, volume 8154, 2013. ISBN 978-3-642-40668-3. doi: 10.1007/978-3-642-40669-0. URL `http://link.springer.com/10.1007/978-3-642-40669-0`.

M. Selim Aktürk, Alper Atamtürk, and Sinan Gürel. Aircraft Rescheduling with Cruise Speed Control. *Operations Research*, 62(4):829–845, 2014. ISSN 0030-364X. doi: 10.1287/opre.2014.1279. URL `http://pubsonline.informs.org/doi/abs/10.1287/opre.2014.1279`.

Michael Francis Arguello. *Framework for Exact Solutions and Heuristics for Approximate Solutions to Airlines Irregular Operations Control Aircraft Routing Problem.* PhD thesis, University of Texas at Austin, 1998.

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. *Proceedings - 8th EUROSIM Congress on Modelling and Simulation, EUROSIM 2013*, pages 265–270, 2015. doi: 10.1109/EUROSIM.2013.55.

Uğur Arikan, Sinan Gürel, and M Selim Aktürk. Flight Network-Based Approach for Integrated Airline Recovery with Cruise Speed Control. *Transportation Science*, 51(4):1259–1287, 2017. ISSN 15265447. doi: 10.1287/trsc.2016.0716.

Uğur Arıkan, Sinan Gürel, and M. Selim Aktürk. Integrated aircraft and passenger recovery with cruise time controllability. *Annals of Operations Research*, 236(2):295–317, 2016. ISSN 15729338. doi: 10.1007/s10479-013-1424-2.

Michael Ball, Cynthia Barnhart, Martin Dresner, Kevin Neels, Amedeo Odoni, Everett Peterson, Lance Sherry, Antonio Trani, and Bo Zou. Total Delay Impact Study. Technical report, 2010. URL `https://www.isr.umd.edu/NEXTOR`.

Jonathan F. Bard, Gang Yu, and Michael F. Arguello. Optimizing aircraft routings in response to groundings and delays. *IIE Transactions (Institute of Industrial Engineers)*, 33(10):931–947, 2001. ISSN 15458830. doi: 10.1080/07408170108936885.

Cynthia Barnhart. 1.206j airline schedule planning. `http://ocw.mit.edu`, 2003. [Online; accessed 09-October-2018].

Cynthia Barnhart, Douglas Fearing, and Vikrant Vaze. Modeling Passenger Travel and Delays in the National Air Transportation System. *Operations Research*, 62(3):580–601, 2014. ISSN 0030-364X. doi: 10.1287/opre.2014.1268. URL `http://pubsonline.informs.org/doi/abs/10.1287/opre.2014.1268`.

J. Bergstra and B. Yoshua. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 2012. ISSN 1532-4435. doi: 10.1162/153244303322533223.

Serge Bisaillon, Jean François Cordeau, Gilbert Laporte, and Federico Pasin. A large neighbourhood search heuristic for the aircraft and passenger recovery problem. *4OR*, 9(2):139–157, 2011. ISSN 16142411. doi: 10.1007/s10288-010-0145-5.

Stephane Bratu and Cynthia Barnhart. Flight operations recovery: New approaches considering passenger recovery. *Journal of Scheduling*, 9(3):279–298, 2006. ISSN 10946136. doi: 10.1007/s10951-006-6781-0.

Leo Breiman, Jerome Friedman, Charles Stone, and RA Olshen. Classification and Regression Trees (Wadsworth Statistics/Probability). *New York: CRC Press*, 1984.

António J M Castro, Ana Paula Rocha, and Eugénio Oliveira. *A New Approach for Disruption Management in Airline Operations Control*, volume 562 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-662-43372-0. doi: 10.1007/978-3-662-43373-7. URL `http://link.springer.com/10.1007/978-3-662-43373-7`.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 2002. ISSN 10769757. doi: 10.1613/jair.953.

Jens Clausen, Allan Larsen, Jesper Larsen, and Natalia J. Rezanova. Disruption management in the airline industry-Concepts, models and methods. *Computers and Operations Research*, 37 (5):809–821, 2010. ISSN 03050548. doi: 10.1016/j.cor.2009.03.027.

Andrew Cook, Graham Tanner, and Adrian Lawes. The hidden cost of airline unpunctuality. *Journal of Transport Economics and Policy*, 46(2):157–173, 2012. ISSN 00225258. doi: 10.2307/24396360.

Chuangyin Dang and Yinyu Ye. A fixed point iterative approach to integer programming and its distributed computation. *Fixed Point Theory and Applications*, 2015(1), 2015. ISSN 16871812. doi: 10.1186/s13663-015-0429-8.

Niklaus Eggenberg, Matteo Salani, and Michel Bierlaire. Constraint-specific recovery network for solving airline recovery problems. *Computers and Operations Research*, 37(6):1014–1026, 2010. ISSN 03050548. doi: 10.1016/j.cor.2009.08.006.

Qiang Gao, Xiao Wei Tang, and Jin Fu Zhu. 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*, pages 1469–1475, 2009. ISBN 9781424449149. doi: 10.1109/GSIS.2009.5408145.

Lotfy K. Hassan. Aircraft Disruption Management - Case Study Results and DSS Verification Results. 2019. URL `http://doi.org/10.4121/uuid:aec54216-ef2f-4f08-978d-29a9cb3c3c46`.

Lotfy K. Hassan, Bruno F. Santos, and J Vink. Airline Disruption Management: A Literature Review and Practical Challenges. 2019. URL `https://surfdrive.surf.nl/files/index.php/s/PGnmB3vLjm3PU8T`.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research*. McGraw-Hill, New York, NY, USA, tenth edition, 2015.

M Hossin and M.N Sulaiman. A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process*, 2015. ISSN 2231007X. doi: 10.5121/ijdkp.2015.5201.

Yuzhen Hu, Baoguang Xu, Jonathan F. Bard, Hong Chi, and Min'Gang Gao. Optimization of multi-fleet aircraft routing considering passenger transiting under airline disruption. *Computers and Industrial Engineering*, 80:132–144, 2015. ISSN 03608352. doi: 10.1016/j.cie.2014.11.026.

Yuzhen Hu, Yan Song, Kang Zhao, and Baoguang Xu. Integrated recovery of aircraft and passengers after airline operation disruption based on a GRASP algorithm. *Transportation Research Part E: Logistics and Transportation Review*, 87:97–112, 2016. ISSN 13665545. doi: 10.1016/j.tre.2016.01.002.

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, 2017. ISSN 09574174. doi: 10.1016/j.eswa.2017.04.031.

Niloofar Jafari and Seyed Hessameddin Zegordi. The airline perturbation problem: Considering disrupted passengers. *Transportation Planning and Technology*, 33(2):203–220, 2010. ISSN 03081060. doi: 10.1080/03081061003643788.

Niloofar Jafari and Seyed Hessameddin Zegordi. Simultaneous recovery model for aircraft and passengers. In *Journal of the Franklin Institute*, volume 348, pages 1638–1655, 2011. doi: 10.1016/j.jfranklin.2010.03.012.

Jin Huang and C.X. Ling. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):299–310, 3 2005. ISSN 1041-4347. doi: 10.1109/TKDE.2005.50. URL `http://ieeexplore.ieee.org/document/1388242/`.

N. Jozefowiez, C. Mancel, and F. Mora-Camino. A heuristic approach based on shortest path problems for integrated flight, aircraft, and passenger rescheduling under disruptions. *Journal of the Operational Research Society*, 64(3):384–395, 2013. ISSN 01605682. doi: 10.1057/jors.2012.20.

S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, 26(3):159–190, 11 2006. ISSN 0269-2821. doi: 10.1007/s10462-007-9052-3. URL `http://link.springer.com/10.1007/s10462-007-9052-3`.

Meilong Le, Jinmin Gao, and Chenxu Zhan. Solving the Airline Recovery Problem Based on Vehicle Routing Problem with Time Window Modeling and Genetic Algorithm. pages 822–828, 2013.

Ledislav Lettovsky. *Airline Operations Recovery: An Optimization Approach*. PhD thesis, Georgia Intitute of Technology, 1997.

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, 2010. ISSN 09574174. doi: 10.1016/j.eswa.2009.07.068. URL http://dx.doi.org/10.1016/j.eswa.2009.07.068.

Stephen J. Maher. A novel passenger recovery approach for the integrated airline recovery problem. *Computers and Operations Research*, 57:123–137, 2015. ISSN 03050548. doi: 10.1016/j.cor.2014.11.005.

Stephen J. Maher. Solving the Integrated Airline Recovery Problem Using Column-and-Row Generation. *Transportation Science*, 50(1):216–239, 2016. ISSN 0041-1655. doi: 10.1287/trsc.2014.0552. URL http://pubsonline.informs.org/doi/10.1287/trsc.2014.0552.

R. Mansi, S. Hanafi, C. Wilbaut, and F. Clautiaux. Disruptions in the airline industry: Math-heuristics for re-assigning aircraft and passengers simultaneously. *European Journal of Industrial Engineering*, 6(6):690–712, 2012. ISSN 17515254. doi: 10.1504/EJIE.2012.051074.

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, 2017. ISSN 0041-1655. doi: 10.1287/trsc.2015.0609.

Randal S. Olson, William La Cava, Zairah Mustahsan, Akshay Varik, and Jason H. Moore. Data-driven advice for applying machine learning to bioinformatics problems. In Russ B Altman, A Keith Dunker, Lawrence Hunter, Marylyn D Ritchie, Tiffany A Murray, and Teri E Klein, editors, *Proceedings of the Pacific Symposium of Biocomputing 2018*, pages 192–203. WORLD SCIENTIFIC, 2018. ISBN 978-981-3235-52-6. doi: 10.1142/9789813235533$_0$018.

Jon D. Petersen, Gustaf Sölveling, John-Paul Clarke, Ellis L. Johnson, and Sergey Shebalov. An Optimization Approach to Airline Integrated Recovery. *Transportation Science*, 46(4):482–500, 2012. ISSN 0041-1655. doi: 10.1287/trsc.1120.0414. URL http://pubsonline.informs.org/doi/abs/10.1287/trsc.1120.0414.

Romesh Ranawana and Vasile Palade. Optimized Precision - A New Measure for Classifier Performance Evaluation. In *2006 IEEE International Conference on Evolutionary Computation*, pages 2254–2261. IEEE, 2006. ISBN 0-7803-9487-9. doi: 10.1109/CEC.2006.1688586. URL http://ieeexplore.ieee.org/document/1688586/.

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, 2 2014. ISSN 03772217. doi: 10.1016/j.ejor.2013.08.034. URL http://linkinghub.elsevier.com/retrieve/pii/S0377221713007182.

Karine Sinclair, Jean-François Cordeau, and Gilbert Laporte. A column generation post-optimization heuristic for the integrated aircraft and passenger recovery problem. *Computers & Operations Research*, 65:42–52, 1 2016. ISSN 03050548. doi: 10.1016/j.cor.2015.06.014.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. 6 2012. URL https://arxiv.org/abs/1206.2944.

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*, pages 398–405, 2015. ISBN 978-989-758-073-4. doi: 10.5220/0005205303980405. URL http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005205303980405.

Carolin Strobl, Anne Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 2007. ISSN 14712105. doi: 10.1186/1471-2105-8-25.

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 03772217. doi: 10.1016/0377-2217(84)90207-8.

Dušan Teodorović and Goran Stojković. Model for operational daily airline scheduling. *Transportation Planning and Technology*, 14(4):273–285, 1990. ISSN 0308-1060. doi: 10.1080/03081069008717431. URL http://www.tandfonline.com/doi/abs/10.1080/03081069008717431?journalCode=gtpt20.

Dušan Teodorović and Goran Stojković. Model to Reduce Airline Schedule Disturbances. *Journal of Transportation Engineering*, 121(4):324–331, 1995. ISSN 0733-947X. doi: 10.1061/(ASCE)0733-947X(1995)121:4(324).

Benjamin G. Thengvall, Jonathan F. Bard, and Gang Yu. Balancing user preferences for aircraft schedule recovery during irregular operations. *IIE Transactions (Institute of Industrial Engineers)*, 32(3):181–193, 2000. ISSN 0740817X. doi: 10.1080/07408170008963891.

United States Department of Transportation - Bureau of Transportation Statistics. Air carrier financial: Schedule p-5.2. `https://www.transtats.bts.gov/DatabaseInfo.asp?DB_ID=135`, 2018a. [Online; accessed 12-November-2018].

United States Department of Transportation - Bureau of Transportation Statistics. Reporting carrier on-time performance database (1987-present). `https://www.transtats.bts.gov/DatabaseInfo.asp?DB_ID=120`, 2018b. [Online; accessed 23-September-2018].

United States Federal Aviation Administration. N-number inquiry database. `https://registry.faa.gov/aircraftinquiry/`, 2018. [Online; accessed 24-September-2018].

J Vink, B F Santos, and W J C Verhagen. Selection Heuristic for Solving the Dynamic Aircraft Recovery Problem. *Unpublished*, 2019. URL `https://surfdrive.surf.nl/files/index.php/s/yFwBQgIWDXRMxsn`.

Jeroen Vink. Aircraft routing recovery. Master's thesis, Delft University of Technology, 11 2016. `https://surfdrive.surf.nl/files/index.php/s/czcqasHkWUqA7rb`.

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

Gary M Weiss. Foundations of Imbalanced Learning. *Imbalanced Learning*, 2013. doi: 10.1002/9781118646106.ch2.

Cong Cong Wu and Meilong Le. A New Approach To Solve Aircraft Recovery Problem. In *Proceedings of the Second International Conference on Advanced Communications and Computation (INFO- COMP 2012)*, page 148–154. IARIA, 2012. ISBN 9781612082264.

Zhengtian Wu, Qing Gao, Benchi Li, Chuangyin Dang, and Fuyuan Hu. A Rapid Solving Method to Large Airline Disruption Problems Caused by Airports Closure. *IEEE Access*, 5:26545–26555, 2017a. ISSN 21693536. doi: 10.1109/ACCESS.2017.2773534.

Zhengtian Wu, Benchi Li, and Chuangyin Dang. Solving Multiple Fleet Airline Disruption Problems Using a Distributed-Computation Approach to Integer Programming. *IEEE Access*, 5: 19116–19131, 2017b. ISSN 21693536. doi: 10.1109/ACCESS.2017.2747155.

Zhengtian Wu, Benchi Li, Chuangyin Dang, Fuyuan Hu, Qixin Zhu, and Baochuan Fu. Solving long haul airline disruption problem caused by groundings using a distributed fixed-point computational approach to integer programming. *Neurocomputing*, 269:232–255, 2017c. ISSN 18728286. doi: 10.1016/j.neucom.2017.02.091.

www.chioka.in. Differences between receiver operating characteristic auc (roc auc) and precision recall auc (pr auc). `http://www.chioka.in/differences-between-roc-auc-and-pr-auc/`, 2014. [Online; accessed 13-January-2019].

Zhao Xiuli and Guo Yanchi. Study on GRAPS-ACO Algorithm for Irregular Flight Rescheduling. In *2012 International Conference on Computer Science and Service System*, number 71073071, pages 266–269. IEEE, 8 2012. ISBN 978-0-7695-4719-0. 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, 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. *Proceedings - 2015 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2015*, pages 695–700, 2016. doi: 10.1109/SMC.2015.131.

Seyed Hessameddin Zegordi and Niloofar Jafari. Solving the Airline Recovery Problem By Using Ant Colony Optimization. *International Journal of Industrial Engineering & Production Research*, 21(3): 121–128, 2010. URL `http://ijiepr.iust.ac.ir/article-1-214-en.pdf`.

Cheng Zhang. Two-Stage Heuristic Algorithm for Aircraft Recovery Problem. *Discrete Dynamics in Nature and Society*, 2017, 2017. ISSN 1607887X. doi: 10.1155/2017/9575719.

Dong Zhang, H. Y.K. Henry Lau, and Chuhang Yu. A two stage heuristic algorithm for the integrated aircraft and crew schedule recovery problems. *Computers and Industrial Engineering*, 87:436–453, 2015. ISSN 03608352. doi: 10.1016/j.cie.2015.05.033.

Dong Zhang, Chuhang Yu, Jitamitra Desai, and H. Y.K.Henry Lau. A math-heuristic algorithm for the integrated air service recovery. *Transportation Research Part B: Methodological*, 84:211–236, 2016. ISSN 01912615. doi: 10.1016/j.trb.2015.11.016. URL `http://dx.doi.org/10.1016/j.trb.2015.11.016`.

Bo Zhu, Jin Fu Zhu, and Qiang Gao. A Stochastic Programming Approach on Aircraft Recovery Problem. *Mathematical Problems in Engineering*, 2015, 2015. ISSN 15635147. doi: 10.1155/2015/680609.

A

Subset of IATA US Airport Codes

IATA Code	Airport	City	State
ATL	Hartsfield-Jackson Atlanta Intl. Airport	Atlanta	GA
BDL	Bradley Intl. Airport	Windsor Locks	CT
BNA	Nashville Intl. Airport	Nashville	TN
BOS	Gen. Edward Lawrence Logan Intl. Airport	Boston	MA
BTR	Baton Rouge Metropolitan Airport	Baton Rouge	LA
BTV	Burlington International Airport	Burlington	VT
DAL	Dallas Love Field	Dallas	TX
DCA	Ronald Reagan Washington National Airport	Arlington	VA
DTW	Detroit Metropolitan Airport	Detroit	MI
HNL	Honolulu Intl. Airport	Honolulu	HI
HOU	William P. Hobby Airport	Houston	TX
IAH	George Bush Intercontinental Airport	Houston	TX
JFK	John F. Kennedy Intl. Airport	New York	NY
KOA	Kona Intl. Airport at Keahole	Kailua/Kona	HI
LAS	McCarran Intl. Airport	Las Vegas	NV
LAX	Los Angeles Intl. Airport	Los Angeles	CA
LGA	LaGuardia Airport (Marine Air Terminal)	New York	NY
MCI	Kansas City Intl. Airport	Kansas City	MO
MCO	Orlando Intl. Airport	Orlando	FL
MEM	Memphis Intl. Airport	Memphis	TN
MIA	Miami Intl. Airport	Miami	FL
MLB	Melbourne Intl. Airport	Melbourne	FL
MSP	Minneapolis-Saint Paul Intl. Airport	Minneapolis	MN
ORD	Chicago O’Hare Intl. Airport	Chicago	IL
ORF	Norfolk Intl. Airport	Norfolk	VA
PBI	Palm Beach Intl. Airport	West Palm Beach	FL
PHL	Philadelphia Intl. Airport	Philadelphia	PA
PHX	Phoenix Sky Harbor Intl. Airport	Phoenix	AZ
PNS	Pensacola Intl. Airport	Pensacola	FL
PVD	Theodore Francis Green State Airport	Providence	RI
RSW	Southwest Florida Intl. Airport	Ft. Myers	FL
TPA	Tampa Intl. Airport	Tampa	FL

B
Overview Input Data

Chapter 4 discussed the framework and inputs to the decision support system. This appendix elaborates on the different files that are used by the DSS and their contents.

ACType.csv
Contains all information on the aircraft in the fleet. Within this file the following information is required:

- Tail number, e.g. N429DL
- Aircraft type, e.g. A320-212
- Aircraft family, e.g. A320
- Number of seats for business class passengers - no distinction is made between business & first class
- Number of seats for economy class passengers

OperatingCost.csv
This file contains the operating cost per aircraft type. The model uses direct operating costs (DOC) per block hour. These should reflect the costs of the airline.

TAT.csv
This file contains the minimal turn-around time (TAT) required per aircraft type. This is used to ensure that consecutive flights do not violate the required TAT for an aircraft.

RangesNM.csv
The range of all aircraft types in nautical miles. In combination with the distance between airports, this information is used to check if an aircraft can fly a certain route.

Schedule.h5
Contains information on the scheduled flights. For each flight the following information is required:

- Flight number, e.g. DL2340
- Tail number scheduled for flight
- Origin Airport IATA code, e.g. ORD
- Destination Airport IATA code
- Scheduled Time of Departure (STD) in UTC datetime format, e.g. 01-01-2015 17:10:00
- Scheduled Time of Arrival (STD) in UTC datetime format
- Economy passengers booked

- Business passengers booked

Disruptions.h5
Contains information on all disruptions in the dataset. This file should contain the following information:

- Flight number
- Tail number scheduled for flight
- Disruption type, e.g. delay or aircraft unavailability
- Disruption cause, e.g. Weather
- Disruption minutes
- Time found out in UTC datetime format
- Origin Airport IATA code
- Destination Airport IATA code
- Scheduled Time of Departure (STD) in UTC datetime format
- Scheduled Time of Arrival (STD) in UTC datetime format

Connections.h5
Passengers on an itinerary that consists of multiple flight legs are provided to the model using this file. This file contains the following information:

- Flight number of incoming flight
- Scheduled Time of Departure (STD) of incoming flight in UTC datetime format
- Scheduled Time of Arrival (STD) of incoming flight in UTC datetime format
- Flight number of connecting flight
- Scheduled Time of Departure (STD) of connecting flight in UTC datetime format
- Scheduled Time of Arrival (STD) of connecting flight in UTC datetime format
- Origin incoming flight
- Connecting airport
- Destination connecting flight
- Number of economy passengers on this itinerary
- Number of business passengers on this itinerary.

DistancesNM.csv
Distance between all airports in the schedule in nautical miles. Each flight leg of a route is a row. So for example one for ORD-ATL, and one for ATL-ORD.

Airports.csv
This file contains the minimum time required, in minutes, for passengers to connect to another flight at this airport.

Delay_cost_10min_split.csv
The resolution of the model is 10 minutes. This could be reduced to a smaller time step, but this will increase the required run-time of the model, since the problem then becomes more detailed. Within the delay costs, there is a split between soft and hard cost. Both are presented in Appendix C.

C Delay cost

Table C.1: Delay cost per passenger in USD per 10 minutes of delay

Delay [min]	Soft cost	Hard cost	Total cost
10	$0,45	$0,00	$0,45
20	$1,99	$0,00	$1,99
30	$5,13	$0,00	$5,13
40	$10,40	$0,00	$10,40
50	$18,08	$0,00	$18,08
60	$28,05	$10,00	$38,05
70	$39,88	$10,00	$49,88
80	$53,04	$10,00	$63,04
90	$67,06	$10,00	$77,06
100	$81,87	$10,00	$91,87
110	$96,68	$10,00	$106,68
120	$111,50	$15,00	$126,50
130	$126,31	$15,00	$141,31
140	$141,13	$15,00	$156,13
150	$155,94	$15,00	$170,94
160	$170,76	$15,00	$185,76
170	$185,57	$15,00	$200,57
180	$200,38	$15,00	$215,38
190	$215,20	$15,00	$230,20
200	$230,01	$15,00	$245,01
210	$244,83	$15,00	$259,83
220	$259,64	$15,00	$274,64
230	$274,45	$15,00	$289,45
240	$289,27	$20,00	$309,27
250	$304,08	$20,00	$324,08
260	$318,90	$20,00	$338,90
270	$333,71	$20,00	$353,71
280	$348,53	$20,00	$368,53
290	$363,34	$20,00	$383,34
300	$378,15	$20,00	$398,15
310	$392,97	$20,00	$412,97
320	$407,78	$20,00	$427,78
330	$422,60	$20,00	$442,60
340	$437,41	$20,00	$457,41
350	$452,22	$20,00	$472,22
360	$467,04	$40,00	$507,04
370	$481,85	$40,00	$521,85
380	$496,67	$40,00	$536,67
390	$511,48	$40,00	$551,48
400	$526,30	$40,00	$566,30
410	$541,11	$40,00	$581,11
420	$555,92	$40,00	$595,92
430	$570,74	$40,00	$610,74
440	$585,55	$40,00	$625,55
450	$600,37	$40,00	$640,37
460	$615,18	$40,00	$655,18
470	$629,99	$40,00	$669,99
480	$644,81	$250,00	$894,81
490	$659,62	$250,00	$909,62
500	$674,44	$250,00	$924,44
510	$689,25	$250,00	$939,25
520	$704,07	$250,00	$954,07
530	$718,88	$250,00	$968,88
540	$733,69	$250,00	$983,69
550	$748,51	$250,00	$998,51
560	$763,32	$250,00	$1.013,32
570	$778,14	$250,00	$1.028,14
580	$792,95	$250,00	$1.042,95
590	$807,76	$250,00	$1.057,76
600	$822,58	$250,00	$1.072,58

D

Optimization Model

This appendix presents the mathematical formulation of the optimization model used in this research. Chapter 5 explains the optimization model line-by-line.

Sets

F	set of flights
A	set of airports
E	set of aircraft types
P	set of aircraft
P(e)	set of aircraft *p* of type *e*
N	set of all nodes = $N_O \cup N_I \cup N_G \cup N_S$
N_O	set of origin nodes
N_I	set of intermediate nodes
N_S	set of sink nodes
T	set of delay steps
S	set of slack variables

Indices

i	flight index
t	delay time index
a	airport index
p	aircraft index
e	aircraft type index
n	node index
j	artificial variable index

Parameters

β_{bus}	Cost multiplier for business passengers
C_{canx}	Additional hard cost per passenger in case of cancellation
$C_{conn_{i,t}}$	Delay cost for connecting passengers on flight i, for delay time step t
$C_{D_{i,t}}$	Cost of delay for flight i, for delay time step t
C_{DS_t}	Soft cost of delay for delay time step t
C_{DH_t}	Hard cost of delay for delay time step t
$C_{OP_{p,i}}$	Operating cost of flight i with aircraft p
C_{C_i}	Cancellation cost of flight i
C_{G_n}	Cost of operating ground arc originating from node n
$C_{C_{SCH}}$	Penalty for operating a different aircraft than scheduled
C_{DOC_p}	Direct operating cost of aircraft p, per block hour

dist_i	Distance of flight i
orig_i	Origin airport of flight i
dest_i	Destination airport of flight i
STA_i	Scheduled Time of Arrival of flight i
STD_i	Scheduled Time of Departure of flight i
FT_i	Flight time of flight i, in hours
PaxY_i	Economy passengers on flight i
PaxJ_i	Business passengers on flight i

range_p	Range of aircraft p
seatsY_p	Economy seats on aircraft p
seatsJ_p	Business seats on aircraft p
h_n^e	Number of aircraft belonging to fleet e that should terminate at sink node $n \in \text{N}_S$
Dur_m	Duration of maintenance task m, as number of required consecutive ground arcs
TW	Number of time steps in time window
M	Big M cost factor
T_{swap}	Tail swap limit in minutes

Decision variables

$\delta_{F_{p,i}}$	= 1, if flight arc *i* is flown by aircraft *p*
$\delta_{D_{p,i,t}}$	= 1, if flight arc *i* is flown by aircraft *p* with delay time step *t*
δ_{C_i}	= 1, if flight *i* is cancelled
$\delta_{G_{p,n}}$	= 1, if aircraft *p* uses the ground arc originating from node *n*
$\delta_{F'_i}$	= 0, if flight *i* is flown by same aircraft *p* as scheduled
s_j	= 1, if problem is infeasible, one or more slack variables are part of the basic solution

Objective Function

$$min \sum_{p\in P}\sum_{i\in F} \delta_{F_{p,i}}\cdot C_{OP_{p,i}} + \sum_{p\in P}\sum_{i\in F}\sum_{t\in T} \delta_{D_{p,i,t}}\cdot(C_{OP_{p,i}}+C_{D_{i,t}}) + \sum_{i\in F}\delta_{C_i}\cdot C_{C_i} + \sum_{p\in P}\sum_{n\in N}\delta_{G_{p,n}}\cdot C_{G_n} + \sum_{i\in F}\delta_{F'_i}\cdot C_{C_{SCH}} + \sum_{j\in S} s_j\cdot M$$

$$\begin{aligned} C_{OP_{p,i}} &= C_{DOC_p}\cdot FT_i \\ C_{D_{i,t}} &= C_{conn_{i,t}} + (C_{DS_t} + C_{DH_t})\cdot(PaxY_i + \beta_{bus}\cdot PaxJ_i) \\ C_{C_i} &= (C_{DS_{max}} + C_{DH_{max}} + C_{canx})\cdot(PaxY_i + \beta_{bus}\cdot PaxJ_i) \end{aligned}$$

Time-space continuity constraints Aircraft flight coverage

$$\sum_{p\in P}\left[\delta_{F_{p,i}} + \sum_{t\in T}\delta_{D_{p,i,t}}\right] + \delta_{C_i} = 1 \qquad \forall i \in F \tag{D.1}$$

Origin node continuity

$$\delta_{G_{p,n}} + \sum_{i\in F_{out}}\delta_{F_{p,i}} + \sum_{i\in F_{out},t\in T}\delta_{D_{p,i,t}} = 1 \qquad \forall p\in P, n = \text{scheduled } N_O \text{ of } p \tag{D.2}$$

Node continuity

$$\left[\delta_{G_{p,n-1}} + \sum_{i\in F_{in}}\delta_{F_{p,i}} + \sum_{i\in F_{in},t\in T}\delta_{D_{p,i,t}}\right] - \left[\delta_{G_{p,n}} + \sum_{i\in F_{out}}\delta_{F_{p,i}} + \sum_{i\in F_{out},t\in T}\delta_{D_{p,i,t}}\right] = 0 \qquad \forall p\in P, n\in N_I \tag{D.3}$$

Sink node continuity - fix specific aircraft

$$\delta_{G_{p,n-1}} + \sum_{i\in F_{in}}\delta_{F_{p,i}} + \sum_{i\in F_{in},t\in T}\delta_{D_{p,i,t}} + s_j \geq 1 \qquad \forall p\in P, n = \text{scheduled } N_S \text{ of } p \tag{D.4}$$

Sink node continuity - fix aircraft type (i.e. allow tail swaps)

$$\sum_{p\in P(e)}\left[\delta_{G_{p,n-1}} + \sum_{i\in F_{in}}\delta_{F_{p,i}} + \sum_{i\in F_{in},t\in T}\delta_{D_{p,i,t}}\right] + s_j \geq h_n^e \qquad \forall e\in E, n\in N_S \tag{D.5}$$

Airline constraints

Aircraft seats capacity

$$\delta_{F_{p,i}} + \sum_{t \in T} \delta_{D_{p,i,t}} = 0 \qquad \forall p, i \text{ where } (\text{seatsY}_p < \text{PaxY}_i \wedge \text{seatsJ}_p < \text{PaxJ}_i) \tag{D.6}$$

Aircraft range

$$\delta_{F_{p,i}} + \sum_{t \in T} \delta_{D_{p,i,t}} = 0 \qquad \forall p, i \text{ where } (\text{range}_p < \text{dist}_i) \tag{D.7}$$

Original schedule penalty

$$\delta_{F_{p,i}} + \sum_{t \in T} \delta_{D_{p,i,t}} - \delta_{F'_i} = 1 \qquad \forall i \in F, p = \text{aircraft scheduled for } i \tag{D.8}$$

Tail swap limit constraint

$$\delta_{F_{p,i}} + \sum_{t \in T} \delta_{D_{p,i,t}} = 0 \qquad \forall p \in P, i \text{ where } STD_i - T_{now} < T_{swap} \tag{D.9}$$

Disruption constraints

Flight delay

$$\sum_{p \in P} \delta_{F_{p,i}} + \delta_{D_{p,i,t}} = 0 \qquad \forall t \in T \leq delay, i = \text{delayed flight} \tag{D.10}$$

Flight cancellation

$$\delta_{C_i} = 1, \qquad i = \text{cancelled flight} \tag{D.11}$$

Aircraft unavailability

$$\begin{gathered} \delta_{F_{p,i}} + \sum_{i \in F} \delta_{D_{p,i,t}} = 0 \\ \forall i \in F \text{ where } (\text{t}_{start} \leq \text{STD}_i \leq \text{t}_{end} \cup \text{t}_{start} \leq \text{STA}_i \leq \text{t}_{end}), \\ \forall t \in T \text{ where } (\text{t}_{start} \leq \text{STD}_i + t \leq \text{t}_{end} \cup \text{t}_{end} \leq \text{STA}_i + t \leq \text{t}_{end}) \end{gathered} \tag{D.12}$$

Draft Crew Recovery Model

Sets	
$K_{f_{sch}}$	set of scheduled flight crew
$K_{f_{res}}$	set of reserve flight crew
K_f	set of all flight crew = $K_{f_{sch}} \cup K_{f_{res}}$
$K_{c_{sch}}$	set of scheduled cabin crew
$K_{c_{res}}$	set of reserve cabin crew
K_c	set of all cabin crew = $K_{c_{sch}} \cup K_{c_{res}}$
K	set of all crew members = $K_f \cup K_c$
Q_k	set of eligible pairings for crew k

Indices	
k	crew index
q	pairing index

Pairing q is eligible for crew k if:

- pairing q starts at the source airport of crew k at beginning of time window
- pairing q ends at the sink airport of crew k at end of time window
- crew k can fly all aircraft types e included in pairing q
- all flight, duty and pairing legalities are satisfied

Parameters	
$C_{k,q}$	Cost of assigning crew k to pairing q
$C_{DHD,i}$	Cost of deadheading crew on flight $i \in$ F
$C_{DHDB,k}$	Cost of deadheading crew k back to base
γ_q^i	= 1, if flight $i \in$ F is part of pairing q
n_q	minimum number of cabin crew needed on pairing q

Decision variables	
$\delta_{k,q}$	= 1, if crew k is assigned to pairing q
v_k	= 1, if crew k is deadheaded back to base
x_i	= number of extra/surplus flight crew (deadheading) on flight $i \in$ F
y_i	= number of extra/surplus cabin crew (deadheading) on flight $i \in$ F

Objective Function

$$min \sum_{k \in K} \sum_{q \in Q_k} C_{k,q} \cdot \delta_{k,q} + \sum_{i \in F} C_{DHD,i} \cdot x_i + \sum_{i \in F} C_{DHD,i} \cdot y_i + \sum_{k \in K_{f_{sch}} \cup K_{c_{sch}}} v_k \cdot C_{DHDB,k}$$

Constraints

Flight crew coverage constraint

All non-cancelled flights should have a flight crew assigned

$$\sum_{k \in K_f} \sum_{q \in Q_k} \gamma_q^i \cdot \delta_{k,q} - x_i = 1 - \delta_{C_i} \qquad \forall i \in F \tag{D.13}$$

Cabin crew coverage constraint

All non-cancelled flights should have enough cabin crew assigned

$$\sum_{k \in K_c} \sum_{q \in Q_k} \gamma_q^i \cdot \delta_{k,q} - y_i = n_q \cdot (1 - \delta_{C_i}) \qquad \forall i \in F \tag{D.14}$$

Original crew constraint

All non-reserve crew that was originally scheduled, should either be assigned to flights or deadheaded back to base

$$\sum_{q \in Q_k} \delta_{k,q} + v_k = 1 \qquad \forall k \in K_{f_{sch}} \cup K_{c_{sch}} \tag{D.15}$$

E Case Study Features

Table E.1: Description of initial features for the classifier

Feature	**Description**
Iteration	The iteration number (for debugging purposes)
Candidate	The tail number of the candidate aircraft
Disrupted_AC	The tail number of the disrupted aircraft
Disrupted_Fl	The flight number of the disrupted flight
Disruption_type	The type of disruption
d_ac_type	The aircraft type of the disrupted aircraft
d_ac_family	The aircraft family of the disrupted aircraft
d_cap_pax_econ	Economy passengers capacity of the disrupted aircraft
d_cap_pax_buss	Business passengers capacity of the disrupted aircraft
d_pax_econ_fl	Economy passengers on the disrupted flight
d_pax_buss_fl	Business passengers on the disrtuped flight
d_pax_econ_max	Maximum No. Economy passengers on the flight string of the disrupted aircraft
d_pax_buss_max	Maximum No. Business passengers on the flight string of the disrupted aircraft
d_range	The range of the disrupted aircraft
d_range_flight	The flight distance of the disrupted flight
d_range_max	Maximum flight distance on the flight string of the disrupted aircraft
d_DOC	Direct Operating Cost per Hour of the disrupted aircraft
d_TAT	Turn Around Time of the disrupted aircraft
c_ac_type	The aircraft type of the candidate aircraft
c_ac_family	The aircraft family of the candidate aircraft
c_cap_pax_econ	Economy passengers capacity of the candidate aircraft
c_cap_pax_buss	Business passengers capacity of the candidate aircraft
c_econ_lf_mean	The mean economy passenger load factor of all flights scheduled for the candidate aircraft
c_econ_lf_std	The standard deviation of the economy passenger load factor of all flights scheduled for the candidate aircraft

Description of initial features for the classifier

Feature	**Description**
c_buss_lf_mean	The mean business passenger load factor of all flights scheduled for the candidate aircraft
c_buss_lf_std	The standard deviation of the business passenger load factor of all flights scheduled for the candidate aircraft
c_range	The ranfe of the candidate aircraft
c_DOC	Direct Operating Cost per Hour of the candidate aircraft
c_TAT	Turn Around Time of the candidate aircraft
same_airport_min_before	The minimum number of minutes the canidate aircraft is on the same airport of the disrupted flight before the STD/STA of the disrupted flight
same_airport_min_after	The minimum number of minutes the canidate aircraft is on the same airport of the disrupted flight after the STD/STA of the disrupted flight
same_airport_1hr_before	Indicates if the candidate aircraft is on the origin airport of the disrupted flight 1 hour before the STD of the disrupted flight
same_airport_2hr_before	Indicates if the candidate aircraft is on the origin airport of the disrupted flight 2 hour before the STD of the disrupted flight
same_airport_3hr_before	Indicates if the candidate aircraft is on the origin airport of the disrupted flight 3 hour before the STD of the disrupted flight
same_airport_1hr_after	Indicates if the candidate aircraft is on the origin airport of the disrupted flight 1 hour after the STD of the disrupted flight
same_airport_2hr_after	Indicates if the candidate aircraft is on the origin airport of the disrupted flight 2 hour after the STD of the disrupted flight
same_airport_3hr_after	Indicates if the candidate aircraft is on the origin airport of the disrupted flight 3 hour after the STD of the disrupted flight
sameFamily	Indicates if the candidate and disrupted aircraft are of the same family (for debugging purposes)
c_DOC_vs_d_DOC	d_DOC - c_DOC
c_TAT_vs_d_TAT	d_TAT - c_TAT
c_range_vs_d_range	c_range - d_range
c_range_vs_d_range_flight	c_range - d_range_flight
c_range_vs_d_range_max	c_range - d_range_max
c_pax_econ_vs_d_cap_econ	c_cap_pax_econ - d_cap_pax_econ
c_pax_buss_vs_d_cap_buss	c_cap_pax_buss - d_cap_pax_buss
c_pax_econ_vs_d_econ_fl	c_cap_pax_econ - d_pax_econ_fl
c_pax_buss_vs_d_buss_fl	c_cap_pax_buss - d_pax_buss_fl
c_pax_econ_vs_d_econ_max	c_cap_pax_econ - d_pax_econ_max
c_pax_buss_vs_d_buss_max	c_cap_pax_buss - d_pax_buss_max
conn_pax_econ_d_to_c	The number of connecting economy passengers from the disrupted aircraft to the candidate aircraft
conn_pax_buss_d_to_c	The number of connecting business passengers from the disrupted aircraft to the candidate aircraft
Result	Indicates if the candidate aircraft was used to recover the disruption

Table E.2: Description of features that were added after feature analysis

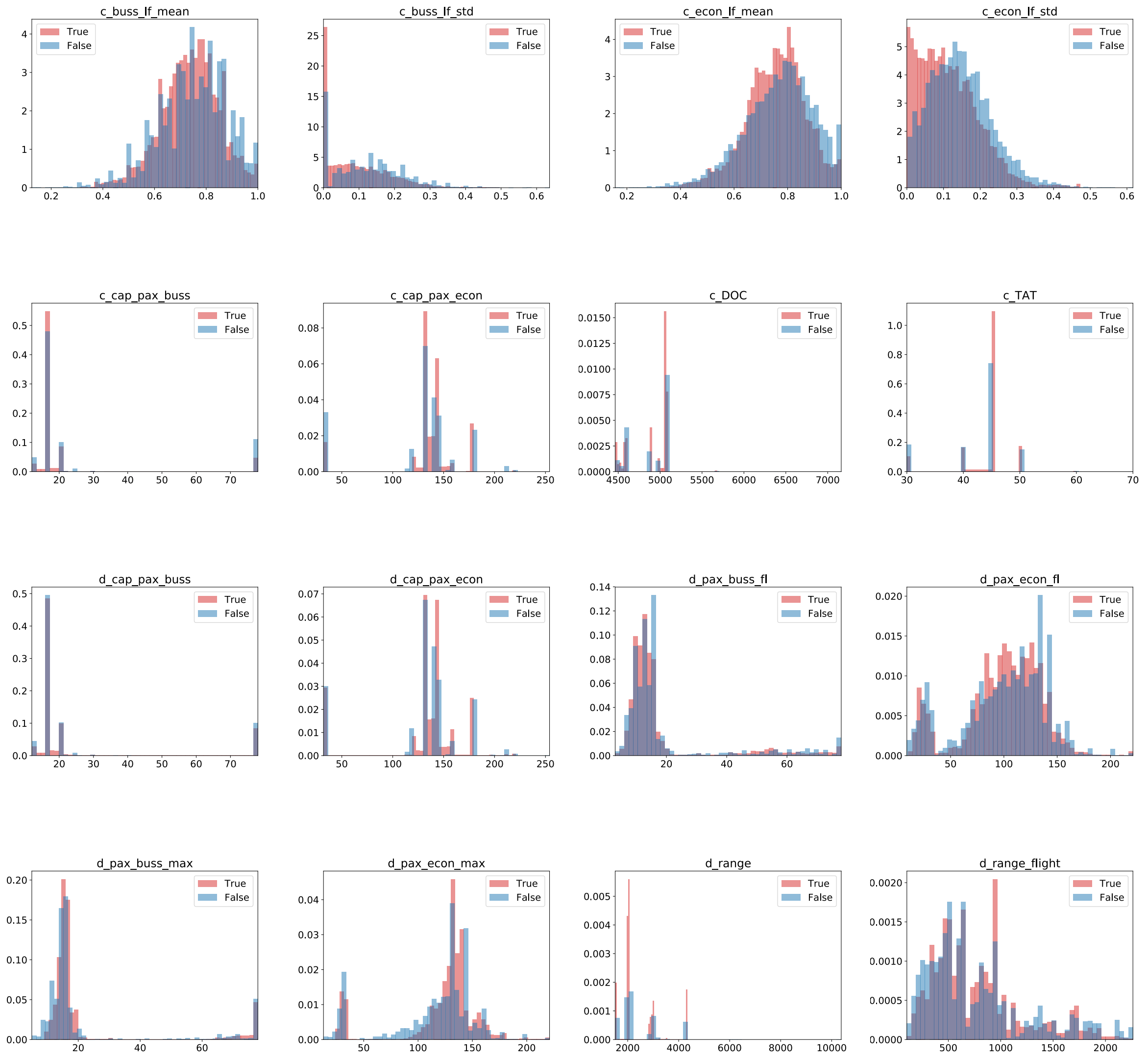
Feature	Description
Disruption_cause	The cause of the disruption
Disruption_duration	The duration of the disruption in minutes
c_no_flights	The number of flights scheduled for the candidate aircraft
c_flights_duration	The sum of the flight durations for the flights scheduled for the candidate aircraft
c_ground_time_d_orig_airport	The total time the candidate aircraft spends on the ground at the origin airport of the disrupted aircraft
d_dest_airport_1_hr_before	Indicates if the candidate aircraft is on the destination airport of the disrupted flight 1 hour before the STA of the disrupted flight
d_dest_airport_2_hr_before	Indicates if the candidate aircraft is on the destination airport of the disrupted flight 2 hour before the STA of the disrupted flight
d_dest_airport_3_hr_before	Indicates if the candidate aircraft is on the destination airport of the disrupted flight 3 hour before the STA of the disrupted flight
d_dest_airport_1_hr_after	Indicates if the candidate aircraft is on the destination airport of the disrupted flight 1 hour after the STA of the disrupted flight
d_dest_airport_2_hr_after	Indicates if the candidate aircraft is on the destination airport of the disrupted flight 2 hour after the STA of the disrupted flight
d_dest_airport_3_hr_after	Indicates if the candidate aircraft is on the destination airport of the disrupted flight 3 hour after the STA of the disrupted flight
d_dest_airport_min_before_min	The minimum number of minutes the canidate aircraft is on the destination airport of the disrupted flight before the STA of the disrupted flight
d_dest_airport_min_after_min	The minimum number of minutes the canidate aircraft is on the destination airport of the disrupted flight after the STA of the disrupted flight
c_ground_time_d_dest_airport	The total time the candidate aircraft spends on the ground at the dest airport of the disrupted aircraft
d_orig_airport_1_hr_before	Indicates if the candidate aircraft is on the origin airport of the disrupted flight 1 hour before the STD of the disrupted flight
d_orig_airport_2_hr_before	Indicates if the candidate aircraft is on the origin airport of the disrupted flight 2 hour before the STD of the disrupted flight
d_orig_airport_3_hr_before	Indicates if the candidate aircraft is on the origin airport of the disrupted flight 3 hour before the STD of the disrupted flight
d_orig_airport_1_hr_after	Indicates if the candidate aircraft is on the origin airport of the disrupted flight 1 hour after the STD of the disrupted flight
d_orig_airport_2_hr_after	Indicates if the candidate aircraft is on the origin airport of the disrupted flight 2 hour after the STD of the disrupted flight
d_orig_airport_3_hr_after	Indicates if the candidate aircraft is on the origin airport of the disrupted flight 3 hour after the STD of the disrupted flight
d_orig_airport_min_before_min	The minimum number of minutes the canidate aircraft is on the origin airport of the disrupted flight before the STD of the disrupted flight
d_orig_airport_min_after_min	The minimum number of minutes the canidate aircraft is on the origin airport of the disrupted flight after the STD of the disrupted flight

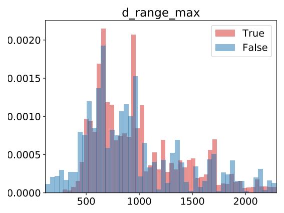
F Top 100 Feature Correlations

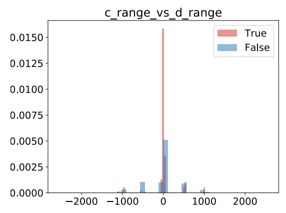
Feature 1	**Feature 2**	**Correlation**
Disruption_type_AC_Unavailable	Disruption_type_Delay	-1,00
c_ac_type_777-232	c_ac_family_777	1,00
d_ac_family_757	c_ac_family_757	1,00
d_ac_type_777-232LR	d_ac_family_777	1,00
d_ac_type_777-232LR	c_ac_type_777-232	1,00
d_ac_type_777-232LR	c_ac_family_777	1,00
d_ac_family_MD	c_ac_family_MD	1,00
d_ac_family_737	c_ac_family_737	1,00
d_ac_family_A320	c_ac_family_A320	1,00
d_ac_family_777	c_ac_family_777	1,00
d_ac_family_777	c_ac_type_777-232	1,00
d_ac_family_767	c_ac_family_767	1,00
c_TAT_vs_d_TAT	c_pax_buss_vs_d_cap_buss	1,00
d_cap_pax_buss	d_ac_type_717-200	0,99
c_cap_pax_buss	c_ac_type_717-200	0,99
c_econ_lf_std	c_buss_lf_std	0,99
c_econ_lf_mean	c_buss_lf_mean	0,99
c_pax_buss_vs_d_cap_buss	c_pax_buss_vs_d_buss_max	0,99
c_TAT_vs_d_TAT	c_pax_buss_vs_d_buss_max	0,99
d_cap_pax_buss	d_pax_buss_max	0,98
d_pax_buss_max	d_ac_type_717-200	0,98
c_TAT_vs_d_TAT	c_pax_econ_vs_d_cap_econ	0,98
c_pax_buss_vs_d_buss_fl	c_pax_buss_vs_d_buss_max	0,97
c_pax_econ_vs_d_cap_econ	c_pax_buss_vs_d_cap_buss	0,97
c_pax_buss_vs_d_cap_buss	c_pax_buss_vs_d_buss_fl	0,96
c_TAT_vs_d_TAT	c_pax_buss_vs_d_buss_fl	0,96
c_pax_econ_vs_d_cap_econ	c_pax_buss_vs_d_buss_max	0,96
d_pax_buss_fl	d_pax_buss_max	0,95
d_cap_pax_econ	d_TAT	0,95
d_cap_pax_buss	d_pax_buss_fl	0,94
c_cap_pax_econ	c_TAT	0,94
c_pax_econ_vs_d_cap_econ	c_pax_econ_vs_d_econ_max	0,94
d_pax_buss_fl	d_ac_type_717-200	0,94
c_pax_econ_vs_d_cap_econ	c_pax_buss_vs_d_buss_fl	0,92
d_cap_pax_econ	d_pax_econ_max	0,92
c_DOC_vs_d_DOC	c_pax_buss_vs_d_cap_buss	0,92
d_cap_pax_econ	d_ac_type_717-200	0,92
c_TAT_vs_d_TAT	c_pax_econ_vs_d_econ_max	0,91
c_cap_pax_econ	c_ac_type_717-200	0,91
c_DOC_vs_d_DOC	c_TAT_vs_d_TAT	0,91

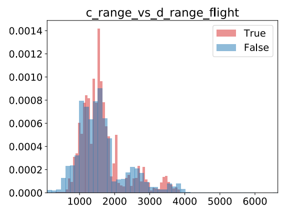
Feature 1	**Feature 2**	**Correlation**
c_DOC_vs_d_DOC	c_pax_buss_vs_d_buss_max	0,91
c_pax_buss_vs_d_cap_buss	c_pax_econ_vs_d_econ_max	0,90
c_pax_econ_vs_d_econ_fl	c_pax_econ_vs_d_econ_max	0,90
d_ac_family_767	c_ac_type_767-332	0,90
c_ac_type_767-332	c_ac_family_767	0,90
d_range	c_range	0,90
c_range_vs_d_range_flight	c_range_vs_d_range_max	0,90
d_pax_econ_max	d_TAT	0,89
d_TAT	d_ac_type_717-200	0,88
c_DOC_vs_d_DOC	c_pax_buss_vs_d_buss_fl	0,88
c_TAT	c_ac_type_717-200	0,88
d_cap_pax_econ	d_cap_pax_buss	0,87
c_pax_econ_vs_d_econ_max	c_pax_buss_vs_d_buss_max	0,87
d_cap_pax_econ	d_pax_buss_max	0,87
c_pax_econ_vs_d_cap_econ	c_pax_econ_vs_d_econ_fl	0,87
d_ac_type_767-332	c_ac_family_767	0,87
d_ac_type_767-332	d_ac_family_767	0,87
d_pax_econ_fl	d_pax_econ_max	0,87
d_pax_econ_max	d_ac_type_717-200	0,86
c_cap_pax_econ	c_cap_pax_buss	0,86
c_pax_buss_vs_d_buss_fl	c_pax_econ_vs_d_econ_max	0,86
c_DOC_vs_d_DOC	c_pax_econ_vs_d_cap_econ	0,85
c_TAT_vs_d_TAT	c_pax_econ_vs_d_econ_fl	0,84
d_cap_pax_buss	d_TAT	0,84
d_pax_buss_max	d_TAT	0,83
c_cap_pax_buss	c_TAT	0,83
c_pax_buss_vs_d_cap_buss	c_pax_econ_vs_d_econ_fl	0,83
d_cap_pax_buss	d_pax_econ_max	0,83
d_cap_pax_econ	d_pax_econ_fl	0,83
c_range	d_ac_family_757	0,83
c_range	c_ac_family_757	0,83
d_range	d_ac_family_757	0,82
d_range	c_ac_family_757	0,82
c_range	c_range_vs_d_range_max	0,82
c_range	c_range_vs_d_range_flight	0,82
d_cap_pax_econ	d_pax_buss_fl	0,82
d_ac_family_737	c_ac_type_737-832	0,81
c_ac_type_737-832	c_ac_family_737	0,81
c_pax_econ_vs_d_econ_fl	c_pax_buss_vs_d_buss_max	0,81
c_DOC_vs_d_DOC	c_pax_econ_vs_d_econ_max	0,80
d_pax_econ_fl	d_TAT	0,80
d_ac_type_767-332	c_ac_type_767-332	0,79
d_range	d_ac_family_MD	0,79
d_range	c_ac_family_MD	0,79
c_range	d_ac_family_MD	0,79
c_range	c_ac_family_MD	0,79
d_pax_buss_fl	d_TAT	0,78
d_pax_econ_max	d_pax_buss_max	0,78
d_range_flight	d_range_max	0,78
d_ac_type_737-832	d_ac_family_737	0,76
d_ac_type_737-832	c_ac_family_737	0,76
c_pax_econ_vs_d_econ_fl	c_pax_buss_vs_d_buss_fl	0,76
c_ac_type_757-232	c_ac_family_757	0,75
d_ac_family_757	c_ac_type_757-232	0,75
d_pax_econ_fl	d_ac_type_717-200	0,75
d_pax_buss_fl	d_pax_econ_max	0,75
c_DOC_vs_d_DOC	c_pax_econ_vs_d_econ_fl	0,73
d_ac_type_757-232	c_ac_family_757	0,73
d_ac_type_757-232	d_ac_family_757	0,73

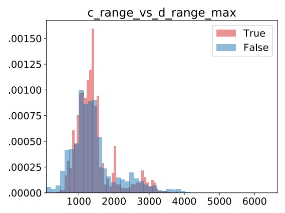
G
Case Study Feature Histograms

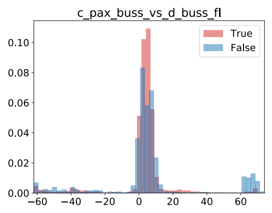


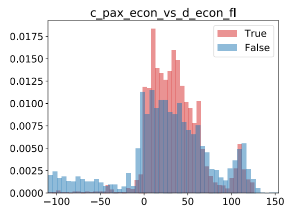


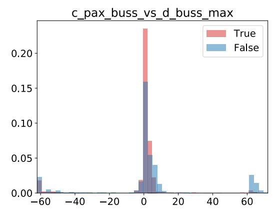


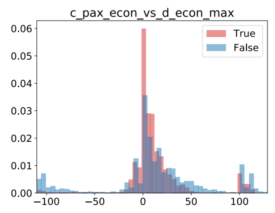


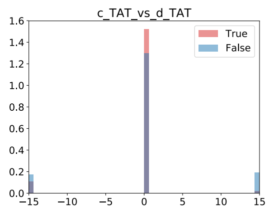


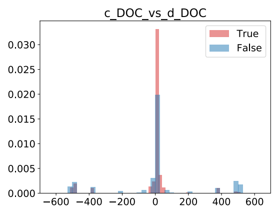


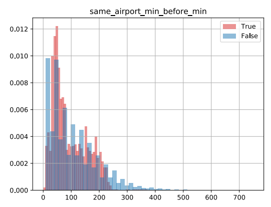


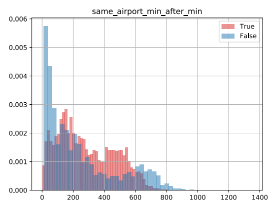




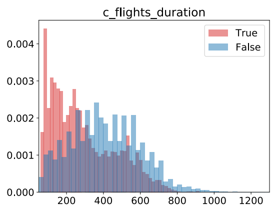


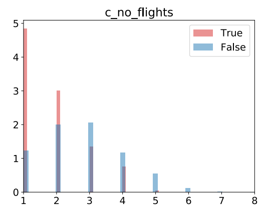


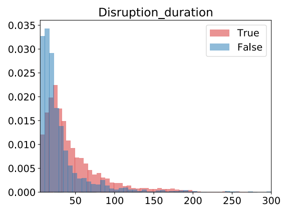


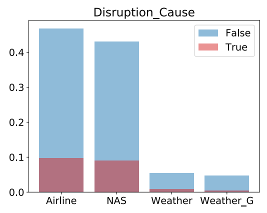


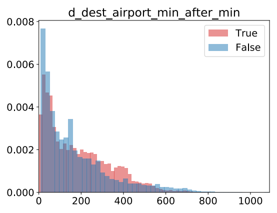
Features added after feature analyses

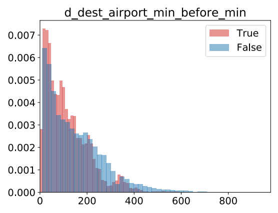


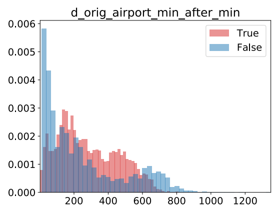


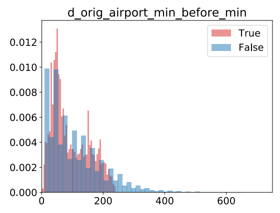


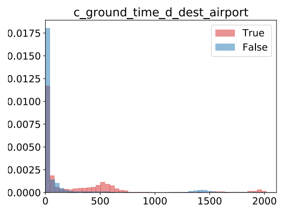


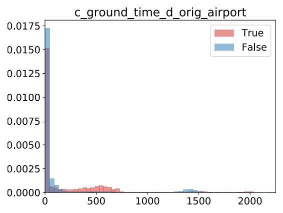




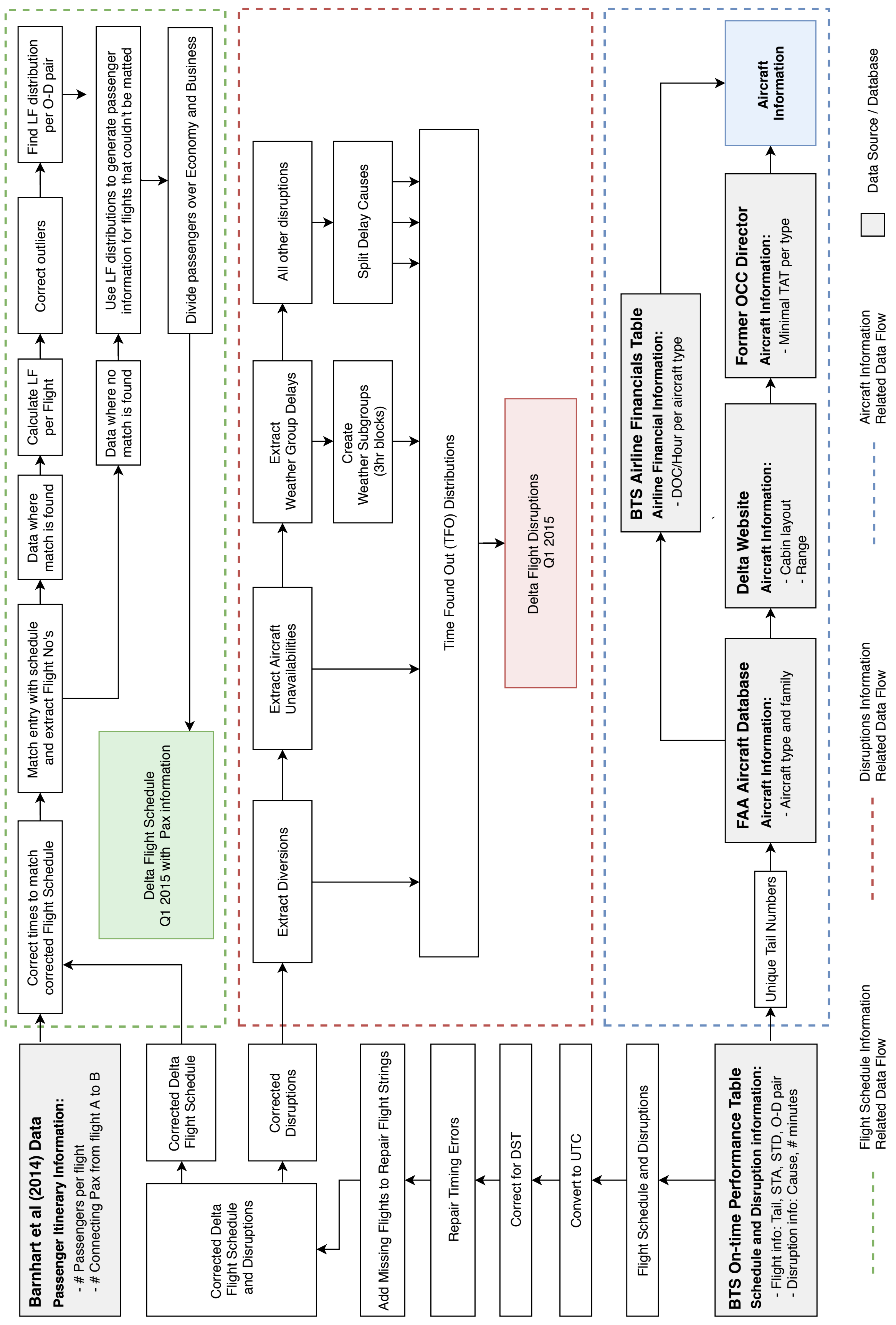








Case Study Data Processing Flowchart



BTS On-time Performance Table
Schedule and Disruption information:
- Flight info: Tail, STA, STD, O-D pair
- Disruption info: Cause, # minutes
Flight Schedule and Disruptions
Convert to UTC
Correct for DST
Repair Timing Errors
Add Missing Flights to Repair Flight Strings
Corrected Delta Flight Schedule and Disruptions
Corrected Delta Flight Schedule
Corrected Disruptions
Barnhart et al (2014) Data
Passenger Itinerary Information:
- # Passengers per flight
- # Connecting Pax from flight A to B
Correct times to match corrected Flight Schedule
Match entry with schedule and extract Flight No's
Data where match is found
Calculate LF per Flight
Correct outliers
Find LF distribution per O-D pair
Data where no match is found
Use LF distributions to generate passenger information for flights that couldn't be matted
Divide passengers over Economy and Business
Delta Flight Schedule Q1 2015 with Pax information
Extract Diversions
Extract Aircraft Unavailabilities
Extract Weather Group Delays
All other disruptions
Create Weather Subgroups (3hr blocks)
Split Delay Causes
Time Found Out (TFO) Distributions
Delta Flight Disruptions Q1 2015
Unique Tail Numbers
FAA Aircraft Database
Aircraft Information:
- Aircraft type and family
Delta Website
Aircraft Information:
- Cabin layout
- Range
Former OCC Director
Aircraft Information:
- Minimal TAT per type
Aircraft Information
BTS Airline Financials Table
Airline Financial Information:
- DOC/Hour per aircraft type
Flight Schedule Information Related Data Flow
Disruptions Information Related Data Flow
Aircraft Information Related Data Flow
Data Source / Database

BTS Delay Causes

The following delay cause classification and definitions were obtained from United States Department of Transportation - Bureau of Transportation Statistics. (2018b).

Carrier Delay
Carrier delay is within the control of the air carrier. Examples of occurrences that may determine carrier delay are: aircraft cleaning, aircraft damage, awaiting the arrival of connecting passengers or crew, baggage, bird strike, cargo loading, catering, computer, outage-carrier equipment, crew legality, damage by hazardous goods, engineering inspections, fueling, handling disabled passengers, late crew, lavatory servicing, maintenance, oversales, potable water servicing, removal of unruly passengers, slow boarding or seating, stowing carry-on baggage, weight and balance delays.

Late Aircraft Delay
Arrival delay at an airport due to the late arrival of the same aircraft at a previous airport. The ripple effect of an earlier delay at downstream airports is referred to as delay propagation.

NAS Delay
Delay that is within the control of the National Airspace System (NAS) may include: non-extreme weather conditions, airport operations, heavy traffic volume, air traffic control, etc. Delays that occur after Actual Gate Out are usually attributed to the NAS.

Security Delay
Security delay is caused by evacuation of a terminal or concourse, re-boarding of aircraft because of security breach, inoperative screening equipment and/or long lines in excess of 20 minutes at screening areas.

Weather Delay
Weather Delay is caused by extreme or hazardous weather conditions that are forecasted or manifest themselves on point of departure, enroute or on point of arrival.

J

Disruption Dataset Statistics per Type and Cause

Table J.1: Disruption statistics per type and cause

Type	Cause	Count	Share%	Duration Mean	Duration St.Dev	Duration Max.
AC Unavailability	Airline	773	3%	176	49	308
Delay	Airline	12757	43%	28	24	119
	NAS	13006	44%	29	28	306
	Security	11	0%	79	78	241
	Weather Cluster	1286	4%	39	50	306
	Weather	1542	5%	70	56	308

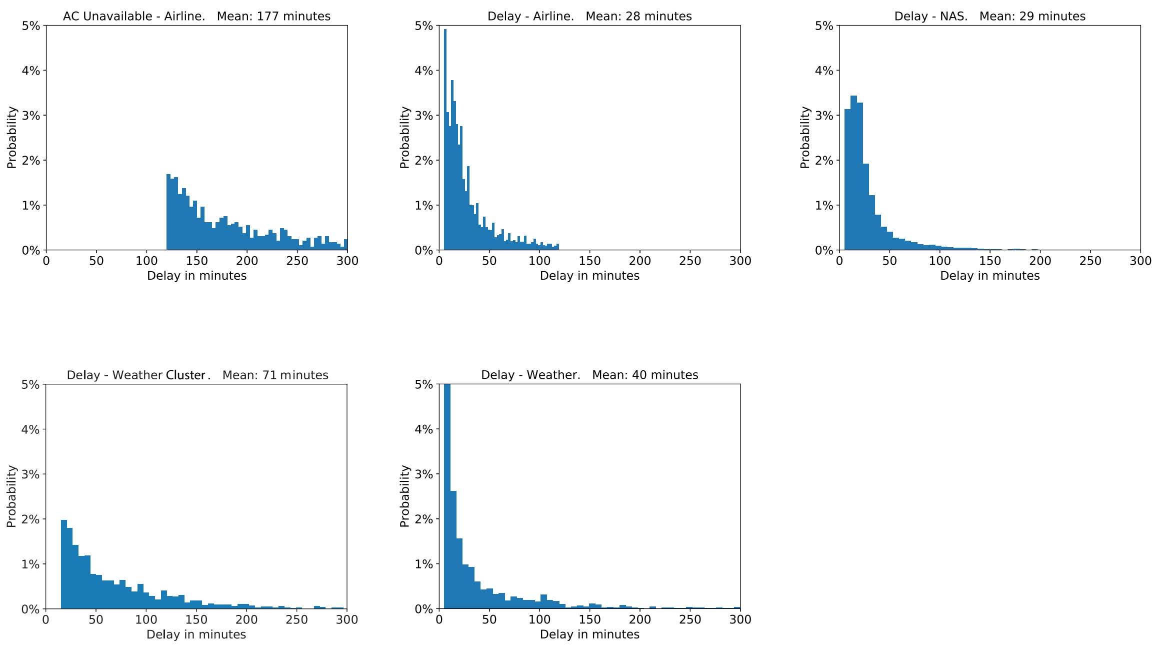


Figure J.1: Disruption duration histograms per disruption type and cause