Large-Scale Airline Recovery using Mixed-Integer Optimization and Supervised Learning

by

Ahmet Esat Hızır

B.S., Industrial Engineering, Marmara University (2004)
M.S., Industrial Engineering, Sabanci University (2006)
M.S., Air Transport Management, Istanbul Technical University(2015)

Submitted to the Department of Civil and Environmental Engineering in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Transportation

at the

Massachusetts Institute of Technology

February, 2024

Authored by: Ahmet Esat Hızır
Civil and Environmental Engineering
January 12, 2024

Certified by: Cynthia Barnhart
Provost and Abraham J. Siegel Professor of Management Science
Professor of Operations Research
Thesis Supervisor

Certified by: Vikrant S. Vaze
Stata Family Career Development Associate Professor
Thayer School of Engineering, Dartmouth College
Thesis Supervisor

Accepted by: Heidi Nepf
Donald and Martha Harleman Professor of Civil and Environmental Engineering
Chair, Graduate Program Committee

Large-Scale Airline Recovery using Mixed-Integer Optimization and Supervised Learning

by

Ahmet Esat Hızır

Submitted to the Department of Civil and Environmental Engineering
on January 12, 2024, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Transportation

Abstract

Airlines plan their aircraft and crew schedules using operations research methods. However, these schedules are often disrupted due to the irregular nature of flight operations. Airline recovery is the process in which airlines take various actions to adjust and repair their aircraft routes, crew schedules, and passenger itineraries. This process has a sequential structure in practice, where aircraft recovery is followed by crew and then passenger recovery. Although recovery problems are smaller in scope than their planning counterparts, limited solution timeframes prevent airlines from using a full-scale optimization approach.

This thesis proposes fast solution methods that combine mixed-integer optimization and supervised machine learning techniques to find better solutions to large-scale airline recovery problems than those found with other exact and heuristic approaches. Our approach reduces the solution space for a given disruption by adding constraints (cuts) based on the patterns discovered in the solutions to historical disruptions. The model with the added cuts is solved using mixed-integer optimization solvers.

During the day of flight operations, the available time for airlines to handle disruptions may vary. The overall solution process we propose allows parameter tuning to match the extent of solution space reduction with the available solution time. This feature helps the proposed methods to effectively navigate the trade-off between solution quality and runtime. Our computational studies are conducted using real flight and crew schedules from major US airlines with more than 2,500 daily flights. Experiments demonstrate that our approach can generate solutions of significantly higher quality than benchmark methods.

We use tree-based classification methods to predict recovery decisions. Due to their interpretable structures, we are able to discover insights into the attributes of effective recovery decisions. We demonstrate that these insights can be incorporated into currently used heuristic-based airline recovery processes to improve solution quality by up to 15%.

Thesis Supervisor: Cynthia Barnhart
Title: Provost and Abraham J.Siegel Professor of Management Science and Professor of Operations Research

Thesis Supervisor: Vikrant S. Vaze
Title: Stata Family Career Development Associate Professor
Thayer School of Engineering, Dartmouth College

Acknowledgments

First, I would like to thank my advisors, Prof. Cynthia Barnhart and Prof. Vikrant Vaze, for their guidance and support in this journey. Cindy: despite your busy schedule, you have dedicated your valuable time and energy to our research. Your comments and suggestions were always inspiring. Vikrant: your attention to detail and continuous pursuit of excellence helped me improve as a researcher.

I would like to extend my appreciation to my committee members, Prof. Alexandre Jacquillat and Prof. Ilker Birbil. Alex: your comments brought a different perspective to the committee discussions, enriching our research. You even named a key concept *"micro-solution"* that we ended up using in the thesis.

Ilker: since the first time we met during my master's program at Sabanci University almost 20 years ago, you have always believed in me and motivated me to seek greater challenges. It was our discussions on the complexity of real-life airline scheduling problems that inspired me to go into the aviation industry.

Special thanks to Prof. Amedeo Odoni. Amedeo: this thesis would not have existed without your motivation and guidance. I decided to pursue a Ph.D. thanks to our conversations during the Air Transport Management Master's program at Istanbul Technical University. Starting way before the application process and continuing throughout my journey, your guidance was invaluable.

I had the privilege of meeting many brilliant scholars and practitioners. I significantly improved my understanding of the U.S. airline industry during my conversations with Prof. Peter Belobaba from MIT, Dr. Joseph Mathews from United Airlines, Captain Erich Schnitzler, and Captain Greg Auld from Southwest Airlines.

The lion's share of my appreciation goes to my wife, Saime, and my daughter, Elif, to both of whom I dedicate this thesis. They gave me the strength and motivation to overcome this challenge. Saime: having the same experience of pursuing a Ph.D., you have understood and supported me in my struggles. Elif: since we started this journey, I was not always available to spend enough time with you. But without the joy you brought to my life, this day would not be possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Airline recovery refers to the process by which airlines take action to adjust their aircraft routes, crew schedules, and passenger itineraries in response to disruptions. This process typically follows a sequential structure, even though solutions of each step are highly dependent on other steps. The complexity of integrating aircraft, crew, and passenger recovery steps and the solution time constraints during the day of operations prevent airlines from solving the integrated problem.

Airline recovery problems are among the most complex integer optimization problems encountered regularly by the industry practitioners. What makes them particularly challenging is the solution time limitation on the day of flight operations. When a disruption occurs, airlines must act quickly to recover their resources so that the consequences of the disruption do not significantly propagate to their network. Limited time availability prevents airlines from adapting optimization-based solution methods, especially for large-scale problem instances. Heuristic methods based on expert judgments or rules of thumb are common, potentially leading to low-quality solutions, resulting in flight delays and cancellations, some of which may be avoidable. These disruptions can cost an airline millions of dollars per day. The total cost of flight delays in 2007 was estimated to be $33 billion in the US alone (Ball et al., 2010). More recently, the annual cost of flight delays worldwide was predicted to be

around $60 billion (Wang & Vaze, 2016).

One of the primary objectives of this research is to develop recovery methods that airlines can implement in practice. We believe that there are four major requirements for an efficient and practical recovery system: quality, speed, flexibility, and interpretability. Our objective in this thesis is to develop a set of solution methods that can meet all these requirements.

1. *Quality*

   The recovery method should generate high-quality recovery solutions. The main objective of *disruption management* is to minimize the effects of disruptions on flight operation costs, including crew and passenger costs. In this context, a high-quality solution corresponds to a low-cost solution. Airlines would like to find the highest possible quality (optimal) solution for every disruption. Since this is rarely attainable, they instead seek applicable (feasible) solutions with relatively good quality.

2. *Speed*

   The recovery method should be fast because, on the day of flight operations, solutions to disruptions must be generated within limited timeframes. This is important because waiting for a recovery solution itself can cause further delays. In this thesis, we assume that the practical solution time limit is five minutes for individual steps and ten minutes for the entire recovery process. We develop methods that can generate recovery solutions within these limits.

3. *Flexibility*

   The recovery method should be flexible enough to easily be tuned to adapt to the available solution time limit. The disruptions that airlines experience have wide ranges with respect to scope, severity, and urgency. Severe disruptions, such as an airport closure for several hours, correspond to very large problems that may potentially affect the entire flight network. On the other hand, minor disruptions, such as a crew calling out sick, would not have the same scope and severity. Airlines would be inclined to spend a longer time looking for a solution

to the former problem than to the latter. Therefore, it is important to ensure that the recovery approach being used for each disruption type has a run-time that is consistent with the airline's practical needs for that type of disruption.

4. *Interpretability*

   Airlines use decision support systems to help recovery staff handle the disruptions during the day of operations. Whether it is based on heuristic solution approaches or optimization-based methods, when a Decision Support System (DSS) recommends a solution, it is still up to the recovery teams to decide whether or not to follow the recommendation. In practice, many experienced staff may be inclined to follow their own instinct rather than applying the DSS recommendation if its rationale is not clear. In such a setting, the DSS suggestions must be interpretable so that decision-makers can understand the reasons for the recommendation. That would help build trust in the DSS and lead to better overall recovery performance for the airline.

Machine learning (ML) methods, or artificial intelligence (AI) in general, have recently been incorporated into many aspects of our lives. From customer-specific shopping suggestions to image recognition tasks, ML/AI has shown tremendous success in many fields. Recently, generative AI models, like ChatGPT, have initiated discussions on whether human-level artificial general intelligence (AGI) is possible. Many even argue that progress in AI developments should be slowed down and regulated to avoid unintended consequences.

On the basis of these success stories and recent developments, one may be tempted to believe that AI/ML methods alone might be sufficient to tackle some of the most complex problems, such as combinatorial optimization (CO) problems. Although some studies have tried to pursue this idea, methods lacking any optimization integration usually fail to generate viable solution approaches on a consistent enough basis to be usable in practice. A recent survey paper by Bengio et al. (2020) reviews studies that use ML to solve CO problems. They argue that the major drawback of using only ML is the difficulty of ensuring the feasibility of the solution. Airline

recovery problems are examples of CO problems. They involve huge numbers of constraints modeling aircraft, crew, and passenger rules. Therefore, even with recent progress, handling airline recovery problems by relying solely on AI/ML does not seem possible. Such efforts fail to meet the most essential requirement of an airline recovery system: generating feasible and high-quality solutions.

In this research, we show that combining ML and optimization helps develop solution methods that meet all the major requirements listed above: *quality, speed, flexibility,* and *interpretability*. ML accelerates the solution process by identifying a solution space tailored for a given disruption and thus helps meet the *speed* requirement. The selected ML method, *classification trees*, meets the *interpretability* requirement due to the tree structure of the resulting classification models. Optimization methods help achieve the *quality* requirement by creating a realistic model of the problem and ensuring solution feasibility. The *flexibility* of the proposed methods is achieved by tuning some key hyper-parameters of the presented overall approach for different solution time limits.

1.2 General Framework

The general framework in this study is based on the idea that, under *similar* disruptions, the solution characteristics that lead to high-quality solutions can also be similar. Therefore, to find high-quality solutions in limited timeframes, we can use ML methods to discover the similarities between the current disruption and previous disruptions, solutions of which are generated in advance. Although ML methods cannot predict the entire solution, partial predictions of recovery decisions help to significantly accelerate the solution process.

Figure 1-1 summarizes the general flowchart of the solution methodology. The main objective is to find a solution for the disruption on the day of operations. We call this the *online phase*. The dots inside the *online phase* box represent alternative solutions to the same disruption. Due to the time limitations, it is not practical to consider all solution alternatives in the optimization model. Therefore, we filter the

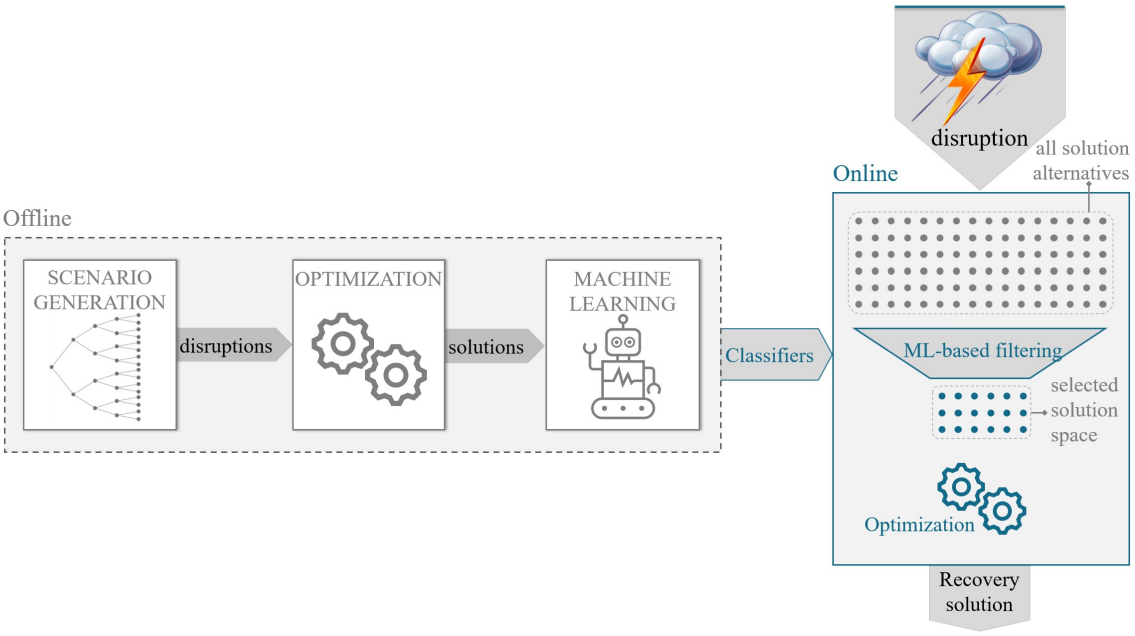


Figure 1-1: General flowchart of the solution approach

solution alternatives using ML predictions and determine the solution space for the recovery problem by adding the corresponding constraints (cuts) to the model. The regions of the solution space that are deemed not suitable for the given disruption are removed by these cuts. When all cuts are added, we use a mixed-integer optimization solver to find a high-quality solution within this reduced solution space.

The solution methodology also involves an *offline phase* to set up the methods for online use. It starts with generating the disruption scenarios for the day of operations based on historical delay data. After scenario generation, all scenarios are solved with an optimization-based solution approach. The next step is to train ML classification models to learn the relationship between disruption characteristics and certain recovery decisions. The resulting classification models, also called *classifiers*, are used in the online phase to filter the solution alternatives.

Solution spaces that lead to high-quality solutions may vary for different disruptions (Figure 1-2). For example, alternative solutions to a disruption corresponding to airport closure for a certain airport would not share too many similarities with solution alternatives to address a minor disruption, such as a crew member not showing up for duty.

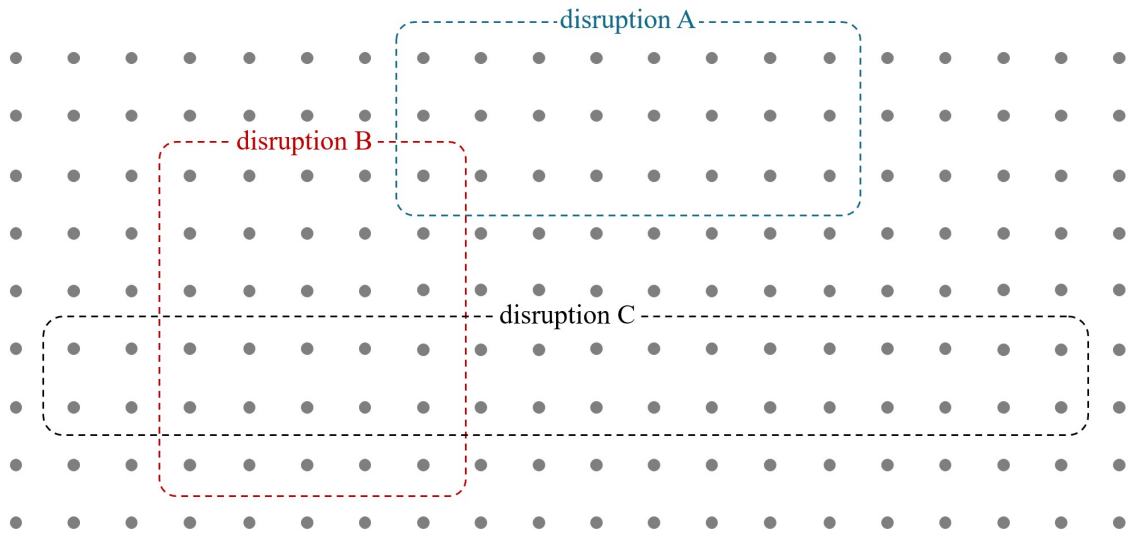


Figure 1-2: Efficient solution spaces for different disruptions

Figure 1-3 demonstrates the expected relationship between the average solution quality (measured in terms of the recovery cost) and the size of the solution space being considered, under different solution time limits. With shorter time availability, e.g., 1 minute, a smaller solution space with fewer alternatives makes it easier to find a higher-quality solution. If the available solution time is longer, e.g., 5 minutes, it is better to include more solution alternatives in the model. Our solution process allows tuning its hyper-parameters that govern the solution space reduction and aligns it with the available solution time. This ability allows our proposed methods to effectively navigate the trade-off between the solution quality and run time.

We argue above that the most effective solution space for a given disruption is a function of the disruption characteristics and the available solution time limit. ML methods help us to learn this mapping between the historical disruption scenarios and their corresponding offline solutions.

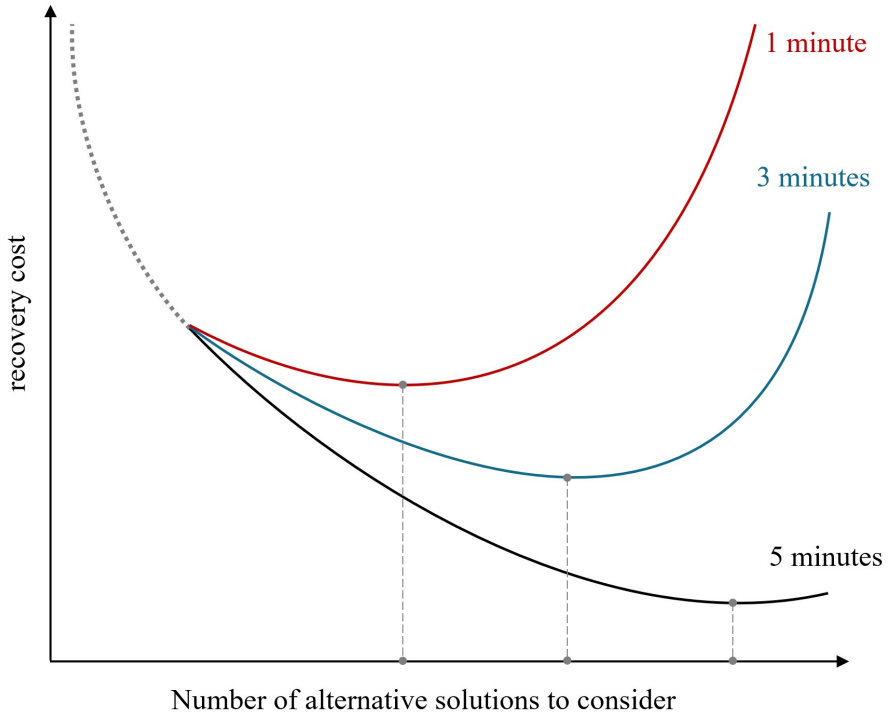


Figure 1-3: Relationship between available solution time and optimal size of the reduced solution space

1.3 Thesis Outline

Chapter 2: Crew Recovery with Aircraft and Passenger Considerations

We introduce a mixed-integer linear optimization model for crew recovery with aircraft and passenger considerations. We represent a disruption as the set of primary delay predictions for the flights in the network.

We follow the general framework presented in Section 1.2 and propose fast solution methods that combine optimization and ML methods to find solutions to crew recovery problems within limited timeframes that are better than existing benchmark approaches.

We conduct a computational study using actual flight and crew schedule information from a major US airline with 2,870 daily flights to evaluate the performance of our proposed solution approaches. The disruption scenarios are generated using actual delay and cancellation data. Our experiments demonstrate that we can generate recovery solutions in 3 minutes with a 3% higher cost than the baseline solutions, which are generated by optimization-based methods that require up to 2 hours. This corresponds to a solution quality that is more than two times better than other practical benchmark methods.

To predict recovery decisions, we use tree-based classification methods that provide interpretable results, allowing us to discover insights into effective recovery decisions. For example, our ML models allow us to identify the disruption conditions under which a specific flight connection should be used. These types of insights can enhance manual recovery processes by building the recovery team members' trust in the ML-based decision support systems.

Our ML-based approach generates more robust solutions to inaccurate delay predictions than those generated by optimization-only approaches. Compared to the optimization-only method, our approach, using a combination of optimization and ML, reduces the extent to which the solutions are overfitted to individual disruption events. By generating solutions suitable for a wider range of disruptions than those generated by optimization-only approaches, our approach ensures that the solutions

perform relatively well even when the actual primary delays are somewhat different than their predicted values.

Chapter 3: Integrated Aircraft, Crew, and Passenger Recovery

We introduce a tractable integrated recovery model that captures the major aspects of schedule, aircraft, crew, and passenger recovery steps. It is based on a modeling approach first presented by Bertsimas and Patterson (1998) that provides tractability due to the way the main decision variables are defined.

We follow the general framework presented in Section 1.2 and propose fast solution methods that combine optimization and ML techniques to find solutions to integrated recovery within limited timeframes that are better than existing benchmark approaches.

We conduct a computational study using real flight and crew schedule information from a major US airline with 3,706 daily flights to evaluate the performance of the proposed methods. Our experiments demonstrate that the presented model improves the tractability of the integrated problem, and the overall solution approach can generate integrated recovery solutions in 8 minutes within 5% of the baseline solutions, which are generated by an exact optimization-based solution method in 2 hours. Compared to other practical benchmark methods that we tested, solutions generated by our proposed method, under identical run-time budgets, are 2-3 times better in terms of solution quality difference measured relative to the baseline solutions.

The interpretable structure of tree-based classification methods allows us to discover valuable insights into the decisions that lead to effective and trustworthy integrated recovery strategies.

Chapter 4: Rule-based Improvements to Recovery Heuristics

We propose simple modifications to heuristic solution methods based on rules of thumb discovered using ML methods. Our results show that it is possible to improve the performance of the existing recovery processes to some extent and reap some of the benefits of the ML-based methods, without undertaking the complex implementation

efforts required to adopt full-fledged ML-based methods.

Our benchmark solution approach uses a common matheuristic for crew recovery problems and for integrated recovery problems. It relies on a solution space reduction mechanism that keeps some of the planned aircraft and crew schedules intact while allowing modifications to others to create a recovery solution to a given disruption. The aircraft and crew schedules that are allowed to be modified are determined based on the number of swap opportunities with disrupted aircraft routes and crew schedules. We show that by incorporating simple rules into the process of determining the schedules to keep, we can improve the solution quality by up to $\sim$15%.

Finally, in Chapter 5, we summarize the thesis and discuss future research directions.

Chapter 2

Crew Recovery with Aircraft and Passenger Considerations

2.1 Introduction

For decades, operations research methods have been successfully applied to airline scheduling. A wide range of airline scheduling problems have been formulated as integer or mixed integer optimization problems, and various solution algorithms and heuristics have been developed to solve them. Due to the complex structure of scheduling problems and the massive scale of flight networks, major airlines use optimization tools, developed in-house or provided by software vendors, to optimize their primary resources, namely, aircraft and crew.

However, optimized aircraft and crew schedules are rarely operated precisely as planned due to operational disruptions, such as weather impacts, aircraft malfunctions, crew absences, etc. While incorporating robustness into schedules can help reduce recovery costs (Barnhart & Vaze, 2015b), airlines still need to closely monitor their flight operations and take actions to repair disrupted aircraft and crew schedules and passenger itineraries to minimize the effects of disruptions. This process is called airline disruption management. The operational recovery decisions include delaying or canceling flights, re-routing or swapping aircraft, re-scheduling crews, calling in reserve crews, using spare aircraft, rebooking passengers, etc.

In theory, jointly optimizing the recovery of the flight schedules and aircraft, crew, and passenger itineraries, should generate the best solutions. However, the complexity of the integrated problems and practical time limitations during the day of operations prevent airlines from adopting such an approach. Instead, many major airlines follow a sequential recovery process in which the joint problem of aircraft and schedule recovery is solved first, followed by crew recovery and then passenger recovery. This study focuses on the crew recovery problem, which attempts to repair disrupted crew schedules during the day of operations while ensuring aircraft recovery solution feasibility and indirectly accounting for passenger disruption costs.

Although recovery problems are smaller than their planning counterparts, the limited time availability makes it more challenging to use optimization approaches to solve recovery problems to an acceptable solution quality. Instead, airlines usually adopt methods based on a combination of simplified optimization problems, heuristic solutions, and expert judgment to find solutions in limited timeframes. These solutions tend to be selected from a smaller set of alternatives than available, and hence can result in increased recovery costs. These recovery costs can add up to significant amounts. According to Ball et al. (2010), the total delay cost in the US airline industry in 2007 was around $33 billion, more than $8 billion of which were additional fuel, crew, and maintenance expenses. The worldwide annual cost of flight delays was predicted to be around $60 billion (Wang & Vaze, 2016).

The airline industry is known to have low profit margins (Doganis, 2005). The average profit margin since 2006 is less than 2% (IATA, 2023b). During the most profitable years within that period, 2015-2019, it was around 4.2%. In 2023, despite exceeding the initial forecasts, the net profit margin in the airline industry is expected to be 1.2% (IATA, 2023a). Due to the large magnitudes of recovery costs, efficient re-planning of expensive resources, such as aircraft and crew, plays a vital role in maintaining profitability. With the increasing complexity associated with expanding network sizes and growing passenger demand, it has become increasingly difficult for airlines to use manual or heuristic-based simplified recovery processes to find high-quality recovery solutions. Consequently, there is a growing need for fast and

automated optimization-based methods to handle increasing complexity and generate low-cost recovery solutions.

Some airlines and software vendors have undertaken initiatives to develop automated recovery systems with optimization capabilities. A common philosophy in these initiatives is to reduce the size of the problem by considering only a limited number of flights, aircraft, and crews. For example, when recovering crews associated with one crew base, a common way to reduce the size of the problem is to exclude changes to crew schedules for crews at other bases. Another common approach is to exclude from optimization all aircraft and crews that are not themselves disrupted *and* not seen as a direct swap opportunity for the disrupted aircraft and crews. However, these size reduction heuristics may worsen the recovery costs. By not exploring the entire available solution space, promising but non-obvious recovery alternatives may go unnoticed. Specifically, a promising yet underutilized opportunity lies in *learning* from the solutions to similar recovery problems that succeeded in lowering the recovery costs in previous instances or offline settings. By identifying similarities across disruption instances using data-driven methods, such as machine learning (ML) techniques, to guide the online optimization approach, it may be possible to accelerate the solution process without sacrificing solution quality. That is the main idea behind the methods developed in this research.

Specifically, we propose a new mixed-integer optimization model for crew recovery while accounting for aircraft and crew considerations, enabling partial integration of these other recovery aspects into crew recovery optimization. We then propose a new solution approach leveraging similarities across disruption instances to accelerate the process of generating solutions to the optimization model. Our computational experiments, performed using some of the largest recovery instances found in existing literature, demonstrate that our method achieves near-optimal solutions within much shorter runtimes than all benchmarks tested for such highly challenging instances.

The remainder of this chapter is organized as follows. In this section, we review prior literature, discuss the motivation behind this study, and list its contributions. Section 2.2 presents the modeling approach, discusses our disruption representation,

and provides a mathematical formulation for crew recovery. In Section 2.3, we present the solution methodology, including how we leverage ML techniques. Section 2.4 is dedicated to the computational study, including the experimental setup, results, and a discussion of the insights gathered from the experiments. Section 2.5 presents further improved results obtained using an extended version of our solution method, and also demonstrates the robustness of our solutions in the face of inaccuracies in delay predictions used when making the recovery decisions. Finally, Section 2.6 summarizes the chapter and discusses future research directions.

2.1.1 Literature Review

Early studies in airline disruption management primarily focused on the aircraft recovery problem (Teodorović and Guberinić (1984), Teodorović and Stojković (1990)). Since then, many others have studied aircraft recovery, often modeling the problem as an integer optimization problem (Clarke (1998), Rosenberger et al. (2003)).

From a mathematical perspective, the crew recovery problem is similar to the aircraft recovery problem but has additional constraints concerning crew legality rules due to government regulations and collective bargaining agreements. Stojković et al. (1998), Nissen and Haase (2006), and Medard and Sawhney (2007) formulated the crew recovery problem assuming that the flight schedule is recovered first. Stojković and Soumis (2001), Abdelghany et al. (2004), and Zhao et al. (2007) are some of the first crew recovery studies that took flight delays into account. Johnson et al. (1994), Lettovskỳ et al. (2000), and Yu et al. (2003) extended crew scheduling formulations with additional variables to represent flight cancellations. Considering that the crew recovery problem is usually handled after the aircraft recovery problem, the modeling strategy that avoids further cancellations in the crew recovery step better reflects the airline practices.

There are two main types of recovery formulations: an arc-based multicommodity network flow model and a string-based set-covering model. In the arc-based model, the underlying network consists of flight nodes, and arcs connect nodes whose corresponding flights could be consecutively assigned to the same crew. This type of model

includes an elaborate set of constraints to ensure that the crew duties in the solution are feasible. A *crew duty* is a set of consecutive flights that can be operated by a crew on a given day. In a string-based set covering model, alternative crew duties (also called strings) that individual crews can operate are generated in advance. In this modeling approach, binary decision variables are introduced to reflect whether or not a string is included in the solution.

Many previous recovery studies that employ set-covering models introduce flight copies to model flight delay decisions. Each flight copy is defined as a combination of flight departure and arrival time, and different copies of the same flight represent different extents to which a flight is delayed. The main downside is that this significantly increases the size of the problem. Consequently, these flight delay modeling approaches may lead to tractability issues in larger problem instances.

Liang et al. (2018) used an alternative set-covering-based modeling approach to airline recovery. Specifically, they model the aircraft recovery problem as a set-covering problem with continuous delays instead of the time-increment-based discrete delays approach used by the models adapting a flight copies approach. Liang et al. (2018) present a column generation heuristic that considers delay propagation to calculate the aircraft rotation durations. *Rotation* is defined as the sequence of flights assigned to an aircraft. The flight delays in the final solution can take any continuous values rather than being restricted to a discrete set, and thus offer greater solution flexibility. *Crew duty* and *rotation* are similar concepts as they both correspond to a sequence of flights assigned to a crew and an aircraft, respectively. Therefore, the continuous delay approach is also viable for the crew recovery problem.

An essential requirement for any recovery method is the ability to generate solutions in limited timeframes. A recent airline disruption management survey (Hassan et al., 2021) argues that the time available for a recovery solution generation can sometimes be as low as 1-2 minutes. Some past studies have tested their airline recovery ideas and methods on small and synthetic problems, where it is possible to generate solutions in a reasonable time. However, they often do not scale well to real-life instances. Other studies that try to tackle relatively larger problems usually allow

solution times to be extended, for example, to 10-30 minutes. Some studies reduce the problem size to generate solutions within available solution time by limiting modifications to planned aircraft and/or crew schedules. These rule-based restrictions, while easy to use, do not consider disruption characteristics. For example, Stojković and Soumis (2001) and Abdelghany et al. (2004) limit the delay of a flight to 60 minutes, while Lettovskỳ et al. (2000) and C.-H. Chen and Chou (2016) limit the number of crew swap alternatives. These restrictions are applied across all flights and across instances regardless of the disruption attributes, potentially eliminating good solutions.

The idea of applying ML to aviation operations is not new. Rebollo and Balakrishnan (2014), Kim et al. (2016), Choi et al. (2016), Wang and Vaze (2016) and Gopalakrishnan and Balakrishnan (2017) use data related to flight operations, such as weather events and declared airport capacities, and apply a wide range of ML and statistical methods to capture delay propagation mechanisms and predict flight delay distributions. Gopalakrishnan et al. (2016), Kuhn (2016) and Gorripaty et al. (2017) leverage ML techniques to analyze the airspace system, identify patterns, and classify days based on their disruption characteristics. However, none of the aforementioned studies directly tackle the problem of minimizing airline recovery costs.

Airline recovery problems, like their planning counterparts, are combinatorial in nature. Recent years have seen considerable interest in using ML for combinatorial optimization (CO) problems, for example, for developing better branching and cutting plane techniques. Alvarez et al. (2017) approximate branching decisions using decision trees and supervised ML, while Gasse et al. (2019) use graph neural networks (GNN) to learn an offline approximation for branching decisions. Tang et al. (2020) use reinforcement learning to select good cutting planes, while Baltean-Lugojan et al. (2018) use a neural network to approximate the lower bound improvement generated by cutting planes.

Some researchers have recently been developing ML-based methods specifically for solving certain classes of airline recovery problems, which are specialized mixed-integer and combinatorial optimization formulations. Hondet et al. (2018) experiment

with using reinforcement learning methods for aircraft recovery decisions without relying on optimization. Rashedi et al. (2023) present a hybrid approach that leverages offline optimization solutions to train supervised ML techniques, to expedite the solution of the aircraft recovery problem.

A recent survey paper by Bengio et al. (2020) classifies the studies that combine ML and optimization for solving CO problems based on their learning methods and algorithmic structures. They consider two learning methods — *imitation*, based on supervised ML methods, and *experience*, based on reinforcement learning methods — and three classes of algorithmic structures. The first algorithmic structure is *end-to-end learning*, which uses ML to predict and create the solution to a problem without using optimization. The second is *learning-to-configure*, which uses ML and optimization sequentially to initiate, partially generate, or complete the solution. The last class, *ML alongside optimization*, involves an iterative use of ML and optimization within the same algorithm. Studies that replace existing branching mechanisms with the faster, ML-based branching methods fall into this last category. The procedures developed in this chapter follow a supervised ML approach, and an algorithmic structure that falls within the *learning-to-configure* category.

Studies employing ML for combinatorial optimization usually aim to accelerate the solution process, in a way that is not directly informed by the available solution time, by replacing some of the computationally heavy tasks with ML procedures (Alvarez et al. (2017), Balcan et al. (2018), Alabi et al. (2019) and Morabit et al. (2021)). A crucial, yet often neglected, desirable property of solution approaches is the ability to adjust to the available solution time. This is especially important in the airline recovery context because airlines may face varying time constraints when searching for solutions to address different types of disruptions. Some disruptions, such as a sudden equipment failure delaying a flight departure, require a quick solution within just a few minutes, while others may allow for several minutes, such as an airport likely to experience capacity reduction later in the day. Running the same solution procedure for as long as time availability allows is not an ideal approach. Instead, there is value in developing methods that can be tuned to the computational needs.

When employing ML for combinatorial optimization, one faces a clear trade-off. On the one hand, delegating less work to ML can lead to higher quality solutions, at the expense of potentially longer run-times. On the other hand, delegating more can provide faster solution convergence, but at the expense of potentially greater suboptimality. Our experiments showed that, especially with shorter solution time availability, the size of the solution space can be too large to find even a feasible solution, let alone a high-quality one. The extent of solution space reduction that leads to the best solution varies with varying solution time availability. For the best performance, the recovery method should be adjustable to the available solution time and should be able to identify the best possible solution quality for each run-time budget. In this study, we develop methods that help find the highest possible quality solution for a given solution time limit, by tuning some of the hyper-parameters of our solution approach and demonstrating their stability through out-of-sample tests. This helps the proposed methods to adjust to the available solution time, thus effectively navigating the trade-off between solution quality and run-time.

2.1.2 Contributions

A major challenge in using ML for recovery problems is ensuring the feasibility of the generated solutions. ML-only methods do not provide a feasibility guarantee (Bengio et al., 2020), thus necessitating a combination of optimization and ML.

We develop and employ effective ML-driven solution space reduction strategies before running optimization, and then solve the reduced optimization problem using state-of-the-art optimization solvers. ML accelerates the solution process by quickly evaluating recovery decision alternatives as a function of the disruption scenario attributes, fixing some of them in advance, and thus limiting the solution space. Our results show that this method provides a more effective alternative to the common solution space reduction methods used in practice and reported in the literature, such as adding only a limited number of candidate crews to the solution space.

The major contributions of this research are listed below.

1) Model: We introduce a new modeling approach for the crew recovery problem

with aircraft and passenger considerations that accounts for the various types of propagation of delays through the network. The model, partly based on the set-covering model, uses continuous delay modeling (instead of flight copies) similar to the aircraft recovery model by Liang et al. (2018). To accurately reflect the airline recovery processes in practice, our model does not allow additional flight cancellations beyond those decided in the aircraft recovery step. Instead, it ensures the feasibility of aircraft rotations under crew-based propagation of flight delays. Disrupted passenger recovery considerations are added as approximate cost components to the objective function to incentivize the model to seek more passenger-friendly solutions.

2) Solution Method: We present a data-driven and ML-guided approach to reduce solution space for crew recovery optimization. We define the set of decisions corresponding to a small portion of the overall recovery solution as a *micro-solution* and train ML models to predict the use of these micro-solutions as part of the overall crew recovery solutions. The feature set used in ML model training includes disruption information and a feature representing the use of the corresponding micro-solution in the aircraft recovery solution. Note that the aircraft recovery solution is obtained before the crew recovery process begins and hence it can be used as an input to the crew recovery process. This feature representing the use of the corresponding micro-solution in the aircraft recovery solution is needed because our crew recovery optimization model ensures the feasibility of the aircraft recovery solution and micro-solutions in aircraft and crew recovery solutions are expected to be correlated.

Unlike other approaches that use decision variable-based predictions (Furian et al., 2021) or try to predict the entire solution (Bertsimas & Stellato, 2022), our micro-solution predictions focus on a smaller portion of the overall solution. Specifically, each micro-solution corresponds to a follow-on (F/O) pair, defined as two consecutive flights in an assigned crew duty. The proposed solution method fixes the F/Os that have a high probability of being in the overall solution, based on the ML predictions. Fixing an F/O corresponds to adding constraints (cuts) to the model that require that either the flights in the F/O are assigned to the same crew duty or at least one of the flights in the F/O is assigned to high-cost reserve crews (which are deemed as

last resort recovery actions). Each such cut eliminates from the model some recovery decisions that are unlikely to be in an optimal or near-optimal solution.

3) Recovery Cost Reduction: Our fast and tunable solution methods, combining optimization and ML tools, generate solutions with recovery costs that are consistently lower than the solutions generated with other heuristics and state-of-the-practice solution approaches. Moreover, we generate these solutions within limited timeframes of under 5 minutes. Recovery costs for solutions generated with our approach in under 5 minutes of runtime are also compared with those generated using a direct optimization approach, allowing for run-times of up to 2 hours. Our solutions found in 5 minutes are within ∼3% of the cost of the direct optimization solutions, while the best benchmark method generates solutions with a ∼6% cost above the direct optimization solutions.

4) Efficient Setup for Offline Training: The primary objective of this research is to develop practical methods that airlines can adopt. The proposed methods require that an offline training phase be completed before the day of operations. The time and computational resources required during this offline process may be a bottleneck for some airlines. To reduce the resources needed for our approach, we introduced procedures to significantly accelerate even the offline phase while still generating high-quality solutions. This is accomplished by a) generating tailored solution databases for different disruption types, b) defining a concise feature set for each micro-solution, and c) accepting the use of lower-quality solutions, with a 20% optimality gap target, for generating the database and for training the ML models.

5) Improved Solution Robustness: We show that our methods can yield solutions that are as robust or more robust to uncertain delay predictions than those found by the direct optimization approach. Our experiments show that the recovery costs for the direct optimization solutions are ∼3% lower than those for our ML-based optimization solutions when the recovery cost is calculated assuming perfect delay information. However, the costs of our ML-based optimization solutions are about the same as those of the direct optimization solutions, when the delay predictions are imperfect. We observed that the F/Os used in ML-based optimization solutions

occur more frequently in operations than those used in the direction optimization solutions, suggesting that the former are suitable for a broader range of disruptions and thus enhance the solution robustness.

6) Interpretability: We use tree-based classification methods so that the resulting classifier structures can be easily interpreted. These interpretable classifiers can not only improve the level of trust among human decision-makers towards the automated ML-driven decision support tools, but they can also be used to guide manual rule-of-thumb-based recovery processes.

2.2 Problem Statement and Mathematical Model

2.2.1 Crew Recovery Problem

Broadly speaking, crew recovery is the process by which airlines modify their crew schedules to minimize the effects of disruptions. Major crew recovery decisions include re-scheduling crews, calling in reserve crews, and re-scheduling flights. The decisions taken by airlines must comply with crew scheduling and recovery regulations published by government agencies, such as the Federal Aviation Administration (FAA) in the United States. Crew recovery solutions must comply with numerous rules and regulations (Barnhart & Smith, 2012). We considered some of the most important crew recovery rules in our model. However, our model also allows considering many other crew recovery rules due to our string-based modeling approach.

A critical regulation concerns flight duty period (FDP) limits. FAA defines FDP as *"a period that begins when a pilot is required to report for duty with the intention of conducting a flight or series of flights and ends when the aircraft is parked after the last flight with no intention for further aircraft movement by the same pilot."* A crew is deemed to be *on duty* from the check-in time for the first flight in the assigned duty until the arrival time of the last flight. The sequence of flights assigned to a crew within an FDP is called a crew duty. The actual FDP duration is affected by flight delays and their propagation through crew duties. It is crucial to keep track of

the actual FDP values because it is illegal for a crew to operate a flight if the legal FDP limit is exceeded.

In addition to the maximum FDP time, other rules should be followed when generating a crew duty, like the minimum connection times between two consecutive flights in the duty. If the flights are assigned to the same aircraft, the connection time should be greater than or equal to the minimum aircraft turnaround time to allow flight preparation activities such as unloading and reloading baggage and cargo, refueling, cleaning and maintenance of the aircraft, and boarding and disembarking passengers. Although the minimum turn-around time scheduled by different airlines may vary, values of around 25 to 30 minutes are common in practice for narrow-body aircraft, and those ranging from 45 to 90 minutes are common for wide-body aircraft. If there is an aircraft change in the duty, that is, if the consecutive flights in a crew duty are assigned to different aircraft, an additional time (usually around 15 minutes) is needed for the crew to connect to the new aircraft.

Another important rule is the minimum duty rest time, which specifies the minimum off-duty time that crews are entitled to after completing a duty. This rule ensures that crews are well-rested and can perform their duties effectively, even during long and demanding schedules. Our model ensures that FDP limits are not exceeded by computing delay propagation through the crew duty strings. Moreover, it also ensures that the legality rules on the minimum connection time between the flights (with and without aircraft change), and the minimum rest time between consecutive duties are satisfied.

2.2.2 Disruption Definition

Since disruption information is a key input to the mathematical model, before sharing the details of the model, we present the disruption definition used in this study and discuss the underlying assumptions.

Since, in practice, the crew recovery problem is usually addressed after the schedule and aircraft recovery steps, we define the disruptions from a crew recovery perspective. A major issue in crew recovery is the extension of crew duty duration resulting

from flight delays. As discussed in Section 2.2.1, it is against regulations to operate crew duties that exceed FDP limits. Additionally, many other problems can also arise due to flight delays. For example, minimum duty rest time requirements could be violated, rendering the planned crew schedules illegal for the next day. The passenger itineraries can get significantly impacted by the flight delays. Some passengers may miss their connections, while others may experience extensive delays, resulting in inconvenience and loss of passenger goodwill. Taking into account the impacts of flight delays on the crew recovery problem, as well as on passenger itineraries, we characterize the disruption of a given day of flight operations as a set of expected flight delays in the entire network.

However, not all types of flight delays can be used to characterize disruptions. A flight delay could occur because its aircraft or the crew did not arrive on time due to a delay on a previous flight. This kind of delay is called propagated delay. It does not occur because of an external disruption event for the current flight that the airline cannot control. For example, the propagated delay for the current flight may be avoided if the airline decides to use a backup aircraft or reserve crew instead, thus preventing the delay from the previous flight from propagating.

Similar to Lan et al. (2006), we assume that the total delay of a flight d_i is the sum of two components: propagated delay and non-propagated delay, described as:

- *Propagated delay* (p_i): The delay of flight i caused by the late arrival of the flight immediately before flight i on the same aircraft rotation, crew duty or passenger itinerary. The propagated delay of a flight depends on the operated aircraft rotations, crew duties, and realized passenger itineraries.

- *Non-propagated delay* (NP_i): The delay of flight i caused by external sources of disruptions such as air traffic conditions, equipment failures, or other unforeseen factors. This type of delay is also called *independent delay* (Lan et al., 2006), as it does not depend on the operated aircraft rotations, crew duties, or realized passenger itineraries.

Let i and j be consecutively assigned flights to a crew and $SCij$ be the slack time

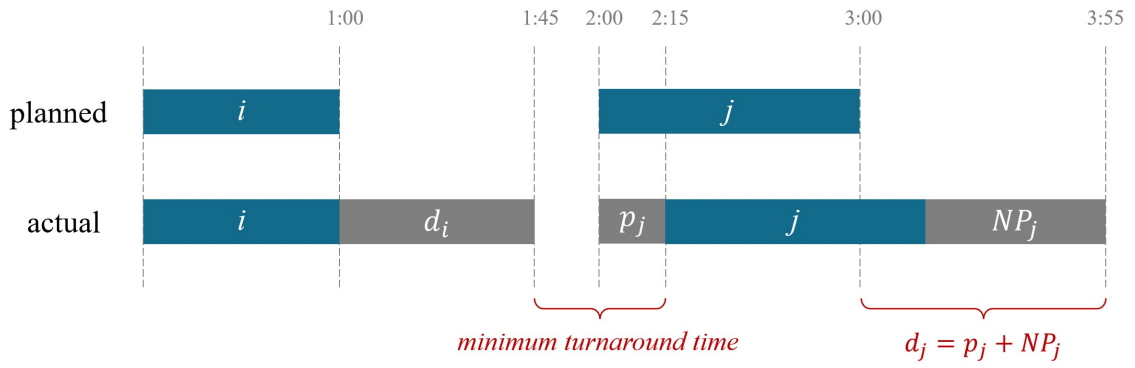


Figure 2-1: Delay propagation mechanism through operated aircraft rotations, crew duties and realized passenger itineraries

between the flights, which is the difference between the planned connection time and the minimum turnaround time. *Minimum turnaround time* for crew is defined as the minimum time required between the arrival of the previous flight and the departure of the next flight assuming both are assigned to the same aircraft. The planned *connection time* is the time planned between the arrival of the earlier flight and the departure of the later flight. The total delay of flight j, denoted d_j, equals $p_j + NP_j$, where $p_j = max(d_i - SC_{ij}, 0)$. The delay propagation mechanism through aircraft rotations and passenger itineraries, which the model decides to be kept intact, is similar where the corresponding slack time values are defined as SR_{ij} and SP_{ij}, respectively. Slack time for passengers, SP_{ij}, depends on minimum passenger connection times rather than minimum turnaround time.

Figure 2-1 shows an example of a flight connection where the planned connection time is 60 minutes, the minimum turnaround time is 30 minutes, and hence the slack time between flights i and j, SC_{ij}, is 30 minutes. The delay of flight i, d_i, is 45 minutes, and the non-propagated delay of the flight j, NP_j, is 40 minutes. To calculate the total delay of flight j, d_j, we first calculate the propagated delay, p_j, as 15 minutes, $p_j = max(d_i - SC_{ij}, 0) = max((45 - 30), 0)$. Thus, d_j equals 55 minutes, the sum of the propagated departure delay of 15 minutes, and the non-propagated delay of 40 minutes.

NP_i is defined by the scope and magnitude of the actual disruption but does

not depend on the operated aircraft rotations, crew duties, or realized passenger itineraries. Therefore, we mainly characterize the disruptions in terms of the NP_i values for individual flights. The total delay of a flight, d_i, and the propagated delay, p_i, are introduced as decision variables and calculated as described above, within the mathematical model formulation. In contrast, the NP_i values are defined by the current disruption scenario and are provided as input to the model.

This approach to defining disruptions provides three practical benefits. First, it accounts for a wide variety and combination of issues in the airline network that may disrupt flight operations, instead of focusing narrowly on each disruption. Due to the interconnected nature of flight networks, even small disruptions may have network-wide propagation impacts. Therefore, it is crucial to consider all forecasted flight delays and disruptions when optimizing recovery plans. Second, this definition not only considers problematic flights but also takes the predicted early arrivals (represented as negative NP_is) into account to discover and exploit the flexibility provided by them. For example, by creating a crew duty in which a flight that is predicted to arrive early due to a block time slack is assigned immediately after a problematic flight, it may be possible to reduce or even prevent the propagation of the delay further downstream. This is particularly relevant since many airlines have increased block time buffers in recent years to further pad their schedules. Finally, this definition offers a comprehensive set of features, which makes it convenient for the ML model to detect similarities between different disruption scenarios.

2.2.3 Modeling Approach

As discussed in Section 2.1.1, our model uses a set-covering formulation with continuous delay modeling (instead of requiring flight copies) that calculates the propagation of flight delays through the network. To achieve this, we first generate a large set of crew duty strings based on the planned flight schedules. This set also includes crew duty strings that are infeasible with respect to minimum connection time rules if the flights are operated as scheduled. We allow our mathematical optimization model to determine the departure and arrival times for each flight in the crew duty

by introducing a set of constraints to calculate delay propagation. These constraints ensure that the flight departure time is no earlier than the earliest time when the crew operating the flight is available, the earliest time when the aircraft assigned to the flight is available, the earliest time when all the passengers whose itineraries are decided to be kept intact are available or the flight's scheduled departure time, whichever happens last.

2.2.4 Crew Recovery Model with Aircraft and Passenger Considerations

The Crew Recovery Model with Aircraft and Passenger Considerations (CRM-APC) minimizes crew recovery costs and approximate passenger delay and disruption costs, while ensuring that the aircraft recovery solution remains feasible. It assumes that the aircraft recovery problem is solved first, and its output is used as an input to the CRM-APC. The recovery decisions captured by the CRM-APC model include delaying flights, re-scheduling crews, using reserve crews, and breaking passenger itineraries.

A passenger itinerary is deemed broken when passengers cannot fly at least one of the original flights in their itinerary due to delay-induced missed connections. Passengers on such an itinerary are considered disrupted. The CRM-APC approximates the passenger disruption cost from broken itineraries. It only considers itineraries broken due to delay decisions taken by the CRM-APC model. Broken itineraries due to cancellations in the aircraft recovery step are not considered.

In this formulation, a crew corresponds to a cockpit crew consisting of one captain and one first officer. *Normal reserve crews* are those who have no active duties planned during the recovery period, but are tasked with being ready during the recovery period to fly if called in. In the event of insufficient availability of normal reserve crews, airlines may, as a last resort, call in crews on their off-days or vacations. We call these crews as *high-cost reserve crews* because they are paid at a higher rate than other crews.

In addition to the crew duties generated to cover the flights in the network, the model also uses a set of dummy crew duties, S^D, that correspond to crew staying where they are if their next day's duty starts at the same airport or transferring the crew to the airport where their next day's duty starts, by some other means than the scheduled flights in the network, such as deadheading on another airline's scheduled flight and using a mode of transportation other than flights. Deadheading corresponds to a crew flying as passengers.

Notation

Sets

I: the set of flights

K: the set of crews, including normal reserves

S: the set of crew duties, also called crew strings

S^D: the set of dummy crew duties

R: the set of aircraft rotations selected in the aircraft recovery step

P: the set of planned passenger itineraries

F_s: the set of consecutive flight pairs $(i, j) \in I$ in crew string $s \in S$

F_p: the set of consecutive flight pairs $(i, j) \in I$ in passenger itinerary $p \in P$

F_r: the set of consecutive flight pairs $(i, j) \in I$ in aircraft rotation $r \in R$

S_k : the set of crew strings $s \in S$ that can be assigned to crew $k \in K$

K_s : the set of crews $k \in K$ that can be assigned to crew string $s \in S$

Data

NP_i: non-propagated delay of flight $i \in I$

CC_s^k: cost of assigning string $s \in S$ to crew $k \in K$

WC_s^k: crew delay cost per minute when string $s \in S$ is assigned to crew $k \in K$

ZC_i: cost of assigning flight $i \in I$ to a high-cost reserve crew

QC_p: approximate cost associated with broken passenger itinerary $p \in P$

VC_p: passenger delay cost per minute of arrival delay for itinerary $p \in P$

SC_{ij}: planned slack time between flights $i, j \in I$ for a crew connection

SR_{ij}: planned slack time between flights $i, j \in I$ for an aircraft connection

SP_{ij}: planned slack time between flights $i, j \in I$ for a passenger connection

a_{is}: 1 if crew string $s \in S$ contains flight $i \in I$; 0 otherwise

LS_s: last flight in crew string $s \in S$

LR_r: last flight in aircraft rotation $r \in R$

LP_p: last flight in passenger itinerary $p \in P$

LD_r: maximum allowable delay of the last flight in aircraft rotation $r \in R$ (considering both the maintenance requirements and the start time of the next rotation).

LD_s^k: maximum allowable delay of the last flight in crew string $s \in S$ when assigned to crew $k \in K$ (considering the legality of that duty and the start time of the next duty)

M: an arbitrarily large number.

Decision Variables

y_s^k : 1 if crew string $s \in S_k$ is assigned to crew $k \in K$; 0 otherwise

w_s^k : delay of crew string $s \in S_k$ when assigned to crew $k \in K$

z_i : 1 if flight $i \in I$ is assigned to a high-cost reserve crew; 0 otherwise

d_i : total arrival delay of flight $i \in I$

p_i : propagated departure delay of flight $i \in I$

q_p : 1 if the passenger itinerary $p \in P$ becomes broken; 0 otherwise

v_p : arrival delay of passenger itinerary $p \in P$

Formulation

$$\min \quad \sum_{k\in K}\sum_{s\in S_k}(CC_s^k y_s^k + WC_s^k w_s^k) + \sum_{i\in I} ZC_i z_i + \sum_{p\in P}(QC_p q_p + VC_p v_p) \tag{2.1}$$

$$\text{s.t.} \quad z_i + \sum_{k\in K}\sum_{s\in S_k} a_{is} y_s^k \geq 1 \qquad \forall i \in I \tag{2.2}$$

$$\sum_{s\in S_k} y_s^k = 1 \qquad \forall k \in K \tag{2.3}$$

$$\sum_{k\in K_s} y_s^k \leq 1 \qquad \forall s \in S \setminus S^D \tag{2.4}$$

$$NP_i + p_i \leq d_i \qquad \forall i \in I \tag{2.5}$$

$$d_i - SC_{ij} - M(1 - \sum_{k\in K_s} y_s^k) \leq p_j \qquad \forall (i,j) \in F_s, \forall s \in S \tag{2.6}$$

$$d_i - SR_{ij} \leq p_j \qquad \forall (i,j) \in F_r, \forall r \in R \tag{2.7}$$

$$d_i - SP_{ij} - Mq_p \leq p_j \qquad \forall (i,j) \in F_p, \forall p \in P \tag{2.8}$$

$$d_{LR_r} \leq LD_r \qquad \forall r \in R \tag{2.9}$$

$$d_{LS_s} - M(1 - y_s^k) \leq w_s^k \qquad \forall s \in S_k, \forall k \in K \tag{2.10}$$

$$w_s^k \leq LD_s^k \qquad \forall s \in S_k, \forall k \in K \tag{2.11}$$

$$d_{LP_p} - Mq_p \leq v_p \qquad \forall s \in S_k, \forall k \in K \tag{2.12}$$

$$y_s^k, z_i, q_p \in \{0,1\} \qquad \forall s \in S_k, \forall k \in K, \forall i \in I, \forall p \in P \tag{2.13}$$

$$w_s^k, p_i, d_i, v_p \in \mathbb{Z}^+ \qquad \forall s \in S_k, \forall k \in K, \forall i \in I, \forall p \in P \tag{2.14}$$

The objective function (2.1) minimizes the total recovery costs, including those due to modifications to planned crew duties, CC_s^k, crew delays, WC_s^k, high-cost reserve crew use, ZC_i, disrupted passenger costs, QC_p, and passenger delay costs, VC_p.

CC_s^k corresponds to the additional crew pay incurred when duty $s \in S$ is assigned to crew $k \in K$ compared to that crew's planned duty. WC_s^k calculates the additional crew pay incurred due to the extension of duty $s \in S$ when assigned to crew $k \in K$. ZC_i is the total crew pay for assigning flight $i \in I$ to high-cost reserve crew. QC_p is an approximate cost of a disrupted passenger that was originally scheduled to

take itinerary $p \in P$. It covers the costs of passenger accommodation, passenger inconvenience and loss of passenger goodwill. VC_p is the cost of delay to all non-disrupted passengers scheduled to take itinerary $p \in P$. It may account for passengers inconvenience and loss of passenger goodwill costs.

These recovery costs are only those in excess of the planned ones. The CRM-APC does not allow additional cancellations beyond those in the aircraft recovery solution. Flights that the existing crews cannot cover are assigned to high-cost reserve crews.

Constraints (2.2) ensure that each flight is assigned to at least one crew. Crew strings correspond to duties that can be assigned to crews, including normal reserve crews. If more than one crew is assigned to a flight, then all additional crews fly as passengers, i.e., they are *deadheaded*. Constraints (2.3) model that each crew must be assigned to exactly one actual or dummy duty. Constraints (2.4) ensure that a duty, except for dummy duties, is assigned to at most one crew. Assignment of multiple crews to the same duty is not allowed for the non-dummy duties. Constraints (2.5)-(2.8) calculate the total delay for each flight and the delay propagation through crew duties, aircraft rotations, and unbroken passenger itineraries. The model can delay flights to ensure the feasibility of duties, rotations, and unbroken itineraries. Passenger itineraries are also allowed to be broken, but the CRM-APC model cannot break aircraft rotations. Constraints (2.9) ensure that the aircraft will be ready on time for the next day's rotation. Constraints (2.10)-(2.11) calculate the crew delay while ensuring that the duty will remain legal with respect to the FDP limits and that the crew will be ready on time for the next duty. Constraints (2.12) calculate the delay in reaching the final destinations for passengers whose itineraries are not broken. Constraints (2.13)-(2.14) define the domains of the decision variables.

As discussed in Section 2.2.3, CRM-APC is a string-based model with continuous delay modeling instead of flight copies. This modeling approach entails a key trade-off. On the positive side, this modeling approach makes it possible to overcome memory limitations for large-scale problems and significantly improves tractability due to the lack of flight copies. On the other hand, it requires that the delay propagation mechanism be explicitly defined in the form of constraints. Implementing some logical

dependencies necessitates big-M type of constraints (namely, (2.6), (2.8), (2.10), and (2.12)) which may make the model less tractable. We partly mitigate this effect by carefully determining the values of each M in these constraints. Specifically, we set the M to 16 hours, so that the corresponding constraints would allow any practically possible flight delays, up to 16 hours, while keeping the added complexity at minimum.

2.3 Solution Methodology

2.3.1 General Framework

In our CRM-APC mathematical model, the assignments of feasible crew duties to crews are represented by binary decision variables, and the final solution consists of the set of assignments whose corresponding decision variables are set to 1. Each assignment can be seen as a separate recovery decision, collectively forming the solution, like pieces of a puzzle.

To generalize this idea, we present the concept of a *micro-solution* that is defined as a set of decisions corresponding to a small portion of the overall recovery solution. Aside from the assignment decisions, there are other types of micro-solutions for the given formulation, like the crew duty strings, and the consecutive flight pairs in a string — called follow-on (F/O) pairs. A recovery solution can be characterized as a set of micro-solutions of a certain type. An important research question is: which micro-solutions should be utilized to build the best possible solution given the characteristics of the current disruption? When there are no computational time limitations, solving the problem optimally by considering all micro-solutions is the best way to find the minimum-cost solution. This corresponds to solving the model without any reduction in solution space. However, due to the limited timeframes in recovery operations, it is not possible. Therefore, we need a faster and more practical method to help us find the best solution within the available solution time limit.

The general framework in this study is based on the idea that, under *similar* disruption conditions, the micro-solutions leading to high-quality solutions will likely

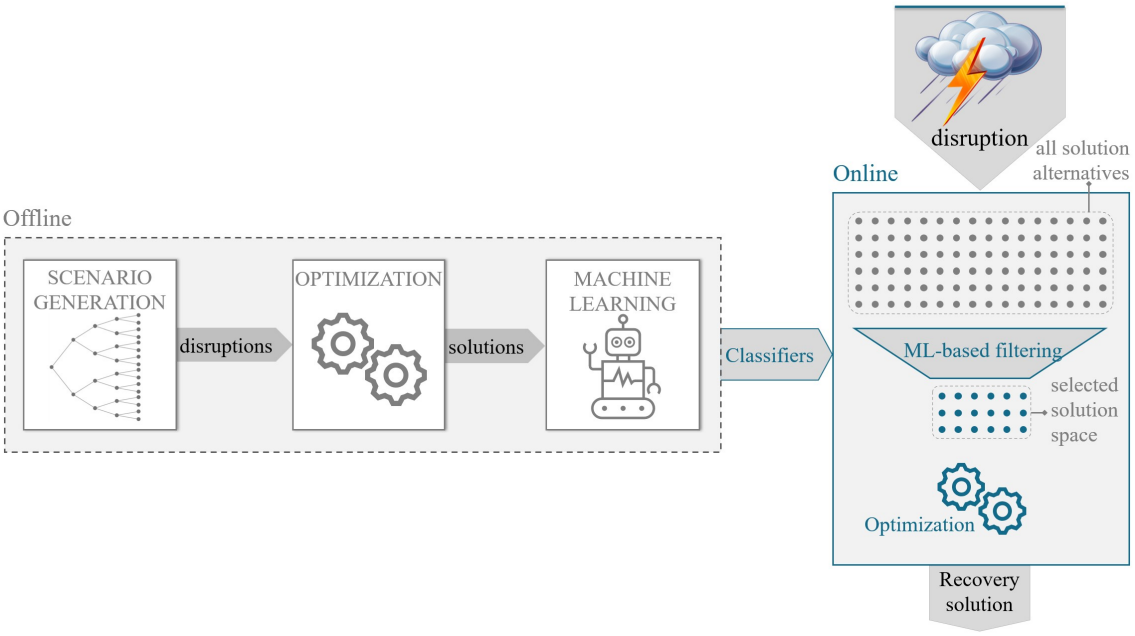


Figure 2-2: General flowchart of the solution approach

also be similar. Therefore, to find high-quality solutions in limited timeframes, we leverage ML methods to discover the similarities between the current disruption and previous disruptions, solutions of which are already available.

Figure 2-2 summarizes the general flowchart of the solution methodology. The main objective is to find a solution for the disruption faced during the day of operations, that is, during the *online phase*. The dots inside the box on the right represent alternative solutions to the same disruption. Due to the time limitations, it is not practical to consider all alternative solutions in the optimization model. Hence, we filter the alternative solutions using ML predictions and determine the solution space for the recovery problem at hand by adding constraints (cuts) to the model and fixing some micro-solutions to be included in the solution. The overall solutions that do not include these fixed micro-solutions are removed from the solution space by these cuts. After adding all cuts, we use a mixed-integer optimization solver to find a good solution within this reduced solution space.

Unlike what some prior studies have proposed for simpler online optimization problems (Bertsimas & Stellato, 2022), it is not possible to predict the entire solution of recovery problems due to their complex combinatorial nature. In comparison,

optimizing over a reduced solution space is a more viable method. Some other studies for other problem contexts (Furian et al., 2021) have proposed decision-variable-based predictions. In our experiments, the proposed micro-solution-based prediction and solution space reduction approach yielded better solutions for the CRM-APC recovery problem than using these decision-variable-based predictions, due to the increased flexibility provided by predicting and fixing smaller parts of the solutions than fixing a decision variable. See Appendix A.3 for more details on these results.

The overall solution methodology involves an *offline phase* to set up the methods for online use. This offline phase starts with generating disruption scenarios based on historical delay data. Then a *solutions database* is created by solving recovery problems for the disruption scenarios with an optimization-based solution approach. The next step is to train ML classification models for a subset of micro-solutions. It is not practical to train models for all micro-solutions because there are usually too many. Furthermore, it is difficult to train accurate ML models for micro-solutions that are rarely included in the solutions, due to the resulting imbalance in training data. On the other hand, if a micro-solution is included in all of the solutions, there is no need to train classifiers since we fix all such micro-solutions. Therefore, we train ML models for those micro-solutions that are not included in all solutions but in more than a specific fraction of them in the solutions database, which is determined considering the computational resources available. Training inputs consist of disruption scenario features, as well as an additional binary feature reflecting whether that micro-solution is in aircraft recovery solution for that scenario, and binary labels reflecting whether the micro-solution is in the crew recovery solution for that scenario. In the next section, we discuss the offline steps in more detail.

2.3.2 Scenario Generation

Disruption scenarios are generated using the NP_i distributions for each flight, which in turn are derived from historical data. The first step involves clustering the historical days into clusters of similar disruption profiles. A *disruption profile* is defined as a class or label that characterizes the state of the airspace system, and it affects the NP_i

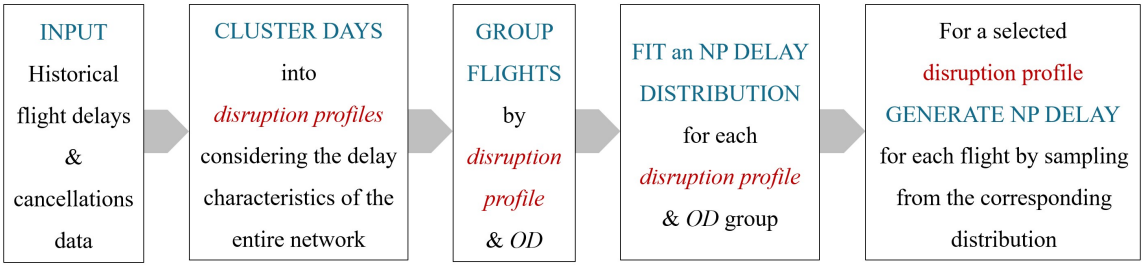


Figure 2-3: Scenario generation procedure

delay for each flight. For example, a disruption profile may represent the inclement weather conditions in the Northeast, decreasing the capacities of the airports in the region. The reduced capacities would result in congestion, consequently leading to delays and cancellations for flights arriving and departing from the congested airports, as well as for other airports due to delay propagation. Appendix A.1 provides the details and results of the clustering process.

In addition to the disruption profiles, the expected delay behavior of a specific flight also depends on the departure and arrival airports. Therefore, after the historical days are clustered into disruption profiles, we group flights by disruption profile and origin-destination information (OD). Then, a log-normal distribution is fit to the NP_i values in each group. Prior research shows that the log-normal distribution is a suitable candidate for delay distributions (Lan et al., 2006). We sample NP_i values for each flight from the distribution corresponding to that particular combination of disruption profile, origin, and destination. Figure 2-3 includes an overview of the scenario generation procedure.

2.3.3 Solutions Database

The next step in the offline phase is to generate solutions for the disruption scenarios to create a solutions database. The mathematical model presented in Section 2.2.4 is solved for each disruption scenario with a selected optimality gap target, and the solutions are saved to the database.

The optimality gap target affects the time an optimization run takes. Generating an optimal solution or setting the target gap very low (e.g., 0.1%) significantly

increases the runtime, thus making it more time-consuming to generate the solutions needed for the ML training process. In our experiments, we noted that most of the micro-solutions in the optimal solutions were also included in the sub-optimal solutions generated by the optimization approach with a larger target gap. More importantly, for many of the micro-solutions, lower quality solutions with optimality gap targets of 10% to 20% were good indicators of whether they should be used in solving the recovery problem for the corresponding disruption. Ultimately, we chose an optimality gap target of 20% to limit the time and computational resources needed for the offline phase to populate the solutions database. This decision speeds up database generation by ten times, on average, compared to setting the gap to 1%. Details on a comparison between different optimality gap targets and their impact on run time and F/O classifier precision are provided in Appendix A.7.

2.3.4 Micro-solution Prediction

We train and use ML models to predict whether or not a micro-solution is in the recovery solution for a given disruption scenario. These problems are called classification problems, where the objective is to decide the class or label of each observation, from a discrete set of two or more labels, based on a set of features (Dodge, 2008). Since a micro-solution is either in the recovery solution or not, this is a binary classification task and these types of classification models are called binary classifiers.

The input used in binary classifier training is generated using the solutions database. It includes the disruption feature set for each data point in the training input and a feature representing the usage of that particular micro-solution in the aircraft recovery solution. The label, which we train the classifiers to predict, reflects the presence of the corresponding micro-solution in the crew recovery solution. Appendix A.2 provides a more detailed view of how disruption information is combined with the scenario solutions to create the training input.

As discussed above, there are alternative types of micro-solutions, like crew duties, duty string assignments or F/Os, that one may consider fixing. The F/O is a smaller micro-solution type than crew duties and duty string assignments, because an F/O

corresponds to exactly two flights assigned consecutively, while there are many crew duties and assignments corresponding to two or more flights. Classifiers can be trained to predict whether a specific F/O will be in the solution of a given disruption or not. There may be several crew duties and assignments that contain a given F/O. When that F/O is fixed by adding the corresponding constraints to the model, the optimizer decides which crew duties and assignments containing that fixed F/O would be used in the final solution.

The number of possible crew duties and duty string assignments is much larger than the number of possible F/Os. Therefore, compared to crew duties and duty string assignments, a lot fewer F/Os need to be predicted in order to predict a sizable subset of the overall solution, and thus a lot fewer classifiers are needed. This suggests that F/Os are a more practical type of micro-solution than the crew duties or the duty string assignments. Additionally, because the same F/O can be a part of several crew duties and duty string assignments, fixing F/Os rather than fixing duties or assigning crews to duty strings gives the optimization approach more options to construct the best crew duties and duty string assignments. Furthermore, our experiments show that, among all three micro-solutions types considered by us, the F/O classifiers provide a greater prediction performance improvement compared to database frequency statistics. The improvement in prediction performance is calculated as the precision of the classifier minus the database frequency of the corresponding micro-solution. *Precision* or *positive predictive value* (PPV) is the ratio of true positives to all positive predictions (Dodge, 2008).For example, for an F/O used in 70% of the database solutions, an ML model with 90% precision would provide a prediction improvement of 20% over the database frequency. Therefore, we selected F/Os as the type of micro-solutions to focus on for predictions. A more detailed comparison is presented in Appendix A.3.

An efficient way to solve airline scheduling and recovery problems is to follow branch-and-bound based solution methods (Vance et al., 1997). Branch-and-bound methods are tree-based search methods that partition the set of feasible solutions into smaller subsets and systematically evaluate the subsets to find the best solution

(Bertsimas & Tsitsiklis, 1997). Partitioning to subsets is performed using a certain set of rules, called *branching rules*. In the crew scheduling and recovery context, a common rule of branching is called *branch-on-follow-on* or *F/O branching*, which partitions the feasible set of solutions based on forbidding an F / O on one branch and fixing it on the other (Vance et al., 1997). Thus, our selection of F/O as the preferred type of micro-solutions to fix can be considered a special type of *greedy* branch-and-bound based solution method using the *F/O branching* rule, where only one branch — the one that fixes the follow-on — is considered.

The classifier output is an estimate of the conditional probability (PRB) of the micro-solution being in the recovery solution for the given disruption. PRB indicates whether a micro-solution is a good fit for the corresponding disruption. A PRB value close to 1 implies that using at least one crew duty covering the corresponding F/O has a strong potential of leading to a good solution.

A crucial hyper-parameter for classifier training is the training criterion, the performance metric the classifier is trained to maximize. The selected training criterion impacts the performance of the adopted methods. We selected *precision* or *positive predictive value* (PPV) as the training criterion. *Precision* is chosen instead of *accuracy*, the ratio of all correct predictions to all predictions, because we consider fixing a micro-solution only when the predicted probability of the micro-solution being in a high-quality solution is high. Therefore, the proposed solution approach focuses only on positive predictions.

Feature selection is crucial for training meaningful ML models, and should reflect the nature of the classification task. Setting the feature set size too large may render the classification task too time-consuming. In Section 2.2.2, we characterized a disruption scenario as the NP_i values for the flights in the entire network. Assuming that this characterization accurately reflects the disruptions, using the entire definition of the disruption scenario as part of the feature set (in addition to the feature reflecting the usage of the F/O in the corresponding aircraft recovery solution), can give accurate predictions, at least in theory, as it uses the entirety of the available information. However, this approach can significantly slow down the classifier train-

ing. During our experiments, we found that we could effectively reduce the number of flights whose NP_i values are considered for each micro-solution through careful filtering. For example, for predicting an F/O micro-solution, the relevant NP_i values could be restricted to only those flights which can connect to at least one of the flights in the F/O. This method of focusing on a subset of flights instead of the entire network significantly accelerates the training process.

Such acceleration of the training process is achieved by focusing on the flights that are temporally and spatially close to the F/O in question. We define a neighborhood network, shown in Figure 2-4, with F/O pairs as nodes and feasible connections to other F/Os as edges and determine the flights in the 1, 2, or 3 hop neighborhoods of each F/O. A feasible connection between F/Os is nothing but a pair of F/Os that share a flight. A 3-hop neighborhood of an F/O includes all flights at most three edges away from that F/O (node in the center). This set is sufficiently large to include almost all flights that may be part of a crew duty that also includes the given F/O, since the number of flights in a crew duty for the flight network used in the computational study was at most 5. The average 1-, 2-, and 3-hop neighborhood sizes for our case study network with 2,780 flights are 25.1, 125.3, and 576.8 flights, respectively — a significant reduction compared to the size of the entire network. Experiments showed that classifier training with the 3-hop neighborhood feature set is at least five times faster than that with the full feature set while the loss in average classifier precision is less than 1%. We call the feature set based on the 1-, 2-, or 3-hop neighborhood flights, the *local feature set*, and the feature set including all flights the *full feature set*.

It is typically not practical to train classifiers for each of the potential micro-solutions because there are many such micro-solution candidates, and because not all candidates are suitable for training high-precision classifiers. A more practical approach is to focus on micro-solution candidates with a frequency in database solutions that justifies the prediction effort. For instance, if a micro-solution is included in less than 5% of the database solutions, training a reliable classifier would be difficult due to the limited number of positive observations since classification methods struggle

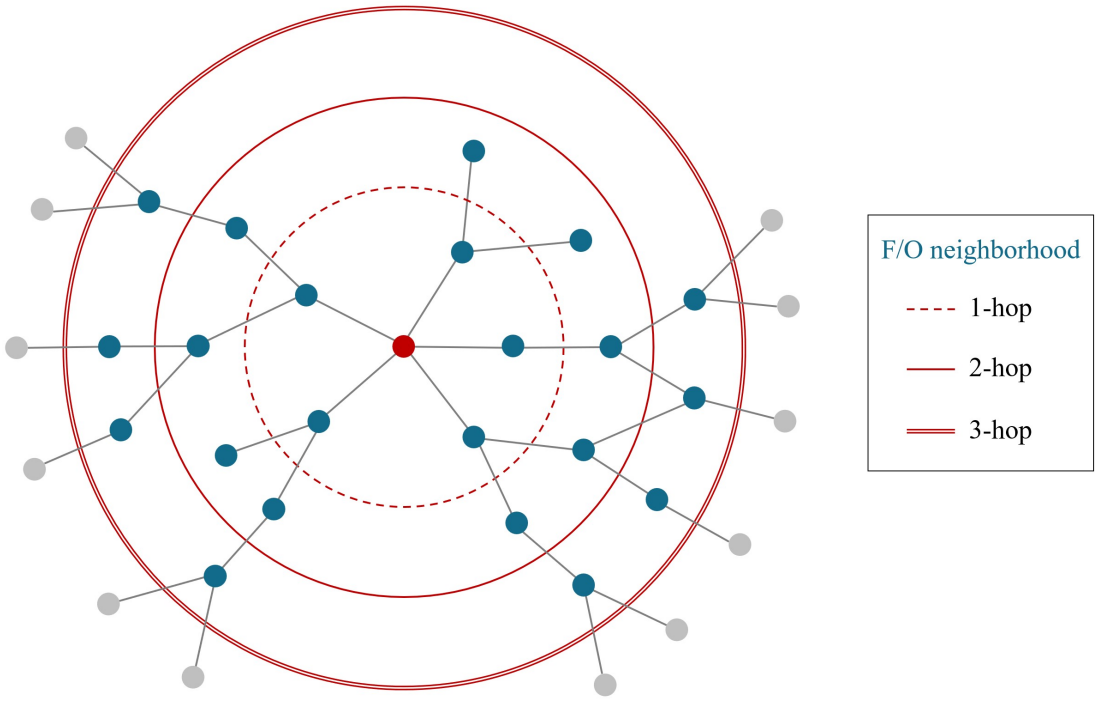


Figure 2-4: Local neighborhoods of an F/O

with unbalanced data. On the other hand, for a micro-solution included in more than half but less than all of the database solutions, it would be relatively easier to train a high-precision classifier. The database frequency interval, which determines the micro-solutions for which we choose to train classifiers, can be selected during the offline phase considering the available computational resources.

Even for micro-solutions in the database frequency interval, there is no guarantee that a high-precision classifier can be trained. There may not be a discoverable pattern justifying the selection of the micro-solution. Specifically, our experiments showed that, in any given solution, up to about half of the micro-solutions in the chosen database frequency interval demonstrated some kind of pattern that can be learned by the classifiers. The remaining showed little to no relationship with the input features. They seemed to be selected only to finalize the solution, that is, to cover flights that are not covered otherwise. Our solution method reflects these characteristics of different micro-solutions. Rather than having classifiers to generate predictions for all micro-solution alternatives, we discover patterns wherever possible,

fix the corresponding set of micro-solutions, and run the optimization model to create the final solution that includes the remaining micro-solutions making up that overall solution.

Training a single classifier per micro-solution alternative helps reap most of the benefits of our approach. However, it is additionally beneficial to train multiple classifiers for the same micro-solution, since differently structured classifiers may be able to capture different aspects of the same classification task. Indeed, there may be several ways to describe why a specific observation belongs to a particular class. Different classification models can discover different unique explanations for the same label prediction — e.g., one classifier may discover a pattern in flights that can connect to the first flight in the F/O, while another classifier may find an alternative explanation involving only the flights which the second flight in the F/O can connect to. Hence, training multiple classifiers for a given micro-solution increases the chance of discovering better patterns and making more accurate predictions for different types of disruptions. A more detailed discussion of the approach involving the usage of multiple classifiers, and the corresponding results, are included in Appendix A.4.

The required number of disruption scenarios, i.e., the size of the training input, increases with the size of the selected feature set. While having a very large number of disruption scenarios may improve prediction accuracy, that is not feasible due to the time constraints in the offline phase. Furthermore, since the feature set size is not too large especially when the local neighborhood feature sets are used, the performance improvement with additional training data beyond a certain point is minimal. We found that depending on the size of the feature set, a training input size as small as 200 may be sufficient. See Appendix A.6 for a detailed discussion on the effects of training input.

2.3.5 ML-guided Solution Space Reduction

After generating the solutions database, one can use simple aggregate statistics to analyze the solutions, determine the usage frequency of the different alternative types of micro-solutions, and reduce the feasible solution space accordingly to accelerate the

solution process. However, the benefits of such an approach are limited as indicated by our results in Section 2.4.3. We do use database statistics to guide our solution space reduction procedure. But, more importantly, we also utilize ML predictions for micro-solutions to enhance the process further. Specifically, we first remove a specific type of micro-solutions, crew duty assignments that correspond to main decision variables, y_s^k in the CRM-APC model, which are never used in database solutions or used extremely rarely (controlled by a parameter, called $LF\_threshold$). This significantly improves the solution time, with negligible effects on the solution quality. We do not remove any F/O micro-solutions, instead we use classifiers to evaluate a subset of them to examine whether they are suitable candidates for the current disruption. This step is where we leverage the ML predictions.

While evaluating the predictions for a micro-solution, in addition to the conditional probability (PRB) calculated by the classifier, we also consider the out-of-sample precision (PPV) performance. Taking only one of these metrics into account yields undesirable results. If only the PRB is considered, it may lead to fixing a micro-solution with a high prediction probability, but calculated by a low PPV classifier, implying that the prediction may not be reliable. On the other hand, if only the PPV is considered, it may lead to fixing a micro-decision with a low PRB, implying that the probability of this micro-solution being in a high-quality overall solution might not be very high. Instead, we introduce a new metric called *prediction confidence* (PC) which is calculated for each classifier prediction as $PC = PPV^{\alpha} * PRB$, where α determines the relative importance of PPV and PRB. When evaluating the predictions, we compare the PC values against the prediction confidence threshold value ($PC\_threshold$), which is a pre-specified value that determines whether the constraints corresponding to a prediction should be added to the model. It is defined as a multiplication because what we are interested in is a compound probability of the final prediction. The α value was set to 1 after evaluating a set of candidate values (Appendix A.8), which implies that for our experiment setup, PPV and PRB are of similar importance. In general, the α value may be airline-specific and may need to be tuned separately for each airline. For different airlines and flight networks, α can

be different than 1.

Now, we present the main steps and the added constraints in the process of creating a solution space tailored to the given disruption. The main decision variables in our model are the assignment variables, y_s^k, which equal 1 if the crew string $s \in S$ is assigned to the crew $k \in K$, 0 otherwise. Hence, the added constraints are defined in terms of y_s^k, even though some of them are related to F/O micro-solution predictions.

1. Remove low-frequency assignments from the solution space by adding,

$$y_s^k = 0 \qquad \forall y_s^k \in Y_0 \tag{2.15}$$

where Y_0 is the set of crew assignments with a low frequency in the database solutions. The low frequency assignment threshold, $LF\_threshold$, to filter such assignments is determined in the offline phase, considering the characteristics of the problem instance.

2. For each F/O candidate that is included in all database solutions or has a trained classifier, if the PC for F/O for the current disruption, is at least equal to the $PC\_threshold$, then fix the F/O by adding,

$$\sum_{i \in f} z_i + \sum_{Y^f} y_s^k \geq 1 \qquad \forall f \in F_1 \tag{2.16}$$

Here, F_1 is the set of F/Os with PC of at least $PC\_threshold$ and Y^f is the set of assignment variables that contain the F/O f. Recall that each F/O is an ordered pair of flights and hence the first summation in Constraints (2.16) is always over two terms. We denote the $PC\_threshold$ value of an F/O f by $PC(f)$. We set $PC(f)$ to be equal to 100% for the F/Os that are included in all database solutions and 0% for F/Os that are neither included in all database solutions nor have trained classifiers. The first summation in Constraints (2.16) is added to provide the model the flexibility to consider assigning one or both of the flights in the F/O to a high-cost reserve crew instead of forcing them both to be consecutively assigned in at least one crew duty.

The version of Constraints (2.16) without this first summation, $\left(\sum_{Yf} y_s^k \geq 1\right)$, led to infeasibilities or low-quality recovery solutions.

3. The CRM-APC model (2.1-2.13), with the added constraints (2.15)-(2.16), is solved using a mixed-integer optimization solver. Algorithm 1 summarizes the overall flow of the solution method.

Algorithm 1 CRM-APC Solution Approach

Initialize:
$disruption \leftarrow \{NP_i, \forall i \in I\}$
Load the aircraft recovery solution
Load the CRM-APC
Set the $PC\_threshold$ value
for $y_s^k \in Y_0$ **do**
    add the constraint $y_s^k = 0$ to CRM-APC
end for
for $f \in F$ **do**
    if $PC(f) \geq PC\_threshold$ **then**
        add the constraint $\sum_{i \in f} z_i + \sum_{Yf} y_s^k \geq 1$ to CRM-APC
    end if
end for
Solve the CRM-APC with the added constraints

The $PC\_threshold$, which is used to evaluate the classifier predictions, affects the run time and solution quality. Setting it too low can lead to lower-quality solutions due to the inclusion of less accurate predictions. On the other hand, setting it too high can cause the feasible solution space to remain too large to solve quickly. The threshold values for different available solution time limits are tuned in the offline phase. Details of this process are provided in Section 2.4.2. By tuning the $PC\_threshold$ value, the user can adjust the solution approach to maximize solution quality generated within a given computational time limit.

2.4 Computational Experiments and Results

2.4.1 Data Sources, Network Description and Pre-processing

The computational experiments are based on operational data from the US domestic flight network with over 2,800 daily flights from a major US carrier. Flight schedules and historical flight delay information were obtained from the airline on-time performance (AOTP) database from the Bureau of Transportation Statistics (BTS, 2020). The training set consists of the disruption scenarios used in classifier training and it was generated using the historical delay data for 15 winter months (December, January, and February) spanning from December 2012 to February 2017 following the procedure outlined in Section 2.3.2. For this research, the airline provided to us the planned crew schedules, including the reserve assignments. See Appendix A.5 for more information on the data pre-processing steps.

2.4.2 Offline Phase

The offline phase is time- and resource-intensive. However, we developed and implemented multiple techniques to accelerate the offline process without compromising performance, which is particularly important when airlines have limited time and resource availability. In particular, we made three major practical decisions that accelerated offline runtimes.

The first decision is related to the selected solution quality level for the solutions database, as expressed in terms of the optimality gap target. In these experiments, the optimality gap target was selected as 20% as discussed in Section 2.3.3.

The second decision was to generate a single consolidated solutions database and a single set of classifiers spanning all ten disruption profiles instead of generating separate databases and training separate classifiers for each. As mentioned in Section 2.3.2, the flight operation days in the historical database are clustered into ten distinct disruption profiles (shown in Appendix A.1). Instead of generating N disruption scenarios for each disruption profile and creating separate databases and classifiers,

we generated N/10 scenarios for each profile and then used the resulting N scenarios to generate a single consolidated solutions database and train a single set of classifiers, involving both interpretable and non-interpretable classifiers, for a subset of micro-solutions to predict their usage in high-quality recovery solutions. The resulting classifiers can be used effectively for any real disruption scenario, regardless of the disruption profile. This decision also leads to a 10-fold acceleration as compared to generating separate databases for each disruption profile. Taking into account the computational requirements to solve each scenario and the fact that using 1,000 or 1,500 scenarios provided a less than 1% improvement in precision, we decided to set the number of disruption scenarios, N, to 500 in the computational study. Details are provided in Appendix A.6

The last decision concerns the feature set used in classifier training. We need a feature set comprehensive enough to contain the most relevant information, but also relatively small to train classifiers fast enough to meet the time constraints in the offline phase. This is achieved by using the n-hop neighborhood feature sets, introduced in Section 2.3.4. Classifiers based on n-hop neighborhood feature sets yield only slightly lower precision results compared to classifiers trained with the full feature sets (Appendix A.6). The size of the neighborhood of the local feature set, n, is set to 2 in the computational study.

We used the Optimal Classification Tree (OCT) method, a decision tree classifier with parallel splits provided in the module developed by InterpretableAI (2020), as one of the classification methods for the computational study because they provided precision values at least as good as any other method and because tree classifiers provide interpretable results, as will be discussed in Section 2.4.5. We also use Random Forest and XGBoost, which are tree-based ensemble methods in which multiple decision trees are trained to build more accurate prediction models.

As argued in Section 2.3.4, the presented framework can additionally benefit from training multiple classifiers for each candidate F/O to decide whether to fix it. This can increase the probability of capturing different aspects of the classification task. The set of classifiers trained for each F/O included two parallel tree models (differ-

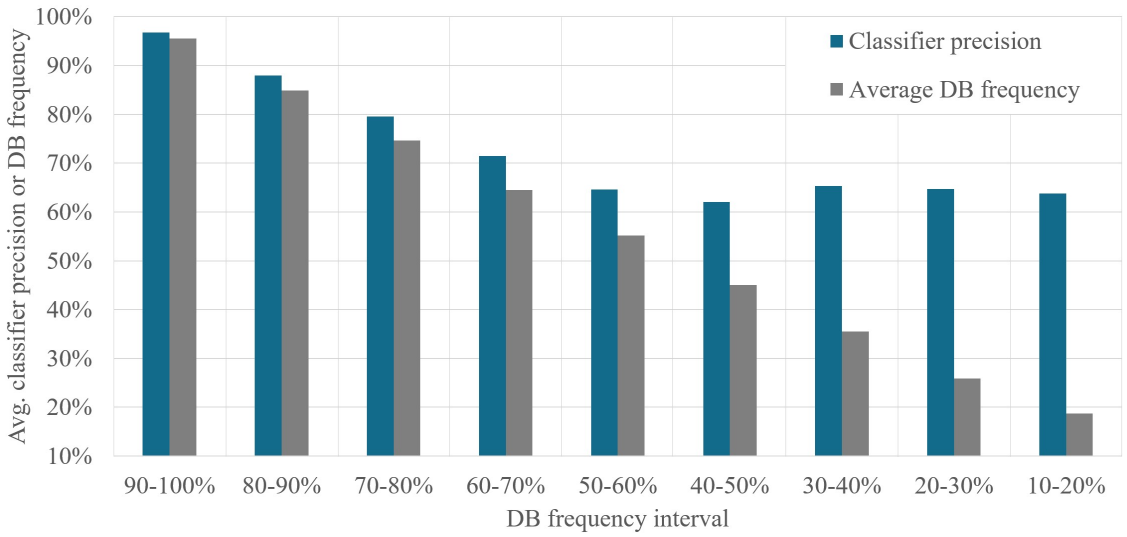


Figure 2-5: Classifier precision performance w.r.t. DB frequency intervals

ing in the maximum tree depth parameter, set to 5 and 10), one Random Forest, and one XGBoost classifier. Tree classifiers are included for their interpretable structures, while the Random Forest and XGBoost methods are chosen for their precision performance (Breiman, 2001, T. Chen & Guestrin, 2016).

Figure 2-5 shows the classifier precision performance for F/Os with a precision greater than 50%, as a function of the database frequencies. For higher values of database frequencies, e.g. >90%, the precision improvement over the DB frequency is limited. But for the F/Os with lower DB frequencies, e.g. <50%, the precision improvement becomes more significant. For a subset of F/Os with a DB frequency between 10% and 20%, trained classifiers provide a precision of ∼65% which is a 45 percentage point improvement. These strong results justify the use of our ML-based approach.

The last step of the offline phase is the calibration of the *prediction confidence* threshold, $PC\_threshold$, and low-frequency assignment threshold, $LF_threshold$, for the day of flight operations. The best threshold values depend on the available solution time. We focused on time durations of 5 minutes and shorter as solution time limits in our experiments. Our experiments demonstrated a one-to-one match between solution time availability and the most appropriate specific threshold val-

ues to achieve the best solution. The longer the solution time, the higher the best threshold value for $PC\_threshold$ since it allows considering a larger set of solution alternatives. On the other hand, the longer the solution time, the lower the best threshold value for $LF\_threshold$ since it allows considering a larger set of solution alternatives. The threshold values to be used under different solution time limits are determined in the offline phase based on a set of out-of-sample disruption scenarios, called the calibration set and evaluated for their performance using the online test instances. In computational study, assignments included in less than $<0.5\%$ or less than 1.5% of the solutions are removed depending on the solution time limit. See Appendices A.9 and A.10 for more details.

2.4.3 Results

The disruption scenarios used for calibration and testing are based on actual January flight operation days from the years 2018 and 2019, respectively. Thus, we have 31 scenarios in the calibration set and a separate set of 31 scenarios as the test set. Each disruption scenario maps into an individual day of flight operations, and the recovery period spans from the scheduled departure time of the first flight in the current day's schedule until right before the scheduled departure of the first flight in the next day's schedule. All experiments are carried out on a desktop computer equipped with an Intel i9-13000K CPU and 64 GB of memory using Gurobi 10.0 (Gurobi Optimization Inc., 2020) as the integer and mixed-integer optimization solver. We compared the solutions found by the following five methods.

1. *Default*: The CRM-APC model is sent directly to the Gurobi optimizer with the specified solution time limit, without any a priori reduction in solution space.

2. *Reduced*: All 0-frequency assignments are removed from the solution space before sending it to the Gurobi optimizer.

3. *Swap*: Inspired by a common solution space reduction heuristic (Yu et al., 2003), this method limits optimization to include: A) all broken crew schedules; plus

B) two unbroken crew schedules with swap opportunities with each of the broken crew schedules. All remaining crew schedules are left untouched to make the problem tractable. The number of swap opportunities between two crew duties is defined as the cardinality of the set of ordered pairs of flights that can connect with each other where each flight belongs to one of the two crew duties and the two flights belong to different crew duties from each other.

4. *Freq*: It starts with the reduced solution space defined in the *Reduced* method, and further reduces the solution space on the basis of F/O frequencies in the database. The F/Os occurring in the database with a frequency above a certain threshold are fixed. This threshold is tuned for each solution time limit.

5. *ML*: It also starts with the reduced solution space defined in the *Reduced* method, and further reduces the solution space based on PC values calculated by the trained classifiers, as described in Section 2.3.5. The threshold above which the F/Os are fixed $PC\_threshold$ is tuned for each solution time limit.

The cost parameters used in the experiments are given in Table 2.1. The cost of deadheading crews to the next day's duty start location is based on the values used in prior recovery studies (Petersen et al., 2012). The cost of crew delay per minute is derived from the average crew duty costs as estimated by the FAA (2020). Regular and high-cost reserve crew pay per hour is assumed to be 25% and 50% higher than standard crew duty pay rate. The overall passenger delay and disruption cost estimates were obtained from the cost of passenger delay per minute, planned passenger itinerary flow information and aircraft seat configuration information. Of these, the cost of passenger delay per minute is based on the analysis conducted by Cook and Tanner (2011). The aircraft seat capacity configuration information was obtained from the airline's website. The estimated passenger itinerary information was obtained from Barnhart et al. (2014).

Figure 2-6 compares the solution quality achieved by the methods listed above. The underlying network corresponds to a Saturday in January 2019. It has 2,870 flights, ten crew bases, and 1,200 daily crews. All flights belong to the same fleet

Parameter	Value
Cost of a crew deadheading to the next day's duty start location	$2,000
Cost of passenger delay per minute	$30
Cost of crew delay per minute	$12

Table 2.1: Cost parameters used in the computational study

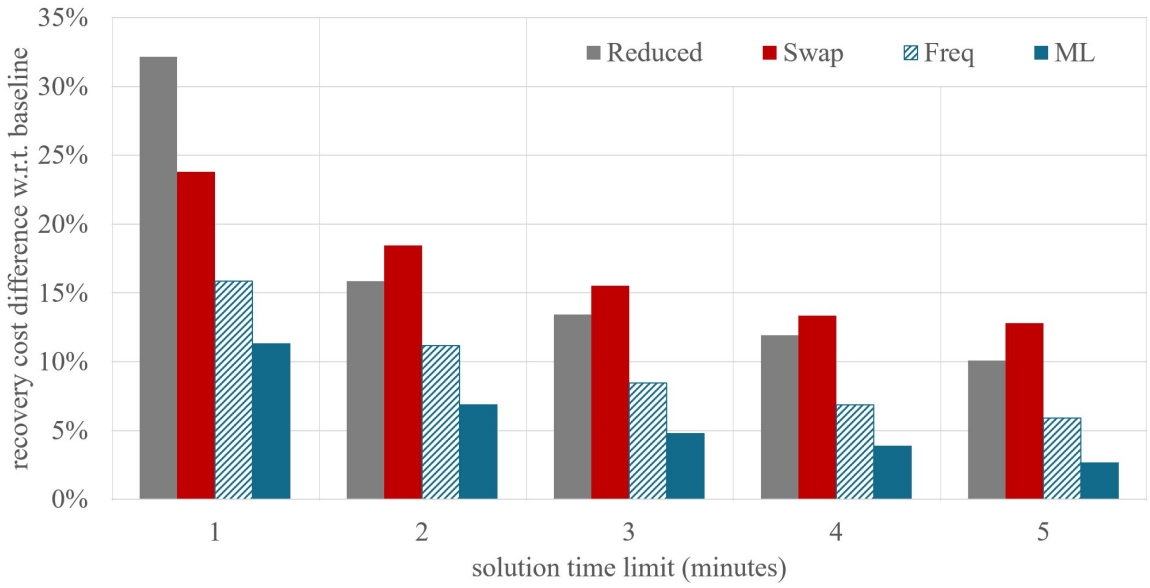


Figure 2-6: Solution quality comparison

family and hence all crew are certified to operate all flights in the network. This makes our problem instances more challenging to solve, compared to other airlines where the crew cannot be used interchangeably across the flight networks of different fleet types. Crew recovery instances for these other airlines allow decomposing the overall problem instances into multiple, smaller separable instances, one for each fleet family, and solving them separately. We had access to the actual planned crew schedules and reserve crew information provided by the airline. Table 2.2 shows the average recovery cost differences with respect to the baseline solutions under different solution time limits. Baseline solutions are defined as those given by the default approach with an optimality gap target of 0.1% and a run time limit of 2 hours. The average optimality gap target achieved was ∼1%.

With the *Default* approach, which sends the optimization problem directly to the

Solution time limit (minutes)	Default	Reduced	Swap	Freq	ML
1	N/A	32.17%	23.78%	15.85%	11.35%
2	N/A	15.86%	18.44%	11.18%	6.89%
3	N/A	13.44%	15.54%	8.47%	4.80%
4	N/A	11.92%	13.36%	6.87%	3.89%
5	N/A	10.08%	12.80%	5.90%	2.70%

Table 2.2: Average recovery cost difference with respect to the baseline solutions (31 test scenarios)

Gurobi solver, it was not possible to generate even a feasible solution in 5 minutes. The *Reduced* method generates feasible solutions but has much higher costs than the other database-driven *Freq* and *ML* methods. The *Reduced* method achieves its lowest-cost solution at the end of 5 minutes with a ∼10% average recovery cost difference.

The *Swap* method is found to be suitable for generating fast solutions. Notably, it produces better solutions than the *Reduced* method for a 1-minute runtime limit. However, the average solution quality achieved in longer time limits does not improve as quickly as it does for the *Reduced* method making it perform worse than the *Reduced* method for time limits of 2 minutes and longer. The FO-based methods, *Freq* and *ML*, perform significantly better. A key difference between the *Swap* method and the FO-based methods, *Freq* and *ML*, is that the *Swap* method fixes larger micro-solutions, like crew duty assignments, and hence loses a lot of flexibility, while the F/O based methods also reduce the solution space, but retain a lot more flexibility, allowing for improved performance.

The *Freq* method outperforms *Default*, *Reduced* and *Swap* methods for all five tested solution time limits (1, 2, 3, 4 and 5 minutes). When the solution time limit is 1 minute, it fixes many low-frequency F/Os to sufficiently accelerate the solution process. Those low-frequency F/Os lead to poor solution quality in many disruption scenarios producing an average recovery cost difference of over 15%. The solution quality improves with longer runtime limits, because only the relatively higher fre-

quency F/Os are fixed, and achieves under 6% recovery cost difference at 5 minutes limit.

Our ML-based method outperforms all other methods for all tested solution time limits. The *ML* method finds solutions with recovery cost differences of below 12% in 1 minute and below 3% in 5 minutes. In fact, the ML-based solutions are around two times better than the next best-performing method for all solution time limits of 2 minutes and longer.

We calculated an approximate monthly savings estimate for the network above to quantify the recovery cost savings our ML-based approach could provide over other approaches. Assuming a solution time limit of 5 minutes for all methods, the annual savings over the next two best-performing methods, *Freq* and *Swap*, are ∼$5.5 million and ∼$21 million, respectively. When the solution time is limited to 2 minutes for all methods, the annual savings over the next two best-performing methods, *Freq* and *Swap*, are ∼$8 million and ∼$20.5 million, respectively. These figures are adjusted for inflation to 2019 dollars.

2.4.4 Solution Analysis

We conducted an F/O overlap analysis to understand some of the factors that determine the quality of the solution generated by the *ML* method. The performed analysis includes ML-based and baseline solutions to 62 disruption scenarios (combining the calibration and test sets). ML-based solutions were generated under a 5-minutes solution time limit. The diagram in Figure 2-7 shows the relative solution quality and F/O overlap of individual solutions, which are represented by dots. A 50% overlap means that half of the F/Os in the baseline solution are also in the final solution obtained using the combination of optimization and ML. There seems to be a linear relationship (indicated by the dashed line) between the solution quality and the F/O overlap. As the F/O overlap increases, the recovery cost difference with respect to the baseline decreases. It is interesting to see that in our experiments, an F/O overlap of around 50-60% was sufficient to generate similarly high-quality solutions as the baseline solutions. This is due to the combinatorial nature of the airline recovery

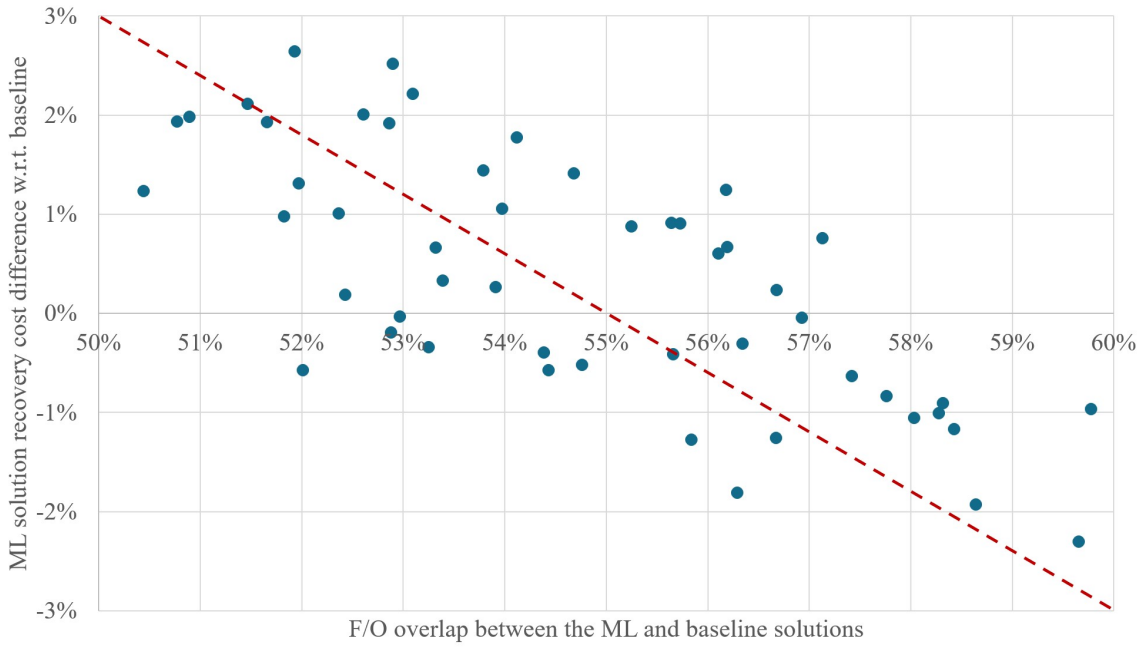


Figure 2-7: F/O overlap vs. solution quality analysis

problems. While certain F/Os are crucial for maintaining the solution quality for a given disruption, there are also many F/O alternatives that complement the solution without affecting the solution quality significantly.

There were many cases in which the quality of the solution found by the ML method was superior to the baseline. This is due to the fact that the baseline solutions have an average optimality gap of ∼1%, which means that there is still room for improvement in the quality of the solution. The ML method seems to leverage this, due to its accuracy in solution space reduction and the resulting acceleration in the solution process.

To understand the differences between the baseline and ML solutions, we analyzed their solutions focusing on certain aggregate statistics of the individual F/Os used in the solutions. All F/Os used in at least one of the baseline or ML solutions are included in this analysis. The frequency of the corresponding F/Os in the solutions database is retrieved. Figure 2-8 shows, on average, how many F/Os within a given database frequency interval are used in the corresponding recovery solutions created by each method — *Default* and *ML*. The average F/O frequencies in the baseline

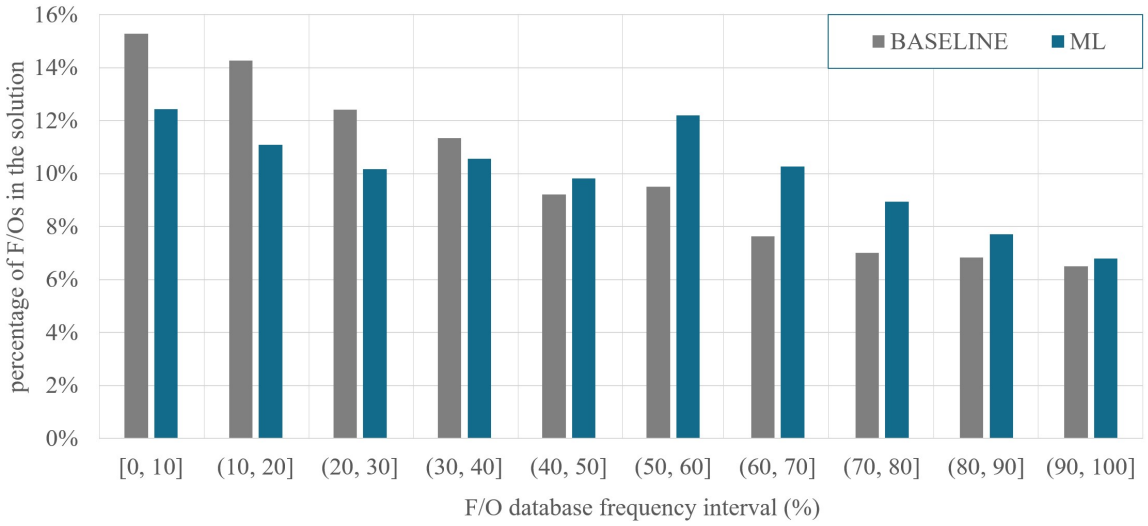


Figure 2-8: F/O frequency distribution with respect to solutions database statistics

and ML solutions are 41.6% and 46.2%, respectively. The results show that the ML solutions rely more on the more frequent F/Os as compared to the baseline solutions. The *Default* method considers all F/Os and uses those most suited for the given disruption. This leads to solutions that include rarely used F/Os. The *ML* method, on the other hand, prioritizes high-precision F/Os, which usually do not have very low database frequencies.

One could argue that a solution is more flexible if it has more F/Os with higher database frequency or fewer F/Os with lower database frequency. A higher database frequency indicates that the corresponding F/Os can be used for a broader range of disruptions. In comparison, a lower database frequency suggests suitability for only a limited range of disruptions. From this perspective, we expect the ML solutions to be more robust than the baseline solutions. The solution robustness is explored further in Section 2.5.2.

2.4.5 Classifier Insights

One benefit of using tree-based classification methods is that they provide interpretable results. Solution space reduction based on classifier predictions does not necessarily require interpretability, but having interpretable classifiers offers practi-

cal insights into why certain recovery decisions are made. By only considering the flight delay information in the local neighborhood of the F/O, the trained classifiers help to discover and understand some of the latent factors that may affect the delay performance of the entire network. Without adequate optimization-based recovery tools, airlines often resort to manual recovery approaches, leveraging decision support systems. In such a setting, high-precision classifiers can be used to develop guidelines for specific recovery measures during the day of operations.

The solution method presented in Section 2.3 involves both interpretable and non-interpretable classifiers. It was designed in this way to maximize the extent of solution space reduction. ∼95% of all F/Os suggested to be fixed by our ML-based crew recovery solution method have readily available interpretable classifiers that explain the rationale behind fixing the F/O. Moreover, we also conducted experiments exclusively using interpretable classifiers. We focused on the solution quality achieved by the ML method in 5 minutes. With the presented structure of the solution method, we generate recovery solutions with 2.70% higher cost than the baseline solutions as presented in Section 2.4.3. With exclusively interpretable classifiers, the ML method can generate recovery solutions with 2.91% higher cost than the baseline solutions. These results suggest that by only using interpretable classifiers, the presented ML method can still generate high-quality recovery solutions.

Figure 2-9 shows a simple tree classifier example. A tree classifier is a supervised learning-based classification method that has a hierarchical tree-like structure consisting of nodes and branches to classify observations. Relevant information on the flights considered by the classifier is given in Table 2.3. The topmost node is the root node where the algorithm starts. The numbers 0 and 1 correspond to the prediction label at that node. The nodes with no outgoing branches are the leaf nodes corresponding to the predictions and are shaded red or blue for 0 and 1 predictions, respectively. The remaining nodes are internal nodes. All non-leaf nodes are shaded white. The percentage values on the nodes are the prediction probabilities for the leaf nodes and the node composition ratios for the others. The saturation of the red or blue shades on the leaf nodes reflects the prediction probabilities, such that the darker the shade,

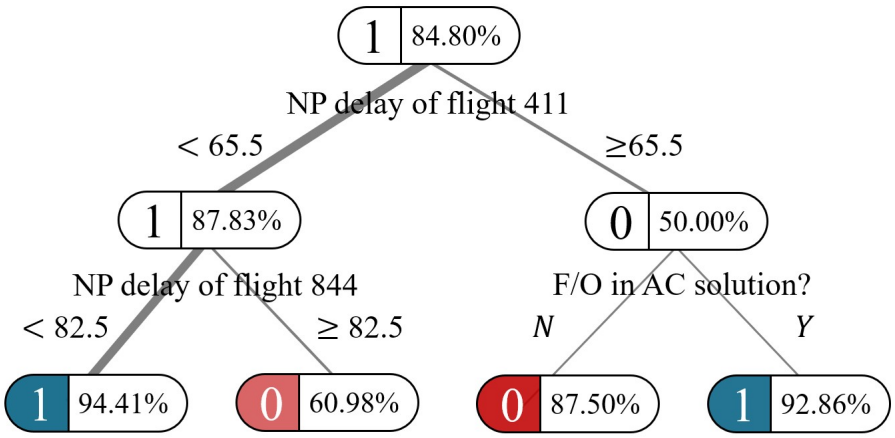


Figure 2-9: Classifier trained for the OMA-MDW-MCA follow-on pair

the higher the prediction probability. The thickness of the lines connecting the nodes represents the fraction of observations from the parent nodes that fall into the child nodes.

The database frequency of this specific F/O, representing the consecutive assignment of flights 411 (OMA-MDW) and 844 (MDW-MCA), is 84.80%, while the classifier achieves a precision of 94.51%. When we analyze the resulting classifier, we see that the only disruption-related information considered is the NP_i information of the flights in the F/O. It also checks whether the F/O is included in the aircraft (AC) recovery solution. If the flights in the F/O, 411 and 844, do not delay beyond 65.5 and 82.5 minutes, respectively, the classifier suggests connecting 411-844 in the recovery solution to the current disruption regardless of whether the same connection is being utilized in the aircraft recovery solution. If the first flight is delayed beyond 65.5 minutes, the connection is still considered to be a good option for the cases where it is included in the aircraft recovery solution.

Flight ID	Origin	Destination	Departure_time (in UTC minutes)	Arrival_time (in UTC minutes)
411	OMA	MDW	855	945
844	MDW	DCA	980	1075

Table 2.3: Flight information for the classifier example in Figure 2-9

Different classifiers focus on different types of information. Figure 2-10 and the corresponding Table 2.4 demonstrate a classifier that focuses on flights that are not part of the F/O but disregards the aircraft recovery solution. It is trained to predict whether the 1971-2770 connection (OAK-MDW-MEM) should be used in the recovery solution. This classifier achieves a precision of 100% despite the fact that the database frequency is 48.73%, indicating strong predictive performance. Interestingly, the classifier does not use the disruption information of the first flight, but focuses on flights 2224 (SLC-MDW) and 2788 (MDW-CLT). The classifier strongly suggests avoiding the 1971-2770 connection (with 100% predicted probability) if the delay of 2224 exceeds roughly 2.5 hours. If that is not the case, the classifier implies that the 1971-2770 connection becomes the best option (with 94.87% predicted probability) when 2788 is delayed beyond 34.5 minutes and if the delay of 2770 is less than 5.5 minutes. One possible interpretation is that for the 1971-2770 connection to be utilized, it has to be efficient in terms of additional time beyond the minimum connection time between flights during the day of operations. If the second flight, 2770, does not have an NP delay beyond 5.5 minutes, the time between the arrival of the first flight, 1971, and the departure of the second flight, will be a very short, unless 1971 arrives significantly early. This is because the slack time between the flights, based on the planned schedule, is already negative, as the minimum connection time required is 30 minutes while the connection time based on the planned schedule is 20 minutes.

Flight ID	Origin	Destination	Departure_time (in UTC minutes)	Arrival_time (in UTC minutes)
1971	OAK	MDW	1315	1555
2770	MDW	MEM	1615	1715
2224	SLC	MDW	1390	1565
2788	MDW	CLT	1620	1720

Table 2.4: Flight information for the classifier example in Figure 2-10

Finally, there are classifiers that consider disruptions to flights in the F/O as

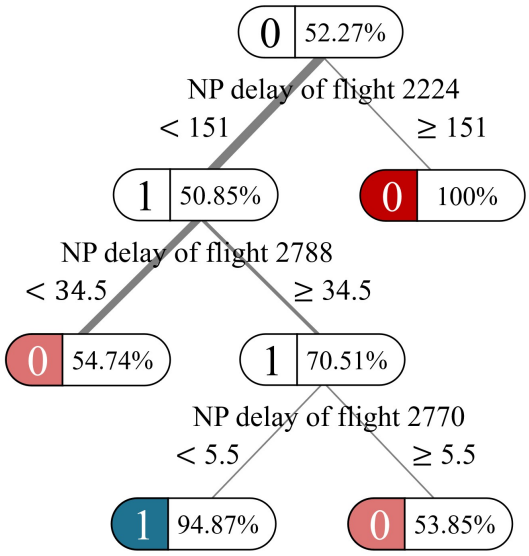


Figure 2-10: Classifier trained for the OAK-MDW-MEM follow-on pair

well as other flights, while also considering the aircraft recovery solution. Figure 2-11 includes such an example. The corresponding information is provided in Table 2.5. The database frequency is 86.13% while the precision is 92.31%. What we are trying to predict is the usage of the connection 1783-2543 (DAL-LGA-MDW) in the crew recovery solution. The classifier suggests connecting 1783-2543 as long as the corresponding flights are not delayed significantly. If the second flight, 2543 (LGA-MDW), is delayed beyond 69.5 minutes, the usage of the F/O in the aircraft recovery solution determines the classifier suggestion.

Flight ID	Origin	Destination	Departure_time (in UTC minutes)	Arrival_time (in UTC minutes)
1783	DAL	LGA	1260	1455
2543	LGA	MDW	1495	1655
749	DTW	MDW	950	1020
2132	LGA	MDW	1360	1515

Table 2.5: Flight information for the classifier example in Figure 2-11

What makes the above classifier examples interpretable is the relatively simple structure of the resulting models. There is a trade-off between classifier interpretabil-

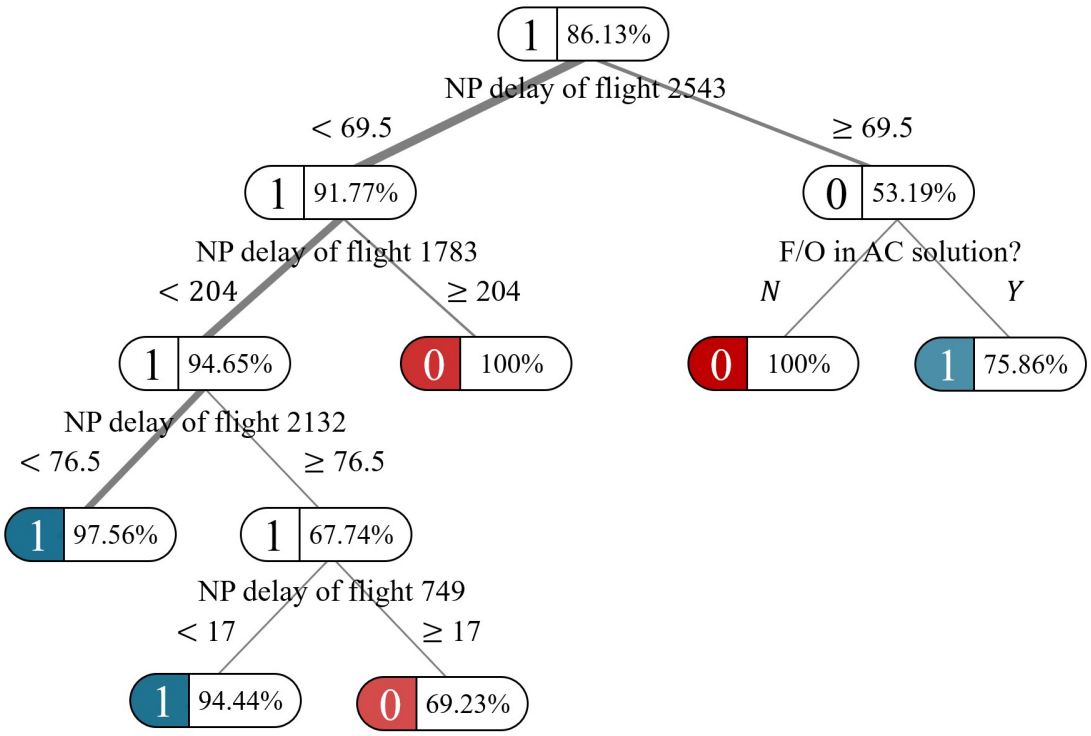


Figure 2-11: Classifier trained for the DAL-LGA-MDW follow-on pair

ity and precision performance. Figure 2-12 shows the average classifier precision values with different classification methods including *Optimal Classification Trees (OCT)*, *Random Forest (RF)*, and *XGBoost (XG)*. OCT-5 and OCT-10 are OCT classifiers using tree depth parameter values of 5 and 10 respectively. XG and RF methods do not create interpretable models. OCT-5 generates more interpretable models than OCT-10, since a tree with a depth of 10 is more difficult to interpret than simpler trees such as those shown in Figures 2-9, 2-10, and 2-11.

The variable importance analysis of the F/O classifiers with high precision revealed that many relied on information from flights that were not in the corresponding F/Os. Information about the delays related to the flights in the F/O and the remaining flights have an importance of 16.4% and 81%, respectively. The information on whether the corresponding F/O was included in the aircraft recovery solution has, on average, 2.6% importance.

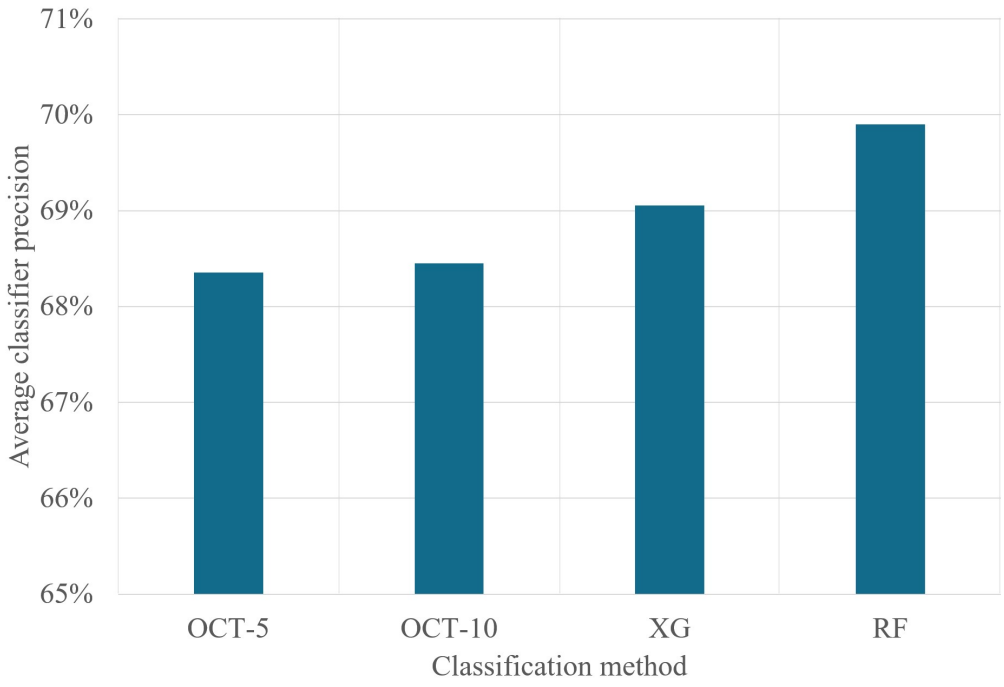


Figure 2-12: Classifier precision performance with different classification methods

2.5 Extensions

2.5.1 Multi-Threshold Search

When a disruption occurs, airline recovery teams act quickly to resolve the issue as soon as possible. For moderate to severe disruptions, considerable resources, including multiple staff members and computers, are often allocated to tackling the most urgent problem. Our proposed approach performs even better under such circumstances. Depending on the number of machines/processors dedicated to the recovery effort, two or more $PC\_threshold$ values can be selected. Each value corresponds to a separate optimization run with the same solution-time limit. We can run two or more optimization instances in parallel with different threshold values and pick the best solution. We call this the *Multi-Threshold Search (MTS)* method. Our experiments show that this method could improve solution quality even when only two parallel runs are performed, because the threshold values that lead to high-quality solutions for different disruptions can also be different. Picking two or more threshold values and running the optimization jobs in parallel increases the probability of finding the

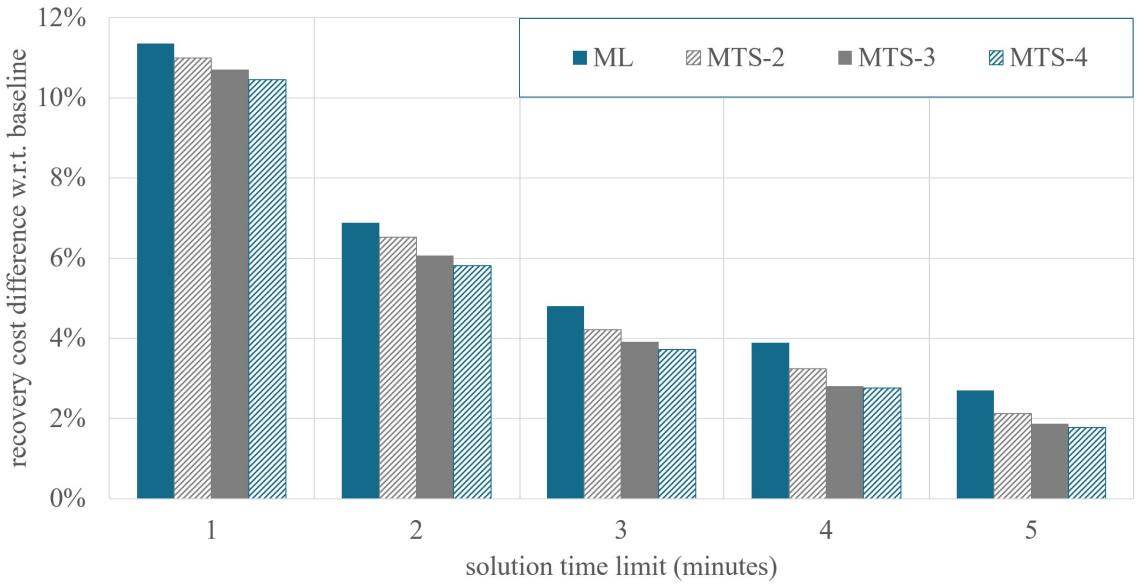


Figure 2-13: MTS method solution quality performance comparison

most effective solution space reduction strategies.

The results shared in Section 2.4.3 reflect the case in which a single threshold value is used. When multiple processors or threads are available on the day of operations, we can apply the *MTS* method and run parallel instances with different threshold values. Figure 2-13 shows the effects of running 2 to 4 parallel instances on the solution quality for the network with 2,870 flights used in Section 2.4.3. The resulting solution quality levels are given in Table 2.6. Compared to the best performance of 2.70% provided by the *ML* method using a single threshold, the *MTS* method using 4 parallel runs generated solutions with a recovery cost difference of 1.78% with respect to the baseline solutions that took up to 2 hours to generate.

2.5.2 Stochastic Evaluation

The results presented in Section 2.4.3 correspond to the cases where the recovery action is taken once for the disruption period and not modified again. This is reasonable only when we have accurate delay predictions and can make our decisions accordingly. But in practice, the recovery solutions are updated several times a day, because the predictions are never 100% accurate. Therefore, we developed a simulation procedure

Solution time limit (minutes)	ML	MTS-2	MTS-3	MTS-4)
1	11.35%	10.99%	10.71%	10.45%
2	6.89%	6.52%	6.07%	5.81%
3	4.80%	4.21%	3.91%	3.72%
4	3.89%	3.24%	2.81%	2.76%
5	2.70%	2.12%	1.87%	1.78%

Table 2.6: Average recovery cost difference for the MTS method with respect to the baseline solutions

to investigate the performance of the proposed methods in a more realistic manner. We now relax the strong assumption of accurate delay predictions, and respond to the updated delay information by allowing the recovery decisions to be altered several times during the day of operations. The objective is to simulate the actual recovery processes and evaluate the performances of the baseline and ML-generated initial solutions under uncertainty and recurring updates.

Figure 2-14 depicts the iterative process for delay predictions and recovery decision updates. The day of operations is divided into six stages with each stage being four hours long. First, an initial recovery solution is created based on the predictions available at the start of the day, i.e., at node 0. Then, at the end of each stage, airlines update their delay predictions for the remaining flights that day and then modify the recovery solutions accordingly. There are five update points after the initial recovery solution. We expect airlines to have more accurate delay predictions for flights departing within a short time period, and the accuracy of the predictions to decrease for the flights that are further out into the future. We also assume that the delay predictions for the flights departing during the first half of each four-hour stage are more accurate than those for the flights departing in the second half. This reflects the fact that, while airlines may receive continuous updates regarding the disruptions, they may still want to avoid re-scheduling operations too frequently. We implemented a parameter, $PredictionNoiseRatio$, to quantify the inaccuracy in delay predictions at each update point for flights departing in the future.

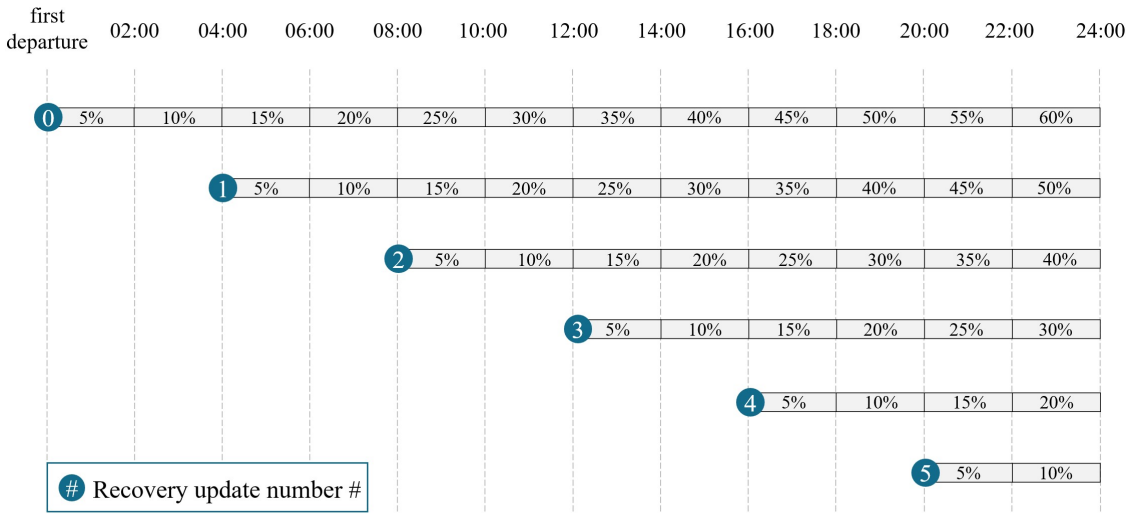


Figure 2-14: Rolling horizon recovery simulation

Predicted NP_i values are calculated as a weighted average of the actual realized delay value and a noise term, where weights are determined based on the remaining time until the flight's scheduled departure time. Let us define the notation needed for defining the formula for predicted NP_i before providing the formula. Let $RealizedNP_i$ be the actual realized NP delay value of flight i on that day, and let $NoiseNP_i$ be the NP delay for that flight obtained from a randomly selected day in the historical data (for the same day-of-week). Let $PredictionNoiseRatio$ be the weight that determines the extent of added noise. Finally, let $LookAhead$ be how far each time period is from the recovery update point and is measured in number of 2-hour periods. Then the formula for calculating noisy NP_i values is $RealizedNP_i * (1 - PredictionNoiseRatio * LookAhead) + NoiseNP_i * PredictionNoiseRatio * LookAhead$.

Figure 2-14 provides an example. When the prediction noise ratio is set to 5%, the NP_i predictions at the recovery update point 1 (i.e., node 1) for the individual flights scheduled to depart between 8:00 and 10:00 hours (i.e., the first half of the time period between nodes 2 and 3 in Figure 2-14) will be calculated as $RealizedNP_i * (1 - 0.05 * 3) + NoiseNP_i * 0.05 * 3$ since the period between 8:00 to 10:00 is three 2-hour periods away from the update point 1 ($LookAhead = 3$).

In the implemented simulation, the flights assigned to a crew and scheduled to

depart within the first 4-hour period cannot be unassigned during any of the later recovery updates. When a crew duty becomes infeasible due to the crew duty legality rules within the first 4-hour period, a high-cost reserve crew is assumed to cover the corresponding flights.

We run the simulation with the $PredictionNoiseRatio$ value of 5% for the 2,870 flight network used in the computational study after five recovery updates. At each point, the method used to update the recovery decisions is the same method (it involves solving the CRM-APC directly using the Gurobi optimizer with an optimality gap target of 2%) for default and ML-based solutions so that we can compare the initial solutions' performance fairly. The *Default* method for the initial solution is run with a two-hour runtime limit and with a target optimality gap of 0.1% while the *ML* method is run with a runtime budget of 5 minutes.

The results show that even though the initial baseline solution was 2.70% better than the initial solution found by the ML-based approach, the final recovery costs calculated after completing the simulation procedure are very similar. In this set of experiments, the average final recovery cost of the solutions generated by our ML-based method was actually ∼0.5% less than that of the baseline solutions. This is an interesting result considering that the *Default* approach runs for almost two hours on average, while the ML-based approach runs only for 5 minutes. One of the reasons for this performance difference is that, since the *Default* approach includes the entire feasible solution space, it generates solutions highly specific to the disruption input, which is the initial set of NP_i predictions. Since the realized NP_i values differ from the predictions, solutions are more severely affected, resulting in higher actual costs. The ML-based approach, on the other hand, relies on probabilities; hence the resulting solutions are not over-optimized for the given disruption input, making the resulting solutions more flexible. Consequently, the difference between the NP_i predictions and the realized NP_i values does not increase the actual costs as significantly.

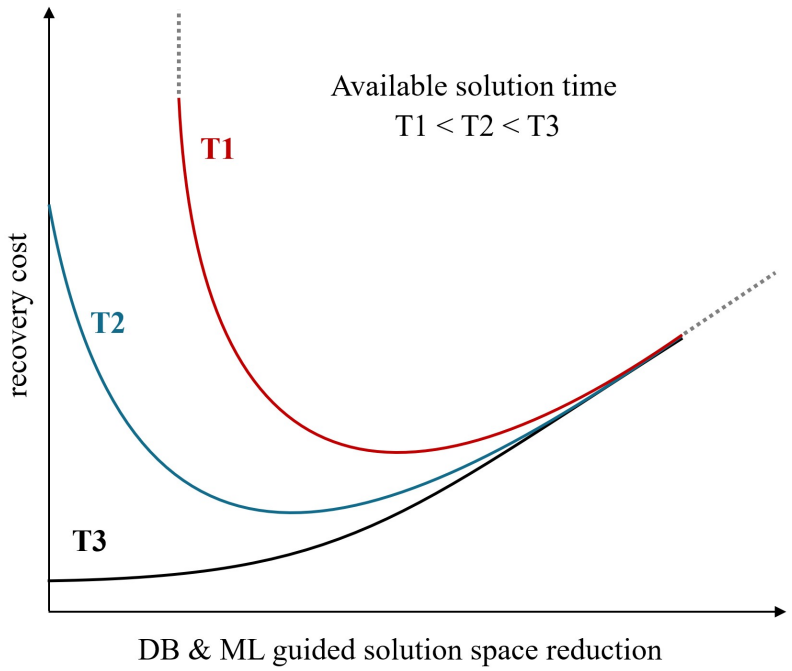


Figure 2-15: Relationship between recovery cost and solution space reduction.

2.6 Conclusion

Airlines that incorporate optimization methods into their recovery processes generally adopt an approach that reduces the size of the problem by only including a limited number of flights and crews in the solution space. However, these procedures typically use simple heuristic rules for reducing the problem size. Consequently, the resulting solutions can be far from optimal. Our experiments confirm that reducing the feasible solution space significantly speeds up the optimization process. More importantly, our results demonstrate that ML techniques can reduce the size of the problem without sacrificing significantly the solution quality. We determine a solution space tailored for the given disruption by predicting and fixing some of the micro-solutions. Our methods can create high-quality solutions in under 1 to 5 minutes, depending on the size of the underlying flight network.

For each problem instance and the available solution time T, there appears to be a relationship between the solution quality and the database (DB)-guided & ML-guided solution space reduction. A conceptual depiction of this general behavior is

given in Figure 2-15. Empirical evidence suggests that when T is sufficiently large, solution space reduction does not help, but for most practical cases, there exists a reduction strategy that provides the best solution quality. The framework presented in this study relies on finding good solution space reduction strategies, and it also has the capability to adapt to the available solution time limit, a property that other heuristic approaches lack.

The time and other resources required for the offline phase is a drawback of the presented framework. To ensure the practicality of the approach, we developed procedures to significantly accelerate the offline phase without sacrificing the quality of the solutions.

Our experiments showed that classifiers trained for follow-on (F/O) pairs achieve higher precision levels than those trained for crew duty assignments, helping to find better solutions in limited timeframes. Consequently, we picked F/Os as the type of micro-solutions to predict and fix in this framework.

The best threshold value for a given solution time differed for individual disruptions. This fact motivated a *Multi-Threshold Search (MTS)* method, in which several instances of the same problem are solved in parallel with different threshold values to filter and fix F/Os. Even with only two parallel runs, the MTS approach demonstrated considerable improvements. This method is consistent with airline practices because, in many cases, multiple computers and recovery personnel are dedicated to solving the given disruption.

We evaluated the performance of the ML-based solutions under incomplete information settings with a simulation-based approach to better reflect real-life recovery operations and quantify the impacts. Our experimental results using our ML-based approach were found to be similarly (in fact slightly more) robust to inaccurate flight delay predictions compared to the baseline solutions generated by the default optimization approach. They perform similarly or better when delay predictions change and solutions need to be modified multiple times during the day of operations.

Variable importance analysis showed that most high-precision F/O-based classifiers use information from flights that are not part of the F/O. This is an interesting

and practical discovery that, together with the interpretable structure of the resulting tree classifiers, provides actionable insights about which F/Os to consider or rule out under certain operational conditions.

Chapter 3

Integrated Aircraft, Crew, and Passenger Recovery

3.1 Introduction

Operations research methods have greatly benefited airline scheduling problems, such as fleet assignment, aircraft maintenance routing, and crew scheduling. These problems are handled during the planning phase, usually weeks or even months in advance. Therefore, they cannot consider disruptions during the day of operations, such as those due to weather conditions or mechanical failures.

Disruptions can significantly affect operational performance, leading to undesirable economic and passenger-related consequences. Passenger delays have become increasingly problematic, especially since the growth of air transportation demand has outpaced the capacity of major airports (Barnhart et al., 2012). The total cost of flight delays in 2007 was estimated to be $33 billion in the United States alone (Ball et al., 2010). More recently, the annual cost of flight delays worldwide was predicted to be around $60 billion (Wang & Vaze, 2016).

The process through which airlines intervene to mitigate the effects of disruptions is called *disruption management* or *airline recovery*. The main objective of this process is to minimize recovery costs, including those due to changes in fuel and crew requirements, delays, passenger re-accommodation, and loss of passenger good-

will. Recovery decision alternatives include delaying and canceling flights, rerouting or swapping aircraft, using backup aircraft, rescheduling planned crews or calling in reserve crews, and re-accommodating disrupted passengers.

The airline recovery process consists of flight schedule, aircraft, crew, and passenger recovery steps. These steps are similar to their scheduling counterparts, but in practice, their solution processes are further complicated due to limited time availability. Consequently, airline recovery is usually tackled sequentially with limited feedback from later stages. However, this lack of interaction between stages can lead to low-quality or even infeasible solutions. Integrating multiple recovery stages presents complex challenges, which some studies address using decomposition techniques (Petersen et al., 2012). However, solution time requirements remain a major bottleneck for such efforts.

Solving a fully integrated recovery model considering all aspects of the problem and including all solution alternatives could, in theory, create the best possible solution, but this is impossible for most real-world cases. The combinatorial nature of decisions involving schedule, aircraft, crew, and passenger recovery makes it impractical to rely solely on optimization-based approaches. Therefore, airline disruption management processes generally do not involve a full-scale optimization-based solution approach that considers all solution alternatives. A common approach is to reduce the size of the recovery problem by considering only a limited number of flights, aircraft, and crews (Clausen et al., 2010) before utilizing optimization methods. However, these problem-size reduction approaches often rely on heuristics that produce far-from-optimal solutions.

Timing plays a crucial role during the day of operations. Since decisions are made in real time at the operations control centers, solutions to recovery problems must be achieved within a limited timeframe (for example, 10 minutes) after a disruption in most cases (Hassan et al., 2021). This leads to the use of fast heuristic-based solution approaches in practice, even though they do not guarantee the generation of high-quality solutions.

While many researchers have presented integrated recovery models, the practi-

cal implementation of these ideas has been limited. The airlines would benefit from a comprehensive approach that can generate high-quality solutions in limited timeframes while accounting for the most critical aspects of all recovery steps. This study presents such an approach and develops fast methods for the integrated airline recovery problem, incorporating the recovery steps for schedule, aircraft, crew, and passengers.

Although the actual details of the disruptions change from one operational day to another, several key disruption patterns occur repeatedly. In this sense, the disruptions on a given day can often resemble the disruptions that occurred on some of the past days. Therefore, instead of solving similar problems each time from scratch, one can discover patterns in suitable recovery actions to historical disruptions with the help of machine learning (ML) methods and utilize them as a means of accelerating the solution process without significantly sacrificing solution quality. In Chapter 2, we showed that this approach performs better than other practical approaches in the context of crew recovery. In this chapter, we present a framework for the integrated recovery problem and develop practical solution methods.

The remainder of this chapter is organized as follows. In this section, we review the prior literature and present the contributions of this study. Section 3.2 presents an original airline recovery optimization model, discusses our representation of disruptions, and provides the mathematical formulations. In Section 3.3, we present the solution methodology, including how we leverage ML techniques. Section 3.4 is dedicated to the computational study, focusing on the experimental setup and providing results. Section 3.5 contains a solution analysis and discusses the insights gathered from the experiments. Finally, Section 3.6 concludes with a summary and future research directions.

3.1.1 Literature Review

Many existing studies on airline recovery have focused on individual steps in the process, including aircraft recovery (Teodorović & Guberinić, 1984, Teodorović & Stojković, 1990, Jarrah et al., 1993), crew recovery (Song et al., 1998, Wei et al.,

1997, Stojković et al., 1998) or passenger recovery (Thengvall et al., 2000, Bratu & Barnhart, 2006), due to the complexity of real-world problems and computational limitations. See Clausen et al. (2010) and Hassan et al. (2021) for more comprehensive reviews of studies that focus on individual recovery steps.

Handling the recovery steps individually leads to a sequential approach, which often results in high recovery costs due to the solutions that get fixed at earlier stages. With improvements in both the combinatorial optimization solution algorithms and the computational power available to researchers, there has been a recent increase in efforts to integrate two or more steps. These studies can be classified into three groups with respect to the recovery steps that are integrated. The first group integrates the aircraft and crew recovery steps, while the second integrates the aircraft and passenger steps. The studies in the last group consider all three steps. Aircraft recovery is the common step included in all groups. It should be noted that decisions related to schedule recovery, such as delaying and canceling flights, are usually considered together with the aircraft recovery step.

Aircraft and crew are two of the most critical and expensive airline resources. Although aircraft and crew recovery problems share some similarities, the latter is further complicated due to the crew duty legality rules that need to be satisfied. A *crew duty* is a set of consecutive flights that can be operated by a crew. An aircraft recovery solution that does not consider the implications for the crew recovery step can significantly increase crew recovery costs and hence the overall recovery cost. Studies that integrate aircraft and crew recovery steps aim to avoid such an outcome. One such study is by Maher (2016) that develops a solution approach following a column-and-row generation framework. Zhang et al. (2015) present a heuristic method in which an aircraft recovery model with crew considerations and a crew recovery model with aircraft considerations are run iteratively as long as the solution continues to improve.

The primary indicators of success for airline recovery operations are the amount of flight delays and the number of cancellations. On-time performance (OTP) is a widely used metric in air travel and other transportation services, reflecting how well

a service provider operates with respect to its published schedules. In the context of air travel, a flight is considered *on-time* if it arrives at its destination no more than 15 minutes after its scheduled arrival time. The U.S. Department of Transportation has been publishing airline on-time performance data since 1987 (BTS, 2020).

It is crucial to manage both delays and cancellations to minimize the effects of disruptions on passengers. Extended delays and many cancellations can result in loss of passenger goodwill and damage the brand image of the airline. However, not all flight delays and cancellations have similarly severe impacts on passengers. Several factors determine the impact of flight delays and cancellations on passengers, including the number of connecting passengers, connection times, load factors, and flight frequencies (Barnhart et al., 2014).

In order to deal with passenger delay costs in a holistic way, some studies integrate aircraft and passenger recovery steps by considering passenger itineraries when making schedule and aircraft recovery decisions. A passenger itinerary corresponds to the travel plan of a passenger consisting of one or more flights. Bisaillon et al. (2011) present a large neighborhood search based heuristic to obtain fast solutions for integrated aircraft and passenger recovery problems, using data sets from the ROADEF 2009 challenge organized by the French Operational Research and Decision Support Society. For detailed information about the challenge, see Palpant et al. (2009). The approach used by Bisaillon et al. (2011) is a multi-phase one in which an initial solution is generated, repaired, and improved through a large neighborhood search. Marla et al. (2017) incorporate flight planning related recovery actions, like increasing the flight cruise speeds, into schedule and aircraft recovery. Their modeling approach employs flight copies for departure-time decisions and cruise-speed alternatives.

Both sets of integration approaches discussed above leave either crew or passenger recovery out of consideration. Few studies have tried to integrate all three steps. Lettovsky (1997) was an early study to present such a solution approach with complete integration of schedule, aircraft, crew, and passenger recovery using a Benders' decomposition approach. This framework was improved by Petersen et al. (2012). They evaluate their solution approach in response to major disruptions, achieving

run times of less than 30 minutes for selected scenarios. Maher (2015) extends the column-and-row generation framework developed in Maher (2016) by incorporating passenger recovery decisions. These methods have been applied to networks with fewer than 800 daily flights and restricted to a limited number of flight modifications to keep the solution space manageable. The 30-minute solution time limit is longer than the preferred solution times in practice. Additionally, due to the combinatorial nature of recovery problems, these methods are unlikely to scale well for larger instances. A more tractable modeling approach is needed for the practical implementation of the developed methods.

Due to the solution time limitations on the day of operations, academic researchers and industry professionals commonly follow two main methods to reduce the solution space and accelerate the recovery optimization process (Clausen et al., 2010). In the first method, called the *time window technique*, only a portion of the flight schedule is used in the recovery problem, spanning from the time of the disruption to a specific number of hours into the future. Depending on the problem context and network size, the *time window length* can range from a couple of hours to the entire day. In the second method, the number of aircraft and crew schedules that can be modified during the recovery process is limited in advance (Petersen et al., 2012). We call this method the *candidate resource limitation technique*, because the candidate aircraft and crew to consider when tackling the disruption are limited. Although these methods help accelerate the solution process, they are not designed to specifically consider the characteristics of the disruption. Consequently, they may lead to solutions that are not suitable or feasible for some disruptions.

3.1.2 Contributions

In this chapter, we present a multi-stage modeling and solution framework for the integrated recovery problem that involves an aggregate and tractable integrated formulation, followed by post-processing steps for repairing and refining the aircraft, crew, and passenger recovery solutions. For each step, we develop original optimization models as well as a fast and tunable solution method by combining optimization

and ML. We accelerate the solution process by determining a restricted solution space tailored for each disruption. The trained classifiers provide insights into recovery decisions.

We now list the key contributions.

1) Model: We propose a tractable integrated recovery model that captures key decisions about the schedule, aircraft, crew, and passenger recovery steps. It is based on the modeling approach with facet-type constraints introduced in the context of air traffic flow management by Bertsimas and Patterson (1998). We show that it can generate solutions that are up to 50% better than those generated by a fully sequential approach.

2) Process Acceleration: The developed methodology accelerates both the *offline* and *online* solution phases due to the way the solution space is reduced based on classifier predictions and the definition of the feature set. The *offline* phase refers to a set of steps that must be completed in advance to make the solution methods ready for the *online* phase, while the *online* phase corresponds to generating solutions on the day of operations. The reduction in solution space is achieved by reducing the number of copies for individual flights by predicting the maximum allowable delay limits, instead of creating the same number of copies for all flights in all disruption instances. The feature set used in the prediction to describe the schedules and disruptions has four types of information: disruption, flight, crew, and passenger. This feature set allows classifier training to be performed for a group of flights instead of individual flights, which reduces the total model training time.

3) Crew vs Passenger Recovery Trade-off: Our solution method has an inherent exploration-exploitation trade-off, which allows insights into the relative importance of crew and passenger considerations in an integrated recovery approach. The integrated recovery model includes the major decisions related to crew and passenger recovery, namely infeasible crew duties and missed passenger connections, rather than including all crew and passenger-related decisions. The approximate costs of crew and passenger recovery are included in the objective function and the corresponding cost coefficients are calibrated to achieve the minimum recovery cost after completing the

post-processing steps. The calibrated cost coefficient values show that it is more challenging to repair infeasible crew duties, possibly due to crew duty feasibility rules; hence, it guides the solution to the appropriate balance between passenger and crew infeasibilities in the solution.

4) Interpretability: The variable importance analysis conducted on the maximum allowable flight delay limit classifiers yielded some insightful results. Disruption information, which consists of airport capacities and planned number of operations, was the most important information in determining the maximum allowed flight delay. Passenger-related information was found to be more important than crew-duty-related information for training the ML classifiers. Our findings imply that while passenger itineraries are easier to repair than crew schedules, passenger information is still more important in deciding the maximum allowable flight delay limits. This may be due to the fact that the average number of passengers on each itinerary is much lower than the average number of passengers on each flight and the delay on a single flight may affect a significant number of passenger itineraries. In contrast, each flight is operated by a single crew member, which limits the extent to which the rest of the network gets disrupted through crew-based propagation of delays.

3.2 Problem Statement and Mathematical Model

3.2.1 Disruption Definition

From an airline perspective, the main external factors that disrupt flight operations are airport closures and airport and airspace capacity reductions due to severe weather conditions. If a flight is not allowed to depart from or arrive at an airport within a time period, a delay or cancellation may be inevitable. However, reduced capacities alone do not necessarily result in disruptions. The extent of disruptions is correlated with congestion levels rather than actual capacities. If there are only five flights departing from an airport during a particular time period where the capacity is reduced to 10 departures, no flight would be affected. On the other hand, if the number of planned

departures exceeds 10 by a significant margin, many flights will inevitably experience delays or cancellations.

Many prior studies have considered airport capacity limitations as the main types of disruptions that need to be addressed in the airline recovery context. Petersen et al. (2012) and Maher (2016) present integrated recovery formulations with airport capacity constraints. The disruption scenarios they use reflect capacity reductions or closures at a single airport for a limited number of hours.

In this study, we also focus on reductions in airport capacity levels as the main source of disruptions. Our way of defining the scope of disruptions addressed in this study has two major advantages. First, the disruption instances correspond to disruptions during all hours of the day at the top 15 airports in the airline's network, covering ∼90% of the flights, instead of only focusing on a single airport and a limited time window. This definition provides a more comprehensive representation of the disruption state of the entire network. Second, in addition to the airport capacities, the disruption definition also includes the number of scheduled flight departures and arrivals by *all* airlines for a given airport and time period to calculate the congestion levels. As argued above, if there is no congestion, then reduced airport capacity may not lead to delays or cancellations. Information related to hourly airport capacities and the scheduled number of departures and arrivals is retrieved from the *Aviation System Performance Metrics (ASPM)* database published by FAA (2021).

We assume that the reduction in airport capacity affects all airlines operating at the airport in a proportional way. For example, suppose that the departure capacity during a specific time period is reduced by 50% in a case where the scheduled number of flight departures was already equal to the capacity. In that case, all airlines must delay or cancel half of their flights originally scheduled to depart within the corresponding time period.

3.2.2 Airline Recovery Problem

In practice, airline recovery usually follows a sequential process with limited feedback mechanisms, as illustrated in Figure 3-1. The first two steps, schedule and aircraft

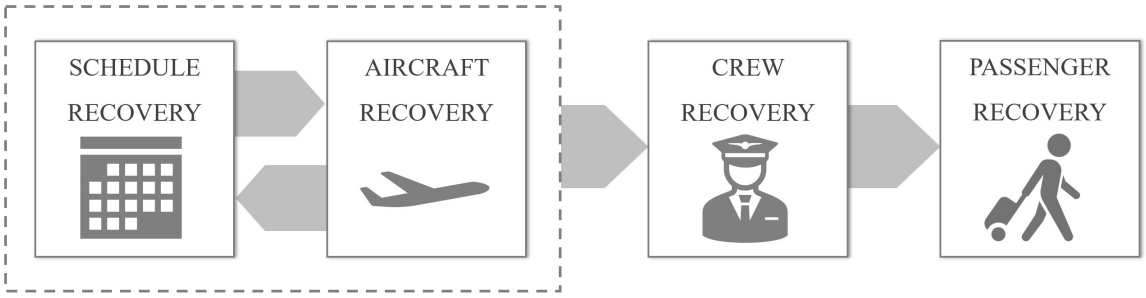


Figure 3-1: Sequential airline recovery process

recovery, are usually handled in a more integrated manner than the rest of the process. *Schedule recovery* aims to repair disrupted flight schedules mainly by delaying or canceling flights, while *aircraft recovery* involves determining suitable routes for individual aircraft to accommodate the revised flight schedules while satisfying aircraft availability and maintenance requirements. The next step is *crew recovery*, which ensures that each flight that is to be flown has a crew assigned to it by rescheduling planned crews or using reserve crews as needed, while meeting crew legality requirements. The last step is *passenger recovery*, which reassigns disrupted passengers to alternative itineraries to enable them to reach their destinations.

Due to the lack of full integration, the resulting sequential process requires extensive collaboration among operation controllers, crew planners, and passenger service coordinators to assess the feasibility of the recovery solutions and anticipate the severity of the potential consequences.

Aircraft Recovery Problem

The aircraft recovery problem aims to reroute the aircraft or use backup aircraft to minimize the effects of disruptions. The set of flights that are assigned to a single aircraft is called a *rotation*. Available rerouting options include ferrying, diverting, and swapping aircraft. Ferrying refers to flying the aircraft without any passengers, and swapping refers to reassigning flights between two aircraft. Any modification to planned rotations must satisfy maintenance requirements, airport curfew restrictions, and aircraft balance requirements. All aircraft must be positioned at their scheduled

locations at the end of the recovery period for operations to continue as planned. The evaluation of alternative recovery decisions should consider operational restrictions and preferences.

Government regulatory agencies, such as the FAA in the United States, require the operated aircraft to undergo regular maintenance checks to ensure the safety of air travel (Barnhart & Vaze, 2015a). There are different types of maintenance checks depending on their required frequencies, ranging from 3-5 days to 10 years. Airlines schedule the near-term maintenance checks in advance to ensure continuous operation. However, these schedules may also be disrupted during the day of operations. Hence, they need to be tracked and rescheduled when needed.

The most important type of check is called the *line check*, which must be completed every 24 to 60 hours of accumulated flight time, depending on the type of aircraft (NAA, 2022). The actual frequency of line checks depends on the characteristics of the airline network and aircraft utilization policies. For some airlines, each aircraft on average undergoes a line check every 4 days, while for others, every 7 days (Heinold, 2008). The *line check* is the most relevant maintenance check in the recovery context, since the recovery period is usually limited to 1 or 2 days. Maintenance checks that are required every few months or years do not need to be addressed within the recovery period.

During a line check, the aircraft is refueled, and all critical instruments are checked for defects. Although the duration of a line check can also differ from airline to airline, it usually takes 1 to 3 hours. Line checks are usually performed at the gates (NAA, 2022). Some airlines call them *overnight checks*, as they frequently schedule them after the last arrival of the aircraft on a given day. Although line checks are easy to perform and do not require much time, ensuring that all aircraft meet the line maintenance requirements can be challenging, especially when the flight schedule has many overnight red-eye flights.

Crew Recovery Problem

The schedule and aircraft recovery decisions made in the previous step often result in crew disruptions due to flight cancellations, delays, diversions, and aircraft swaps. These issues are handled in the crew recovery step, which involves generating new schedules for disrupted crews, reassigning crews to alternative schedules, utilizing reserve crews, and deadheading crew members while adhering to union agreements and civil aviation regulations, to cover all flights. The objective is to minimize incremental crew costs to operate the modified schedule while returning to the plan within a specified time window.

One of the most important civil regulation rules relates to the flight duty period (FDP) limits. FDP is defined as *"a period that begins when a pilot is required to report for duty with the intention of conducting a flight or series of flights and ends when the aircraft is parked after the last flight with no intention for further aircraft movement by the same pilot"* by the Federal Aviation Administration (FAA). The sequence of flights assigned to a crew within an FDP is called a crew duty and from the check-in time for the first flight of the duty until the arrival time of the last flight, crew is considered *on-duty*. The actual duration of the FDP is affected by flight delays, and hence it needs to be tracked for every crew duty during the recovery operations.

In addition to the maximum FDP time, other relevant rules include the minimum connection times between two consecutive flights in a crew duty and the minimum duty rest time between two duties assigned to the same crew. For a more detailed description of these rules, see Chapter 2 - Section 2.2.1. Our model ensures that FDP limits are not exceeded and that the recovery solutions satisfy the minimum connection and rest time rules.

Our model ensures that the FDP limits are not exceeded and that the legality rules on the minimum connection time between the flights and the minimum rest time between consecutive duties are satisfied.

Passenger Recovery Problem

The earlier decisions made in the sequential recovery process can cause significant disruptions for passengers. When flights are canceled or delayed, passengers may need to be rebooked on different itineraries. This step is called passenger recovery. The appropriate recovery actions depend on various factors, including the number of alternative itineraries available, the number of passengers affected, and the availability of seats on alternative flights. Minimizing flight delays may not always minimize passenger delays, indicating the importance of factoring in passenger disruption costs when making recovery decisions.

Decision support systems for airline recovery usually focus on addressing one of these steps at a time due to the complexity of the integrated problem. However, this sequential decision-making process often lacks comprehensive information on the resources affected in the subsequent steps and their potential impacts. Recovery decisions should consider the cost associated with passenger recovery when performing the aircraft and crew recovery steps, because decisions that may be considered *optimal* from the perspective of aircraft or crew recovery can be costly or even infeasible when all three factors are considered jointly.

3.2.3 Modeling Approach

In this section, we present an integrated recovery approach in which we first solve an integrated model that focuses on schedule recovery decisions while capturing the most important aspects of the aircraft, crew, and passenger recovery steps. The integrated model ensures that there exists a feasible flow of aircraft for the set of delay and cancellation decisions being made while also considering the approximate costs of crew misconnections, crew duty infeasibilities, and passenger disruptions due to cancellations and misconnections.

The proposed solution method involves post-processing steps in which the integrated solution is repaired and refined by handling detailed aircraft, crew, and passenger recovery steps using simpler integer optimization models. The aircraft

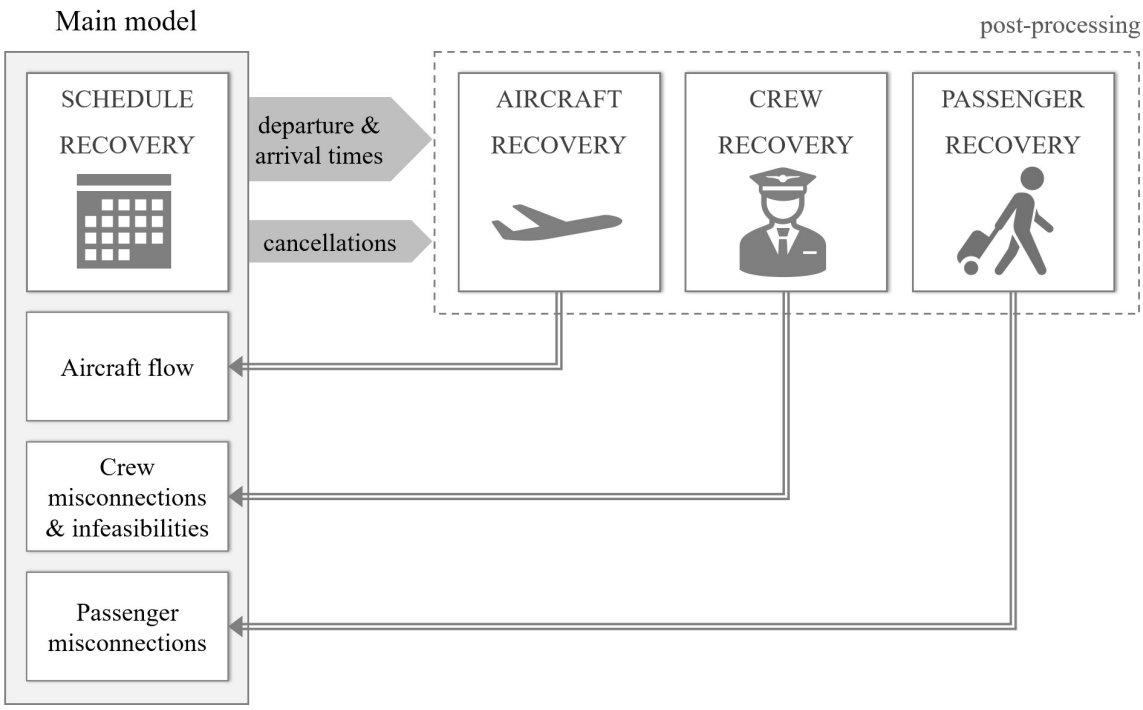


Figure 3-2: Integrated schedule, aircraft, crew, and passenger recovery

recovery post-processing step ensures that operated rotations are feasible while minimizing aircraft operations costs and the passenger costs resulting from limited seat capacity. The crew recovery post-processing step repairs disrupted crew schedules via crew rescheduling, deadheading, and reserve crew utilization. Passengers who cannot reach their final destination on their original itinerary or an alternative itinerary are considered stranded and need to be provided accommodation by the airline for the night. The passenger recovery post-processing step minimizes the number of stranded passengers by assigning them to alternative itineraries. Using this approach, depicted in Figure 3-2, we generate solutions with up to 50% lower costs than a fully sequential approach.

A key modeling idea in the integrated model is similar to that used by Bertsimas and Patterson (1998) for the air traffic flow management problem with enroute capacities. Their definitions of decision variables and the corresponding constraints allow their model to remain tractable even for large networks with several thousand flights. Inspired by their formulation, we have developed a novel integrated recovery model

that uses binary flight departure (arrival) variables for each time period, t, indicating whether the flight has already departed (arrived) by time t.

The logic behind the definitions of the departure and arrival decision variables is demonstrated in Figure 3-3. Each cell represents a time period for departure or arrival events. The flight is not allowed to depart or arrive during any of the time periods corresponding to white-shaded cells. The blue and red-shaded cells correspond to the first and last allowable departure (and arrival) times, respectively. All other time periods in between, shaded gray, correspond to the other departure and arrival time alternatives. Arrows represent flight copies that correspond to different departure-time (arrival-time) decisions for the same flight. The model determines how much each flight will be delayed by setting the values of the decision variables as those dictated by the departure and arrival time periods.

In our computational study, the duration of the time periods (that is, the time increment for flight copies) is set to 15 minutes, similar to that used by Petersen et al. (2012) and several other researchers. For our primary computational experiments, we assume that the flight times are fixed. This means that if the departure of a flight is delayed by a certain duration, then the arrival should also be delayed by the same duration. However, our model allows for changes in cruise speeds with a slight modification to the corresponding constraints. Details are discussed in the next section (Section 3.2.4).

The modeling approach described above leads to strong formulations, since some constraints are facets of the convex hull of the set of feasible integer solutions. Interested readers are referred to Bertsimas and Patterson (1998) for more details on this modeling idea and the strength of the resulting formulation.

3.2.4 Integrated Recovery Model

This section provides a mathematical formulation of the integrated recovery model, IRM, with the corresponding notation. Recovery decisions captured by the model include flight delays and cancellations, and which planned crew duty schedules and passenger itineraries to maintain and which to disrupt. The objective function is to

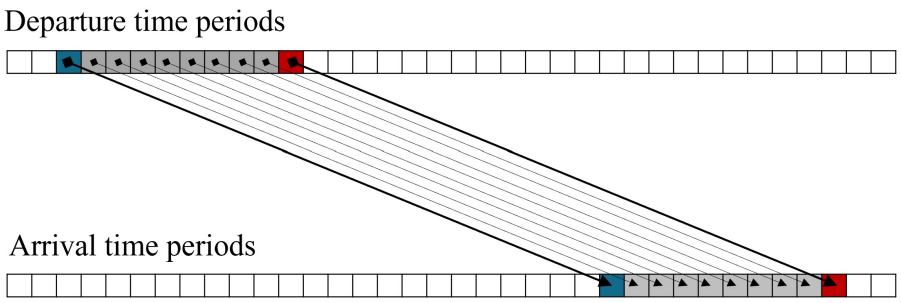


Figure 3-3: Flight departure and arrival time periods

minimize recovery costs, including crew and passenger delay costs, approximate costs of crew and passenger recovery, and cancellation savings reflecting fuel and other costs avoided by not operating the flight. The cost of cancellations is included in passenger recovery costs and crew infeasibility costs. Approximate crew and passenger recovery cost components are added to the objective function to guide the optimization toward crew and passenger recovery-friendly solutions. Constraints cover departure and arrival time modeling, flight time consistency, airport capacity limits, aircraft flow balance, passenger disruptions due to cancellations and misconnections, crew misconnections, crew duty limits, and aircraft maintenance requirements.

Several concepts need to be described before sharing the notation and formulation. First is a *flight connection pair* (follow-on) that corresponds to a pair of consecutively assigned flights in an aircraft rotation, crew duty, or passenger itinerary. Aircraft, crew, and passenger follow-ons are generated as separate sets, because for example, two flights assigned consecutively in an aircraft rotation may not be assigned consecutively in a crew duty or in a passenger itinerary.

Certain minimum connection time rules must be satisfied for two flights to be assigned consecutively. Minimum connections times for aircraft, crew and passenger follow-ons depend on a variety of factors including airline type, network structure, airport characteristics, scheduling practices, and so on. Note that, for the crew follow-ons, a higher minimum connection time is needed when the aircraft operating the flights in the crew follow-on are not the same. Additional time is needed to accommodate the crew moving from one aircraft to another. These rules are followed to

determine the feasibility of aircraft rotations, crew duties, and passenger itineraries.

A passenger itinerary is deemed *broken* if one or more flights in the itinerary are canceled and/or the available passenger connection time is reduced below the minimum needed. The IRM includes an estimate of the costs of these broken itineraries and an estimate of delay costs to passengers due to late arrival of the last flight in their itinerary. However, IRM does not take into account the costs associated with spilled passengers when fleet assignment changes reduce flight capacity.

Crew duties are sets of flights that can be sequentially operated by a crew. In crew scheduling problems, a large set of alternative crew duties is included in the solution space, and the optimization model selects a subset of crew duties to generate the planned crew schedules, usually for the entire month. The crew duties considered in the IRM correspond to the subset of planned crew duties that overlap with the current day of flight operations.

One of the objectives of the IRM is to minimize the number of crew duties that become infeasible due to the FDP limits. To ensure this, we calculated the latest legal end time (dt_s^l) for crew duties considering the duty start time, the legal FDP limits, and the next day's duty start time for the assigned crew.

In this study, it is assumed that delays occur when the aircraft is on the ground. Therefore, the effects of flight delays on the costs of crew duties are completely captured by considering the extension to the crew duty time, which in turn equals the arrival delay to the last flight in that crew duty. Additionally, the IRM ensures that the crew duty extensions do not violate the next day's crew duty. This maintains the feasibility of the crew pairing, where crew pairings are defined as one- or multi-day flight schedules for crews that start and end at the crew's home base.

Airlines often have different types of aircraft in their fleets. There are two main aircraft body types: narrow-body single aisle-aircraft, and wide-body two-aisle aircraft. The major aircraft manufacturers, Boeing and Airbus, produce different types of aircraft to meet the commercial requirements of the airlines. Having aircraft with different seat capacities provides airlines with the flexibility to schedule flights that reflect the demand for flights. The downside is that they need to train and certify

separate sets of pilots for each fleet due to the differences in the aircraft cockpits. This is especially important in the recovery context, as assigning a flight to another type of aircraft than the one originally planned would require finding a new crew to operate the flight. The IRM model ensures that there exists a feasible flow for each aircraft group and that maintenance requirements are met.

In IRM, we define an aircraft group $a \in A$ using distinct attributes, including the type of the fleet, as defined by crew and maintenance requirements. Each set of aircraft requiring the same fleet type and with the same maintenance requirement (specifically, line maintenance not required during the recovery period or line maintenance required during the recovery period) is considered an aircraft group. Constraints in the model ensure the feasiblity of the aircraft flow for each type and will ensure that aircraft requiring maintenance will arrive at their planned maintenance airport before the end of the recovery period.

The main decision variables in the model are $d_i^{a,t}$ and $r_i^{a,t}$, corresponding to flight departure and arrival decisions and aircraft group assignment decisions. The variable $d_i^{a,t}$ equals 1 if the flight $i \in I$ assigned to the aircraft group $a \in A$ has departed by the end of time period $t \in T$, 0 otherwise. The variable $r_i^{a,t}$ equals 1 if the flight $i \in I$ assigned to the aircraft group $a \in A$ has arrived by the end of time period $t \in T$, 0 otherwise.

The modeling logic behind the variables $d_i^{a,t}$ and $r_i^{a,t}$ was presented in Figure 3-3. Let us consider a simple example to demonstrate how the values of individual variables corresponding to different time periods are related to each other. Assume that the arrow in Figure 3-4 corresponds to the flight departure and arrival time decisions made by the model. Then, all the decision variables $d_i^{a,t}$ corresponding to time periods prior to the actual departure time, and all the decision variables $r_i^{a,t}$ corresponding to time periods prior to the actual arrival time are set to 0. Similarly, all the decision variables $d_i^{a,t}$ corresponding to time periods equal to and after the actual departure time, and all the decision variables $r_i^{a,t}$ corresponding to time periods equal to and after the actual arrival time are set to 1.

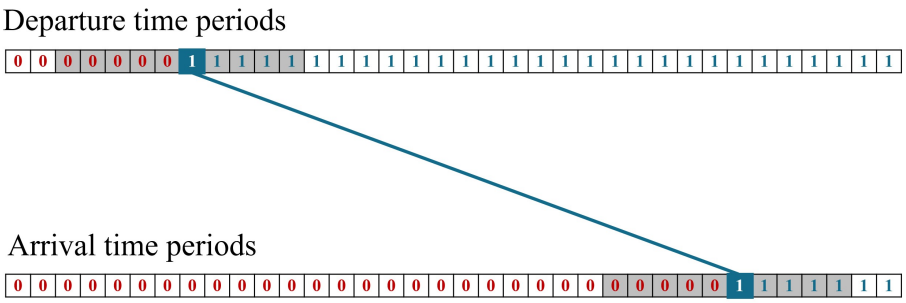


Figure 3-4: Flight departure and arrival time decision variables

Notation

Sets

I: the set of flights that depart and arrive during the recovery period T

A: the set of aircraft groups. Each fleet is divided into multiple groups with all aircraft in $a \in A$ having the same maintenance requirements and the same crew qualification requirements

P: the set of airports

$T = \{1, 2, 3, ...|T|\}$: the set of (15-minute) time periods in the recovery period

$I_p^D \subseteq I$: the subset of flights scheduled to depart from airport $p \in P$ during the recovery period

$I_p^R \subseteq I$: the subset of flights scheduled to arrive at airport $p \in P$ during the recovery period

F: the set of planned passenger itineraries on flights in I

S: the set of planned crew duty strings covering flights in I

$CX_s \subseteq (I \times I)$: the set of flight connection pairs in planned crew duty $s \in S$

$CX_f \subseteq (I \times I)$: the set of flight connection pairs in planned passenger itinerary $f \in F$

Data

DC_f: approximate passenger delay cost for itinerary $f \in F$, in dollars per (15-minute) time period

PC_f: approximate cost associated with broken passenger itinerary $f \in F$

CC_s: approximate cost associated with a crew duty $s \in S$ becoming infeasible

EC_s: crew duty extension (delay) cost for duty $s \in S$, in dollars per (15-minute) time period

ZS_i: savings associated with canceling flight $i \in I$

AC_i^a: cost of assigning flight $i \in I$ to aircraft of group $a \in A$

ft_i: actual block time of flight $i \in I$ measured in number of time periods

td_i^p: scheduled departure time period of flight $i \in I$ as per the planned flight schedule

td_i^l: latest allowed departure time period of flight $i \in I$

tr_i^p: scheduled arrival time period of flight $i \in I$ as per the planned flight schedule

tr_i^l: latest allowed arrival time period of flight $i \in I$

$NB_p^{a,t}$: planned number of aircraft of group $a \in A$ at airport $p \in P$ available for recovery operations beginning in time period $t \in T$. Additional aircraft arrive during the recovery period when an aircraft completes its planned flight or maintenance task that was ongoing at the beginning of the recovery horizon.

$NE_p^{a,t}$: planned number of aircraft of group $a \in A$ at airport $p \in P$, required to operate flights that depart during time period $t \in T$ (but do not arrive until after the recovery period). We let $NE_p^{a,|T|+1}$ equal the planned number of aircraft of group $a \in A$ at airport $p \in P$ at the end of the recovery period.

CR_p^t: arrival capacity at airport $p \in P$ during the time period $t \in T$

CD_p^t: departure capacity at airport $p \in P$ during the time period $t \in T$

$SR_{p,a}$: minimum turnaround time for aircraft group $a \in A$ at airport $p \in P$ (in time periods)

$SP_{i,j}$: minimum passenger connection time between flights $i, j \in I$ (in time peri-

ods)

$SC_{i,j}$: minimum crew connection time between flights $i, j \in I$ (in time periods)

NF_s: number of flights in the planned crew duty $s \in S$

LS_s: the last flight of the planned crew duty $s \in S$

LF_f: the last flight of the passenger itinerary $f \in F$

dt_s^l: the latest legal end time period for the planned crew duty $s \in S$

Decision Variables

$d_i^{a,t}$: 1 if flight $i \in I$ assigned to aircraft group $a \in A$ has departed by the end of time period $t \in T$, 0 otherwise. Let $d_i^{a,0} = 0, \forall a \in A, \forall i \in I$.

$r_i^{a,t}$: 1 if flight $i \in I$ assigned to aircraft group $a \in A$ has arrived by the end of time period $t \in T$, 0 otherwise. Let $r_i^{a,0} = 0, \forall a \in A, \forall i \in I$.

q_f: 1 if the passenger itinerary $f \in F$ becomes infeasible (i.e., broken), 0 otherwise.

u_s: 1 if the planned crew duty $s \in S$ becomes infeasible, 0 otherwise.

z_i: 1 if flight $i \in I$ is canceled; 0 otherwise.

Formulation

$$\begin{aligned}\min \sum_{f \in F} \Bigg(PC_f \cdot q_f + \sum_{a \in A} \sum_{tr_{LF_f}^p < t \le tr_{LF_f}^l} DC_f \cdot (t - tr_{LF_f}^p) \cdot (r_{LF_f}^{a,t} - r_{LF_f}^{a,t-1}) \Bigg) \\ + \sum_{s \in S} \Bigg(CC_s \cdot u_s + \sum_{a \in A} \sum_{tr_{LS_s}^p < t \le tr_{LS_s}^l} EC_s \cdot (t - tr_{LS_s}^p) \cdot (r_{LS_s}^{a,t} - r_{LS_s}^{a,t-1}) \Bigg) \\ + \sum_{i \in I} \sum_{a \in A} AC_i^a \cdot d_i^{a,td_i^l} - \sum_{i \in I} ZS_i \cdot z_i \end{aligned} \tag{3.1}$$

$$\text{s.t.} \quad d_i^{a,t} - d_i^{a,t-1} \geq 0 \qquad \forall i \in I, \forall a \in A, \forall t \in \{td_i^p, .., td_i^l\} \tag{3.2}$$

$$r_i^{a,t} - r_i^{a,t-1} \geq 0 \qquad \forall i \in I, \forall a \in A, \forall t \in \{tr_i^p, .., tr_i^l\} \tag{3.3}$$

$$r_i^{a,t+ft_i} - d_i^{a,t} = 0 \qquad \forall i \in I, \forall a \in A, \forall t \in \{td_i^p, .., td_i^l\} \tag{3.4}$$

$$z_i + \sum_{a \in A} d_i^{a,td_i^l} = 1 \qquad \forall i \in I \tag{3.5}$$

$$\sum_{a \in A} \left(\sum_{i \in I_p^D} (d_i^{a,t} - d_i^{a,t-1}) + NE_p^{a,t} \right) \leq CD_p^t \qquad \forall p \in P, \forall t \in T \tag{3.6}$$

$$\sum_{a \in A} \left(\sum_{i \in I_p^R} (r_i^{a,t} - r_i^{a,t-1}) + NB_p^{a,t} \right) \leq CR_p^t \qquad \forall p \in P, \forall t \in T \tag{3.7}$$

$$\sum_{1 \leq t \leq t'} (NB_p^{a,t} - NE_p^{a,t}) + \sum_{i \in I_p^R} \sum_{SR_{p,a}+1 \leq t < t'} (r_i^{a,t-SR_{p,a}} - r_i^{a,t-SR_{p,a}-1}) - \sum_{i \in I_p^D} \sum_{1 \leq t < t'} (d_i^{a,t} - d_i^{a,t-1}) \geq 0 \qquad \forall a \in A, \forall p \in P, \forall t^{'} \in T \cup \{|T|+1\} \tag{3.8}$$

$$SP_{i,j} - \sum_{a \in A} \sum_{t \leq td_j^l} (r_i^{a,t} - d_j^{a,t}) \leq |T| \cdot q_f \qquad \forall f \in F, \forall (i,j) \in CX_f \tag{3.9}$$

$$z_i + z_j \leq 2 \cdot q_f \qquad \forall f \in F, \forall (i,j) \in CX_f \tag{3.10}$$

$$SC_{i,j} - \sum_{a \in A} \sum_{t \leq td_j^l} (r_i^{a,t} - d_j^{a,t}) \leq |T| \cdot u_s \qquad \forall s \in S, \forall (i,j) \in CX_s \tag{3.11}$$

$$\sum_{i \in s} z_i \leq NF_s \cdot u_s \qquad \forall s \in S \tag{3.12}$$

$$\sum_{a \in A} \sum_{t \leq tr_{LS_s}^l} (t - dt_s^l) \cdot (r_{LS_s}^{a,t} - r_{LS_s}^{a,t-1}) \leq |T| \cdot u_s \qquad \forall s \in S \tag{3.13}$$

$$z_i, q_f, u_s, r_i^{a,t}, d_i^{a,t} \in \{0,1\} \qquad \forall i \in I, \forall f \in F, \forall s \in S, \forall a \in A, \forall t \in T \tag{3.14}$$

The objective function (3.1) has six cost components: approximate costs of passenger disruptions, PC_f, $f \in F$; passenger delay cost, DC_f, $f \in F$; approximate

costs of crew recovery, CC_s, $s \in S$; crew delay cost, EC_s, $s \in S$; cost of assigning a flight to an aircraft group, AC_i^a, $i \in I$, $a \in A$; and cancellation savings, ZS_i, $i \in I$.

The cost of crew delay, EC_s, corresponds to the cost of a 15-minute extension of crew duty $s \in S$ (i.e., the additional crew pay) due to propagated delay. Consistent with practice at many airlines, it is assumed that with any crew duty extension during the day of flight operations, the crew needs to be paid extra, as it is an additional duty on top of the originally planned and notified duty schedule. The cost of passenger delay, DC_f, calculates the cost per 15-minute delay for all passengers on the itinerary $f \in F$. For any itinerary, the passenger delay is calculated as the delay of the last flight on the itinerary. The cost of assigning a flight to an aircraft group, AC_i^a, represents the change in operating cost associated with assigning that flight to a different aircraft group than originally planned and is equal to zero if the assigned aircraft group is the same as the planned one. The cost coefficients, CC_s, correspond to the approximate cost of crew recovery for crew duty $s \in S$, if it becomes infeasible in the IRM solution. Similarly, the cost coefficients, PC_f, reflect the approximate cost of passenger recovery for the passenger itinerary $f \in F$ which becomes broken in the IRM solution. These values of costs of broken crew duties and broken passenger itineraries are approximate, rather than being exactly equal to the actual cost figures. That is because the IRM model considers the fact that some of these crew duty infeasibilities and the broken passenger itineraries will be fixed in the post-processing phase. The process to calculate the approximate values is discussed in Section 3.5.3. Cancellation savings, ZS_i, reflect the cost savings in fuel and maintenance due to canceling a flight. Recall that, as stated at the beginning of this section, IRM does not take into account the costs associated with spilled passengers when fleet assignment changes reduce flight capacity.

Constraints (3.2)-(3.3) model that when a flight departure (respectively, arrival) time variable is set to 1, all subsequent departure (respectively, arrival) time variables for that flight should be set to 1 (see Figure 3-4 for a concrete example). These are the facet-defining constraints mentioned in Section 3.2.3.

Constraints (3.4) ensure that flight departure and arrival times are consistent

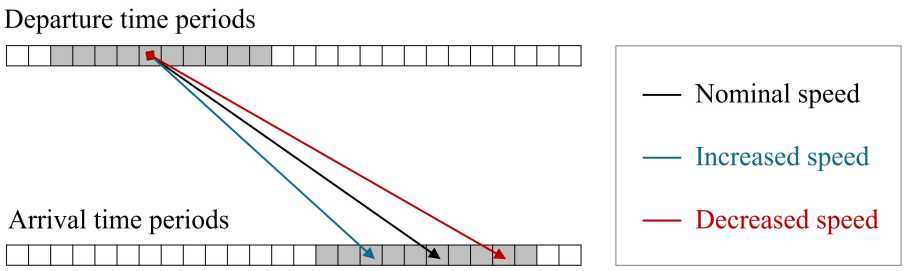


Figure 3-5: The flight time is a function of the cruise speed

with the flight time. We assumed the flight time to be fixed, but it is also possible to allow changes in cruise speed during the flight. Let ft_i^{min} and ft_i^{max} be the minimum and maximum possible values of the actual block times, as dictated by the allowable range of cruise speeds. If we replace Constraints (3.4) with Constraints (3.15), the model considers different arrival time periods for a flight departing in a given time period, as shown in Figure 3-5, and is allowed to modify the cruise speed. The blue and red arrows correspond to flight copies with increased and decreased cruise speeds, respectively, of the same flight. The cost of increasing or the savings from decreasing the cruise speed can be incorporated into the objective function by adding the objective cost component given by expression (3.16) where CSC_i^a is defined as the cost of increasing the cruise speed to reduce the actual block time of flight i by 15 minutes when assigned to aircraft group of $a \in A$.

$$d_i^{a,t} - \sum_{max(t+ft_i^{min},tr_i^p)\leq t' \leq min(t+ft_i^{max},tr_i^l)} (r_i^{a,t'} - r_i^{a,t'-1}) = 0 \quad \forall a \in A, \forall i \in I, \forall t \in \{td_i^p, .., td_i^l\} \tag{3.15}$$

$$\sum_{i \in I} \sum_{a \in A} CSC_i^a \cdot \left(ft_i - \sum_{td_i^p < t \leq tr_i^l} (r_i^{a,t} - d_i^{a,t}) \right) \tag{3.16}$$

Constraints (3.5) ensure that a flight is canceled if it cannot be assigned to any aircraft group. Using arrival variables, $r_i^{a,t}$, instead of departure variables, $d_i^{a,t}$, in

these constraints would have the same effect, because Constraints (3.4) ensure that a flight can have an arrival time if and only if it has a departure time.

Constraints (3.6) and (3.7) model departure and arrival airport capacities, respectively. They count departures and arrivals for each time period t and ensure that they do not exceed the airport capacities.

Constraints (3.8) are modeled to maintain the consistency of aircraft flow with departure and arrival decisions. They ensure that the number of departures in time period $t \in T$ of aircraft of group $a \in A$ at airport $p \in P$ is less than or equal to the number of aircraft of group $a \in A$ on the ground — at time $t \in T$ and at airport $p \in P$ — for at least the minimum aircraft turn time. These constraints also ensure that only aircraft available at the start of the recovery period or made available during the recovery period are used. Furthermore, these constraints ensure that aircraft are positioned at the end of the recovery period as planned. This ensures that planned operations can resume after the recovery period and that maintenance requirements are satisfied. Finally, Constraints (3.8) ensure the availability of the aircraft needed for flights that depart but do not end during the recovery period.

Figure 3-6 includes a simple example of how Constraints (3.8) work. Assume that the airline has four flights scheduled to depart in the time period t (blue-shaded aircraft) and that an aircraft has been available on the ground at this airport since the beginning of the recovery period (gray-shaded aircraft). If an aircraft arrives during the time period $t-3$ and another during the time period $t-2$, then there would be three aircraft available for four flights during the time period t. The minimum turnaround time is assumed to be 30 minutes, which corresponds to two time periods. In this case, at least one of the scheduled flights should be delayed or canceled.

Decision variables in the model select the passenger itineraries to be broken. This is achieved by Constraints (3.9)-(3.10) that force q_f for passenger itinerary $f \in F$ to take on value 1, indicating that itinerary f is infeasible (or broken), because one or more flights in f are canceled and/or passenger connection time is reduced below the minimum needed. The minimum passenger connection times between flights $i, j \in I$, $SP_{i,j}$, are defined in 15-minute time increments by rounding the actual values to the

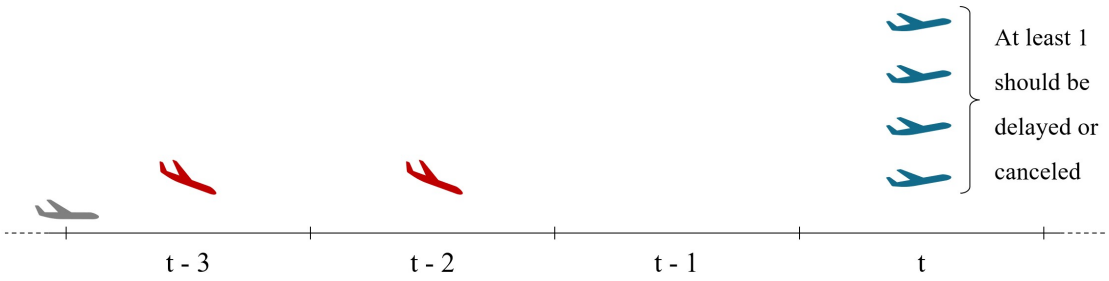


Figure 3-6: Aircraft flow modeling

nearest integer. For example, the minimum connection times of 25 and 35 minutes are considered as 2 time increments.

Similarly, the model includes decision variables that determine which duties are infeasible or broken. From Constraints (3.11)-(3.13), a planned crew duty $s \in S$ becomes infeasible if any of its flights are canceled, if the crew connection time is reduced below the minimum needed, or if the duty duration becomes infeasible due to the arrival time of the last flight on the duty exceeding the latest legal duty end time. Minimum crew connection times between flights $i, j \in I$, $SC_{i,j}$, are also defined in 15-minute time increments similar to $SP_{i,j}$ above.

Constraints (3.14) define the variable domains. All decision variables are binary. Therefore, the IRM model, (3.1)-(3.14), is a binary integer optimization model.

3.2.5 Post-processing Models

Aircraft Recovery Model

The IRM solution sets aircraft departure and arrival times, makes cancellation decisions, and assigns flights to aircraft groups to ensure satisfaction of maintenance requirements and conversation of aircraft flows (that is, feasible aircraft rotations) while minimizing approximate recovery costs. In the aircraft recovery post-processing step, we solve an aircraft recovery model, ARM(a), for each IRM aircraft group $a \in A$. We classify the aircraft in group $a \in A$ into types $e \in E^a$, with each type distinguished by the number of seats (i.e., the capacity) of the aircraft. The ARM decisions include

assignment to each flight of an aircraft with specific capacity, ensuring flight coverage and feasible aircraft rotations that minimize approximate passenger recovery costs resulting from passengers being spilled when insufficient aircraft capacity is assigned. In other words, the ARM refines the decisions of the IRM to also account for approximate spill costs, without compromising feasibility of flight coverage, aircraft rotations, or crew duties, and without creating any additional broken itineraries.

With departure times, arrival times, and cancellation decisions fixed, the aircraft recovery problem becomes a multi-commodity network flow problem. Multi-commodity network flow problems arise in various real-life settings where different types of item (that is, commodities) sharing the same underlying network are transported from the source nodes to the sink nodes (Ahuja et al., 1988). For example, traffic flow in an urban environment can be cast as a multi-commodity network flow problem, because vehicles with different origins and destinations use the same network for their travel. In this example, vehicles correspond to commodities, origins and destinations correspond to source and sink nodes, and roads correspond to arcs in the underlying network. In an aircraft recovery context, the different types of aircraft (varying in seating capacity and in maintenance due requirements) are the commodities, the airports at the beginning and end of the day are the source and sink nodes, and the flight legs and connections between them are the arcs.

The set of all nodes includes two types of nodes — flight nodes and source/sink nodes. There are two nodes for each flight that is not canceled in the IRM solution (a flight-start node and a flight-end node), and a source node and a sink node for each airport-aircraft group combination in the network. The set of all arcs includes four sets of arcs. Every potential flight connection is represented by an arc from the flight-end node of the first flight to the flight-start node of the second flight in the connection. There are *flight arcs* (denoted as $flight_{arcs}$), one from the flight-start node to the flight-end node for each flight. There are *day-start arcs* from each source node to the flight-start node of each flight that originates at that airport and can be flown by that aircraft group. Finally, there are *day-end arcs* from the end-flight node of each flight ending at an airport and that can be operated by a particular aircraft

group to the sink node of that airport for that aircraft group.

All flight-start and flight-end nodes have zero demand/supply values but an upper and lower bound of 1 to ensure coverage. Each source node has a supply equal to the available number of aircraft of the associated aircraft type (or commodity) at that node, and all sink nodes have a demand equal to the required number of aircraft of that type at that node. Arc costs for flight arcs correspond to the sum of the flight operating costs and the passenger spill costs due to insufficient seat capacity, and the arc costs are zero for all other types of arcs in the network. Note that two aircraft may be in different aircraft groups due to differences in their seat capacities, aircraft operating costs, or simply because they are available/needed at different times of the day.

Figure 3-7 depicts the nodes and arcs in an underlying network with a single airport, eight flights, and two aircraft types. Blue and red-shaded nodes correspond to the source and sink nodes where s_1 and s_2 are the number of available aircraft of different types at the beginning of the day, while d_1 and d_2 are the number of required aircraft of different types at the end of the day. The grey-shaded nodes are flight nodes. Each flight has two nodes, a flight-start node i and a flight-end node $i^{'}$, to ensure that it will be covered by setting the lower and upper bounds of the arc between the nodes to 1. At each node, the total inbound flow should be equal to the total outbound flow. Positive arc flow values in the solution correspond to aircraft assigned to flights, assigned to connections between flights or to aircraft waiting on the ground. (The dashed line arcs correspond to ground arcs reflecting the case of some of the available aircraft waiting idle at the airport until the end of the day). Paths from the source nodes to the sink nodes correspond to aircraft rotations.

Notation for ARM(a)

Sets

E^a: the set of all aircraft types

$Nodes$: the set of all nodes, representing airport locations in time or supply and demand nodes or flight-start and flight-end nodes for commodity $e \in E^a$.

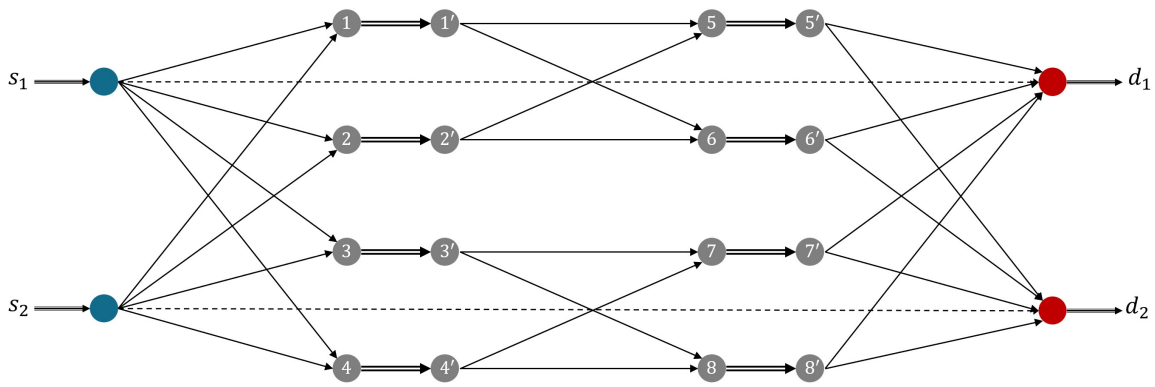


Figure 3-7: Network representation of the multi-commodity network flow formulation for the aircraft recovery model with two commodities

Arcs: the set of all arcs.

$Arcs_{flight}$: the set of arcs, one for each flight assigned to any aircraft group $a \in A$ in the IRM solution.

$in(n)$: the set of all inbound arcs (m,n) to node $n \in Nodes$ from any node $m \in Nodes$

$out(n)$: the set of all outbound arcs (n,m) from node $n \in Nodes$ to any node $m \in Nodes$

Data

$b_{n(e)}$: the number of aircraft of type $e \in E^a$ available ($b_{n(e)} > 0$) or required ($b_{n(e)} < 0$) at node $n(e) \in Nodes$. Note, $b_{n(e)} = 0$ for all nodes other than source and sink nodes.

$c^e_{m,n}$: approximate spill cost associated with assigning aircraft of type $e \in E$ to arc $(m, n) \in Arcs$.

Decision Variables

$f^e_{m,n}$: Flow of aircraft of type $e \in E$ on arc $(m, n) \in Arcs$.

ARM(a) Formulation

$$\min \sum_{e \in E^a} \sum_{(m,n) \in Arcs_{flight}} c^e_{m,n} \cdot f^e_{m,n} \tag{3.17}$$

$$\text{s.t.} \quad \sum_{(n,m) \in out(n)} f^e_{n,m} - \sum_{(m,n) \in in(n)} f^e_{m,n} = b_{n(e)} \qquad \forall e \in E^a, \forall n \in Nodes \tag{3.18}$$

$$\sum_{e \in E^a} f^e_{m,n} = 1 \qquad \forall (m,n) \in Arcs_{flight} \tag{3.19}$$

$$f^e_{m,n} \in \mathbb{Z}^+ \qquad \forall e \in E^a, \forall (m,n) \in Arcs \tag{3.20}$$

The objective function (3.17) is to minimize the approximate cost of spilled passengers. The cost coefficients, $c^e_{m,n}$, consider the number of passengers that are spilled if the flight corresponding to the arc (m,n) is assigned to an aircraft of type $e \in E$ using the scheduled number of passengers on the flight and the seat capacity of each aircraft.

Constraints (3.18) ensure conversation of flow of each aircraft type at each node. Constraints (3.19) require that active flights be covered by exactly one aircraft. Constraints (3.20) define variable domains.

Crew Recovery Model

The crew recovery model (CRM) used in post-processing is based on the model (2.1)-(2.14) presented in Chapter 2. The major difference is that it holds the flight schedule fixed and does not allow for delay propagation or flight cancellations. Flights that existing crews cannot cover are assigned to high-cost reserve crews. Therefore, constraints (2.5)-(2.12) are not included. The resulting model CRM, which is presented below, minimizes the cost of crew recovery, including those due to crew delays and high-cost reserve crew use.

In addition to the crew duties generated to cover the flights in the network, the model also uses a set of dummy crew duties, S^D, that correspond to crew staying where they are if their next day's duty starts at the same airport or transferring the

crew to the airport where their next day's duty starts, by some other means than the scheduled flights in the network, such as deadheading on another airline's scheduled flight and using a mode of transportation other than flights. Deadheading corresponds to a crew flying as passengers.

Notation for CRM

Sets

I: the set of flights

K: the set of crews, including normal reserves

S: the set of crew duties, also called crew strings

S^D: the set of dummy crew duties

S_k: the set of crew strings $s \in S$ that can be assigned to crew $k \in K$

K_s: the set of crews $k \in K$ that can be assigned to crew string $s \in S$

Data

CC_s^k: cost of assigning string $s \in S$ to crew $k \in K$

ZC_i: cost of assigning flight $i \in I$ to a high-cost reserve crew

a_{is}: 1 if crew string $s \in S$ contains flight $i \in I$; 0 otherwise

Decision Variables

y_s^k: 1 if crew string $s \in S_k$ is assigned to crew $k \in K$; 0 otherwise

z_i: 1 if flight $i \in I$ is assigned to a high-cost reserve crew; 0 otherwise

CRM Formulation

$$\min \quad \sum_{k \in K} \sum_{s \in S_k} CC_s^k y_s^k + \sum_{i \in I} ZC_i z_i \tag{3.21}$$

$$\text{s.t.} \quad z_i + \sum_{k \in K} \sum_{s \in S_k} a_{is} y_s^k \geq 1 \qquad \forall i \in I \tag{3.22}$$

$$\sum_{s \in S_k} y_s^k = 1 \qquad \forall k \in K \tag{3.23}$$

$$\sum_{k \in K_s} y_s^k \leq 1 \qquad \forall s \in S \setminus S^D \tag{3.24}$$

$$y_s^k, z_i \in \{0, 1\} \qquad \forall s \in S_k, \forall k \in K, \forall i \in I \tag{3.25}$$

The objective function (3.21) minimizes the total recovery costs, including those

due to modifications to planned crew duties, CC_s^k, and high-cost reserve crew use.

CC_s^k corresponds to the additional crew pay incurred when duty $s \in S$ is assigned to crew $k \in K$ compared to that crew's planned duty. ZC_i is the total crew pay for assigning flight $i \in I$ to high-cost reserve crew.

Constraints (3.22) ensure that each flight is assigned to at least one crew. Crew strings correspond to duties that can be assigned to crews, including normal reserve crews. If more than one crew is assigned to a flight, then all additional crews fly as passengers, i.e., they are *deadheaded*. Constraints (3.23) model that each crew must be assigned to exactly one actual or dummy duty. Constraints (3.24) ensure that a duty, except for dummy duties, is assigned to at most one crew. Assignment of multiple crews to the same duty is not allowed for the non-dummy duties. Constraints (3.25) define the domains of the decision variables.

Passenger Recovery Model

The passenger recovery model (PRM) corresponds to the last post-processing step. It minimizes the number of stranded passengers and changes in passenger delay costs by re-assigning spilled passengers and those on broken itineraries to other itineraries when possible. It takes as input the solutions to the IRM and ARM models regarding broken passenger itineraries, canceled flights, flight departure and arrival times, and numbers of spilled passengers due to insufficient seat capacity. Recovery actions include assigning passengers to alternative itineraries and identifying stranded passengers at the end of the recovery period.

Notation for PRM

Sets

I_{active}: the set of flights $i \in I$ that are not canceled in the IRM solution.

$F_{infeasible} = (F_{infeasible}^{broken}, F_{infeasible}^{spill})$: the set of broken passenger itineraries, $F_{infeasible}^{broken}$, and the set of spilled itineraries, $F_{infeasible}^{spill}$, containing flights with spill from the ARM solutions. Note that passengers on broken itineraries are excluded when calculating

the amount of spill (the number of booked passengers minus aircraft seating capacity), for each flight.

F_f: the set of passenger itineraries that the passengers on itinerary $f \in F_{infeasible}$ can be directed to.

F^i: the set of passenger itineraries that include flight $i \in I$

Data

AS^i: number of available seats on flight $i \in I$. This equals the assigned capacity of flight i from ARM solution minus the number of passengers on all non-broken itineraries containing i. For flights i with assigned capacity from ARM insufficient to accommodate all booked passengers on i, excluding passengers on broken itineraries, the number of available seats AS^i is negative, with a magnitude that is equal to the spill from i.

NP_f: number of scheduled passengers on passenger itinerary $f \in F$

NI_f: number of flights on passenger itinerary $f \in F$

$PDC_{f,f'}$: per passenger change in delay cost when reassigned from planned itinerary $f \in F_{infeasible}$ to itinerary $f' \in F_f$

SPC_f: cost of a passenger from itinerary $f \in F_{infeasible}$ being stranded at the end of the recovery period.

Q: penalty multiplier applied to costs $PDC_{f,f'}$ and SPC_f to ensure that passenger are not spilled from their planned itineraries by a reassigned passenger in a broken itinerary. This ensures that booked passengers always have priority on their planned flights over other reaccommodated passengers.

We let $Q \geq \max_{f \in F_{infeasible}} \left(NI_f * \max_{f' \in F_f} (\frac{SPC_f}{PDC_{f,f'}}) \right)$

Decision Variables

$\gamma_{f,f'}$: number of disrupted passengers reassigned from their planned itinerary $f \in F_{infeasible}$ to itinerary $f' \in F_f$.

δ_f: number of passengers from itinerary $f \in F_{infeasible}$ stranded at the end of the recovery period.

PRM Formulation

$$\min \sum_{f \in F^{broken}_{infeasible}} \left(SPC_f \cdot \delta_f + \sum_{f' \in F_f} PDC_{f,f'} \cdot \gamma_{f,f'} \right) + \sum_{f \in F^{spill}_{infeasible}} \left(Q \cdot SPC_f \cdot \delta_f + \sum_{f' \in F_f} Q \cdot PDC_{f,f'} \cdot \gamma_{f,f'} \right) \quad (3.26)$$

$$\text{s.t.} \quad \delta_f + \sum_{f' \in F_f} \gamma_{f,f'} = NP_f \qquad \forall f \in F^{broken}_{infeasible} \quad (3.27)$$

$$\delta_f + \sum_{f' \in F_f} \gamma_{f,f'} \leq NP_f \qquad \forall f \in F^{spill}_{infeasible} \quad (3.28)$$

$$\sum_{f \in F_{infeasible}} \sum_{f' \in (F^i \cap F_f)} \gamma_{f,f'} - \sum_{f \in (F^{spill}_{infeasible} \cap F^i)} \sum_{f' \in (F^i \cap F_f)} \gamma_{f,f'} - \sum_{f \in (F^{spill}_{infeasible} \cap F^i)} \delta_f \leq AS^i \quad \forall i \in I_{active} \quad (3.29)$$

$$\delta_f, \gamma_{f,f'} \geq 0 \qquad \forall f \in F_{infeasible}, f' \in F_f \quad (3.30)$$

The objective function (3.26) is to minimize the costs of passengers on infeasible itineraries from being stranded at the end of the recovery period and/or by delays caused by being spilled to or reaccommodated on other itineraries. It should be noted that the objective function does not correspond to the true cost due to the penalty multiplier Q. When calculating the actual costs in the computational study, we divide the costs of the spilled passengers by Q. Only passengers on infeasible itineraries can be reassigned to other itineraries and booked passengers cannot be displaced by passengers booked on other itineraries. Constraints (3.27) ensure that passengers on a broken itinerary are directed to another itinerary or are considered stranded at the end of the recovery period. Constraints (3.28) ensure that the number of spilled passengers on an itinerary do not exceed the number booked. Constraints (3.29) ensure that the net change in the number of passengers on flight $i \in I$ cannot

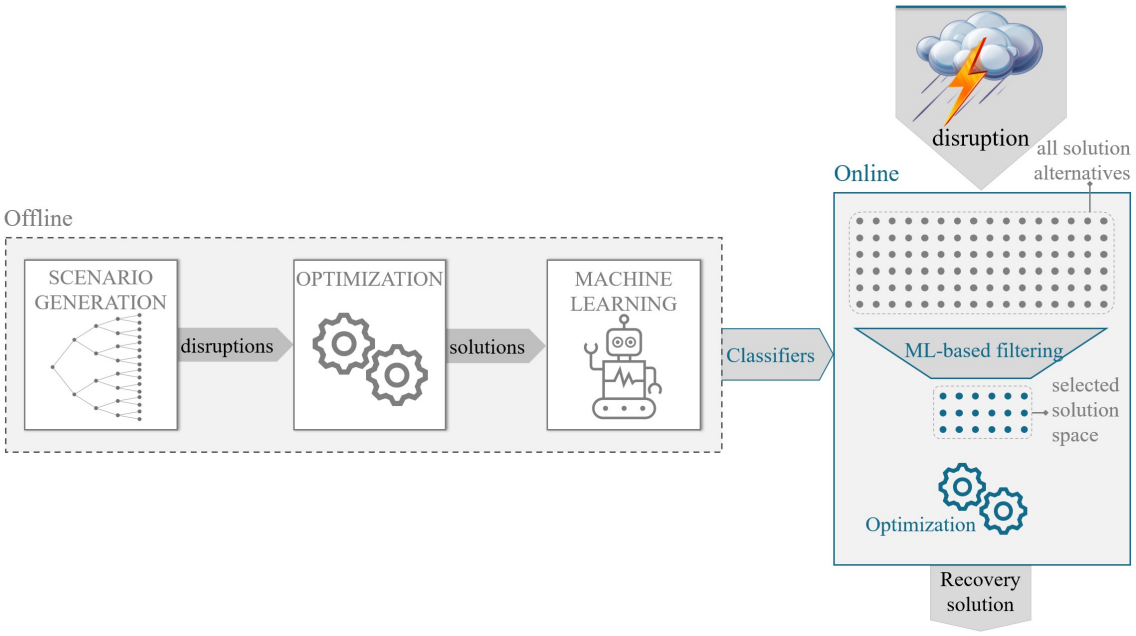


Figure 3-8: General Framework

exceed the remaining available seats on that flight. (Note that the remaining number of seats for $F^{spill}_{infeasible}$ is negative). Constraints (3.30) define variable domains.

3.3 Solution Methodology

3.3.1 General Framework

Our solution methodology determines a solution space tailored for each disruption, using trained ML classifiers, so that the optimization methods can efficiently find a good solution within the available solution time. The general framework is summarized in Figure 3-8. The *online* phase corresponds to the day of operations, where trained prediction models are used to reduce the solution space by eliminating some of the alternative solutions. The problem is then solved with the help of integer optimization solvers over the reduced solution space.

The solution approach requires an *offline* phase in which solutions to several disruption scenarios are generated using optimization methods and ML prediction models are trained to discover patterns in the solutions. The *Offline* phase starts with

the generation of scenarios based on historical flight operations data. Each disruption scenario corresponds to a specific day and is solved by a direct optimization approach with an optimality gap target and non-reduced solution space. Solutions to disruption scenarios are used to generate the training input for the ML model training step.

The approach summarized above builds on the framework presented in Chapter 2. There are some notable differences in all three steps of the offline phase — scenario generation, offline optimization, and classifier training. The scenario generation step is still based on the historical disruption data, but the scenarios correspond to individual days in history rather than sampling from fitted distributions. Significantly fewer disruption scenarios are required due to each ML prediction model being trained for a set of flights instead of a single recovery decision as in Chapter 2. As a result of fewer scenarios, the optimality gap target in the offline optimization step is lower, resulting in a database with higher quality solutions. Finally, each classifier is trained for a subset of flights in the network instead of individual follow-on pairs.

3.3.2 Limiting the Number of Copies for Individual Flights

The modeling approach presented in Section 3.2.3 uses flight copies to model delay decisions. The number of copies for each flight depends on the selected time increment duration and the maximum allowed delay limit. For example, if the time increment duration is set to 15 minutes and flights are allowed to be delayed by up to 10 hours, then there will be 41 copies for each flight (40 delay alternatives plus 1 non-delay alternative). The flight copies approach is often used in the airline recovery context. The typical approach in the existing literature is to set a time increment and a single maximum allowed delay limit applied to all flights included in the recovery problem. Maher (2016) sets the time increment duration to 30 minutes and the maximum allowed delay limit to 3 hours, resulting in 7 copies for each flight.

The number of flight copies is expected to affect the tractability of the resulting problem. Due to the combinatorial nature of recovery problems, the number of solution alternatives, and hence the solution space, increases rapidly with the number of flight copies. Therefore, past studies typically limit the number of copies by imposing

fixed and identical limits to all flights. However, this approach does not consider disruption characteristics and cannot differentiate between different flights.

To consider the disruption and flight characteristics, we propose using classifier predictions to limit the number of copies, instead of creating the same number of copies for all flights in all disruption instances. One way to achieve that is to predict and fix the cancellation decisions, which would eliminate all copies of the canceled flights from the solution space. However, this approach retains the same number of copies for the remaining flights. The resulting reduction in the total number of flight copies is limited since the average number of flight cancellations is usually in the range of ∼2-3% and typically in single-digit percentages even on days of irregular operations.

A more effective approach is to determine the number of copies for individual flights separately. In the proposed solution methods, we use classifier predictions to determine each flight's maximum allowed delay limit and create flight copies accordingly. Consider a case where the time increments are set to 15 minutes and the *network-wide maximum delay* is set to 6 hours. Here, the *network-wide maximum delay* is the maximum delay limit that is applied to all flights in the network to maintain the tractability of the model while avoiding the elimination of most high-quality solutions. If the delay to a particular flight is predicted to not exceed 1 hour, only 5 flight copies would need to be included instead of 25. Even if we can make such predictions for only 25% of all flights, we can achieve a 20% reduction in the total number of flight copies.

These arguments are supported by our experiments (see Appendix B.1 for more details), which showed that limiting the number of flight copies based on the maximum allowed delay limit predictions for individual flights has a significantly greater potential to accelerate the solution process than using cancellation predictions. Furthermore, leaving the cancellation decisions to the model provides more flexibility and leads to higher-quality solutions.

ML models are trained to discover patterns in the disruption characteristics and flight characteristics that can affect the resulting flight delay in the corresponding

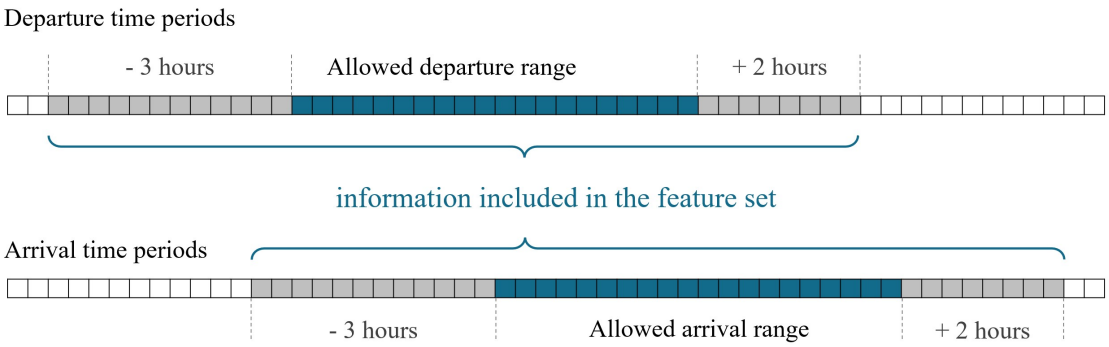


Figure 3-9: Disruption information range included in the feature set

solution. Thus, the disruption characteristics and flight characteristics serve as the set of features for training the ML models.

Our feature set has 80 features belonging to four different information groups. The first group includes information related to disruptions, such as airport capacities and the planned number of departures and arrivals, while the second group consists of information related to the flight schedules, such as fleet type, frequency information, and departure time. The third group is related to the planned crew duties, such as the number of flights in the duty. The last group includes information related to passengers, such as seat capacity and the number of passengers connecting to and from the corresponding flight. Overall, the first group reflects the characteristics of the disruption, while the three later groups include other relevant information so that the same set of classifiers can be used for a wide range of flights. An exhaustive list of all the information included in the feature set for each flight is given in Table 3.1.

To capture the conditions that affect the delay of a flight, we should consider the disruption information that is spatially and temporally relevant. The features in the disruption group cover all the considered departure and arrival time periods, current hour plus 6 hours, as well as an additional 3-hour period before the range and a 2-hour period after the range. Overall, this corresponds to a 12-hour (1+6+3+2) period for each flight. Figure 3-9 shows the included ranges, with the dark-shaded time periods corresponding to the allowed departure and arrival ranges.

Features 59-80 in Table 3.1 reflect the itinerary characteristics of the connecting

Feature id	Information Group	Description
1-24	Disruption	Hourly arrival/departure capacity for a period that contains the relevant range of departure and arrival times for the corresponding airports (ranges from 0 to 30 flight operations per hour).
25-48	Disruption	Number of planned flight arrivals/departures for a period that contains the allowed range of departure and arrival times for the corresponding airports (ranges from 0 to 30 flight operations per hour).
49	Schedule	Total daily frequency of the flight's Origin-Destination (OD) (ranges from 1 to 15)
50	Schedule	Frequency order (The first flight of the day etc., ranges from 1 to 15)
51	Schedule	Departure time
52	Schedule	Flight duration in minutes (in 15-minute time periods, ranges from 3 to 32)
53	Crew	Slack time before the next flight in the crew duty (in 15-minute time periods, ranges from 0 to 16)
54	Crew	Buffer time at the end of the crew duty (calculated as the difference between the maximum legal duty duration and the planned duty duration in 15-minute time periods, ranges from 0 to 48)
55	Crew	Number of legs in the crew duty (ranges from 1 to 5)
56	Passenger	Number of passengers on this flight (ranges from 10 to 175)
57	Passenger	Number of passengers with connection to other flights (ranges from 0 to 175)
58	Passenger	Number of passengers with connection from other flights (ranges from 0 to 175)
59-80	Passenger	Total number of connecting passengers to other flights with respect to planned connection time (ranges from 0 to 175)

Table 3.1: Feature set

passengers on the current flight. *Planned passenger connection time* refers to the time between the scheduled departure of the next flight on a passenger's itinerary and the scheduled arrival of the current flight. For example, if all connecting passengers had more than four hours before their next flight, delaying the current flight by two hours would have less impact on passengers compared to the case where all connecting passengers had only one hour before their next flight. We define an encoding where we bin connection times into 15-minute-long intervals, count how many passengers belong to each interval, and include the set of counts in the feature set. Let us say that we have 60 passengers in total connecting to other flights. If 10 of are connecting in 1 hour, 20 in 2 hours and 30 in 3 hours, this part of the feature set would look something like the following: 0, 0, 10, 0, 0, 0, 20, 0, 0, 0, 30,... etc. The feature set includes the number of passengers with respect to their planned connection times ranging from 2 time periods (30 minutes) to 23 time periods (5.45 hours).

Based on the information provided in the feature set, classification models are trained to estimate the conditional probability (denoted as PRB) of a flight being delayed less than or equal to a specified limit, called the *delay limit separator*, in a high-quality solution to the given disruption. Each flight is evaluated using two sets of classifiers, one each focusing on departures and arrivals.

Since the delay of a flight either exceeds the specified limit or not, the classification task has a binary structure, and the resulting classifiers are called binary classifiers. They separate the flights into two groups. Figure 3-10 illustrates an example where the delays for ten flights are evaluated based on a delay limit separator value of one hour. The delays of the flights in the first group, with label 1, are predicted to be less than the separator; therefore, their maximum allowed delay is set to 1 hour in the optimization run. The flights in this group have 5 flight copies corresponding to one non-delay copy, as well as four delayed copies with up to 1 hour delay. On the other hand, for the flights in the second group, with label 0, which are predicted to be delayed more than the separator, the number of flight copies is not reduced.

The training criterion is selected as the *precision* or *Positive Predictive Value* (PPV), which is the ratio of true positive predictions to all positive predictions. This

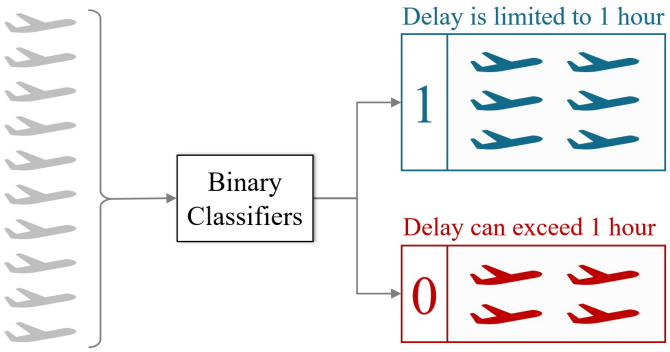


Figure 3-10: Binary classification of maximum allowed flight delays

is because we only reduce the solution space when the label is 1 (positive prediction), by limiting the number of copies for individual flights based on the predictions of the delay exceeding the corresponding delay limit separator value. Details on classifier training are provided in Section 3.4.2. Due to our modeling approach and the way we reduce the feasible solution space, classifier training is less time-consuming compared to the methods proposed in Chapter 2.

3.3.3 ML-guided Solution Space Reduction

As discussed earlier, our solution space reduction procedure in the online phase is guided by using ML predictions for flight delays and setting a delay limit specific to each flight. Following a process similar to that outlined in Chapter 2, both the PRB and the out-of-sample PPV performance of the corresponding classifiers are considered when reducing the solution space. *Prediction confidence (PC)*, is calculated for each classifier prediction as, $PC = PPV^{\alpha} * PRB$. The evaluation of the predictions is similar as well. The PC values are compared against a prediction confidence threshold value ($PC_threshold$), which is a pre-specified value that determines whether the constraints corresponding to a prediction should be added to the model. The value of α, which determines the relative importance of PPV and PRB, is set to 2 after evaluating a set of candidate values (Appendix B.6). It implies that for the integrated recovery experiment setup, overall classifier precision performance, PPV, is

more crucial than the prediction probabilities, PRB, in reducing the solution space effectively without sacrificing the solution quality.

Here are the main steps in the solution process.

1. The values of the parameters *delay limit separator*, *network-wide maximum delay*, and *PC_ threshold* are set.

2. For each flight i, the classifier prediction regarding whether or not flight i's delay would remain under the *delay limit separator* value, and the corresponding PC values, are calculated.

3. If the PC value for the flight i, $PC(i)$, is at least equal to the *the prediction confidence threshold (PC_ threshold)* value, the maximum allowed delay limit for the flight is set to the *delay limit separator*; otherwise, it is set to the *network-wide maximum delay.*

4. The set of time periods for which each flight i is not allowed to arrive ($T_i^{r_{not\_allowed}}$) due to the newly introduced limit is populated.

5. Constraints (3.31) are added to the model so that flights do not arrive during the time periods in $T_i^{r_{not\_allowed}}$, thus removing the flight copies that are not allowed.

$$r_i^{a,t} - r_i^{a,t-1} = 0 \qquad \forall i \in I, \forall a \in A, \forall t \in T_i^{r_{not\_allowed}} \tag{3.31}$$

6. The IRM model (3.1-3.14) with the added constraints (3.31) is solved using a integer optimization solver. The resulting flight delay and cancellation decisions are fixed, and the post-processing steps are completed to calculate the final recovery cost. Algorithm 2 summarizes the overall flow of the solution method.

The model can delay a flight up to the *network-wide maximum delay* value if no other limit is imposed based on the ML predictions. It can also decide to cancel the flights irrespective of the imposed limits.

The *PC_ threshold* affects the run time and solution quality. Setting it too low accelerates the optimization process, but may lead to lower-quality solutions due to

Algorithm 2 IRM Solution Approach with Post-processing

Initialize:
Load the disruption
Load the IRM
Set the *network-wide maximum delay*
Set the $delay\_limit\_separator$, $PC\_threshold$ values
for $i \in I$ **do**
    if $PC(i) \geq PC\_threshold$ **then**
        Determine the set $T_i^{r_{not\_allowed}}$
        for $a \in A, t \in T_i^{r_{not\_allowed}}$ **do**
            add the constraint $r_i^{a,t} - r_i^{a,t-1} = 0$ to IRM
        end for
    end if
end for
Solve the IRM with added constraints
Complete the post-processing steps

less accurate predictions. On the other hand, setting it too high is not helpful in finding the best solution in the available solution time, because the feasible solution space likely remains too large to handle. The set of threshold values for different available solution time limits is determined in the *offline* phase.

3.4 Computational Study

3.4.1 Network Description and Pre-processing

The selected problem instances are based on a network with 3,706 daily flights from a major US carrier. Training data include solutions to disruptions from 2012 to 2016. The test scenarios are based on disruptions from 2017. The *network-wide maximum delay* parameter for flight delays in the experimental setup has a considerable impact on the solution time and quality. We set it to 6 hours based on an out-of-sample analysis provided in Appendix B.4.

The airline in our case study does not operate overnight flights. Therefore, it

is assumed that the line checks mentioned in Section 3.2.2 are handled overnight. Furthermore, the entire fleet of this airline belongs to a single fleet family ensuring that the same set of crews can operate all flights. Consequently, the number of aircraft groups in this network is 1 for the IRM model. Although all aircraft are in the same fleet family, there are two different types of aircraft with 143- and 175-seat configurations, respectively. These differences are considered by the ARM in the post-processing phase. The passenger itineraries used in the computational study are estimated using the method by Barnhart et al. (2014).

3.4.2 Offline Phase

In the *offline phase*, a solutions database is generated based on the selected disruptions from 2012 to 2016, from the months of January and February. 48 disruptions, with the most delays and cancellations, are selected as the training set. They correspond to 16.16% of all days during these 10 months (297). There are more than 100,000 flight observations in the 48 selected disruptions.

Training inputs are generated from the solutions database. Thirty separate classifiers are trained, corresponding to groups of flights departing from and arriving at the 15 airports in the network with the highest level of flight traffic. Each classifier's training input has around 7,000 instances on average, which was found to be sufficient to train high-precision classifiers if discoverable patterns exist. Binary classifiers are trained to separate the data points into two groups based on a selected delay limit separator value. Positive predictions correspond to observations with a delay value less than or equal to the delay limit separator. Negative predictions correspond to observations with a delay value greater than the delay limit separator. The tested set of separation limits ranged from 15 minutes to 3 hours.

The classification methods used in training included *Optimal Classification Trees (OCT)*, *Random Forest (RF)*, *XGBoost (XG)*, and *Logistic Regression (LR)*. OCT was selected for the interpretable structures of the resulting models. LR was selected as it provides an alternative type of interpretable results based on feature weights. RF and XG are both tree-based classifiers. Although they do not provide interpretable

results, they were included because of their precision performance (Breiman, 2001, T. Chen & Guestrin, 2016).

Figure 3-11 presents the average classifier precision performance of different methods and delay separator values. The red line corresponds to the real frequency of the observations. OCT and XG performed consistently better than RF and LR. Therefore, we used classifiers trained by the OCT and XG methods while determining the delay limits for each flight. For classifier insight analysis, we only used classifiers trained by the OCT method because of their interpretable structure.

For all delay limit separator settings, the binary classifiers provided improvements over the actual database frequencies. This means that whichever delay limit separator is selected, classifiers can be used to predict whether the flight delay would be less than the limit in a way that is more accurate than directly using the database statistics. However, from a solution acceleration point of view, not all separator limit settings have similar advantages. For instance, if the separator limit is set to 3 hours, even though we can accurately predict the flights that should have a delay limit of 3 hours, the solution space would remain large, due to the limited number of flight copies removed, which would consequently prevent finding high-quality solutions quickly.

During the offline phase, we evaluated alternative separator limits in terms of both their precision and their solution process acceleration potential. The best settings turned out to be 30 and 45 minutes. Details of this analysis are provided in Appendix B.5. During the experiments presented in the next section, the separator limit is set to 45 minutes. The binary classifiers predict the flights that do not need to be delayed beyond 45 minutes. The maximum allowed delays for such flights are limited to 45 minutes, and the remaining flights are allowed to be delayed by up to 6 hours. In other words, the former group of flights have 4 flights copies, 1 non-delay and 3 delay copies, while the latter group of flights have 25 flight copies, 1 non-delay and 24 delay copies.

The $PC\_threshold$ values, described in Section 3.3.3, are determined in the offline phase based on a set of out-of-sample disruption scenarios, called the calibration set. The specific values depend on the available solution time limit. See Appendix B.7 for

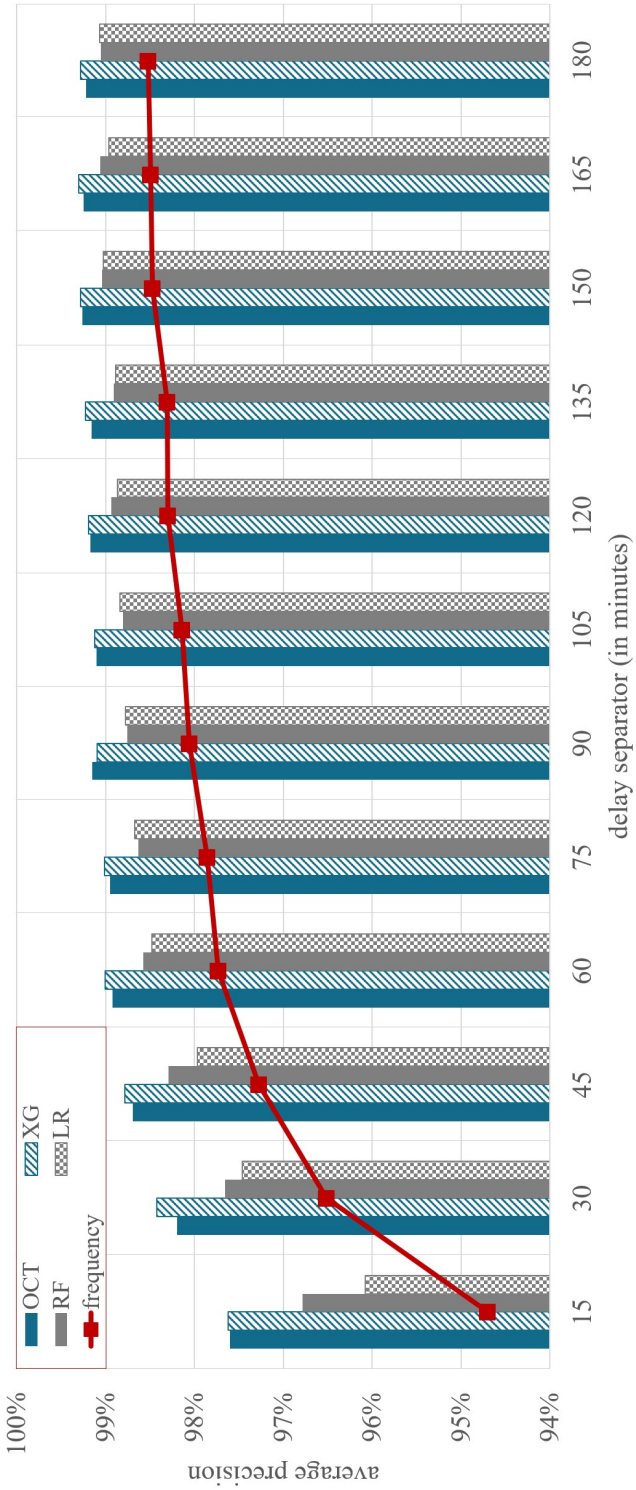


Figure 3-11: Average classifier precision performance

more details.

3.4.3 Results

The scenarios used for calibration and testing are based on actual flight operation days from January 2017 and February 2017 respectively. The underlying network corresponds to a Wednesday in January 2019. All predicted disruptions throughout the day are considered in the model and the recovery period spans from the scheduled departure time of the first flight in the current day's schedule until right before the scheduled departure of the first flight in the next day's schedule. We compared the quality of the IRM solutions found by the following five methods in under 3 and 5 minutes. All experiments are carried out on a desktop computer equipped with an Intel i9-13000K CPU and 64 GB of memory using Gurobi 10.0 (Gurobi Optimization Inc., 2020) as the integer and mixed-integer optimization solver.

1. *Default*: The IRM model is sent directly to the Gurobi optimizer with the specified solution time limit. There is no solution space reduction in this method.

2. *Swap*: This method resembles the methods used in practice. It focuses on repairing the broken rotations. A rotation is considered broken if it becomes infeasible due to disruptions. In addition to the broken planned rotations, the algorithm picks two feasible planned rotations based on the number of swap opportunities with the broken rotations and adds them to the solution space. The number of swap opportunities between two aircraft rotations is defined as the cardinality of the set of ordered pairs of flights in these two aircraft rotations that can connect with each other where each flight belongs to a different aircraft rotation. The remaining feasible planned rotations are kept intact. The delays of the flights in the rotations kept intact are limited to the delays of the corresponding flights in the solution. The resulting IRM with reduced solution space is sent to the Gurobi optimizer.

3. *Sequential*: This is the same as the default method, except that the underlying integrated model does not have crew and passenger considerations. Unlike the

swap method, this approach does not fix any planned rotations. See Appendix B.2 for the complete model.

4. *DB stats*: Flight delay limits are determined using the database statistics, and the resulting IRM with reduced solution space is sent to the Gurobi optimizer. Details of the method are provided in Appendix B.3.

5. *ML*: Flight delay limits are determined based on ML predictions, and the resulting IRM with reduced solution space is sent to the Gurobi optimizer.

The cost parameters used in the computational study are given in Table 3.2. The cancellation cost and the cost of deadheading a crew member to the crew base are based on the values used in other prominent studies (Petersen et al., 2012). The cost of crew delay per hour, fuel & oil, and maintenance costs per block hour for each sub-fleet type are based on the estimates in the benefit-cost analysis conducted by FAA (2020). The latter two are used to calculate the savings from flight cancellations. Regular and high-cost reserve crew costs are assumed to be 25% and 50% higher than standard crew duty costs, respectively. These values are based on discussions with airline practitioners from the specific airline in our case study. The passenger delay cost per hour per passenger is based on the analysis conducted by Cook and Tanner (2011). The cost of a stranded passenger includes delay, accommodation, and loss of passenger goodwill costs. The value, $1,000, is estimated based on the values used by other studies in the literature (including, Vink et al. (2020)).

As discussed in Section 3.2.3 the solution approach requires post-processing steps for aircraft, crew, and passenger recovery to be completed which are formulated as described in Section 3.2.5. Figures 3-12 and 3-13 compare the solution qualities before and after these steps have been completed, respectively. The *default* method could not find feasible solutions within the available solution time limits. Therefore, no results for the *default* method were included in the comparisons. Note that the x-axis of both figures lists the IRM runtime, which excludes the post-processing time. The y-axis of Figure 3-12 lists recovery cost differences calculated based on the approximate costs of the IRM objective function while the y-axis of Figure 3-13 lists recovery cost

Parameter	Value
Fuel & oil operating cost per block hour (B737-700)	$1,750
Fuel & oil operating cost per block hour (B737-800)	$2,050
Maintenance operating cost per block hour	$750
Cost of a crew member deadheading to the location of the next day's duty start	$2,000
Cost of crew delay per hour	$720
Cost of a stranded passenger	$1,000
Cost of delay per hour per passenger	$20

Table 3.2: Cost parameters used in the computational study

differences calculated based on the actual realized total recovery costs according to those obtained after running all the post-processing steps. Post-processing steps are identical for all compared methods, and the post-processing time is around 3 minutes.

The same information as in Figures 3-12 and 3-13 regarding the average recovery cost differences with respect to the baseline solutions under different solution time limits are provided in Tables 3.3 and 3.4. All the recovery cost differences for any given method are calculated as the recovery cost obtained by that method minus the recovery cost obtained using a baseline method, and then the difference is reported as a percentage of the recovery cost obtained using the baseline method. The baseline method refers to sending the IRM model directly to Gurobi with a 0.1% optimality gap target and a 2-hour run time limit. The average optimality gap target achieved was ∼2%.

IRM solution time limit (minutes)	Default	Sequential	Swap	DB stats	ML
3	N/A	20.93%	13.94%	6.40%	2.67%
5	N/A	20.93%	11.88%	6.39%	2.50%

Table 3.3: Average recovery cost difference of the final solution before the post-processing steps with respect to the baseline solutions

The solution quality improvement gained by extending the solution time limit from 3 to 5 minutes was limited for all methods. The *sequential* recovery method provides

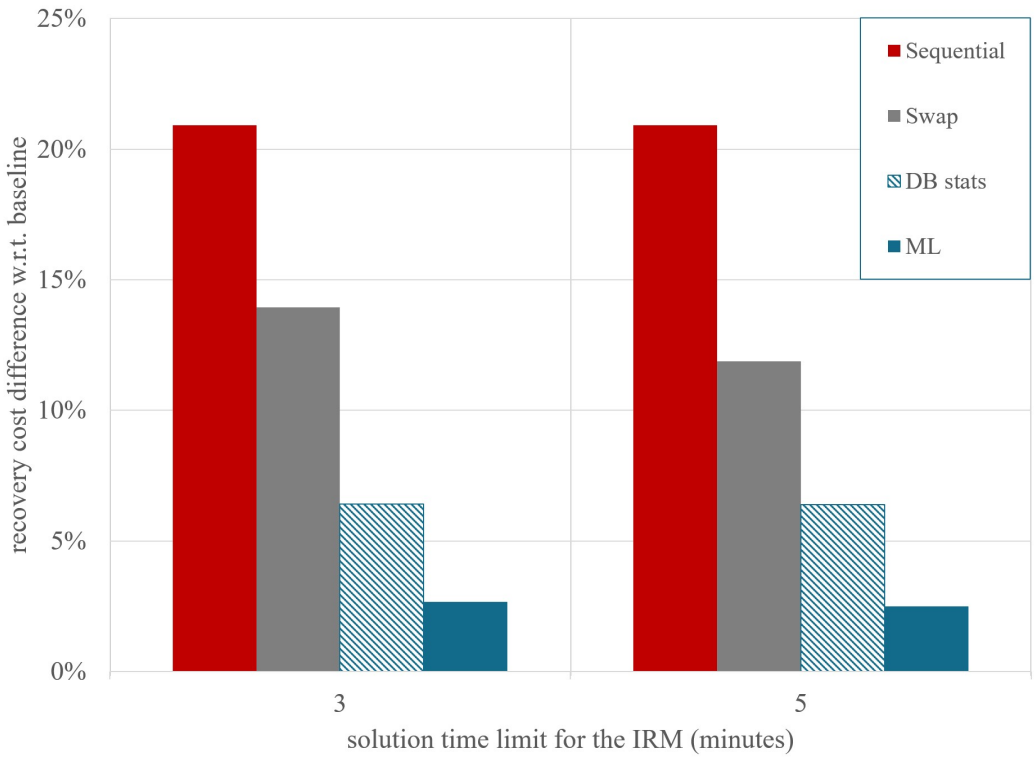


Figure 3-12: Solution quality comparison (before post-processing)

an acceleration over the *default* method, due to the simplified mathematical model, but its solution quality is worse than other methods. The *swap* method consistently finds better solutions than the sequential approach.

Data-driven methods, *DB stats*, and *ML*, perform significantly better than others. Despite relying on the same solutions database, the *ML* method generated 2-3 times better solutions than the *DB stats* method based on recovery cost differences with respect to the baseline solutions. These results show that while a straightforward data-driven approach such as flight-based analysis in the DB stats method can achieve improvements, training *ML* prediction models using the same solutions database provides significantly better results.

There are three aspects of our ML solution method that contribute to its superior performance. First is the integration of crew and passenger considerations into the main IRM model. This helps to achieve significant solution quality improvement over the *sequential* method. Second is reducing the solution space considering

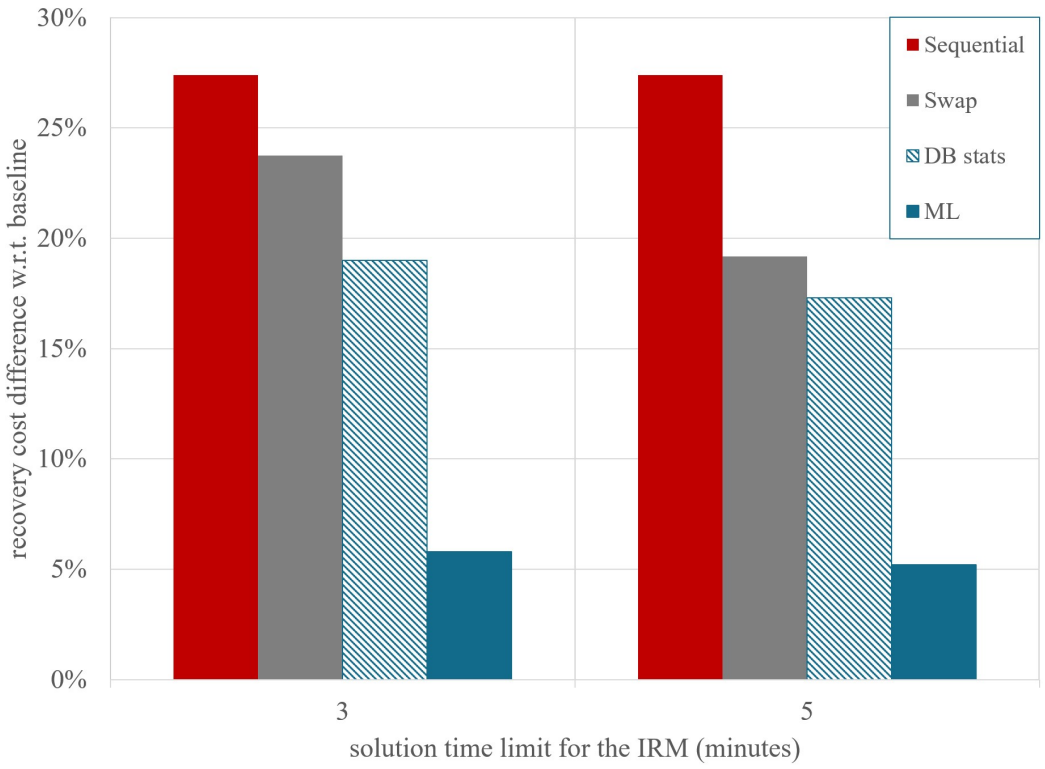


Figure 3-13: Solution quality comparison (after post-processing)

the disruption characteristics (as opposed to keeping some of the planned rotations based on simple heuristics as in the *Swap* method). Third is the integration of ML, and the flight-specific classification of likely delay. This contributes to the quality improvement as compared to the other data-driven method, *DB stats*.

We calculated approximate monthly savings for this airline's network to quantify the recovery cost savings our *ML*-based approach could provide over other approaches. Annual savings over the next best-performing method, *DB stats*, were found to be around ∼$15.5 million and ∼$15 million for 3- and 5-minute solution time limits, respectively. These figures are adjusted for inflation to 2019 dollars.

IRM solution time limit (minutes)	Default	Sequential	Swap	DB stats	ML
3	N/A	27.40%	23.75%	18.99%	5.81%
5	N/A	27.40%	19.18%	17.31%	5.22%

Table 3.4: Average recovery cost difference of the final solution after the post-processing steps with respect to the baseline solutions

3.5 Discussion

3.5.1 Solution Analysis

To understand why the solutions generated by the *ML* method have a higher quality than those from other methods, we analyzed the solutions produced by the *ML* method and the next best-performing benchmark, the *DB stats* method. The results show that for ∼8% of the flights, the delay limits imposed by the DB stats method are 3 hours longer than the limits imposed by the *ML* method. These higher delay limits set by the *DB stats* method make the corresponding solution space reduction efforts less efficient than those performed by the *ML* method.

We also compared the solutions generated by both methods with the baseline solutions. The comparison result shows that, in the baseline solutions, ∼5.4% of the flights were delayed beyond the delay limit determined by the *DB stats* method, and ∼3.8% of the flights were delayed beyond the delay limit specified by the *ML* method. These results imply that our ML-based approach allows selecting the overall delay limits in a smarter, more accurate way than the *DB stats* method, leading to higher quality solutions. The main contributor to this performance is the fact that the *ML* method considers the disruption and individual flight characteristics while determining the solution space, but the *DB stats* method relies on aggregate flight statistics of the solutions database.

Due to the effectiveness of solution space reduction and the higher accuracy of delay limit predictions, the *ML* method generates higher-quality solutions than the *DB stats* method.

3.5.2 Classifier Insights

An advantage of the selected tree-based classifier methods is that they provide interpretable models that help explain the reasons for recovery decision predictions. Airline flight operation controllers are usually reluctant to apply suggestions made by the decision support systems if the rationale is unclear. If integrated into such systems, the interpretable structure of the trained classifiers helps build user trust and allow the controllers to make more informed decisions.

The solution method presented in Section 3.3 considered both interpretable and non-interpretable classifiers, but in 100% of the cases, the classifiers chosen by our ML-based integrated recovery solution ended up being the interpretable ones.

Figure 3-14 shows a classifier example that focuses on flight arrivals. It is trained to predict whether the delay of a flight arriving at LAX should be limited to 1 hour. The topmost node is the root node where the algorithm starts. The numbers 0 and 1 correspond to the prediction labels at that node. The label is 1 if the classifier suggests that the flight delay is limited to 1 hour, 0 otherwise. Nodes without outgoing branches are the leaf nodes corresponding to predictions and are shaded red or blue, depending on the label. The remaining nodes are internal nodes. All non-leaf nodes are shaded white. The percentage values on the nodes are the prediction probabilities for the leaf nodes and the node composition ratios for the others. The node composition ratio refers to the ratio of the number of observations with the dominant label in the node to the number of all observations in the node. The thickness of the lines connecting the nodes represents the fraction of observations from the parent nodes that fall into the child nodes.

One of the interpretations of the classifier structure in Figure 3-14 is that, in addition to the disruption characteristics, the decision to delay a flight beyond 1 hour should also consider the number of passengers on the flight. The lower the number of passengers, the higher the probability of delay exceeding the 1-hour limit. For example, if the arrival capacity at the scheduled arrival time is low during the 0^{th} and the -1^{th} hours, and the number of passengers is less than 82.5, the classifier

predicts that the flight delay will exceed 1 hour with 89.29% probability. It should be noted that while the labels on the leaf nodes reflect the classifier prediction, our solution method also uses the predicted probabilities as described in Section 3.3.3. We can limit the delay to 1 hour with confidence if the flight falls into one of the high-probability leaf nodes, such as those with 99.24% and 100% probability in Figure 3-14. We would not have the same confidence for flights that fall into the node that corresponds to a probability of only 64.27%.

Figure 3-15 shows a classifier example that focuses on flight departures. It is trained to predict whether the delay of a flight departing from LAX should be limited to 1 hour. When the departure and arrival capacities at scheduled times are higher than a certain threshold, the classifier suggests limiting the flight delay to 1 hour. For the cases where the arrival capacity is low and the planned number of arrivals before the scheduled arrival time is high, the classifier suggests allowing the flight to be delayed beyond 1 hour.

A variable importance analysis is performed, including for all trained classifiers. The results show that the most important information group is the disruption-related one with 79% importance, which includes airport capacities and the planned number of flight operations (Figure 3-16). This result is as expected and it confirms the strong dependence of the recovery decisions on disruption conditions, which helps us discover predictive signals in the training data. Further analysis revealed that the disruption and planned flight operation information corresponding to the arrival time periods is more important than the information corresponding to the departure time periods.

The second most important information group is the passenger related one with 14% average importance. The numbers of non-stop and connecting passengers seem to be crucial in determining the delay of a flight. Classifiers suggest keeping the delay of a flight low when too many passengers are impacted.

The schedule and planned crew duty-related information have 5% and 2% importance, respectively. These results indicate that the crew duty-related information included in the feature set, such as the slack time before the next flight in the crew duty, does not provide much value for predicting the flight delays.

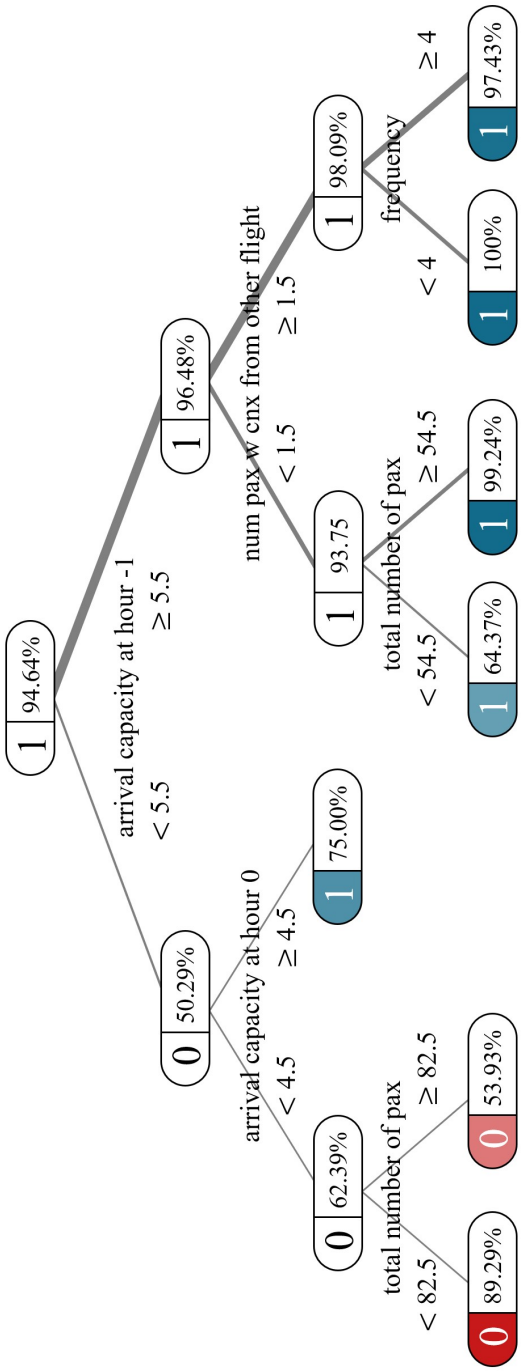


Figure 3-14: Classifier example for LAX arrivals

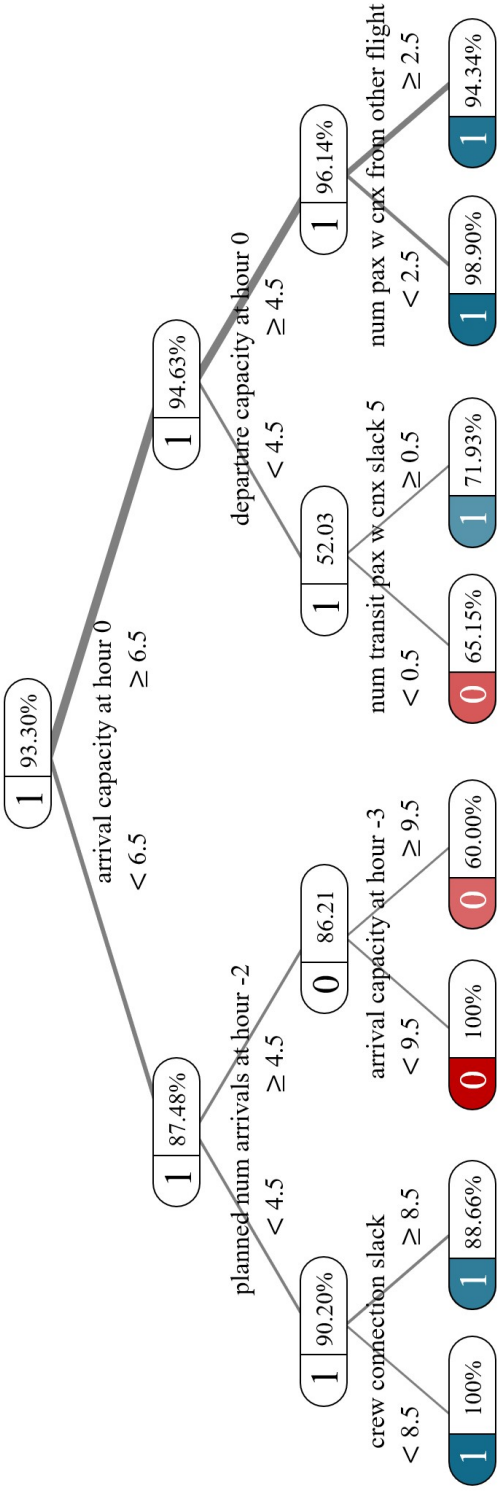


Figure 3-15: Classifier example for LAX departures

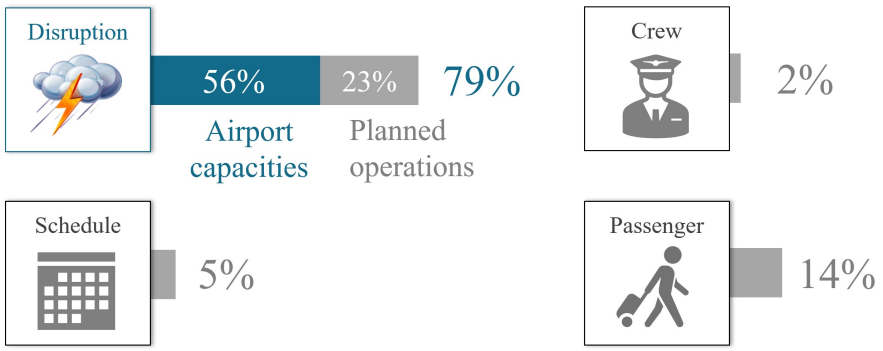


Figure 3-16: Variable importance distribution among information groups

3.5.3 Crew vs. Passenger Recovery Trade-off

The variable importance analysis conducted in the previous section provides some insights into which type of information is more relevant in predicting the delay of a flight. A related, but different, set of questions include: what are the relative costs of crew and passenger recovery components? Also, what is the relative difficulty associated with recovering crews and passengers? These types of questions are the focus of this section.

The presented integrated solution method has an inherent exploration-exploitation trade-off that allows evaluating the relative importance of crew and passenger considerations in an integrated recovery approach. This is due to the fact that the integrated model includes major crew and passenger recovery considerations, such as infeasible crew duties and broken passenger itineraries. In our context, exploitation refers to trying to minimize infeasible crew duties and broken passenger itineraries in the IRM, whereas exploration refers to leaving some of the crew duties infeasible and passenger itineraries broken intentionally in the IRM, assuming that some of them would be recovered in post-processing. These are the two opposite recovery strategies that need to be balanced.

The objective function (3.1) has two components that reflect the approximate costs of the broken passenger itineraries $\left(\sum_{f \in F} PC_f \cdot q_f\right)$ and of the crew duties becoming infeasible $\left(\sum_{s \in S} CC_s \cdot u_s\right)$. Since our solution method involves post-processing steps to finalize the solution, we cannot be sure whether passengers on broken itineraries

or crews on infeasible duties can be feasibly reassigned in the post-processing steps. Therefore, the corresponding cost coefficients, PC_f and CC_s, in the objective function reflect approximate values rather than actual values. The actual recovery cost of the solution is calculated after the completion of the post-processing steps and is based on the actual passenger itinerary costs and actual crew duty costs.

It is crucial to carefully select the values of PC_f and CC_s so that the overall solution method can strike the right balance between avoiding infeasibilities up front and allowing them to be handled in post-processing. Setting these cost estimates higher results in a solution that has fewer infeasibilities to be handled in the post-processing steps, but the resulting total recovery cost can be too high. On the other hand, setting them lower would generate many infeasibilities to repair in the post-processing step, which may not be possible, and thus produces a higher cost solution.

PC_f, $f \in F_{infeasible}$ and CC_s, $s \in S$ values are calibrated in the offline phase to achieve the overall minimum recovery cost after completing the post-processing steps. The ratio of PC_f to the actual cost of the broken itinerary f and the ratio of CC_s to the actual cost of the crew duty s becoming infeasible are measures of how difficult it is to recover passengers and crew in the post-processing steps. Based on calibrated values, these ratios were in the ranges of $[0.6, 0.9]$ and $[0.3, 0.5]$ for crew and passenger cost components, respectively. These results imply that it is more challenging to repair infeasible crew duties than passenger itineraries, likely due to crew duty feasibility rules. Therefore, it is better to allow passenger infeasibilities in the main IRM model's solutions rather than crew infeasibilities. This also coincides with practice, because passenger recovery is almost always addressed after crew recovery.

3.6 Conclusion

The objective of this study was to develop a framework that would help find high-quality solutions for integrated recovery problems within limited timeframes. To achieve this goal, we combined optimization and machine learning methods.

We presented a novel and tractable mathematical model for the integrated recov-

ery problem that integrates major recovery decisions, inspired by a modeling idea introduced by Bertsimas and Patterson (1998) in the context of air traffic flow management. As shown by Bertsimas and Patterson (1998), their modeling approach can lead to strong formulations and improved tractability.

The primary recovery actions available to airlines are delaying and canceling flights. The underlying mathematical formulation uses flight copies to model these decisions. Predictions regarding the delay and cancellation decisions can be used to reduce the number of flight copies. We demonstrated that the former has greater potential for accelerating the solution process. Therefore, we used predictions of the maximum allowed delay limits for individual flights to reduce the solution space.

We created a feature set for training ML prediction models that contains information on disruption characteristics as well as schedule, crew duty, and passenger-related information. This feature set allows training classifiers applicable to a large subset of flights instead of one classifier per flight.

We highlighted the advantages of incorporating ML techniques for solving this model and conducted a wide range of experiments to analyze the performance of the presented approach. Our findings indicated that reducing the feasible solution space significantly speeds up the optimization process, and ML techniques are able to reduce the problem size without significantly compromising solution quality. We developed solution methods that determine a restricted solution space tailored to each disruption by removing some regions of the feasible solution space based on ML predictions. Our approach created near-optimal solutions in less than 5 minutes for large-scale flight networks. The flight networks handled in this study were among the largest in the literature.

Our experiments further revealed a relationship between solution quality and the database and ML-guided solution space reduction for each problem instance and available solution time. We observed that solution space reduction does not have an impact when the available solution time is sufficiently large. However, for most practical cases, a reduction strategy exists that provides the best solution quality. Our framework relies on finding good solution space reduction strategies for each problem

instance and adapts to the available solution time limit. Such adaptive ability is not found in existing solution space reduction heuristics in prior studies.

The modeling approach, the solution space reduction strategy, and the feature set definition for ML training help to increase the practicality of our approach. Although the proposed solution approach requires an *offline* phase for each flight network, we showed that solving less than 50 disruption scenarios and using the resulting solutions database is sufficient to reap significant benefits of our approach.

The variable importance analysis showed that the disruption information is the most relevant information in predicting the maximum allowed delay for each flight. Disruption information includes features related to airport departure and arrival capacities and the planned number of operations. The latter is also crucial because it determines the level of congestion, which limits arrivals and departures. Our results of the analysis showed that among the four groups of information included in the feature set, the second-most important group of information is passenger-related, with greater importance than crew and schedule-related information combined.

Our integrated recovery model which includes crew and passenger recovery considerations has trade-off mechanisms between avoiding crew and passenger infeasibilities in the main model and allowing them to be handled in the post-processing step. Trade-off mechanisms are regulated by coefficient values reflecting approximate crew and passenger recovery costs. The ratios of approximate and actual crew and passenger recovery costs imply that recovering from crew infeasibilities is more challenging than recovering from passenger infeasibilities. This result is consistent with airline practices in which crew recovery is typically solved before passenger recovery.

Chapter 4

Rule-based Improvements to Recovery Heuristics

4.1 Introduction

In previous chapters, we developed fast solution methods using optimization and ML tools to tackle airline recovery problems that can generate high-quality solutions in limited timeframes. We also demonstrated that it is practical for airlines to use these methods in their recovery processes. However, their implementations may require significant modifications to airline recovery systems. Implementing these changes can be costly and time consuming for some airlines. In such cases, it may be more practical to improve existing systems than to completely replace them.

Key contributions of the previous two chapters are the practical insights offered by the tree-based classifiers. In both chapters, classifiers are trained to discover the relationship between disruption characteristics and certain recovery decisions. The interpretable structures of the resulting models lead to actionable rules-of-thumb for crew and integrated recovery problems.

The general guidance derived from the trained classifiers and the experimental results can be summarized with the following three principles. First, the rules that constitute the basis of the heuristic and manual solution approaches should consider disruption characteristics. Second, the solution alternatives should be selected based

on a multi-perspective evaluation of the flights, aircraft, and crew resources, as well as planned passenger itineraries, rather than relying on simple rules like filtering based on a time period. Third, the recovery method should seek solutions suitable for a wider range of disruptions, and avoid unconventional solutions overly optimized for a particular delay prediction, to improve robustness against uncertainties.

In addition to providing general guidance, trained classifiers can also help uncover additional actionable insights tailored to specific recovery problems. The interpretable structure of the resulting models allows for determining rules of thumb that otherwise are not easily discoverable. For example, the (follow-on) F/O classifiers trained in Chapter 2 showed that for some F/Os, the most relevant disruption information to consider when deciding whether or not to fix the F/O does not include the flights in the F/O. This is a rather counter-intuitive result that may not be obvious to practitioners.

These types of insights can also help manual recovery processes by providing recovery staff with the ability to understand the rationale behind the suggested solutions, as demonstrated in previous chapters. They also have the potential to improve various existing heuristic-based recovery methods. A common approach adopted by heuristic-based solution procedures in practice and in the literature is to rely on a set of rules of thumb or expert judgments to generate solutions quickly. These rules can be replaced or enhanced with guidance based on the insights provided by the classifiers.

In this chapter, we first examine the classifiers trained in previous chapters. Then, we train higher-level classifiers to discover more generalizable insights. Finally, we suggest ways to utilize these insights to enhance existing heuristic methods and demonstrate performance improvements.

The remainder of this chapter is organized as follows. In this section, we review the previous literature and present the scope and contributions of this study. Section 4.2 focuses on the crew recovery problem with aircraft and passenger considerations, presented in Chapter 2, and proposes ways to discover rules of thumb and improve common heuristics. Section 4.3 follows a similar path of rule discovery and heuris-

tic improvements, focusing on the integrated aircraft, crew, and passenger recovery problem presented in Chapter 3. Finally, Section 4.4 concludes the chapter with a summary and with future research directions.

4.1.1 Literature Review

Due to time constraints, researchers and practitioners frequently adopt heuristic solution approaches when solving airline recovery problems. Some studies use well-known meta-heuristics such as the Greedy Randomized Adaptive Search Procedure (GRASP) (Hu et al., 2016), Simulated Annealing (Gao et al., 2010), and Genetic Algorithm (Yang & Hu, 2019). Others implement problem-specific heuristic approaches.

To address the crew recovery problem, Yu et al. (2003) develop a depth-first search tree algorithm which was also implemented in practice. Zhao et al. (2007) follow a grey programming modeling approach coupled with a local search heuristic. Guo (2005) present a hybrid heuristic using genetic algorithms and local search methods. Novianingsih et al. (2015) develop a heuristic based multi-stage method which is based on construction and improvement heuristics. C.-H. Chen and Chou (2016) propose an approach using combinatorial optimization formulations and a non-dominated sorting genetic algorithm.

The integrated problem is more challenging than tackling any individual recovery step. Therefore, researchers have typically developed heuristic-based, rather than exact, solution approaches for the integrated problem. Aguiar et al. (2011) use a combination of different meta-heuristics, including a genetic algorithm for the aircraft recovery problem and a simulated annealing algorithm for the crew recovery problem. Jozefowiez et al. (2013) develop a three-phase shortest path-based heuristic. The first phase generates an initial feasible solution. The second and third phases recover passengers by reassigning to alternative itineraries and generating new flights, respectively. Zhu et al. (2016) follow a sampling-based algorithmic framework. Their approach involves two separate modules. The first module is focused on schedule and fleet assignment, while the second is focused on crew and passenger recovery. Bisaillon et al. (2011) present a large neighborhood search for aircraft and passenger recovery,

which won the ROADEF 2009 challenge (Palpant et al., 2009), a competition to develop fast solutions for integrated aircraft and passenger recovery problems organized by the French Operational Research and Decision Science Society. Their approach is a multi-phase one, in which an initial solution is generated, repaired, and improved through a large neighborhood search. It relies on solving shortest-path problems for broken aircraft rotations and passenger itineraries.

With recent improvements in mixed-integer optimization solver algorithms and the availability of computational power, many scholars have been developing heuristic solution methods involving mathematical optimization models. These types of approaches are defined as *math-heuristics* or *matheuristics* (Boschetti et al., 2009). For further information on matheuristics developed for different problem contexts, see Archetti and Speranza (2014) and Maniezzo et al. (2021).

The complexity of airline recovery problems makes matheuristic based methods suitable solution alternatives. There is recent interest among researchers in using mathematical optimization as part of a recovery heuristic. Mansi et al. (2012) develop a matheuristic method that solves aircraft and passenger recovery problems simultaneously. Zhang et al. (2016) presents a three-stage matheuristic for the integrated schedule, aircraft, and passenger recovery problem.

4.1.2 Outline

In this chapter, we focus on a special type of matheuristics, commonly used in the airline recovery context. We call it *fix-and-dive matheuristics*, where some of the decision variables are fixed before solving the optimization model. Fixing refers to setting the values of a subset of decision variables, and diving refers to looking for a solution by setting the values of the remaining decision variables (Bixby et al., 1999). Past studies have suggested *fix-and-dive* type of heuristics for different kinds of mixed-integer programming problems (Danna et al. (2007), Salvagnin (2016)).

Figure 4-1 presents a high-level overview of a generic fix-and-dive matheuristic in the airline recovery context. When a disruption occurs, the objective is to find a recovery solution within the overall set of all solution alternatives. This overall set

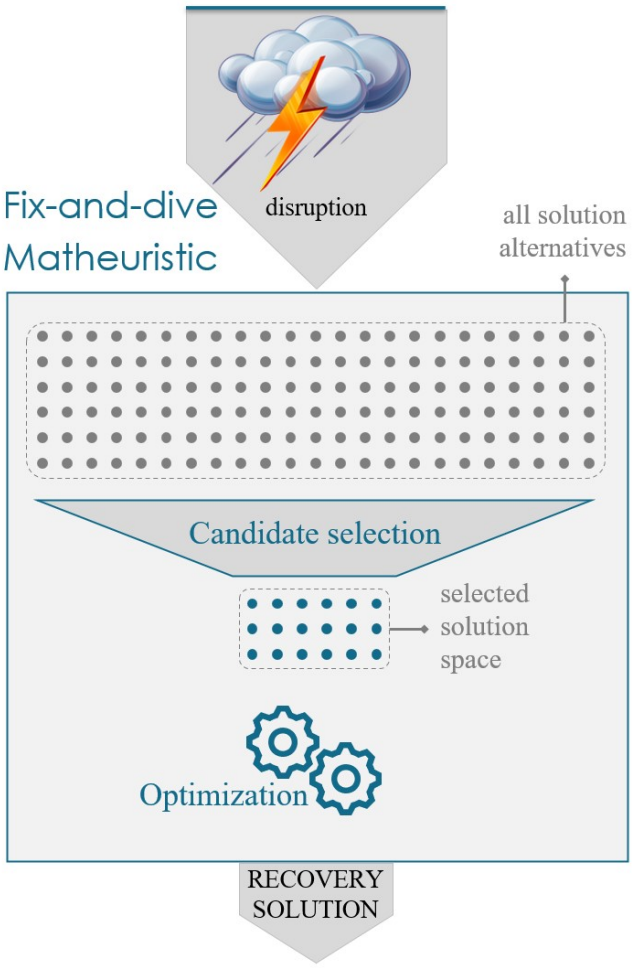


Figure 4-1: Fix-and-dive matheuristic solution approach for recovery problems

of solutions can be extremely large. However, the available solution time does not allow all solution alternatives to be evaluated. This is the main reason why exact optimization methods are not practical for recovery problems. Instead of considering all alternatives, a fix-and-dive matheuristic follows a candidate selection procedure and determines a subset of alternatives to include in the solution space, effectively eliminating many of the solution alternatives, before running the optimization model.

Figure 4-1 is almost identical to the *online* part of the general framework presented in Figure 1-1. This is because, at a high level, the two solution methods are very similar in the sense that both reduce the solution space by picking a subset of solution alternatives. The main difference relates to how the subset of the solution space is selected. A generic fix-and-dive matheuristic determines a reduced solution space by

fixing the decision variables corresponding to a subset of solution alternatives. In our approach, we determine the reduced solution space using ML-based filtering and add constraints to the solution space, in addition to fixing some of the decision variables. The similarity of these two approaches makes it easier to incorporate ML-derived insights into the *candidate selection* step of a fix-and-dive matheuristic.

4.1.3 Contributions

The main contributions of this chapter are listed below.

1) We introduce a new approach for incorporating ML methods into airline recovery processes. Instead of developing new solution methods that may require considerable implementation efforts by airlines, we investigate how existing recovery approaches can be improved by minor modifications without any major changes in the overall solution process.

2) We train high-level classifiers for crew and integrated recovery problems to discover rules of thumb that can replace or enhance the rules used in practice and literature.

3) The discovered crew recovery rules focus on whether or not the F/O is included in the aircraft recovery solution, with additional focus on slack times and the non-propagated delay to the first flight in the F/O.

4) The discovered integrated recovery rules are based on the simple idea that a flight should not be delayed beyond a specific limit if the number of passengers is higher than a threshold value, and this threshold value is found to be different for the flights in the first and second half of the day.

5) We propose methods to integrate the discovered rules into common heuristic-based recovery approaches followed by researchers and practitioners.

6) We demonstrate that the solution quality of such approaches can be improved by up to ∼9% and ∼17%, for crew and integrated problems, respectively, with as few as two rules considered.

7) The sensitivity analysis results demonstrate that the key insights obtained from our ML analysis relate to the general structure of the rule, rather than the exact

threshold values. Given the low dependence of these rules on the exact thresholds, the rule structures might be portable relatively easily across schedules and networks, and can still lead to the bulk of the performance improvements even if the thresholds are not tuned for those particular schedules and networks.

4.2 Crew Recovery with Aircraft and Passenger Considerations

4.2.1 SWAP Matheuristic

A common solution approach for crew recovery problems is to keep the planned crew schedules intact as much as possible. Yu et al. (2003) presents a solution methodology based on this idea that was implemented by Continental Airlines. They keep intact all crew schedules that are still feasible after the disruption and recover from the disruption by fixing the broken crew schedules, generating new crew schedules to cover open flights, and using reserve crew. It was one of the first automated decision support systems implemented for crew recovery in the US. Many airlines followed Continental in implementing similar systems. This heuristic method involves repeatedly solving shortest-path problems instead of an optimization model for the recovery problem.

Since then, due to the increase in computational power available to airlines and advances in optimization algorithms, more sophisticated solution methods that involve solving a reduced optimization problem have been developed. An improved version of the Yu et al. (2003) method is based on the general idea of solution space reduction by adding a few candidate crew members to the solution space, in addition to the disrupted crew, and then solving the resulting optimization problem using mixed integer optimization solvers. Candidate crew members are selected based on swap opportunities between the disrupted crew schedule and the candidate crew schedule. The result is a fix-and-dive type of matheuristic discussed in Section 4.1.2. We call this method the *"SWAP matheuristic"*.

The computational study presented in Section 2.4 shows that the *SWAP* method

generates ∼13% higher costs than the baseline solutions in 5 minutes. In comparison, our ML-based methods generated solutions with only ∼3% higher recovery costs than the baseline. In this section, we use the *SWAP matheuristic* as the benchmark heuristic solution method and improve its performance by incorporating the rules of thumb discovered with the help of tree-based classifiers.

A *swap opportunity* refers to swapping flights between two crew duties and creating new feasible crew duties. A crew duty is a set of consecutive flights that can be operated by a crew. Between two crew duties with multiple flights, there may be several swap opportunities. We quantify the actual number of opportunities to swap flights between a given pair of crew duties, as the cardinality of the set of all unique ordered pairs of flights in these two crew duties that form a connection and are in different duties. Figure 4-2 shows a simple example of how swap opportunities are calculated. It is a Gantt chart based representation of the flights covered by two-leg crew duties. The boxes represent the flights, the capital letters in the boxes represent the departure and arrival airports of each flight, and the lengths of the boxes represent the scheduled flight duration. Let the minimum connection time be 30 minutes. Duty 1 and duty 2 in the example consist of two flights, B-C and C-D, and A-C and C-E, respectively. B-C in duty 1 can connect to C-E in duty 2, since the connection time is longer than 30 minutes. Similarly, A-C in duty 2 can connect to C-D in duty 1, since the connection time is exactly 30 minutes. The result is two new alternative duties, B-C→C-E and A-C→C-D. In this example, we say that there are two swap opportunities.

A high-level overview of the *SWAP matheuristic* is provided in Figure 4-3. We define a crew assignment as a specific crew duty operated by a specific crew. The heuristic separates the planned crew assignments into two sets: the set (1) of feasible assignments and the set (2) of broken assignments. Then, all assignments in Set 1 are ranked based on the number of swap opportunities with the assignments in Set 2. Finally, a predetermined number of high-ranking assignments in Set 1 are included in the solution space, the rest of the assignments in Set 1 are fixed, and the resulting optimization model is solved using mixed-integer optimization solvers.

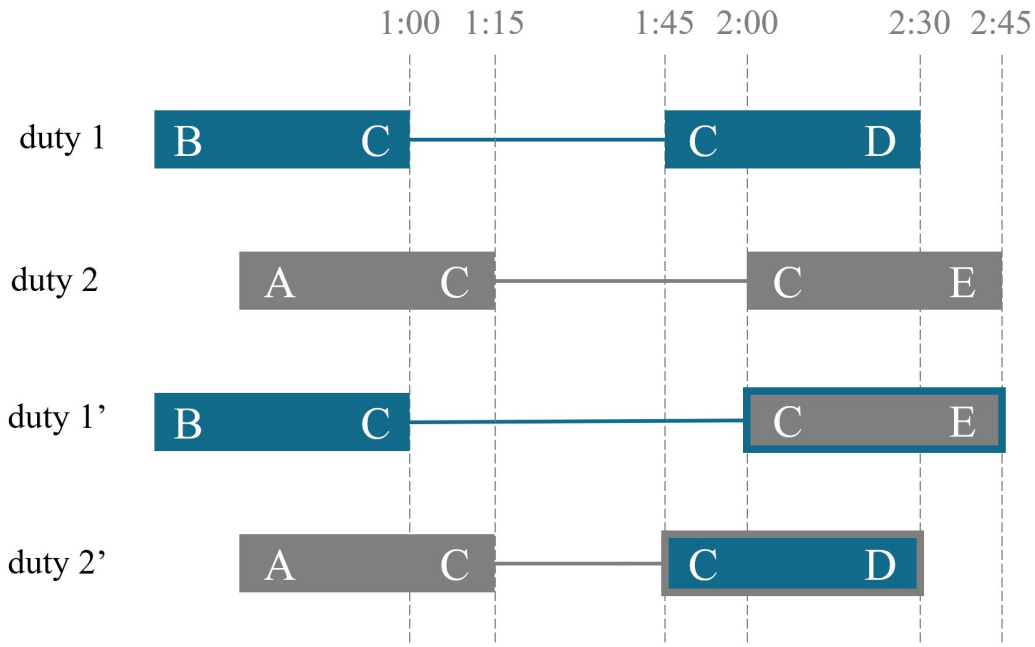


Figure 4-2: Crew swap opportunities determined based on connection alternatives

The crew recovery model used in these experiments is the one presented in Chapter 2 (Model (2.1)-(2.13)). Fixing the assignments corresponds to adding Constraints (4.1) to the model before sending it to the mixed integer optimization solver. Y_1 refers to the set of remaining crew duty assignments in Set 1 to keep as planned. Algorithm 3 summarizes the overall flow of the solution method.

$$y_s^k = 1 \qquad \forall y_s^k \in Y_1 \tag{4.1}$$

4.2.2 Classifier Insights

In Chapter 2, we trained classifiers to predict follow-on (F/O) micro-solutions. The resulting classifiers provide valuable insights into recovery decisions, but are tailored to the underlying network. The proposed methods require an offline phase in which a solutions database is generated and classifiers are trained. This process needs to be rerun for each day.

Algorithm 3 SWAP Matheuristic for Crew Recovery

1: **Initialize:**
2: disruption $\leftarrow \{NP_i, \forall i \in I\}$
3: Load the CRM-APC
4: Determine the set of planned assignments to keep (Y_1)
5: **for** $y_s^k \in Y_1$ **do**
6: add the constraint $y_s^k = 1$ to CRM-APC
7: **end for**
8: Solve the CRM-APC with the added constraints

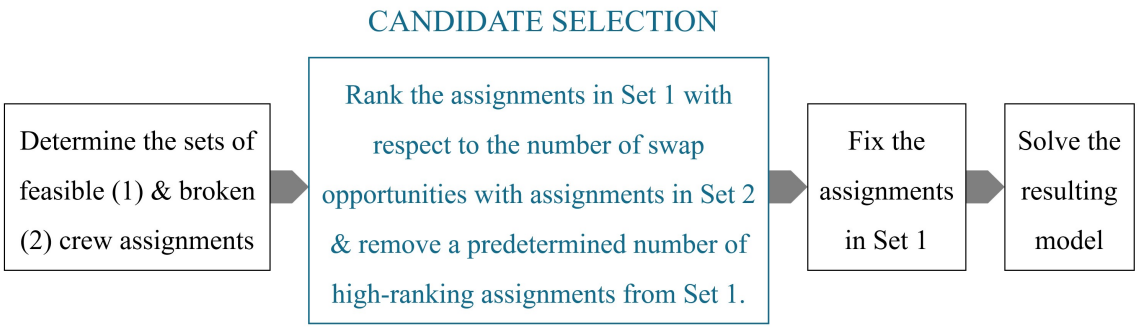


Figure 4-3: High-level overview of the *SWAP matheuristic* for crew recovery

In contrast, in this chapter, we are interested in discovering high-level generalizable insights or rules of thumb. Therefore, we trained a new type of classifier that covers all F/Os instead of focusing on individual F/Os.

The data used for classifier training includes ∼1.2 million data points corresponding to the usage of ∼2,500 F/Os in solutions to 500 disruption scenarios. The *Optimal Classification Trees (OCT)* method is selected because it can provide interpretable results. The maximum tree depth parameter is limited to 5 to keep the resulting model structures interpretable. The trained classifiers are then reviewed to determine common patterns that help derive generalizable rules of thumb.

The training data is separated into four similar-sized groups, quarters, based on the planned UTC arrival time of the first flight in the F/O. This increases the accuracy of the resulting classifiers by focusing on flights during a given time of the day and helps discover insights specific to different time periods. The results of our experiments showed that there are considerable differences in recovery decisions during different parts of the day, such as the first and last quarters of the day.

The feature set is defined to provide aggregate-level schedule and disruption information. It contains 15 attributes from 4 main information groups; *disruption*, *schedule*, *connections*, and *A/C solution* (here A/C is short for aircraft). The disruption and schedule information groups contain only F/O-related information, and are included in the feature set to provide a high-level description of the disruption and schedule characteristics of the F/O. The connections group provides aggregate-level information for the flight connection alternatives. It is added to the feature set to consider the connectivity of the F/O and the relevant disruption conditions. Finally, the *A/C solution* is also a type of F/O level information reflecting the usage of the F/O in the aircraft recovery solution. It is included in the feature set to investigate the correlation between A/C and crew recovery solutions.

An exhaustive list of attributes and their descriptions is given in Table 4.1. Flight block time refers to the time between the scheduled departure and arrival times of a flight. Crew slack time between a pair of flights is the additional duration available between the scheduled arrival time of the first flight and the scheduled departure

Feature id	Information Group	Description
1	Schedule	Flight block time of the first flight.
2	Schedule	Flight block time of the second flight.
3	Schedule	UTC arrival time of the first flight.
4	Schedule	Crew slack time between flights.
5	Disruption	Non-propagated (NP) delay of the first flight.
6	Disruption	NP delay of the second flight.
7	Connections	Number of flights in connection group 1.
8	Connections	Average NP delay of flights in connection group 1.
9	Connections	Number of flights in connection group 2.
10	Connections	Average NP delay of flights in connection group 2.
11	Connections	Number of flights in connection group 3.
12	Connections	Average NP delay of flights in connection group 3.
13	Connections	Number of flights in connection group 4.
14	Connections	Average NP delay of flights in connection group 4.
15	A/C solution	1 if the F/O is used in any rotation in the A/C recovery solution.

Table 4.1: Feature set attributes for crew recovery

time of the second flight beyond the minimum connection or turnaround time of the crew, assuming that the aircraft and crew stay together. The minimum connection or turnaround time is the minimum time required from the arrival of the previous flight to the departure of the next flight to allow consecutive operation of the two flights by the same aircraft or crew. This minimum connection or turnaround time is required for a set of activities including cleaning the aircraft, off-boarding and on-boarding passengers, and for crew getting ready to operate the next flight. Training input also includes a binary label for each data point, reflecting the usage of the corresponding F/O in the crew recovery solution. The label is 1 if the F/O is used, 0 otherwise.

There are four different connection groups depending on the timing of the connec-

Connection group	Description
1	Flights that can connect to the first flight.
2	Flights that the first flight can connect to.
3	Flights that can connect to the second flight.
4	Flights that the second flight can connect to.

Table 4.2: Flight connection groups

tion and the flight in the F/O. Group descriptions are provided in Table 4.2. These groups are introduced in order to have a generalizable set of features to represent the connectivity characteristics of each F/O. Smaller connection group sizes mean that the corresponding F/O has fewer alternatives and hence may still be the best viable option in case of disruptions. On the other hand, larger connection group sizes mean that there are many alternative connections to the current F/O. Therefore, the F/O can be more easily replaced even under minor disruptions.

An example of a classifier for the first quarter of the day is provided in Figure 4-4. The classifier identifies that, for this case, whether or not the F/O is in the A/C recovery solution is the most crucial piece of information in determining the usage of the F/O for crew recovery. If the F/O is in the A/C solution, the classifier considers the NP delay 1, NP delay 2, and slack time information to decide whether to suggest the usage of the corresponding F/O for crew recovery. The main takeaway from the classifier is that more efficient connections with shorter slack times are preferred.

Classifier examples for the other quarters of the day are presented in Appendix C.2. They suggest that for the connections in the second half of the day, some extra information becomes relevant in addition to the information used in the classifiers for the first half of the day. The quarter label for a connection is determined based on the planned UTC arrival time of the first flight in the connection. The flight durations and arrival time of the first flight are the additional pieces of information considered by the third and fourth quarter classifiers, respectively.

We conducted a variable importance analysis to understand the relative impor-

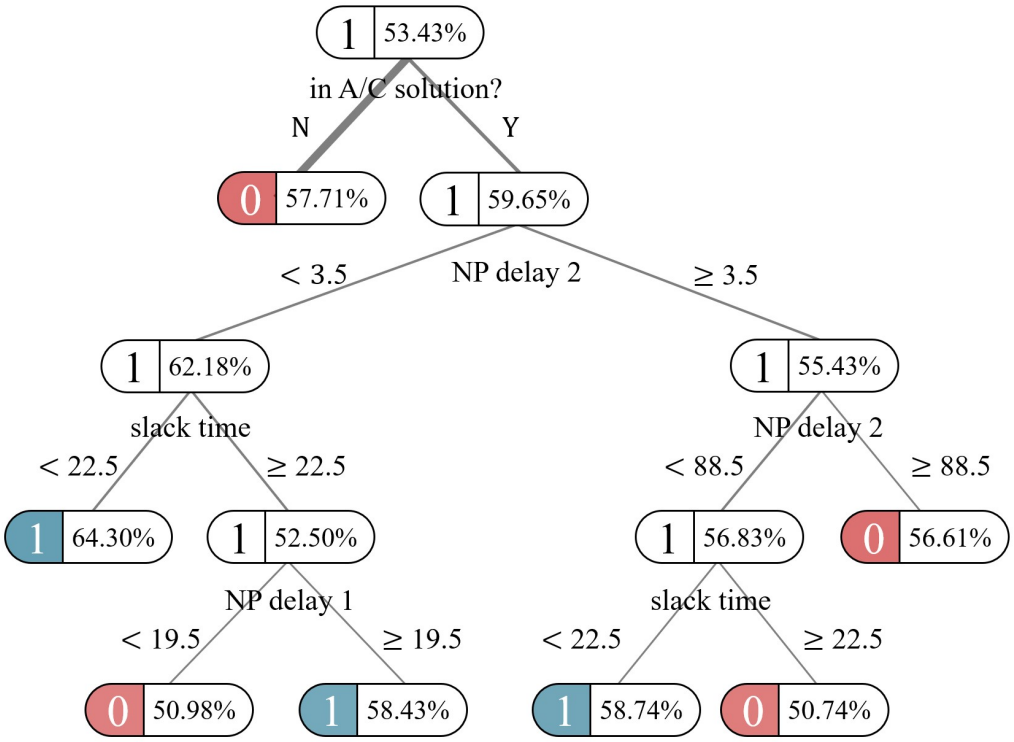


Figure 4-4: Crew classifier example for connections in the first quarter of the day

tance of these information groups. The results are provided in Table 4.3. The most important information is whether the F/O is included in any of the rotations in the A/C recovery solution. The reason for the significant improvement in importance compared to the F/O specific classifiers in Chapter 2 could be that with the absence of detailed F/O specific disruption information, whether the F/O being included in the A/C recovery solution becomes more critical. Connectivity characteristics, reflected by the connection group features, have the second highest importance, reflecting the importance of the connection alternatives for a given F/O.

Four different rulesets are created, ranging from more complex to simplified. The complexity of a ruleset is quantified in terms of the number of rules it contains and the number of features each rule uses. The experiments showed that simpler rulesets provide greater improvement than more complex ones. The tested rulesets are provided in Appendix C.1.

The best-performing ruleset had only two simple rules (Table 4.4). The first rule

Information group	Importance
Schedule	14.19%.
Disruption	13.81%
Connections	28.53%
A/C solution	43.47%

Table 4.3: Variable importance distribution

Rule id	Quarter	Description
1	Any	Prefer an F/O, if it is in the A/C recovery solution and the crew slack time is less than 30 minutes.
2	Any	Avoid an F/O, if it is not in the A/C recovery solution and the NP delay of the first flight is greater than 90 minutes.

Table 4.4: Simplified ruleset for crew recovery

says that if the F/O is included in the A/C recovery solution, prefer the F/O in the crew recovery solution if the slack time between the two flights is less than 30 minutes. The second rule says that if the F/O is not included in the A/C recovery solution, avoid using that F/O in the crew recovery solution if the NP delay of the first flight is greater than 90 minutes. Here, the slack time is found to be relatively less important than in the case of Rule 1. One possible explanation is that Rule 2 detects F/Os to avoid, hence focusing on disruptions, while Rule 1 detects F/Os to prefer, hence focusing on efficiency.

The first rule of this ruleset can be interpreted as a suggestion to prefer the efficient F/Os (i.e., the F/Os with short slack times of less than 30 minutes) that are also included in the A/C recovery solution. In cases where the F/O is not in the A/C recovery solution, the ruleset does not automatically suggest avoiding the corresponding F/O. According to the second rule of this ruleset, such connections can still be included in the solution unless the NP delay of the first flight exceeds 90 minutes.

4.2.3 Rule-based Modifications

The benchmark heuristic (*SWAP matheuristic*) defined in Section 4.2.1 selects candidate crew members for each broken crew duty and includes them in the solution space. To demonstrate the improvements obtained by the ML-insight-based heuristic rules, we incorporated the simplified ruleset presented in Table 4.4 into the step where candidates are selected. The rest of the *SWAP matheuristic* is kept the same.

The original *SWAP matheuristic* looks for candidates among the feasible planned crew duties based on the number of swap opportunities with the broken crew duties and adds them to the solution space. It breaks any ties arbitrarily. Instead, the ML-insight-based rules break ties based on the extent to which the feasible planned crew duties involved in the ties satisfy the rules. With these ML enhancements, the heuristic follows a two-level sorting procedure. First, the feasible planned crew duties are sorted in decreasing order of the number of swap opportunities, and all ties are broken based on a measure reflecting the level of conformity to the ML-insight-based rules. The level of conformity to the ML-insight-based rules for a crew duty is defined as the number of F/Os in the crew duty that satisfy these rules. With this sorting, if two candidate duties have the same number of swap opportunities, the one that has a lower level of conformity is selected to be added to the solution space. This leads to keeping intact the feasible planned duties that have a higher level of conformity with the rules. The rationale is that if a crew duty satisfies the rules, then it is a preferred duty; hence, it should be kept as planned.

Figure 4-5 demonstrates how rule-based modifications affect the candidate selection step in the SWAP matheuristic. Each row represents the sorting of the feasible planned crew duties to fix for the default and modified SWAP heuristics. The blue and gray-shaded cells correspond to the fixed duties and the duties included in the solution space by the default SWAP matheuristic, respectively. The red dashed line reflects the predetermined number of planned duties to fix. For both methods, all cells to the left of the dashed line correspond to the fixed crew duties and the ones on the right correspond to the feasible planned crew duties that are included in the

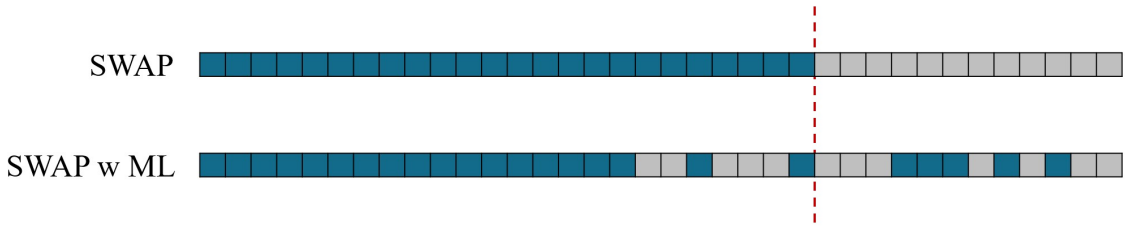


Figure 4-5: Rule-based modifications affect the candidate selection step

solution space of the optimization model. For the modified SWAP method (denoted as *SWAP w ML* in the figure), some of the feasible planned duties selected to be fixed by the default SWAP matheuristic are no longer fixed. The duties with high ranks (those to the left of the figure) are fixed by both methods since the ML-insights-based rules are used purely as tie-breakers, and such second-order sorting does not affect the overall ranking of such high-rank duties. However, the moderate-ranked to low-ranked duties may get sorted differently by the modified SWAP method, as shown in the figure.

4.2.4 Results and Sensitivity Analysis

The set of disruption scenarios and the underlying network are the same as those used in Chapter 2. All experiments are carried out on a desktop computer equipped with an Intel i9-13000K CPU and 64 GB of memory using Gurobi 10.0 (Gurobi Optimization Inc., 2020) as the integer and mixed-integer optimization solver.

Figure 4-6 demonstrates the solution quality improvements achieved by enhancing the heuristic using ML-based insights. The y-axis is the recovery cost difference with respect to the cost of the baseline solution. Baseline solutions are obtained by directly sending the CRM-APC optimization model to a Gurobi solver until a target optimality gap of 0.1% is achieved or the solution time exceeds 2 hours. Lower values of the recovery cost difference correspond to higher solution quality.

To evaluate the sensitivity of performance improvements to small variation in the ruleset, we performed a sensitivity analysis by changing the values of the two main parameters in the ruleset: the slack time threshold in Rule-1 and the NP delay

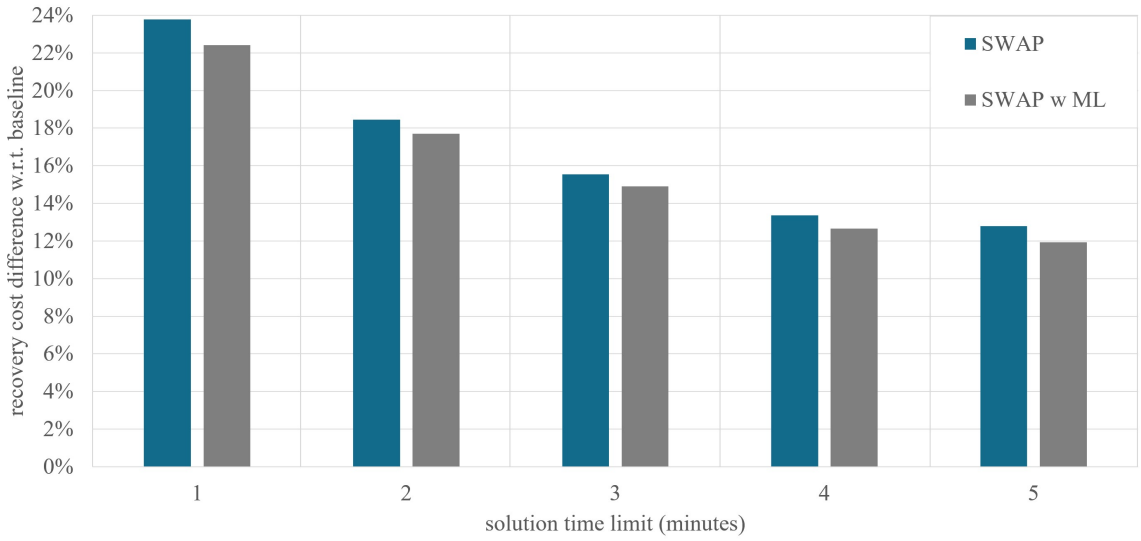


Figure 4-6: Results of the rule-based improvements to the default SWAP matheuristic for crew recovery

threshold for the first flight in Rule-2. Table 4.5 shows the parameter values used in the different versions of the tested ruleset. The ruleset version-0 corresponds to the original ruleset given in Table 4.4.

The results of the sensitivity analysis provided in Figure 4-7 demonstrate a consistent performance improvement of around 7% to 9% over the default SWAP matheuristic, due to the incorporation of these ML-discovered rules of thumb. The y-axis shows the solution quality gain achieved with the ML-insight-based rules ($GAIN$) over the default SWAP matheuristic. It is calculated considering the cost avoided compared to the best solution quality that was achieved in the computational study presented in Chapter 2. It was generated by our proposed solution method combining optimization and ML. The equation used for the $GAIN$ calculation is provided in 4.2 where $Cost^{SWAP}_{default}$ and $Cost^{SWAP}_{improved}$ are the recovery costs achieved by the default and improved versions of the SWAP matheuristic, while $Cost^{ML}$ is the recovery cost achieved by our ML-based method for crew recovery presented in Chapter 2. Higher value of solution quality gain corresponds to a greater improvement in the solution quality.

$$GAIN = 1 - \frac{Cost^{SWAP}_{improved} - Cost^{ML}}{Cost^{SWAP}_{default} - Cost^{ML}} \tag{4.2}$$

Ruleset versions with the slack threshold set to 20 minutes or the first flight's NP delay threshold set to 80 minutes seem to perform worse than others. The former parameter is related to the first rule, and the latter to the second. The results imply that, for an F/O being included in the A/C recovery solution, having a slack time of 20 minutes or less does not provide the same level of confidence in the suggestion, compared to having a slack time of 30 minutes or less. Similarly, for an F/O not being included in the A/C recovery solution, the greater the value of the NP delay of the first flight, the stronger the suggestion to avoid the F/O for the crew recovery solution.

Despite these differences, all tested versions of the ruleset were found to have a performance improvement in the range from 6% to 9%. These results demonstrate that the key insights obtained from our ML analysis relate to the general structure of the rule, rather than the exact threshold values, and this general structure drives the performance improvements rather than the specific values determined. This finding can have significant implications for the generalizability of this modified SWAP matheuristic approach. Given the low dependence of these rules on the exact thresholds, the rule structures might be portable relatively easily across schedules and networks and can still lead to the bulk of the performance improvements even if the thresholds are not tuned for those particular schedules and networks.

Version	Slack time (Rule-1)	NP delay of the first flight (Rule-2)
0	30	90
1	30	80
2	30	100
3	20	90
4	40	90
5	20	80
6	40	100
7	40	80
8	20	100

Table 4.5: Simplified ruleset versions for crew recovery

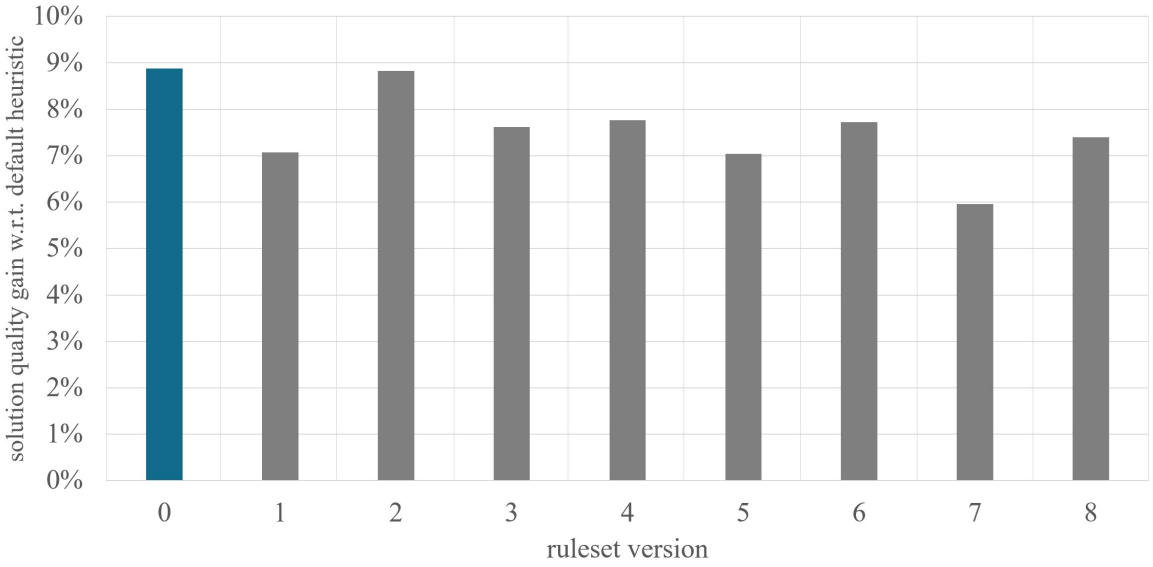


Figure 4-7: Sensitivity analysis of the rule-based improvements to the default SWAP matheuristic for crew recovery

4.3 Integrated Aircraft, Crew, and Passenger Recovery

4.3.1 SWAP Matheuristic

Airline recovery problems are rarely solved in a fully integrated fashion. For most major airlines, it is not practical to do so. Instead, airlines usually solve recovery problems sequentially with some feedback mechanisms. In such an approach, low-quality solutions generated in earlier steps can prevent good solutions in the following steps, resulting in high-cost overall recovery solutions.

Chapter 3 presented a multi-step solution approach in which an integrated recovery model, called IRM, is solved first. The aircraft, crew, and passenger solutions are repaired and finalized subsequently as part of the post-processing steps. Although the IRM model also includes crew and passenger considerations, its primary focus is schedule and aircraft recovery. The main recovery actions included in the IRM model correspond to deciding the flight departure times, arrival times, and cancellations. It also ensures that we can find a feasible aircraft flow for the flight schedules and cancellations decided by the IRM. The benchmark solution methods included in the computational study (Section 3.4.3) were also focused on solving the main IRM model. The solution methods used in the post-processing steps were kept identical for all benchmarks. The computational study in this section focuses on solving the main IRM model and evaluates the performance of the improvements to the benchmark heuristics accordingly.

The benchmark heuristic we focus on in this section also follows an approach similar to the one used for crew recovery in the previous section. It reduces the solution space by adding a few feasible planned aircraft rotations to the solution space in addition to the disrupted aircraft rotations. These feasible planned aircraft rotations are selected based on swap opportunities between them and the disrupted aircraft rotations. Swap opportunities between aircraft rotations are determined in the same way as the swap opportunities between crew duties in the SWAP matheuristic for crew

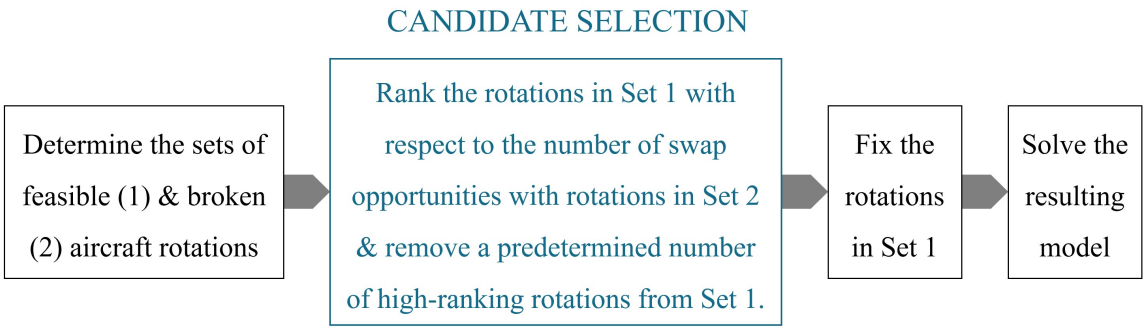


Figure 4-8: High-level overview of the *SWAP matheuristic* for integrated recovery

recovery, as shown in Figure 4-2. The result is a fix-and-dive type of matheuristic similar to that discussed in Section 4.1.2. For consistency, we also call this heuristic *"SWAP matheuristic"* for integrated recovery.

The computational study presented in Chapter 3 shows that this method generates ∼19% higher costs than the baseline solutions in 8 minutes (5 minutes for the main model and 3 minutes for the post-processing steps). In comparison, our ML-based methods generated solutions with only ∼5% higher costs within the same time limits (viz., 5 minutes for the main model and 3 minutes for the post-processing steps). In this section, we improve the solution quality of the *"SWAP matheuristic"* for integrated recovery by incorporating rules of thumb discovered with the help of tree-based classifiers.

A high-level overview of the heuristic is provided in Figure 4-3. The heuristic separates the planned aircraft rotations into two sets: the set (1) of feasible rotations and the set (2) of broken rotations. Then, all the rotations in Set 1 are ranked in decreasing order of the number of swap opportunities with the rotations in Set 2. Finally, a predetermined number of rotations with the highest ranks are included in the solution space, and the rest of the rotations in Set 1 are fixed.

The integrated recovery model used in these experiments is the one presented in Chapter 3 (3.1)-(3.14). Fixing the rotations corresponds to adding Constraints (4.3) to the model before sending it to the mixed integer optimization solver. $T_i^{r_{not\_allowed}}$ refers to the set of time periods for which the flight i is not allowed to arrive. These

sets are determined for each flight included in the fixed rotations (Set 1), based on the departure and arrival times of the flights in those rotations. Algorithm 4 summarizes the overall flow of the solution method.

$$r_i^{a,t} - r_i^{a,t-1} = 0 \qquad \forall i \in I, \forall a \in A, \forall t \in T_i^{r_{not\_allowed}} \tag{4.3}$$

Algorithm 4 SWAP Matheuristic for Integrated Recovery

Initialize:
Load the disruption
Load the IRM
Set the *network-wide maximum delay*
Determine the set of planned rotations to fix (Set 1)
Determine the set of flights covered by the rotations in Set 1 (I_{fixed})
for $i \in I_{fixed}$ **do**
    Determine the set $T_i^{r_{not\_allowed}}$
    for $a \in A, t \in T_i^{r_{not\_allowed}}$ **do**
        add the constraint $r_i^{a,t} - r_i^{a,t-1} = 0$ to IRM
    end for
end for
Solve the IRM with the added constraints

4.3.2 Classifier Insights

In Chapter 3, we trained classifiers to predict the maximum allowed delay limits for flights. Even though our training approach enables the classifiers to handle large collections of flights, they are still primarily tailored to the underlying network. This is because the feature set includes an extensive list of information for each flight and the classifiers focus on arrivals at or departures from a single airport.

In this chapter, our objective is to discover generalizable rules of thumb for integrated recovery problems. Therefore, we trained a new type of classifier that covers

all departures and arrivals instead of the airport-specific classifiers trained in Chapter 3. This is a binary classifier trained to separate flights into two classes based on the expected delays and the selected delay separator limit. The classifier predicts whether or not the flight delay exceeds the specified delay limit.

The data used for classifier training includes ∼100,000 data points corresponding to individual flight delays in solutions to 48 disruption scenarios. The *Optimal Classification Trees (OCT)* method is selected because it can provide interpretable results. The maximum tree depth parameter is limited to 5 to keep the resulting model structures interpretable. The trained classifiers are then reviewed to determine common patterns that would help derive generalizable rules of thumb.

Overfitting refers to the behavior of ML models that provide accurate predictions for training data, but fail to do so for other data inputs (Mitchell, 1997). If overfitting occurs during the training process, the resulting model cannot generate generalizable insights. A concise feature set is defined to avoid overfitting to the training input.

The feature set has 13 features, including information related to schedule and airport capacities, and the number of passengers on that flight. The latter was added to the feature set because it is the most critical passenger-related information that airlines consider when making delay decisions. Schedule information was included to provide a high-level description of different flights. It contains the OD frequency, departure time, and duration. Airport capacity information was added to the feature set to provide a snapshot of the disruption conditions surrounding the departure and arrival times of the flight. Provided information corresponds to hourly capacities covering the planned departure (respectively arrival) time spanning from the previous hour (hour -1) to two hours ahead (hour +2) including the capacities for the planned departure (respectively arrival) time (hour 0). More detailed descriptions of the attributes included in the feature set are provided in Table 4.6. The training input also includes a binary label for each data point or observation, reflecting the flight delay in the disruption solution. The label is 1 if the delay does not exceed the delay separator limit and 0 otherwise.

Binary classifiers are trained with different values for the maximum tree length

Feature id	Information Group	Description
1	Schedule	OD frequency, i.e., the number of daily flights in this airport OD pair.
2	Schedule	OD frequency order, i.e., starting from the beginning of that day how many flights (including the current one) have been flown until the current flight in this airport OD pair.
3	Schedule	Departure time in UTC time increments.
4	Schedule	Flight duration in time increments.
5	Passengers	Number of passengers on this flight.
6	Airport capacity	Departure capacity during hour -1 at origin airport.
7	Airport capacity	Departure capacity during hour 0 at origin airport.
8	Airport capacity	Departure capacity during hour 1 at origin airport.
9	Airport capacity	Departure capacity during hour 2 at origin airport.
10	Airport capacity	Arrival capacity during hour -1 at destination airport.
11	Airport capacity	Arrival capacity during hour 0 at destination airport.
12	Airport capacity	Arrival capacity during hour 1 at destination airport.
13	Airport capacity	Arrival capacity during hour 2 at destination airport.

Table 4.6: Feature set attributes for integrated recovery

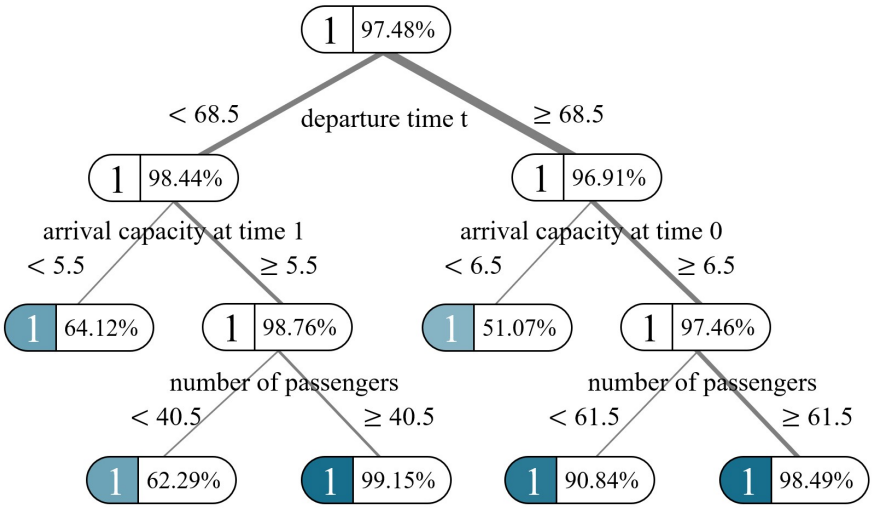


Figure 4-9: Binary tree classifier example with one-hour delay separator limit and maximum tree length of 3

and delay separator limit parameters. The set of values used for the former parameter include 3,4 and 5, and for the latter parameter, 30 minutes, 45 minutes, and 1 hour. An example classifier is given in Figure 4-9 with a one-hour delay separator limit and a maximum tree length of 3. Label 1 corresponds to limiting the delay to 1 hour. The predicted probability values that reflect the strength of the prediction are given in the nodes. Although the labels for all leaf nodes are 1, the corresponding prediction probabilities are different. The main takeaway of this classifier is that the number of passengers that would justify limiting the flight delay to 1 hour depends on the time of the flight. Flights in the roughly first half of the day have a lower threshold value (40.5 passengers) than flights in the second half (61.5 passengers).

We conducted a variable importance analysis to understand the relative importance of these information groups. The results are provided in Table 4.7. The importance distribution is consistent with the one provided in Section 3.5.2 for classifiers focused on individual airports.

The resulting classifiers contained many alternative rules of thumb. We generated different rulesets, ranging from complex to simpler, where the complexity is quantified in terms of the number of rules and attributes involved. As in the case of crew

Information group	Importance
Schedule	11.66%.
Passengers	16.16%
Airport Capacity	72.18%

Table 4.7: Variable importance distribution for aggregate classifiers for integrated recovery

Rule id	Delay limit (minutes)	Description
1	60	Limit the flight delay if the departure time ≥ 70 and the number of passengers > 75.
2	60	Limit the flight delay if the departure time < 70 and the number of passengers > 55.

Table 4.8: Simplified ruleset for integrated recovery

recovery, simpler rulesets usually performed better. The tested rulesets are provided in Appendix C.3.

The best performing ruleset has only two simple rules (Table 4.8) based on departure time and number of passengers. The delay separator limit is set to 1 hour. The first rule says that the delay of a flight departing during roughly the first half of the day should not exceed 1 hour unless the number of passengers is less than or equal to 55. For the flights departing in the second half of the day, delay should be limited to 1 hour for flights with more than 75 passengers. A classifier example with two features is given in Appendix C.4.

4.3.3 Rule-based Modifications

The benchmark *SWAP matheuristic* defined in Section 4.3.1 selects planned feasible aircraft rotations with most swap opportunities with broken aircraft rotations and includes them in the solution space of the optimization model. To demonstrate the improvements from using ML-insight-based heuristic rules, we incorporated the

rulesets presented above into the candidate selection step. The rest of the SWAP matheuristic was kept the same.

The heuristic modifications we propose are similar to those discussed in Section 4.2.3. The original candidate selection step shown in Figure 4-8 involves ranking the feasible planned aircraft rotations in Set 1 based on the swap opportunities with the rotations in Set 2, with ties broken arbitrarily. Instead, we introduced another ranking level based on the determined rules, in order to break the ties, to make the algorithm lean towards fixing the planned rotations that satisfy the rules. If a feasible planned aircraft rotation satisfies the rules, then it is a preferred rotation; hence, it should be kept intact as planned.

4.3.4 Results and Sensitivity Analysis

The set of disruption scenarios and the underlying network are the same as those used in Chapter 3. All experiments are carried out on a desktop computer equipped with an Intel i9-13000K CPU and 64 GB of memory using Gurobi 10.0 (Gurobi Optimization Inc., 2020) as the integer and mixed-integer optimization solver.

Figure 4-10 demonstrates the solution quality improvements achieved by enhancing the SWAP matheuristic using ML-based insights. The y-axis is the recovery cost difference with respect to the baseline solution cost, and the x-axis corresponds to the solution time limit imposed for solving the main IRM model. Baseline corresponds to sending the full IRM model (without any solution space reduction) directly to a Gurobi optimizer and allowing it to run until the optimality gap is reduced to 0.1% or lower or the run time exceeds 2 hours. Ultimately, the recovery cost is calculated by performing all post-processing steps required to generate full solutions to the aircraft, crew and passenger recovery problems. The post-processing steps take around 3 minutes regardless of the solution method used for solving the IRM model. Lower values of recovery cost differences correspond to higher solution quality.

To evaluate the sensitivity of performance improvements to different versions of the simple ruleset, we performed a sensitivity analysis by changing the values of the three main threshold parameters in the ruleset: departure time separator used in both

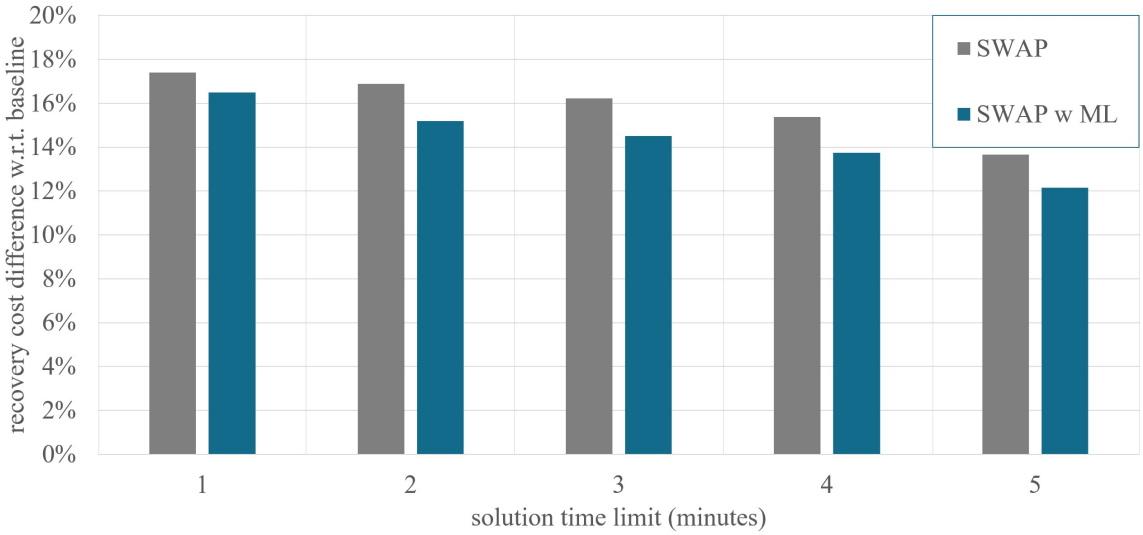


Figure 4-10: Results of the rule-based improvements to the default SWAP matheuristic for integrated recovery

rules, the threshold number of passengers used in Rule-1, and the threshold number of passengers used in Rule-2. Table 4.9 shows the details of the different tested versions of the ruleset. The ruleset version-0 corresponds to the original ruleset given above in Table 4.8.

The results provided in Figure 4-11 demonstrate a consistent performance improvement over the default SWAP matheuristic, due to the incorporation of the ML-discovered rules of thumb. The solution time limit for solving the IRM model is set to 5 minutes. The y-axis is the solution quality gain achieved with the ML-insight-based rules ($GAIN$) with respect to the default SWAP matheuristic. It is calculated considering the cost avoided compared to the best solution quality that was achieved in the computational study presented in Chapter 3. It was generated by our proposed solution method combining optimization and ML. The equation used for the $GAIN$ calculation is same as provided in 4.2 for crew recovery:

$$GAIN = 1 - \frac{Cost^{SWAP}_{improved} - Cost^{ML}}{Cost^{SWAP}_{default} - Cost^{ML}}$$

In this section, $Cost^{ML}$ refers to the recovery cost achieved by our ML-based method for integrated recovery presented in Chapter 3. Higher value of solution

Version	Departure time separator (in time increments starting from midnight UTC)	Rule-1 # pax	Rule-2 # pax
0	70	75	55
1	60	75	55
2	60	90	75
3	60	100	50
4	80	75	55
5	80	90	75
6	80	100	50
7	90	75	55
8	90	90	75
9	90	100	50

Table 4.9: Simplified ruleset versions for integrated recovery

quality gain corresponds to a greater solution quality improvement.

All tested versions of ruleset were found to have a performance improvement in the range from 12% to 17%. Once again, as in the crew recovery context discussed earlier in this chapter, these results demonstrate that the key insights obtained from our ML analysis relate to the general structure of the rule, rather than the exact threshold values, and this general structure drives the performance improvements rather than the specific values determined. As mentioned in the crew recovery case, the low dependence of these rules on the exact thresholds imply that the rule structures might be portable relatively easily across schedules and networks, and can still provide much of the performance improvements even if the thresholds are not tuned for the particular schedules and networks.

4.4 Conclusion

In this chapter, we investigate how existing heuristic-based solution approaches can be improved without significant implementation efforts. We showed that simple rules

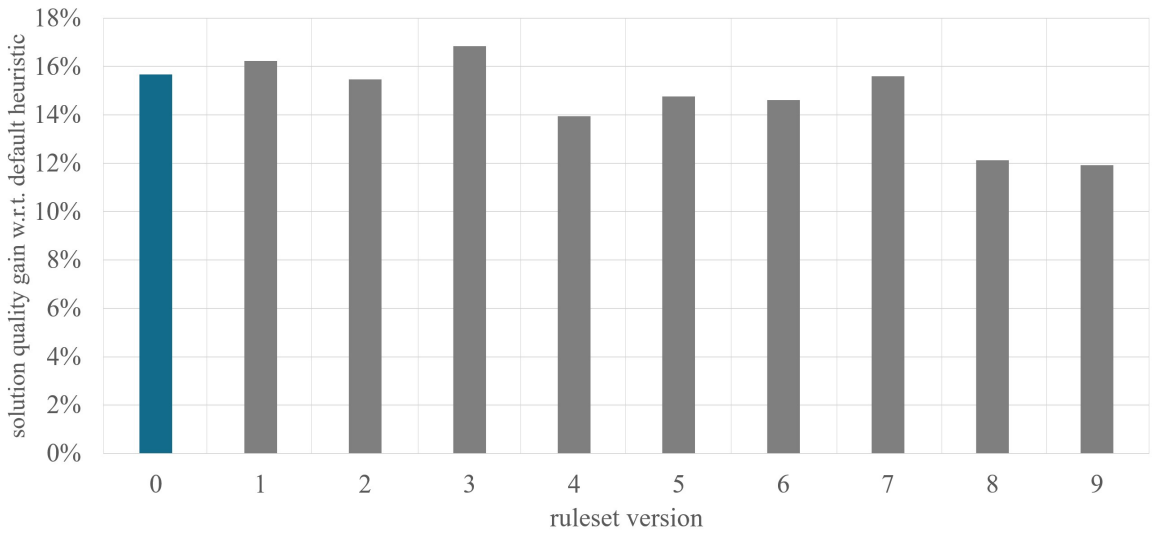


Figure 4-11: Sensitivity analysis of the rule-based improvements to the default SWAP matheuristic for integrated recovery

derived from the interpretable structure of the tree classifiers can be directly incorporated into the commonly used airline recovery heuristics to improve solution quality.

In crew recovery experiments, two simple rules concerning F/Os were determined. Both rules focus on whether or not the F/O is included in the A/C recovery solution, with additional focus on slack times and the non-propagated delay to the first flight. All of these criteria make intuitive sense, which is important to build trust with the recovery staff. The results show that by incorporating these simple rules into the SWAP matheuristic, we can improve the solution quality performance by ∼9%. Sensitivity analysis showed that the performance improvements are stable regardless of the exact values of the threshold parameters used by the rules, thus enhancing their generalization potential. These experiments were conducted assuming that the non-propagated delay predictions are accurate. Further experimentation is needed to evaluate the robustness of the results to inaccurate non-propagated delay predictions.

In the integrated recovery experiments, the selected ruleset is based on the simple idea that a flight should not be delayed beyond a specific limit if the number of passengers is higher than a threshold value, which is discovered to be different for the flights in the first and second half of the day. Experiments showed that an improvement of ∼15% can be achieved by integrating these rules into the *SWAP matheuristic*

solution method. These rules also proved to be quite robust, such that changing the values of the threshold parameters did not significantly affect the performance.

Although this chapter focuses on one type of recovery heuristic, *SWAP matheuristic*, a similar approach can be followed to improve other types of heuristics, including meta-heuristic or domain-specific heuristic solution methods. For a greedy heuristic approach, we can redefine what makes a solution alternative appealing to the algorithm at each stage. For a genetic algorithm, we can update the rules followed in the crossover and mutation steps, which are the mechanisms to iteratively generate better solutions. For local neighborhood search heuristics, we can enhance the rules that define a solution's neighborhood based on ML-derived insights. Similarly, for other meta-heuristics and domain-specific heuristic methods, we can analyze the structure of the algorithms and improve their performances using the insights from the classifiers.

Chapter 5

Conclusion and Future Research

5.1 Concluding Remarks

In this research, we introduce a general framework that helps develop fast and tunable airline recovery methods by combining optimization and ML tools. This framework can allow airlines to meet the major practical requirements of a recovery system: quality, speed, flexibility, and interpretability. The main idea behind the framework is that, under similar disruption conditions, recovery decisions that lead to high-quality solutions can also be similar. It relies on an offline phase, where solutions to historical disruption scenarios are generated, and ML methods are used to discover patterns in recovery decisions. Then, on the day of operations, the trained classifiers are used to determine a restricted solution space for a given disruption, within which the optimization model searches for a high-quality solution. An important feature of the framework is that it leverages interpretable ML methods so that the rationale behind the suggested recovery decisions is clearer to human decision-makers, thus building trust in the automated decision-support tools.

In Chapter 2, we first formulate the crew recovery problem as a string-based set-covering model that assumes that an aircraft recovery solution is available from a previous step in the recovery process. The crew solution maintains the cancellation decisions and the aircraft rotations decided in the previous step, and determines crew duties and potentially altered flight schedules that minimize crew costs plus

approximate passenger disruption costs associated with the crew recovery decisions. We define a disruption feature set that allows us to characterize different kinds of disruptions. We create a recovery solutions database using hundreds of disruption scenarios generated based on historical flight delay and cancellations data. We train classifiers to discover patterns in the solutions. Classifiers help predict whether a specific flight-connection pair (follow-on) should be used in the crew recovery solution for a given disruption. We use the trained classifiers in our experiments to find high-quality solutions in limited timeframes, for recovery problems involving a ∼3000-flight network. In less than 5 minutes, our proposed methods can generate solutions that are within 3% of the baseline solutions. The baseline solutions, which have a ∼1% optimality gap on average relative to the true optimal solution value of the crew recovery model, are obtained by direct optimization and require ∼2 hour run time.

In Chapter 3, we develop a tractable integrated model that recovers flight schedules while considering the most important aspects of the airline recovery process, namely ensuring aircraft flow and reducing crew infeasibilities and passenger disruptions. The solution approach includes three post-processing steps to finalize the recovery solutions for aircraft, crew, and passengers, respectively. The computational case study focuses on major disruptions caused by airport closures and airport capacity reductions. In the offline phase, solutions to historical disruption scenarios are generated, and ML classification models are trained to discover patterns in the flight delays associated with the obtained solutions. Classifiers help predict how much each of the flights in the network should be allowed to be delayed for a given disruption. We use trained classifiers in our computational experimentation to find high-quality solutions in limited timeframes, for recovery problems involving a ∼3,700-flight network. In less than eight minutes, our proposed methods can generate solutions that are within 5% of the baseline solutions. The baseline solutions, which have a ∼2% optimality gap on average relative to the true optimal solution value of the integrated recovery model, are obtained by direct optimization and require ∼2 hour run time.

In Chapter 4, we demonstrate that reaping partial benefits of the presented framework does not require significant changes in the existing recovery processes of airlines.

The interpretable structure of the trained classifiers helps to discover simple rules of thumb that can be used to improve existing solution approaches. We train generalizable classifiers, at an aggregate level, for crew and integrated recovery problems to discover simple rules that can be incorporated into existing solution approaches. One of the most commonly used solution space reduction methods are the matheuristics, which include for broken crew duties or aircraft rotations, a number of candidate non-broken duties or rotations, that can be used in the solution to repair broken duties and rotations. We run a mixed-integer optimization model over this reduced solution space and show that the candidate selection step of these matheuristics can easily be improved using the simple rules of thumb discovered with the help of our trained classifiers. Improvements of the order of $\sim$9% and $\sim$15% were obtained for the crew and integrated recovery experiments, respectively. Moreover, these improvements were found to be stable and robust to changes in the parameter values of the ML-derived rules, indicating the potential to apply them effectively across different schedules and airline networks.

5.2 Future Research Directions

In Chapter 2, follow-on pairs are selected as the micro-solutions to predict. This requires hundreds of disruption scenarios to be solved in the offline phase. Our experiments showed that such classifiers may not generalize very well to other flight schedules. Therefore, we recommend that the process be repeated before each day to achieve maximum benefits. More research is needed to develop enhanced procedures that would allow training generalizable classifiers, hence reducing the time and computational resources airlines would need to implement such a system.

In Chapter 3, the developed solution method uses classifiers trained to predict whether the delay of a flight should be allowed to exceed a specified limit. The underlying mathematical optimization model includes many other recovery decisions, such as canceling a flight, breaking a crew duty, or breaking a passenger itinerary. Additional classifiers can potentially be trained to predict these multiple decisions.

In Chapter 4, we used simple rules of thumb derived from trained classifiers to improve the performance of a commonly used matheuristic that selects candidate resources to be included in the solution space to accelerate the solution process. There are many different heuristic approaches in the literature that researchers have developed based on intuitive rules and expert judgments. Further research would be helpful in determining whether ML-derived rules can consistently improve these other heuristic solution approaches as well.

The presented framework was created to meet the major requirements for a practical and efficient airline recovery system. The primary focus was *speed* as airlines prefer solutions be found in less than 5 minutes. The algorithmic structure called *learning to configure algorithms* was followed since it was the most suitable structure in the recovery context. However, a similar framework can also be developed for airline scheduling problems where solution time limitations are not as strict. In the scheduling phase, the quality of the solutions is more important than how quickly they can be generated. In other words, while accelerating the solution process would make the scheduling process more efficient, solution quality remains the priority. In this context, another algorithmic structure, called *ML alongside optimization*, can be a better approach to incorporate ML into the solution process. Predicting branching decisions in a *Branch-and-Bound* based solution process or predicting columns to be added in a column generation-based solution approach fall into this category. Although other studies have tried these ideas for general integer optimization models, there are many research opportunities to use these ideas within the airline scheduling context.

Appendix A

Computational study details for Chapter 2: Crew Recovery with Aircraft and Passenger Considerations

A.1 Disruption Profile Clustering

A comprehensive analysis of historical disruptions is crucial for generating realistic disruption scenarios and ensuring the success of the proposed framework. A widely used method to analyze disruptions is clustering, an unsupervised ML approach that aims to partition a given data set into groups based on similarities (Mitchell, 1997). Some applications of clustering to the national airspace system (NAS) disruptions by other researchers are mentioned in Section 2.1.1. A common approach is to use a list of network characteristics to identify disruption patterns and day types in the NAS. In our study, we cluster historical flight operation days into disruption profiles based on the NP delay characteristics using a commonly used approach, namely, k-means clustering.

Clustering of the disruption profiles is the first step in the scenario generation

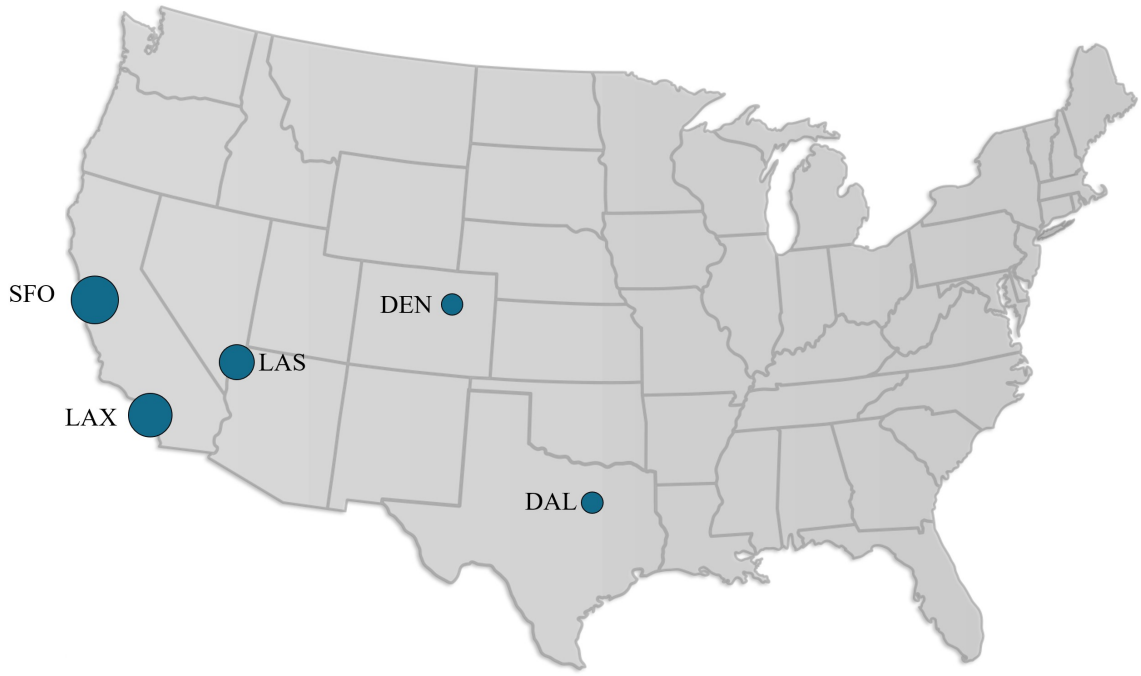


Figure A-1: A disruption profile for the airline used in the computational study

procedure (Section 2.3.2-Figure 2-3). The network used in our experiments is based on flight operation days in the winter season; therefore, we used five years (2013 to 2017) of winter flight operation data (December, January, and February) as input. We first clustered 451 winter days with respect to their disruption profiles. The information used as features in the clustering process includes the average delay of each Origin-Destination (OD) pair that has at least two non-stop flights on average each day. There are 559 such OD pairs in the network. There were ten distinct disruption profiles. In this process, the value of k was set to 16 motivated by analysis for another airline network in which k was set to 16 and the analysis resulted in 10 disruption profile clusters.

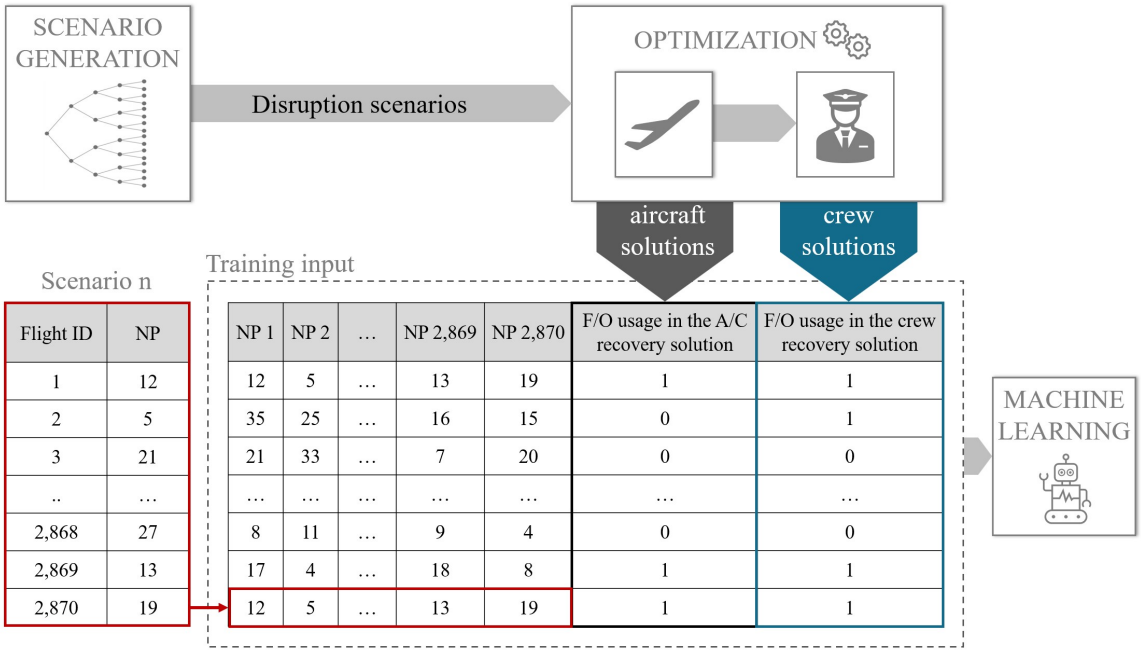
The disruption profile in Figure A-1 corresponds to one of the disruption profile clusters, where the disruptions originate from multiple airports in the south and midwest of the United States. The size of the circles represents the level of disruptions measured in terms of the average delays of flights that touch the corresponding airport. Creating another cluster for days that do not fall into any other clusters, we ended up with ten disruption profile classes.

A.2 ML Training Input Generation

We generate the input for training the classifiers for each micro-solution selected for classifier training using the optimization solutions to the disruption scenarios. Figure A-2 describes how the training input with a full feature set is created for a network with 2,870 flights. Each row in the training input corresponds to a disruption scenario. The first 2,870 columns are the NP delays of the flights in the network representing the disruption characteristics. The next two columns reflect whether or not the corresponding F/O is used in the aircraft and crew recovery solutions respectively. The former is part of the feature set because we assume that the aircraft recovery problem has already been solved. The latter is the output or label that we are trying to predict using the information included in the feature set.

The process is slightly different when the local neighborhood feature sets are used instead of the full feature sets. As described in Section 2.3.4, with local feature sets, we only consider the NP delays of the flights in the neighborhood of the F/O. NP delay information for flights outside the neighborhood is discarded. Therefore, the set of columns corresponding to the disruption characteristics contains different sets of flights for different F/Os. The last two columns are included in all training inputs.

The resulting dataset is provided as input to classifier training. In the ML training step, the training input is divided into training, validation, and test subsets, with a 2/1/1 ratio, consistent with the standard training approach used in ML practice and literature (Mitchell, 1997). Each subset has the same distribution of disruptions from 10 distinct disruption profile clusters. These sets are parts of the input used in classifier training and, hence, are different from the sets used in the computational study for calibration and testing. The InterpretableAI package (InterpretableAI, 2020) is used for the classifier training adopting a single-fold validation approach. It internally tunes two hyper-parameters while training the classifiers: complexity parameter (cp) which has a continuous range from 0 to 1, and tree depth, which has a discrete range from 1 to the maximum tree depth parameter we set during the training. The values used were 5 and 10 to allow training classifiers with different structures.



Flight ID	NP
1	12
2	5
3	21
..	…
2,868	27
2,869	13
2,870	19

NP 1	NP 2	…	NP 2,869	NP 2,870	F/O usage in the A/C recovery solution	F/O usage in the crew recovery solution
12	5	…	13	19	1	1
35	25	…	16	15	0	1
21	33	…	7	20	0	0
…	…	…	…	…	…	…
8	11	…	9	4	0	0
17	4	…	18	8	1	1
12	5	…	13	19	1	1

Figure A-2: Input generation and classification model training

A.3 Micro-solution Prediction

The binary decision variables of the underlying model represent the assignment of a crew duty to a specific crew member. Naturally, these variables can be deemed as micro-solutions since they correspond to a smaller portion of the solution, and the set of variables selected (i.e., those with values equal to 1 in the solution) form the entire solution. Therefore, classifiers can be trained to predict whether a specific assignment would be in the solution to a given disruption.

There are alternative candidates to use as micro-solutions, like crew duties and F/Os. The latter is the smallest micro-solution type suitable for the provided mathematical formulation, in terms of the number of flights represented by the micro-solution types. An F/O has only two flights by definition, while crew duty and assignment micro-solutions may correspond to several flights (up to five flights in our computational experiments) assigned consecutively. Classifiers can be trained to predict whether a specific F/O will be in the solution of a given disruption. If such an F/O is fixed, meaning that the corresponding constraints are added to the model in advance, the optimizer decides the crew duty selection and assignment to crew members.

The number of possible crew duties and the number of possible assignments are much higher than the number of possible F/Os, making F/Os the better micro-solution choice for practical purposes since fewer classifiers would have to be trained. Additionally, it provides more flexibility for the model. This is so because there may be several different crew duties and assignments corresponding to a given F/O. The model can choose those that minimize the objective function.

The network used in this analysis has 1,403 flights. It is different from the network used in the computational study. This is to prevent the developed solution methods from being specifically tailored to a single network. Table A.1 includes average database (DB) frequency and precision for the 710 assignment classifiers and the 350 F/O classifiers trained using a training input of 2,000 scenarios. The results show that the F/O classifiers also provide a greater average precision improvement

over database frequency statistics than the assignment classifiers.

Micro-solution	Avg. DB frequency	Avg. precision	Improvement over DB frequency
Assignment	81.0%	84.9%	3.9%
Follow-on	82.6%	87.8%	5.2%

Table A.1: Micro-solution classifier performance comparison

We also performed experiments to compare the use of different combinations of assignment and F/O classifiers in the developed framework. Using only F/O classifiers performed consistently better than using only assignment classifiers or both F/O and assignment classifiers. When an assignment is suggested to be fixed based on classifier predictions, the value of the corresponding decision variable, y_s^k is fixed to 1. When an F/O is suggested to be fixed based on classifier predictions, the model is enforced to pick at least one assignment y_s^k that contains the F/O. The analysis includes 100 disruption scenarios and each point on the figure represents a set of runs using a $PC\_threshold$ value of 70%, 80%, 90%, or 95%. A summary of the performance of these alternative approaches can be seen in Figure A-3. Being closer to the axes implies better performance, since the goal is to find the best possible solution in the shortest possible time. On the basis of these results, we selected F/Os as the micro-solution to predict and fix in advance for the remaining part of the experiments.

A.4 Multiple Classifiers

As argued in Section 2.3.4, developed solution methods benefit from training multiple classifiers for the same F/O micro-solution. For an F/O representing two flights, i and j, being assigned consecutively in any crew duty string, other upstream and downstream flights might help explain why i - j is a good (or bad) micro-solution for the given disruption. For example, if all upstream flights that can connect to i are expected to be significantly delayed, connecting i and j would cause the delay to propagate to j and other downstream flights. Further delays may be avoidable if

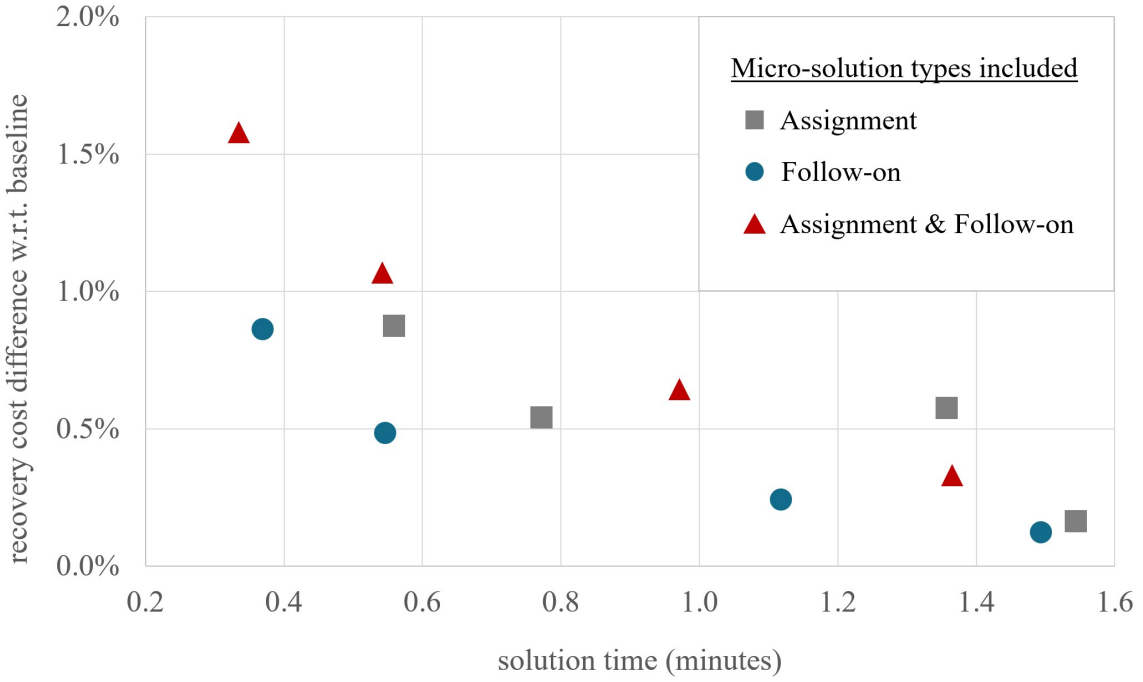


Figure A-3: Micro-solution alternatives and solution approach performance

i and j are not connected. Similarly, if some of the upstream flights j can connect to are expected to not have any non-propagated delays (or expected to have negative non-propagated delays), connecting i and j would be a good choice regardless of the expectation of delays for the upstream flights. One of the classifiers can focus on the upstream flight legs, while the other can focus on the downstream flight legs. An upstream-focused classifier would use the NP delay information for earlier flights, which can connect to i, while the downstream-focused classifier would use the corresponding information for the later flights to which flight j can be connected.

The procedure followed to fix a micro-solution in the case of multiple classifiers is summarized in Figure A-4. Let us assume that we trained N classifiers for each micro-solution that are different from each other in terms of the classification method or maximum tree depth parameter used during training. Before initiating the optimization run to find a solution for the given disruption, the PC values for each micro-solution are calculated using the corresponding classifiers. If any of the PC values is greater than or equal to the threshold value selected for the run, the micro-

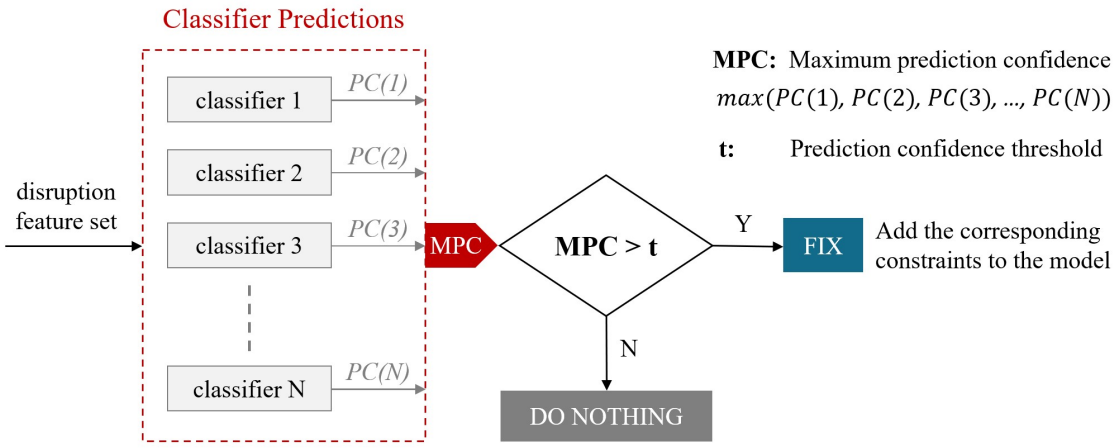


Figure A-4: The procedure to fix a micro-solution.

solution is fixed in advance, meaning that the corresponding constraints are added to the model. Otherwise, it is kept in the solution space, so the optimizer can decide whether to include it in the final solution. The resulting problem with the reduced solution space is sent to the optimizer after the entire subset of micro-solution alternatives with trained classifiers is evaluated.

If we continue with the previous example, in the case that we have two classifiers for an F/O, one of which focuses on upstream flight legs while the other focuses on downstream flight legs, we should consider the predictions from both classifiers to discover the latent factors governing the use of the micro-solution so that we can add the corresponding constraints to the model. In essence, we consider the maximum of the PC values across all classifiers for a given micro-solution to maximize the extent of solution space reduction.

A.5 Pre-processing Details

Several pre-processing steps need to be conducted before the actual experimentation. The first step is to populate the feasible crew duty strings that satisfy the crew legality rules. The set of parameters that we used in this process includes the maximum crew duty duration, the minimum connection time between two flight legs in a crew duty, and the maximum connection time between two flight legs in a crew duty. Since the

model allows delays to propagate, it can assign two consecutive flights to the same crew even if they do not have sufficient connection time between them and hence such duties are also generated.

In this study, the minimum connection time used for the delay propagation calculation is 30 minutes, which is also a generally applied setting in practice for narrow-body aircraft. This duration tends to be longer for wide-body aircraft, but the vast majority of domestic US flights are operated by narrow-body aircraft; hence, the selection of 30 minutes as a single value is justified. It is important to note that when there is an aircraft change in a crew duty, the minimum connection time requirement increases to accommodate the additional time needed for the crew to travel to the second aircraft, which was set to 15 minutes in this study.

To ensure that each flight has a sufficient number of alternative crew duty strings to cover it, we followed an adaptive maximum connection time algorithm when generating the crew duty strings. The string generation procedure determines the maximum connection time for each flight, $max\_cnx\_time(i)$, to ensure a predetermined number of connection alternatives, $target\_num\_cnxs$. The $global\_max\_cnx\_time$ is the highest connection time that is set for any flight. The general flow of the algorithm is given in Algorithm 5.

The third step is to determine the set of assignments for each crew. Several conditions must be met to assign a duty to a specific crew member during the day of operation. The feasibility of a crew duty assignment is determined by the location of the crew at the beginning and end of the day, the earliest possible start time and the latest possible end time for an assigned duty, which are set by the crew duties on the previous and the next day. Figure A-5 is a Gantt chart view that demonstrates a case with a single crew duty and three crew members with the same day start and end locations. The gray bars represent the time duration for which the crew members cannot be assigned to any duty due to the duties of the previous and the next day and the time required to rest between two consecutive duties. It is easy to see that the crew duty can only be assigned to crew members 1 and 2 but not to 3 since the scheduled duty end time violates that crew's latest duty end time.

Algorithm 5 Adaptive Maximum Connection Time

For each flight i
Initialize:
$max\_cnx\_time(i) \leftarrow 30$
$num\_cnx(i) \leftarrow 0$
while $num\_cnx(i) < target\_num\_cnxs$ **and**
    $max\_cnx\_time(i) \leq global\_max\_cnx\_time$ **do**
  for flight $j \in I$ **do**
    if (i, j) or (j, i) is a feasible connection **then**
      $num\_cnx(i) \leftarrow num\_cnx(i) + 1$
    end if
  end for
  $max\_cnx\_time(i) \leftarrow max\_cnx\_time(i) + 15$
end while
return $max\_cnx\_time(i)$

In the computational study, we generate the set of feasible crew duty assignments, which correspond to the decision variables y_s^k in the model, using the monthly crew schedules provided to us by the airline for this research.

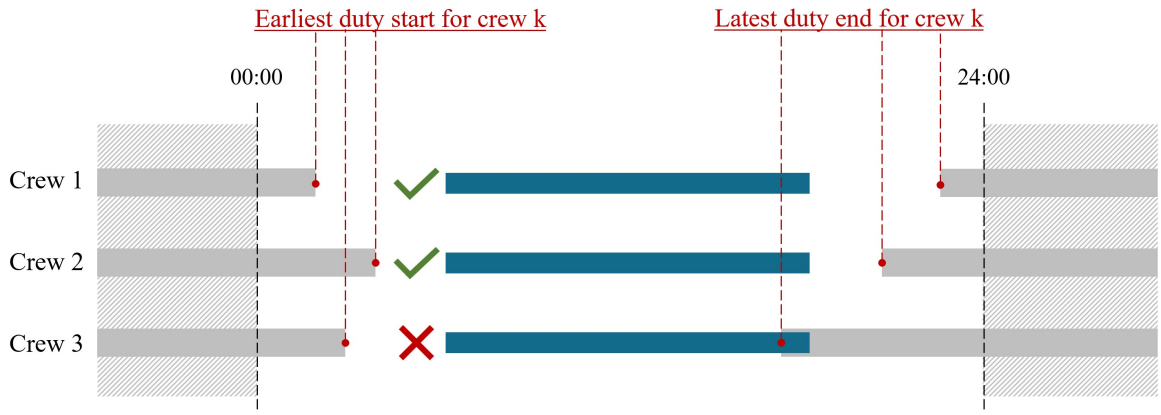


Figure A-5: Assignment generation in pre-processing

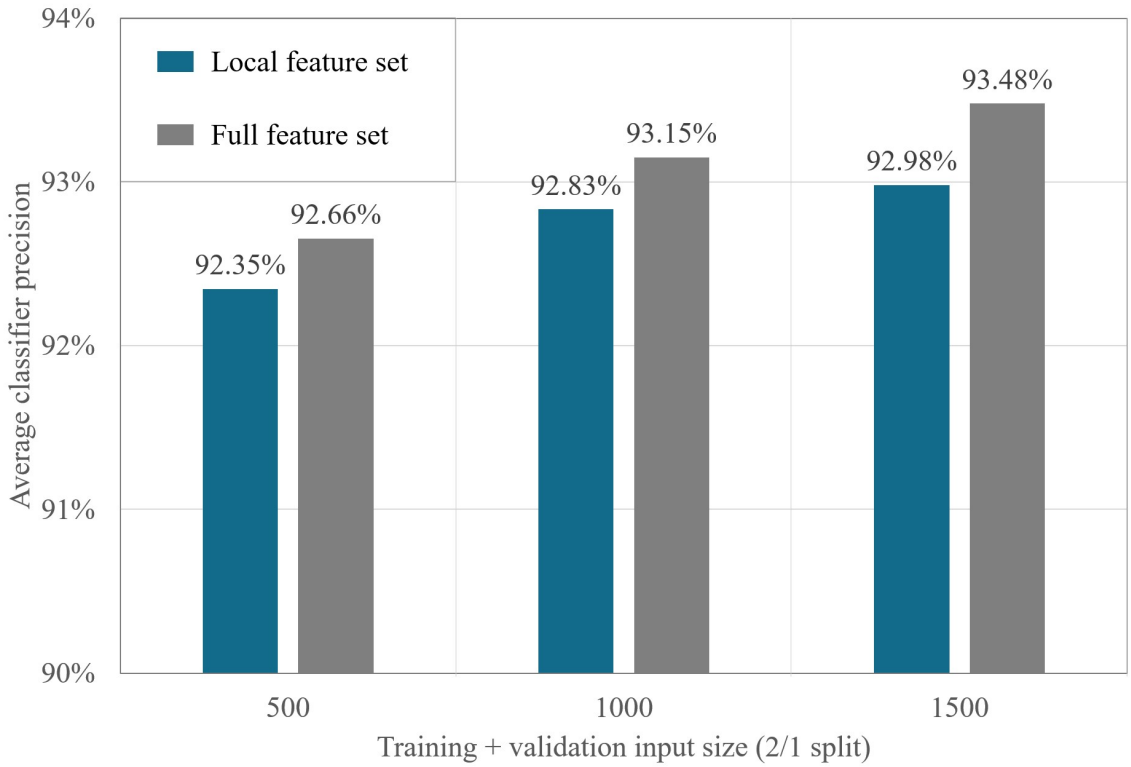


Figure A-6: Classifier performance comparison between local and full feature sets

A.6 Feature Set and Training Input Size Selection

In Section 2.3.4, we proposed a way to define the disruptions for each F/O micro-solution to reduce the amount of information needed for classifier training and accelerate the training process. It relies on focusing on the local neighborhoods of each F/O and using the corresponding disruption information.

Using these local neighborhood feature sets in classifier training can decrease classifier precision performances, because some of the delay information from flights not in the local neighborhood can also potentially affect the selection of an F/O. However, our experiments showed that this loss in precision performance is not significant. In most cases, local feature sets seem to cover most of the vital information.

This analysis was performed for a network with 1,403 flights to analyze the effects of different feature set and training input sizes on the overall classifier precision. This network is different from the one used in the computational study, which helps us to build methods suitable for different networks rather than tailoring an algorithm for

a given network. Figure A-6 compares the average precision of 327 classifiers, with maximum tree lengths of 3, 6, and 9, trained for 109 F/Os using local feature sets and full feature sets with different input sizes. The numbers on the x-axis correspond to the total number of scenarios used in classifier training (excluding the test set), which is divided into training and validation sets in a 2 to 1 ratio. To ensure a fair comparison, the number of test scenarios was set to 500 for all cases. The results show that the loss in average precision performance is at most 0.5% when the local feature set is used instead of the full feature set. Furthermore, classifier training with the local feature set is more than five times faster, making it the more practical feature set.

Increasing the input size has a positive impact on the average precision. But the marginal improvement seems to be decreasing. The results of the analysis show that increasing the training input size has marginal impact on the precision performance especially when local feature sets are used. Motivated by this observation, we decided to use the local feature set approach and generated 500 scenarios for the solutions database.

A.7 Optimality Gap Target for Solutions Database

As discussed in Section 2.3.3, the value selected for the optimality gap target for generating the solutions database can have a significant impact on the computational requirements for the offline phase. During the algorithm development phase, we performed an analysis to understand the effects of different optimality gap targets on the effectiveness of the trained set of classifiers. Table A.2 includes the results of a comparison of scenario run times and classifier precision statistics between setting the optimality gap to 0.1%, 10%, 20% and 30% for a network with 1,403 flights.

The feature set used during classifier training is the full feature set for the 0.1% database, and the local neighborhood feature set for the other databases. The full feature set includes NP delay information for all flights in the network, while the local neighborhood feature set only includes NP delay information for flights in the local neighborhood, as described in Section 2.3.4. The optimality gap target was set to 20% in the computational study, because it provides a ten-fold solution acceleration compared to the 0.1% setting and generates 311 high-precision classifiers ($\geq$ 70%) compared to 353 for the 0.1% solutions database, corresponding to only $\sim$ 15.5% fewer high precision classifiers.

Optimality Gap Target	0.1%	10%	20%	30%
Feature Set	Full	Local	Local	Local
Run time (minutes)	14.6	2.85	1.41	1.16
number of F/Os with precision $\geq$ 50% (out of $\sim$ 800)	502	463	448	433
number of F/Os with precision $\geq$ 70% (out of $\sim$ 800)	353	319	311	308
Overlap with $\geq$ 70% precision classifiers in database with 0.1% optimality gap solutions	-	85.21%	84.46%	81.70%

Table A.2: Classifier performance comparison

Motivated by the observations above, we decided to use the optimality gap target to 20%, for the solution database for the network with 2,870 flights used in the computational study.

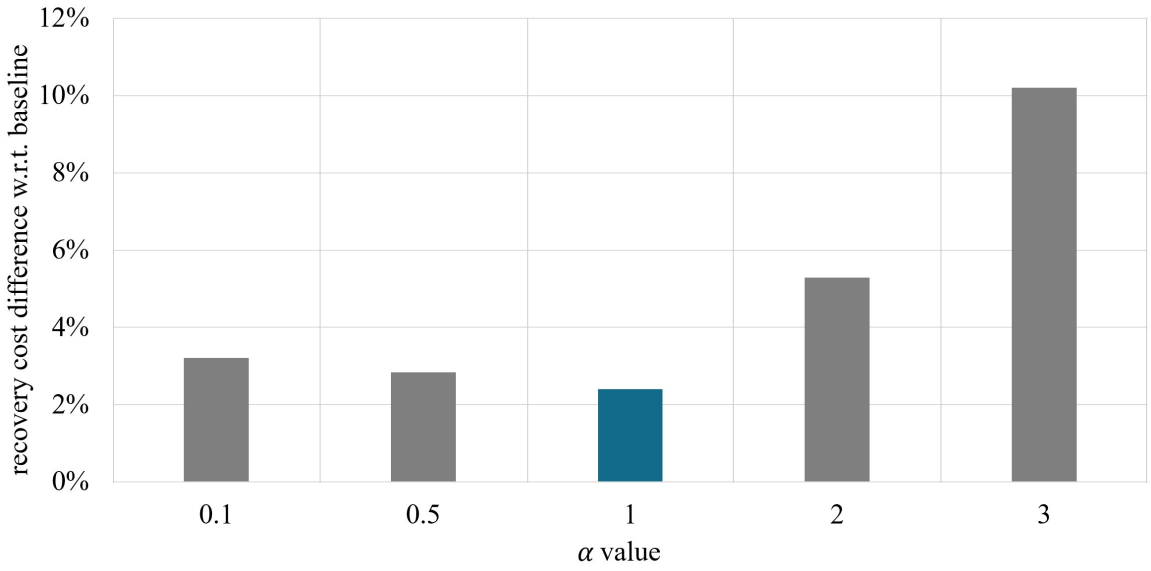


Figure A-7: Prediction confidence α parameter calibration: Recovery cost differences w.r.t. baseline solutions achieved under different solution time limits

A.8 Prediction Confidence Calculation

As described in Section 2.3.4, prediction confidence is calculated for each F/O for a given disruption during the evaluation process before an optimization run. The value α in the given formula was selected from a set of alternatives (0.1, 0.5, 1, 2, 3) based on the resulting solution quality performances. The results for the network with 2,870 flights used in the computational study are presented in Figure A-7. They show that the α value 1 outperforms other candidates for the set of disruption scenarios used for calibration. Based on these results, the value of the parameter α was set to 1 during the computational study presented in Section 2.4.

	Solution time limit (in minutes)				
LF_threshold	1	2	3	4	5
0%	N/A	N/A	N/A	$\geq 6\%$	$\geq 6\%$
0.5%	N/A	N/A	**5.39%**	**4.46%**	**3.32%**
1.5%	**12.54%**	**8.34%**	6.92%	6.13%	5.52%
2.5%	$\geq 15\%$	$\geq 10\%$	$\geq 10\%$	$\geq 10\%$	$\geq 10\%$

Table A.3: Low-frequency threshold calibration: Recovery cost differences w.r.t. baseline solutions achieved under different solution time limits

A.9 Low-frequency Assignment Threshold

Due to the size of the network used in the computational study, even after fixing some of the F/Os, the solution space can still remain large to find high quality solutions in solution time limits of up to 2 minutes. For such cases, an additional solution space reduction measure that removed low-frequency assignments was implemented as described in Section 2.3.5. It filters the assignment decision variables based on the value of *LF_threshold* calibrated for different solution time limits. The value of *LF_threshold* was selected from a set of alternatives (0%, 0.5%, 1.5%, 2.5%) based on the resulting solution quality performances. Results on the calibration set are summarized in Table A.3. The best solution quality levels achieved for each solution time limit are presented in **bold** font. We determined that for 1 and 2 minute solution time limits *LF_threshold* should be set to 1.5%. Below that threshold, the method is unable to generate feasible solutions for at least half of the scenarios while setting it to 2.5% decreases the solution quality. For longer solution time limits *LF_threshold* value 0.5% outperforms other candidates. Lower and higher values decrease the solution quality. Therefore, the value of the parameter *LF_threshold* is set to 1.5% for 1 and 2 minute solution time limits, and to 0.5% for longer solution time limits in the computational study presented in Section 2.4.

	Solution time limit (in minutes)				
PC_threshold	1	2	3	4	5
30%	13.89%	9.02%	6.25%	5.41%	4.33%
35%	**12.17%**	8.55%	5.62%	4.82%	3.83%
40%	12.54%	**8.34%**	**5.39%**	**4.46%**	3.46%
45%	12.96%	9.48%	5.82%	4.67%	**3.32%**
50%	13.24%	10.31%	6.01%	4.83%	3.42%

Table A.4: PC threshold calibration: Recovery cost differences w.r.t. baseline solutions achieved under different solution time limits

A.10 Prediction Confidence Threshold

The primary solution time reduction technique used by our solution approach involves evaluating classifier predictions for a subset F/Os and adding the corresponding constraints to the model. Evaluation requires a *PC_threshold* value to be set for each solution time limit. These values are selected from a set of alternatives (30%, 35%, 40%, 45%, 50%) using a calibration set. Table A.4 includes the achieved solution quality levels with each of these alternatives under different solution time limits on the calibration set. Results show that the value 35% and 45% performs best for the 1 minute and 5 minute solution time limits respectively. For the remaining solution time limits 40% is the best choice. It should be noted that these results are based on the *LF_threshold* set for the solution time limit as described in Appendix A.9 The value of the parameter *PC_threshold* is set to the best threshold values given in Table A.4 for each solution time limit during the computational study presented in Section 2.4. The best solution quality levels achieved for all solution time limits are presented in **bold** font.

Appendix B

Computational study details for Chapter 3: Integrated Aircraft, Crew, and Passenger Recovery

B.1 Limiting the Number of Flight Copies

The solution method presented in Chapter 3 limits the number of copies for individual flights using classifier predictions. We evaluated two alternative ways of reducing the number of flight copies: predicting the maximum allowed delay limits for individual flights and predicting cancellation decisions. Solving recovery problems in limited timeframes requires significant acceleration in the solution process. Therefore, we analyzed these alternatives with respect to their solution acceleration potential. To achieve this, we reduce the solution spaces based on the optimal solutions, instead of ML predictions, which ensures the solution to remain optimal and helps us focus exclusively on the runtime acceleration. This analysis was carried out to pick between predicting maximum allowed delays and predicting cancellations. The network used in this analysis is different from the one used in the computational study, to prevent the developed methods from being tailored to a specific network.

Figure B-1 and B-2 include the results of the comparison for a network with

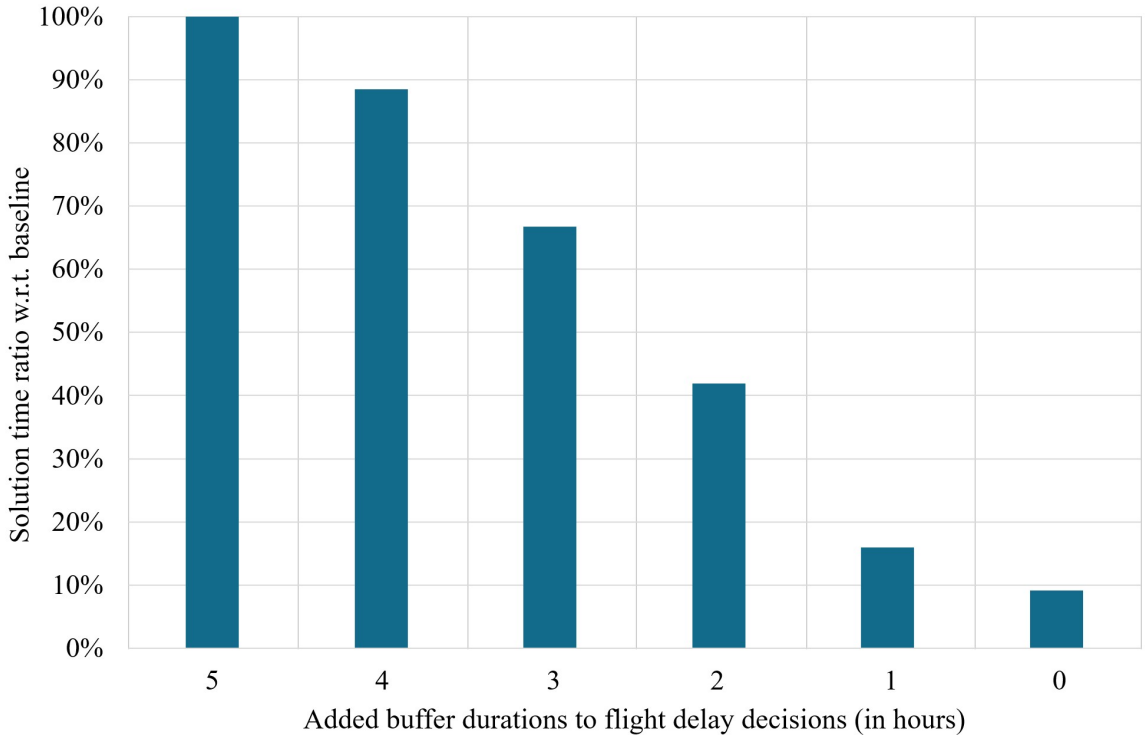


Figure B-1: Solution acceleration performance when predicting delay-related decisions

∼1,400 flights. The y-axes in both figures correspond to the extent of acceleration in the solution time. Lower values are better. In this analysis, we focus only on the solution times. To evaluate the potential acceleration, we consider the case where the predictions are 100% accurate. Classifiers predict the probability that an outcome is in the optimal/near-optimal solution. The optimal/near-optimal solutions provide ∼100% accuracy. Therefore, we used 0.1% optimality gap solutions as predictions in this analysis. Consequently, in this analysis, we have a different delay limit for each flight, as the optimal solutions have different delays for different flights.

For the analysis of the maximum allowed delay limit, we tested various values of the maximum allowed delay limit calculated by adding buffers to flight delays from the 0.1% optimality gap solutions that are generated by solving the IRM without reducing the solution space and as long as necessary to achieve the target optimality gap. The x-axis in Figure B-1 corresponds to the added buffer durations. For example, if a flight is delayed by 1 hour in the 0.1% optimality gap solution, we set the limit to 1, 2, 3, 4, and 5 hours in different runs. All flight delay limits are set in a similar

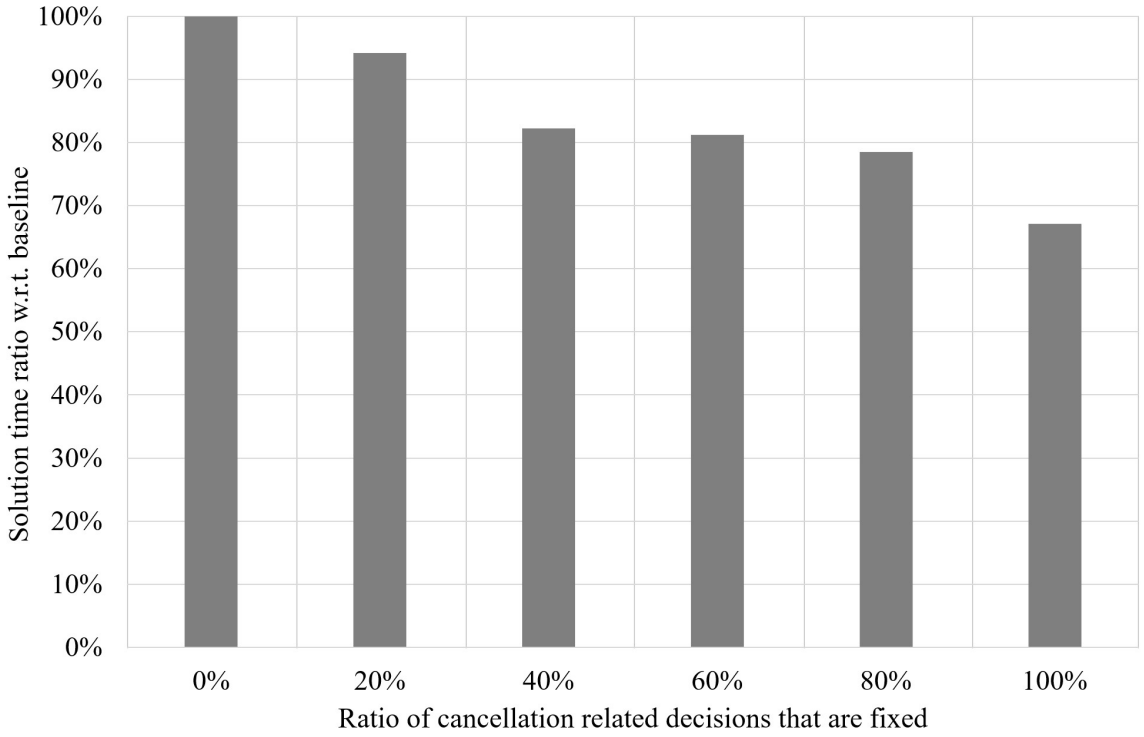


Figure B-2: Solution acceleration performance of predicting cancellation-related decisions

manner. In this analysis, the network-wide limit for the maximum allowed delay was 5 hours (20 time periods). Adding 5 hours of buffer to each delay in the optimal solution is equivalent to solving the original model, the baseline, with only network-wide limits. Therefore, the solution times are also equal (the ratio is 100%). The results show that limiting flight delays can accelerate the solution process by up to 10 times, which happens when all flight delay limits are set based on the optimal solution without any additional buffer.

For cancellation-related decisions, we randomly fixed a specific number of decisions from the 0.1% optimality gap solutions corresponding to the ratio given on the x-axis in Figure B-2. It should be noted that both cancellations and non-cancellations are considered for this analysis. For example, if a flight is not canceled in the 0.1% optimality gap solution, fixing this decision means that cancellation is not allowed for that flight, and if a flight is canceled in the 0.1% optimality gap solution, fixing this decision means that such flight must be canceled. The x-axis value 0% is equivalent to

solving the original model. Therefore, the solution times are also equal (the ratio on the y-axis is 100%). The results shows that even if 100% of the cancellation-related decisions are fixed, the acceleration of the solution process is less than two-fold.

Based on this analysis, we adopted the approach that uses the maximum allowed delay limit predictions for individual flights to limit the number of flight copies included in the solution space.

B.2 Schedule Recovery with Aircraft Considerations

The *sequential* method described in Section 3.4.3 solves the model (B.1)-(B.9) — a simplified version of the IRM — before post-processing. It minimizes the cost associated with flight delays and cancellations without considering the impacts crew schedules and passenger itineraries. The calculation of cancellation and delay costs takes the number of passengers on each flight into account. It also eliminates some of the constraints from the IRM. The retained constraints, (B.2)-(B.8), are the same. The additional notation beyond what is described in Section 3.2.4 and the complete model are presented below.

Notation

ZC_i: cost of canceling flight $i \in I$

DC_i: cost of delay in flight $i \in I$, per (15-minute) time period

Formulation

$$\min \sum_{i \in I} \left(ZC_i \cdot z_i + \sum_{a \in A} \left(AC_i^a \cdot d_i^{a,td_i^l} + \sum_{tr_i^p < t \leq tr_i^l} DC_i \cdot (t - tr_i^p) \cdot (r_i^{a,t} - r_i^{a,t-1}) \right) \right) \tag{B.1}$$

$$\text{s.t.} \quad d_i^{a,t} - d_i^{a,t-1} \geq 0 \qquad \forall i \in I, \forall a \in A, \forall t \in \{td_i^p, .., td_i^l\} \qquad \text{(B.2)}$$

$$r_i^{a,t} - r_i^{a,t-1} \geq 0 \qquad \forall i \in I, \forall a \in A, \forall t \in \{tr_i^p, .., tr_i^l\} \qquad \text{(B.3)}$$

$$r_i^{a,t+ft_i} - d_i^{a,t} = 0 \qquad \forall i \in I, \forall a \in A, \forall t \in \{td_i^p, .., td_i^l\} \qquad \text{(B.4)}$$

$$z_i + \sum_{a \in A} d_i^{a,td_i^l} = 1 \qquad \forall i \in I \qquad \text{(B.5)}$$

$$\sum_{a \in A} \left(\sum_{i \in I_p^D} (d_i^{a,t} - d_i^{a,t-1}) + NE_p^{a,t} \right) \leq CD_p^t \qquad \forall p \in P, \forall t \in T \qquad \text{(B.6)}$$

$$\sum_{a \in A} \left(\sum_{i \in I_p^R} (r_i^{a,t} - r_i^{a,t-1}) + NB_p^{a,t} \right) \leq CR_p^t \qquad \forall p \in P, \forall t \in T \qquad \text{(B.7)}$$

$$\sum_{1 \leq t \leq t'} (NB_p^{a,t} - NE_p^{a,t}) + \sum_{i \in I_p^R} \sum_{SR_{p,a}+1 \leq t < t'} (r_i^{a,t-SR_{p,a}} - r_i^{a,t-SR_{p,a}-1})$$
$$- \sum_{i \in I_p^D} \sum_{1 \leq t < t'} (d_i^{a,t} - d_i^{a,t-1}) \geq 0 \qquad \forall a \in A, \forall p \in P, \forall t' \in T \cup \{|T|+1\} \qquad \text{(B.8)}$$

$$z_i, r_i^{a,t}, d_i^{a,t} \in \{0,1\} \qquad \forall i \in I, \forall a \in A, \forall t \in T \qquad \text{(B.9)}$$

B.3 DB-Stats Method

The *DB-Stats* method relies on database statistics to determine flight-specific delay limits. Each limit is calculated by adding up the average delay of the flight in the solutions database plus four times the standard deviation of the delay. More than 99.5% of the flight delays in the solutions database satisfy these limits.

B.4 Network-Wide Maximum Allowed Delay Limit

The network-wide limit on the maximum allowed delay is a crucial hyperparameter that governs the tractability of the problem. It is the main parameter determining the number of copies to include for each flight. Even in the case of limiting the number of copies based on ML classifier predictions, as discussed above, network-wide maximum allowed delay limit affects many flights that do not have high-probability predictions.

Setting the maximum allowed network-wide delay limit to a low value, e.g., 2 hours, would accelerate the solution process but result in a low-quality solution as many suitable flight copy alternatives would be omitted. On the other hand, setting it to a very high value, e.g., 12 hours, would help find a better solution, but the runtime could increase significantly, due to the huge number of flight copies, rendering the model intractable for practical purposes.

We analyzed the effects of different values of this parameter on the solution quality and time. Figure B-3 compares the values from 5 to 8 hours for the 3,706-flight network used in the computational study. The left axis is the average recovery cost reflecting the solution quality, and the right axis is the average solution time for the tested disruption scenarios. In this analysis, the runtime is limited to 1 hour, and the solution method is the default method presented in Section 3.4.3, which solves the IRM without limiting the number of flight copies other than by the network-wide maximum allowed delay limit. When the network-wide limit is set to 8 hours, the problem becomes too large and results in high-cost solutions to many of the scenarios. This explains the cost difference between the 7- and 8-hour settings. When we compare the 6- and 7-hour settings, we see that the runtime is increased considerably for the 7-hour setting as compared to the 6-hour setting without a significant cost reduction. Based on these results, we set the network-wide limit for the maximum allowed delay limit to 6 hours in our experiments.

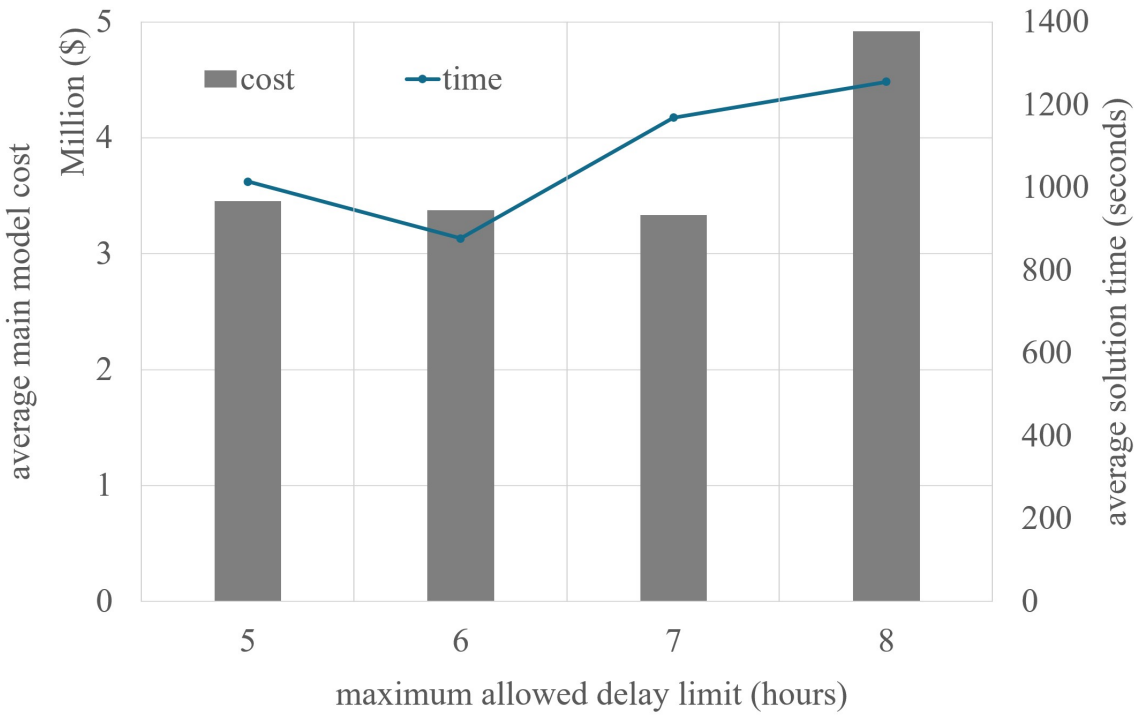


Figure B-3: Network-wide maximum allowed delay limit vs. solution time and quality.

B.5 Delay Separator Limit

The *delay separator limit* used in the classifiers affects the precision performance of the classifiers and the extent of the solution space reduction. For example, when we use 3 hours as the separator limit in the predictions, we can train high-precision classifiers more easily (Section 3.4.2-Figure 3-11), since most flights have a delay of less than 3 hours. However, the solution acceleration can be limited, considering the fact that only a small number of flight copies are removed from the solution space. On the other hand, a significant acceleration is possible with very low values, like 15 minutes. However, this would also remove many suitable flight copies from the solution space and lead to low-quality solutions. Additionally, the precision performance of the classifiers is lower (Section 3.4.2-Figure 3-11).

We conducted an analysis to determine the delay separator limit to use for the network with 3,706 flights in the computational study. For each delay limit separator and prediction confidence threshold value, we calculated a *delay separator limit (DSL) score* using the formula (B.10) below to estimate the acceleration potential of the solution process without significantly affecting the quality of the solution. It is a multiplication of three numbers as we are trying to measure the compound impact. The first $PC\_threshold$ reflects how confident we are in a prediction with an individual PC value of at least $PC\_threshold$. $NUM_{predictions}^{>PCthreshold}$ is the number of predictions that have a PC value greater than the $PC\_threshold$ value. The last number, $(6 - delay\_separator)$, calculates how many hours the allowed delay is reduced by for the corresponding flight. The unit of the $delay\_separator$ value is also hours. By multiplying all these numbers, we get an aggregated measure of the effectiveness of each combination of delay limit separator value and prediction confidence threshold value.

$$DSL_{score} = PC\_threshold \cdot NUM_{predictions}^{\geq PC\_threshold} \cdot (6 - delay\_separator) \quad \text{(B.10)}$$

Figure B-4 shows the results of the analysis. Darker green shaded cells correspond

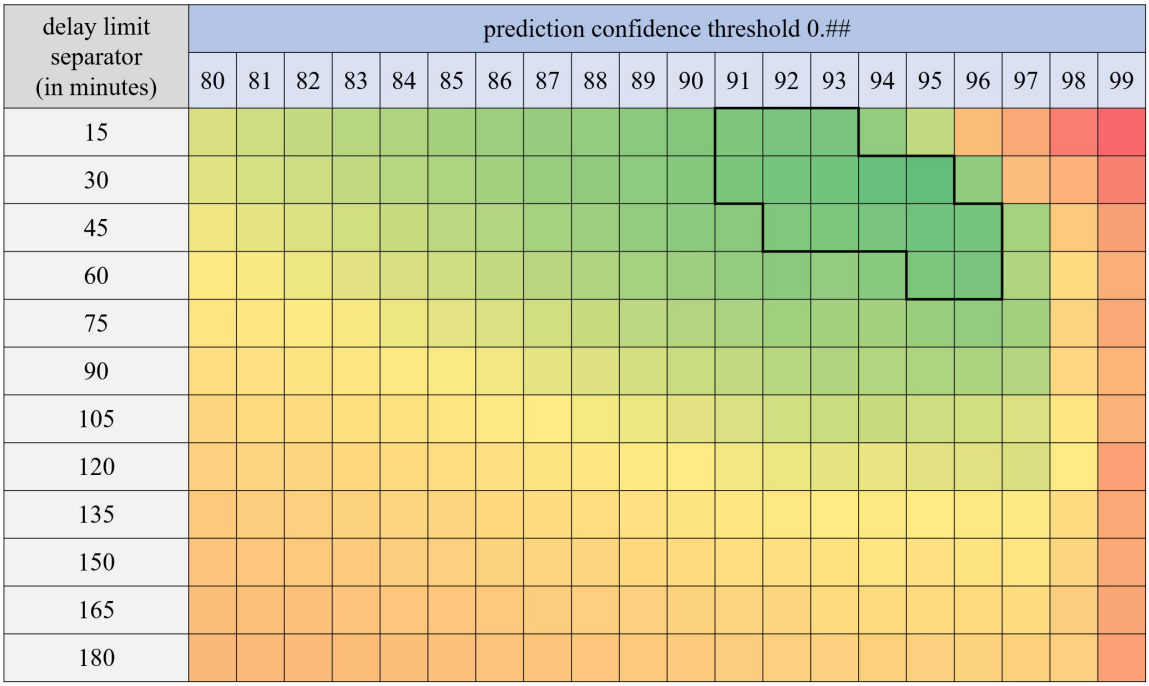


Figure B-4: Delay limit separator prediction confidence scores

to higher scores, while yellow to red shaded cells correspond to lower scores. The area depicted with thick borders contains the top 15 combinations with the highest score. Based on this analysis, we decided to set the delay separator limit value to 45 minutes for the classifiers.

B.6 Prediction Confidence Calculation

As described in Section 3.3.2, we limit the number of flight copies for each flight using the classifier predictions. Before solving a disruption using a mixed-integer optimization solver, the prediction confidence (PC) value is calculated for each flight. The value α in the formula was selected from a set of alternatives (0.1, 0.5, 1, 2, 3) based on the resulting solution quality performances on a set of disruption scenarios used for calibration. Figure B-5 includes the results of this analysis for the network with 3,706 flights used in the computational study. Since the α value 2 outperforms other candidates, the value of the parameter α was set to 2 during the computational study presented in Section 3.4.

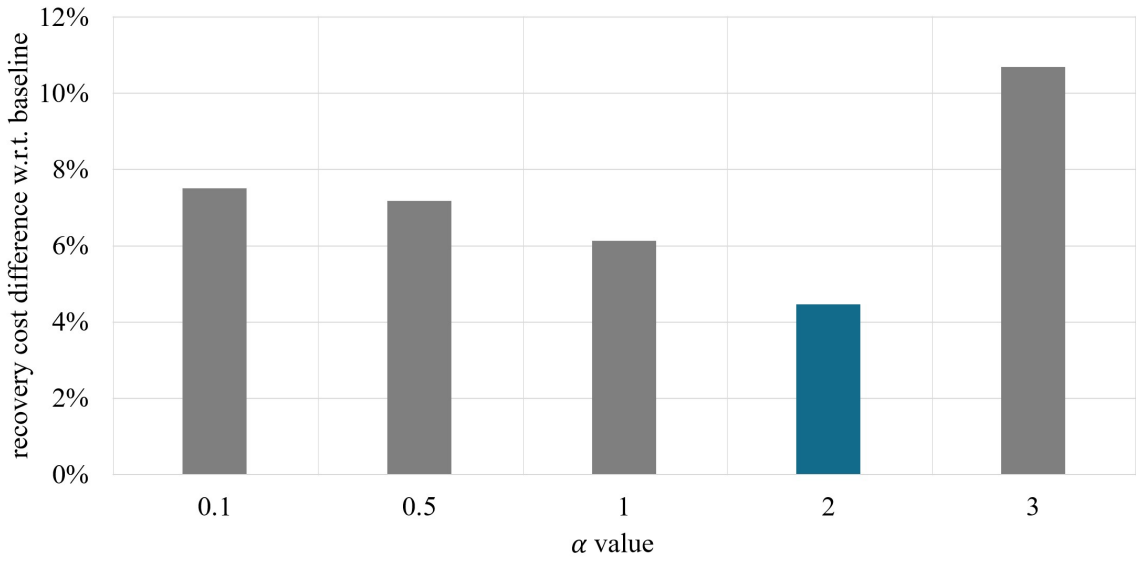


Figure B-5: Prediction confidence α parameter calibration: Recovery cost differences w.r.t. baseline solutions achieved under different solution time limits

	Solution time limit (in minutes)	
PC_threshold	3	5
80%	11.78%	11.76%
85%	9.87%	9.86%
90%	8.25%	7.78%
95%	**5.66%**	**4.96%**
99%	N/A	N/A

Table B.1: PC threshold calibration: Recovery cost differences w.r.t. baseline solutions achieved under different solution time limits

B.7 Prediction Confidence Threshold

Our method relies on reducing the number of copies of each flight based on classifier predictions for flight delay limits. These flight delay limit predictions are evaluated against a *PC_threshold* value as described in Section 3.3.3. These values are selected for each solution time limit from a set of alternatives (80%, 85%, 90%, 95%, 99%) using a calibration set. Table B.1 shows the achieved solution quality levels with each of these alternatives under different solution time limits. The best solution quality levels achieved for each solution time limits are presented in **bold** font. The results show that the value 95% performs best for both solution time limits. The value of the parameter *PC_threshold* is set 95% for each solution time limit during the computational study presented in Section 3.4. The value "N/A" in the table shown for the *PC_threshold* value 99%, refers to the fact that, the solution was infeasible in one fourth of scenarios. For the remaining three fourth of scenarios, the solution quality was quite low ($>$ 25% difference w.r.t. baseline).

Appendix C

Computational study details for Chapter 4: Rule-based Improvements to Recovery Heuristics

C.1 Alternative Crew Recovery Rulesets

Tables C.1, C.2, C.3 and C.4 summarize the rulesets tested. C.1 corresponds to the most complex ruleset with 4 distinct rules. C.2, C.3 and C.4 are subsets of C.1, which includes the rules that are the most frequently observed in trained classifiers. The rulesets C.2 and C.4 have only positive rules, which means that rules that suggest avoiding an F/O in certain circumstances are not included. The rulesets C.1 and C.3 have both positive and negative rules. In our experiments, the simpler rulesets C.3 and C.4 were the best-performing rulesets in that order, implying that it is better to consider only the most generalizable positive and negative rules for a cost-effective candidate ranking.

Rule id	Quarter	Description
1	Any	Prefer an F/O, if it is in the A/C solution and slack $<$ 30 minutes.
2	1	Prefer an F/O, if it is in the A/C solution and NP 2 $<$ 60 minutes.
3	Any	Avoid an F/O, if it is not in the A/C solution and NP 1 $>$ 80 minutes.
4	3-4	Avoid an F/O, if it is not in the A/C solution and NP 2 $>$ 100 minutes.

Table C.1: Ruleset-1 for crew recovery

Rule id	Quarter	Description
1	Any	Prefer an F/O, if it is in the A/C solution and slack $<$ 30 minutes.
2	1	Prefer an F/O, if it is in the A/C solution and NP 2 $<$ 60 minutes.

Table C.2: Ruleset-2 for crew recovery

Rule id	Quarter	Description
1	Any	Prefer an F/O, if it is in the A/C solution and slack $<$ 30 minutes.
2	Any	Avoid an F/O, if it is not in the A/C solution and NP 1 $>$ 80 minutes.

Table C.3: Ruleset-3 for crew recovery

Rule id	Quarter	Description
1	Any	Prefer an F/O, if it is in the A/C solution and slack $<$ 30 minutes.

Table C.4: Ruleset-4 for crew recovery

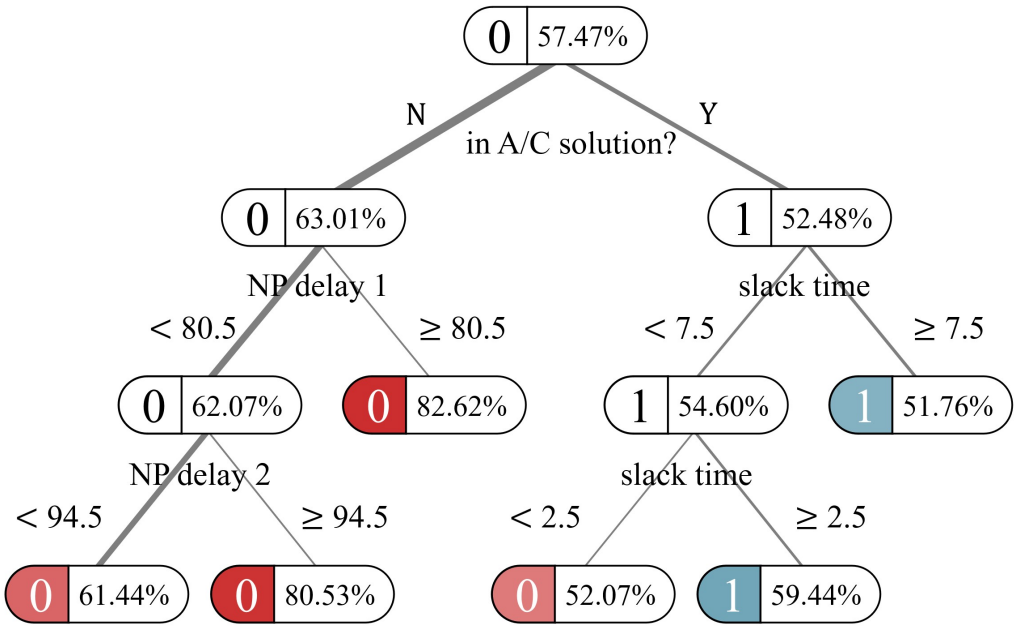


Figure C-1: Crew recovery F/O classifier example for the second quarter of the day

C.2 Crew Recovery Classifier Examples

Figure C-1 shows the connection classifier trained for the second quarter of the day. There are two takeaways from this classifier. First, being in the corresponding A/C recovery solution increases the probability of the F/O being in the crew recovery solution. Second, large NP delay values for the flights in the F/O, lead to circumstances where the F/O should be avoided.

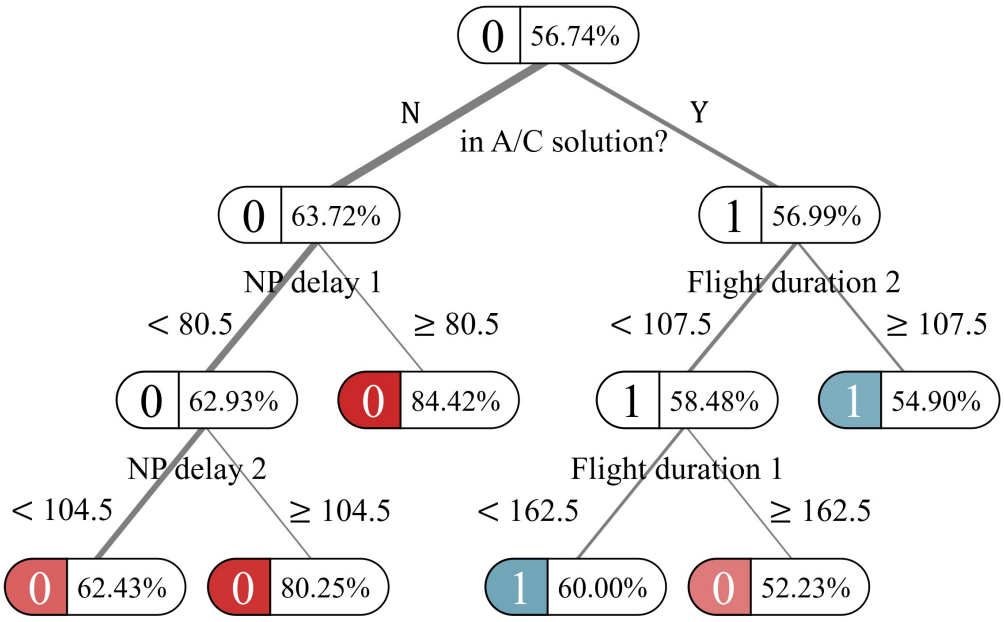


Figure C-2: Crew recovery F/O classifier example for the third quarter of the day

Figure C-2 shows the connection classifier trained for the third quarter of the day. The main takeaway is that for cases where the F/O is included in the A/C solution, the higher the NP delay of the flights in the F/O, the stronger the suggestion to avoid the F/O in the crew solution.

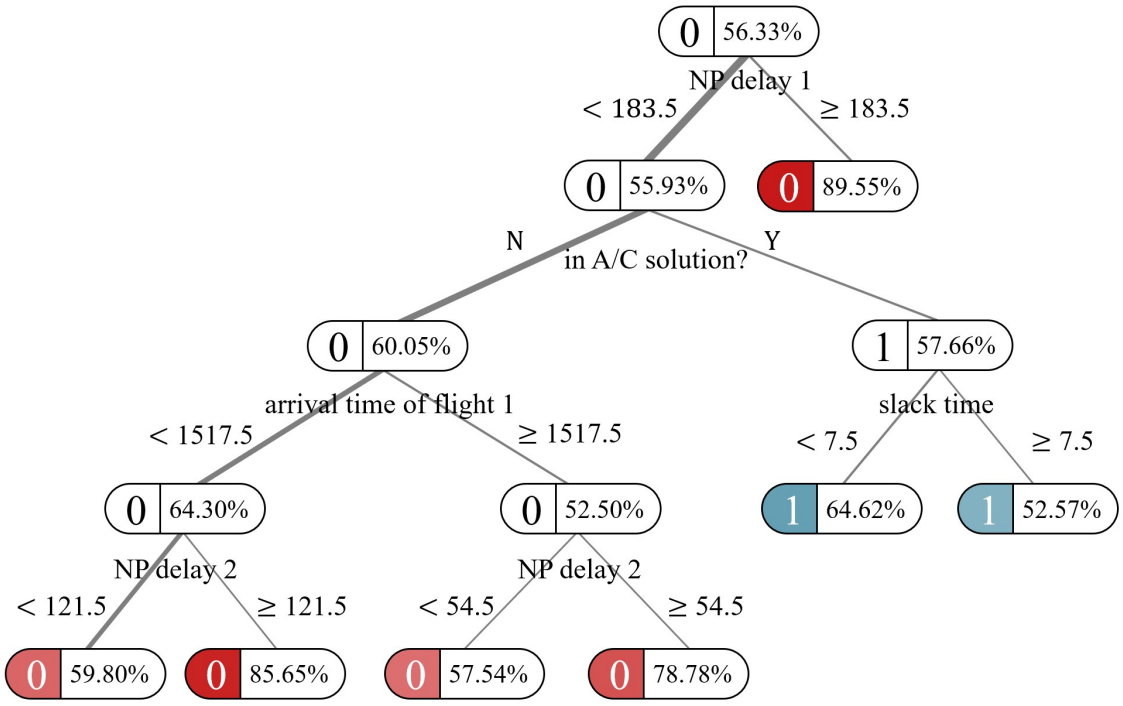


Figure C-3: Crew recovery F/O classifier example for the fourth quarter of the day

Figure C-3 shows the connection classifier trained for the fourth quarter of the day. The takeaway is that when the F/O is included in the A/C solution, shorter slack times lead to a stronger suggestion to utilize the F/O in the crew recovery solution.

Rule id	Delay limit (minutes)	Description
1	30	Limit the flight delay if the departure time < 90 and the number of passengers > 40.
2	30	Limit the flight delay if the departure time < 76 and the number of passengers > 67.
3	30	Limit the flight delay if the frequency < 8
4	45	Limit the flight delay if the departure time > 70 and the number of passengers > 60.
5	45	Limit the flight delay if the departure time < 100 and the number of passengers > 60.
6	45	Limit the flight delay if the frequency > 10, the departure time < 70 and the number of passengers > 40.
7	60	Limit the flight delay if the departure time > 70 and the number of passengers > 75.
8	60	Limit the flight delay if the departure time < 70 and the number of passengers > 55.
9	60	Limit the flight delay if the frequency < 7 and the number of passengers > 40.

Table C.5: Ruleset-1 for integrated recovery

C.3 Alternative Integrated Recovery Rulesets

Tables C.5, C.6, and C.7 summarize the rulesets tested. C.5 corresponds to the most complex ruleset with nine distinct rules. C.6 and C.7 are subsets of C.5. They are created by filtering out rules that are less frequently observed in trained classifiers. In our experiments, the simplest ruleset C.7 performed better than others. With simpler sets, it seems possible to perform a more accurate ranking of the candidates, because the rules are more generalizable. With complex rulesets, some rules are only satisfied by very few candidates, affecting the ranking estimations.

Rule id	Delay limit (minutes)	Description
1	30	Limit the flight delay if the departure time < 90 and the number of passengers > 40.
2	30	Limit the flight delay if the frequency < 8
3	45	Limit the flight delay if the departure time > 70 and the number of passengers > 60.
4	45	Limit the flight delay if the frequency > 10, the departure time < 70 and the number of passengers > 40.
5	60	Limit the flight delay if the departure time > 70 and the number of passengers > 75.
6	60	Limit the flight delay if the departure time < 70 and the number of passengers > 55.

Table C.6: Ruleset-2 for integrated recovery

Rule id	Delay limit (minutes)	Description
1	60	Limit the flight delay if the departure time > 70 and the number of passengers > 75.
2	60	Limit the flight delay if the departure time < 70 and the number of passengers > 55.

Table C.7: Ruleset-3 for integrated recovery

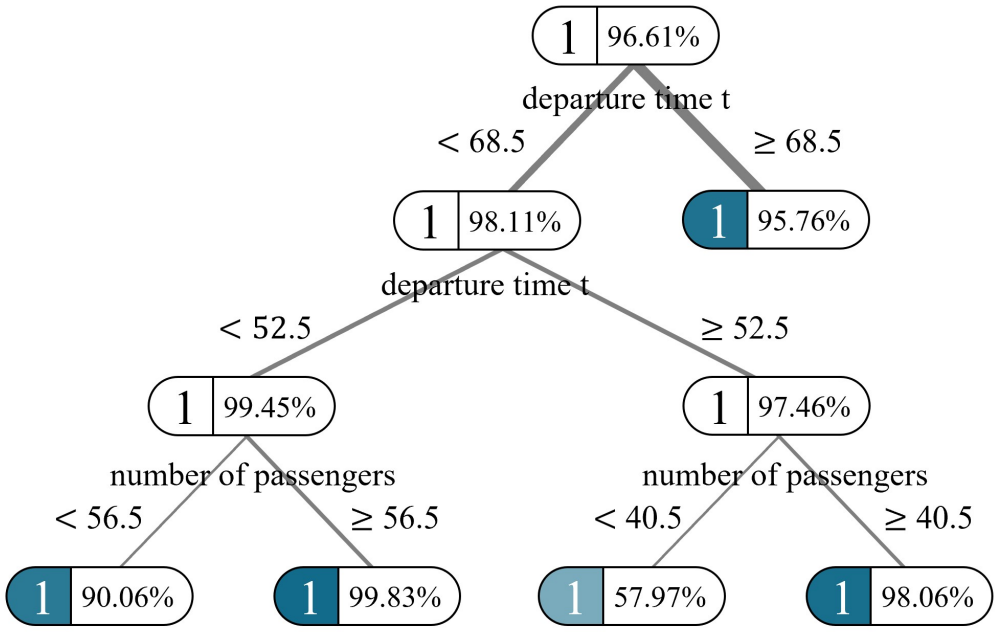


Figure C-4: Integrated recovery maximum allowed delay limit classifier example

C.4 Integrated Recovery Classifier Examples

Figure C-4 shows a 1-hour delay separator limit classifier trained using two features: departure time and number of passengers. Label 1 reflects that the delay of the corresponding flight should be limited to 1 hour. The probabilities, reflecting the confidence in the prediction, are given inside the nodes. The classifier structure demonstrates a behavior that overlaps with the ruleset determined in Table 4.8. With the increasing number of passengers, the classifier suggests limiting the flight delay to 1 hour, and the threshold for the number of passengers to determine the delay limit depends on the departure time.

References

Abdelghany, A., Ekollu, G., Narasimhan, R., & Abdelghany, K. (2004). A proactive crew recovery decision support tool for commercial airlines during irregular operations. *Annals of Operations Research*, *127*(1-4), 309–331.

Aguiar, B., Torres, J., & Castro, A. J. (2011). Operational problems recovery in airlines–a specialized methodologies approach. In *Progress in Artificial Intelligence: 15th Portuguese Conference on Artificial Intelligence, EPIA. Proceedings 15* (pp. 83–97).

Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1988). *Network Flows*. Prentice Hall.

Alabi, D., Kalai, A. T., Liggett, K., Musco, C., Tzamos, C., & Vitercik, E. (2019). Learning to prune: Speeding up repeated computations. In *Conference on Learning Theory* (pp. 30–33).

Alvarez, A. M., Louveaux, Q., & Wehenkel, L. (2017). A machine learning-based approximation of strong branching. *INFORMS Journal on Computing*, *29*(1), 185–195.

Archetti, C., & Speranza, M. G. (2014). A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, *2*(4), 223–246.

Balcan, M.-F., Dick, T., Sandholm, T., & Vitercik, E. (2018). Learning to branch. In *International Conference on Machine Learning* (pp. 344–353).

Ball, M., Barnhart, C., Dresner, M., Hansen, M., Neels, K., Odoni, A., ... Voltes, A. (2010). worldwide. In *NEXTOR Research Symposium, Washington DC.*

Baltean-Lugojan, R., Bonami, P., Misener, R., & Tramontani, A. (2018). Selecting cutting planes for quadratic semidefinite outer-approximation via trained neural networks. *Optimization-Online*.

Barnhart, C., Fearing, D., Odoni, A., & Vaze, V. (2012). Demand and capacity management in air transportation. *EURO Journal on Transportation and Logistics*, *1*(1-2), 135–155.

Barnhart, C., Fearing, D., & Vaze, V. (2014). Modeling passenger travel and delays in the national air transportation system. *Operations Research*, *62*(3), 580–601.

Barnhart, C., & Smith, B. (2012). *Quantitative Problem Solving Methods in the Airline Industry* (Vol. 169). Springer.

Barnhart, C., & Vaze, V. (2015a). Airline schedule optimization. *The Global Airline Industry*, 189–222.

Barnhart, C., & Vaze, V. (2015b). Irregular operations: Schedule recovery and robustness. *The Global Airline Industry*, *23*, 263–286.

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

Bertsimas, D., & Patterson, S. S. (1998). The air traffic flow management problem with enroute capacities. *Operations Research*, *46*(3), 406–422.

Bertsimas, D., & Stellato, B. (2022). Online mixed-integer optimization in milliseconds. *INFORMS Journal on Computing*, *34*(4), 2229–2248.

Bertsimas, D., & Tsitsiklis, J. N. (1997). *Introduction to linear optimization* (Vol. 6). Athena scientific Belmont, MA.

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

Bixby, E. R., Fenelon, M., Gu, Z., Rothberg, E., & Wunderling, R. (1999). Mip: Theory and practice—closing the gap. In *Ifip conference on system modeling and optimization* (pp. 19–49).

Boschetti, M. A., Maniezzo, V., Roffilli, M., & Bolufé Röhler, A. (2009). Matheuristics: Optimization, simulation and control. In *International Workshop on Hybrid Metaheuristics* (pp. 171–177).

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

Breiman, L. (2001). Random forests. *Machine learning*, *45*, 5–32.

BTS. (2020). *Airline on-time performance data.* Retrieved from `https://www.transtats.bts.gov`

Chen, C.-H., & Chou, J.-H. (2016). Multiobjective optimization of airline crew roster recovery problems under disruption conditions. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *47*(1), 133–144.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794).

Choi, S., Kim, Y. J., Briceno, S., & Mavris, D. (2016). Prediction of weather-induced airline delays based on machine learning algorithms. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)* (pp. 1–6).

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., & Rezanova, N. J. (2010). Disruption management in the airline industry—concepts, models and methods. *Computers & Operations Research*, *37*(5), 809–821.

Cook, A. J., & Tanner, G. (2011). *European airline delay cost reference values.* EUROCONTROL Performance Review Unit.

Danna, E., Fenelon, M., Gu, Z., & Wunderling, R. (2007). Generating multiple solutions for mixed integer programming problems. In *International conference on integer programming and combinatorial optimization* (pp. 280–294).

Dodge, Y. (2008). *The concise encyclopedia of statistics.* Springer Science & Business Media.

Doganis, R. (2005). *Airline business in the 21st century.* Routledge.

FAA. (2020). *Benefit-cost analysis.* Retrieved from `https://www.faa.gov/regulations_policies/policy_guidance/benefit_cost`

FAA. (2021). *Aviation system performance metrics.* Retrieved from `https://aspm.faa.gov/`

Furian, N., O'Sullivan, M., Walker, C., & Çela, E. (2021). A machine learning-based branch and price algorithm for a sampled vehicle routing problem. *OR Spectrum*, *43*, 693–732.

Gao, Q., Tang, X., & Zhu, J. (2010). Research on greedy simulated annealing algorithm for irregular flight schedule recovery model. In *Advances in grey systems research* (pp. 503–513). Springer.

Gasse, M., Chetelat, D., Ferroni, N., Charlin, L., & Lodi, A. (2019). Exact combinatorial optimization with graph convolutional neural networks. *Advances in Neural Information Processing Systems*, *32*, 15580–15592.

Gopalakrishnan, K., & Balakrishnan, H. (2017). A comparative analysis of models for predicting delays in air traffic networks. In *ATM seminar.*

Gopalakrishnan, K., Balakrishnan, H., & Jordan, R. (2016). Clusters and communities in air traffic delay networks. In *2016 American Control Conference (ACC)* (pp. 3782–3788).

Gorripaty, S., Liu, Y., Hansen, M., & Pozdnukhov, A. (2017). Identifying similar days for air traffic management. *Journal of Air Transport Management*, *65*, 144–155.

Guo, Y. (2005). A decision support framework for the airline crew schedule disruption management with strategy mapping. In *Operations research proceedings 2004* (pp. 158–165).

Gurobi Optimization Inc. (2020). *Gurobi optimizer reference manual.* Retrieved from https://www.gurobi.com/documentation/current/refman/index.html

Hassan, L., Santos, B. F., & Vink, J. (2021). Airline disruption management: A literature review and practical challenges. *Computers & Operations Research*, *127*, 105-137.

Heinold, A. (2008). Maintenance reachability at southwest airlines. AGIFORS Annual Symposium 2008, Montreal.

Hondet, G., Delgado, L., & Gurtner, G. (2018). Airline disruption management with aircraft swapping and reinforcement learning..

Hu, Y., Song, Y., Zhao, K., & Xu, B. (2016). Integrated recovery of aircraft and passengers after airline operation disruption based on a grasp algorithm. *Transportation Research Part E: Logistics and Transportation Review*, *87*, 97–112.

IATA. (2023a). *Airline profitability outlook strengthens.* Retrieved from https://www.iata.org/en/pressroom/2023-releases/2023-06-05-01/

IATA. (2023b). *Net profit of commercial airlines worldwide from 2006 to 2021 and with a forecast until 2023.* Retrieved from https://www.statista.com/statistics/232513/net-profit-of-commercial-airlines-worldwide/

InterpretableAI. (2020). *Interpretable AI documentation.* Retrieved from https://docs.interpretable.ai

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

Johnson, E., Lettovsky, L., Nemhauser, G., Pandit, R., & Querido, S. (1994). Final report to northwest airlines on the crew recovery problem. *The Logistic Institute, Georgia Institute of Technology, Atlanta, GA*.

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

Kim, Y. J., Choi, S., Briceno, S., & Mavris, D. (2016). A deep learning approach to flight delay prediction. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)* (pp. 1–6).

Kuhn, K. D. (2016). A methodology for identifying similar days in air traffic flow management initiative planning. *Transportation Research Part C: Emerging Technologies*, *69*, 1–15.

Lan, S., Clarke, J.-P., & Barnhart, C. (2006). Planning for robust airline operations: Optimizing aircraft routings and flight departure times to minimize passenger disruptions. *Transportation Science*, *40*(1), 15–28.

Lettovsky, L. (1997). *Airline operations recovery: an optimization approach.* Georgia Institute of Technology.

Lettovskỳ, L., Johnson, E. L., & 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., & 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.

Maher, S. J. (2015). A novel passenger recovery approach for the integrated airline recovery problem. *Computers & Operations Research*, *57*, 123–137.

Maher, S. J. (2016). Solving the integrated airline recovery problem using column-and-row generation. *Transportation Science*, *50*(1), 216–239.

Maniezzo, V., Stützle, T., & Voß, S. (2021). *Matheuristics.* Springer.

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

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

Medard, C. P., & Sawhney, N. (2007). Airline crew scheduling from planning to operations. *European Journal of Operational Research*, *183*(3), 1013–1027.

Mitchell, T. M. (1997). *Machine learning.*

Morabit, M., Desaulniers, G., & Lodi, A. (2021). Machine-learning–based column selection for column generation. *Transportation Science*, *55*(4), 815–831.

NAA. (2022). *Types of aviation maintenance checks.* Retrieved from `https://www.naa.edu/types-of-aviation-maintenance-checks/`

Nissen, R., & Haase, K. (2006). Duty-period-based network model for crew rescheduling in european airlines. *Journal of Scheduling*, *9*(3), 255–278.

Novianingsih, K., Hadianti, R., Uttunggadewa, S., & Soewono, E. (2015). A solution method for airline crew recovery problems. *International Journal of Applied Mathematics & Statistics*, *53*(4), 137–149.

Palpant, M., Boudia, M., Robelin, C., Gabteni, S., & Laburthe, F. (2009). Roadef 2009 challenge: Disruption management for commercial aviation. *Amadeus S.A.S., Operations Research Division*.

Petersen, J. D., Sölveling, G., Clarke, J.-P., Johnson, E. L., & Shebalov, S. (2012). An optimization approach to airline integrated recovery. *Transportation Science*, *46*(4), 482–500.

Rashedi, N., Sankey, N., Vaze, V., & Wei, K. (2023). A machine learning approach for solution space reduction in aircraft disruption recovery. *Available at SSRN 4444548*.

Rebollo, J. J., & Balakrishnan, H. (2014). Characterization and prediction of air traffic delays. *Transportation Research Part C: Emerging technologies*, *44*, 231–241.

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

Salvagnin, D. (2016). Detecting semantic groups in mip models. In *Integration of ai and or techniques in constraint programming: 13th international conference, cpaior 2016, banff, ab, canada, may 29-june 1, 2016, proceedings 13* (pp. 329–341).

Song, M., Wei, G., & Yu, G. (1998). A decision support framework for crew management during airline irregular operations. In *Operations Research in the Airline Industry* (pp. 259–286). Springer.

Stojković, M., & Soumis, F. (2001). An optimization model for the simultaneous operational flight and pilot scheduling problem. *Management Science*, *47*(9), 1290–1305.

Stojković, M., Soumis, F., & Desrosiers, J. (1998). The operational airline crew scheduling problem. *Transportation Science*, *32*(3), 232–245.

Tang, Y., Agrawal, S., & Faenza, Y. (2020). Reinforcement learning for integer programming: Learning to cut. In *International Conference on Machine Learning* (pp. 9367–9376).

Teodorović, D., & Guberinić, S. (1984). Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operational Research*, *15*(2), 178–182.

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

Thengvall, B. G., Bard, J. F., & Yu, G. (2000). Balancing user preferences for aircraft schedule recovery during irregular operations. *IIE Transactions*, *32*(3), 181–193.

Vance, P. H., Atamturk, A., Barnhart, C., Gelman, E., Johnson, E. L., Krishna, A., ... Rebello, R. (1997). A heuristic branch-and-price approach for the airline crew pairing problem. *preprint*.

Vink, J., Santos, B. F., Verhagen, W. J., Medeiros, I., et al. (2020). Dynamic aircraft recovery problem-an operational decision support framework. *Computers & Operations Research*, *117*, 104892.

Wang, S., & Vaze, V. (2016). Modeling probability distributions of primary delays

in the national air transportation system. *Transportation Research Record*, *2569*(1), 42–52.

Wei, G., Yu, G., & Song, M. (1997). Optimization model and algorithm for crew management during airline irregular operations. *Journal of Combinatorial Optimization*, *1*, 305–321.

Yang, T., & Hu, Y. (2019). Considering passenger preferences in integrated postdisruption recoveries of aircraft and passengers. *Mathematical Problems in Engineering*, *2019*, 1–19.

Yu, G., Argüello, M., Song, G., McCowan, S. M., & White, A. (2003). A new era for crew recovery at Continental Airlines. *Interfaces*, *33*(1), 5–22.

Zhang, D., Lau, H. H., & Yu, C. (2015). A two stage heuristic algorithm for the integrated aircraft and crew schedule recovery problems. *Computers & Industrial Engineering*, *87*, 436–453.

Zhang, D., Yu, C., Desai, J., & Lau, H. H. (2016). A math-heuristic algorithm for the integrated air service recovery. *Transportation Research Part B: Methodological*, *84*, 211–236.

Zhao, X., Zhu, J., & Guo, M. (2007). Application of grey programming in irregular flight scheduling. In *IEEE International Conference in Industrial Engineering and Engineering Management* (pp. 164–168).

Zhu, B., Clarke, J.-P., & Zhu, J. (2016). Real-time integrated flight schedule recovery problem using sampling-based approach. *Journal of Computational and Theoretical Nanoscience*, *13*(2), 1458–1467.