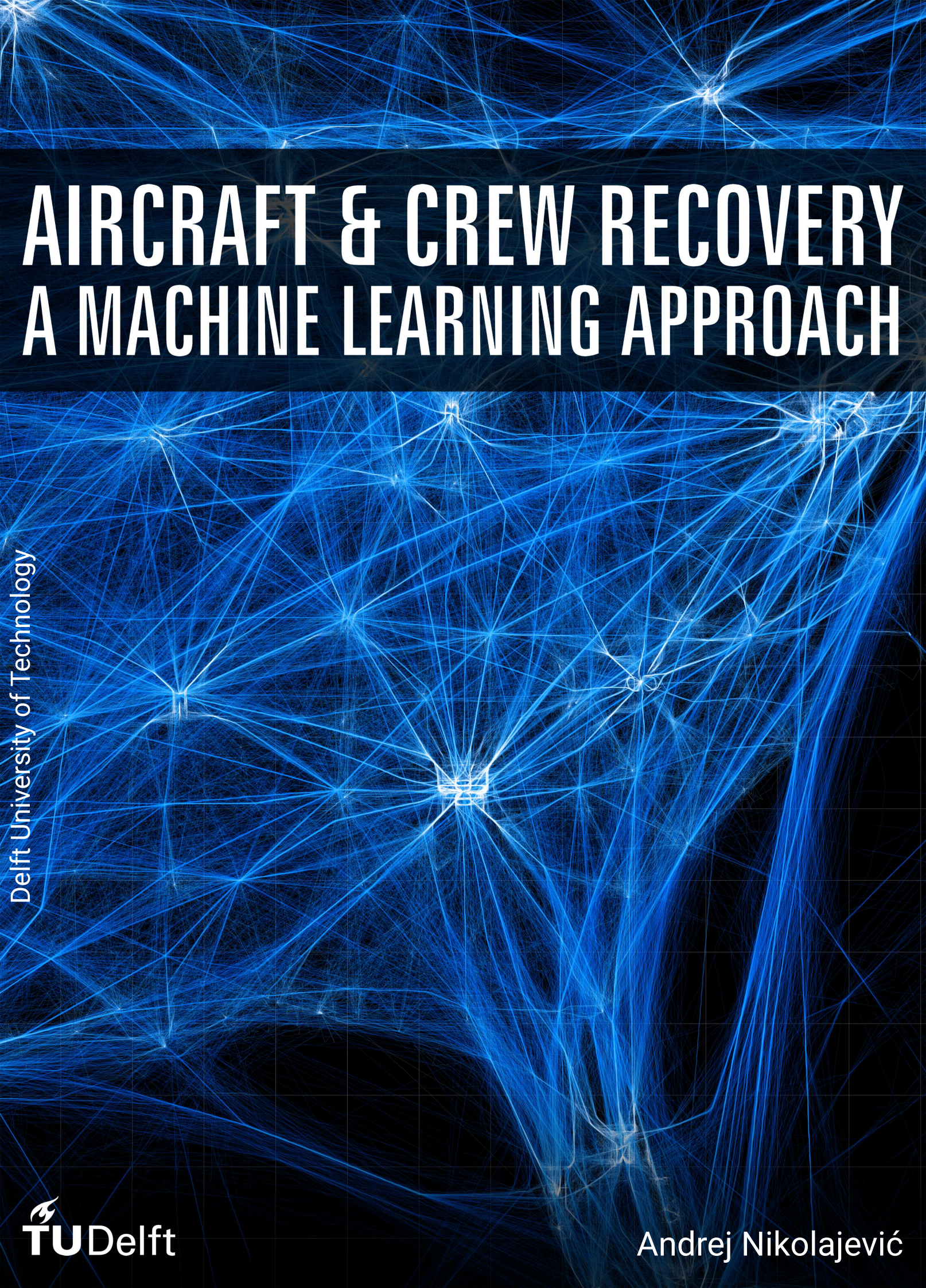
AIRCRAFT & CREW RECOVERY
A MACHINE LEARNING APPROACH



Delft University of Technology

TU Delft

Andrej Nikolajević

This page was intentionally left blank.

Aircraft and Crew Recovery

A Machine Learning Approach

by

Andrej Nikolajević

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on January 29th, 2020.

Student number:	4445953	
Project duration:	February 27, 2020 - January 29, 2021	
Thesis committee:	Prof. Dr. Dick G. Simons	TU Delft, Chairman
	Dr. ir. Bruno F. Lopes dos Santos	TU Delft, Supervisor
	Dr. Alessandro Bombelli	TU Delft, Examiner

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.
Cover photo obtained from the Flight Patterns project by Aaron Koblin and used with permission.

TU Delft

This page was intentionally left blank.

Acknowledgements

This report contains my thesis on airline disruption management, the final deliverable for the degree of Master of Science in Aerospace Engineering. This research comes as a continuation of a series of theses and publications relating to the airline recovery problem, the collective knowledge and research of which this thesis builds upon. Throughout the past year, I have had the opportunity to work on one of the most challenging problems faced in airline operations and learn more about the industry than I thought possible. Every step of the journey was an opportunity for personal and intellectual growth, and I am grateful to have had the opportunity to contribute to such an active, challenging, and relevant field. Though I stand as the sole author of this thesis, its completion would not be possible without the help of certain people whom I would like to thank.

First, I would like to thank my thesis supervisor Bruno Santos. Your lectures and courses were simultaneously the most challenging and interesting I've come across in my studies. Through the discussions we've had, your feedback helped make the final product of this thesis something I can be proud of.

Second, I would like to thank the people close to me for the overwhelming amount of support I have received throughout this project. Thank you Kristina for always being by my side and believing in me. Alejandro and Daniel, thank you for the countless laughs, support, feedback, and company throughout this project.

Finally, I would like to thank my family. Though we are often far away from each other, you're always with me in spirit. To my parents, this work has been made possible in no small part by your continued love and support through thick and thin. I am forever grateful for the opportunities you've enabled me to have.

Andrej Nikolajevic
Delft, January 2021

This page was intentionally left blank.

Contents

List of Figures

List of Tables

List of Abbreviations

AOCC	Airline Operations Control Center
DOC	Direct Operating Cost
DSS	Disruption Set Solver
FAA	Federal Aviation Administration
FN	False Negative
FP	False Positive
KPI	Key Performance Indicator
LP	Linear Programming
MILP	Mixed Integer Linear Programming
ML	Machine Learning
NP	Non-deterministic Polynomial-time
OTP	On-Time Performance
PR	Precision-Recall
RF	Random Forests
ROC	Receiver Operating Characteristic
SDSS	Sequential Disruption Set Solver
SMOTE	Synthetic Minority Oversampling Techniques
STA	Scheduled Time of Arrival
STD	Scheduled Time of Departure
SVM	Support Vector Machines
TFO	Time Found Out
TN	True Negative
TP	True Positive
TW	Time Window
XGB	Extreme Gradient Boosting

Introduction

When airspace congestion, poor weather conditions, or unavailability of crew results in flight delays or cancellations, the Airline Operations Control Center (AOCC) must attempt to resolve the airline recovery problem, comprising of the recovery of flight, aircraft, crew, and passenger schedules (Petersen et al. (2012)). Large airlines must solve this problem hundreds of times daily to ensure adherence to the planned schedule, all with the end-goal of minimising disruption-related costs. The process of solving of the airline recovery problem is referred to as disruption management and represents the operational period directly following the completion of the airline planning process, as seen in Figure 1.

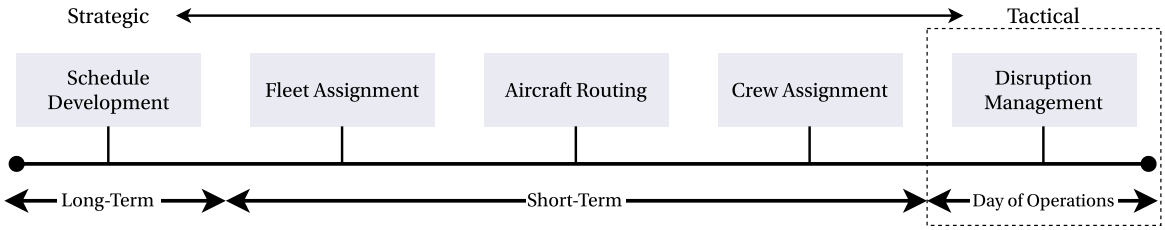


Figure 1: Airline disruption management as part of the larger airline planning process.
Adapted from Bouarfa et al. (2018).

The AOCC aims to preserve the work done on aircraft routing and fleet and crew assignment by monitoring day-to-day operations and taking action when necessary. Airline disruption management is a real-time process governed by (1) the time required to make a recovery decision and (2) the quality of said decision. Due to strict time limits on decision making, the AOCC has been solving this problem by hand, with experienced operators capable of intuitively making quick, good quality decisions. In the past two decades, the field of airline disruption management has made great strides towards the automation of this process and acquiring much higher quality solutions, enabling airlines to incur lower costs as a result of disruptions. Part II of this thesis reviews the history and state-of-the-art publications related to the airline recovery problem, extending the collection of modern literature published by Hassan et al. (2020). Whereas older literature often dealt with the recovery of a single resource (aircraft, crew, or passengers), the main focus in recent years has shifted to development of detailed decision-support systems that integrate the recovery of multiple resources.

Starting with the dynamic aircraft recovery formulation of Vos et al. (2015), the department of Air Transport and Operations at the TU Delft has completed several projects related to airline disruption management, many of which have been published. Bouarfa et al. (2018) developed a multi-agent system to tackle aircraft recovery, while Vink et al. (2020) and Hassan (2018) extended the work of Vos et al. (2015) by enabling faster solution times via use of heuristic and machine learning (ML) based aircraft selection algorithms. Hoeben et al. (2017) developed a tool to tackle the recovery of crew pairings. The objective of this thesis is to integrate the ML-assisted aircraft recovery formulation developed by Hassan (2018) with a newly-developed crew recovery formulation and supplement it with a ML classifier used to narrow the scope of the crew recovery problem. The research question encapsulating the objective of this thesis can be formulated as follows:

How can the use of machine learning methods be applied to the integrated airline recovery problem, and what are the effects of this integration on the solution quality and solution time?

With the aim of answering the main research question and pinpointing the main steps required to complete the research project, the following research sub-questions must be answered:

1. How can multiple resources be computationally efficiently integrated into the mathematical formulation of the aircraft and crew recovery problem?

2. Which simplifications to the aircraft and crew recovery problem formulation are acceptable in the context of industry practice?
3. Given the problem formulation, what form of machine learning algorithm and approach best suits the targeted solution time reduction?
4. How should the two main performance metrics, solution time and solution quality, be balanced during model assessment?

This thesis is split into three parts. Part I will contain the scientific paper resulting from the thesis research. This paper contains the bulk of the problem description, modelling approach, and results obtained as part of the research. Part II will present the literature review done as a prerequisite to the thesis project, elaborating on the history and progression of the airline recovery problem. Part III will contain the thesis report, which elaborates the developed model in more detail and presents supplementary material. The framework of the proposed model is explained in detail in Chapter 1, followed by a more detailed description of the crew recovery model in Chapter 2. Chapter 3 contains details on the acquired and generated data used in the case study. Chapter 4 discusses the model functionality and verification. The development process of the machine learning classifier used in crew selection is explained in more detail in Chapter 5. A sensitivity analysis is performed to test the performance of the selection algorithm for various parameters in Chapter 6, followed by the report conclusions and recommendations for future work in Chapter 7.

I

Scientific Paper

This page was intentionally left blank.

Aircraft and Crew Recovery: a Machine Learning Approach

Andrej Nikolajević

Student No. 4445953, Air Transport and Operations, Faculty of Aerospace Engineering, Delft University of Technology

Abstract

This paper presents a machine learning approach to the aircraft and crew recovery problem. The presented model utilises a two-stage sequential approach recovering the aircraft and flight schedule first, followed by that of cockpit crew. At each stage, the recovery is assisted via use of decision tree-based machine learning classifiers to select a subset of relevant aircraft and crew, reducing the scope of the recovery operation. The aircraft stage is able to directly account for aircraft maintenance constraints, and indirectly account for the impact of the aircraft recovery on crew schedules and passenger itineraries. In the crew stage, the model directly accounts for the common cockpit crew labour constraints set by airlines operating with the US. The combined performance of the recovery model is evaluated via a case study on the US network of Delta Airlines. The results show that the crew selection algorithm can find an optimal solution to the crew recovery problem in 89% of non-trivial disruption instances, completing in 27 s on average, with an average objective function value only 5% higher than optimal. When including aircraft recovery results, the proposed approach increased the percentage of instances that obtain a solution within the AOCC-defined time limit from 47% to 95%, speeding up the average computational time threefold. The solution to the aircraft and crew recovery problem was globally optimal in 83% of cases, with an average objective function value 11% higher than optimal.

Keywords: Disruption Management, Airline Industry, Irregular Operations, Aircraft Recovery, Crew Recovery, Passenger Recovery, Integrated Recovery, Combinatorial Optimization, Machine Learning

---

1. Introduction

Day-to-day airline operations are faced with a large number of differences between planned and actual aircraft, flight, and crew schedules. These originate from factors such as airspace congestion, poor weather conditions, and can even have a reactionary cause. Airlines and passengers alike incur large expenses as a result of disruptions, with estimates by Swiss et al. (2016) of up to $60 billion annually, or 8% of industry revenue. As an airline may face up to several hundred disruptions daily, it is essential that decisions are made quickly. The Airline Operations Control Center (AOCC) must actively take action to manage these disruptions in order to restore the original schedule of flights, aircraft, and crew, while ensuring that passengers are transported to their intended destinations. This is often in the form of flight delays and cancellations, rerouting of aircraft and crew, or use of reserve crew. These decisions come at the cost of additional fuel expenses, crew overtime, and passenger monetary compensation. Therefore, the main objective of the airline recovery problem is to find the minimum-cost solution that recovers the airline schedule back to the original so that operations may resume as planned. Due to complex inter-dependencies between recovery options, the computational time provided by software solutions is often unfit for operational use. Many airlines still solve the

airline recovery problem by hand, with experienced operators that look for a quick, feasible solution rather than an optimal one. Research on the airline recovery problem focuses therefore on automating the recovery decision making process in order to optimize the solution quality and reduce costs incurred by airlines within the decision-making time limit.

Background

The first instance of the use of software to resolve the aircraft recovery problem was the work of Teodorović and Guberinić (1984). The branch and bound algorithm was used to restore aircraft schedules with the objective of minimising total delay experienced by passengers. Lettovsky et al. (2000) were among the first to consider the recovery of crew after a disruption, utilising integer programming to minimise the cost of reassigning crew to flights. This paper was the result of the PhD thesis by Lettovsky (1997), which is the first attempt at integrating the recovery of aircraft, crew, and passengers in a single recovery operation via use of Benders decomposition. These papers formed a large part of the basis of modern literature on the airline recovery problem. Since then, the range of recovery options, disturbances, and network sizes has increased drastically to more accurately reflect the real-life case. Because of this, approaches presented in literature often exceed the solution times required by the AOCC.

Rosenberger et al. (2003) were among the first to utilise a heuristic selection algorithm to reduce the size of the aircraft recovery problem with the end goal of decreasing computational time. By exploring only a part of the solution space, the amount of time required to obtain a solution is drastically reduced. This concept has become a trend in the airline recovery problem, and was applied by Petersen et al. (2012), Vos et al. (2015), and Vink et al. (2020), among others. Vos et al. (2015) and Vink et al. (2020) approached the aircraft recovery problem dynamically, solving disruptions as they occur and building upon previously taken actions. The latter also indirectly accounts for the cost of the recovery decisions on passenger itineraries. In a similar way, Hoeben et al. (2017) use a selection algorithm to solve the crew recovery problem with a subset of crew. Though there is still a focus on the recovery of individual resources (aircraft, crew, passengers), modern literature places a large focus on integrating multiple resources to provide a more complete airline recovery solution. For a comprehensive overview of historical and recent literature relating to the airline recovery problem, the reader is referred to the literature reviews of Clausen et al. (2010) and Hassan et al. (2020).

Paper Contribution

This research contributes to the development of an integrated airline recovery solution with the end-goal of achieving fast, optimal solutions. Three novelties contributing to the field of the airline recovery problem result from this research. (1) The development of a time-space network based crew recovery model, enabling flexibility with respect to recovery decisions and accounting for the most common FAA-mandated cockpit crew labour constraints. (2) The implementation of a machine-learning based crew selection algorithm within the crew recovery stage. Though heuristic selection algorithms have been implemented, the use of machine learning for crew recovery in this context can consider many more features to make a more informed decision. (3) The integration of two time-space network based recovery stages to achieve an aircraft and crew recovery solution that can directly account for the presence of multiple aircraft types, aircraft maintenance constraints, crew labour constraints, presence or absence of reserve crew, and indirectly account for the recovery impact on passengers.

Report Structure

This paper is structured as follows: Chapter 2 describes the problem formulation and main challenges of the airline recovery problem. Chapter 3 describes the methodology and frameworks used within the proposed

approach to solve the aircraft and crew recovery problem. Chapter 4 describes the acquired and generated aircraft, flight, crew, and disruption data used within the problem. Chapter 5 presents the development and evaluation of the machine learning classifier used in the crew recovery stage. Chapter 6 presents the results of the case study performed on a set of real disruptions experienced by Delta Airlines in February 2015. The paper concludes with Chapter 7.

2. Problem Formulation

Following a disruption, the flight schedule, and the planned routes of aircraft, crew, and passengers are broken and must be recovered to resume normal operations. The recovery of the schedules of these four resources constitutes the airline recovery problem. Often, the airline recovery problem is formulated as a limited-resource, smaller-scale airline scheduling problem, with a time window within which the recovery operation takes place. It is traditionally solved sequentially, with aircraft and flight schedules being recovered first, followed by those of crew, and finally those of passengers. Restoring the schedule is no trivial task. Actions taken in recovery of one resource can directly affect the recovery of others, and the trend of increasing levels of detail considered in recent literature often results in solution times that are unfit for operational use. According to Clausen et al. (2010), in an operational setting, the AOCC requires decisions to be made in 2 minutes or less. Given this strict time limit, many simplifications exist in published literature in order to bring solution times closer to the needs of the AOCC.

As rules relating to re-scheduling of aircraft are less strict than those relating to crew, the aircraft recovery problem in particular has received the most attention in literature. When making an aircraft recovery decision, the differences in aircraft type, planned maintenance, and airport slot restrictions must be considered by the AOCC. The common actions traditionally taken in aircraft recovery are tail swaps, involving swapping the aircraft assigned to flights to accommodate changes in schedule, flight delays, and flight cancellations. Each of these recovery actions can have an effect on the scheduled routes of crew, passengers, and cargo. In the work of Aktürk et al. (2014) and Marla et al. (2017), cruise speed control is also considered as a recovery option. Altering the aircraft cruise speed for a flight can enable the absorption of small delays and provide additional opportunities for tail swaps if the flight arrives early.

Commercial flights have two types of crew: cockpit crew which operates the flight, and cabin crew responsible for passenger service. Cabin crew is not subject to stringent labour constraints and is much more readily available than cockpit crew. The recovery of crew therefore often focuses solely on cockpit crew. Strict work time and rest constraints often make cockpit crew recovery more complex than that of aircraft. Often times, no solution is possible when considering only scheduled crew. Therefore, airlines also assign reserve crew to prevent cancellations and increase the robustness of the flight schedule. Reserve crew duties are those that consider operation of flights in the case of absence of scheduled crew, and their role is crucial in maintaining feasible flight schedules. When recovering cockpit crew schedules, flight delays, flight cancellations, crew swaps, deadheading crew, and use of reserve crew are common recovery options.

Since the 2009 ROADEF Challenge, many publications focus on integrating the recovery of aircraft, crew, and passengers. This can either be done sequentially, where separate models for recovery of individual resources are run successively, or in one single recovery operation that considers the recovery of all resources simultaneously. The latter has the added benefit of being able to make a decision while evaluating the impact of recovery actions on all resources. Such approaches were applied by Abdelghany et al. (2008), Petersen et al. (2012), and Arikan et al. (2017), among others. The disadvantage of such an approach is exponential increase

in computational complexity, making integrated recovery unfeasible for large airlines. Though sequential recovery achieves better solution times, effects of recovery actions are only evaluated on a per-resource level.

3. Methodology

This paper presents a two-stage sequential aircraft and crew recovery formulation. The model is built upon the Disruption Set Solver (DSS) formulation originally developed by Vink et al. (2020) and modified by Hassan (2018). The work presented in this paper extends the random forest (RF) assisted aircraft recovery of Hassan (2018) with the development and integration of a time-space network based crew recovery model. This model is supplemented by an extreme gradient boosting (XGB) classifier that selects a subset of crew likely to be used in the optimal solution with the aim of reducing the solution time. As the entire formulation directly accounts for the recovery of aircraft and crew, and indirectly accounts for the recovery of passengers, the solution addresses nearly the entire scope of airline recovery. Throughout this paper, the model is referred to as the Sequential Disruption Set Solver (SDSS). The SDSS is a two-stage sequential airline recovery formulation based on binary linear programming (BIP). An overview of the two stages and their components is present in Figure 1.

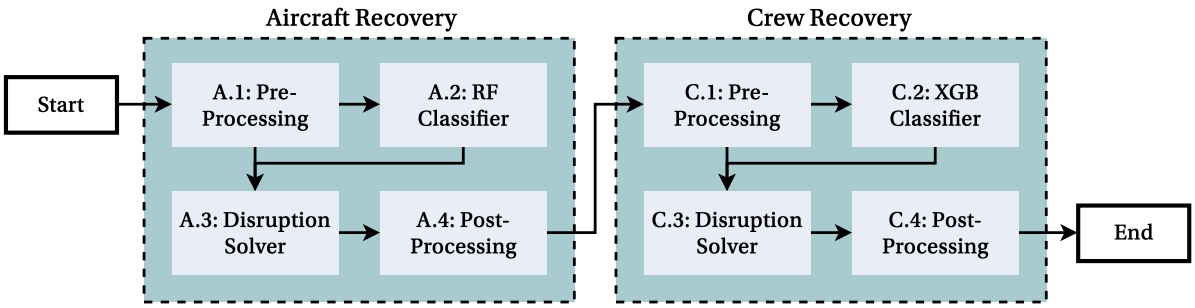


Figure 1: Flowchart of the two sequential recovery stages of the SDSS.

The SDSS relies on a parallel time-space network approach, where a separate time-space network is generated for each aircraft and crew. The nodes in a time-space network represent a location and a time for a given aircraft or crew. Arcs between nodes represent the movement of aircraft and crew through time and space. Those between two nodes at the same airport represent ground arcs, while those between airports are flight arcs. A schematic overview of the concept of parallel time-space networks is presented in Figure 2. Within a time-space network, time is discretized in homogeneous steps, and each aircraft or crew is assigned an origin and a sink node. Origin nodes represent a fixed start time and location for each aircraft and crew, while sink nodes represent the scheduled end-location and time within the time window. Nodes corresponding to times between the origin and sink node times are referred to as intermediate nodes. The advantage of using parallel time-space networks is that the movement of each individual aircraft and crew can be tracked separately. Additionally, aircraft maintenance and crew labour constraints can easily be implemented for specific aircraft and crew via use of sink nodes.

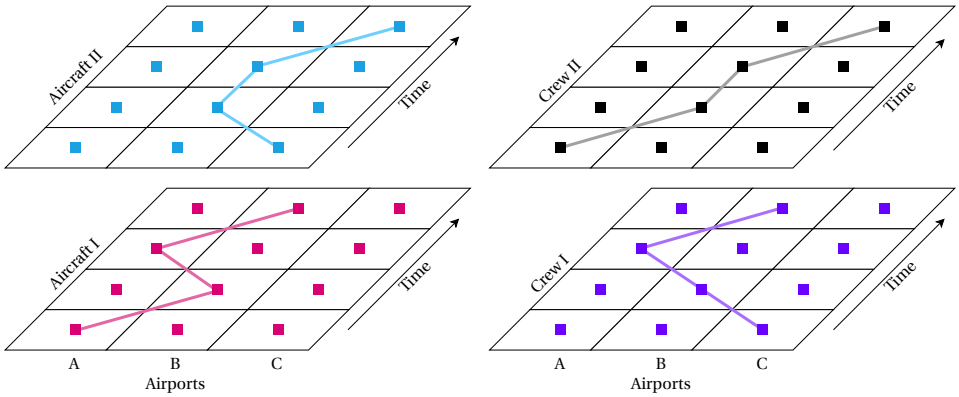


Figure 2: The principle behind parallel time-space networks as shown for aircraft and crew. Adapted from Vink et al. (2020).

Both the aircraft and crew stages make use of several assumptions simplifying the problem. The four main assumptions are present in the following list:

1. Time is discretized in homogeneous steps of 10 minutes.
2. Airport capacity and slot constraints are not considered.
3. Disruption information is static.
4. Crew are assumed to work in pairs.

The first assumption refers to the discretization of time time-space networks used in the model. Though most airlines discretize scheduled departure and arrival times with steps of 5 minutes, a 10-minute discretization is sufficient for the purposes of airline recovery. Air traffic control is responsible for finding a specific departure or arrival slot within the 10-minute range. The second assumption refers to delaying flights within the aircraft recovery stage. Many U.S. airports operate 24 hours a day, and only the largest lack the capacity to accommodate delay decisions made with no consideration for slot constraints. The third assumption implies that the information obtained about the disruption in terms of time of occurrence and severity is assumed not to change. In reality, disruptions may change in severity over time. The final assumption refers to the modelling of crew within the crew recovery stage. Each two-man cockpit crew is treated as a single resource. That is, two cockpit crew members assigned to a flight are assumed to always work together as a pair, with identical schedules and constraints. The following sections will detail the contents of each of the elements present within the aircraft and crew recovery stages, as well as the stages' mathematical formulations.

3.1. Aircraft Recovery

The aircraft recovery stage of the SDSS is functionally largely the same as that of Hassan (2018). The changes made relate to the addition of crew considerations to the aircraft recovery stage, and the restructuring of the aircraft recovery stage to accommodate the addition of crew recovery. Though a larger focus is placed on the evaluation of the crew recovery stage, the understanding of the four main elements of the aircraft recovery stage as presented in Figure 1 is essential to gain an overview of the functionality of the entire SDSS framework. In the pre-processing stage, the aircraft and crew schedules, as well as the disruption(s), are loaded and processed. The information required by the aircraft recovery stage can be divided into four main parts: (1) fleet information, (2) schedule information, (3) disruption information, and (4) cost information. The details of the exact information provided by each of these parts is present in Table 3.1.

Table 3.1: Aircraft Recovery Data.

Fleet Data	Aircraft Schedule Data	Aircraft Cost Data	Disruption Data
Aircraft type	Flight schedule	Direct operating cost	Disruption type
Tail number	Inter-airport distances	Delay cost per minute	Disruption duration
Turn-around-time	Passenger itineraries	Cancellation cost	Time found out
Aircraft range	Minimum connection times		Affected flights, aircraft, airports
Passenger capacity			

The delay and cancellation costs account for both soft and hard costs, determined based on the work of Cook et al. (2012), as well as connecting passenger itineraries. The connecting passenger costs are calculated via the use of a connecting passenger matrix as implemented by Vink et al. (2020). The matrix evaluates the impact of a delay on the connecting passenger itineraries, and assigns a corresponding soft and hard cost per passenger. Delay per passenger is measured as the delay at the end-destination. The evaluation of connecting passenger delay is best explained via the use of Figure 3, where an itinerary from airport A to airport C connects via airport B. If Flight 1 is delayed by 10 minutes, there is still enough time for the connection to happen. If, however, the flight is delayed by 20 minutes, the connection is broken and the passenger must connect at the next flight, or have the outbound connecting flight delayed by 10 minutes. These decisions result in end-destination delays of 10 and 80 minutes respectively for this itinerary. The delay costs for a particular flight therefore depend on all passenger itineraries containing that flight.

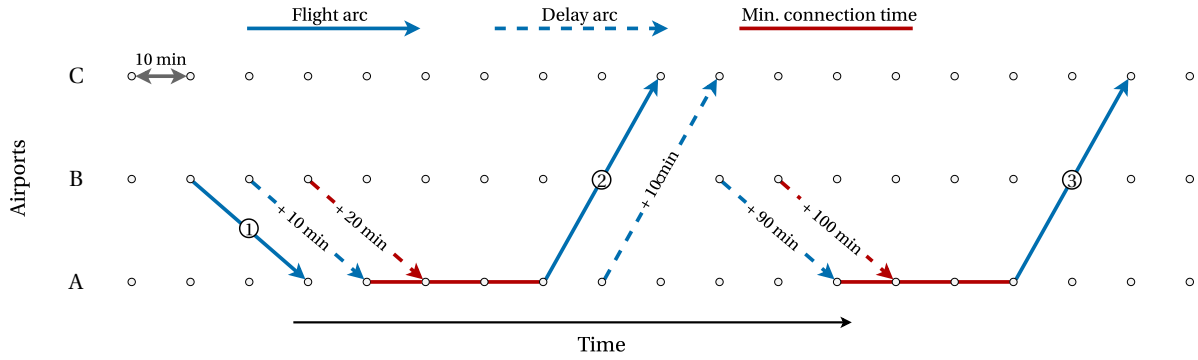


Figure 3: Calculation of connecting passenger end-destination delay.

The possibility of crew missing a connection due to an aircraft recovery decision is also considered within the connecting passenger matrix. Similarly to connecting passengers, if a delay or cancellation decision causes a crew to miss their flight, additional cost is added within the objective function. This ensures crew considerations are present within the aircraft recovery stage, despite the separate nature of the two stages.

The pre-processing phase completes with the generation of features using the fleet, aircraft schedule, and disruption data which are passed on for classification. The random forest classifier outputs a per-aircraft probability of use in the optimal solution, which is passed to the disruption solver along with the data loaded during pre-processing. Here, a subset of aircraft likely to be used in the optimal solution is selected and used to create a time-space sub-network based on which the aircraft recovery problem is written. The recovery of aircraft considers tail swaps, flight delays, and flight cancellations as possible recovery options.

Aircraft Recovery Formulation

This chapter describes the BIP formulation of the aircraft recovery problem. The formulation relies on the concept of parallel time-space networks to implement constraints at an aircraft level. The following lists describe the sets, decision variables, non-decision variables, and mathematical model used by the aircraft recovery. Note that all decision variables are binary, and their description includes only the condition at which they take a nonzero value.

Sets:

- **F** - Set of flights i
- **A** - Set of airports a
- **E** - Set of aircraft types e
- **P** - Set of aircraft p
- **P** - Set of aircraft p
- **N** - Set of all nodes n
- $\mathbf{N_o}$ - Set of origin nodes n
- $\mathbf{N_i}$ - Set of intermediate nodes n
- $\mathbf{N_s}$ - Set of sink nodes n
- **T** - Set of delay steps t
- **S** - Set of slack variables j

Decision Variables

- $\delta_{F_{p,i}}$ - if p allocated to i
- $\delta_{FD_{p,i,t}}$ - if p allocated to i with delay t
- δ_{C_i} - if i cancelled
- $\delta_{GF_{p,n}}$ - if p uses n-originating ground arc
- $\delta_{F'_i}$ - if i flown by unscheduled AC
- s_j - slack if infeasible

Variables:

- β_{bus} - Cost multiplier for business passengers
- C_{canx} - Cancellation hard cost per pax
- $C_{conn_{i,t}}$ - Connecting passenger delay cost on i, for t
- $C_{D_{i,t}}$ - Delay cost for i, for t
- C_{DS_t} - Soft cost for delay time step t
- C_{DH_t} - Hard cost 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 ground arc originating from n
- $C_{C_{SCH}}$ - Unscheduled AC operating penalty
- C_{DOC_p} - Operating cost of p, per block hour
- h_n^e - Number of AC of type e required at node n
- $PaxY_i$ - Number of economy passengers on i
- $PaxJ_i$ - Number of business passengers on i
- $SeatsY_i$ - Number of economy seats on i
- $SeatsJ_i$ - Number of business seats on i

Formulation

The ultimate goal of the aircraft recovery process is to restore the disrupted schedule back to its original state at the minimum cost to the airline. The entirety of the financial considerations included in the aircraft recovery formulation are represented by the following objective function:

$$\begin{aligned} Min \sum_{p\in P}\sum_{i\in F} C_{OP_{p,i}}\cdot\delta_{F_{p,i}} + \sum_{p\in P}\sum_{i\in F}\sum_{t\in T}(C_{OP_{p,i}}+C_{D_{i,t}})\cdot\delta_{FD_{p,i,t}} + \sum_{i\in F} C_{C_i}\cdot\delta_{C_i} + \sum_{p\in P}\sum_{n\in N} C_{G_n}\cdot\delta_{G_{p,n}} \\ + \sum_{i\in F} C_{C_{SCH}}\cdot\delta_{F'_i} + \sum_{j\in S} s_j\cdot M \end{aligned} \tag{1}$$

The first line in the above function represents the sum of all *aircraft-flight* related costs: operating a scheduled flight on-time or with a delay, utilizing ground arcs, and cancelling a scheduled flight. The second line refers to the necessary *slack variables* to ensure feasibility and prevent unwanted behaviour from the model such as changing an aircraft routing or missing an aircraft type at the end of the time window. The objective function is subject to the following set of constraints:

$$\delta_{C_i} + \sum_{p\in P}\left(\delta_{F_{p,i}} + \sum_{t\in T}\delta_{FD_{p,i,t}}\right) = 1 \quad \forall i \in F \tag{2}$$

$$\left(\delta_{G_{p,n-1}} + \sum_{i \in F_{in}} \delta_{F_{p,i}} + \sum_{i \in F_{in}, t \in T} \delta_{FD_{p,i,t}}\right) - \left(\delta_{G_{p,n}} + \sum_{i \in F_{out}} \delta_{F_{p,i}} + \sum_{i \in F_{out}, t \in T} \delta_{FD_{p,i,t}}\right) = 0 \quad \forall p \in P, n \in N_i \tag{3}$$

$$\delta_{GF_{p,n}} + \sum_{i \in F_{out}} \delta_{F_{p,i}} + \sum_{i \in F_{out}, t \in T} \delta_{FD_{p,i,t}} = 1 \quad \forall p \in P, n = \text{scheduled } N_o \text{ of } p \tag{4}$$

$$\delta_{GF_{p,n-1}} + \sum_{i \in F_{in}} \delta_{F_{p,i}} + \sum_{i \in F_{in}, t \in T} \delta_{FD_{p,i,t}} + s_j \geq 1 \quad \forall p \in P, n = \text{scheduled } N_s \text{ of } p \tag{5}$$

$$\sum_{p \in P(e)} \left(\delta_{GF_{p,n-1}} + \sum_{i \in F_{in}} \delta_{F_{p,i}} + \sum_{i \in F_{in}, t \in T} \delta_{FD_{p,i,t}}\right) + s_j \geq h_n^e \quad \forall e \in E, n \in N_s \tag{6}$$

$$\delta_{F_{p,i}} + \sum_{t \in T} \delta_{D_{p,i,t}} - \delta_{F'_i} = 1 \quad \forall i \in F, p = \text{ aircraft scheduled for } i \tag{7}$$

$$\delta_{F_{p,i}} + \sum_{t \in T} \delta_{D_{p,i,t}} = 0 \quad \forall p, i \text{ where (seats } Y_p < \text{Pax} Y_i \wedge \text{seats} J_p < \text{Pax} J_i) \tag{8}$$

$$\delta_{F_{p,i}} + \sum_{t \in T} \delta_{D_{p,i,t}} = 0 \quad \forall p, i \text{ where } (\text{range}_p < \text{dist}_i) \tag{9}$$

Equation 2 ensures that all flights are either flown or time, flown with a delay, or cancelled. The movement of aircraft between nodes is covered by the intermediate node-balance constraint in Equation 3, ensuring net flow between all non-origin and non-sink nodes is equal to zero. The constraint in Equation 4 constrains the net flow out of the origin node to one, while Equation 5 ensures that the flow into the sink node is equal to one. Note that this sink node constraint is only applied in case of scheduled maintenance. Since no maintenance tasks are considered in the case study, the secondary sink node constraint present in Equation 6 ensures that the recovery attempts to end the time window with the required number of aircraft of each type at each airport and was used for the case study. For each flight not operated by the originally scheduled aircraft, a penalty is incurred as shown in Equation 7. Equation 8 ensures that flights cannot be flown by aircraft with less than the booked seat capacity, while Equation 9 ensures that flights can only be flown by aircraft that satisfy the range requirement between the origin and destination airports.

3.2. Crew Recovery

Upon completion of aircraft recovery, the solution is passed on to the crew recovery stage, where the crew schedules are repaired. This stage starts with the recovered schedule of the aircraft recovery. In a similar four-fold data split, the crew recovery stage requires: (1) crew information, (2) crew schedule information, (3) crew cost information, and (4) disruption information. The latter in particular is different from that of the aircraft recovery, as the disruptions are evaluated in the crew recovery stage are based on the changes made during aircraft recovery. If, given a single flight delay as input, the solution of the aircraft recovery stage is to delay n flights, the disruptions in the crew recovery stage are the n delays used in the aircraft recovery solution. The crew recovery therefore often resolves more simultaneous disruptions than the aircraft recovery does. The exact data present in each of the four information blocks the crew recovery is present in Table 3.2

Table 3.2: Crew Recovery Data.

Crew Data	Crew Schedule Data	Crew Cost Data	Disruption Data
Crew aircraft type	Recovered flight schedule	Crew direct operating cost	Changed flight arcs
Crew number	Assigned flights	Crew penalty costs	Change type
Flight hours remaining	Scheduled sink airport		Crew affected
Duty hours remaining	Duty sink airport		Flights affected
Reserve crew data			

Upon receiving the recovered aircraft and flight schedule, the changes are processed and features are generated per crew-pair. These are passed on to the XGB classifier, which outputs a per-crew probability of being used in the optimal solution. The disruption solver uses these probabilities to create a crew time-space subnetwork, and each crew's origin and sink nodes are determined based on regulations. Based on this, the recovery problem is written solved, and the updated schedule is presented to the user.

As the aircraft recovery stage considers flight delays as a recovery option, the addition of flight delay opportunities within the crew recovery stage could lead to unanticipated changes in the aircraft schedule, requiring a re-iteration of the aircraft recovery. This is a limitation of the sequential approach, as no aircraft considerations are present in the crew stage. As such, the consideration of delays is only present within the aircraft recovery stage. The crew recovery stage considers crew swaps, crew deadheading, the use of reserve crew, and flight cancellations as recovery options. Deadheading crew refers to transporting crew as passengers on a flight so that they may take over a missed flight or reach their end-of-duty destination, adding additional recovery options to the recovery solution. If a crew swap results in a crew being unable to reach its end-day airport, the model may choose to deadhead the crew there or use reserve crew instead of swapping scheduled crew. When no crew can be assigned to a flight at all, the flight is cancelled.

Crew Recovery Formulation

In a similar manner to that of aircraft recovery, the BIP formulation of the crew recovery problem utilises parallel time-space networks to enable constraints at a crew level. The sets, decision, and non-decision variables used are similar in formulation to those of the aircraft recovery problem and are present in the following lists. Note that all decision variables are binary, and their description includes only the condition at which they take a nonzero value.

Sets:

- **F** - Set of flights i
- **A** - Set of airports a
- **N** - Set of all nodes n
- $\mathbf{N_o}$ - Set of origin nodes n
- $\mathbf{N_i}$ - Set of intermediate nodes n
- $\mathbf{N_s}$ - Set of sink nodes n
- **K** - Set of crews k

Decision Variables:

- $\delta_{K_{k,i}}$ - if k allocated to i
- $\delta_{G_{k,n}}$ - if k uses n-originating ground arc
- $\delta_{DH_{k,i}}$ - if k deadheaded on i
- $\delta_{K'_i}$ - if i flown by unscheduled crew
- δ_{CX_i} - if i is cancelled
- s_k - slack if sink constraint violated
- s_{FT_k} - slack if scheduled flight time is exceeded

Parameters:

- C_{G_n} - Cost of ground arc originating from n.
- $C_{K_{k,i}}$ - Operating cost of crew k on flight i
- $C_{DH_{k,i}}$ - Deadhead cost of crew k on flight i
- C_{OC} - Penalty if flight flown by non-scheduled crew
- C_{SV} - Penalty if sink node constraint violated
- C_{FT} - Penalty if scheduled flight time exceeded
- C_{CX_i} - Cancellation cost of flight i
- FT_i - Flight time of flight i
- FTL_k - Flight time remaining within TW for crew k
- FTM_k - Maximum additional flight time for crew k

Formulation:

The aim of the crew recovery problem is to minimise the costs of disruptions incurred by the crew schedule. These are in the form of crew operating costs, crew deadhead costs, and flight cancellation costs. These are present in the following objective function:

$$Min \sum_{k\in K}\sum_{i\in F}\left(C_{K_{k,i}}\cdot\delta_{K_{k,i}} + C_{DH_{k,i}}\cdot\delta_{DH_{k,i}}\right) + \sum_{k\in K}\sum_{n\in N} C_{G_n}\cdot\delta_{GK_{k,n}} + \sum_{i\in F} C_{CX}\cdot\delta_{CX_i} + \sum_{i\in F} C_{OC}\cdot\delta_{K'_i} + \sum_{k\in K} C_{SV}\cdot s_k + \sum_{k\in K} C_{FT}\cdot s_{FT_k} \tag{10}$$

The final three elements of the objective function refer to the slack and penalty variables necessary to prevent unwanted behaviour and ensure problem feasibility. These are the original crew penalty, the sink-node penalty, and the scheduled flight time penalty. The original crew penalty is applied whenever a flight is flown by a different crew than scheduled. The sink-node penalty is applied whenever a crew does is not at their scheduled end-location at the end of the time window. The scheduled flight time penalty is applied whenever a crew exceeds their total scheduled flight time due to recovery actions. The way these function within the minimization problem is governed by the following set of constraints:

$$\delta_{CX_i} + \sum_{k\in K_e}\delta_{K_{k,i}} = 1 \quad \forall i \in F, \forall e \in E \tag{11}$$

$$\left(\delta_{GK_{k,n-1}} + \sum_{i\in F_{in}}(\delta_{K_{k,i}} + \delta_{DH_{k,i}})\right) - \left(\delta_{GK_{k,n}} + \sum_{i\in F_{out}}(\delta_{K_{k,i}} + \delta_{DH_{k,i}})\right) = 0 \quad \forall k \in K, n \in N_i \tag{12}$$

$$\delta_{GK_{k,n}} + \sum_{i\in F_{out}}(\delta_{K_{k,i}} + \delta_{DH_{k,i}}) = 1 \quad \forall k \in K, n = \text{scheduled } N_o \text{ of } k \tag{13}$$

$$\delta_{GK_{k,n-1}} + \sum_{i\in F_{IN_n}}(\delta_{K_{k,i}} + \delta_{DH_{k,i}}) + s_k = 1 \quad \forall k \in K, n = \text{scheduled } N_s \text{ of } k \tag{14}$$

$$\delta_{K_{k,i}} - \delta_{K'_i} = 1 \quad \forall i \in F, k = \text{ crew scheduled for } i \tag{15}$$

$$\sum_{i\in F}\delta_{K_{k,i}}\cdot FT_i \le FTL_k + FTM_k\cdot s_{FT_k} \quad \forall k \in K \tag{16}$$

Equation 11 ensures that all flights are either assigned a crew or cancelled. The node-balance between intermediate nodes is governed by equations Equation 12. The origin node net out-flow is ensured by Equation 13, while the sink node net in-flow is governed by Equation 14. The sink node constraint is applied for every crew-pair due to labour constraints. For each flight operated by a crew different than originally scheduled, the objective function incurs a penalty as shown in Equation 15. Finally, the total flight time of a crew within

the time window is kept within limits by Equation 16. Within the time window, crew labour and end-of-time-window location constraints are implemented with a parallel time-space network approach by identifying origin and sink nodes during preprocessing. The violation of these constraints is assigned an appropriate penalty in the objective function.

4. Data set

Five separate datasets were required to implement the SDSS. The flight and passenger schedules, as well as disruption data used are identical to those processed by Hassan (2018). This flight schedule data was obtained from the United States Bureau of Transportation Statistics online public database. The 2015 U.S. flight schedules of Delta Airlines were isolated and used for the entirety of model development and testing. The flight schedules were supplemented with passenger itineraries with data from Barnhart et al. (2014). Delta Airlines was chosen due to its large fleet size and hub-and-spoke operating model. Delta operates 26 different aircraft types on an average of 2400 daily domestic flights and 150 domestic destinations. The flight schedule contains per-flight information on the scheduled time of departure (STD) and scheduled time of arrival (STA), the tail number assigned to the flight, origin and destination airport, and flight number. The passenger data contains the number of economy and business passengers booked on the flight, as well as the flight load factor. Disruption data consists of a disruption type, disruption duration, disruption cause, a time at which the disruption is found out (TFO), and affected flights, aircraft, and airports. To enable the integration of crew into the SDSS, initial and reserve crew schedules are required for the crew recovery stage.

Due to the confidentiality of crew schedules, an initial crew schedule was generated using the acquired flight data. A series of successive flights operated by a crew within a single work day is referred to as a crew duty (Barnhart et al. (2003)). A series of successive duties starting and ending at the crew base is referred to as a crew pairing. The crew schedule generation consisted of first combining individual flights into duties, and then combining individual duties into pairings. The scheduling of cockpit crew is subject to stringent regulations due to possibilities of fatigue deteriorating crew performance capability. The generation of duties and pairings was done with the baseline regulations set out by the Federal Aviation Administration (2011), as well as those listed by Delta Airlines (2020) in the publicly available crew scheduling handbook. These contain the common cockpit crew regulations respected by Delta Airlines in crew scheduling and recovery within the domestic U.S. flight network. The main regulations considered for the generation of crew duties and pairings are present in the following list. Note that exact restrictions for the first two are dependent on the number of flights within a duty and the duty start time.

- The maximum scheduled length of a duty period ranges from 9 to 14 hours.
- The maximum scheduled flight time within a duty period ranges from 8 to 9 hours.
- Crew may not be scheduled to more than 60 duty hours in any consecutive 168-hour (7-day) period.
- Crew may not be scheduled to more than 190 duty hours in any consecutive 672-hour (28-day) period.
- All crew must have a consecutive 10-hour rest between any two duties.
- All crew must have a consecutive 30-hour rest period within any consecutive 168-hour (7-day) period.

A greedy forward-heuristic approach was used to generate crew duties and pairings. When iterating through all flights, the algorithm takes the list of already constructed duties and evaluates whether assigning a flight to a duty is possible with respect to crew work time regulations. Upon finding a compatible duty, the algorithm assigns a flight to it. The duties are sorted in order of ground time at the flight origin airport prior to departure, meaning that a crew that has arrived at the flight origin airport two hours prior to departure will always have

priority over one that arrived 30 minutes prior to departure. If no duty is found, one is created and assigned the flight. This approach, ensures that tight connections are avoided as much as possible when multiple crew are available for a certain flight. A schematic representation of the duty generation algorithm is present in Figure 4.

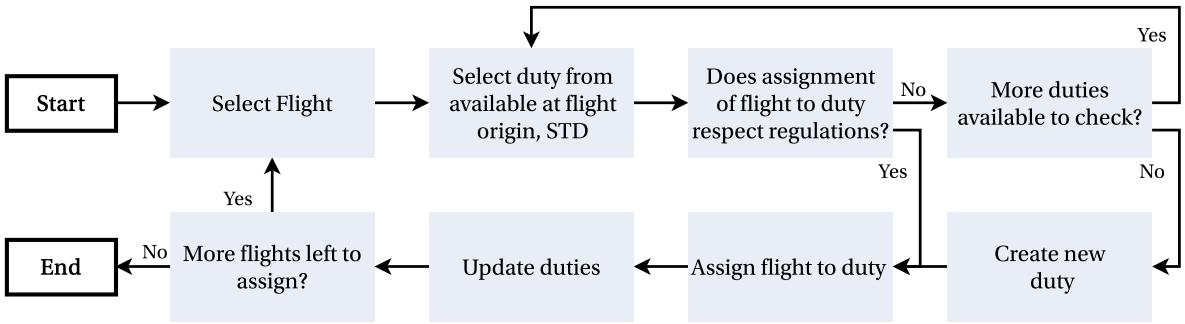


Figure 4: Flowchart of the heuristic crew duty generation algorithm.

Upon completion, a pairing generation algorithm assigns the generated duties to pairings in a similar way. Each duty is evaluated in terms of regulation compatibility with available pairings. If no compatible pairing is found, one is generated and the duty is assigned to it. Together, these two algorithms generate the complete crew schedule. Although the schedule is heuristically generated, it provides a necessary baseline for use within the crew recovery problem. The generated schedule is added to the flight and passenger schedule, supplementing each flight with an assigned crew, as well as that crew's end-of-duty airport, remaining scheduled duty and flight times, and maximum allowable duty and flight times.

The reserve crew schedule, generated on the method proposed by based on the work of Bayliss et al. (2012), is loaded separately and contains information on the reserve crew base, duty start and end times, and aircraft type operated. If the probability of scheduled crew being unavailable for a flight is known for the entire schedule, a fixed number of reserve crew can be assigned in a way that minimises the total probability of crew unavailability over the entire schedule. For this research, it is assumed that crew absence for a certain flight can only be caused by the cancellation of the crew's previous flight, or a delay of the previous flight to the point where the flight considered is missed. Knowing the probabilities of these two independent events allows for assigning each flight a probability of missing scheduled crew. A random forest classifier was trained with Delta Airlines on-time performance (OTP) data for the period 2012-2014 to obtain cancellation and delay probabilities per flight. This classifier is used exclusively for the generation of disruption probabilities and is not called as part of the SDSS.

5. Machine Learning Classifier

Both the aircraft and crew recovery problems can consider several hundreds of their respective resource when attempting to restore a schedule. In most cases, though, the schedule of only a small subset of aircraft or crew is changed. Many papers utilise heuristic selection algorithms in an attempt to select the aircraft or crew that are likely to be used in the optimal solution, such as those of Hoeben et al. (2017) and Vink et al. (2020). In this way, the problem can be formulated using only a small subset of resources, decreasing the solution time. The use of a machine learning-based selection algorithm, however, can consider many more features of each aircraft and crew when evaluating their utility with respect to the final solution. Two machine learning classifiers are used to reduce the solution space in each of the SDSS's two stages. For the aircraft

recovery, the machine learning classifier is identical to that used by Hassan (2018). Namely, a random forest (RF) classifier is used to decide which subset of aircraft are likely to be used in the recovery solution. For the crew recovery stage, another decision tree based algorithm has been developed that selects a subset of crew likely to be used in the crew recovery solution. Both the aircraft and crew selection classifiers were trained using disruption solution data of Delta's January 2015 schedule. This chapter will mainly focus on the description of the crew selection machine learning classifier. The development of the crew selection classifier consisted of identifying an appropriate classification algorithm, identifying appropriate evaluation methods, feature engineering, and hyperparameter optimization. The development process is summarized in Figure 5.

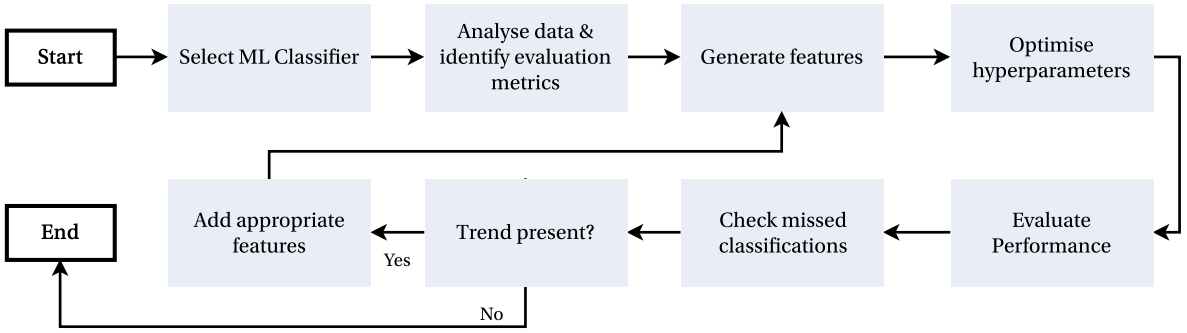


Figure 5: Flowchart of the classifier development process.

Given a set of crews, the target output of the classifier would be a selection of those likely to be used in the optimal solution, i.e. those classified by the algorithm as 'True'. The selection of a good binary classification algorithm is therefore key in obtaining good results. Literature utilising machine learning in a similar way, such as the works of Kruber et al. (2017), Bonami et al. (2018), and Hassan (2018), consistently rated the performance of random forest and support vector machine-based classifiers as best. The validity of these conclusions is strengthened by the work of Olson et al. (2017), where the performance of 13 commonly-used classifiers is evaluated and compared on a set of 165 classification problems. These have been evaluated based on 10-fold cross-validation balanced accuracy. Though the results indicate that support vector machines (SVM) and random forests (RF) generalise particularly well across a variety of classification problems, the use of gradient tree boosting consistently outperforms that of either. For this research, the gradient tree boosting approach has been implemented via the use of the extreme gradient boosting (XGB) library as part of the scikit-learn Python library. The following sections expand on the generation and engineering of features, model training and evaluation, and hyperparameter optimization performed as part of the classifier development.

5.1. Feature Engineering

As the classifier is able to learn exclusively from the data it is provided, this data must be processed in a way that makes the classifier most likely to learn the rules behind the classification. The quality of the classifier therefore depends heavily on the features it is given. Feature engineering was an iterative process that relied upon generation of features, model training and evaluation, and identification of properties of crew that were identified as false negatives, as previously shown in Figure 5. A total of 126 features are used for classification in the final model. The following list contains the description of 14 of the main features used in the classifier. The remaining 112 features are derived from the 14 main features and were omitted for brevity.

Table 5.1: Features used for training the XGB classifier.

Feature	Variable Type	Description
c_at_crit*	Boolean	True if crew is at critical location at critical time
c_at_d_origin*	Boolean	True if candidate crew is at origin of disrupted flight
c_at_d_destination*	Boolean	True if candidate crew is at destination of disrupted flight
critical_time*	Integer	Critical time as measured in minutes after the start of the TW.
d_type_canx*	Boolean	True if disruption is a cancelled flight
d_duration*	Integer	Length of disruption in minutes (0 if cancellation)
future_fl_to_next*	Boolean	True if crew has future flight to destination of disrupted flight
future_fl_to_end*	Boolean	True if crew has future flight to end airport of disrupted crew
ac_family	Integer	Aircraft family, label (integer) encoded.
reserve_crew	Boolean	True if candidate crew is reserve crew.
c_time_start	Integer	Time of candidate duty start within the TW in minutes from TW start time.
c_time_end	Integer	Time of candidate duty end time within the TW in minutes from TW start time .
tw_start	Integer	TW start time in minutes from midnight.
tw_end	Integer	TW end time in minutes as sum of TW start time and TW length.

In the above table, an important feature is `c_at_crit`, representing whether a candidate is present at the critical location at the critical time. These represent the time and location where a flight's originally scheduled crew will not be present to operate it due to a broken schedule, as shown in Figure 6. A crew with this schedule will be able to operate Flight 2 if Flight 1 is not delayed, or delayed by 10 minutes. In these two cases, no critical location or time is present. If Flight 1 is delayed by 20 minutes or more (or cancelled), the crew will not be able to operate Flight 2, forcing a crew reassignment. Therefore, the origin airport and scheduled departure time (STD) of Flight 2 are identified as the critical location and time respectively, as shown by the red octagon on the figure. Action must be taken to recover the schedule at this point: Flight 2 must be cancelled or assigned a different crew.

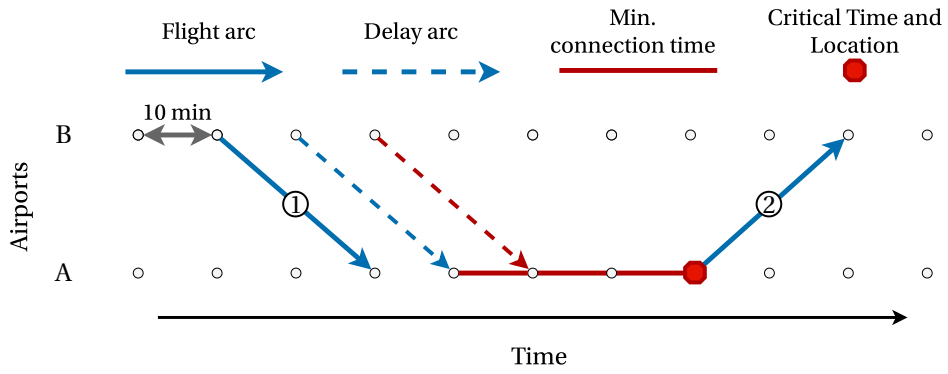


Figure 6: Flowchart of the heuristic crew duty generation algorithm.

Four additional features exists for each of the first three features in Table 5.1. These describe the location of the crew with respect to the critical location/disruption origin/disruption destination in the past and were omitted from the feature list for brevity. The subscripts `_1h`, `_2h`, `_3h`, and `_before` are added to represent the presence of a candidate crew at the specified location 1 hour, 2 hours, and 3 hours before the disruption, as well at any time in the past respectively. The feature `c_at_crit_before`, for example, describes whether the candidate is scheduled to be located at the critical location at any point prior to the critical time.

As the crew recovery may resolve multiple disruptions at once, additional features must be added to represent candidate information with respect to individual disruptions. The only information the recovery solu-

tions used in the training provide is whether a crew had their scheduled changed in the solution or not. It is therefore difficult to determine without manual examination of every recovery solution whether the use of a crew within the recovery solution was directly related to a single disruption, multiple disruptions, or was used to take over flights of non-disrupted crew that were used within the recovery solution. Therefore, for all the features marked with an asterisk (*) in the above list, including the additional 12 features that supplement the first three, a 6-fold split is made to describe properties of a candidate crew with respect to a multiple disruptions. These are supplemented with the subscript `_n`, for $n \in \{1,2,3,...6\}$ to represent candidate information with respect to the first six disruptions solved within the disruption set. More than six simultaneous disruptions were present in only 0.2% of disruption scenarios. Despite this, when more than six disruptions are present, the sixth of the supplementary features considers the candidate information with respect to all disruptions from the sixth to the final disruption present. If seven disruptions are present in a certain scenario and the value of `c_at_crit_before_6` for the sixth is False while that for the seventh is True, the final value of the feature is True. In this way, enough detail is provided to the disruption model to learn the relationship between the disruption properties and how they affect the crew's use in the solution, while reducing the feature space to a reasonable size.

5.2. Training & Evaluation

The input to the machine learning classifier consists of the disruption data and its properties with respect to each crew, while the target output is a binary classification indicating whether the crew is used in the disruption recovery. Up to several hundred crew-pairs are considered at once, while most solutions require schedule changes for under 5 total crews to obtain the best solution. The majority class (i.e. those labelled as 'False') highly outweigh the minority class, making the dataset imbalanced. To avoid the classifier receiving insufficient information about the minority class, class imbalance must be addressed during training.

In literature, four methods are commonly used to address this issue: (1) undersampling, (2) oversampling, (3) synthetic minority oversampling techniques (SMOTE), and (4) cost-sensitive learning. Undersampling involves the selective or random removal of majority class instances, while oversampling involves selective or random replication of minority class instances. Though oversampling techniques generally perform better than undersampling (Mohammed et al. (2020)), both contain some bias and may remove instances containing important information or add instances containing redundant information. Chawla et al. (2002) present SMOTE as an alternative, which adds synthetic examples of the minority instance to the training dataset. These are generated based on instances of the minority class already present within the dataset. Cost-sensitive learning assigns a difference in 'cost' for classification, where an incorrectly classified instance of the minority class is penalised more heavily during training than that of the minority class (Ling and Sheng (2010)). For the XGB classifier used in the crew stage, the best performance was yielded by the use of exclusively cost-based learning. Class imbalance is therefore still maintained in the training dataset, but addressed during the learning process.

Model performance must also be evaluated accounting for the imbalance present in the training dataset. To illustrate the reasoning behind this, some of the most common metrics of classifier performance are listed in Table 5.2. Accuracy, for example, is unsuitable as an evaluation metric for imbalanced datasets. If the minority class makes up 0.1% of the dataset, a classification algorithm that classifies all input as that of the majority class would have an accuracy of 99.9%. This kind of classifier is not useful, as likely no knowledge of either class is present.

Table 5.2: General metrics for evaluation of classifier performance.

Metric	Formula	Description
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Correct classifications over total classifications.
Precision	$\frac{TP}{TP+FP}$	Correct positive classifications over total positive classifications.
Recall	$\frac{TP}{TP+FN}$	Correct positive classifications over total actual positive instances.
Specificity	$\frac{TN}{TN+FP}$	Correct negative classification over total actual negative instances.
Error rate	$\frac{FP+FN}{TP+TN+FP+FN}$	Incorrect classifications over total classifications.

Assuming the minority class is positive, as is the case in this problem, recall is an important metric to consider. To avoid the same problem faced with the exclusive consideration of accuracy, a balance must be made between recall and precision. This can be done with the use of a precision-recall curve. A precision-recall curve contains useful information for imbalanced datasets with a positive minority class, as the number of true negatives (TN) is not considered. Therefore, the relevant metrics are the ratio of correct positive classifications over the total number of instances of the positive class, and the ratio of correct positive classifications over the total number of positive classifications. The area under the precision-recall curve, representing the collective recall and precision performance of the classifier across a range of probability thresholds, has been taken as the objective for model evaluation.

5.3. Hyperparameter Optimization

When initialising and training the XGB classifier, several options with respect to how the algorithm will learn must be specified. These parameters, their respective functions, and final values used in the classifier are present in the following table.

Table 5.3: Hyperparameter values post-Bayesian optimisation.

Parameter	Description	Final Value
n_estimators	Total number of trees within the classifier.	329
max_depth	Maximum depth of a single tree.	44
gamma	Minimum loss reduction required to make a further partition on a leaf node of the tree.	0.791
learning_rate	Step size shrinkage used in update to prevent overfitting.	0.086

The interaction between these parameters is unpredictable and must be evaluated should we want to reach close-to-optimal performance. A time-efficient method of optimising model performance is the use of Bayesian optimisation, which has proven to arrive at near-optimal hyperparameter combinations in much fewer iterations than exhaustive methods such as grid search (Snoek et al. (2012)). A probabilistic model selects the best hyperparameter values, x^*, from a range of allowable values, X, and evaluates their performance given an objective function, $f(x)$. In equation form, this is represented by Equation 17.

$$x^* = argmin_{x \in X} f(x) \tag{17}$$

Bayesian optimisation uses previously evaluated hyperparameter combinations to construct a *surrogate* model of the objective function used to estimate the impact of certain parameters on the objective function. This is done by mapping hyperparameters to a probability of a certain score, i.e. obtaining $P(score|x)$. The surrogate function is initialised by evaluating a set of randomly selected hyperparameter combinations on the real

objective function, after which the model is able to determine the next set of hyperparameters to evaluate via the use of a selection function.

During Bayesian optimization, evaluation must be performed on several datasets to ensure model performance generalises on unseen data. Given that the dataset used in this research is limited, a good way to ensure generalisation is via the use of k-fold cross validation. Assuming k folds or equal splits of data, the classifier is trained on $k-1$ folds and evaluated on a single fold. This is repeated for each fold and the performance of the algorithm between the k folds is averaged to ensure generalisation. For this research, Bayesian hyperparameter optimization was performed with 10-fold cross validation, 100 initialisation iterations, and 1000 iterations optimising the hyperparameters. With the objective of maximising the area under the precision-recall curve, this resulted in the hyperparameter values presented in Table 5.3. The performance of the classifier on the validation data is represented in Figure 7 and Figure 8 in the form of a precision-recall (PR) curve and receiver operating characteristic (ROC) curve, the latter of which shows the a comparison of the true and false positive rates the classifier obtains for varying thresholds.

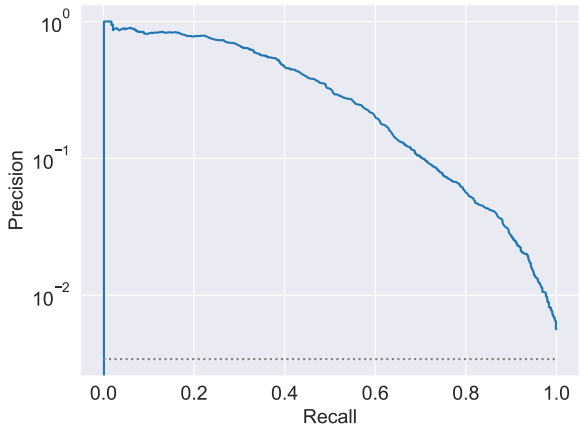


Figure 7: PR curve for the trained classifier on validation data.

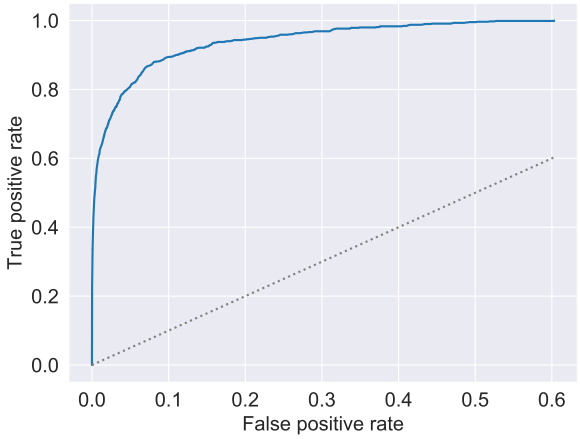


Figure 8: ROC curve for the trained classifier on validation data.

On both figures, the dashed gray line represents the performance of a classifier that randomly generates the probability of classification. The classification is positive when the randomly generated probability is less than or equal to the ratio of the number of instances of positive class to all the instances present in the dataset. As the validation dataset had 902 instances of the positive class and 263692 of the negative, this value is around 0.0034. Note that the y-axis of the PR curve is scaled logarithmically. As expected, the classifier greatly outperforms the random selection. Past a False positive rate of around 0.43 in the ROC curve, the classifier is able to correctly identify 99% of the optimal crews used in the validation. This recall percentage is associated with a precision of around 0.0078.

6. Case Study

The SDSS is tested on the domestic network of Delta Airlines, on flight, crew, and disruption data as explained in Chapter 4. The evaluation of the model is performed on Delta's schedule and disruptions during the first 7 days of February 2015, consisting of 1482 disruption scenarios. Of particular interest during the evaluation of model performance are two factors: (1) the percentage of runs that have achieved the optimal solution for both the aircraft and crew stages, and (2) the percentage of runs that complete in under 120 seconds. The solution times presented show the time required to arrive from an initial disruption to a disruption solution

and include time required for pre-processing, problem writing, and post-processing. The first part of the case study will focus on the performance of the developed crew recovery stage. The case study will conclude with the evaluation of the combined use of the aircraft and crew recovery stages. The SDSS was implemented in Python using Gurobi 9.0.3 as the LP-solver. Computational evaluation for the case study was performed on a computer with a quad-core Intel i7-6700HQ processor and 16 GB of RAM. The following sections will elaborate the parameters and assumptions used during the case study, the results of the crew recovery stage, and the combined results of the aircraft and crew stages.

6.1. Model Parameters & Evaluation

The SDSS uses several user-defined parameters that define the costs, constraints, and scope of the problem. For the aircraft recovery stage, these are present in Table 6.1, while those for the crew recovery stage are present in Table 6.2. Most of the penalty parameters have been defined in Chapter 3. Note that C_{canx} and C_{csch} in the aircraft recovery parameters refer to the fixed cost of cancellation per passenger and cost of a routing change per aircraft, respectively.

Table 6.1: Parameters used for the aircraft recovery stage.

Parameter	Value
TW Length	12 hours
Time Step	10 minutes
C_{canx}	\$250
C_{csch}	\$1000
Big-M	\$1000000
Max Delay	8 hours
Selection	Top 50%
Sink Constraint	AC Type

Table 6.2: Parameters used for the crew recovery stage.

Parameter	Value
C_{OC}	\$2000
C_{FT}	\$25000
C_{SV}	\$50000
C_{SV_R}	\$10000
C_{DH}	\$200
Selection	Dynamic
Sink Constraint	Per Crew

The recovery time window is taken as the time from the TFO of the disruption until 12 hours after the start of the last disruption. The reserve crew sink node penalty, C_{SV_R}, is only incurred when the reserve crew is used and does not return to base before the end of their reserve duty. In essence, since reserve crew is there to be used in case of no available crew, their use is not penalised if they are able to return to their starting base. The penalties applicable to the crew recovery stage have been set such that they provide a hierarchy of desired actions. For example, the cost of reserve crew violating a sink node constraint is five times greater than the cost of non-scheduled crew operating a flight, C_{OC}. The model will therefore only have reserve crew violate a sink node constraint if it negates the need for more than five crew swaps.

The aircraft and crew recovery stages use different methods for selecting their respective subsets of resources. From the RF classifier used in the aircraft recovery, the probabilities of use in the optimal solution are sorted in descending order, and the top 50% of aircraft are selected. The aircraft recovery selection therefore always cuts the number of aircraft in half. The XGB classifier used in the crew recovery stage uses a probability threshold as a baseline for selecting crews. All crews with a probability above the threshold are selected. For selections exceeding 100 crews, the selection is reduced to either the top 100 or top 50% of the crews, as seen in Equation 18. Here, s represents the number of crews in the final selection, while n represents the number of crews considered prior to selection.

$$s = max(\frac{n}{2}, 100) \tag{18}$$

For example, a selection of 120 crews would be reduced to 100, while a selection of 240 crews would be re-

duced to 120. In this way, more room for error is given to the classifier for selections of under 100 crews, while the larger selections are given a reasonable reduction. For all selections containing 100 crews or less, the crew recovery solution time was consistently under 20 seconds.

The aircraft and crew stages also apply two different sink node constraint types. Because no maintenance constraints are considered as part of the case study, aircraft are only constrained by the number of each required per airport at the end of the time window. Each crew, on the other hand, has its own end-of-duty time and location. The KPIs used in the evaluation of both the aircraft and crew stages are present in the following table. Some KPIs are applicable to the crew stage only.

Table 6.3: Key performance indicators used in the evaluation of SDSS solutions.

Abberviation	Name	Unit	Description
Sol.T.	Solution Time	[s]	Solution time required to solve disruption
Crew	Crews Considered	[#]	Number of crews considered in the recovery model
Disr.C.	Disruption Cost	[$]	Cost of disruption solution, excluding scheduled costs.
Opt.S.	Optimal Solutions	[#]	Number of runs that obtained optimal solutions.
Canx	Cancellations	[#]	Number of flight cancellations.
SNV	Sink Violations	[#]	Number of sink node constraint violations.
DH	Deadheads	[#]	Number of crew deadheads.
Swap	Crew Swaps	[#]	Number of crew swaps.
RCU	Reserve Crew Used	[#]	Number of reserve crews used.
U60	Runs Under 60 s	[#]	Number of runs that achieved a solution in under 60 seconds.
U120	Runs Under 120 s	[#]	Number of runs that achieved a solution in under 120 seconds.

6.2. Crew Recovery

The disruptions in the crew recovery stage consist of the recovery actions taken by the aircraft stage. The crew recovery stage therefore always deals with a number of disruptions greater than or equal to that of aircraft recovery. These are solved via the use of all scheduled and reserve crew, and again via the use of the classifier-based sub-network generation. The computational time and results of these two will be referred to as the optimizer and SDSS results respectively. The number of simultaneous disruptions solved by the crew recovery stage and their respective type(s) are present in Figure 9. Note that the bar labelled 'cancellations' also includes runs where both a cancellation and any number of delays are present, while the bars representing the number of delays include no cancellations.

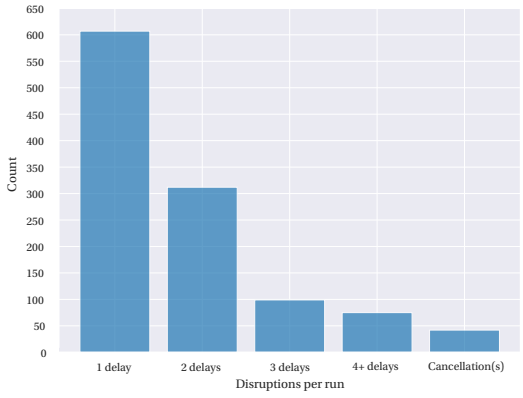


Figure 9: Number of simultaneous disruptions experienced by the crew recovery stage and their types.

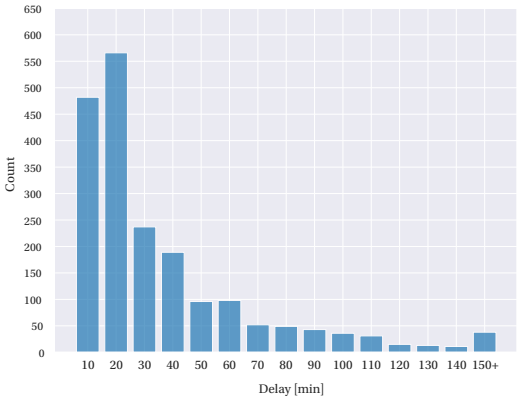


Figure 10: Distribution of delays within the disruptions experienced by the crew stage.

The optimal solutions to the disruption set consisted of 945 trivial runs and 537 non-trivial runs. Trivial runs for the crew recovery stage are defined as those that do not result in any recovery action or penalty, i.e. those that have a recovery cost of zero. If a flight cancellation, for example, causes a crew to miss its sink node, the run is not considered trivial as the solver must attempt to swap the crew's schedule or deadhead the crew back to base. The trivial runs were therefore only those where a set of disruptions can be absorbed by a given crew schedule with no changes. These are evaluated before the optimizer starts. If no crew duty is broken, only the disrupted crews are selected and forwarded to the solver. In this way, the trivial runs can consistently be solved in under 2 seconds. For the non-trivial runs, the performance of the SDSS crew recovery stage is summarized in the following table. The solution time, number of crews considered, and disruption cost were averaged for the non-trivial disruption solutions. The remainder of the KPIs presented are totals over the non-trivial disruption solutions. The cancellation numbers presented in the table refer only to cancellations made as a recovery decision in the crew stage, and not those that came from the aircraft recovery.

Table 6.4: Summary of non-trivial crew recovery results compared to optimizer.

	Sol.T.	Crew	Disr.C.	Opt.S.	Canx	SNV	DH	Swap	RCU	U60
Optimizer	65.6	261.5	65860	537	24	601	584	1599	230	269
SDSS	26.7	138.7	68958	479	25	640	493	1540	233	520

The use of the SDSS cut the solution time of crew recovery by around 60%, while the quality of the solution did not decrease significantly. The average objective function value increased by only 4.7%. Though the number of cancellations, sink node violations, and reserves used increased, the SDSS solutions have a decrease in the number of crew swaps. This is likely a result of considering less crews in the sub-optimal disruption scenarios. A selection that is missing crew that is used in a swap can lead to a sink node violation or use of additional reserve crew. The SDSS manages to find the optimal solution for 89.2% of non-trivial cases. With respect to the solution time, the violin plot present in Figure 12 shows a large overall reduction. The bulk of runs complete in under 60 seconds. How each run performs with and without the use of the ML classifier is shown in Figure 11. Here, the red points indicate a sub-optimal solution was obtained as a result of the classifier use, and the dotted line presents a 1:1 ratio of optimizer to SDSS time. Ideally, all points should be to the right of the dotted line. With the exception of a few outliers under 25 seconds, the solution time is greatly reduced. These outliers are caused by an insufficient reduction of the solution space of the ML classifier, as any selection containing 100 or less crews will not be further reduced.

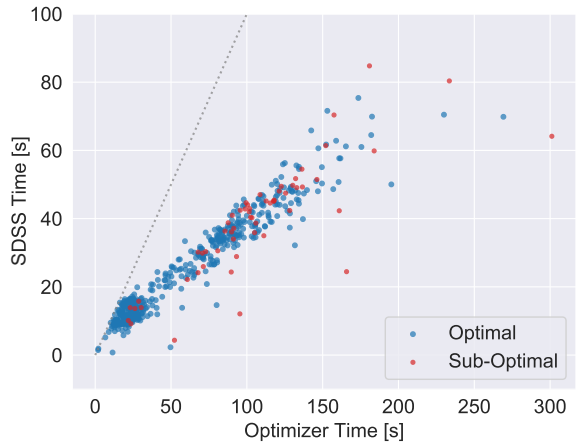


Figure 11: Scatter plot comparing solution times between specific runs of the optimizer and SDSS.

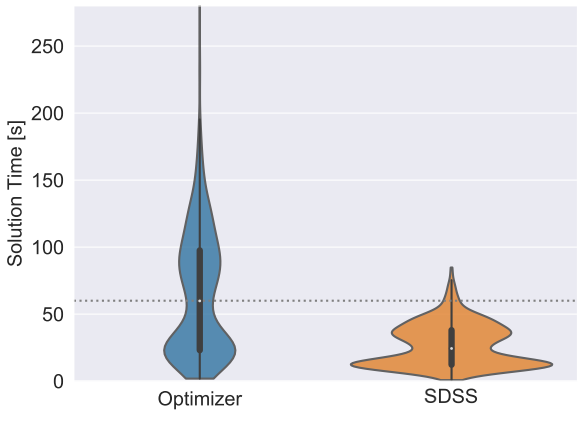


Figure 12: Solution time distribution for Optimizer and SDSS.

6.3. Aircraft and Crew Recovery

Though the crew recovery results are promising, the overall performance of the SDSS depends on the combined solution time and quality of the aircraft and crew stages. These will be evaluated in this section for all 1482 disruption scenarios for Week 1 of February 2015, as well as separately for the non-trivial crew sets. The solution times of each stage, their respective number of optimal solutions, and solution costs will be compared. A non-optimal SDSS solution can originate from either the aircraft or crew recovery stage. If the aircraft recovery classifier selects a sub-network that does not contain the optimum, the sub-optimality is considered to originate from the aircraft recovery. If, however, the aircraft recovery classifier does select a sub-network containing the optimal solution, and the crew recovery classifier does not, the sub-optimality originates from the crew recovery. The following table summarizes the performance of the aircraft and crew SDSS stages for the 537 non-trivial crew runs.

Table 6.5: Summary of full SDSS results compared to optimizer for non-trivial crew runs.

	Sol.T.	Disr.C.	Opt.S.	Canx	U120
Optimizer	166.7	123141	537	132	251
SDSS	56.0	136586	445	145	508

On average, the SDSS provides solution times three times faster than that of the optimizer. Solution quality performs worse when comparing the overall solution to the crew-only solution, with 17.1% of cases being non-optimal. Of these 92 cases, 56 were caused by the crew solution being sub-optimal, 34 were caused by the aircraft solution being sub-optimal, and 2 were caused by both stages acquiring a sub-optimal solution. The average disruption cost increased by only 10.9% for all runs. More importantly, the percentage of disruption sets solved in under 120 seconds increased from 46.7% to 94.6%, indicating that the SDSS is likely to conform to AOCC solution time requirements in a large majority of cases. For the 29 full SDSS runs that exceed the 120 second time limit, the solution time is dominated by the aircraft recovery in 25 cases, and in 4 cases by the crew recovery. The relationship between the optimizer and full SDSS solution times of the non-trivial crew runs is present in Figure 13, while the overall distribution of solution times is present in Figure 14. Note that in the latter, the y-axis is cut off at 500, as the longest run is solved in over 1000 seconds.

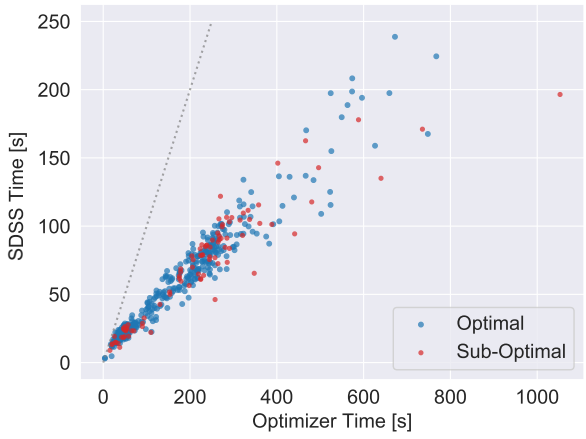


Figure 13: Scatter plot comparing solution times between specific runs of the optimizer and SDSS.

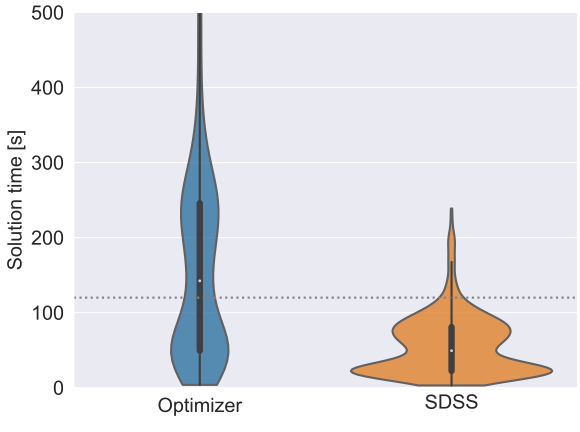


Figure 14: Solution time distribution for Optimizer and SDSS.

To gain an understanding of the overall performance of the SDSS, the following table summarizes the results compared to the optimizer for all disruption scenarios. This includes trivial runs of both the aircraft and crew

stage. Despite some scenarios having trivial solutions, an airline recovery solution would be called to resolve all disruptions. Therefore, the comparison is still relevant. As the crew recovery stage always obtains the optimal solution for its trivial runs, any increase in number of non-optimal solutions and flight cancellations originates exclusively from the aircraft recovery stage. The following table summarizes the main KPIs.

Table 6.6: Summary of full SDSS results compared to optimizer for all runs.

	Sol.T.	Disr.C.	Opt.S.	Canx	U120
Optimizer	90.3	49362	1482	144	1068
SDSS	29.4	54641	1381	159	1453

When considering all 1482 scenarios, the SDSS manages to resolve 93.1% optimally. The 10 additional sub-optimal solutions include two additional cancellations and originate from the aircraft recovery. Around 98% of scenarios are solved in under 120 seconds. This means that all scenarios with a trivial crew recovery solution are solved within the AOCC time limit.

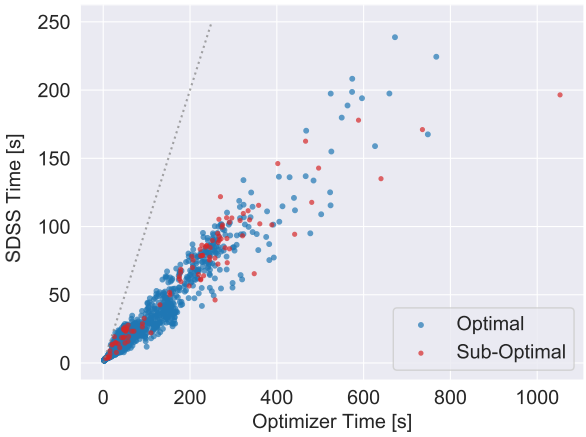


Figure 15: Scatter plot comparing solution times between specific runs of the optimizer and SDSS.

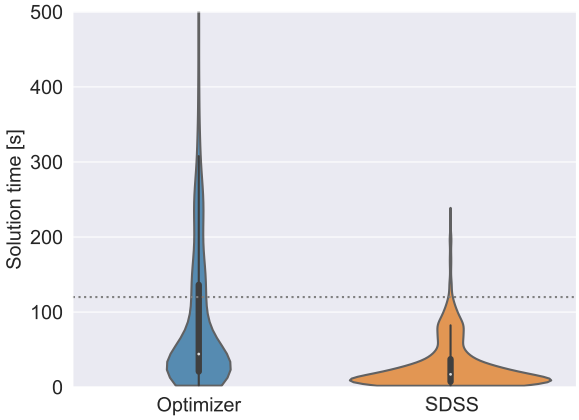


Figure 16: Solution time distribution for Optimizer and SDSS.

A summary of the overall difference between the use of the optimizer and SDSS is present in Table 6.7. Here, the main KPIs obtained via the use of the SDSS are presented as a percentage of those obtained by the optimizer. The non-trivial (NT) case results are especially relevant as the trivial runs are always solved optimally for crew recovery.

Table 6.7: Summary of SDSS results compared to optimizer.

Disruptions	Recovery	%Sol.T.	%Disr.C.	%Opt.S.	%Canx	%U60	%U120
NT	Crew	40.7	104.7	89.2	104.1	193.3	-
All	Crew	31.0	104.7	96.0	104.1	139.1	-
NT	Aircraft and Crew	33.7	110.9	82.9	109.8	-	202.4
All	Aircraft and Crew	32.5	111.3	93.1	110.4	-	136.0

6.4. Discussion

In most non-trivial crew disruption scenarios, the SDSS is able to greatly improve the performance of the optimizer crew recovery stage. Though 58 sub-optimal instances are present, they contain only a single additional cancellation. The influence of the XGB classifier on the number of cancellations within the crew

stage, therefore, is much lower than that of the RF classifier on the aircraft stage, which results in 14 additional cancellations. As a cancellation is the only decision within the crew stage that prompts a re-iteration of the aircraft recovery stage, the low number of additional cancellations indicates promising performance. The use of the SDSS enables airlines to recover the schedules of aircraft and crew quickly and near-optimally. The proposed approach is able to correctly select all crew used in the optimal solution 89% of the time. When selecting both aircraft and crew, the subset contains all those used in the optimal solution 83% of the time. Depending on the computational resources available to the AOCC, the SDSS formulation could prove to be a valuable primary recovery solution.

Despite the good performance of the SDSS selection classifiers, most mistakes made during classification are known. For the aircraft recovery selection, the RF classifier struggles to correctly identify candidates that do not share ground time with the disrupted aircraft. The XGB classifier used in the crew recovery selection also struggles to correctly identify crew members for specific cases where a large number of crew are used in the final solution. When non-disrupted crew take over flights of disrupted crew, leaving their originally scheduled flights with no crew, the classifier struggles to identify the crew used to operate the crew-less non-disrupted flight. These issues indicate that the feature space used in the development of these two classifiers could benefit from additional feature engineering.

A large number of sink-node violations is present in non-trivial SDSS crew recovery solutions. Unlike aircraft, crew members cannot freely be swapped in routing. Stranded crew will eventually have to be recovered, and though the crew recovery stage attempts to do so within the time window considered, a good solution often requires a larger time window, or a larger flight network. The AOCC would have to address the crew violating their sink node constraint, ideally with a secondary recovery operation utilising a large time window. This secondary recovery operating could use the entire flight network of all flights to increase the number of deadhead opportunities, as the AOCC decision making time limit likely does not apply to this case.

The number of sink node violations also points to another issue: An aircraft recovery model that does not fully consider crew in its recovery solution can make globally sub-optimal solutions. Though the aircraft stage identifies a solution as optimal, it does not consider the impact of this decision on the crew recovery stage, where flight schedules are fixed. This leaves little room for flexibility in crew recovery, and is likely the cause of the large number of sink node violations. A fully integrated recovery solution that evaluates the quality of decisions while considering both aircraft and crew constraints in a single optimization would not face the same issue. However, a parallel time-space network approach treating aircraft and crew as separate resources would increase the problem size drastically.

Due to the presence of a separate time-space network for each aircraft and crew, the solution times presented are largely comprised of problem writing time. In particular, for both the aircraft and crew stages, the writing of intermediate node-balance constraints takes up a large amount of time. Though solution times are greatly improved via the use of the two selection algorithms, restructuring the problem formulation, its implementation within the code, or the use of different programming languages or solvers could help further reduce the solution time.

7. Conclusion

The goal of this research was to develop an operational tool that enables real-time use of a software-based aircraft and crew recovery solution. This paper demonstrates that the use of machine learning to reduce the scope of the airline recovery problem can result in near-optimal solutions that require only a fraction of the

solution time. The proposed approach is a two-stage sequential recovery of aircraft and crew based on the concept of parallel time-space networks. Passenger and crew considerations are present within the aircraft recovery stage, where tail swaps, flight delays, and flight cancellations are considered as recovery options. The crew recovery stage considers crew swaps, deadheading crew, use of reserve crew, and flight cancellations as recovery options. Two machine learning classifiers trained on optimal disruption solutions are used to select subsets of aircraft and crew likely to be used in the optimal solution. By considering only a fraction of aircraft and crew, a smaller problem is solved and solution time is reduced.

For instances with non-trivial crew recovery solutions, the results of the performed case study show that the proposed approach is able to select a subset of aircraft and crew containing all those used in the final solution in a majority of cases, while increasing the overall cost by an average of 11%. More importantly, the solution time obtained using the selection is, on average, three times faster than using all aircraft and crew. The selection algorithms are able to increase the percentage of instances that obtain a solution for the aircraft and crew recovery problem within the AOCC time limit from 47% to 95%. For large airline networks, the two-minute decision making time limit means that controllers often only have time to search for a single feasible solution rather than an optimal one. Controllers can therefore benefit from the proposed approach by automating the decision-making process and being able to evaluate thousands of possible solutions when recovering aircraft and crew schedules.

Several limitations exist in the proposed approach and should be considered in future research. First, the limited consideration of passengers and crew within the aircraft recovery stage should be addressed. Ideally, this would be done by integrating aircraft, crew, and passenger recovery into a single formulation. Second, though the machine learning classifiers used by both recovery stages perform well, the selection still causes non-optimal solutions in some cases. To address this, further feature engineering is required for both the aircraft and crew stage. The evaluation of different machine learning algorithms for the purpose of selection could also prove useful. Finally, the solution times of both the aircraft and crew stages are dominated by problem writing, rather than solving. This is caused by the large number of constraints required to implement a parallel time-space network approach. Investigation into whether the proposed approach benefits from using different programming languages or solvers, as well as code restructuring could prove useful in further reducing the solution time.

Bibliography

Abdelghany, K. F., Abdelghany, A. F., and Ekollu, G. (2008). An integrated decision support tool for airlines schedule recovery during irregular operations. *European Journal of Operational Research*, 185(2):825–848.

Aktürk, M. S., Atamtürk, A., and Gürel, S. (2014). Aircraft Rescheduling with Cruise Speed Control. *Operations Research*, 62(4):829–845.

Arikan, U., Gürel, S., and Aktürk, M. S. (2017). Flight network-based approach for integrated airline recovery with cruise speed control. *Transportation Science*, 51(4):1259–1287.

Barnhart, C., Cohn, A. M., Johnson, E. L., Klabjan, D., Nemhauser, G. L., and Vance, P. H. (2003). Airline Crew Scheduling. In Hall, R. W., editor, *Handbook of Transportation Science*, pages 517–560. Springer US, Boston, MA.

Barnhart, C., Fearing, D., and Vaze, V. (2014). Modeling Passenger Travel and Delays in the National Air Transportation System. *Operations Research*, 62(3):580–601.

Bayliss, C., Maere, G. D., Atkin, J., and Paelinck, M. (2012). Probabilistic Airline Reserve Crew Scheduling Model. In Delling, D. and Liberti, L., editors, *12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 25 of *OpenAccess Series in Informatics (OASIcs)*, pages 132–143, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Bonami, P., Lodi, A., and Zarpellon Giulia (2018). Learning a Classification of Mixed-Integer Quadratic Programming Problems. In van Hoeve, W.-J., editor, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 595–604, Cham. Springer International Publishing.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority over-Sampling Technique. *J. Artif. Int. Res.*, 16(1):321–357.

Clausen, J., Larsen, A., Larsen, J., and Rezanova, N. J. (2010). Disruption management in the airline industry-Concepts, models and methods. *Computers and Operations Research*, 37(5):809–821.

Cook, A., Tanner, G., and Lawes, A. (2012). The Hidden Cost of Airline Unpunctuality. *Journal of Transport Economics and Policy*, 46(2):157–173.

Delta Airlines (2020). Delta Pilots' Scheduling Reference Handbook. Technical report.

Federal Aviation Administration (2011). Flightcrew Member Duty and Rest Requirements: RIN 2120–AJ58.

Hassan, L. K. (2018). Aircraft Disruption Management Increasing Performance with Machine Learning Predictions. Technical report.

Hassan, L. K., Santos, B. F., and Vink, J. (2020). Airline Disruption Management: A Literature Review and Practical Challenges. *Computers & Operations Research*, page 105137.

Hoeben, N., Santos, B., and Omondi, T. (2017). Dynamic Crew Pairing Recovery. In *Conference: ATRS 2017 - Air Transport Research Society World Conference*.

Kruber, M., , Lübbecke, M. E., and and Parmentier Axel (2017). Learning When to Use a Decomposition. In Salvagnin Domenico, , and Lombardi, M., editors, *Integration of AI and OR Techniques in Constraint Programming*, pages 202–210, Cham. Springer International Publishing.

Lettovsky, L. (1997). *Airline Operations Recovery: An Optimization Approach*. PhD thesis, Georgia Institute of Technology.

Lettovsky, L., Johnson, E. L., and Nemhauser, G. L. (2000). Airline crew recovery. *Transportation Science*, 34(4):337–348.

Ling, C. X. and Sheng, V. S. (2010). Cost-Sensitive Learning. In Sammut, C. and Webb, G. I., editors, *Encyclopedia of Machine Learning*, pages 231–235. Springer US, Boston, MA.

Marla, L., Vaaben, B., and Barnhart, C. (2017). Integrated disruption management and flight planning to trade off delays and fuel burn. *Transportation Science*, 51(1):88–111.

Mohammed, R., Rawashdeh, J., and Abdullah, M. (2020). Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results. In *2020 11th International Conference on Information and Communication Systems (ICICS)*, pages 243–248.

Olson, R. S., Cava, W. L., Mustahsan, Z., Varik, A., and Moore, J. H. (2017). Data-driven advice for applying machine learning to bioinformatics problems. In *Biocomputing 2018*, pages 192–203.

Petersen, J. D., Sölveling, G., Clarke, J.-P., Johnson, E. L., and Shebalov, S. (2012). An Optimization Approach to Airline Integrated Recovery. *Transportation Science*, 46(4):482–500.

Rosenberger, J. M., Johnson, E. L., and Nemhauser, G. L. (2003). Rerouting Aircraft for Airline Recovery. *Transportation Science*, 37(4):408–421.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms.

Swiss, S. A., Viceroy, Y., and Dhabi, A. (2016). AMADEUS: Shaping the future of Airline Disruption Management (IROPS). Technical report.

Teodorović, D. and Guberinić, S. (1984). Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operational Research V*, 15(2).

Vink, J., Santos, B. F., Verhagen, W. J. C., Medeiros, I., and Filho, R. (2020). Dynamic Aircraft Recovery Problem - An Operational Decision Support Framework. *Computers & Operations Research*, page 104892.

Vos, H.-W. M., Santos, B. F., and Omondi, T. (2015). Aircraft Schedule Recovery Problem – A Dynamic Modeling Framework for Daily Operations. *Transportation Research Procedia*, 10:931 – 940.

II

Literature Study
Previously graded under AE4020

This page was intentionally left blank.

The Airline Recovery Problem: An Exploratory Review of Practical Challenges

Andrej Nikolajević

Student No. 4445953, Air Transport and Operations, Faculty of Aerospace Engineering, Delft University of Technology

Abstract

Airline disruption management (ADM) remains an integral part of the airline operations process. The practical challenges associated with its automation usually relate to a trade-off of computational time and solution quality. This paper presents state-of-the-art methods used in airline disruption management as presented in literature relating to aircraft, crew, and passenger recovery. The possibilities of integration of machine learning and mathematical methods that exploit the structure of large recovery problems are also discussed. The goal is to provide a critical review of the state-of-the art literature with an aim to discover new possibilities in the implementation of ADM software solutions.

Keywords: Disruption Management, Airline Industry, Irregular Operations, Aircraft Recovery, Crew Recovery, Passenger Recovery, Integrated Recovery, Combinatorial Optimization, Machine Learning

---

1. Introduction

Day-to-day operations in the airline industry are hindered by differences between scheduled and actual performance. These are caused by disruptions such as poor weather conditions, airport congestion, or aircraft mechanical issues, among others. Disruptions create great expenses for airlines and passengers alike and disrupt the original schedule of the airline, preventing a regular day of operations. The Airline Operations Control Center (AOCC) must actively take action to manage these disruptions in order to restore the original schedule of flights, aircraft, and crew, while ensuring that passengers are transported to their intended destinations. Actions include cancellation and delays of flights, rerouting aircraft, reassigning crew, and reaccommodating passengers. These come at the cost of additional fuel expenses, crew overtime, and passenger monetary compensation. The main objective of airline disruption management (ADM) is therefore to solve the airline recovery problem, i.e. to find the minimum-cost solution that recovers the initial airline schedule so that operations may resume as planned.

In the third quarter of 2019, 27.8% of European flights departed late according to EUROCONTROL[1]. Amadeus estimates the total annual costs due to irregular operations at over $60 billion[2] annually, making up over 8% of the annual industry revenue. Clearly, the financial impact of disruptions is very significant. Many disruptions, such as weather conditions, cannot be prevented. However, a solution which is able to incorporate the many different factors present in ADM and automate some of the decision making could drastically reduce the costs incurred by airlines, while ensuring passenger satisfaction.

[1] Eurocontrol Library: https://www.eurocontrol.int/library

[2] Amadeus: https://amadeus.com/documents/en/airlines/white-paper/shaping-the-future-of-airline-disruption-management.pdf

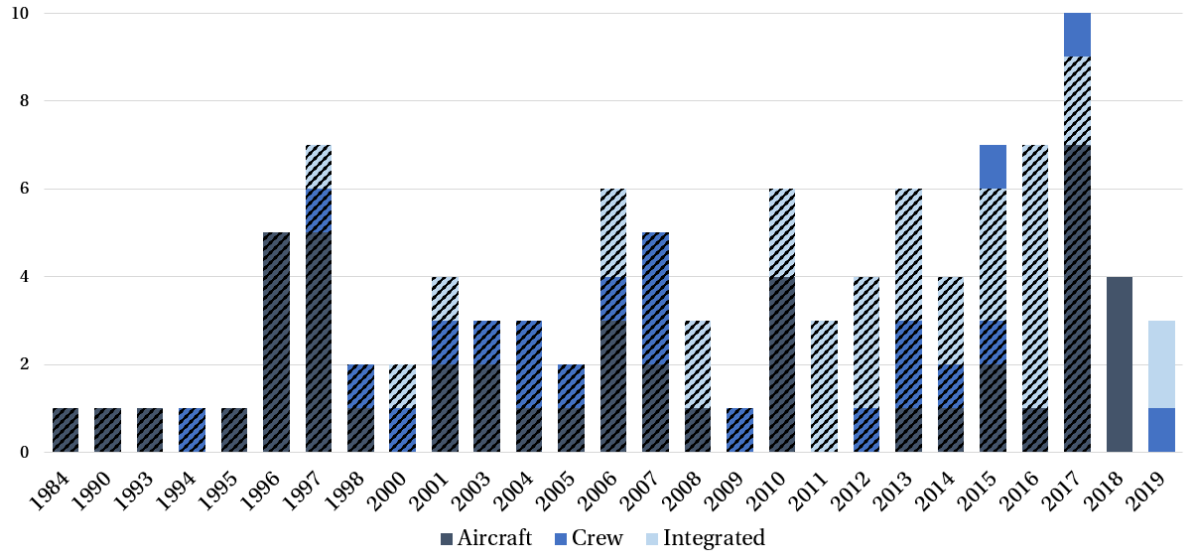


Figure 1.1: Total number of airline recovery problem related publications in the period 1984-2019, adapted from Hassan et al. (2018).

Since the early efforts of Teodorović and Guberinić (1984), a plethora of literature that attempts to solve the airline recovery problem has been published. Clarke (1998) present the first literature review on state-of-the-art decision support systems used in ADM. Kohl et al. (2007) describe an overview of the ADM challenges, along with details and reasoning of the decision-making processes involved in ADM. Several publications being implemented in a major airline are cited as potential solutions. Ball et al. (2007) list promising models on schedule, aircraft, passenger, and crew recovery. They also cite robust scheduling as a preemptive disruption management strategy. Clausen et al. (2010) provide an extensive overview of state-of-the-art ADM solutions. This includes descriptions of network types as well as classification of literature per type of recovery: aircraft, schedule, passenger, crew, and a combination of two or more aspects. A focus was also made on problem dimensions, data types, and solution time due to the more sophisticated nature of recovery problem formulations. More recently, Hassan et al. (2018) extended the work of Clausen et al. (2010) by providing a comprehensive updated collection of relevant publications for the period 2009-2018. They note that the interest in ADM within the academic community is on the rise, with roughly the same number of papers published from 1984-2009 as 2009-2018. Figure 1.1 shows the number of ADM-related publications up to 2019. Note that the patterned publications indicate that the literature has been covered in previous reviews. So far, 2017 has seen the highest number of relevant publications at 10. Since the review by Hassan et al. (2018), 9 new papers have been published on the topic of ADM. Along with the relevant predecessors, these will be discussed in the first part of the literature review.

Due to the scale of airline operations, the size of airline recovery problems can easily get large enough that most methods proposed in literature are simply not fit for real-time use by the AOCC. State-of-the-art solvers implement many algorithmic decisions that require a lot of computational time to reach a solution. In addition, the recovery problem is often repetitive in the sense that the problem structure is very similar from case to case, but with a difference in input. Such solver decisions and problem structures may benefit from a machine learning (ML) assisted approach. The combined use of ML and optimization methods is not a new

concept. Although combinatorial optimization (CO) has been used extensively to improve the performance of ML methods, the reverse is a relatively new area of research with few practical implementation examples. Recently, Bengio et al. (2020) presented a literature review on the state-of-the-art in implementing ML methods to assist in combinatorial optimization. The various types of implementation methods are elaborated on, and relevant literature presented. The second part of this paper will focus on an exploratory study of promising methods integrating ML and CO in an effort to highlight new possibilities in their implementation in airline recovery problems.

This paper is structured as follows: Chapter 2 describes the methodology used to find relevant publications. Chapter 3 gives a brief introduction into the more practical aspects of the airline recovery problem and discusses the relevant literature. Chapter 4 gives an overview of machine learning methods and contains a review of the state-of-the art with respect to machine learning methods used in combination with combinatorial optimization. Finally, Chapter 5 concludes the paper and discusses the possible ways to bring the state-of-the-art closer to implementation in the real life case.

2. Literature Search & Objectives

The purpose of this paper is to review the state-of-the-art in terms of airline disruption management as presented by published scientific literature, with the end goal of defining a research question to pursue within a follow-up MSc Thesis project. The publications discussed in this paper were discovered via use of Google Scholar, SCOPUS, and the Web of Science online databases. Publications were selected for review based on impact (citation count) and relevance, as well as forward and backward snowballing of impactful papers. The entirety of the research history was accounted for in order to gain an overview of the historical trends in implementation, but due to the presence of several literature reviews on the topic, only the most relevant are presented in this paper. For papers relating to the airline recovery problem, several keywords have been identified as providing good results in terms of relevant literature. They are present in the following list.

Keywords: airline recovery, aircraft recovery, crew recovery, pairing recovery, passenger recovery, schedule recovery, integrated recovery, disruption management, irregular operations, airline, aircraft

Combined use of 'AND' and 'OR' conditions for these keywords resulted in 197 papers relating to the airline recovery problem. For papers relating to use of machine learning methods in combinatorial optimization, the relevant keywords are present in the following list.

Keywords: combinatorial optimization, linear programming, integer programming, MILP, MIQP, MIP, computational time, computational performance, solution time, machine learning

In a similar fashion, these keywords resulted in 407 papers combining the use of machine learning and combinatorial optimization. Many of these focused on using CO to improve ML performance, whereas the main interest of this paper focuses on the reverse use. Only papers showing promise for implementation in the

context of the airline recovery problem are discussed in this paper.

3. The Airline Recovery Problem

In order to get to an initial operating schedule, airlines implement a sequential scheduling process. The flight schedule is first determined, followed by the assignment of fleets to individual flights and the generation of flight sequences. Anonymous groups of flight and cabin crew are assigned to flight sequences, and individual crew members are assigned to groups. Finally, individual aircraft are assigned to flights belonging to their fleet. For more detail on the airline scheduling process, the reader is referred to Clausen et al. (2010).

After a disruption, the planned routes of aircraft, crew, and passengers are broken and must be recovered to resume normal operations. Often, the airline recovery problem is formulated as a limited-resource, smaller-scale airline scheduling problem. Traditionally, it is solved sequentially, in a manner similar to the scheduling problem. Due to the complex inter-dependencies between recovery options, the computational time provided by state-of-the-art solvers is often unfit for operational use. The AOCC requires solutions in 2 minutes or less due to the ever-changing nature of the state of operations (Clausen et al. (2010)). This is why many airlines still solve the airline recovery problem by hand, with experienced operators that look for a quick, feasible solution rather than an optimal one. Most of the research on the airline recovery problem focuses therefore on fully or partly automating the disruption management decision making process in order to optimize the solution quality and reduce costs incurred by airlines within the given timeframe. Exact optimization methods and metaheuristic approaches have been the two most common approaches in literature.

Exact optimization methods are those which guarantee a solution that is the global optimum. These are often written in the form of mixed-integer programming (MIP) problems and are solved via use of commercial solvers. As the airline recovery problem is NP-hard, the computational time required to get the optimum solution increases exponentially with the problem size, often making them unfit for real-time use by the AOCC. Much of the published literature therefore utilizes algorithms that reduce the scope of the recovery problem. Metaheuristic methods are also utilized, either in combination with exact optimization methods or individually. Due to extensive literature reviews published by Clausen et al. (2010) and Hassan et al. (2018), only the most significant works in the period 1984-2017 will be mentioned. These were selected based on impact (number of citations) and novelty of the approach. For the years 2018-2019, *all* relevant papers not discussed in aforementioned reviews are presented to provide an overview of new publications.

Section 3.1 will present an overview of the pioneering works relating to the airline recovery problem. Section 3.2 will present recent literature focusing on aircraft and schedule recovery. Section 3.3 will present literature on crew recovery, and literature integrating two or more resources (aircraft, passengers, crew) will be presented in Section 3.4.

3.1. Initial Efforts

Teodorović and Guberinić (1984) were the first to attempt to restore airline schedules after a disruption with an objective to minimize total passenger delay. They used branch and bound on a 3-aircraft fleet with 8 flights and modelled passengers explicitly. This work was later extended by Teodorović and Stojković (1990) with a 14-aircraft fleet and 80 flights. The primary objective was to minimize the total number of cancelled flights, with total passenger delay being used as a secondary objective. Teodorović and Stojković (1995) extend this

model by including crew considerations. The model was tested on over 1000 different randomly generated scenarios with varying fleet and flight numbers and varying disturbance types. These models were largely based on connection networks, as defined by Clausen et al. (2010).

Many works were also based on the concept of time-space networks, where each node in the network corresponds to both a location and a point in time. Jarrah et al. (1993) developed two network flow models as part of United Airlines-backed research that consider flight delays, cancellations, and aircraft swaps as possible recovery options. Thengvall et al. (2003) were the first to implement parallel time-space networks to enable modelling of multiple fleets, where a separate time-space network is created fleet. As an alternative to time-space networks, time-band networks were first utilized for the aircraft recovery problem by Bard et al. (2001) to present a more compact network formulation.

The paper by Argüello et al. (1997) was one of the first to consider a metaheuristic approach to the airline recovery problem by utilizing a greedy randomized adaptive search procedure (GRASP) to repair aircraft routings. The local-search algorithm selects an incumbent solution and then generates a list of candidate neighboring solutions, from which one is randomly selected. The algorithm selects the better solution and re-iterates.

Lettovsky et al. (2000) were some of the first to consider the reassignment of crew duties after disruptions. The crew recovery problem was formulated as an integer programming set-covering model that re-assigns crews to flights while minimizing associated costs. The first approach to integrate the full scope of disrupted resources (aircraft, crew, passengers) was the PhD dissertation by Lettovsky (1997). The author integrated aircraft schedule, crew, and passenger recovery via use of Benders decomposition of the individual resource recovery sub-problems controlled by a master problem.

These papers formed a large part of the basis of modern literature on the airline recovery problem. Since then, the range of recovery options, disturbances, and network sizes has increased drastically. Although exact optimization algorithms have been developed, a lot of the latest literature incorporates the use of metaheuristics or scope-limiting algorithms in order to come closer to the solution time required by the AOCC.

3.2. Aircraft Recovery

Much of the early literature relating to the airline recovery problem focuses on aircraft and schedule recovery. This focus simplifies the recovery problem, enables a sequential approach, and ensures that aircraft schedules, as the most cost-impactful asset, are recovered first. After a disruption, the aircraft recovery problem aims to restore the planned schedules of flights and aircraft by means of e.g. flight delay or cancellation. Despite recent pushes for integration of all resources involved in the recovery problem, aircraft recovery remains an active field of research.

3.2.1. Exact Optimization Methods

Rosenberger et al. (2003) formulate the aircraft recovery problem as a set-partitioning problem that includes airport slot constraints. The model incorporates an aircraft selection heuristic that is used to select a subset of aircraft. The subset is used to preemptively generate the possible spectrum of new routes, which is then optimized over the set-partitioning problem with flight cancellation and delay as the possible recovery options. Several variants of the model are presented with different objectives minimizing delay, cancellations,

and estimated costs of crew and passenger recovery. Results are presented for a recovery solution of 500 days of simulated airline operations, with solution times up to 9 minutes. However, intra-fleet recovery is not considered and no individual disruption solution times are presented.

Andersson and Värbrand (2004) also base their approach on a set-partitioning formulation, but enable intra-fleet recovery. The model utilizes Dantzig-Wolfe decomposition and is solved via two different solution techniques. The first utilizes a column generation (CG) approach, while the second makes use of a Lagrangian relaxation heuristic. Both methods show similar performance and were tested on real data with 30 aircraft and 98 to 215 flights. Solution times under 10 seconds are generally present, but larger instances require times of up to 1100 seconds.

Eggenberg et al. (2010) present a column generation-based model for the aircraft recovery problem that models each aircraft individually. A time-band network approach is utilized, and the use of parallel recovery networks enables the implementation of maintenance constraints per individual aircraft. A formulation of the model is also applied to the passenger recovery problem. The aircraft recovery formulation is tested on real data from Thomas Cook Airlines, with 242 flights and 16 aircraft. A dataset based on the real data is also generated to test the scaleability of the model, with 760 flights and 100 aircraft. Most instances are solved in a minute or less, with the exception of the largest real case which exceeds 1 hour.

Aktürk et al. (2014) were the first to consider aircraft cruise speed control as a recovery option. In most literature, even a 10-minute delay would propagate throughout the flight network. By increasing cruise speed, small delays could be absorbed at the cost of additional fuel. Increasing cruise speed for early arrival also generates additional tail swap options which could drive down the cost of the recovery operation. The authors use a conic-quadratic optimization approach due to the non-linear relationship between cruise speed and fuel burn. The model is tested on two datasets extracted from the U.S. Department of Transport database. These two schedules contain 114 and 207 flights, and 31 and 60 aircraft respectively. For the most computationally expensive case, the solution time does not exceed 250 seconds. The paper concludes that incorporating cruise speed control as a recovery option could lead to significant reductions in recovery costs.

Vos et al. (2015) presented a dynamic framework named the Disruption Set Solver (DSS). The framework recovers the aircraft schedule as disruptions occur, and the recovery actions taken are accounted for when the next disruption occurs. The formulation utilizes parallel time-space networks allowing for explicit modelling of individual aircraft. A candidate aircraft selection algorithm is used to limit the scope of the recovery, and the model is tested on four different cases with fleet data from Kenya Airways. In 93.3% of cases, the model finds the solution in less than 10 minutes. A comparison is made between the solutions of the DSS and of a static model where all the day's disruptions are known upfront. The paper concludes that the use of static disruption scenarios results in underestimation of costs and overestimation of feasible schedule recovery.

3.2.2. Metaheuristics

Løve et al. (2005) used a local search heuristic to solve the aircraft recovery problem. The approach enables aircraft swaps and ferry flights and utilizes steepest ascent hill climbing (or steepest ascent local search, SALS) to get from an initial solution to a local optimum. The objective was to maximize (estimated) airline profit. The model is tested on real data from British Airways with a single fleet of 80 aircraft and 344 flights. Solution times for different weighing of airline costs range from 3.8 to 9.4 seconds.

Qiang et al. (2009) make use of a combination of a greedy randomized adaptive search procedure (GRASP) and simulated annealing to explore the solution space. This reduces the probability of falling into local optima and improves the efficiency of neighborhood selection. The objective of the algorithm is to minimize the total delay time experienced by passengers. A test performed on one disruption scenario with a single-fleet of 30 aircraft and 149 flights resulted in a solution time of 57 seconds.

Guimarans et al. (2015) present a large neighborhood search algorithm based on a constraint programming formulation. A simulation-based approach that demonstrates the stochasticity of real-life disruptions is used to test the discovered solution for robustness. If a solution is deemed too easily disruptable in many of the simulated scenarios, it is rejected. This presents a large advantage with respect to deterministic recovery solutions, which do not account for the future 'disruptability' of the solution. The model is tested on three real flight networks with 9 to 40 aircraft and 49 to 163 flights. For the largest case, the solution time is just under 5 minutes.

Zhao and Chen (2018) developed a weight-table based heuristic to solve the aircraft recovery problem under temporary airport closure disturbances, e.g. due to bad weather. To reduce computational time, a heuristic is used to weigh the possibility of an aircraft swap improving the objective value. The problem is iteratively solved with the highest-weighed aircraft-flight swap options. After each iteration, the weights are adjusted. Flight cancellation is only included as a last-resort measure and thus carries a large penalty in the objective function. Results indicate feasibility of application for a single-fleet, 6-aircraft network, but no computational times are presented.

Liang et al. (2018) present a column-generation based heuristic approach to the aircraft recovery problem. They integrate airport capacity and maintenance constraints. In addition, maintenance flexibility is implemented as a possible recovery option alongside flight delays, flight cancellation, and aircraft swaps. A novelty is modelling continuous delay options instead of discrete. The model is tested on 8 different scenarios with small, medium, and large fleets and flight schedules. A comparison is made between the performance of discrete and continuous delay models. Computational times generally stay under 6 minutes for all cases. The paper concludes that the benefit of modelling of delay as continuous is twofold: (1) the runtime is at least as fast as the discrete version, with large-scale cases being up to 260 times faster, and (2) the costs associated with recovery are reduced. It is also noted that implementation of flexible maintenance could reduce recovery costs by 20-60%.

Lin and Wang (2018) present a two-stage fast variable neighborhood search (FVNS) algorithm for the aircraft recovery problem under airport closure. In the first stage, a feasible flight plan is restored by delaying all flights. In the second stage, the algorithm searches within the local solution space for better solutions. The difference between FVNS and other neighborhood search algorithms is that, once the optimal solution is updated, the FVNS continues searching from the same neighborhood. FVNS also does not consider potentially bad solutions, thus reducing the search space. The model is tested on two separate flight networks from a Taiwanese airline, both operating 70 flights. The solution time is in the order of milliseconds for the used dataset. However, the disruption scenarios are limited and the global optimum solution is not provided as reference. Table 3.1 presents an overview of discussed aircraft recovery literature. Note that only the largest case dimensions and solution times are presented.

Table 3.1: Overview of literature related to aircraft recovery.
Maint. = Maintenance Constraints, Canx = Cancellations, R = Real-life, G = Generated

Paper	Approach/Novelty	Functionalities					Data	Case Dimensions			CPU
		Maint.	Swap	Canx	Retime	Multi-Fleet		Aircraft	Fleets	Flights	(sec)
Rosenberger et al. (2003)	Set-Partitioning	Y	Y	Y	Y	N	G	96	3	469	16
Andersson (2006)	DW	N	Y	Y	Y	N	R	30	2	215	1139
Løve et al. (2005)	SALS	N	Y	Y	Y	N	R	80	1	344	12
Qiang et al. (2009)	GRASP	N	Y	Y	Y	N	G	30	1	149	57
Eggenberg et al. (2010)	CG	Y	Y	Y	Y	N	R	100	1	760	63
Aktürk et al. (2014)	Cruise Speed	N	Y	N	Y	Y	R	60	6	207	202
Vos et al. (2015)	Dynamic Solving	Y	Y	Y	Y	N	R	43	1	-	900
Guimarans et al. (2015)	LNS	N	Y	Y	Y	N	R	48	1	294	178
Zhao and Chen (2018)	Weight Table	N	Y	Y	Y	N	R	6	1	41	N/A
Liang et al. (2018)	CG	Y	Y	Y	Y	N	G	44	1	638	356
Lin and Wang (2018)	FVNS	N	Y	Y	Y	N	R	12	1	70	1

3.3. Crew Recovery

In the sequential recovery approach, crew recovery comes after the remainder of the schedule has been recovered. After aircraft have been re-assigned to different flight schedules, the crew duties are broken. The crew duties must then be restored such that the repaired flight schedule is feasible. Due to strict international regulations on flight and work time for cockpit crew, crew recovery adds a lot of complexity to the airline recovery problem.

3.3.1. Exact Optimization Methods

Abdelghany et al. (2004) present a cockpit crew recovery model that incorporates flight delay as a possible recovery option. Other recovery options include crew swapping, deadheading, and utilizing reserve or standby crew. Crew members are assigned to flights using a MIP formulation, and the objective is to minimize total delay. The model is tested on real data from a U.S. airline and assumes 18 disrupted crew and 12 candidate crew. The solution is obtained in under 2 minutes.

Hoeben et al. (2017) used a dynamic approach to recover cockpit crew under disruptions. Similarly to Vos et al. (2015), disruptions are solved as they occur. A repaired flight schedule is used as input after which the infeasible crew pairings are identified. The model clusters groups of candidate crew based on aircraft and flights operated to reduce problem size. The formulation enables generation of schedules for individual crew members, which enables distinction between captains and first officers. Model performance is evaluated on a set of four disruption scenarios using data from Kenya Airways. Each disruption scenario is a set of disruptions occurring at different times of day. For all scenarios, the average solution time per disruption ranges from 4.9 to 8.2 seconds, while the entire scope of disruptions are consecutively solved in 27.2 to 49.4 seconds, providing solutions fit for operational use. The use of the crew selection algorithm enables 3 to 4 times faster computational times, while differing from the global optimum with a maximum of 21%.

Haouari et al. (2019) present a compact formulation of the daily crew pairing problem. A non-linear MIP formulation incorporating crew regulations is linearized using techniques developed by Sherali and Adams (1994). The model is tested on real data from major European and Asian airlines with to 336 flights and 29 to 172 crew. In most instances, the model performs the optimization in under 1 minute. However, multiple large outliers in the order of thousands of seconds exist, and no clear relationship between problem size and computational time is present.

3.3.2. Metaheuristics

Yu et al. (2003) presented a modified version of the model developed by Wei et al. (1997) which has been implemented for use at Continental Airlines. The model uses a multi-commodity flow formulation and the same depth-first search procedure developed by Wei et al. (1997). The objective is to minimize the cost associated with recovery of crew and flight cancellations. Several recovery options are presented to the operator, which then selects the best solution. The model is tested on various disruption scenarios and fleet sizes, with the largest test case taking 442 seconds to obtain a solution. The authors report that the implementation of this system saved Continental Airlines approximately $40 million in 2001 alone. It should be noted that, due to the sequential recovery process used by most airlines, enabling flight cancellation at the crew recovery stage may prompt a re-iteration of the aircraft recovery problem following the crew recovery.

Castro et al. (2009) use a distributed multi-agent system in which agents perform different roles within the AOCC. The agents compete in finding the best solution to the crew recovery problem. Objectives included minimizing total associated costs, as well as including cost of increasing passenger satisfaction. Though the solution times were around 25 seconds, no case dimensions are presented.

Chen et al. (2013) present a multi-objective genetic algorithm based approach to the integrated crew pairing and crew rostering scheduling problem. This work was later adapted for the crew roster recovery problem in Chen and Chou (2017). The objectives include minimizing the number of changes from initial schedule and the largest duty time change. A variant of the non-dominated sorting genetic algorithm II (NSGA-II) is used, and data is tested on a real-world schedule with simulated disruptions. Although the algorithm presents multiple feasible solutions to the user, details on computational time are not present. Table 3.2 shows an overview of discussed crew recovery literature. Note that only the largest case dimensions and solution times are presented.

Table 3.2: Overview of literature related to crew recovery.
Deadh. = Deadheading, R = Real-life, G = Generated

Paper	Approach	Crew Functionalities				Data	Case Dimensions			CPU
		Swap	Deadh.	Retime	Reserve		Fleets	Flights	Crew	(sec)
Yu et al. (2003)	MCF	Y	Y	N	Y	R	1	-	-	442
Castro et al. (2009)	MAS	N	N	Y	N	G	-	-	-	25
Abdelghany et al. (2004)	MIP	Y	Y	Y	Y	R	-	-	121	120
Hoeben et al. (2017)	MILP	Y	N	Y	Y	R	-	-	-	8
Chen and Chou (2017)	NSGA-II	Y	Y	Y	Y	G	1	12	5	-
Haouari et al. (2019)	Linearization	N	N	N	N	R	1	336	172	2831

3.4. Integrated Recovery

The sequential recovery approach many airlines and researchers take has two disadvantages: (1) the quality of the solution may be suboptimal, and (2) the solution feasibility of secondary recovery stages is not guaranteed. To prevent this, the entire scope of recovery options, including the inter-dependencies between aircraft, passenger, and crew recovery must be accounted for in a single, global-level optimization. In 2009, the French Operational Research and Decision Support Society (ROADEF) hosted a disruption management challenge aiming to integrate recovery of aircraft and passengers. This resulted in numerous publications and gave the airline recovery problem much exposure in the scientific community. Since then, many publications relating

to the airline recovery problem have focused on integrating two or more resources in a single optimization (aircraft, crew, passengers).

3.4.1. Exact Optimization Methods

Bratu and Barnhart (2006) present an integrated model considering only flight delay and cancellation as recovery options. The presence of reserve crew and aircraft is assumed to maintain solution feasibility. Two models are presented: one with passenger delay costs being approximated, and one where passengers are being explicitly modelled. The explicit modelling of passengers results in computational times up to 25 times higher than the approximated case. The model was tested on a four-fleet, 302 aircraft data set, and computational times range from 201 to 5042 seconds.

Abdelghany et al. (2008) present an integrated aircraft and crew recovery formulation that accounts for recovery of both cockpit and cabin crew. A simulation-based approach is used to classify flights likely to be disrupted due to a ground delay program (GDP). The schedule aircraft and crew is then (preemptively) recovered using a rolling horizon approach with the objective to minimize total recovery cost. A test including encompassing 522 aircraft, 1360 pilots, 2040 flight attendants serving 1100 daily flights results in a maximum solution time of 46 seconds. The low runtime compared to the large problem size is aided by the assumption of a single fleet and presence of reserve crew.

Arikan et al. (2017) propose a network flow model that accounts for transport of aircraft, crew, and passengers. Cruise speed control is incorporated via use of conic quadratic optimization, similarly to Aktürk et al. (2014), and the problem size is kept within limits by means of constructing partial networks containing only flight arcs that are likely to be used in the final solution. The model is tested on a flight network of a major U.S. airline operating 1442 daily flights and 402 aircraft. The considered disruptions were varying times of delay for a given flight, as well as varying closure times of a hub. The solution times exceed 1000 seconds for the largest disturbances. The authors note that, especially in large networks, cruise speed control can be used as a valuable recovery option.

Marla et al. (2017) incorporate cruise speed control by creating alternative flight plans corresponding to higher cruise speed with pre-determined cost. Similarly to Vos et al. (2015), the authors use parallel time-space networks to model aircraft individually, but include passenger recovery. The solution is dynamic, i.e. disruptions are solved as they occur. The objective is to minimize the cumulative cost of additional fuel, delays, and cancellations. The model is tested on a 250 flight network. To comply with AOCC-required performance, the runtime is limited to 2 minutes. The MIP gap of the largest recovery operations was 13-15%. The authors note that the inclusion of cruise speed control reduces both the cost of the recovery operation and the propagation of delays throughout the flight network. However, a model without alternative flight plans was consistently solved in under 2 minutes.

Vink et al. (2020) also utilize parallel time-space networks and a similar aircraft selection algorithm to that of Vos et al. (2015), but with a candidate selection based on aircraft location rather than ground time. The focus of the paper is enabling operational use. Passengers are implicitly modelled, and maintenance constraints are considered. An initial trivial solution is found within seconds and presented to the user. The model then re-iterates the recovery problem by adding a single candidate aircraft per iteration and presents the updated best solution once solved. This enables the model's real-time use by the AOCC, as the operator will always

have a feasible solution to act on if the runtime ends up being too high. The model is tested on 10 different disruption scenarios consisting of 2 to 6 disruptions occurring at different times of the day. The selection algorithm found the best solution in 50 seconds or less, and the final solution was identical to the dynamic global optimum (optimizing with all aircraft) in 90% of the cases. On average, the solutions presented by the model were obtained in 22.1 seconds and were only 6% more expensive than the dynamic global optimum.

3.4.2. Metaheuristics

Bisaillon et al. (2011) present a large neighborhood search approach for the integrated aircraft and passenger recovery problem. This model won the 2009 ROADEF Challenge. Starting from a feasible solution, the model iteratively builds and destroys the solution space in search of a better solution and stops after a predetermined time limit or after a certain number of iterations showed no improvement. The work of Sinclair et al. (2014) improved the algorithmic structure of the previous approach and resulted in faster and higher quality solutions. Sinclair et al. (2016) extend this by implementing a column generation heuristic used post-optimization and is the only one of the three works to find the best solution for all cases presented in the ROADEF dataset within 10 minutes.

Petersen et al. (2012) propose a solution to the integrated recovery problem using a decomposition approach similar to that of Lettovsky (1997). Separate recovery sub-problems relating to schedule, aircraft, crew and passengers are developed and linked via Benders decomposition. A column generation algorithm is employed along with a candidate aircraft selection algorithm to reduce problem size. The model is tested on real data from a U.S. airline operating around 800 daily flights and two fleets. Three different disruption scenarios are considered: 50%, 75% and 100% reduction in hub flow rate (arrivals and departures). For the largest case, the solution time is in excess of 30 minutes. A comparison between the integrated and sequential approach concluded that although the solution time for a sequential approach is slightly lower, the integrated approach provides lower recovery costs. In addition, the integrated approach resulted in a feasible solution in 100% of the cases, while the sequential approach only managed to do so for 75% of the cases. This is due to the tendency of the sequential recovery approach to make choices in the high-level recovery problems (aircraft & schedule) that do not consider compatibility with lower level problems. Table 3.3 shows an overview of all discussed integrated recovery literature. Note that only the largest case dimensions and solution times are presented.

Table 3.3: Overview of literature related to integrated recovery.
Maint. = Maintenance Constraints, Canx = Cancellations, R = Real-life, G = Generated

Paper	Elements			Approach	Functionalities					Data	Case Dimensions			CPU
	AC	Pax	Crew		Maint.	Swap	Canx	Retime	Multi-Fleet		Aircraft	Fleets	Flights	(sec)
Bratu and Barnhart (2006)	Y	Y	N	MIP	Y	Y	Y	Y	Y	G	302	4	-	201
Abdelghany et al. (2008)	Y	N	Y	MIP	Y	Y	Y	Y	N	R	5222	1	1100	46
Bisaillon et al. (2011)	Y	Y	N	LNS	Y	Y	Y	Y	N	R	256	1	1423	600
Petersen et al. (2012)	Y	Y	Y	BD	Y	Y	Y	Y	Y	R	-	2	800	1080
Sinclair et al. (2016)	Y	Y	N	LNS, CG	Y	Y	Y	Y	N	R	618	1	2178	1315
Arikan et al. (2017)	Y	Y	Y	Cruise Speed	Y	Y	Y	Y	Y	R	402	-	1254	480
Marla et al. (2017)	Y	Y	N	Cruise Speed	N	Y	Y	Y	N	R	N/A	1	250	120
Vink et al. (2020)	Y	Y	N	Dynamic Solving	Y	Y	Y	Y	N	R	100	1	600	50

3.5. Discussion

Although integration of more resources and recovery options has been proven to achieve more realistic and higher quality solutions, individual recovery problems are still an active area of research. The aircraft recovery problem remains the component with the highest research interest.

Early literature often focused on objectives that would only intuitively help the airline recovery process, such as minimizing total delay, number of cancellations, or number of deviations from the original schedule. These objectives are not the best metric of the effectiveness of the disruption management process as they do not capture the complete financial impact of the solution. A noticeable trend in more recent literature is the formulation of objectives which aim to quantify the direct and indirect costs of delays, cancellations, and other recovery methods. Though precise quantification is impossible due to the involvement of factors such as passenger experience quality and loss of future business, reasonable estimates can ensure that solving the airline recovery problem ultimately focuses on optimizing airline business performance.

A relatively recent addition to the solution of the airline recovery problem is the idea of dynamic solving, i.e. restoring schedules progressively as disruptions occur and building on previously taken recovery actions. This clearly more accurately reflects the real-life situation but brings to light a new issue: solutions obtained as disruptions occur only account for the current state of airline operations, i.e. they build upon local optima. Ideally, a model making decisions in real time should be able to quantify (at least in terms of probability) the quality of its solution with respect to possible future disruptions, making the recovered schedule more robust. Some literature presented acknowledges the potential of this method, but not enough focus is placed on this aspect of airline operations. Though robust recovery of an airline schedule has its benefits, the ultimate goal of recovery is to restore the original schedule. If the original schedule is not generated with the same degree of robustness as the recovery solution, a robust recovery simply restores the schedule to its less-robust original state. In this case, generation of the original schedule with the same degree of robustness may prove more useful than a robust recovery operation. The evaluation of the effectiveness of this approach over the long-term could prove that the robustness of a recovery solution is a significant contributor to cost reduction in the recovery process.

A large variety of real-life constraints applicable to the airline recovery problem are considered in the published literature. Maintenance constraints can often play a critical role in limiting the solution space yet are commonly left out for the sake of simplifying the problem or increasing computational performance. Crew constraints in particular are prone to simplifications due to the complexity of modelling them explicitly. This is the case with many of the constraining factors of the airline recovery problem, such as airport slot constraints or the presence of multiple fleets for which aircraft and crew swaps cannot be performed.

Similarly, though a large variety of recovery options are presented in literature, no currently available publication includes the full scope of recovery options. Cruise speed control, for example, is a relatively recent addition to the recovery options considered in literature that enables not only the absorption of small delays, but arriving earlier in order to introduce additional recovery options. In the case of a flight cancellation, it may also be financially sensible for an airline to re-book disrupted passengers to a flight offered by a competing airline instead of booking hotel rooms for all passengers and introducing an additional flight the next day.

Although many different constraints and recovery options are present in practice, considering all of them in a software-driven airline recovery solution would require levels of processing power that are not currently commercially available. For example, including passenger reallocation to a flight from a competing airline would require the inclusion of the entire network of flights of all airlines. This trade-off of model complexity and solution time is a clear topic of discussion in the presented publications due to the requirements set by the AOCC.

Modern literature often tackles this trade-off with the inclusion of a form of scope-limiting algorithm that generates a subset of recovery options likely to contain the optimal solution. Though good quality solutions can be achieved with these methods, these algorithms are often heuristic and thus do not always result in the optimum. Another approach is to provide the AOCC with at least a single feasible solution in the required time limit, which is then iteratively improved upon as better solutions are found. This approach ensures that the AOCC will at least be able to make use of the software for every disruption, even though the quality of the solutions obtained in the time limit may not be good.

4. Machine Learning & Combinatorial Optimization

Mixed-integer programming has become a powerful tool for solving real-life problems. Due to advances in processing power and continuous addition of heuristics in commercial solvers, we are able to to solve MIP problems millions of times faster than just a few decades ago (Bixby et al. (2004)). Despite this, the NP-hard nature of most MIP problems remains a barrier to their use in real-time settings where solution times are key. The airline recovery problem is a clear example of such a case, where the AOCC is continuously working to resolve disruptions as they occur and must have a solution within minutes. Clearly, the literature presented in the previous chapter has a large focus on bridging this gap.

In the past decade, machine learning has emerged as an extensive set of techniques that are able to extract relevant information from sets of data which have no clear mathematical formulation. As the airline recovery problem is still largely being solved manually via use of skilled operators, machine learning provides the possibility to automate a part or all of that expert intuition. This chapter will discuss the main concepts of machine learning, its use in combination with optimization techniques, and related literature that implements or shows promise for implementation in the airline recovery problem. The nature of this part of the paper is exploratory. That is, papers presented will not always be directly related to the airline recovery problem. However, only papers with topics or conclusions relevant to the issues faced in the airline recovery problems will be discussed. For a more in-depth review of the spectrum of possibilities in using machine learning to assist in combinatorial optimization, the reader is referred to Bengio et al. (2020).

The following sections present an overview of preliminary knowledge used in the remainder of the paper. First, the concepts of combinatorial optimization are presented, followed by those of machine learning. Then, three different outlines for their combined use are presented.

4.1. Combinatorial Optimization

Combinatorial optimization problems are constrained problems aiming to minimize a certain function. The constraints present restrictions on the solution space, and they can define the relationship between variables. Decision variables represent the decisions to be made in the solution space, and the objective func-

tion defines the formulation of variables to be minimised. Linear and non-linear constraints and objective formulations correspond to linear programming (LP) and quadratic programming (QP), respectively. If some variables are constrained to only integer values, then the problem is of a mixed-integer linear (MIL) or mixed-integer quadratic (MIQ) formulation.

The use of mixed-integer linear programming (MILP) formulations is especially widespread due to their ability to model a variety of problems. Unlike MILPs, LP formulations are easily solvable in theory and in practice. The difficulty of solving MILP formulations comes from the integer requirement on certain decision variables. A branch and bound (B&B) approach is commonly applied to solve MILP problems. This algorithmic approach starts with relaxing the MILP formulation, i.e. solving the equivalent LP problem without integer requirements. The feasible region of the LP problem contains the feasible region of the MILP problem. B&B branches on one of the fractional LP-solution variables (nodes). This means that the relaxed formulation is re-solved twice: firstly by constraining the fractional variable to its upper, then to its lower integer bound. This process is reiterated until the best solution satisfying all integer constraints is found. Many modern solvers use large collections of heuristic methods to speed up the computational process. The reader is referred to Lodi (2010) for an in-depth description of B&B and other algorithms utilized by modern MILP solvers.

4.2. Machine Learning

Machine learning is generally split into three main parts: supervised, unsupervised, and reinforcement learning. In supervised learning, a machine learning algorithm attempts to optimize a provided function such that it best relates a given input to a given target output. The process of learning refers to the calibration of that function. As both the input and respective target output are provided, the algorithms are said to be 'supervised'. The loss function relates the difference between the actual and predicted output, and the aim is to minimize its value over a certain 'training' dataset. A common challenge with supervised learning algorithms is generalization of performance, i.e. ensuring that the algorithm's performance on training data is similar to that of any test data within the problem domain. If an algorithm performs much better on training data than unseen test data, it is said to overfit. Learning the training data too well can result in noise and outliers in the data being learned as concepts, which can generalize poorly. On the other hand, learning the training data too loosely can lead to poor model performance on both training and test data.

Reinforcement learning (RL) assumes the presence of a software agent and an environment it interacts with. This is modelled as a Markov decision process. At each time step, the agent is free to take action with respect to its environment and then enters a new state. Based on this state, the agent receives a reward. The aim of the agent is to maximise the sum of future rewards, called the return. The agent then learns the expected value of this return over multiple training instances. A large challenge in RL is setting up a suitable reward function such that the agent does not get stuck in local optima and has sufficient indication of progress.

In unsupervised learning, no target output is provided to the model. Unsupervised learning is used to cluster elements of large datasets with no apparent relationship. As unsupervised learning lacks a target output, measuring the performance of an unsupervised ML algorithm is difficult. Due to the lack of publications combining CO and unsupervised learning, especially in a way relevant to the airline recovery problem, the focus of this paper will remain on supervised and reinforcement learning. For more information on the theo-

retical side of ML and more specifically RL, the reader is referred to textbooks by Deisenroth et al. (2020) and Sutton and Barto (2018) respectively.

4.3. End-to-End Learning

The goal of end-to-end learning is to have a machine learning algorithm that directly solves an optimization problem given a certain input. To enable end-to-end learning in CO problems, both the input data and target output (i.e. optimal solution) must be provided to the model. End-to-end algorithms are able to provide solutions orders of magnitude faster than traditional solvers, though they offer no guarantee of solution optimality or feasibility. Given the often large number of training instances required for a high quality end-to-end model and the long computational time required to solve large NP-hard CO problems, creating a well-functioning end-to-end algorithm can take a lot of time. A schematic representation of an end-to-end learning algorithm is presented in Figure 4.1.

Figure 4.1: Schematic representation of end-to-end learning.

Kirschner and Realff (1999) presented one of the early works related to end-to-end learning. Namely, a decision-tree based binary classification ML algorithm was trained to simulate a MILP formulation that selects the optimal waste containers to be processed at nuclear waste plants. Given features correlated to the impact on the objective function of the MILP model, the ML model was able to correctly classify 88% of waste containers as either to be processed or not, and significantly outperformed random selection methods. However, the inclusion of unfavourable containers negatively impacted the overall objective function value, which was significantly worse than that of the MILP solution. The authors state that the use of a ML classifier that predicts the usefulness of a variable within a MILP formulation may be a better option to improve performance in similar problems, as it more easily avoids the issue of infeasibility. This topic will be elaborated on in the next section.

Larsen et al. (2019) present a regression feedforward neural network approach to solve a stochastic load planning problem dealing with the assignment of containers to train slots. This problem, much like the airline recovery problem, must be solved in real time. A well-solved deterministic MILP formulation exists for this problem, which provides a reasonable solution time. However, the case considered is done under uncertainty of container weights, making a pure MILP formulation unsuitable for use. Therefore, the authors provide exact solutions of an approximated MILP model that accounts for the stochasticity of the problem as the target output to the ML model. While a solver can take up to a full minute to provide a solution, the ML model consistently provides it in a millisecond or less, and it is generally similar in quality to that of the deterministic model. However, a large disadvantage of such end-to-end approaches is the lack of guarantee of solution feasibility. For the best-performing model tested, 2.5% of solutions fell outside of constraint limits, while for the worst up to 38.6% of solutions were infeasible. Despite being unsuitable for direct operational use, this paper indicates that ML algorithms used in end-to-end learning are able to implicitly learn some of the constraints

of a model by seeing many examples.

Bertsimas and Stellato (2019) also presented a neural network-based approach to solve online (real-time, computing time restricted) MIP problems. The method exploits the repetitive problem structure of online MIQ problems to redefine optimization as a multiclass classification problem that learns the logic behind the optimal solution. The resulting model is compared to results obtained by Gurobi. The time required by the ML algorithm to compute a solution stays consistent with increasing problem size at around 1ms. Gurobi presents times one to two orders of magnitude slower. However, the ML model is farther off from the optimal solution as the problem size increases. The scale of problems tested is also relatively small, implying a relatively simple problem structure that may easily be learned by the classifier. For large scale problems such as the ARP, the question of scalability remains unanswered.

Hondet et al. (2018) present a reinforcement learning approach based on Q-learning to solve the aircraft recovery problem. It considers two possible actions that the agent can take given a disrupted aircraft: swap or do not swap aircraft. The number of swap options at each decision are limited to prevent too many possible actions, slowing convergence of the model. The reward the agent receives is the negative of the final cost of the disruptions. The model is trained over 20 000 training instances and its performance is compared to that of the idle case (i.e. do not swap any aircraft). For heavily disrupted scenarios, the agent performs better than the idle case, whereas it is outperformed by the idle case for mild disruptions. The authors note that due to the large number of scenarios possible, a reinforcement learning algorithm applied to aircraft recovery might take orders of magnitude more training instances to be effective. This is especially true if more recovery options are to be presented to the agent, such as delaying other flights or cruise speed control. The use of a reinforcement learning agent to directly solve the airline recovery problem has not yet shown promise.

4.4. Pre-Optimization Machine Learning

As suggested by Kirschner and Realff (1999), another way to utilize ML in combination with CO is to have a ML algorithm learn meaningful properties of CO problems with the goal of reducing computational time. A machine learning algorithm can be used to provide the CO problem with information such as which preprocessing techniques may be useful given the problem structure, or which subset of the solution space is likely to contain the optimal solution. This is presented schematically in Figure 4.2.

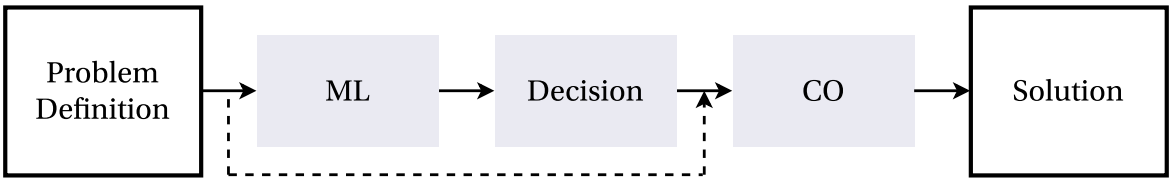


Figure 4.2: Schematic representation of pre-optimization ML.

A clear example of this structure was provided by Kruber et al. (2017). The authors utilize use multiple binary classification algorithms from the scikit-learn library to evaluate whether applying Dantzig-Wolfe (DW) decomposition on a MIP problem will reduce its computational time. The MIP problem characteristics, such as number of variables, proportion of variable types, and number of constraints are used as features, among others. Over all instances, the random forest (RF) classifier performed best and was able to correctly identify

whether a DW approach would speed up solution time in 64.9% of cases. It is unclear whether the use of this classifier reduces the average computational time over all cases considered.

With the end-goal of faster computation, Bonami et al. (2018) utilize a machine learning classifier on a quadratic programming problem to predict whether linearization of the problem would be useful. Similarly to Kruber et al. (2017), the problem characteristics are used as features on multiple binary classification algorithms from the scikit-learn library. The support vector machine (SVM) and RF algorithms performed best, and showed significant improvements in computational time with respect to default CPLEX settings.

Fischetti et al. (2019) utilise ML in a somewhat different context with respect to computational time. Five different ML classifiers from the scikit-learn library are trained to estimate whether a MILP problem will be solved within a given time limit. Given the problem properties as features, RF was the best performing algorithm with a precision and recall of 96%. In the context of airline recovery, this information could be very valuable to the AOCC. Namely, given a recovery operation, a classifier can predict whether the CO formulation will provide a solution within the given time limit or not. In the latter case, the solution can be obtained manually by an operator, ensuring action is taken before the 2 minute time limit.

With respect to the airline recovery problem, Hassan (2018) developed a random forest classifier for the aircraft recovery problem. The enitre recovery formulation is similar to that of Vink et al. (2020), where the candidate aircraft selection algorithm is based on machine learning rather than a heuristic approach. A smaller sub-network is generated with the candidate aircraft for which the aircraft recovery problem is solved. The model is tested on a total 565 different disruption scenarios, applied to a dataset of Delta Airlines flights with 2200 daily flights, 147 airports, and 8 fleets totalling 827 aircraft. On average, the system finds a solution within 48 seconds, with 180 seconds being required for the largest case. The optimal solution is found in 81.3% of the tested cases, and only 4 additional cancellations are present as a result of the selection algorithm. The author mentions that the proposed approach to the airline recovery problem could be extended to add crew while maintaining a compact problem size.

4.5. In-the-Loop Machine Learning

A machine learning algorithm can continuously be called by the CO problem to assist in decisions made during the solution process. An example of such a structure is shown in Figure 4.3. Such algorithms have been applied on the many heuristic decisions made by modern solvers. For example, selecting an unexplored branching node (branching), as well as deciding whether to proceed exploring the selected node (pruning) are often a heuristic decision. Machine learning could therefore be applied in this context to make better choices with respect to which node to branch on. The process of utilizing machine learning to approximate good branching decisions is called learning branching.

He et al. (2014) apply an imitation learning policy for pruning and branching. The solver is based on simple B&B methods and imitates an Oracle that knows the optimal solution and only expands on nodes which contain it. For different problems are tested and their performance compared between the developed model, SCIP, and Gurobi. The authors conclude that given a runtime limit, the developed model outperforms both SCIP and Gurobi in computational time, optimality gap, and integrality gap. Furthermore, in one of the tested problems, neither Gurobi nor SCIP were able to find a feasible solution in the required time, unlike the devel-

oped model.

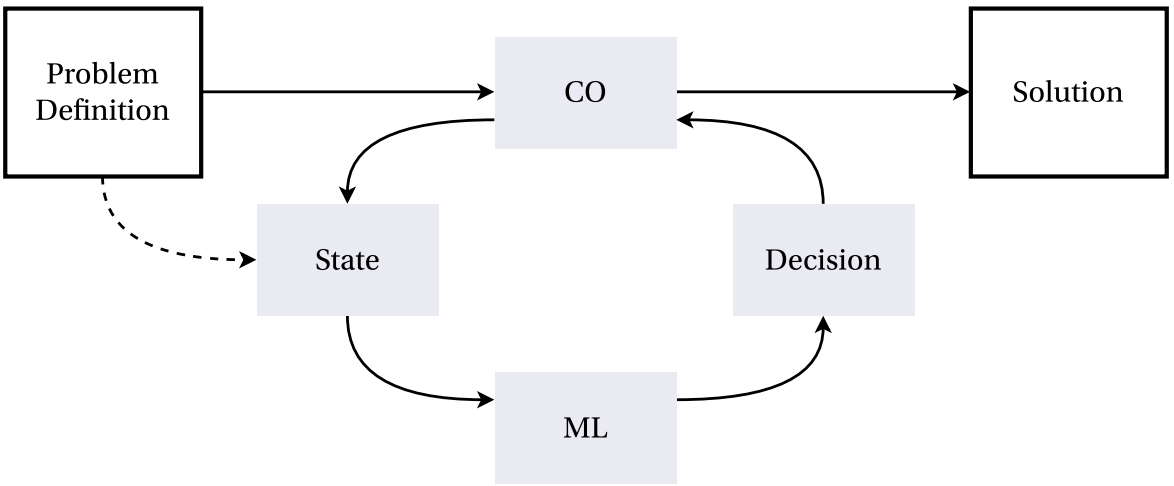


Figure 4.3: Schematic representation of in-the-loop ML.

Similarly, Alvarez et al. (2017) implement imitation learning to approximate strong branching decisions. Strong branching is the most efficient branching strategy in terms of total number of nodes explored. However, the calculation time required to evaluate which node to branch can sometimes be longer than branching on multiple nodes. The authors report results that can, depending on the problem, outperform CPLEX. However, certain problems do not benefit at all from the approximated strong branching approach. Gasse et al. (2019) also approximate strong branching decisions via the use of convolutional neural networks. Similarly, the authors conclude that, in some cases, the imitation of strong branching could be faster than commercial solvers.

Dai et al. (2018) developed a reinforcement learning framework that learns the structure of CO problems on graphs and creates end-to-end heuristics that aim to replace the entirety of the CO solving process. Q-learning is trained over node branching decisions made by CPLEX. The focus of the paper is on online CO problems, as the structure of the problem is key in learning the correct branching decisions. The algorithm shows good performance on several common CO problems, including the Minimum Vertex Cover, Maximum Cut and Traveling Salesman problems. Furthermore, the algorithm develops several greedy heuristics that have not been investigated before and may be a good addition to commercial solvers.

Generally, learning branching is still at an early stage in terms of commercial application. For an in-depth survey of learning branching literature and its success and failure in application, the reader is referred to the review by Lodi and Zarpellon (2017). Table 4.1 presents an overview of the various approaches and goals of the ML use in discussed literature. Note that, when multiple ML algorithms were used, only the best performing one was identified in the ML Type field.

Table 4.1: Overview of literature related to use of ML to assist in CO.
E2E. = End-to-end, PO = Pre-optimization, ITL = In-the-loop

Paper	Approach	ML Type	Goal
Kirschner and Realff (1999)	E2E	Decision Trees	Replace CO
He et al. (2014)	ITL	Imitation Learning	Learn Branching Decisions
Kruber et al. (2017)	PO	Random Forest	Evaluate DW Utility
Alvarez et al. (2017)	ITL	Imitation learning	Learn Strong Branching Decisions
Dai et al. (2018)	ITL	Q-Learning	Learn Branching Decisions
Bonami et al. (2018)	PO	Random Forest	Evaluate Linearization Utility
Hondet et al. (2018)	E2E	Q-Learning	Replace CO
Hassan (2018)	PO	Random Forest	Reduce Solution Space
Gasse et al. (2019)	ITL	Graph Convolutional Neural Networks	Learn Strong Branching Decisions
Fischetti et al. (2019)	PO	Random Forest	Evaluate Computational Time
Bertsimas and Stellato (2019)	E2E	Neural Networks	Replace CO
Larsen et al. (2019)	E2E	Neural Networks	Replace CO

4.6. Discussion

With faster computational time as the end-goal, some publications combining the use of ML and CO focus on the implementation of ML to directly output a solution, i.e. leave out CO solvers. The advantage of end-to-end approaches is that they do not require the use of a commercial solver once the ML algorithm has been trained. Although this has been a promising solution to small-scale online applications, the level of complexity of airline recovery problems would require immense sets of input and output data to train the ML model. Obtaining exact, optimal solutions (i.e. target data) is a very computationally expensive process, and the lack of guarantee of a feasible solution could prove very problematic in real-time application.

The use of ML before initiating the optimization is perhaps most promising to address some of the heuristic scope-limiting algorithms present in airline recovery literature. ML algorithms that make decisions to change problem structure, solution space, and identification of best heuristics to use have all been implemented in recent literature and may be able to generalize well over a vast array of CO problem types. These types of ML applications can also run into similar problems as that of the end-to-end algorithms, especially when in use for reduction of the solution space. However, fine tuning and reasonable amounts of training instances can much more easily prevent these issues than in the case of end-to-end learning.

In-the-loop learning, although proven feasible and sometimes effective, is still far from operational application feasibility. The limited amount of models that outperform commercial solvers indicates that there is still room for research. However, it is a potential future addition to commercial solvers used in online applications. It should be noted that although three separate techniques have been mentioned to classify literature, there is no limitation on their combined use. For example, a scope-limiting algorithm can be used in the pre-optimization stage, while an in-the-loop algorithm can be used during the optimization stage to help with branching decisions.

5. Conclusion & Research Questions

In the context of combinatorial optimization, recent machine learning literature proposes a number of methods that seem suitable for integration with the airline recovery problem. Though end-to-end and in-the-loop machine learning may be feasible in their application, the use of ML prior to starting the recovery operation seems to be the most promising of the three to address the needs of the AOCC in a real-time setting.

The solution times required by the AOCC have led to the inclusion of large simplifications in most of the published airline recovery literature, often relating to airport slot constraints, multi-fleet constraints, and crew labour regulations. Furthermore, the scope of recovery actions is often limited in most literature, whereas many others are present in daily operations. Due to model formulations, cruise speed control in particular is often left out as a recovery option. Full compatibility of the solution with real-life operations, therefore, remains an issue.

These simplifications are helped by the fact that even a good (sub-optimal) solution can improve upon the current disruption management process. Throughout the publications presented in this paper, the majority of the scope of the airline recovery problem, including its constraining factors and recovery options, is well-documented. With the ongoing exponential growth in computing power, perhaps the future will enable a relatively simplification-free application of an airline recovery solution. As a business, airlines will strive for optimality in the long term rather than accept satisfactory performance. This factor will likely be the backbone of airline disruption management research for years to come.

The research question resulting from this literature review can therefore be formulated as follows:

How can the use of machine learning methods be applied to the airline recovery problem, and what are the effects of this integration on the solution quality and solution time?

With the aim of answering the main research question and pinpointing the main steps required to complete the follow-up research project, the following research sub-questions must be answered:

1. What is the most computationally efficient way to incorporate multiple resources into the mathematical formulation of the ARP?
2. Which simplifications to the ARP formulation are acceptable in the context of industry practice?
3. Given the problem formulation, what form of machine learning algorithm and approach best suits the targeted solution time reduction?
4. How should the two main performance metrics, solution time and solution quality, be balanced during model assessment?

Answering all four sub-questions should give a clear idea on what the answer to the main research question is, which will hopefully result in another significant contribution to the domain of airline disruption management.

Bibliography

Abdelghany, A., Ekollu, G., Narasimhan Ram, and Abdelghany Khaled (2004). A Proactive Crew Recovery Decision Support Tool for Commercial Airlines During Irregular Operations. *Annals of Operations Research*, 127(1):309–331.

Abdelghany, K. F., Abdelghany, A. F., and Ekollu, G. (2008). An integrated decision support tool for airlines schedule recovery during irregular operations. *European Journal of Operational Research*, 185(2):825–848.

Aktürk, M. S., Atamtürk, A., and Gürel, S. (2014). Aircraft Rescheduling with Cruise Speed Control. *Operations Research*, 62(4):829–845.

Alvarez, A. M., Louveaux, Q., and Wehenkel, L. (2017). A Machine Learning-Based Approximation of Strong Branching. *INFORMS Journal on Computing*, 29(1):185–195.

Andersson, T. (2006). Solving the flight perturbation problem with meta heuristics. *Journal of Heuristics*, 12(1):37–53.

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

Argüello, M. F., , Bard, J. F., and and Yu Gang (1997). A Grasp for Aircraft Routing in Response to Groundings and Delays. *Journal of Combinatorial Optimization*, 1(3):211–228.

Arikan, U., Gürel, S., and Aktürk, M. S. (2017). Flight network-based approach for integrated airline recovery with cruise speed control. *Transportation Science*, 51(4):1259–1287.

Ball, M., Barnhart, C., Nemhauser, G., and Odoni, A. (2007). Chapter 1 Air Transportation: Irregular Operations and Control. In Barnhart, C. and Laporte, G., editors, *Handbooks in Operations Research and Management Science*, volume 14, pages 1–67. Elsevier.

Bard, J. F., , Yu, G., , and Argüello, M. F. (2001). Optimizing Aircraft Routings in response to Groundings and Delays. *IIE Transactions*, 33(10):931–947.

Bengio, Y., Lodi, A., and Prouvost, A. (2020). Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*.

Bertsimas, D. and Stellato, B. (2019). Online Mixed-Integer Optimization in Milliseconds. *ArXiv*, abs/1907.02206.

Bisaillon, S., Cordeau, J.-F., Laporte, G., and Pasin, F. (2011). A large neighbourhood search heuristic for the aircraft and passenger recovery problem. *4OR*, 9(2):139–157.

Bixby, R., Fenelon, M., Gu, Z., Rothberg, E., and Wunderling, R. (2004). Mixed-Integer Programming: A Progress Report. In *The Sharpest Cut: The Impact of Manfred Padberg and His Work*, pages 309–324.

Bonami, P., Lodi, A., and Zarpellon Giulia (2018). Learning a Classification of Mixed-Integer Quadratic Programming Problems. In van Hoeve, W.-J., editor, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 595–604, Cham. Springer International Publishing.

Bratu, S. and Barnhart, C. (2006). Flight operations recovery: New approaches considering passenger recovery. *Journal of Scheduling*, 9(3):279–298.

Castro, A. J. M., , and Oliveira, E. (2009). A Multi-Agent System for Airline Operations Control. In Demazeau Yves, , Pavón, J., and Corchado Juan M., and and Bajo Javier, editors, *7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009)*, pages 159–168, Berlin, Heidelberg. Springer Berlin Heidelberg.

Chen, C. H. and Chou, J. H. (2017). Multiobjective Optimization of Airline Crew Roster Recovery Problems under Disruption Conditions. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):133–144.

Chen, C. H., Liu, T. K., and Chou, J. H. (2013). Integrated short-haul airline crew scheduling using multiobjective optimization genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 43(5):1077–1090.

Clarke, M. D. D. (1998). Irregular airline operations: a review of the state-of-the-practice in airline operations control centers. *Journal of Air Transport Management*, 4(2):67–76.

Clausen, J., Larsen, A., Larsen, J., and Rezanova, N. J. (2010). Disruption management in the airline industry-Concepts, models and methods. *Computers and Operations Research*, 37(5):809–821.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., and Song, L. (2018). Learning Combinatorial Optimization Algorithms over Graphs.

Deisenroth, M. P., Faisal, A. A., and Ong, C. S. (2020). *Mathematics for Machine Learning*. Cambridge University Press.

Eggenberg, N., Salani, M., and Bierlaire, M. (2010). Constraint-specific recovery network for solving airline recovery problems. *Computers and Operations Research*, 37(6):1014–1026.

Fischetti, M., Lodi, A., and Zarpellon, G. (2019). Learning MILP Resolution Outcomes Before Reaching Time-Limit. In Rousseau, L.-M. and Stergiou, K., editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 275–291, Cham. Springer International Publishing.

Gasse, M., Chetelat, D., Ferroni, N., Charlin, L., and Lodi, A. (2019). Exact Combinatorial Optimization with Graph Convolutional Neural Networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32, pages 15580–15592. Curran Associates, Inc.

Guimarans, D., Arias, P., and Mota, M. M. (2015). Large Neighbourhood Search and Simulation for Disruption Management in the Airline Industry. In *Applied Simulation and Optimization*, pages 169–201. Springer International Publishing.

Haouari, M., Mansour, F. Z., and Sherali, H. D. (2019). A new compact formulation for the daily crew pairing problem. *Transportation Science*, 53(3):811–828.

Hassan, L. K. (2018). Aircraft Disruption Management Increasing Performance with Machine Learning Predictions. Technical report.

Hassan, L. K., Santos, B. F., and Vink, J. (2018). Airline Disruption Management: A Literature Review and Practical Challenges.

He, H., Daume III, H., and Eisner, J. M. (2014). Learning to Search in Branch and Bound Algorithms. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3293–3301. Curran Associates, Inc.

Hoeben, N., Santos, B., and Omondi, T. (2017). Dynamic Crew Pairing Recovery. In *Conference: ATRS 2017 - Air Transport Research Society World Conference*.

Hondet, G., Delgado, L., and Gurtner, G. (2018). Airline Disruption Management with Aircraft Swapping and Reinforcement Learning. In *SESAR Innovation Days 2018 (SID 2018)*, Salzburg, Austria.

Jarrah, A. I., Yu, G., Krishnamurthy, N., and Rakshit, A. (1993). Decision support framework for airline flight cancellations and delays. *Transportation Science*, 27(3):266–280.

Kirschner, K. J. and Realff, M. J. (1999). A Joint Optimization and Machine Learning Method for a Selection and Grouping Problem. *Chemical Engineering Research and Design*, 77(4):271–280.

Kohl, N., Larsen, A., Larsen, J., Ross, A., and Tiourine, S. (2007). Airline disruption management-Perspectives, experiences and outlook. *Journal of Air Transport Management*, 13(3):149–162.

Kruber, M., , Lübbecke, M. E., and and Parmentier Axel (2017). Learning When to Use a Decomposition. In Salvagnin Domenico, , and Lombardi, M., editors, *Integration of AI and OR Techniques in Constraint Programming*, pages 202–210, Cham. Springer International Publishing.

Larsen, E., Lachapelle, S., Bengio, Y., Frejinger, E., Lacoste-Julien, S., and Lodi, A. (2019). Predicting Tactical Solutions to Operational Planning Problems under Imperfect Information. *ArXiv*, abs/1901.07935.

Lettovsky, L. (1997). *Airline Operations Recovery: An Optimization Approach*. PhD thesis, Georgia Institute of Technology.

Lettovsky, L., Johnson, E. L., and Nemhauser, G. L. (2000). Airline crew recovery. *Transportation Science*, 34(4):337–348.

Liang, Z., Xiao, F., Qian, X., Zhou, L., Jin, X., Lu, X., and Karichery, S. (2018). A column generation-based heuristic for aircraft recovery problem with airport capacity constraints and maintenance flexibility. *Transportation Research Part B: Methodological*, 113:70–90.

Lin, H. and Wang, Z. (2018). Fast Variable Neighborhood Search for Flight Rescheduling after Airport Closure. *IEEE Access*, 6:50901–50909.

Lodi, A. (2010). Mixed Integer Programming Computation. In Jünger Michael, , Liebling, T. M., and Naddef Denis, and Nemhauser George L., and Pulleyblank William R., and Reinelt Gerhard, and Rinaldi Giovanni, and and Wolsey Laurence A, editors, *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, pages 619–645. Springer Berlin Heidelberg, Berlin, Heidelberg.

Lodi, A. and Zarpellon, G. (2017). On learning and branching: a survey. *TOP*, 25(2):207–236.

Løve, M., Sørensen, K. R., Larsen, J., and Clausen, J. (2005). Using Heuristics to Solve the Dedicated Aircraft Recovery Problem. *Central European Journal of Operations Research*, 13(2):189–207.

Marla, L., Vaaben, B., and Barnhart, C. (2017). Integrated disruption management and flight planning to trade off delays and fuel burn. *Transportation Science*, 51(1):88–111.

Petersen, J. D., Sölveling, G., Clarke, J.-P., Johnson, E. L., and Shebalov, S. (2012). An Optimization Approach to Airline Integrated Recovery. *Transportation Science*, 46(4):482–500.

Qiang, G., Xiao-wei, T., and Jin-fu, Z. (2009). 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.

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

Sherali, H. D. and Adams, W. P. (1994). A hierarchy of relaxations and convex hull characterizations for mixed-integer zero—one programming problems. *Discrete Applied Mathematics*, 52(1):83 – 106.

Sinclair, K., Cordeau, J. F., and Laporte, G. (2014). Improvements to a large neighborhood search heuristic for an integrated aircraft and passenger recovery problem. *European Journal of Operational Research*, 233(1):234–245.

Sinclair, K., Cordeau, J.-F., and Laporte, G. (2016). A column generation post-optimization heuristic for the integrated aircraft and passenger recovery problem. *Computers & Operations Research*, 65:42 – 52.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA.

Teodorović, D. and Guberinić, S. (1984). Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operational Research V*, 15(2).

Teodorović, D. and Stojković, G. (1990). Model for operational daily airline scheduling. *Transportation Planning and Technology*, 14(4):273–285.

Teodorović, D. and Stojković, G. (1995). Model to Reduce Airline Schedule Disturbances. *Journal of Transportation Engineering*, 121(4):324–331.

Thengvall, B. G., Bard, J. F., and Yu, G. (2003). A bundle algorithm approach for the aircraft schedule recovery problem during hub closures. *Transportation Science*, 37(4):392–407.

Vink, J., Santos, B. F., Verhagen, W. J. C., Medeiros, I., and Filho, R. (2020). Dynamic Aircraft Recovery Problem - An Operational Decision Support Framework. *Computers & Operations Research*, page 104892.

Vos, H.-W. M., Santos, B. F., and Omondi, T. (2015). Aircraft Schedule Recovery Problem – A Dynamic Modeling Framework for Daily Operations. *Transportation Research Procedia*, 10:931 – 940.

Wei, G., Yu, G., and Song, M. (1997). Optimization Model and Algorithm for Crew Management During Airline Irregular Operations. *Journal of Combinatorial Optimization*, 1(3):305–321.

Yu, G., Argüello, M., Song, G., McCowan, S. M., and White, A. (2003). A New Era for Crew Recovery at Continental Airlines. *INFORMS Journal on Applied Analytics*, 33(1):5–22.

Zhao, T. and Chen, X. (2018). A Weight-Table Based Heuristic Algorithm for Aircraft Recovery Problem. In *2018 37th Chinese Control Conference (CCC)*, pages 2242–2246.

III

Supporting work

This page was intentionally left blank.

1

Model Framework

This chapter will provide more detail into how the Sequential Disruption Set Solver (SDSS) functions. The framework of the aircraft recovery stage remains largely identical to that of Hassan (2018), with the exception of the addition of the loading of scheduled and reserve crew data during the pre-processing phase and inclusion of crew considerations within the connecting passenger matrix. The four main parts constituting the aircraft recovery stage are present in Figure 5.7. In this figure, bullet points marked with an asterisk (*) represent those that were changed as part of this research. Starting with a disruption scenario, the aircraft recovery stage loads the relevant aircraft schedules and processes disruption information into features. These are passed on to the RF classifier which outputs a per-aircraft probability of use within the optimal solution. The disruption solver uses these probabilities to make a selection, and write and solve the linear programming formulation of the aircraft recovery problem. The solution is sent to post-processing, and finally forwarded to the crew recovery stage. As the specifics of the aircraft recovery stage have been elaborated on in detail by Hassan (2018), this chapter will focus on the details of the crew recovery stage.

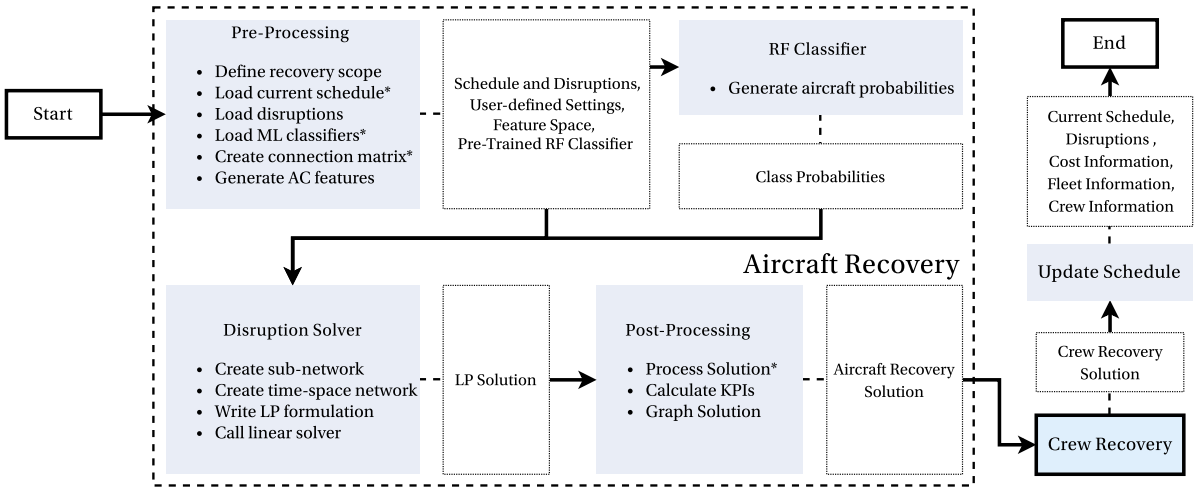


Figure 1.1: SDSS aircraft recovery stage framework.

Much like the aircraft recovery stage, the crew recovery stage is divided into four main parts: (1) pre-processing, (2) classification, (3) LP-solving, and (4) post-processing. An overview of the crew recovery stage as part of the entire SDSS formulation is present in Figure 1.2. Each of these parts will be elaborated on in the following four sections. The crew recovery process begins with the actions taken by aircraft recovery, which are transformed into relevant input data during the pre-processing phase.

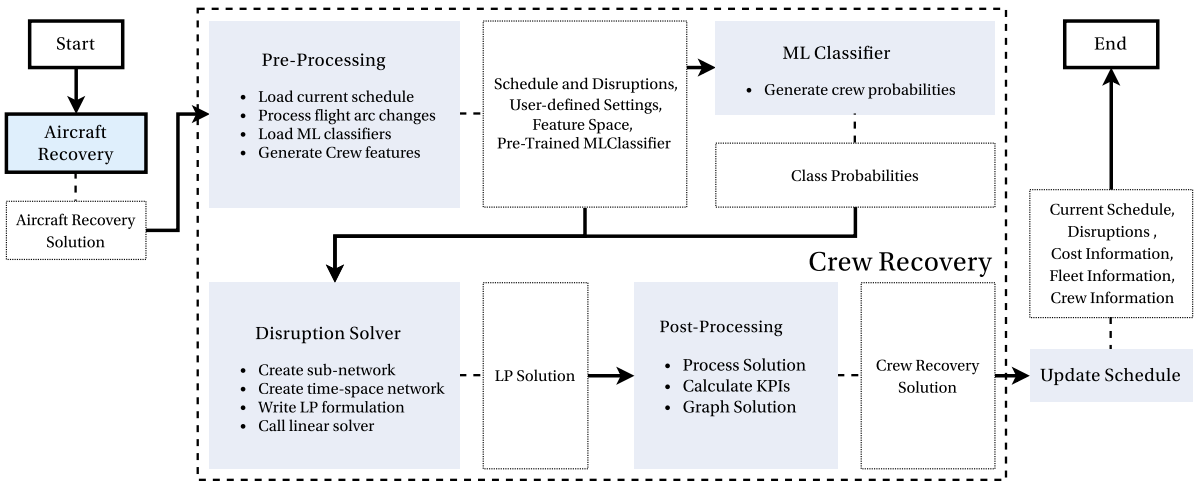


Figure 1.2: SDSS crew recovery stage framework.

1.1. Pre-processing & Classification

During the pre-processing phase, the changes made to the schedule by the aircraft recovery stage are loaded and processed into disruptions within the crew schedule. The original disruptions to the aircraft and flight schedule are also considered, as they are included in the solution. The crew schedule is passed on from the aircraft recovery stage where it was loaded, and the machine learning classifier for crew selection is loaded. The feature space is also generated during this step, as shown in Figure 1.3.

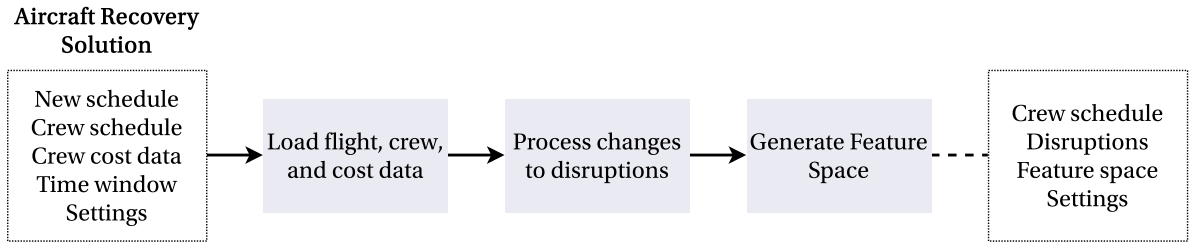


Figure 1.3: SDSS crew recovery stage pre-processing flowchart

The disruptions in the crew schedule originate from the changes made during the aircraft recovery stage and can be of two types: (1) delays to flight arcs and (2) flight arc cancellations. The aircraft recovery stage also considers tail swaps as recovery options, but these occur as a result of a delay or cancellation. The effect of tail swaps on the crew schedule is therefore already captured in the said delay and/or cancellation. In essence, the crew recovery stage assigns crew to flight arcs, with no regard for the aircraft flying them, as these have already been assigned in the previous stage. Following the acquisition of the disruptions, the feature space is generated by processing each crew's scheduled flights and duty and flight times with respect to the disruptions. Before the classification, the feature space consists of a dataframe with a layout similar to that of Table 1.1, though it lacks the target vector Y. For each candidate crew C, the value of each of the model's n features is determined. These are then passed on to the machine learning classifier for classification.

Table 1.1: The post-classification feature space.

	F_1	F_2	F_3	...	F_n	Y
C_1	$v_{1,1}$	$v_{1,2}$	$v_{1,3}$	...	$v_{1,n}$	$y_{1,1}$
C_2	$v_{2,1}$	$v_{1,2}$	$v_{2,3}$	...	$v_{2,n}$	$y_{1,1}$
C_3	$v_{3,1}$	$v_{1,2}$	$v_{3,3}$	...	$v_{2,n}$	$y_{1,1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
C_m	$v_{m,1}$	$v_{m,2}$	$v_{m,3}$	...	$v_{m,n}$	$y_{m,n}$

As no flight delay opportunities exist within the crew recovery stage, an evaluation of whether crew duties are broken is performed within the feature space generation. If no duties are broken and the changed schedules can be absorbed by scheduled crew, the solution is identified as trivial, and only the crew scheduled to operate the disrupted flights are passed to the solver. When this is not the case, the classifier uses the generated feature space to output a per-crew probability of being used in the optimal solution, as seen in Figure 1.4.

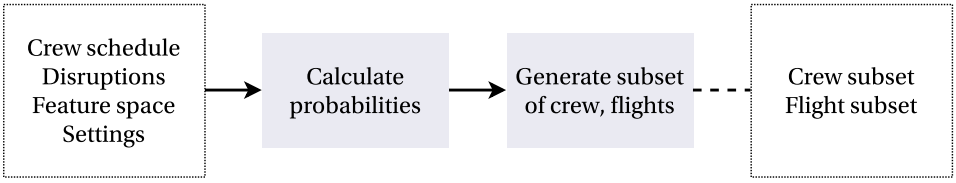


Figure 1.4: SDSS crew recovery stage classifier flowchart

Several ways of classification are presented to the user within the settings, the more basic of which is a fixed probability threshold. The user-defined probability threshold classifies a crew positively (True) or negatively (False) if the probability of their use in the optimal solution as determined by the classifier are above and below the threshold respectively. During evaluation, some large disruption cases were not reduced by a significant amount via the use of the threshold alone. This has been addressed via the addition of a further reduction of the initial selection, which is present as the second classification option within the user-defined settings. Namely, any selection that is larger than 100 crews and does not reduce the number of crews by at least 50% is revisited. The crews are sorted by probability of use in the optimal solution, and either the top 100 or top 50% of crews are selected, whichever is larger. 100 crews was chosen as a sufficient reduction as disruption sets with 100 crews or less were generally solved in under 20 seconds. With this selection, crew members likely to be used within the optimal solution are identified, and the classifier passes on the set of these crews and their scheduled flights to the disruption solver, where they will be used to generate a sub-network. The quality of this selection algorithm is evaluated during the sensitivity analysis.

1.2. Disruption Solver

The disruption solver takes the processed disruptions, settings, and the subset of the crew and flight schedules as input, and uses it to create and solve the LP formulation of the crew recovery problem. The schedules are used to create a time-space sub-network containing only the relevant flights and crews as identified by the machine learning classifier. This is followed by a per-flight and per-crew cost calculation, and finally by the writing and solution of the LP problem, as seen in Figure 1.6.

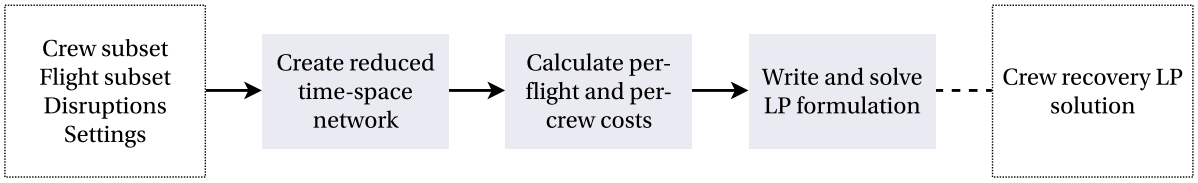


Figure 1.5: Determination of origin and sink nodes for crews entering or exiting the time window.

During the generation of the time-space network, each crew's specific origin and sink nodes are identified. The way this is done can be seen in Figure 1.6. For crew starting before the time window and entering it via a flight, such as Crew I on the figure, the origin node is considered the arrival node of the flight. The sink node is determined as the node at which the crew reaches its maximum allowable duty time. If a crew's maximum allowable duty time allows them to work outside of the time window, the sink node is the last node within the time window at the location of the last ground arc used, as seen on the figure for Crew II. For crews whose duties start and end within the time window, the origin and sink nodes are identified as the departure node of the first and arrival node of the final flight, respectively.

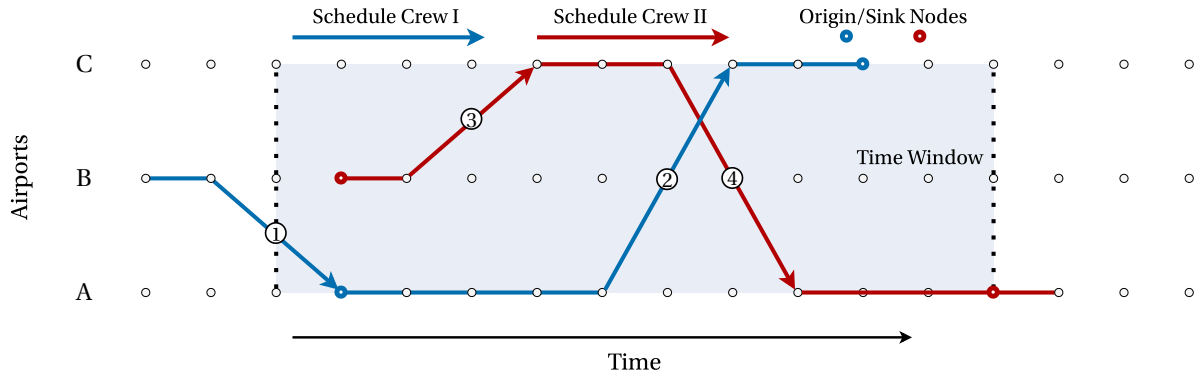


Figure 1.6: Determination of origin and sink nodes for crews entering or exiting the time window.

Once the time-space network is generated, the per-flight and per-crew costs are calculated for use in the LP model objective function. These contain the operating cost of crew on each flight, flight cancellation costs, as well as the penalty costs for each of the penalties incurred by the crew LP model. The five types of penalties the crew model incurs are:

1. Crew swap penalty - incurred when a flight is operated by non-scheduled crew.
2. Flight time limit penalty - incurred when crew exceeds the daily scheduled flight time to an amount no greater than the maximum legal allowable flight time.
3. Sink node violation penalty - incurred when a crew is not located at its designated airport(s) at its designated sink time.
4. Flight cancellation penalty - incurred upon flight cancellation and equal to the estimated flight cancellation cost as calculated by the aircraft recovery stage

The order in which these penalties are sorted also corresponds to the order of their impact on the objective function from lowest to highest. Following the cost calculation, the decision variables and objective function are written, along with the corresponding constraints. The solution obtained by the disruption solver is passed on to the post-processing phase, where the aircraft and crew recovery process is completed.

1.3. Post-processing

The post-processing phase aims to write the solution into a schedule and provide the user with useful information regarding the solution via the use of time-space network graphs and key performance indicators (KPIs). Following the LP solution, the values of the decision variables must be converted into crew schedule decisions. The decision variables also provide insight into the main KPIs used to evaluate the solution. These are present in Table 1.2. In essence, the cost of the reserve crew solution is based on the values of the five penalties the model uses. The quality of solution can therefore be compared in terms of the number of cancelled flights, number of swaps, number of flight time violations, and number of scheduled and reserve crew sink violations, as well as by the final objective function value.

Table 1.2: Key performance indicators used in the evaluation of the crew recovery stage solutions.

Abberviation	Name	Unit	Description
Sol.T.	Solution Time	[s]	Solution time required to solve disruption
Crew	Crews Considered	[#]	Number of crews considered in the recovery model
Disr.C.	Disruption Cost	[$]	Cost of disruption solution, excluding scheduled costs.
Opt.S.	Optimal Solutions	[#]	Number of runs that obtained optimal solutions.
Canx	Cancellations	[#]	Number of flight cancellations.
SNV	Sink Violations	[#]	Number of sink node constraint violations.
DH	Deadheads	[#]	Number of crew deadheads.
Swap	Crew Swaps	[#]	Number of crew swaps.
RCU	Reserve Crew Used	[#]	Number of reserve crews used.
U60	Runs Under 60 s	[#]	Number of runs that achieved a solution in under 60 seconds.
U120	Runs Under 120 s	[#]	Number of runs that achieved a solution in under 120 seconds.

The KPIs are stored, after which a time-space network graph is generated to graphically display the changes made to the schedule. This is the final step of the crew recovery stage and therefore the SDSS. a flowchart of the post-processing stage of the SDSS is shown in Figure 1.7.

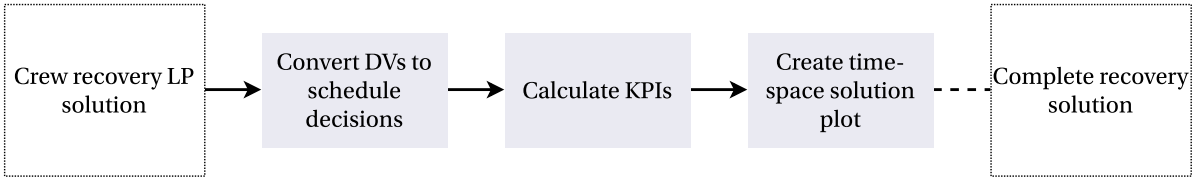


Figure 1.7: Flowchart of post-processing within the crew recovery stage.

2

Crew Recovery Formulation

This chapter describes the detailed binary integer programming (BIP) formulation of the crew recovery problem. To avoid overlap with the work of Hassan (2018), only the developed crew recovery formulation has been elaborated on. The SDSS relies on the concept of parallel time-space networks to implement constraints at a crew level. This means that the movement of each crew through time and space is tracked and constrained separately, as each crew has, in essence, its own time-space network. The constraints between parallel time-space networks ensure that a single flight is always operated by a single crew. As the aircraft recovery stage relies on a formulation with similar logic, the explanations presented here can easily be extrapolated to the formulation of the aircraft recovery problem. The following lists describe the sets, decision variables, non-decision variables, and mathematical model used by the crew recovery.

Sets:

- **F** - Set of flights i
- **A** - Set of airports a
- **N** - Set of all nodes n
- $\mathbf{N_o}$ - Set of origin nodes n
- $\mathbf{N_i}$ - Set of intermediate nodes n
- $\mathbf{N_s}$ - Set of sink nodes n
- **K** - Set of crews k

In the above list, the sets of origin, sink, and intermediate nodes are defined at a crew level, i.e. each crew has its own set of the three node types. These sets enable constraining different crews to different start and end locations and times, and are created during the pre-processing phase. The set of crews K also includes reserve crews. The following decision variables are used in the crew recovery model. As all the decision variables are binary, their explanation contains a description of the condition at which they take on a positive value.

Decision Variables:

- $\delta_{K_{k,i}}$ - if k allocated to i
- $\delta_{G_{k,n}}$ - if k uses n-originating ground arc
- $\delta_{DH_{k,i}}$ - if k deadheaded on i
- $\delta_{K'_i}$ - if i flown by unscheduled crew
- δ_{CX_i} - if i is cancelled
- s_k - slack if sink constraint violated
- s_{FT_k} - slack if scheduled flight time is exceeded

As the decision variables imply, the three actions a crew may take between their origin and sink node are (1) flying a flight, (2) being deadheaded (flying as a passenger) on a flight, or (3) being on the ground. δ_{CX_i} is present as a decision variable in the case no crew may be assigned to a flight. If this is the case, the flight is cancelled. The remaining variables define the slack variables used for maintaining problem feasibility and penalising unwanted behaviour. The variables for constraining the origin and sink nodes ensure that a crew starts and ends their duty at their designated times and locations. Each crew has a penalty associated with missing their sink node, though this cost is lower for reserve crew than regularly scheduled crew. Finally, each flight not flown by its scheduled crew is penalised with the aim of minimising the number of flights not flown by originally scheduled crew. Each of these variables is associated with a cost in the objective function, and the parameters used to determine it are present in the following list.

Parameters:

- C_{G_n} - Cost of ground arc originating from n.
- $C_{K_{k,i}}$ - Operating cost of crew k on flight i
- $C_{DH_{k,i}}$ - Deadhead cost of crew k on flight i
- C_{OC} - Penalty if flight flown by non-scheduled crew
- C_{SV} - Penalty if sink node constraint violated
- C_{FT} - Penalty if scheduled flight time exceeded
- C_{CX_i} - Cancellation cost of flight i
- FT_i - Flight time of flight i
- FTL_k - Flight time remaining within TW for crew k
- FTM_k - Maximum additional flight time for crew k

Though ground arc costs are set at zero for this research, their addition is necessary to ensure proper movement of crew through time and space. The FT_i and FTL_k parameters are used to constrain the flight time of crew within the time window. The aforementioned decision and non-decision variables are used to create the objective function. The aim of the crew recovery problem is to minimise the costs of disruptions incurred by the crew schedule, in the form of crew operating costs, crew deadhead costs, and flight cancellation costs. These are present in the following objective function:

$$\begin{aligned} Min \underbrace{\sum_{k\in K}\sum_{i\in F}\left(C_{K_{k,i}}\cdot\delta_{K_{k,i}} + C_{DH_{k,i}}\cdot\delta_{DH_{k,i}}\right)}_{\text{Flight operating/deadhead costs}} + \underbrace{\sum_{k\in K}\sum_{n\in N} C_{G_n}\cdot\delta_{GK_{k,n}}}_{\text{Ground arc costs}} + \underbrace{\sum_{i\in F} C_{CX}\cdot\delta_{CX_i}}_{\text{Cancellation costs}} \\ \underbrace{+\sum_{i\in F} C_{OC}\cdot\delta_{K'_i} + \sum_{k\in K} C_{SV}\cdot s_k + \sum_{k\in K} C_{FT}\cdot s_{FT_k}}_{\text{Slack costs}} \end{aligned} \tag{2.1}$$

The first element represents the costs incurred by assigning crew to operate or be deadheaded on flights. The second and third elements represent the costs of assigning crew to ground arcs and cancelling flights, respectively. The final element represents the slack costs that penalise the model for taking unfavorable actions. This is done only when the penalty cost is lower than the other options available for recovery. The way these elements function within the minimization problem is governed by the following set of constraints:

$$\delta_{CX_i} + \sum_{k\in K_e}\delta_{K_{k,i}} = 1 \quad \forall i \in F, \forall e \in E \tag{2.2}$$

Equation 2.2 forces all flights to either be assigned a crew or cancelled. The following three constraints ensure the node-balance of the time-space network. Equation 2.3 ensures that the net flow between any two intermediate (i.e. not origin or sink) nodes is equal to zero. This means that a crew may not begin or end their duty at any node other than the origin or sink node respectfully. To supplement this constraint, Equation 2.4 ensures that the net flow out of the origin node is equal to one, while Equation 2.4 constraints the net flow into the sink node at one.

$$\left(\delta_{GK_{k,n-1}} + \sum_{i\in F_{in}}(\delta_{K_{k,i}} + \delta_{DH_{k,i}})\right) - \left(\delta_{GK_{k,n}} + \sum_{i\in F_{out}}(\delta_{K_{k,i}} + \delta_{DH_{k,i}})\right) = 0 \quad \forall k \in K, n \in N_i \tag{2.3}$$

$$\delta_{GK_{k,n}} + \sum_{i\in F_{out}}(\delta_{K_{k,i}} + \delta_{DH_{k,i}}) = 1 \quad \forall k \in K, n = \text{scheduled } N_o \text{ of } k \tag{2.4}$$

$$\delta_{GK_{k,n-1}} + \sum_{i\in F_{IN_n}}(\delta_{K_{k,i}} + \delta_{DH_{k,i}}) + s_k = 1 \quad \forall k \in K, n = \text{scheduled } N_s \text{ of } k \tag{2.5}$$

Equation 2.6 ensures the value of $\delta_{K'_i}$ is equal to one whenever a flight is operated by a non-scheduled crew. In this way, a penalty is added to the objective function for every crew swap performed.

$$\delta_{K_{k,i}} - \delta_{K'_i} = 1 \quad \forall i \in F, k = \text{ crew scheduled for } i \tag{2.6}$$

Finally, the following constraint ensures that the flight time of a crew within the time window does not exceed the allowable flight time. If the flight time within the time window exceeds the scheduled flight time, a penalty is added to the objective function.

$$\sum_{i\in F}\delta_{K_{k,i}}\cdot FT_i \leq FTL_k + FTM_k\cdot s_{FT_k} \quad \forall k \in K \tag{2.7}$$

Within the time window, crew labour and end-of-time-window location constraints are easily implemented within a parallel time-space network approach. During the pre-processing phase, the origin and sink nodes are determined, and relevant OF penalties are assigned depending on how strict the sink node penalty is.

3

Data set

Five separate datasets were required to implement the SDSS: the flight schedule, the passenger data, the disruption data, the crew schedule, and the reserve crew schedule. This chapter will present details on how each of them was acquired and processed, and what kind of information it provides. The entirety of the data is tied to the US flight schedule of Delta Airlines in the first quarter of 2015. Delta Airlines has the world's second largest flight network, operating 26 different aircraft types to 150 US destinations, with an average of 2400 daily flights. As the airline recovery problem is NP-hard, difficulties meeting AOCC-defined solution time limits are highlighted best for large-scale networks. The following sections describe each of the five datasets acquired and generated for use within the SDSS.

3.1. Flight & Disruption Data

The flight and passenger schedules, as well as the disruption sets were acquired from the work of Hassan (2018). This flight schedule data was obtained from the United States Bureau of Transportation Statistics online public database. The 2015 U.S. flight schedules of Delta Airlines were isolated and used for the entirety of model development and testing. The flight schedules were supplemented with passenger itineraries with data from Barnhart et al. (2014). The data contained the number of passengers on different O-D pairs, and STD and STA of different flights. Using the flight data, around 90% of passenger statistics were matched to their respective flights. For flights where no matching passenger statistics were identified, the average load factor on that particular O-D pair was used. The flight and passenger data used contains properties as shown in Table 3.1. Note that, in the initial dataset, per-flight information on delays and their cause was available.

Table 3.1: List of information provided by flight schedule. Adapted from Hassan (2018)

Property	Description
Flight Number	Delta Airlines flight number, in the form of DL1111. Note that flight number is not unique.
Tail Number	Unique tail number of aircraft scheduled to operate flight.
Origin Airport	Flight origin airport, as identified by three letter IATA code.
Destination Airport	Flight destination airport, as identified by three letter IATA code.
Scheduled Time of Departure (STD)	Flight scheduled departure time as measured in Coordinated Universal Time (UTC).
Scheduled Time of Arrival (STA)	Flight scheduled arrival time as measured in Coordinated Universal Time (UTC).
Passengers Economy Class	Number of economy class passengers booked on flight.
Passengers Business Class	Number of business class passengers booked on flight.

The delay information provided by the original Delta Airlines flight schedule was used to create a set of dis-

ruptions to be resolved within the SDSS. This set of disruptions was obtained from Hassan (2018). Though disruption duration and cause were present, the information regarding when the AOCC was made aware of the disruptions (time found out, TFO) was artificially generated. Each delay contains six properties, as shown in Table 3.2.

Table 3.2: List of information provided by disruption set.

Property	Description
Flight Number	Flight number of the disrupted flight.
Tail Number	Tail number of the disrupted flight.
Type	Type of disruption: delay or aircraft unavailability.
Cause	Disruption cause: NAS, weather, airline delay.
Duration	Disruption duration in minutes.
Origin Airport	Origin airport of disrupted flight.

3.2. Initial Crew Schedule

The difficulties in long-term crew scheduling due to uncertainty and regulation-induced complexity has led to proprietary crew scheduling models for each airline, all of which remain a confidential part of their business. To establish an initial cockpit crew schedule to recover, therefore, one must be generated. The scheduling of cockpit crew in particular is subject to stringent regulations due to possibilities of fatigue deteriorating crew performance capability. Within the US, baseline regulations which by law must never be broken are set out by the Federal Aviation Administration (2011). Each airline can then choose to extend these regulations with stricter, airline-specific ones, which are not legally binding. Delta's crew regulations were obtained from the publicly available crew scheduling handbook (Delta Airlines (2020)). These regulations contain the most common cockpit crew regulations respected by Delta Airlines in crew scheduling and recovery within the domestic U.S. flight network. These were used to develop a crew schedule generation algorithm, as well as to allow flexibility in recovery options when augmented operations (disrupted schedules) occur. The main crew regulations considered within the SDSS are:

- The maximum allowable scheduled length of a duty period ranges from 9 to 14 hours depending on number of scheduled flights and duty start time.
- The maximum allowable duty period length in the case of augmented operations can be increased to up to 17 hours depending on the type of aircraft and original start time of duty.
- The maximum allowable scheduled flight time within a duty period ranges from 8 to 9 hours depending on duty start time.
- The maximum allowable flight time may be extended to 13 hours for augmented operations.
- Crew may not be scheduled to more than 60 duty hours in any consecutive 168-hour (7-day) period.
- Crew may not be scheduled to more than 190 duty hours in any consecutive 672-hour (28-day) period.
- All crew must have a consecutive 10-hour rest between any two duties.
- All crew must have a consecutive 30-hour rest period within any consecutive 168-hour (7-day) period.

A greedy forward-heuristic approach is used to generate crew duties. Starting with the first flight in the schedule, the algorithm checks the possibility of adding the flight to an existing duty. If the flight cannot be added to any existing duty due to violation of labour constraints, a new duty is generated, and the next flight is evaluated. When a compatible duty is found for a flight, the flight is immediately added to it. The duties are sorted in order of ground time at the flight origin airport prior to departure, meaning that a crew that has arrived at the flight origin airport two hours prior to departure will always have priority over one that arrived 30 minutes prior to departure. In this way, unnecessarily tight connections or large ground times are avoided. A schematic representation of the duty generation algorithm is present in Figure 3.1.

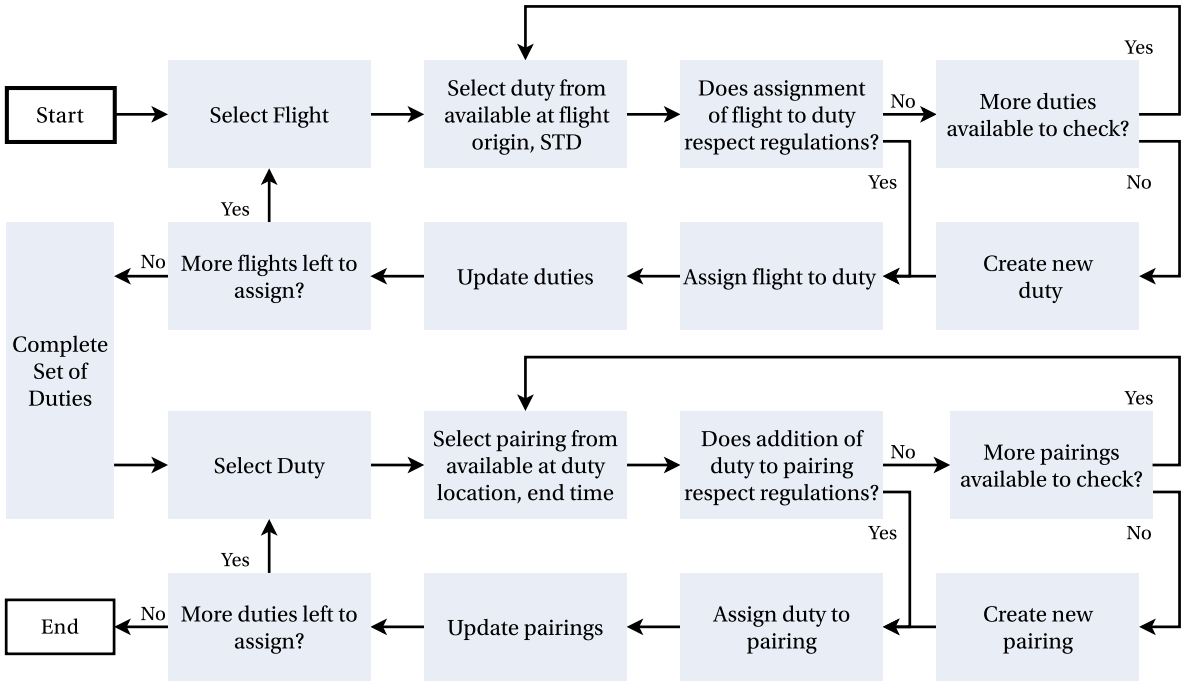


Figure 3.1: Flowchart of the full crew generation algorithm.

Upon completion of the duty generation, the pairing generation algorithm is called, as shown in the lower half of Figure 3.1. The algorithm attempts to add each duty to an existing pairing while respecting regulations. If no compatible pairing is found, one is generated and the duty is assigned to it. Once all the pairings are generated, the initial crew schedule is complete. The crew schedule is written to the flight schedule on a per-flight basis, and supplements the flight information with that present in Table 3.3.

Table 3.3: List of information provided by scheduled crew data.

Property	Description
Crew Number	Unique crew number used as an identifier, e.g. 111.
End Airport	The scheduled end-of-duty airport for the crew.
Flight Time	Total flight time crew has accrued after operating flight.
Duty Time	Total duty time crew has accrued after operating flight.
Maximum Flight Time	Maximum allowable flight time per regulations.
Maximum Duty Time	Maximum allowable duty time per regulations.

3.3. Reserve Crew Schedule

In addition to the regular crew schedule, airlines also assign reserve crew to prevent cancellations and increase the robustness of the flight schedule. Though many modern airlines use mixed-duty schedules where a crew is scheduled for regular operations and may have a reserve duty within the schedule, this thesis paper assumes only the presence of separate scheduled and reserve crew. Reserve crew duties are those that consider operation of flights in the case of absence of scheduled crew. The role of reserve crew is crucial in maintaining feasible flight schedules for airlines and their assignment is not trivial. A reserve crew schedule is generated based on the work of Bayliss et al. (2012). Namely, if the probability of scheduled crew being unavailable for a flight is known for the entire schedule, a fixed number of reserve crew R can be assigned in a way that minimises the total probability of crew unavailability over the entire schedule. The reserve crew generation utilises the following parameters:

P - Vector containing the initial probabilities of scheduled crew absence for each departure
P' - Vector containing the updated probabilities of crew absence after reserve assignment
X - Vector containing the start times for each of the assigned reserve crew duties
L - Matrix containing flights each assigned reserve crew can directly cover
R - Number of reserve crew available for scheduling
N - Number of flights in the schedule

The evaluation of the quality of a reserve crew schedule can be evaluated by the sum of vector P', that is the total probability that a crew will be missing for a flight over the entire schedule. Minimising this objective has experimentally proven to provide the best results in practice. A MIP-based formulation of this problem would be difficult to solve within the entire flight network, as, given a fixed number of reserve crew R and fixed number of flights N, the solution space scales with Equation 3.1:

$$\textit{Number of feasible solutions} = \frac{N!}{R!(N-R)!} \tag{3.1}$$

A greedy backwards heuristic approach is therefore used to evaluate the impact of reserve crew placement on the entire network. This approach is shown to obtain results within 0.8% of the optimum. For reference, the results obtained by the worst performing heuristic, the uniform distribution of reserve crews based on number of flights, obtain a 13% higher objective function value. The algorithm is initiated with the assumption that every flight has an associated reserve crew duty that starts at its STD, i.e. with $R' = N$. For each flight, the maximum static crew duty period is used as a baseline for determining which future flights operated by the same aircraft family and originating from the same airport can also be flown by this reserve crew, assuming it is not used to replace scheduled crew at its 'initiation' flight. These flights are contained within the matrix L. Let Pr denote the probability of having a reserve crew available and Pm denote the probability of a flight missing a scheduled crew. For a given reserve duty's first flight, the probability of reserve crew presence Pr is 1. The probability of this reserve crew being present for the next flight is therefore $1 - Pm$. The cumulative probabilities of presence of reserve crews for a flight are determined under the assumption that each probability of crew unavailability is an independent event. Given a fixed number of flights N, an initial number of reserve crews $R' = N$, and a fixed-number of reserve crews R, the algorithm evaluated the removal of a single reserve crew on the objective function. The reserve crew associated with the smallest increase in the sum of P' is removed, after which the remaining crews are reevaluated. The algorithm stops once the desired number of reserve crew R is reached.

Reserve crew is assumed to be generated prior to knowledge of the disruptions faced during the period present in the dataset. The evaluation of the model in practice during development could lead to designing a reserve crew schedule that fits particularly well to the disruption faced. For this reason, theoretical performance as evaluated by the sum of P' is used as the primary metric. The value of R was set to 20% of the number of duties in the schedule. This number is realistic relative to the corresponding real-life case, as some airlines have up to 30% of their crew scheduled as reserve for a day. The theoretical performance of the reserve crew generation algorithm did not improve significantly with the use of a higher percentage.

The biggest difficulty in the application of the proposed formulation is initialising the vector P. It is assumed that scheduled crew absence for a certain flight can only be caused by the cancellation of the crew's previous flight, or the delay of the previous flight to the point where the flight considered is missed. Therefore, the probability of a crew missing a flight is based on two mutually exclusive events: (1) the probability of previous flight cancellation Pc, and (2) the probability of previous flight delay past the maximum allowable delay $Pd \geq d_{max}$. In this case, the only information required to initialise the vector P are the values of Pc and Pd for all scheduled flights. To obtain these probabilities, a random forest classifier was trained with Delta Airlines OTP data for the period 2012-2014. This classifier is used exclusively for the generation of disruption probabilities and is not called as part of the SDSS. The features used to train the model are determined per flight and are:

- `AC_family` - aircraft family of the assigned aircraft
- `quarter` - quarter of the year for STD
- `day_month` - day of the month for STD
- `day_week` - day of the week for STD
- `origin` - origin airport of the flight
- `destination` - destination airport of the flight
- `std` - STD of the flight
- `sta` - STA of the flight

The target array contains 37 boolean columns. The first is the classification of whether the flight was cancelled or not. The remaining target columns represent the classification of whether the flight experienced an arrival delay greater than or equal to d, $Pd \geq d$, where $d \in \{10, 20, 30...360\}$. A schematic view of how the probability of a scheduled crew being missing is present in figure Figure 3.2. A 20 minute delay or cancellation of Flight 1 would result in the scheduled crew being unavailable for Flight 2. If we assume the probability of cancellation Pc at 0.05, and the probability of a delay of 20 or more minutes $Pd \geq 20$ at 0.02, then the combined probability of the scheduled crew being missing Pm is the sum of the two i.e. 0.07. The vector P is constructed by evaluating these values over the entire schedule.

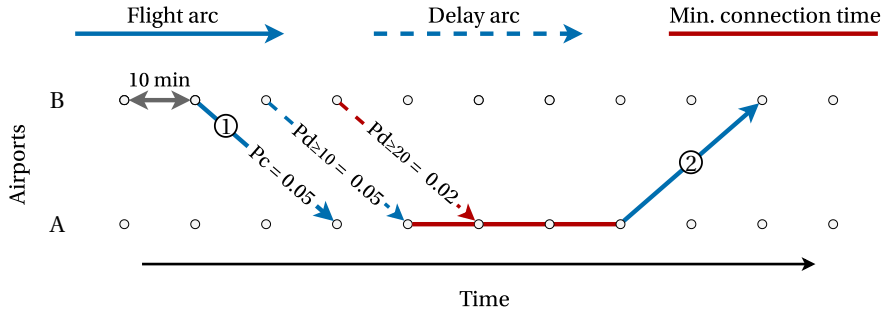


Figure 3.2: Influence of delay and cancellation probabilities on crew absence.

Once the reserve duties are created, they are combined into pairings in a similar way to the pairing generation of the initial schedule. Namely, duties are ordered by start times and grouped together if the combination respects the crew schedule regulations set out by Delta. Note that for reserve pairings, a maximum of five duties are assigned per crew. The information provided by the reserve crew schedule is present in Table 3.4

Table 3.4: List of information provided by reserve crew data.

Property	Description
Reserve Number	Unique crew number used as an identifier, e.g. R111.
Base	The scheduled reserve base for the crew.
Aircraft Family	Aircraft family operated by crew.
Duties	Number of duties scheduled for crew.
Start Time	Start time for each duty scheduled for crew.
End Time	End time for each duty scheduled for crew.

4

Model Verification

This chapter presents the ways the verification of the crew recovery formulation of the SDSS. The main focus is on ensuring correct functionality of the mathematical formulation as solved by the MILP solver. This was done by evaluating the model solutions on artificially generated disruptions where a solution is easily identified as optimal by the user. The generation of the initial and reserve crew schedules is also evaluated in this chapter. Section 4.1 provides a look into the main aspects of the SDSS crew recovery model to be verified, as well as a few generated examples. Section 4.2 describes the ways the generated crew schedules were verified.

4.1. Crew Model Verification

To verify that the model is able to make the right decisions, each of its recovery options was tested to ensure it functions as expected. Several artificial scenarios were used to evaluate the ability of the model to perform crew swaps, use reserve crew, deadhead crew, and cancel flights. These were evaluated by hand and compared to the results of the optimizer. The optimizer can use a combination of four actions to recover the crew schedule: (1) swap crew schedules, (2) use reserve crew, (3) deadhead crew, and (4) cancel flights. To minimise unwanted behaviour, appropriate penalties have been implemented for flight cancellation, utilisation of reserve crew, and absence of crew at their sink nodes. Though flight time penalties are also implemented, testing indicated that flight time is very rarely a limiting factor in the application of crew recovery. The parameters used for the evaluation of the artificial disruption scenarios are present in Table 4.1. In these scenarios, the hourly operating cost, C_{op}, and the flight cancellation cost, C_{cx}, are assumed to be constant.

Table 4.1: Parameters used for the crew recovery verification.

Parameter	Value
C_{OP}	\$500
C_{OC}	\$2000
C_{CX}	\$250000
C_{SV}	\$50000
C_{SV_R}	\$10000
C_{DH}	\$200

These penalties are applied on a set of six artificial disruption scenarios generated with the aim of testing the functionality of the MILP formulation of the crew recovery stage. The following table contains information on each scenario, the number of scheduled and reserve crew present, as well as the number of delays and cancellation present as disruptions. Throughout the following scenario evaluations, note that the graphic representations consist of relatively small time-space networks. In each plot, the line style of each flight arc represents which crew was scheduled to operate the flight. The line colour represents which crew operated the flight, or whether it was cancelled. Though results per scenario are presented in a single time-space plot, each scheduled or reserve crew operates within a separate time-space network.

Table 4.2: Overview of disruption scenarios tested.

Scenario	Crews	Reserve Crews	Delays	Cancellations
D1	1	0	1	0
D2	2	0	1	0
D3	2	0	0	1
D4	1	1	1	0
D5	2	1	1	0
D6	1	0	1	0

Scenario D1 is graphically represented in Figure 4.1. Turnaround times are included in the length of the flight arcs and are assumed at 20 minutes for all flights. Crew I is scheduled to Fly Flights 1 and 2. A 20-minute delay of Flight 1, represented as Flight arc 1d in the figure, breaks the schedule of Crew I. The sink node for Crew I is set at the last node within the time window at Airport A. The solver is called to make a decision on how to restore the schedule.

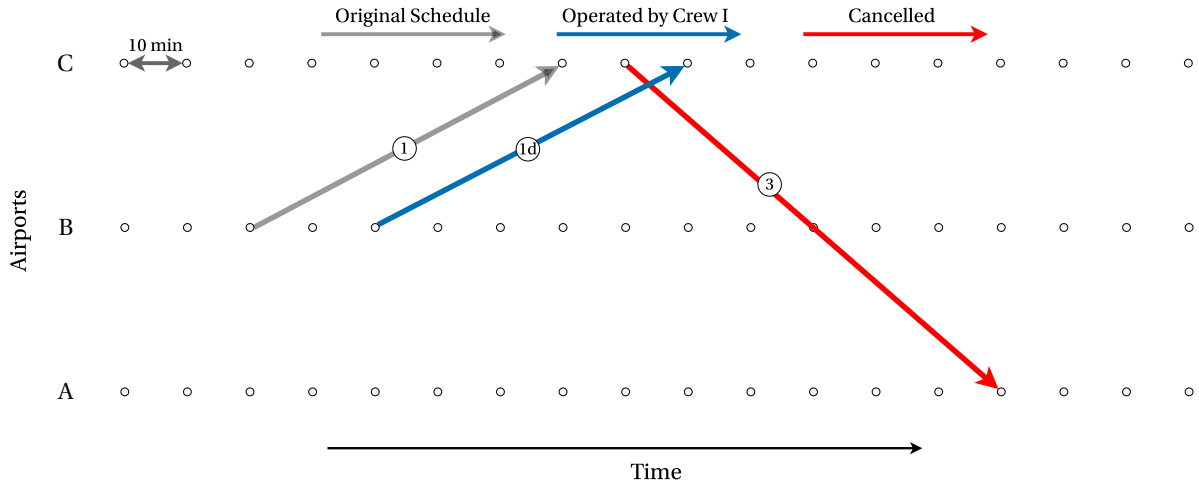


Figure 4.1: Artificial disruption scenario D1.

As the figure shows, the solver chooses to fly Flight 1 and cancel Flight 2. In this case, this is clearly the optimal solution as flight delays are not considered in the crew stage. The cancellation of Flight 2 also results in Crew I missing their scheduled sink node at Airport A. The total objective function value is \$300250, \$250 of which is for the 30-minute flight, and \$300000 for penalties incurred for the missed sink node and flight cancellation. A breakdown of these costs is present in Table 4.3. This table also contains the costs of other disruption scenarios considered in this section.

Table 4.3: Overview of disruption scenario solutions and their respective costs.

Scenario	Objective	Operating	Swap	Reserve	Sink	Cancellation	Deadhead	Disruption
D1	\$300250	\$250	\$0	\$0	\$50000	\$250000	\$0	\$300000
D2	\$5250	\$1250	\$4000	\$0	\$0	\$0	\$0	\$4000
D3	\$1200	\$1000	\$0	\$0	\$0	\$0	\$200	\$200
D4	\$60583	\$583	\$1000	\$10000	\$50000	\$0	\$0	\$61000
D5	\$52117	\$917	\$1000	\$0	\$50000	\$0	\$200	\$51200
D6	\$583	\$583	\$0	\$0	\$0	\$0	\$0	\$0

In Scenario D2, the same delay from Scenario D1 is considered. A second crew, Crew II, operating two additional flights, 3 and 4, is also present in the network. Scenario D2 is graphically represented in Figure 4.2. The solid lines represent flight arcs scheduled for Crew I, while dashed lines represent arcs scheduled for Crew II. Upon having its schedule broken by a 20-minute delay in Flight 1, Crew I cannot operate Flight II. Crew II, however, can, and is chosen by the solver to operate Flight II. Flight 4 is left without a crew, and Crew I is assigned to it. In this way, by swapping the scheduled crew of two flights, no cancellations are required and

both crews end up at their sink node locations. The objective function for this scenario is $5250, of which $1250 are made up of 150 minutes of operating costs. The remaining $4000 in the objective function value comes from the two crew swaps performed for the two flights. These are incurred for every flight operated by a non scheduled crew. In this scenario there are two. These two examples show correct decision making by the solver for crew swaps and flight cancellations. One of the benefits of using a time-space network based approach is the addition of the option to deadhead crew. The following scenario, D3, attempts to have the model deadhead crew.

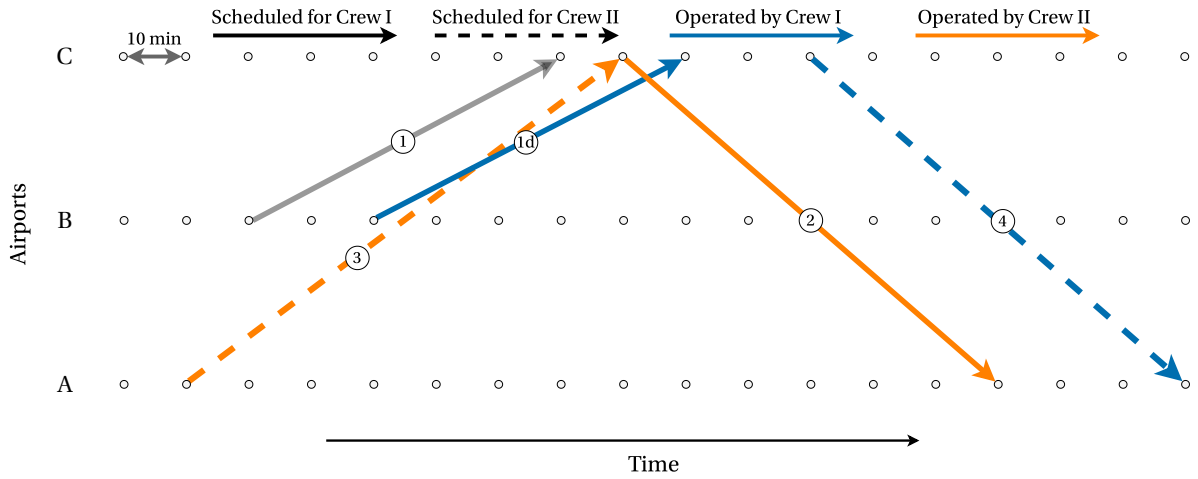


Figure 4.2: Artificial disruption scenario D2.

In Scenario D3, the same set of disruptions and crew as that of D2 is considered. However, it is assumed that a cancellation of Flight 2 was a result of the aircraft recovery stage. This means that operating Flight 4 is still an option for Crew I, but also for Crew II. This scenario is shown in Figure 4.3. The solver selects Crew II to operate Flight 4, and deadheads Crew I to airport A on the same flight. In this way, no swap costs are incurred, and Crew I is still able to reach its sink node. The objective function value for this scenario is $1117. This consists of $917 of operating costs for 110 minutes of flight time, and a $200 cost for deadheading a single crew-pair. Despite the disruptions experienced during this scenario being seemingly worse, the disruption cost is lower than that of scenario D2. As the cancellation of flight 2 comes as a disruption to the schedule and not as a decision, the cancellation costs of Flight 2 have already been included in the aircraft recovery phase. The costs present are only those that result from the crew recovery.

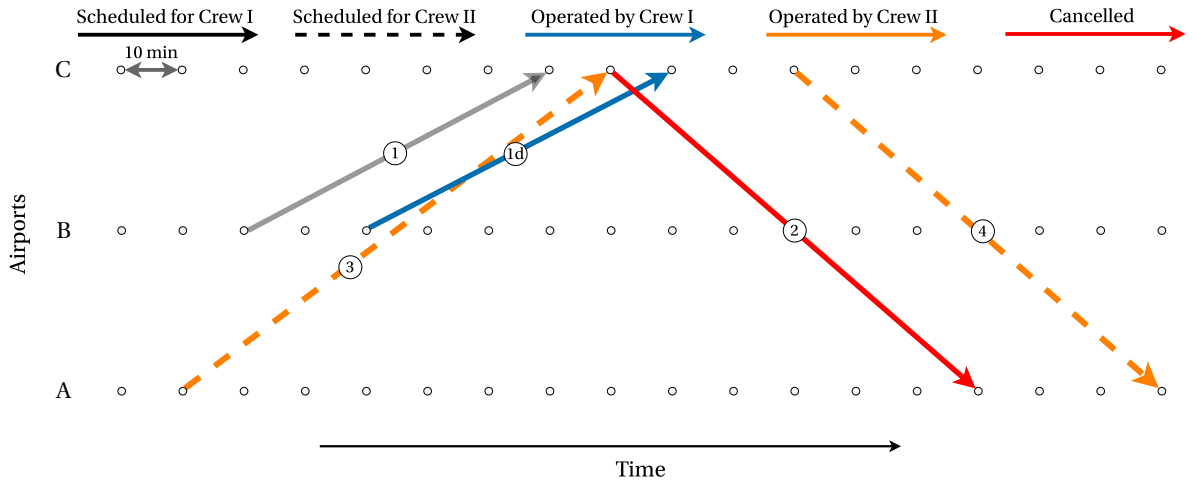


Figure 4.3: Artificial disruption scenario D3.

Disruption scenarios D4 and D5 aim at verifying the correct use of reserve crew and application of relevant penalties. Namely, the use of reserve crew is penalised only if the reserve crew does not return back to base during its duty period. In this sense, the reserve crew use is penalised in the same way a sink-node penalty is applied. Scenario D4 assumes the presence of the initial disruption scenario of D1, i.e. a single crew whose schedule is broken due to a 20-minute delay in Flight 1. A single reserve crew is available at Airport C at the start of the time window. This is shown in Figure 4.4. Here, the solver decides to operate Flight 2 with the only available crew, R1. Crew I remains at Airport C, and crew R1 remains at Airport A, both missing their sink nodes. The objective function value of this solution is $61583. Flight costs amount to $583 of that, $50000 is incurred as a penalty for violating the sink node constraint of Crew I, and an additional $10000 is incurred for using reserve crew but not returning them back to base. As Flight 2 is not operated by scheduled crew, an additional swap penalty of $1000 is applied.

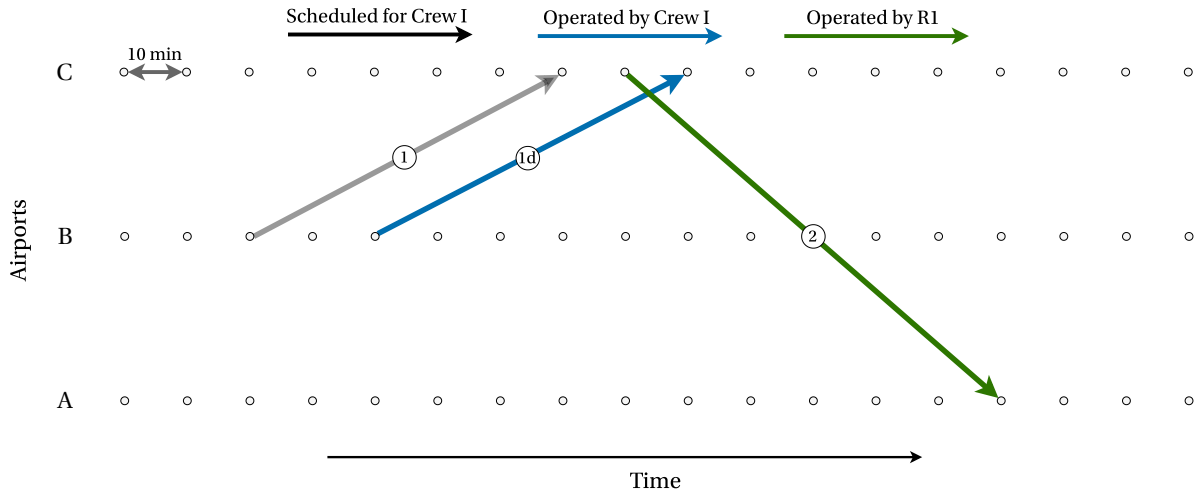


Figure 4.4: Artificial disruption scenario D4.

Scenario D5 aims to demonstrate how the model allows freedom in the use of reserve crew in the case that they do not violate their sink node constraint. In essence, it is identical to Scenario D4, with the addition of a flight operated by Crew II from Airport A to Airport C. This is shown in Figure 4.5. The addition of this flight enables a deadhead of crew R1 back to airport C. As a a result of this, the objective function value of this disruption scenario is $52116.7.

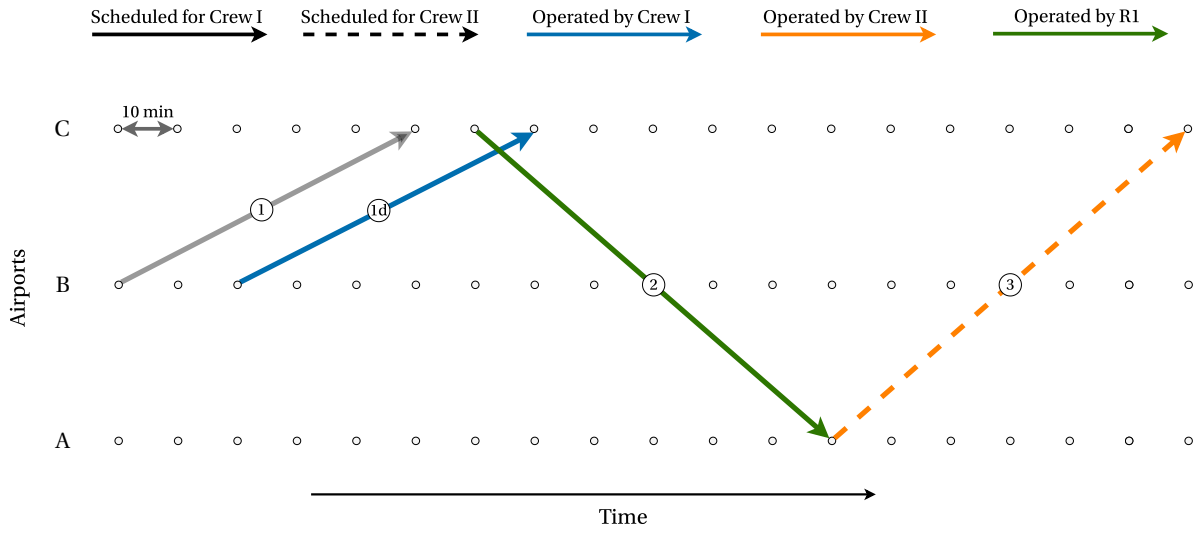


Figure 4.5: Artificial disruption scenario D5.

Disruption costs are $51200, where $50000 comes from Crew I violating its sink node constraint, $1000 comes from the crew swap on Flight 2, and $200 comes from deadheading crew R1 back to base via Flight 3. The remaining $916.7 consists of the operating costs for the three flights. In this application of the reserve crew penalty, the reserve crew is deadheaded back to base and is able to continue its duty. No penalty is therefore incurred as a result of its use. The final disruption scenario, D6, aims to verify the trivial solution check used

within the SDSS formulation. As flight delays aren't considered as a recovery option, the SDSS can check for broken crew duties during preprocessing. An example where a disruption occurs but no crew duties are broken is present in Figure 4.6. In this scenario, a 20-minute delay of Flight 1 does not break the crew schedule. This means that the crew schedule does not need to change, and the solver outputs a disruption cost of $0, with an objective function value of $583, consisting of only operating cost. In the SDSS, these types of disruptions are identified during preprocessing. The solver is called using only the crews of the disrupted flights, and the crew schedule is updated with the trivial solution to consider the delay(s) present. With scenario D6, all recovery options of the crew recovery mathematical formulation are verified.

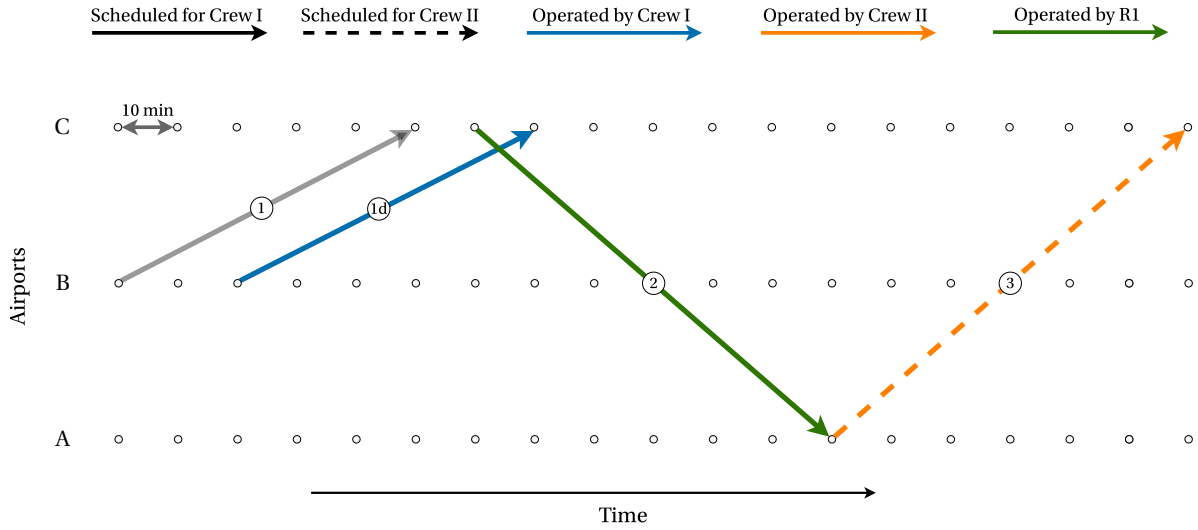


Figure 4.6: Artificial disruption scenario D6.

4.2. Crew Schedule Verification

The crew schedules generated based on the initial flight schedule must also be verified to ensure correct functionality. Though a simple algorithm was made to verify adherence of each crew pairing to regulations, the functionality of the crew schedule was also verified within the SDSS. By initiating the SDSS crew recovery stage for all disruptions within the disruption set and removing them prior to creating the time-space networks and calling the solver, the problem passed on to the solver is one where only the initial schedule is present, and no duties should be broken. For all the runs tested, the solver returned only the trivial solution, indicating the crew schedule functions as expected. Note that this test was not performed with the use of the broken crew duty check the SDSS applies to identify trivial runs, but rather by calling the solver and having it obtain a solution that considers all crews.

The aforementioned test also considers the presence of reserve crews. To verify that the generated reserve crews are a useful asset to the SDSS formulation, the disruptions experienced by the crew stage were solved twice: with and without the consideration of reserve crew in the solution. The results of this evaluation are present in the following table.

Table 4.4: Summary of non-trivial optimizer results for disruption scenarios with and without reserve crew.

	Sol.T	Crew	Disr.C	Canx	SNV	Swap	RCU	U60
Without Reserves	78.8	232.1	113960	78	713	3104	0	213
With Reserves	65.6	261.5	65860	24	601	1599	230	269

Clearly, the use of reserve crew is a valuable asset in crew recovery. The use of reserve crew reduced the number of cancellations from 78 to 24, and the number of sink node violations from 713 to 601. Despite adding more crew to consider, the computational time of the reserve crew-free solutions is over 13 seconds higher on average. This is likely due to the higher number of cancellations, swaps, and sink node violations. As the solver must weigh multiple options with a large objective function impact instead of being able to make a lower-impact decision (use of reserve crew), the computational time increases. The use of reserve crew reduces the average disruption cost by over 40%, enabling the SDSS to obtain much higher quality solutions. These results also show why assuming reserve crew are available at all times is not valid. Though the reserve crew schedule present in this research has its limitations, the realistic consideration of reserve crew with respect to the schedule is a valuable asset of the SDSS.

5

Machine Learning Classifier

This chapter provides more detail on the steps taken to develop the machine learning classifier used for selection of crew. Machine learning is a specific application of artificial intelligence aiming to generate a system with the ability to learn from experience without being explicitly programmed to do so. It is generally divided into three parts: (1) supervised learning, (2) unsupervised learning, and (3) reinforcement learning. A supervised learning approach assumes the presence of labelled input and output data. The process initiates with a function aiming to correlate the input to the output by processing the input data, observing the output, and adjusting the corresponding function weights. Unsupervised learning considers machine learning applied to datasets with no labels or target output. This is useful when attempting to cluster elements of large datasets with no apparent relationship. Finally, reinforcement learning utilises a software agent interacting with an environment. Through a series of decisions, each of which results in a user-defined reward, the agent aims to learn the best steps to take such that the sum of future rewards is maximised. As the data used for this research is fully labelled, supervised learning was used for the entirety of the selection algorithm development.

The data available for training consists of a set of disruptions, set of crews, and a resulting solution, or selection of crews whose schedules are changed as a result of the disruption. The most logical implementation of machine learning for identifying the most likely crews within the optimal solution is that of binary classification. Crews likely to be used in the optimal solution are classified by the algorithm as 'True', while those classified as 'False' are discarded. On average, around 1.2% of crew schedules are changed in the optimal solution. However, this number includes crew that is disrupted directly which can be identified easily without the use of a machine learning classifier. The main interest in using machine learning to identify the correct crew selection comes from non-disrupted crew that are used to recover a certain disruption. Only 0.43% of crews in the data are of such a type. This dataset is therefore highly imbalanced, with the negative class making up the majority. A summary of the process of algorithm development is present in Figure 5.1.

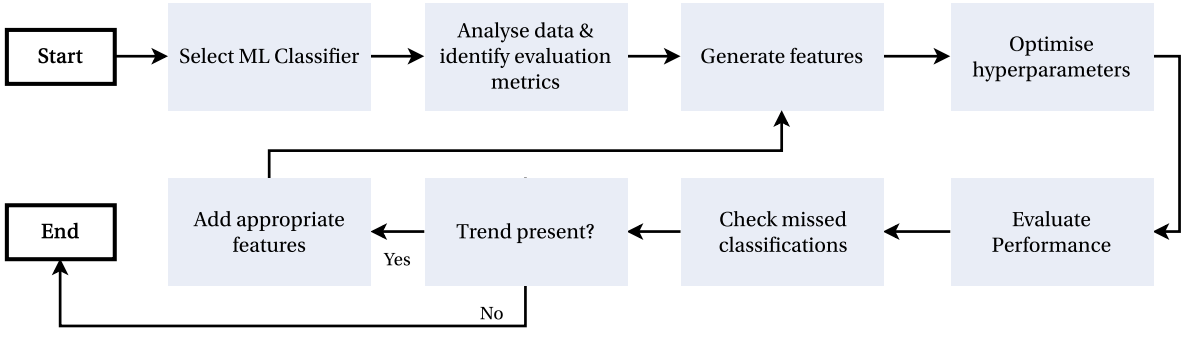


Figure 5.1: Flowchart of the classifier development process.

Starting with the selection of a suitable classifier, the data used is analysed to identify relevant evaluation metrics. An initial set of features relating to disruption location, disruption time, disruption duration, and

disruption type were used to initiate the process of feature engineering. This feature set is optimised for hyperparameters and its classification performance is evaluated on test data. Using this test data, wrongly classified instances of the positive class were identified, and the solutions containing these instances were manually examined. When a trend or feature was identified that was not present in the feature set, it was added and the process was reiterated. The machine learning classifier development completes when no clear trends are present. The following sections will elaborate on the process of algorithm selection, model evaluation and training, and the final classifier properties.

5.1. Algorithm Selection

As a single missing crew in a selection could lead to non-optimal results, the selection of a good binary classification algorithm greatly influences the final results of the work. The aircraft recovery stage of the SDSS utilises a random forest classifier to select relevant aircraft. During the qualitative analysis of machine learning classifiers done by Hassan (2018), however, an effective tie between boosted trees and random forests was identified in terms of potential for use with the aircraft recovery problem. Olson et al. (2017) compared the performance of 13 commonly-used classifiers available within the scikit-learn library. A 10-fold cross validation balanced accuracy approach was used to rank these across a set of 165 different classification problems. The ranking is present in Figure 5.2.

Though relatively recent literature still incorporates RF- and SVM-based classifiers, tree boosting has emerged as a versatile classification algorithm that is able to cope with a large variety of data. Chen and Guestrin (2016) originally proposed the concept of extreme gradient boosting, a specific implementation of tree boosting, which was later made publicly available as XGBoost within the scikit-learn library. Some recent papers, such as those of Chang et al. (2019) and Zhang et al. (2021), rate the performance of XGBoost as better than that of Random Forests. However, a general consensus is that random forests generalise better for default settings. Because of this, hyperparameter optimization must be performed for a large number of iterations. Classifier performance is also heavily dependent on the problem structure. That is, a classifier that generalises well to a variety of problems does not guarantee the best performance on a given problem type. The performance of a specific machine learning classifier on a specific problem type is not easy to predict. As random forests have already been applied with success on the aircraft recovery problem by Hassan (2018), and due to the success of XGBoost within many recent machine learning competitions[1] and its promising potential in recent literature, it was selected as the main machine learning classifier to be used in this research.

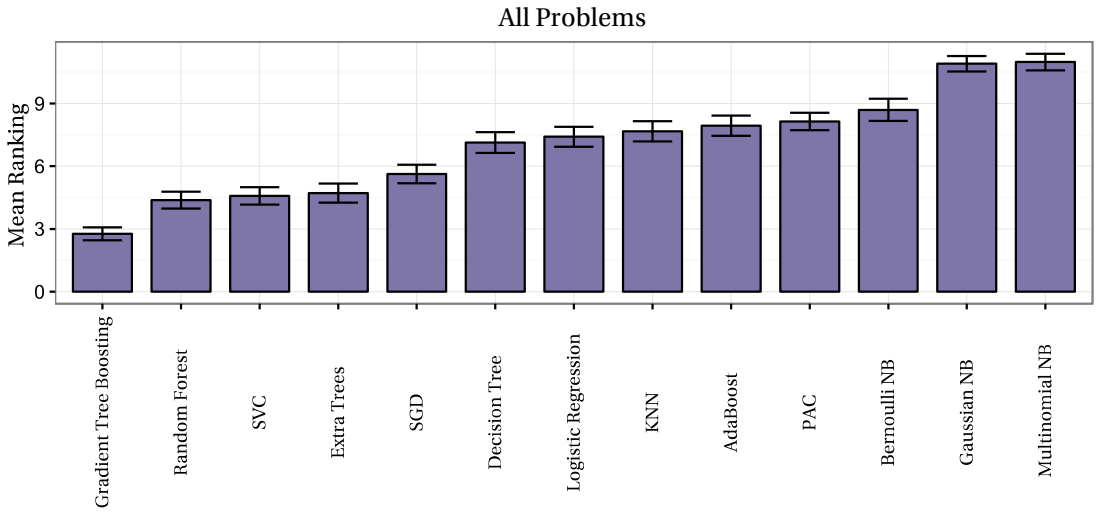


Figure 5.2: Average ranking of ML classification algorithms over 165 problems. Adapted from Olson et al. (2017).

[1] XGBoost: Machine Learning Challenge Winning Solutions - https://github.com/dmlc/xgboost/tree/master/demo

5.2. Model Evaluation & Training

Evaluation of binary classification algorithms is most commonly based on threshold metrics according to Hossin and Sulaiman (2015). In binary classification, a classifier namely outputs a probability that a certain instance is of the positive class. Given a certain probability threshold, all instances with a probability value greater than or equal to the threshold are classified as the positive class, while the rest are classified as the negative class. In this way, some important threshold metrics for classification can be extracted, as shown in the confusion matrix of Figure 5.3.

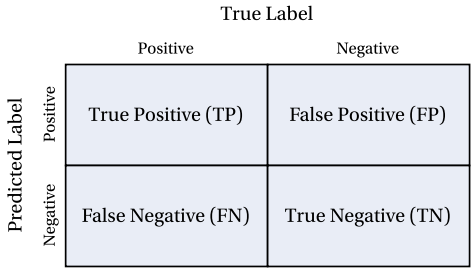


Figure 5.3: Confusion matrix for binary classification.

The four metrics presented in the above figure can be manipulated to provide more detail into how a classifier functions. Different metrics are valuable for different problem types, and the most commonly used ones are shown in Table 5.1. Though these are individually valuable, all are threshold dependent. When evaluating a classifier, performance across a range of thresholds provides better insight into the predictive capabilities of a classifier.

Table 5.1: General metrics for evaluation of classifier performance.

Metric	Formula	Description
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Correct classifications over total classifications.
Precision	$\frac{TP}{TP+FP}$	Correct positive classifications over total positive classifications.
Recall	$\frac{TP}{TP+FN}$	Correct positive classifications over total actual positive instances.
Specificity	$\frac{TN}{TN+FP}$	Correct negative classification over total actual negative instances.
Error rate	$\frac{FP+FN}{TP+TN+FP+FN}$	Incorrect classifications over total classifications.

Two common ways to do this are by using a receiver operating characteristic (ROC) curve or a precision-recall (PR) curve. The ROC curve shows a comparison of the true positive rate of a classifier compared to the false positive rate across a variety of probability thresholds. The PR curve shows a comparison between a classifier's precision and recall across a variety of probability thresholds. As the calculation of neither precision or recall is based on the number of true negatives, a PR curve is a useful tool in evaluating classifier performance where there is a relatively high presence of negative instances. By calculating the total area under the ROC and PR curves, classifier performance across a variety of thresholds can be compared with a single numerical value.

When using these metrics to compare performance between different classifiers, training and evaluation should ideally be performed on several datasets to avoid bias. When limited data is available, a good way to do this is via the use of k-fold cross validation. Namely, assuming k folds or equal splits of data, the classifier is trained on $k-1$ folds and evaluated on a single fold. This is repeated for each fold and the performance of the algorithm between the k folds is averaged to ensure generalisation. An overview of 3-fold cross validation used within a ML model evaluation is graphically represented in Figure 5.4. The content below the training data block represents the 3-fold train-test split of the training data. The content below the test data block represents the final data used for validation. This is used only in the final stage of model evaluation, i.e. only when the entirety of feature engineering and hyperparameter optimization is completed.

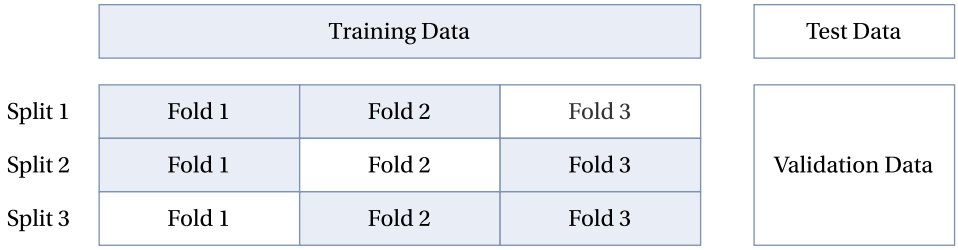


Figure 5.4: Schematic representation of 3-fold cross validation as part of a classifier evaluation process.

When training a classifier on imbalanced datasets, there is a risk of the classifier lacking sufficient information on the minority class to acquire useful knowledge. To avoid this, a common approach is to restructure the data by over or undersampling it to introduce class balance. Oversampling refers to the selective or random replication of the minority class to achieve class balance. Undersampling refers to the selective or random removal of the majority class to achieve class balance. The difference between the two is illustrated in Figure 5.5 and Figure 5.6.

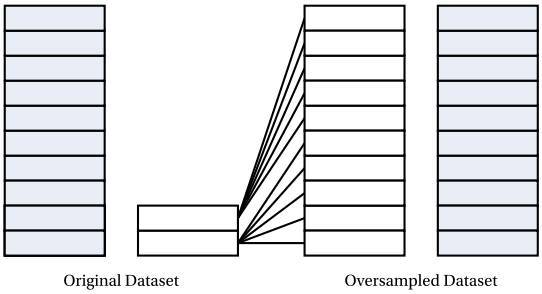


Figure 5.5: Schematic representation of oversampling to achieve class balance.

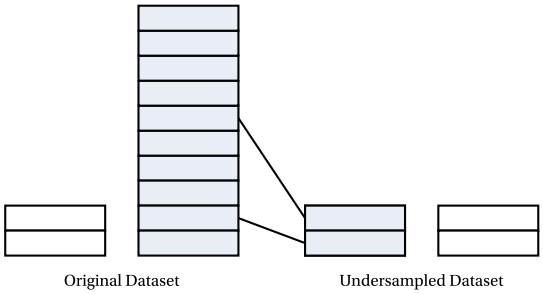


Figure 5.6: Schematic representation of undersampling to achieve class balance.

Though class balance is achieved upon the use of these methods, replication and removal of data risks addition of noise or removal of useful information. To avoid such issues, Chawla et al. (2002) propose an interpolation method called synthetic minority oversampling technique (SMOTE) that constructs artificial instances of the minority class based on information present in the dataset. When SMOTE is used, the artificial information is used in combination with real data to train the classifier, which is evaluated exclusively on real data. The use of cost-sensitive learning is another technique addressing class imbalance within a dataset. Unlike the aforementioned methods, cost-sensitive learning does not alter the dataset directly. Instead, a user-defined penalty is defined for classifying instances of the minority class. In this way, the classifier places a larger focus on learning about the minority class during training.

5.3. Hyperparameter Optimization

Several user-defined parameters must be set when initialising the training of the XGB classifier. These parameters define key classifier properties that ultimately affect classification performance. The interaction between these parameters varies for every feature set. Therefore, an evaluation of the interaction of these parameters is necessary in order to select their values. A simple way to do this is to use a grid search. For n hyperparameters, grid search would perform an exploration of an n-dimensional parameter space. With a range of values for each parameter to take on, the performance of all combinations of different parameter values is tested and the best combination is selected. This quickly becomes impractical in the case of large ranges of values per hyperparameter, as the computational time required to assess all possible combinations can be excessive. Furthermore, k-fold cross validation must be performed at each combination of hyperparameter values to ensure generalisation. For m hyperparameters, each with a set of possible values of length n, a k-fold cross-validated dataset would require the training of $m^n \cdot k$ separate algorithms. A more efficient alternative to grid search is Bayesian optimization, where a probabilistic model selects the best hyperpa-

rameter values, x^*, from a range of allowable values, X, and evaluates their performance given an objective function, $f(x)$. In equation form, this is represented by Equation 5.1

$$x^* = argmin_{x \in X} f(x) \tag{5.1}$$

The advantage of Bayesian optimisation is that, unlike grid search, it accounts for the entire history of hyperparameter evaluation and uses it to construct a *surrogate* model of the objective function. This model is used to estimate the impact of certain parameters on the given objective function by mapping hyperparameters to a probability of a score, i.e. obtaining $P(score|x)$. The surrogate function is initialised by evaluating a set of randomly selected hyperparameter combinations on the real objective function, after which the model is able to determine the next set of hyperparameters to evaluate via the use of a selection function. The most common criterion for the selection function is that of expected improvement, i.e. determining which combination of parameters is expected to yield the largest improvement in the objective function value compared to the current-best value, as formalised by Equation 5.2.

$$EI(x) = E(max(f(x) - \hat{f}, 0)) \tag{5.2}$$

By selecting the combination of hyperparameters with the best expected improvement, Bayesian optimisation is able to obtain near-optimal hyperparameter combinations in much fewer iterations than grid search (Snoek et al. (2012)).

5.4. Classifier Properties

This section contains information about the final properties and performance of the classifier. The iterative feature selection process described at the beginning of this chapter resulted in a total of 126 features, most of which are described in detail in Appendix A. The 20 most important features and their relative importance are shown in Figure 5.7.

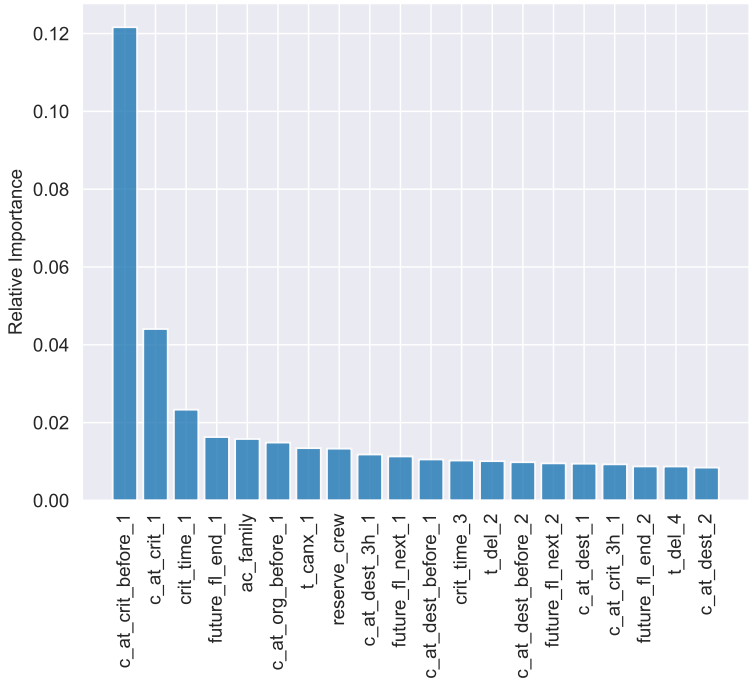


Figure 5.7: Top 20 most important features used in the XGB classifier.

The three most important features relate to the critical time and location. When a disruption such as a flight cancellation causes a crew to be unable to reach the origin airport of their next scheduled flight prior to its departure, this origin airport and STD are identified as the critical time and location. Candidate crew that are

scheduled to be located at the critical location at or prior to the critical time are therefore able to take over the flight whose scheduled crew is missing. The fourth feature on the list represents whether a candidate will fly to a disrupted crew's end-of-duty airport at or after the STD of the disrupted flight. Although this feature was designed to be used for deadhead decisions, it could also be valuable to the classifier for crew swap decisions. Generally, as a disruption scenario often contains a low number of disruptions, the most important features are those relating to the first three in the dataset.

Addressing class imbalance was performed exclusively using cost-sensitive learning, which outperformed oversampling, undersampling, and SMOTE. Using the area under the precision-recall curve as the objective function, Bayesian hyperparameter optimization was performed with 100 initialisation iterations and 1000 optimization iterations. 10-fold cross validation was used to compare results between iterations, and the final parameters and their optimised values are present in Table 5.2.

Table 5.2: Optimal hyperparameter values post-Bayesian optimisation.

Parameter	Description	Final Value
n_estimators	Total number of trees within the classifier.	329
max_depth	Maximum depth of a single tree.	44
gamma	Minimum loss reduction required to make a further partition on a leaf node of the tree.	0.791
learning_rate	Step size shrinkage used in update to prevent overfitting.	0.086

With these hyperparameter values, classifier performance across a number of probability thresholds was evaluated. The precision-recall curve of the final classifier as evaluated on the validation dataset used in the case study is present in Figure 5.8. For the same data, a comparison between recall and specificity is made across a variety of thresholds in Figure 5.9.

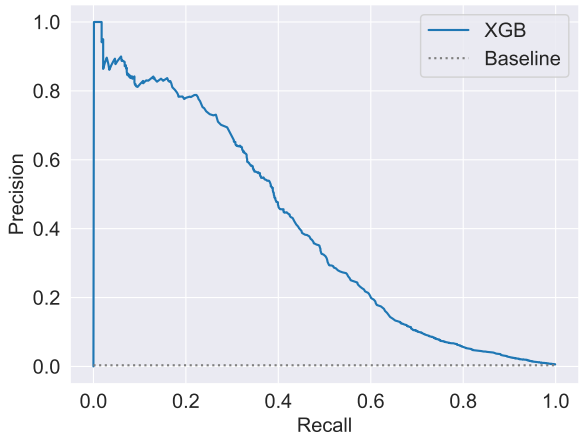


Figure 5.8: PR curve for the trained classifier on validation data.

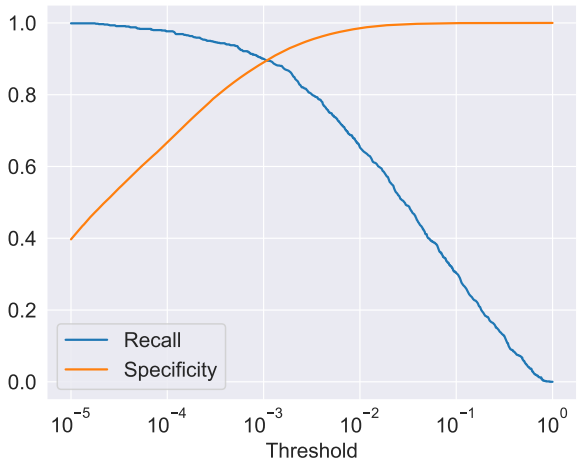


Figure 5.9: Recall and Specificity curves for varying probability thresholds on validation data.

At a specificity of around 0.4, the classifier is able to correctly identify 100% of crews used in the optimal solution. Though this indicates good model performance, it also illustrates the need for a further reduction in the number of selected crews. If, given a set of 300 crews with similar features, the classifier selects 250, the reduction provided by the selection will likely not contribute significantly to the reduction of solution time. To prevent this, several methods of using the classifier output are evaluated in a sensitivity analysis in the following chapter.

6

Sensitivity Analysis

Though the performance of the crew recovery model is satisfactory, model parameters can be tweaked to reduce the overall solution time or increase the solution quality. This chapter will present a sensitivity analysis done on the selection algorithm. There are two ways to use the aircraft recovery solution in the crew recovery stage: (1) by taking the schedule of the sub-network created by the aircraft recovery stage or (2) by taking the entire flight network and creating a separate sub-network for the crew stage only. The issue with the former is that, if the aircraft recovery selection uses only two aircraft to find a solution, the crew stage receives only the crew operating those aircraft. As a flight delay may cause crew unavailability due to duty time constraints, this approach may result in neglecting valuable recovery options provided by crew operating different flights. All crews are therefore be considered in the crew recovery stage prior to the initiation of the ML classifier. The selection algorithm used in the case study uses a classifier threshold with a 99% recall rate. Any selection made with the threshold that contains less than 100 crews is accepted. If a selection contains 100 or more crews, the selection algorithm reduces the selection to the maximum of half the original crew size and 100. The maximum size of the selection is shown in Figure 6.1. 100 crews was selected as a sufficient reduction as disruption instances containing 100 crews or less were consistently solved in under 20 seconds.

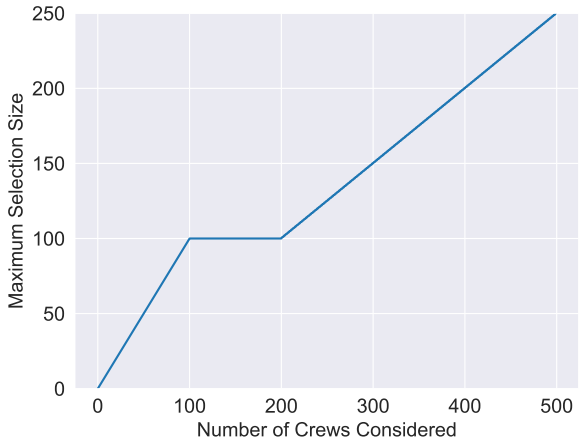


Figure 6.1: Maximum size of final selection relative to initial number of crews considered.

To compare the performance of different selection algorithms, the case study was evaluated via the defined KPIs for multiple ways of selection. The performance was evaluated on a selection based on the top 20%, 30%, 40%, 50%, and 60% of crews. This was supplemented by a classifier threshold-only based selection, where the selection was done by taking all crews above a certain threshold for a 99%, 95% and 90% recall rate of the classifier. The selections resulted in the results presented in Table 6.1 for the instances tested on the case study disruption set.

Table 6.1: Summary of results for all selection algorithms tested.

	T	CC	DC	OS	CX	SNV	SWP	RCU	U60
Optimizer	65.6	261.5	65860	537	24	601	1599	230	269
SDSS	26.7	138.7	68958	479	25	640	1540	233	520
T60%	31.6	156.3	67649	491	25	616	1571	241	490
T50%	25.6	130.0	69135	475	25	642	1554	238	521
T40%	20.0	103.8	70057	448	28	646	1505	233	533
T30%	15.0	77.7	72745	437	30	663	1541	228	537
T20%	9.8	51.9	77513	402	35	707	1461	220	537
R99%	55.2	247.8	66444	535	25	605	1596	244	305
R95%	41.4	185.5	68670	517	27	617	1533	234	386
R90%	32.5	148	70227	497	29	631	1515	233	430

The results imply that the dynamic selection algorithm has the best balance of solution time and quality. Though R99% selection results in a 92.5% optimality rate, there are five additional cancellations as a result of the sub-optimal solutions. This implies an aggressive reduction in solution space for disruption cases which are not as simple as the classifier identifies them. The percentage of solutions under 60 seconds also reduces to only 80.1%, and the average disruption cost increases by 6.6%, likely as a result of the additional cancellations. A similar case is present with the fixed top 60% selection, where optimality and percentage of sub-60 second solutions are both around 91%. The most aggressive selection algorithms show a near-100% rate of sub-60 second solutions. T40% selection, for example, has a lower average disruption cost than that of R90%, and solves the problem in under 60 seconds in over 99% of instances. The number of optimal solutions is lower, however. The use of T30% and T20% show a steep drop in solution quality. T20% in particular has an average disruption cost over 11% higher than that of the optimal. Depending on whether an airline prioritises speed or solution quality, any of these settings could be used instead of the dynamic selection used in the case study. The comparison of average solution times, average number of crew considered, and average disruption costs for all of these are present in Figure 6.2 and Figure 6.3

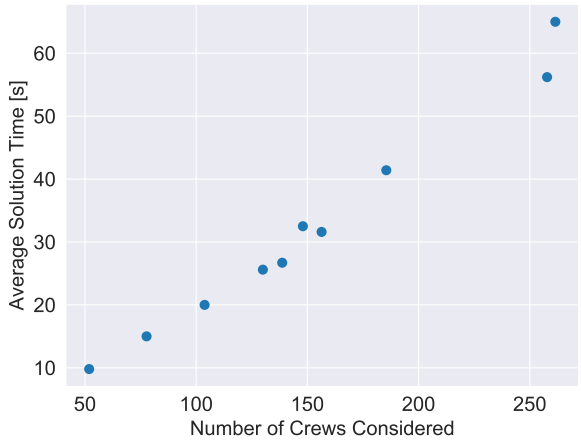


Figure 6.2: Solution times and number of crews considered for different selection algorithms.

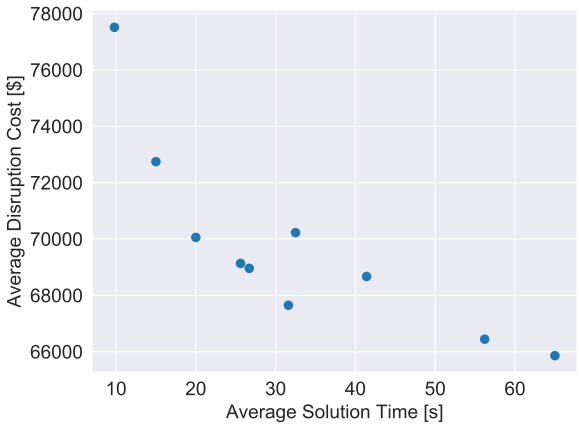


Figure 6.3: Solution times and disruption costs for different selection algorithms.

These plots show two trends for the effects of selection algorithm use. The more crews that a selection algorithm considers, the more time is required to solve the disruption. Similarly, the more crews are selected, the more likely it is that the selection contains the optimal solution. Figure 6.3 paints a good picture of the trade-off of solution time and solution quality present in the field of airline disruption management. In essence, the faster a solution is obtained, the less likely it is that it is the optimum. Solution speed and solution quality are therefore inversely proportional.

7

Conclusion and Recommendations

This chapter presents the main conclusions and recommendations for further research resulting from this thesis. Section 7.1 presents the conclusions based on the results obtained from the Delta Airlines case study. The limitations of the Sequential Disruption Set Solver (SDSS) and recommendations for future research are discussed in Section 7.2. Finally, the research questions presented at the beginning of the document are revisited in Section 7.3.

7.1. Conclusions

This work presented a machine learning approach to the aircraft and crew recovery problem. The proposed approach consists of a two-stage sequential recovery operation where the aircraft schedule is first recovered, followed by that of crew. Through the use of decision-tree based machine learning classifiers, the proposed approach reduces the number of aircraft and crews considered within a recovery operation with the end-goal of achieving faster solution times. Starting with the aircraft recovery stage developed by Hassan (2018) a time-space network based crew recovery formulation was developed and supplemented with an extreme gradient boosting (XGB) classifier. This classifier used features describing the characteristics of candidate crew with respect to disruptions in an attempt to identify those likely to assist in the recovery solution.

The performance of the SDSS was evaluated in a case study using the domestic flight network of Delta Airlines. When considering crew recovery, the developed XGB classifier was able to correctly select a subset of crew containing all those used in the optimal solution in 89% of non-trivial cases, finding a solution in 27 seconds on average. This solution time corresponds to 40% of the time required to solve the crew recovery problem when considering all crews. Despite finding a non-optimal solution to the crew recovery problem in 11% of cases, the use of the selection algorithm increased the disruption cost by only 5%.

When considering both aircraft and crew recovery for non-trivial crew solutions, the combined use of the two selection algorithms resulted in optimal selections of aircraft and crew in 83% of cases. The solution time provided by the SDSS was 56 seconds on average, corresponding to 34% of the time required when considering all aircraft and crews. The use of the selection algorithms increased the percentage of solutions obtained within the AOCC-required time limit from 47% to 95%, while increasing the average disruption cost by 11%.

7.2. Recommendations

From the research done as part of the SDSS development and testing, three distinct areas of improvement were identified. They relate to the XGB classifier used in the crew recovery stage, the formulation of the SDSS as a whole, and the solution time results obtained during testing. These will be evaluated in three separate subsections.

Extreme Gradient Boosting Classifier

Though the use of the XGB classifier in crew selection resulted in large reductions in solution time without significant deterioration of solution quality, integration with the aircraft recovery stage still leads to solution times greater than 120 seconds. More aggressive crew selection methods have proven to decrease solution quality, sometimes significantly, and a trend exists with failing to identify crew optimally in disruption instances where large amounts of crew are used in the solution. If a recovery solution prompts a non-disrupted crew to take over a flight scheduled for a disrupted crew in favour of their own, the non-disrupted crew's scheduled flight is left without crew. As the feature space is unable to provide the classifier with sufficient information for these cases, further feature engineering can enable better classifier performance and more aggressive reductions in crew selection.

XGB has proven to work well with the proposed crew recovery formulation. Classifier performance can vary greatly with problem type, however. In essence, how well a classification algorithm will perform on a given problem is difficult to predict without actual implementation, and this is one of the reasons XGB was chosen for this research. Comparable or better performance may be yielded by support vector machine classifiers, random forests, or neural networks. For this reason, benchmarking different machine learning algorithms is recommended.

Recovery Formulation

Though the recovery formulation of the SDSS provides a complete solution to aircraft and crew recovery, passenger itineraries and cabin crew schedules are not recovered. This means that only a part of the airline recovery problem is addressed with the SDSS. Furthermore, sink node violations present in the crew recovery stage can prove difficult to handle in the long run. Namely, if a recovery solution is unable to bring a crew back to their end-of-duty location, the crew can be left stranded. The AOCC would have to take action to prevent this, ideally in the form of a secondary recovery operation. Assuming the same strict time limits do not apply for this recovery operation, the problem scope could include a large time window and the entire flight network to provide more deadhead opportunities.

The presence of sink node violations is caused in part due to the mostly-separate nature of the two recovery stages. As the aircraft recovery solution only partially accounts for the implications of recovery actions on crew, preventable flight cancellations and sink node violations are present as part of the recovery. The only way to address these issues is via the direct integration of aircraft and crew within a single recovery operation. The increased computational complexity associated with this is likely to pose a large challenge to the operational applicability of such an approach, however.

Solution Time

Of the 1482 disruption scenarios tested during the case study, 29 exceed the 120-second decision making time limit. The solution time of both the aircraft and crew stages is dominated by the problem writing time rather than solving. In particular, the intermediate node-balance constraints applied to both stages as part of the parallel time-space network approach can greatly influence the solution time of the SDSS. Without changing the problem formulation or code structure, the only way to resolve this is with the use of more aggressive selection of candidate aircraft and crew. As the sensitivity analysis performed in this research proved that this can lead to solutions with higher cost, this should ideally be done after a re-iteration in feature engineering for both aircraft and crew stages.

7.3. Remarks on Research

This research project investigated the utility of machine learning in reducing the solution time of the aircraft and crew recovery problem. Four research sub-questions were posed with the aim of pinpointing the steps necessary to answer the main research question. The remainder of this section will discuss the answers to these four questions prior to providing an answer to the main research question.

1. *How can multiple resources be computationally efficiently integrated into the mathematical formulation of the aircraft and crew recovery problem?*

Initial work on this project focused on solving the integrated aircraft and crew recovery problem within a single recovery operation. Though a working mathematical formulation was developed, the solution times it provided greatly exceeded the AOCC-defined decision making time limit, one of the main constraints of the airline recovery problem. For this reason, the recovery of aircraft and crew was approached sequentially. To prevent complete separation of the two stages, crew and passenger considerations are present in the aircraft recovery stage. Reasonable solution times were obtained without the use of selection algorithms, indicating an efficient integration of the two resources within the SDSS.

2. Which simplifications to the airline recovery problem formulation are acceptable in the context of industry practice?

Though the sequential approach to aircraft and crew recovery may sometimes obtain non-optimal solutions, it is a commonly applied approach by AOCC controllers in daily operations. The main simplifications applied to the aircraft recovery problem often involve the absence of maintenance constraints, which present a large restriction in recovery options in the real-life case. Though no maintenance data is considered within the performed case study, the SDSS formulation is able to account for maintenance constraints should data be available. In crew recovery, the most common simplification is the assumption of presence of reserve crew whenever no crew is available. In reality, reserve crew are a limited resource, and their assignment is a difficult task. Despite the fact that reserve crew data is artificially generated, the presence and absence of reserve crew are accounted for in the SDSS formulation. With the possible exception of the assumption that cockpit crew work in pairs, the remainder of the simplifications applied within the SDSS do not create significant differences with respect to the real life case.

3. Given the problem formulation, what form of machine learning algorithm and approach suits the targeted solution time reduction?

With respect to the developed crew recovery formulation, a binary classification algorithm provides the most intuitive solution. The developed classifier uses extreme gradient boosting to identify crew members likely to assist in the recovery solution. The data provided by this classifier is used to create a sub-network of crew used to reduce the problem size and therefore the solution time. Several methods of sub-network selection were evaluated and the best-performing one tested, providing promising results.

4. How should the two main performance metrics, solution time and solution quality, be balanced during model assessment?

Assuming a hard time limit of 120 seconds, an airline recovery solution should aim to make a decision that minimises cost in that time limit. Therefore, though the SDSS is able to respect the AOCC-defined time limit in 98% of all 1482 disruption scenarios tested, there is a possibility of the time limit being exceeded before a recovery decision is made. The origin of these issues is the large number of constraints required to implement a parallel time-space network approach, which results in large problem write times. Despite the limitations of the SDSS, the results stemming from this research provide an answer to the main research question, which was formulated as follows:

How can the use of machine learning methods be applied to the aircraft and crew recovery problem, and what are the effects of this integration on the solution quality and solution time?

This research project demonstrated that the use of machine learning can be applied to the aircraft and crew recovery problem to provide solutions within AOCC-defined time limits. The use of machine learning to select a subset of aircraft and crew resulted in a threefold reduction in required solution time while incurring costs only 11% higher than optimal.

Bibliography

Barnhart, C., Fearing, D., and Vaze, V. (2014). Modeling Passenger Travel and Delays in the National Air Transportation System. *Operations Research*, 62(3):580–601.

Bayliss, C., Maere, G. D., Atkin, J., and Paelinck, M. (2012). Probabilistic Airline Reserve Crew Scheduling Model. In Delling, D. and Liberti, L., editors, *12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 25 of *OpenAccess Series in Informatics (OASIcs)*, pages 132–143, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Bouarfa, S., Müller, J., and Blom, H. (2018). Evaluation of a Multi-Agent System approach to airline disruption management. *Journal of Air Transport Management*, 71:108–118.

Chang, W., Liu, Y., Xiao, Y., Yuan, X., Xu, X., Zhang, S., and Zhou, S. (2019). A Machine-Learning-Based Prediction Method for Hypertension Outcomes Based on Medical Data. *Diagnostics*, 9(4):178.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority over-Sampling Technique. *J. Artif. Int. Res.*, 16(1):321–357.

Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. Association for Computing Machinery.

Delta Airlines (2020). Delta Pilots' Scheduling Reference Handbook. Technical report.

Federal Aviation Administration (2011). Flightcrew Member Duty and Rest Requirements: RIN 2120–AJ58.

Hassan, L. K. (2018). Aircraft Disruption Management Increasing Performance with Machine Learning Predictions. Technical report.

Hassan, L. K., Santos, B. F., and Vink, J. (2020). Airline Disruption Management: A Literature Review and Practical Challenges. *Computers & Operations Research*, page 105137.

Hoeben, N. J. M., Santos, B. F., and Omondi, T. (2017). Dynamic Crew Pairing Recovery AIRMES: Airline Maintenance Operations implementation of an E2E Maintenance Service Architecture and its enablers View project Dynamic Crew Pairing Recovery. Technical report.

Hossin, M. and Sulaiman, M. N. (2015). A Review On Evaluation Metrics For Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5:1–11.

Olson, R. S., Cava, W. L., Mustahsan, Z., Varik, A., and Moore, J. H. (2017). Data-driven advice for applying machine learning to bioinformatics problems. In *Biocomputing 2018*, pages 192–203.

Petersen, J. D., Sölveling, G., Clarke, J.-P., Johnson, E. L., and Shebalov, S. (2012). An Optimization Approach to Airline Integrated Recovery. *Transportation Science*, 46(4):482–500.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms.

Vink, J., Santos, B. F., Verhagen, W. J. C., Medeiros, I., and Filho, R. (2020). Dynamic Aircraft Recovery Problem - An Operational Decision Support Framework. *Computers & Operations Research*, page 104892.

Vos, H.-W. M., Santos, B. F., and Omondi, T. (2015). Aircraft Schedule Recovery Problem – A Dynamic Modeling Framework for Daily Operations. *Transportation Research Procedia*, 10:931 – 940.

Zhang, W., Wu, C., Zhong, H., Li, Y., and Wang, L. (2021). Prediction of undrained shear strength using extreme gradient boosting and random forest based on Bayesian optimization. *Geoscience Frontiers*, 12(1):469–477.

This page was intentionally left blank.

A

XGB Classifier Features

The following table contains the features used in training the XGB model used in the crew recovery stage. Note that features ending in _1 marked with an asterisk (*) also exist with the suffix _2, _3, _4, and _5 to represent their value with respect to the second, third, fourth, and fifth disruptions being solved within the disruption set. Their descriptions were omitted for brevity. The final feature presented in the table is the target output.

Table A.1: List of features used in XGB model training.

Feature Name	Type	Description
c_time_start	Integer	Duty start time of candidate crew as measured in minutes from time window start time.
c_time_end	Integer	Duty end time of candidate crew as measured in minutes from time window start time.
tw_start	Integer	Time window start time as measured in minutes from midnight on day of first departure in time window.
tw_end	Integer	Time window start time as measured in minutes from midnight on day of first departure in time window.
ac_family	Integer	Aircraft family, label encoded.
type_canx_1*	Boolean	Represents whether first disruption is a cancellation.
type_canx_6	Boolean	Represents whether sixth or any subsequent disruption is a cancellation.
t_del_1*	Boolean	Represents whether first disruption is a delay.
t_del_6	Boolean	Represents whether sixth or any subsequent disruption is a delay.
d_dur_1*	Integer	Delay duration in minutes for first disruption.
d_dur_6	Integer	Maximum delay duration in minutes for sixth or any subsequent disruption.
c_at_org_1*	Boolean	Represents whether candidate is at origin airport of first disrupted flight at time of disruption.
c_at_org_6	Boolean	Represents whether candidate is at origin airport of sixth or any subsequent disrupted flight at time of disruption.
c_at_org_1h_1*	Boolean	Represents whether candidate is at origin airport of first disrupted flight within 1 hour prior to disruption.
c_at_org_1h_6	Boolean	Represents whether candidate is at origin airport of sixth or any subsequent disrupted flight within 1 hour prior to disruption.
c_at_org_2h_1*	Boolean	Represents whether candidate is at origin airport of first disrupted flight within 2 hours prior to disruption.

Table A.1 continued from previous page

Feature Name	Type	Description
c_at_org_2h_6	Boolean	Represents whether candidate is at origin airport of sixth or any subsequent disrupted flight within 2 hours prior to disruption.
c_at_org_3h_1*	Boolean	Represents whether candidate is at origin airport of first disrupted flight within 3 hours prior to disruption.
c_at_org_3h_6	Boolean	Represents whether candidate is at origin airport of sixth or any subsequent disrupted flight within 3 hours prior to disruption.
c_at_org_before_1*	Boolean	Represents whether candidate is at origin airport of first disrupted flight at any point prior to disruption.
c_at_org_before_6	Boolean	Represents whether candidate is at origin airport of sixth or any subsequent disrupted flightat any point prior to disruption.
c_at_dest_1*	Boolean	Represents whether candidate is at destination airport of first disrupted flight at time of disruption.
c_at_dest_6	Boolean	Represents whether candidate is at destination airport of sixth or any subsequent disrupted flight at time of disruption.
c_at_dest_1h_1*	Boolean	Represents whether candidate is at destination airport of first disrupted flight within 1 hour prior to disruption.
c_at_dest_1h_6	Boolean	Represents whether candidate is at destination airport of sixth or any subsequent disrupted flight within 1 hour prior to disruption.
c_at_dest_2h_1*	Boolean	Represents whether candidate is at destination airport of first disrupted flight within 2 hours prior to disruption.
c_at_dest_2h_6	Boolean	Represents whether candidate is at destination airport of sixth or any subsequent disrupted flight within 2 hours prior to disruption.
c_at_dest_3h_1*	Boolean	Represents whether candidate is at destination airport of first disrupted flight within 3 hours prior to disruption.
c_at_dest_3h_6	Boolean	Represents whether candidate is at destination airport of sixth or any subsequent disrupted flight within 3 hours prior to disruption.
c_at_dest_before_1*	Boolean	Represents whether candidate is at destination airport of first disrupted flight at any point prior to disruption.
c_at_dest_before_6	Boolean	Represents whether candidate is at destination airport of sixth or any subsequent disrupted flightat any point prior to disruption.
c_at_crit_1*	Boolean	Represents whether candidate is at critical location of first disruption at the critical time.
c_at_crit_6	Boolean	Represents whether candidate is at critical location of sixth or any subsequent disruption at the critical time.
c_at_crit_1h_1*	Boolean	Represents whether candidate is at critical location of first disruption within 1 hour prior to the critical time.
c_at_crit_1h_6	Boolean	Represents whether candidate is at critical location of sixth or any subsequent disruption within 1 hour prior to the the critical time.
c_at_crit_2h_1	Boolean	Represents whether candidate is at critical location of first disruption within 2 hours prior to the critical time.
c_at_crit_2h_6	Boolean	Represents whether candidate is at critical location of sixth or any subsequent disruption within 2 hours prior to the the critical time.

Table A.1 continued from previous page

Feature Name	Type	Description
c_at_crit_3h_1	Boolean	Represents whether candidate is at critical location of first disruption within 3 hours prior to the critical time.
c_at_crit_3h_6	Boolean	Represents whether candidate is at critical location of sixth or any subsequent disruption within 3 hours prior to the the critical time.
c_at_crit_before_1	Boolean	Represents whether candidate is at critical location of first disruption at any time prior to the critical time.
c_at_crit_before_6	Boolean	Represents whether candidate is at critical location of sixth or any subsequent disruption at any time prior to the the critical time.
future_fl_next_1	Boolean	Represents whether candidate is scheduled to fly to the origin airport of the disrupted crew's next scheduled flight after STD of first disrupted flight.
future_fl_next_6	Boolean	Represents whether candidate is scheduled to fly to the origin airport of the disrupted crew's next scheduled flight after STD of sixth or any subsequent disrupted flight.
future_fl_end_1	Boolean	Represents whether candidate is scheduled to fly to the end-of-time-window airport of the disrupted crew after STD of first disrupted flight.
future_fl_end_6	Boolean	Represents whether candidate is scheduled to fly to the end-of-time-window airport of the disrupted crew after STD of sixth or any subsequent disrupted flight.
crit_time_1	Integer	Critical time of first disruption as measured in minutes from time window start time.
crit_time_6	Integer	Minimum critical time of sixth or any subsequent disruption as measured in minutes from time window start time.
reserve_crew	Boolean	Represents whether candidate is a reserve crew.
result	**Boolean**	**Represents whether candidate has schedule changed in optimal solution.**

This page was intentionally left blank.

B

Draft Integrated recovery model

This appendix contains a draft of an integrated recovery model that simultaneously recovers aircraft and crew. Note that this model has not been verified or validated. In essence, the use of this model removes the inefficiencies of the sequential approach in terms of solution quality. By directly being able to consider the financial impact of recovery while accounting for both aircraft and crew constraints, the integrated recovery model should be able to acquire higher quality solutions. The following sections describe the sets, decision and non decision variables used, as well as the proposed mathematical formulation of the model.

Sets

- **P** - Set of aircraft p
- **K** - Set of crew
- **F** - Set of flights i
- **E** - Set of aircraft types e
- **A** - Set of airports a
- **N** - Set of all nodes n
- $\mathbf{N_O}$ - Set of origin nodes n
- $\mathbf{N_I}$ - Set of intermediate nodes n
- $\mathbf{N_S}$ - Set o sink nodes n
- **T** - Set of delay steps t
- **S** - Set of slack variables j
-

Decision Variables

Aircraft Decision Variables:

- $\delta_{F_{p,i}}$ - If p allocated to i
- $\delta_{FD_{p,i,t}}$ - If p allocated to i with d
- δ_{C_i} - If i cancelled
- $\delta_{GF_{p,n}}$ - If p uses n-originating ground arc
- $\delta_{F'_i}$ - If i flown by unscheduled AC
- $s_{a,e}$ - Slack if infeasible.

Crew Decision Variables:

- $\delta_{K_{k,i}}$ - If k allocated to i
- $\delta_{KD_{k,i,t}}$ - If k allocated to i with t
- $\delta_{GF_{p,n}}$ - If k uses n-originating ground arc
- $\delta_{K'_i}$ - If i flown by unscheduled crew
- s_k - Slack if sink constraint violated
- s_{FT_k} - Slack if scheduled flight time is exceeded
- $\delta_{DH_{k,i}}$ - If k deadheaded on i
- $\delta_{DHD_{k,i,t}}$ - If k deadheaded on i with t

Variables

Aircraft Variables:

- $C_{OP_{p,i}}$ - Operating cost of p on i
- $C_{D_{i,t}}$ - Delay cost of for i, t
- C_{C_i} - Cancellation cost for i
- C_{G_n} - Cost of ground arc originating from n
- $h_{e,n}$ - Number of AC of type e required at node n
- $C_{C_{SCH}}$ - Unscheduled AC operating penalty

Crew Variables:

- $C_{OP_{k,i}}$ - Operating cost of k on i
- $C_{DH_{k,i}}$ - Deadhead cost of k on i
- C_{OC} - Unscheduled crew operating penalty
- C_{SV_k} - Sink node violation cost for k
- C_{FT} - Flight time exceeded penalty
- FT_i - Flight time of i

Objective Function and Constraints

The following equation encompasses the entirety of the financial considerations present in the integrated aircraft and crew recovery operation:

$$\begin{aligned}\text{Min} \sum_{p\in P}\sum_{i\in F} C_{OP_{p,i}}\cdot\delta_{F_{p,i}} + \sum_{p\in P}\sum_{i\in F}\sum_{t\in T}(C_{OP_{p,i}}+C_{D_{i,t}})\cdot\delta_{FD_{p,i,t}} + \sum_{p\in P}\sum_{n\in N} C_{G_n}\cdot\delta_{GF_{p,n}} + \sum_{i\in F} C_{C_i}\cdot\delta_{C_f} \\ + \sum_{k\in K}\sum_{i\in F}\left(C_{OP_{k,i}}\cdot\delta_{K_{k,i}} + C_{DH_{k,i}}\cdot\delta_{DH_{k,i}} + \sum_{t\in T}(C_{OP_{k,i}}\cdot\delta_{KD_{k,i,t}} + C_{DH_{k,i}}\cdot\delta_{DHD_{k,i,t}})\right) + \sum_{k\in K}\sum_{g\in G} C_{G_n}\cdot\delta_{G_{k,n}} \\ + \sum_{i\in F} C_{C_{SCH}}\cdot\delta_{F'_i} + \sum_{j\in S} s_j\cdot M + \sum_{i\in F} C_{OC}\cdot\delta_{K'_i} + \sum_{k\in K} C_{SV}\cdot s_k + \sum_{k\in K} C_{FT}\cdot s_{FT_k}\end{aligned} \tag{B.1}$$

In the above function, the first line is the sum of all *aircraft-flight* related costs: operating a scheduled flight on-time or with a delay, utilizing ground arcs, and cancelling a scheduled flight. The second line contains the *crew-flight* costs: operating and deadheading crew on an on-time flight, operating and deadheading crew on a delayed flight, and utilizing ground arcs. Finally, the third line refers to the necessary *slack variables* to ensure feasibility and prevent unwanted behaviour from the model. These are: missing an aircraft or crew type at an airport at the end of the time window, changing an aircraft or crew routing, and violating crew flight time constraints. Equation B.2 ensures all flights are flown, delayed, or cancelled.

$$\delta_{C_i} + \sum_{p\in P}\left(\delta_{F_{p,i}} + \sum_{t\in T}\delta_{FD_{p,i,t}}\right) = 1 \quad \forall i \in F \tag{B.2}$$

The following two constraints ensure that if a flight is flown by an aircraft p, a single crew k must be assigned to the flight. Two separate constraints are required to ensure that each flight and delayed flight arc can only deadhead crew if it is flown.

$$\delta_{F_{p,i}} = \sum_{k\in K}\delta_{K_{k,i}} \quad \forall p \in P, \forall i \in F \tag{B.3}$$

$$\delta_{FD_{p,i,t}} = \sum_{k\in K}\delta_{KD_{k,i,t}} \quad \forall p \in P, \forall i \in F, \forall t \in T \tag{B.4}$$

Deadheading crew on cancelled flights is prevented with the following constraint:

$$\sum_{k\in K}(\delta_{DH_{k,i}} + \sum_{t\in T}\delta_{DHD_{k,i,t}} \leq M\cdot(1-\delta_{C_f}) \quad \forall i \in F \tag{B.5}$$

Equation B.6 ensures the the net flow between intermediate nodes is always equal to zero for all aircraft. Equation B.7 ensures the net flow out of the aircraft origin node is equal to one, while Equation B.8 ensures the net flow into the aircraft sink node is equal to one.

$$\left(\delta_{G_{p,n-1}} + \sum_{i\in F_{in}}\delta_{F_{p,i}} + \sum_{i\in F_{in},t\in T}\delta_{FD_{p,i,t}}\right) - \left(\delta_{G_{p,n}} + \sum_{i\in F_{out}}\delta_{F_{p,i}} + \sum_{i\in F_{out},t\in T}\delta_{FD_{p,i,t}}\right) = 0 \quad \forall p \in P, n \in N_i \tag{B.6}$$

$$\delta_{GF_{p,n}} + \sum_{i \in F_{out}} \delta_{F_{p,i}} + \sum_{i \in F_{out}, t \in T} \delta_{FD_{p,i,t}} = 1 \quad \forall p \in P, n = \text{scheduled } N_o \text{ of } p \tag{B.7}$$

$$\sum_{p \in P(e)} \left(\delta_{GF_{p,n-1}} + \sum_{i \in F_{in}} \delta_{F_{p,i}} + \sum_{i \in F_{in}, t \in T} \delta_{FD_{p,i,t}} \right) + s_j \geq h_n^e \quad \forall e \in E, n \in N_s \tag{B.8}$$

For every flight operated by a non-scheduled aircraft, a penalty is incurred in the objective function as governed by the following constraint.

$$\delta_{F_{p,i}} + \sum_{t \in T} \delta_{D_{p,i,t}} - \delta_{F'_i} = 1 \quad \forall i \in F, p = \text{ aircraft scheduled for } i \tag{B.9}$$

Similarly to the logic followed in the aircraft node-balance constraints, the following three constraints govern the net flow between intermediate, origin, and sink nodes of crew.

$$\begin{aligned} &\left(\delta_{GK_{k,n-1}} + \sum_{i \in F_{in}} (\delta_{K_{k,i}} + \delta_{DH_{k,i}}) + \sum_{i \in F_{in}, t \in T} \delta_{KD_{k,i,t}} + \delta_{DHD_{k,i,t}} \right) \\ -&\left(\delta_{GK_{k,n}} + \sum_{i \in F_{out}} (\delta_{K_{k,i}} + \delta_{DH_{k,i}}) + \sum_{i \in F_{out}, t \in T} \delta_{KD_{k,i,t}} + \delta_{DHD_{k,i,t}} \right) = 0 \quad \forall k \in K, n \in N_i \end{aligned} \tag{B.10}$$

$$\delta_{GK_{k,n}} + \sum_{i \in F_{out}} (\delta_{K_{k,i}} + \delta_{DH_{k,i}}) \sum_{i \in F_{out}, t \in T} (\delta_{KD_{k,i,t}} + \delta_{DHD_{k,i,t}} = 1 \quad \forall k \in K, n = \text{scheduled } N_o \text{ of } k \tag{B.11}$$

$$s_k + \delta_{GK_{k,n-1}} + \sum_{i \in F_{IN_n}} (\delta_{K_{k,i}} + \delta_{DH_{k,i}}) + \sum_{i \in F_{IN_n}, t \in T} (\delta_{KD_{k,i,t}} + \delta_{DHD_{k,i,t}}) = 1 \quad \forall k \in K, n = \text{scheduled } N_s \text{ of } k \tag{B.12}$$

For all flights operated by non-scheduled crew, a penalty is incurred in the objective function according to the following constraint.

$$\delta_{K_{k,i}} + \sum_{t \in T} \delta_{KD_{k,i,t}} - \delta_{K'_i} = 1 \quad \forall i \in F, k = \text{ crew scheduled for } i \tag{B.13}$$

The following constraint ensures that crew flight time within the time window remains less than or equal to the time maximum allowable time limit. In the case that the scheduled flight time is exceeded to an amount no greater than the legal limit, a penalty is incurred in the objective function.

$$\sum_{i \in F} (\delta_{K_{k,i}} + \sum_{t \in T} \delta_{KD_{k,i,t}}) \cdot FT_i \leq FTL_k + FTM_k \cdot s_{FT_k} \quad \forall k \in K \tag{B.14}$$

This page was intentionally left blank.

C

Crew Cost Data

This appendix elaborates on the costs assigned to each of the constituent elements of the crew recovery stage of the SDSS. Crew operating costs for U.S. airlines are assumed to be based almost exclusively on the number of flight hours operated by crew member. This is true within the considered model, as the secondary objectives of swapping, using reserve crew, ending up at the sink node, and not cancelling flights are all quantified based on a qualitative weight. Though crew wages per airline are officially confidential information, cockpit crew members regularly leak such information to enthusiast websites that can help aspiring pilots weigh their options. The cockpit crew wages are based on the type of aircraft, as well as the number of years of experience in flying the specific type of aircraft. For the fleet of Delta Airlines, these are present in Table C.1 and were obtained from the online database of Airline Pilot Central[1].

Table C.1: Hourly wages per crew member for each of the aircraft fleets used within Delta's domestic U.S. network. These are displayed for years of experience with given fleet as captain/first officer.

Experience	B717	B737	B747	B757	B777	A320	A330	A350	MD88/90	B717
12	$354/$242	$354/$242	$339/$232	$334/$228	$334/$228	$296/$202	$286/$195	$284/$194	$274/$187	$269/$184
11	$351/$240	$351/$240	$336/$229	$332/$226	$332/$226	$293/$200	$283/$193	$282/$192	$272/$186	$266/$182
10	$349/$237	$349/$237	$334/$227	$329/$224	$329/$224	$290/$198	$281/$191	$280/$191	$270/$183	$263/$179
9	$346/$234	$346/$234	$331/$224	$327/$221	$327/$221	$287/$194	$279/$189	$278/$188	$268/$181	$260/$176
8	$343/$232	$343/$232	$329/$222	$324/$219	$324/$219	$285/$192	$277/$187	$276/$186	$266/$179	$258/$174
7	$341/$226	$341/$226	$326/$217	$322/$214	$322/$214	$283/$188	$275/$183	$273/$182	$264/$175	$256/$170
6	$338/$220	$338/$220	$324/$211	$319/$208	$319/$208	$281/$183	$273/$178	$271/$177	$262/$171	$254/$166
5	$335/$215	$335/$215	$321/$206	$317/$203	$317/$203	$278$178	$270/$173	$269/$173	$260/$166	$252/$162
4	$333/$210	$333/$210	$318/$201	$314/$198	$314/$198	$276/$174	$268/$169	$267/$168	$257/$162	$250/$158
3	$330/$205	$330/$205	$316/$196	$312/$194	$312/$194	$274/$170	$266/$165	$265/$164	$255/$159	$248/$154
2	$327/$175	$327/$175	$313/$168	$309/$165	$309/$165	$272/$145	$264/$141	$263/$141	$253/$136	$245/$131
1	$325/$92	$325/$92	$311/$92	$307/$92	$307/$92	$269/$92	$262/$92	$261/$92	$251/$92	$243/$92

For the work done in this thesis, each crew pair was assumed to consist of one captain and one first officer, both with 6 years of work experience. The cancellation cost of a flight is equal to that determined by the aircraft recovery, with an additional penalty incurred in the case of crew being missing at its sink airport.

[1]Airline Pilot Central: Delta Airlines hourly wages - https://www.airlinepilotcentral.com/airlines/legacy/delta_air_lines

This page was intentionally left blank.