Airline Disruption Management

A Deep Reinforcement Learning Framework for the Aircraft Recovery Problem: A Comparative Analysis of Proactive and Reactive Strategies focussing on the State-Space and Reward Formulations

Elisabeth Oosthoek

Airline Disruption Management

A Deep Reinforcement Learning Framework for the Aircraft Recovery Problem: A Comparative Analysis of Proactive and Reactive Strategies focussing on the State-Space and Reward Formulations

Thesis report

by

Elisabeth Oosthoek

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on February 10th, 2025 at 09:00

Thesis committee:

Chair:	Dr. Ir. J. Ellerbroek
Supervisors:	Dr. M.J. Ribeiro
External examiner:	Dr. R. Merino Martinez
Place:	Faculty of Aerospace Engineering, Delft
Project Duration:	March, 2025 - January, 2025
Student number:	5056470

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Faculty of Aerospace Engineering · Delft University of Technology

Preface

This thesis marks the end of my journey at Delft University of Technology and is the final fulfilment for the degree MSc Aerospace Engineering.

I would like to sincerely thank my amazing supervisor Marta Ribeiro and the brilliant PhD student Fynn Oppermann for their unwavering support. I like to express my heartfelt appreciation for your constant guidance. During our many meetings neither of you ever left any questions unanswered and were always there no matter what. In times when I felt lost, you motivated me and assured that I could do it. Most importantly, although the thesis was overwhelming at times and significant roadblocks were encountered along the way, you gently encouraged me and gave me the confidence to go on. Without your generous time and effective feedback, this thesis could not have attained the level it has now.

I like to dedicate this thesis to my family, whose constant, loving, and unwavering support has guided me throughout my university journey. First to my mama, who inspires me everyday. When people talk about strong women and female pioneers dedicated to their careers while willing to sacrifice everything for their children, I think of you. Most people can only dream of all you have achieved. Thank you for raising me so well and installing a love for beautiful gardens, movies, music, and art in your three girls. Your mind and character remind me of Monet's *Promenade sur la falaise, Pourville*. Not only are you the woman standing above all else, watching over us, but you are also the wind in the boats' sails and the water that supports them, ensuring they will always find their path. Next, here's to my father, who I am incredibly grateful for. Your painting would be Paul Klee's *Der Niesen*. The mountain that provides safety to our family and never budges. I can always lean against it and it will never fail to support me. It signifies all the hikes in Switzerland, that hold our laughter in every crevice of the muddy and rocky terrain. To both of you, I thank you most of all for giving me my two best friends, Hester and Frederiek. Dear Hester, you embody *the girl with the pearl earring*; a beautiful, kind hearted soul with an incredible talent for- and knowledge of gems and jewellery. I am so proud of your journey and I know you will be the best mother in the world. Lastly, to you Kiki, the strongest amongst us all. You remind me of *Liberty Leading the People*; one whose dedication, passion, and perseverance unites people and inspires me daily. You have a resilience most people can only dream of. Or the work *Cecilia Gallerani*. Like her, you are incredibly smart, educated, and have a talent and passion for poetry. Always remember that there is a reason for everything and every moment makes you stronger. To both of you, I will always be here and support you like you have always supported me.

Furthermore, I'd like to thank Isabelle, without whom I would not be standing here. Your support has meant more than you can ever imagine and I am forever grateful that you are part of my life. Thank you for always picking up my calls, for motivating me when I was at my lowest, and for inspiring me everyday. I hope we will forever continue our hot girl walks. Many thanks too to Catherine, whose dedication and resilience to her medicine study inspires me greatly. Thank you for entertaining me with all your life's stories. I know that all you do will be worth it and that you will be an incredible doctor. From innocent crushes, to boring Latin lessons, to first beers, ski holidays, and university stories, I love you guys.

Lastly, thanks to all my university friends without whom I could not have done it. To Jan, who believed more in me than I did myself and who is the smartest human I know. To my fellow Dutchie, Emma, who motivated me since day 1 of the Bachelors, and to whom I owe many laughs in our little MdR studios. To Archie, whom I am forever grateful for for an amazing 2025 in Delft. To Pavan and Alexis, who have made me laugh constantly and whose (sometimes nonsensical and dark) humour has made my life a little better. To Alper, whose positive attitude and funny stories have kept me laughing. To Sahir, who taught me that there is nothing more important than the power of friendship. You are an incredible person and I am forever grateful for our many coffee and tea runs! To you all and all other friends I could not name, thank you for your friendship.

I know that down the road (or shall I say down the runway), I will look back with a large smile on my time in Delft, so at last, thank you to all!

Elisabeth Oosthoek
Delft, January 2026

Contents

Nomenclature

List of Abbreviations

Abbreviation	Description
A2C	Advantage Actor-Critic
AC	Aircraft
ADM	Airline Disruption Management
ADMM	Alternating Direction Method of Multipliers
ARP	Aircraft Recovery Problem
CG	Column Generation
DDBA	Decision-Decomposition-Based Algorithm
DDQL	Deep Double Q-Learning
DP	Dynamic Programming
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
EX	Exact Method
HH	Hybrid Heuristic (includes Matheuristics & SimHeuristics)
ID	Identifier
LBBD	Logic-Based Benders Decomposition
LGB	Light Gradient Boosting
MDP	Markov Decision Process
MIP	Mixed Integer Programming
NN	Neural Network
RL	Reinforcement Learning
SAA	Sample Average Approximation
SGD	Stochastic Gradient Descent
SRPDM	Stochastic Reactive and Proactive Disruption Management
TD	Temporal Difference

Latin Symbols

Symbol	Description
a_t	Action taken by the agent at time step t
ac	Aircraft identifier index in action tuple
AC_{mtx}	Aircraft Unavailability Matrix (Model 2)
$\mathscr{A}$	Discrete action space
$\mathscr{B}$	Mini-batch of experiences sampled from replay buffer
B_{resolve}	Positive reward bonus for resolving a conflict
C	Number of columns in state matrix
c_{curr}	Current number of conflicts
c_{init}	Initial number of conflicts
$\mathscr{D}$	Experience replay buffer
D	Dimension of the observation space
f	Flight identifier index in action tuple
f_i	Flight ID feature for flight i
$f_{i,arr/dep}$	Arrival and Departure times for flight i
$f_{i,conf}$	Conflict count for flight i
FL_{mtx}	Flight Matrix (Model 2)

Symbol	Description
$J(\pi)$	Objective function representing expected long-term performance
K	Number of consecutive states stacked in observation
$L(\theta)$	Loss function for the Neural Network
N_{ac}	Number of aircraft considered
N_{fl}	Number of flights considered
$N_{fl,max}$	Maximum number of flights
$\mathbf{o}_t$	Observation vector perceived by the agent at time t
$o_{i,j}$	Binary occupancy indicator for flight i in time interval j
p	Probability of occurrence of a disruption
p_i	Unavailability probability for aircraft i
$p_{i,start/end}$	Time remaining until start/end of unavailability period
$Q(s,a;\theta)$	Approximate Q-function (Neural Network) parameterized by θ
$Q^{\pi}(s,a)$	Expected cumulative reward under policy π
$Q^{*}(s,a)$	Optimal Q-function (maximum expected return)
r	Immediate reward (shorthand for $R(s,a,s')$)
R	Number of rows in state matrix (Model 1)
$R(s,a,s')$	Reward function
R_x	Reward penalty components (where $x \in$ {auto, cancel, delay, inaction, time, unresolved})
S	General state space
S_1	Matrix representation of state space for Model 1
S_2	State space for Model 2
$\mathbf{s}_{base}$	Base state vector including flattened matrix and temporal features
$\mathbf{s}_t$	State vector at time t
$\mathbf{T}$	Temporal feature vector (Model 1)
T	Episode length; also Number of time intervals in Model 2
t, t_{end}	Current time step and time remaining in simulation

Greek Symbols

Symbol	Description
α	Learning rate; also probability transition parameter
β	Probability transition parameter (probability decrease)
γ	Discount factor for future rewards
Δp_i	Magnitude of change in probability value
θ, θ^{-}	Neural network parameters (weights) and target network parameters
π, π^{*}	Policy followed by the agent and Optimal policy
τ	Trajectory of states and actions
∇_{θ}	Gradient with respect to parameters θ

I. Scientific Article

A Deep Reinforcement Learning Framework for the Aircraft Recovery Problem: A Comparative Analysis of Proactive and Reactive Strategies focussing on the State-Space and Reward Formulations

Elisabeth Oosthoek

Delft University of Technology, Delft, The Netherlands

Abstract

With a rising demand for air travel, Airline Disruption Management (ADM) ensures successful schedule recovery in the event of disruptions. The Aircraft Recovery Problem or ARP, part of ADM, solely focusses on aircraft. Previous research has concentrated on exact optimisation, as well as simple-, meta-, or hybrid-heuristic solution methods. However, in order to prevent significant delays, resolution decisions must be made fast. This need combined with the rise of deep learning, has led to the emergence of deep reinforcement learning (DRL) as a viable solution strategy. Nevertheless, the performance of DRL remained often limited to specific state space formulations and reward designs. In order to close this gap, the primary objective of this work is to further optimise a reinforcement learning (RL) formulation for the aircraft recovery problem (ARP) while minimising disruption effects. It investigates and compares two models with alternate state space formulations. First, we test a single, aircraft-centric and continuous design. Second, we presents a dual, sparse, flight-centric, and primarily binary formulation. Each model compares computational efficiency, action distribution, and conflict resolution effectiveness across three DRL environments; proactive, reactive, and myopic, subject to different levels of stochastic state information. It was found that the state space formulation significantly influences computation time, which is a prominent issue faced by big action- and state space sizes. Furthermore, it is shown that proactive environments result in better conflict resolution. However, significant challenges of the model were revealed by the unexpected negative learning trend. This counterintuitive result was further underlined by the notably higher performance during exploration than during exploitation, indicating the DRL agent's inability to learn an optimal policy. Finally, sensitivity analyses of the reward and a hyperparameter underlined the high susceptibility of RL to minor parameter tweaks, stressing the challenging implementation of DRL models for real-life applications.

1 Introduction

Air travel is experiencing increasing demand with more people choosing flights as their preferred method of transportation. In 2026 it is expected that passenger numbers will reach 5.1 billion, exceeding the five billion mark for the first time in aviation history, underlining the enormous demand (IATA 2025). A collaborative, structured, and meticulously organised approach is salient to ensure smooth operations worldwide. Hence, significant efforts are poured into the creation of optimal flight schedules. In real life, operations scheduling is affected by unforeseen disruptions such as, amongst others, airport closures, flight delays, mechanical issues, and weather events. This has caused the emergence of airline disruption management (ADM), a field in operational research focussing on the recovery of disrupted flight schedules to ensure optimal efficiency while minimising costs. An airline's success depends on good disruption management for it highly impacts customer satisfaction.

The relevance of airline disruption management has led to the establishment of three core problems guiding scientific research: the aircraft recovery problem (ARP), the aircraft and crew recovery problem, and finally the integrated recovery problem investigating aircraft, passengers and crew alike. Many works focus on exact optimisation solution methods (Wu et al. 2017; Peng et al. 2024; Aktürk et al. 2014) such as integer programming, dynamic programming, or decomposition algorithms. Next to exact methods, simple heuristics and hybrid- and meta-heuristics have been investigated as viable approaches. However, all these solution methods are accompanied by a trade-off between computation time and quality of solution, that is, efficiency and effectiveness (Hassan et al. 2021). Furthermore, a reactive approach to conflict detection is most often taken, meaning that disruptions are resolved only when they are certain and imminent. Thus a key gap has been the lack of anticipatory planning.

This led to the discovery of deep reinforcement learning (DRL) as a good candidate for airline disruption management as used by Ding et al. (2023) and Wang et al. (2025). It improves speed and is computationally efficient, highlighting its real world adaptability. Moreover, deep neural networks can analyse large and complicated amounts of data and identify patterns otherwise impossible. In addition, Ding et al. (2023) underlined that DRL makes it viable to introduce uncertainty modelling when simulating disruptions. By using stochastic evolving disruptions, real life can better be mirrored. Furthermore, Becking (2024)'s work used not only a DRL framework but also a novel proactive instead of reactive approach, focussing on the anticipation of possible disruptions.

This research aims to further investigate the impact of- and difference between proactive and reactive DRL strategies applied to the aircraft recovery problem. Furthermore, current DRL solution methods are often limited by their state space formulation. Hence, this paper will contribute and improve research into DRL for the ARP. More specifically, it will investigate two alternate state space designs and observe the effects on the DRL agent.

This paper is structured as follows. Chapter 2 outlines the problem description. Next, Chapter 3 presents relevant works in- or related to the field of ADM. Chapter 4 discusses the methodology, laying out crucial DRL concepts as well as the model implementation. This includes the training procedure and network architecture. The hypotheses are explained in Chapter 5. Chapter 6 critically investigates the results while Chapter 8 discusses their impacts, identifies crucial limitations, and stresses important considerations for future research. Chapter 7 details a sensitivity analysis and debates the model's robustness. Finally, Chapter 9 concludes the work's findings.

2 Problem Description

The ARP can be characterised by five entities: the aircraft fleet, the flight schedule, the recovery period, the number and probability of aircraft unavailability periods, and existing flight conflicts. The overall objective of the ARP is to minimise operational costs by maximising conflict resolution using proactive actions i.e. resolving conflicts before flights depart, minimising both manual and automatic cancellations, and optimising action choices. This work will focus on solving the aircraft recovery problem using proactive and reactive DRL approaches. The possible recovery actions constitute cancelling a flight, delaying a flight, and tail swaps. The network is presented in the form of a hub and spoke network.

The first entity, the aircraft fleet, comprises an arbitrary amount of aircraft (AC), depending on the scope of problem. The flight schedule comprises a predefined set of flights per aircraft with departure and arrival times as well as the total flight durations. Thirdly, the recovery period marks the start and end of an individual scenario. It is the time in which flights are initially scheduled before disruptions occur. Next, the number and probability of unavailability periods. The unavailability periods can be subject to different probabilities of occurrence, p. If $p == 1$, the unavailability is certain, if $p = 0$ it is certain not to occur, and if $0 < p < 1$, the unavailability is uncertain. The unavailability probabilities evolve stochastically over time. Finally, a conflict arises when a flight overlaps with an aircraft's unavailability.

Formulating the ARP as a RL model is extremely challenging as it means translating a complex environment and all its information into a state representation, an action space, and a reward function that together enable the agent to learn sequential decision making. The goal of the model is to resolve conflicts and thus maintain operational feasibility. For this work, a typical scenario consists of a schedule with stochastic and/or deterministic disruptions, 3 AC, and up to 17 flights per AC. One such scenario is shown in Figure 2.1. Certain breakdowns are shown in blue, and uncertain breakdowns are green. For the latter the probability of occurrence is indicated. For example in Figure 2.1 the probability of AC 2's unavailability period actually occurring is initially 13%.

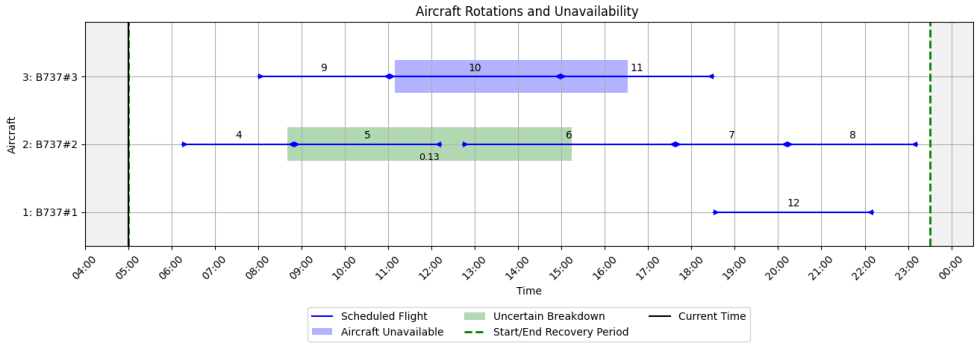


Figure 2.1: Typical Schedule

3 Literature Review

This chapter aims to present relevant state-of-the-art work in- or related to the research field of airline disruption management (ADM). Section 3.1 will focus on the cause and mitigation of flight disruptions. Section 3.2 will dive into the ARP and its solution methods. Section 3.3 highlights the works that incorporate reinforcement learning. Finally, Section 3.4 identifies the research gaps.

3.1 Disruptions in Aviation

Uncertainty in airline planning can have a significant ripple effect on logistics and operations. Already a small delay in the initial schedule can lead to serious consequences down the road. Hence, in the event of disruptions, ADM aims to successfully recover flight schedules. It primarily deals with aircraft, crew, and passengers, leading to the establishment of three problems with increasing complexity: aircraft, aircraft and crew, and integrated recovery.

Disruptions have many causes as identified by Kohl et al. (2007), including airport congestion, mechanical or maintenance problems, and weather events. Furthermore, temporary airport closures can noticeably impact punctuality, and taxiing in and out of the runway can create congestions that introduce delays into the schedule (Hondet et al. 2018). In order to prevent overlaps between flights and an aircraft's unavailability period, Peng et al. (2024) have identified four possible recovery actions: flight cancellations, flight delays, creating a flight, and reserving an aircraft. Vink et al. (2020) add cruise speed control to the list and Hu et al. (2015) also consider tail swaps and passenger itinerary changes. Papers on crew recovery additionally propose crew dead-heading, crew swaps and reserve crew. In real life operations, flight delays and tail swaps are the preferred choices as they are the most cost-friendly.

3.2 Aircraft Recovery Problem

The likes of Clausen et al. (2010) and Hassan et al. (2021) have collected and summarised all relevant works tackling the three ADM problems. As aircraft are the most constraining resource, the majority of published articles before 2009 solely dealt with the ARP (Hassan et al. 2021). Leaving out crew and passengers resulted in a simpler problem to solve. Throughout the years, the efforts around the ARP have shifted towards increasing its complexity while still keeping the problem smaller than e.g the integrated recovery.

The ARP was first introduced by Thengvall et al. (2000) who used a simple heuristic to solve it. Several ways exist to present the problem. On the one hand, it can be modelled as a network routing problem, either consisting of an arc based approach as employed by Su et al. (2021), or a path based approach as used by Eggenberg et al. (2010), Wu et al. (2017), and Liang et al. (2018) where aircraft are assigned to a route instead of a single flight leg. On the other hand, Becking (2024) and Vos (2024) use environment driven simulations and function approximations, which this paper will also consider.

The strive for optimality has led to the use of exact optimisation as a solution method. The advantage of an exact method (EM) is that it can guarantee optimal minimisation in an airline's cost or total delay, depending on the objective. Popular choices are dynamic programming, integer programming, and LP based algorithms (Wu et al. 2017; Peng et al. 2024). Aktürk et al. (2014) for example uses a conic quadratic mixed integer programming (MIP) method to represent the ARP. It solves the problem for a case of 60 aircraft and 207 flights with a total time of three

minutes. Over a decade later, (Peng et al. 2024) has chosen a more advanced decomposition technique over a simple MIP, showing the improvement over the years. They divide the ARP master problem of 44 aircraft and 633 flights into sub problems and solve it using a Logic-Based Benders Decomposition (LBBD). However, due to the larger fleet size, it took around 5 minutes to solve.

As the problem size becomes larger, exact methods have struggled with increasing computing time. Hence, in order to tackle NP hard problems, hybrid- or meta-heuristics are preferred as these are able to provide the required computing power while yielding satisfactory results. This furthermore highlights that suboptimal solutions are tolerated for faster CPU times. The dominant choice in literature are hybrid heuristics (HH). This is underlined by Table 3.1, which provides a detailed overview of academic works covering the ARP. Each row depicts a specific paper and the columns show the solution method, the number of AC, the number of flights, the computation time and the type of recovery actions. HH methods also include Matheuristics (a combination of machine learning (ML) and EM) and SimHeuristics (a combination of simulation and optimisation). The reason for using HH is that it has both, the speed of heuristics and the precision of an exact solver. Earlier research, such as Eggenberg et al. (2010) and Liang et al. (2018), focussed on Column generation (CG) to solve the ARP. Recent, more modern approaches, especially during the last five years, have massively settled for Matheuristics. Here, ML is employed to trim the search spaces after which an exact solver takes over. A standout example would be Rashedi et al. (2024)'s use of ML in combination with a MIP, which successfully allows them to tackle a large problem of up to 1000 flights in just 166 seconds. This large scale handling of HH makes it the current industry standard.

3.3 DRL for ADM

Next to exact optimisation and heuristic based solution methods, DRL has emerged as a viable solution strategy fuelled by advancements in deep learning. The rapid evolving intersection between ADM and RL has resulted in it being either used as a stand alone solution method, or as a heuristic to guide a more traditional approach of using integer programming solvers. A significant advantage of RL is that it provides instant decision making capabilities, sometimes less than a few seconds.

Although reinforcement learning is employed in a number of works, the total research into DRL for the ADM remains scarce. This is highlighted by Figure 3.1, which shows the results of a SCOPUS search specifically focussed on RL or DRL solution methods for the ADM. The detailed query can be found in Appendix B. Regarding RL for the ARP, its application is limited to smaller scale problems. Although Hondet et al. (2018) prove their Q-learning concept, only a small amount of 15 aircraft were used. Lee et al. (2022) also focussed on relatively small numbers, namely 20 and 15 aircraft. Instead, complex DRL is more often applied to the integrated recovery such as by Ding et al. (2023) and Wang et al. (2025).

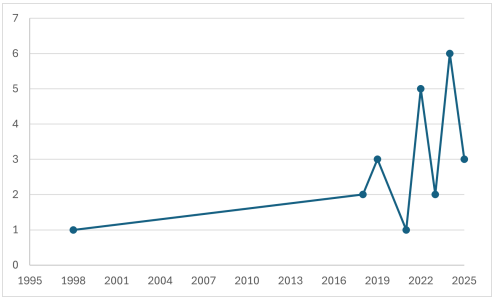


Figure 3.1: Published papers for the SCOPUS query with a focus on RL

DRL for ADM also comes with dedicated state space formulations, s_t. Hondet et al. (2018) investigate the ARP using standard Q-learning and depicts the state as a fixed-length vector, including extensive flight information such as departure and arrival times as well as leg duration and swap status. It mainly uses continuous values. Whereas Hondet et al. (2018) focus primarily on aircraft swapping, Lee et al. (2022) broaden the field by analysing multi-fleet cases, using double Q-learning (DQL). In their paper, Ding et al. (2023) expand to the integrated recovery problem, formulating aircraft, passengers, and crew recovery as a markov decision process (MDP). Here, a proximal policy optimisation (PPO) RL algorithm is integrated in a variable neighbourhood search (VNS) framework, where s_t consists of a composite vector telling the VNS algorithm what to do. It combines flight information such as departure and arrival times, with previous assignment status i.e. which aircraft or crew members are assigned

where. Furthermore, Żurek et al. (2024) use a combination of several RL strategies to tackle disruptions, where the state is a combined vector of global and local properties. Wang et al. (2025) also investigate the integrated recovery problem but, unlike Ding, use attention based DRL and formulate the state as a high dimensional space-time network. Here each state is made up of a set of features for each flight leg, such as slack time, and availability of resources. In contrast to Wang, Rashedi et al. (2024) do not use ML as the final solver but rather as a means for column selection. Here, the state is thus formulated in such a way that it helps select which column are most likely part of the optimal solution.

Table 3.1: Comparison of Aircraft Recovery Problem (ARP) approaches

Paper	Solution Method	Time	Fleets	AC	Flights	Recovery Actions
Eggenberg et al. 2010	**HH**: CG & DP	63 s	1	100	760	Cancel, Delay, Swap
Aktürk et al. (2014)	**EX**: Conic Quadratic MIP	40–196 s	6	60	207	Speed, Delay, Swap
Vos et al. (2015)	**HH**: Rolling Horizon Heuristic & LP	$<$ 600 s	Multi	43	Full Day	Cancel, Delay, Swap
Xu et al. (2016)	**HH**: Weighted Time-Band Approx.	$<$ 5 s	1	238	596	Cancel, Delay, Swap
Wu et al. 2017	**HH**: Distr. Fixed-Point & IP	287–6k s	1	27	162	Cancel, Delay, Swap
Hu et al. (2017)	**HH**: ϵ-constraint & Neighborhood Search	4–57 s	1	320	612	Cancel, Delay, Swap
Zhang (2017)	**HH**: Two-Stage Heuristic	420–540 s	1	127	494	Cancel, Delay, Swap
Hondet et al. (2018)	**RL**: Q-Learning	$<$ 1 s	1	15	72	Swap, Delay
Liang et al. (2018)	**HH**: CG with Diving	315–547 s	1	48	174	Cancel, Delay, Swap, Reserve AC
Hassan (2019)	**HH**: ML (Random Forest) + ILP	48 s	4	827	2200	Cancel, Delay, Swap, Conn. Delay
Lee et al. (2020)	**HH**: Stochastic IP (SAA)	180–300 s	3	NA	900	Cancel, Delay, Swap, Speed
Rhodes-Leader et al. (2021)	**HH**: SimHeuristic (IP + Simulation)	60–120 s	1	50	192	Delay, Swap, Re-timing
Huang et al. (2022)	**HH**: Iterative Copy Generation	10–300 s	Multi	157	592	Cancel, Delay, Swap
Lee et al. (2022)	**RL**: Deep Double Q-Learning	47–117 s	3	20	94	Delay, Swap
Zhao et al. (2023)	**HH**: Two-Stage Heuristic	6–570 s	4	84	608	Cancel, Delay, Swap
Haider et al. (2024)	**HH**: ML (LGB) + CG	1–63 s	Multi	50	166	Cancel, Delay, Swap
Peng et al. (2024)	**EX**: Logic-Based Benders (LBBD)	339 s	1	44	633	Cancel, Delay, Swap
Zang et al. (2024)	**HH**: DDBA (Integer LP)	0.3s–6h	Multi	560	1926	Cancel, Delay, Swap
Żurek et al. (2024)	**RL**: Ensemble (DDQL & A2C)	1–2 s	1	15	105	Delay, Swap
Rashedi et al. (2025)	**HH**: ML (Random Forest) + MIP	83–166 s	Multi	220	1000	Cancel, Delay, Swap
Jiang et al. (2025)	**HH**: ADMM & CG	1–807 s	1	61	353	Cancel, Delay, Swap
Lu et al. (2025)	**HH**: RL-initialized CG	12–45k s	1	44	611	Cancel, Delay, Swap

***Key: EX**: Exact Method; **HH**: Hybrid Heuristic (includes Matheuristics & SimHeuristics); **RL**: Reinforcement Learning.*
***Abbreviations: AC**: Aircraft; **CG**: Column Generation; **DP**: Dynamic Programming; **MIP**: Mixed Integer Programming; **LGB**: Light Gradient Boosting; **SAA**: Sample Average Approximation; **LBBD**: Logic-Based Benders Decomposition; **DDBA**: Decision-Decomposition-Based Algorithm; **DDQL**: Deep Double Q-Learning; **A2C**: Advantage Actor-Critic; **ADMM**: Alternating Direction Method of Multipliers.*

3.4 Research Gaps

To begin with, Figure 3.1 underlines the limited amount of research into DRL for the ARP. This constitutes the most significant gap and is the main motivation for this work's focus on DRL as a solution method.

Moreover, a further identifiable gap is that the previously mentioned works consider a dense state with continuous observations. Or, in the case of Ding et al. (2023), rather guides a different algorithm. Hence this paper aims to investigate the effects of a sparser, dual matrix formulation with a primarily binary structure on a DRL agent. This helps to depict the schedule into a binary grid, making it simpler for the DRL agent to identify patterns and thus increasing computational efficiency. Furthermore, by including stacked observations allowing for backwards looking properties, this work will make use of the rate of change of disruption probabilities, a feature missing from Lee et al. (2022) and Hondet et al. (2018).

Another gap is the scarcity of proactive conflict resolution. Only a few identified works implement probabilistic modelling to anticipate future conflicts. The first one, by Lee et al. (2020) develops their Stochastic Reactive and Proactive Disruption Management (SRPDM) framework which anticipates delays. Next, Becking (2024) implements a proactive DRL approach that investigates four strategies: a reactive heuristic, a reactive DRL agent, a proactive DRL agent without stochastic indications of possible future disruptions, and finally an all-knowing proactive DRL agent able to see the stochastic evolution. Here, DRL is the clear winner in terms of speed and scalability. Thirdly, Zang et al. (2024) propose a proactive DDBA integer LP approach. However, as this is an exact method, it is limited by expansive computation times. Lastly, Vos (2024) introduces a model-based RL approach formulated as a MDP and uses fixed uncertainty modelling. As proactive conflict resolution better mirrors a real-life necessity, this paper, similar to Becking (2024), will use state masking in order to compare proactive and reactive strategies.

Furthermore, another key improvement is the investigation into alternate reward function designs. As in current state-of-the-art research, the primary objective of minimising costs and cancellations remains for this paper. However, this work aims to include risk mitigation. This is done by focussing on proactive risk taking, scaling the rewards relative to the probability of occurrence of a disruption. In comparison, Becking (2024) penalises proactive behaviour by a constant penalty. Furthermore, while papers such as Hondet et al. (2018) and Lee et al. (2022) focus on immediate resolution, this work allows mistakes and penalises a final state.

Thus, to conclude, this work will perform a detailed comparison and analysis of reactive and proactive DRL strategies while considering alternate state space formulations and an elaborate reward function design.

3.4.1 Research Objective and Research Questions

Following the discussion on the identifiable research gaps, a primary research objective can be formulated:

Primary Objective: *Optimising a reinforcement learning formulation for the aircraft recovery problem with the aim of minimising disruption effects.*

Primary Research Question: *How do different state space formulations and the reward function design affect the results of a proactive deep reinforcement learning framework for the aircraft recovery problem?*

4 Methodology

In this chapter, the methodology will be detailed. First, Section 4.1 offers a comprehensive outline on the Deep Q-Network (DQN) algorithm, focussing on its foundation, Q-learning, and neural networks. Next, Section 4.2 describes the mathematical formulation, including the the state space and reward design. Furthermore, Section 4.3 details the network architecture. Lastly, the training procedure and data flow are explained in Section 4.4. Here, the use-case data set is also described.

4.1 Deep Q-Network (DQN) Algorithm

To begin with, Section 4.1.1 provides a salient foundation on Q-learning. Further, Section 4.1.2 goes into Neural Networks focussing on loss functions, experience replay, and the epsilon-greedy policy.

4.1.1 Q-Learning Foundation

In order to understand deep Q-network RL, the inner workings of Q-learning must first be explained, being the foundation upon which the DQN algorithm functions. In Q-learning an agent learns good decisions by interacting with the environment. It is a model free algorithm meaning that learning comes solely from experiences. Key components are Q-values, Q-function, optimal policies, and the Bellman equation.

Q-values are the expected rewards for a certain action taken in a specific state, and are the output of the Q-function depicted in Equation 4.1. The Q-function is also known as the action-value function. Here, $Q^{\pi}(s,a)$ represents the expected cumulative discounted reward for choosing action a in state s at time t, and taking policy π. $\gamma \in [0,1]$ is the discount factor, T is the episode length, $R(s_{t+k}, a_{t+k}, s_{t+k+1})$ is the reward obtained at a future time step k, and $\mathbb{E}_{\pi}$ is the expectation under policy π.

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{T} \gamma^{k} R(s_{t+k}, a_{t+k}, s_{t+k+1}) \middle| s_t = s, a_t = a\right] \tag{4.1}$$

The optimal Q-function, $Q^{*}(s,a)$ is the maximum expected return achievable by any policy and is depicted in

Equation 4.2. The corresponding optimal policy, π^*, which aims to choose that action from the action space $\mathscr{A}$ that maximises the Q-value, is shown in Equation 4.3. The optimal Q-function satisfies Equation 4.4, the Bellman optimality equation, which outlays a recursive relationship in a Markov decision process(MDP). Here, the optimal Q-value is equal to the immediate reward $R(s,a,s')$ plus the discounted optimal Q-value of the next state s'. Q-learning is thus a model-free, off-policy temporal difference (TD) learning algorithm that uses the Bellman equation to iteratively estimate $Q^*(s,a)$.

$$Q^*(s,a) = \max{}_{\pi} Q^{\pi}(s,a) \tag{4.2}$$

$$\pi^*(s) = \underset{a \in \mathscr{A}}{\operatorname{argmax}}\, Q^*(s,a) \tag{4.3}$$

$$Q^*(s,a) = \mathbb{E}\left[R(s,a,s') + \gamma \max{}_{a' \in \mathscr{A}} Q^*(s',a')\right] \tag{4.4}$$

4.1.2 Neural Networks

In problems with very large state space sizes, Q-tables are infeasible and thus function approximation is used. In the case of DQN RL, a neural network (NN), $Q(s,a;\theta)$, is used, where Q is the action-value function, s is the input state or observation vector, a is the action, and θ represents the parameters, which are trainable weights, of the network. The output are the Q-values for all actions. Neural Networks have 4 main advantages: generalisations, scalability, feature learning, and continuous states. Generalisation means that similar states have similar Q-values. Scalability means being able to handle large state spaces, and feature learning means that the neural network automatically learns useful state representations. Finally, NNs are able to work with continuous and high-dimensional state spaces.

Loss Function

The loss function plays an important role in neural networks. It minimises the mean squared error between the predicted Q-values and the target ones. It is depicted in Equation 4.5. $L(\theta)$ is the loss with respect to parameters θ, $\mathbb{E}$ is the expectation calculated over a batch of transitions (s,a,r,s') sampled from the experience replay buffer $\mathscr{D}$. The term $\left(r + \gamma \max\ a' Q(s',a';\theta^-)\right)$ denotes the target Q-value, where θ^- is the target network, and $r = R(s,a,s')$ is the immediate reward.

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathscr{D}}\left[\left(r + \gamma \max\ a' Q(s',a';\theta^-) - Q(s,a;\theta)\right)^2\right] \tag{4.5}$$

The target network is used to provide stable Q-value estimates. Whereas the main network, θ, updates its Q-values, $Q(s,a;\theta)$, every training step with the help of gradient descent, the target network, θ^- keeps a frozen network copy and updates its parameters every so many steps. The inner workings are handled automatically by the Stable-Baseline3 algorithms. For this model, the implementation uses hard updates (Equation 4.6). A target network is useful as it is more likely to converge to the optimal Q-function. Although it can be slower in training, it is thus more stable.

$$\theta^- \leftarrow \theta \tag{4.6}$$

Gradient Descent Update

Gradient descent is used to update network parameters. The theoretical formulation of the gradient involves the loss function and can be seen in Equation 4.7, where $\nabla_\theta L(\theta)$ is the gradient of the loss with respect to weights θ, and $\nabla_\theta Q(s,a;\theta)$ is the gradient of the Q-function.

Here, stochastic gradient descent is used and depicted in Equation 4.8 and Equation 4.9. α is the learning rate, $\mathscr{B}$ represents a batch of experiences sampled from $\mathscr{D}$, and $|\mathscr{B}|$ is the size of that batch.

$$\nabla_\theta L(\theta) = -\mathbb{E}_{(s,a,r,s')\sim\mathscr{D}}\left[\left(r + \gamma \max\ a' Q(s',a';\theta^-) - Q(s,a;\theta)\right)\nabla_\theta Q(s,a;\theta)\right] \tag{4.7}$$

$$\theta \leftarrow \theta - \alpha \nabla_\theta L(\theta) \tag{4.8}$$

$$\theta \leftarrow \theta + \alpha \frac{1}{|\mathscr{B}|} \sum_{(s,a,r,s')\in\mathscr{B}} \left(r + \gamma \max\ a' Q(s',a';\theta^-) - Q(s,a;\theta)\right)\nabla_\theta Q(s,a;\theta) \tag{4.9}$$

where $\mathscr{B}$ is a batch of experiences sampled from the replay buffer,

Experience Replay & Sampling
DQN importantly stores past experiences in a replay buffer, $\mathscr{D}$, from which it then samples a random batch, $\mathscr{B}$, for training. For this model, the replay buffer size is set to $|\mathscr{D}| = 100,000$. It is presented as a tuples, $(s_t, a_t, r_t, s_{t+1}, d_t)$, where s_t is the current state, a_t is the action taken, r_t is the reward received, s_{t+1} is the next state, and d_t is the done flag indicating whether an episode terminated).

The sampling batch, $\mathscr{B}$, has a size of $|\mathscr{B}| = 256$ experiences. As mentioned, sampling happens randomly, which is good as is ensures more stable learning from diverse experiences and is more efficient. Finally, the loss function Equation 4.5 can be rewritten to mirror the implemented version of this model by including the experience replay buffer and the conditional 'done flag':

$$L(\theta) = \frac{1}{|\mathscr{B}|} \sum_{(s,a,r,s',d)\in\mathscr{B}} (y - Q(s,a;\theta))^2 \tag{4.10}$$

$$\text{where } y = \begin{cases} r & \text{if } d = \text{true (terminal state)} \\ r + \gamma \max\ a' Q(s', a'; \theta^-) & \text{if } d = \text{false} \end{cases} \tag{4.11}$$

Epsilon-Greedy
During training, the agent faces a difficult trade-off: exploitation vs exploration. Exploitation means using the best current policy, thus choosing actions with the highest Q-values. Exploration tries using new actions in order to potentially find better policies. It is best to use a well established portion of both as using only exploitation could lead to being stuck in a local optima with suboptimal policies, and using only exploration wastes valuable time on mediocre actions.

The epsilon greedy policy, π_ϵ, used for this model is:

$$\pi_\epsilon(s) = \begin{cases} \text{a random action} & \text{with probability } \epsilon \\ \underset{a\in\mathscr{A}}{\text{argmax}}\, Q(s,a;\theta) & \text{with probability } 1-\epsilon \end{cases} \tag{4.12}$$

where $\epsilon \in [0,1]$ is the exploration rate. With probability ϵ, the model explores and chooses a random action, whereas with probability $1-\epsilon$ it exploits. For this model, the exploration decreases exponentially over time with a decay rate of $\epsilon_{\text{decay}})^t$. The minimum exploration rate, ϵ_{min} is reached when $t = t_{decay\_end}$, which differs per case. Early training has a high epsilon, starting with $\epsilon_{start} = 1$, and later training, eventually reaches a minimum exploration rate of $\epsilon_{min} = 0.05$.

$$\epsilon(t) = \begin{cases} \epsilon_{\text{start}} \times (\epsilon_{\text{decay}})^t & \text{if } t < t_{\text{decay\_end}} \\ \epsilon_{\text{min}} & \text{if } t \geq t_{\text{decay\_end}} \end{cases}$$

4.2 Mathematical Model Formulation

This section describes the mathematical formulation of the model. It does so by first focussing on the state-space formulation in Section 4.2.1. Next, the decision making actions are discussed in Section 4.2.3. Section 4.2.4 explains the probability evolution and finally Section 4.2.5 details the reward function design.

4.2.1 State Space Design

The comparative analysis focusses on two distinct state space designs, referred to as Model 1 and Model 2 hereafter. The state space S represents the complete information available to the agent at each timestep. Model 1 uses a vector-based state representation from a structured flattened matrix and includes a stacked observation space with temporal features. Model 2 heavily prioritises a binary state space representation i.e. 1s and 0s and consists of two separate matrices.

Model 1: Compact Aircraft-Centric State Space
The state space is represented as a matrix $S_1 \in \mathbb{R}^{R\times C}$ depicted in Equation 4.13, where R and C are the number of rows and columns of the state respectively. $R = 1 + N_{ac}$, where N_{ac} is the number of AC. Here, $N_{ac} = 3$ and thus Equation 4.13 has 4 rows. Row 0 is a header row containing the current time t, the time till the end of the run t_{end},

and the received penalties $reward_i$, which will be discussed in Section 4.2.5. The penalties are updated after each step.

The column structure can be written as $C = 3 + 2 + 4 \times N_{fl}$. The first three columns are the probability features of the uncertainty periods, where p_i is the probability that AC i is unavailable from time $p_{i,start}$ to time $p_{i,end}$. The next 2 columns depict conflict counts i.e. c_{init} are the conflicts present at timestep 0 and c_{curr} are the conflicts still present at the current timestep, t. The remaining columns detail flight information with 4 features per flight. The state space accommodates up to N_{fl} number of flights per AC and thus per row. The four features are the flight ID f_i, the flight departure time $f_{i,dep}$, the flight arrival times $f_{i,arr}$, and a binary conflict count $f_{i,conf}$ showing 1 if the flight is still conflicted and 0 otherwise. Here, N_{fl} equals 17. Hence, the matrix has a shape of R x C = 4 x (3 + 2 + 4 x 17) = 4 x 73 = 292 elements.

$$S_1 = \begin{bmatrix} t & t_{\text{end}} & reward_1 & \cdots & \cdots & \cdots & \cdots & reward_i & & \\ p_1 & p_{1,start} & p_{1,end} & c_{\text{init}} & c_{\text{curr}} & f_i & f_{i,dep} & f_{i,arr} & f_{i,conf} & \cdots \\ p_2 & p_{2,start} & p_{2,end} & c_{\text{init}} & c_{\text{curr}} & f_i & f_{i,dep} & f_{i,arr} & f_{i,conf} & \cdots \\ p_3 & p_{3,start} & p_{3,end} & c_{\text{init}} & c_{\text{curr}} & f_i & f_{i,dep} & f_{i,arr} & f_{i,conf} & \cdots \end{bmatrix} \tag{4.13}$$

Model1 possesses both temporal features and observation stacking. The purpose of temporal features is to give the agent extra context on when disruptions will occur. Furthermore, it's purpose is to track the probability evolution and see whether values are increasing. In total there are 4 features: time remaining until start of the unavailability period, the time remaining until its end, the probability slope considering their values in previous states, and the normalised disruption start time. Since there are 3 aircraft the temporal feature vector, T, is: $\mathbb{R}^{3\times4} = \mathbb{R}^{12}$. The base state vector for one time step is the combination of the flattened state and the temporal feature vector, shown in Equation 4.14.

$$\mathbf{s}_{base} = [\text{flattened}(S_1), \mathbf{T}] \in \mathbb{R}^{R\times C+12} = \mathbb{R}^{4\times73+12} = \mathbb{R}^{304} \tag{4.14}$$

Observation stacking gives the agent limited memory by concatenating, in this case, $K = 2$ consecutive states. Now, the DRL agent can use information from the temporal features to make informed decisions with respect to the probability evolution of the disruptions. The total dimensions, D, are shown in Equation 4.15.

$$\begin{aligned} D &= \text{length } s_{base} \cdot K \\ \leftrightarrow D &= 608 \end{aligned} \tag{4.15}$$

Equation 4.16 shows the observation space, o_t, i.e. what the agent sees at each time step. It is a single flattened vector of D dimensions. $\mathbf{s}_{t-1}$ is the state vector from one time step ago (first 304 dimensions) and $\mathbf{s}_t$ is the current state vector (latter 304 dimensions). It is the DQN's input layer and is fed directly into the neural network.

$$\mathbf{o}_t = [\mathbf{s}_{t-1}, \mathbf{s}_t] \in \mathbb{R}^D \tag{4.16}$$

Model 2: Binary focussed dual State Space

In comparison, whereas Model 1 uses continuous entries, Model 2 heavily prioritises a binary state space representation i.e. 1s and 0s. Furthermore, the state space S_2 of Model 2 consists of two matrices. First, the Aircraft Unavailability Matrix, AC_{mtx} and second, the Flight Matrix, FL_{mtx}. Similarly to Model 1, N_{ac} = 3, and N_{fl} = 17 in order for the models to be comparable.

Equation 4.17 pictures the Aircraft Unavailability Matrix. It shows the evolution of the disruption probabilities over time, where $p_{i,j}$ is the unavailability probability for aircraft i in time interval j. Each row corresponds to one AC and the columns each represent a 60 minute time interval spanning 24 hours (T = 24). If p = 1 the disruption is certainly happening, if p = 0 no disruption exists, and 0 < p < 1 means that the probability has not been resolved yet. The size of the AC_{mtx} is thus $AC_{mtx} \in \mathbb{R}^{N_{ac}\cdot T} = \mathbb{R}^{3x24=72}$.

$$AC_{mtx} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & \cdots & p_{1,T} \\ p_{2,1} & p_{2,2} & \cdots & \cdots & p_{2,T} \\ p_{3,1} & p_{3,2} & \cdots & \cdots & p_{3,T} \end{bmatrix} \tag{4.17}$$

Equation 4.18 shows the Flight Matrix. Each row corresponds to a flight, resulting in $R = N_{ac} \times N_{fl} = 51$ rows. The first column houses the AC IDs of said flight and the remaining columns, similarly to the AC_{mtx}, represent the

number of time intervals, T. The matrix is filled with 1s if a flight is flying and 0s if the flight is on the ground. Thus the size of the flight matrix is $FL_{mtx} \in \mathbb{R}^{R\times(T+1)} = \mathbb{R}^{51x25=1275}$.

$$FL_{mtx} = \begin{bmatrix} ac_1 & o_{1,1} & o_{1,2} & \cdots & o_{1,T} \\ ac_2 & o_{2,1} & o_{2,2} & \cdots & o_{2,T} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ ac_{N_{fl}} & o_{N_{fl},1} & o_{N_{fl},2} & \cdots & o_{N_{fl},T} \end{bmatrix} \tag{4.18}$$

$o_{i,j}$ is the binary occupancy indicator for flight i in time interval j. If $o_{i,j} = 1$, flight i occupies interval j. Otherwise $o_{i,j} = 0$. The final observation at time step t, s_t, is obtained by flattening and concatenating both matrices as seen in Equation 4.19. The dimension, $D_3 = |AC_{mtx}| + |FL_{mtx}| = 1347$.

$$\mathbf{s}_t = [\text{flatten}(AC_{mtx}), \text{flatten}(FL_{mtx})] \in \mathbb{R}^{D_3} \tag{4.19}$$

Model Comparison
Table 4.1 shows the comparative differences between Models 1 and 2.

Table 4.1: State Space Comparison Between Model 1 and Model 3

Property	**Model 1**	**Model 2**
Representation	Dense matrix	Sparse binary matrices
Organization	Aircraft-centric	Flight-centric
Observation Space	608	1347 (2.2x larger)
Temporal Encoding	Explicit temporal features + stacking	Implicit in time-interval structure
Sparsity	Dense (most entries filled)	Sparse (binary, mostly zeros)
Conflict Encoding	Explicit conflict counts	Implicit via matrix overlap

4.2.2 Environment-Dependent State Masking

Both models consider three different environment types: proactive, myopic, and reactive. This affects the information available to the agent.

The proactive strategy allows the agent to be 'all knowing'. It sees all the unavailability durations and probabilities at every time step, no matter if they are 1, 0 or in between. The proactive environment represents a system with perfect information and long-term planning capabilities. The agent can make proactive decisions based on all available probabilistic information. The myopic strategy on the other hand can also see in advance but solely those unavailability periods that are certain (p = 1), ignoring probabilistic information. It thus only reacts to guaranteed conflicts but cannot act on uncertain information and must wait until the probability resolves. Finally, the reactive strategy is the most restrictive as it is limited to seeing imminent unavailability periods with p = 1 whose start time is within one time step of the current time. No advance planning is possible in contrast to the other two environment types and conflicts can only be tackled as they become imminent. All three environment types are shown in Table 4.2.

Consequentially, the observed state $\mathbf{o}_t$ is different from the actual state $\mathbf{s}_t$ depending on the environment types. This is achieved by masking. Model 1 can when necessary mask the probability, the probability start time, and the probability end time in S_1. Model 2 can mask the probabilities in AC_{mtx}. No masking occurs for the proactive environment. For the myopic environment all probabilities that have not yet resolved to 1 or 0 are masked. Similarly, for the reactive environment, all disruptions that have not yet resolved to 1 or 0 and whose start times exceed one time step ahead of the current time are masked.

Env Type	**Prob. Visibility**	**Timing Visibility**
Proactive	All $p \in [0, 1]$	Full horizon
Myopic	Only $p = 1.0$	Full horizon
Reactive	Only $p = 1.0$	Imminent ($t_{\text{start}} \leq t + \Delta t$)

Table 4.2: Environment types and their characteristics

4.2.3 Decision Making

Action Choices
There are three primary actions, a_t, that can be taken at time t by the decision maker, here the DRL agent. One of these actions must be taken at each time step. They are:

1. **Wait (No Action):** $a_t = (0, 0)$
   - Time advances while no changes are made to the scenario
   - Even if conflicts exist this action can be chosen
2. **Cancel Flight:** $a_t = (f_i, 0)$
   - removes flight f_i from the schedule
   - once cancelled, cannot reappear
3. **Move Flight:** $a_t = (f_i, ac_k)$
   - **Tail Swap**: flight f_i moved to different aircraft ac_k
   - **Tail Remain**: flight f_i remains on the same aircraft
   - **Delay**: tail swaps and tail remain may require delay if aircraft ac_k has other flights scheduled or conflicts exist

Action Space
Both models use a discrete action space $\mathscr{A}$ with size:

$$|\mathscr{A}| = (N_{ac} + 1) \times (N_{fl,max} + 1) \tag{4.20}$$

The total possible actions in the action space are combinations for the 'move flight: $a_t = (f_i, ac_k)$' action, plus the inaction (0,0) and the cancel flight $(f_i, 0)$ action. Each action $a \in \mathscr{A}$ is a tuple (f, ac) where $f \in \{0, 1, 2, \dots, N_{fl,max}\}$ is the flight ID and $ac \in \{0, 1, 2, \dots, N_{ac}\}$ is the AC ID. With three aircraft and seventeen flights, $|\mathscr{A}| = 72$. The action space is combined with a dynamic action mask whose purpose it is to filter out invalid actions. These consist of flights that have already departed, aircraft that are not available, or environment specific restriction such as, for example, the reactive environment only allowing actions when conflicts are imminent. It outputs a binary vector where 1 means that the action can be chosen and 0 means that the Q-value is set to minus infinity, making the action not valid. The action mask changes from step to step as the state is updated.

4.2.4 Probability Evolution

The probability of the unavailability periods evolve stochastically over time. At each time step, t, the probability p_i for aircraft i is updated until it is resolved to either 1 or 0 after which it remains unchangeable. Resolution occurs when the current time reaches or exceeds the disruption start time. If $p_i = 1$ the unavailability period materializes and if a flight overlaps, the conflict is certain. If $p_i = 0$, the unavailability period resolves and the conflict for any overlapping flight is removed.

The update mechanism is shown in Equation 4.21. α and β are environment dependent transition probabilities. Δp_i consist of a random choice from $[-0.05, 0.05]$ and a bias term. When $p_i > 0.5$, the bias $0.05 \times (1 - p_i)$ pushes the probability upward toward 1.0, while when $p_i \leq 0.5$, the bias $-0.05 \times p_i$ pushes the probability downward toward 0.0. This bias mechanism ensures that the unavailability periods' probabilities do not remain stagnant but rather eventually resolve. This makes sure that uncertainties will reach a definitive end value. The probabilities are constraint to the interval $[0.05, 0.95]$ before final resolution, preventing premature ending while keeping the uncertainty.

$$p_i(t+1) = \begin{cases} p_i(t) + \Delta p_i & \text{with probability } \alpha \\ p_i(t) - \Delta p_i & \text{with probability } \beta \\ p_i(t) & \text{with probability } 1 - \alpha - \beta \end{cases} \tag{4.21}$$

4.2.5 Reward Formulation

The reward function denotes the immediate rewards $R(s, a, s')$. It guides the DRL agent's choices and indicates which actions are favourable and which are not. It follows a mainly negative structure with one positive bonus and is depicted by Equation 4.22. A local reward is given at each single time step i.e after each action. The total

rewards per scenario and finally per episode are calculated by summing them up. All rewards are normalised with respect to each other. Despite different state-space formulations, both Model 1 and 2 have the same reward design.

$$R(s, a, s') = -R_{\text{delay}} - R_{\text{cancel}} - R_{\text{inaction}} - R_{\text{auto}} - R_{\text{time}} - R_{\text{unresolved}} + B_{\text{resolve}} \tag{4.22}$$

The overall objective function, $J(\pi)$, can be written as in Equation 4.23. It depicts the expected long term performance of following policy π. The expectation $\mathbb{E}_{\tau\sim\pi}$ is the average of the cumulative reward ($\sum_{t=0}^{T} \gamma^t R(s_t, a_t, s_{t+1})$) across all possible trajectories τ. It quantifies the importance of future rewards relative to immediate rewards. T is the episode length and varies per scenario.

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t, s_{t+1})\right] \tag{4.23}$$

R has seven components in total which can be enabled or disabled to see their individual effects. First, the **delay** component (R_{delay}) which is only given if the delay exceeds 300 minutes or 6 hours. Anything below this threshold and the delay is accepted. In real life applications a delay of 3h is acceptable for passengers. However, to encourage move actions, this was increased to 300min. Secondly, the **manual cancellation** penalty (R_{cancel}) is given when the 'cancel flight' action is chosen. It is the second worst penalty as the agent should be guided towards 'move' actions rather than cancellations or inactions as these do not solve conflicts efficiently. Similarly **inactions** are also penalised by R_{inaction}. However, if no conflicts remain and the environment simply needs to wait in order for the probability to resolve, inactions are tolerated, indicating that waiting is not always bad. The highest penalty is given for **automatic cancellations** (R_{auto}) which occur when the agent does not act in time. The logic behind this is that it is better to act and take a risk than not to act at all. Next, a standard **time** penalty (R_{time}) is always given to encourage proactive behaviour. Furthermore, to extra encourage move actions, a positive **resolution bonus** is given each time a flight is moved away from a disruption, even if its probability of occurrence is not yet 100% certain. The resolution bonus is hence adapted to the probability resulting in a lower reward for low probabilities and a higher reward for increasing ones. Finally, a penalty for **unresolved conflicts** ($R_{\text{unresolved}}$) is given which is a delayed reward. It is not obtained immediately but rather at the end of a scenario for all overlaps that were not successfully resolved. Through the Bellman equation and temporal credit assignment, DQN can handle delay rewards well. The discount factor, γ, controls how far backward the received penalty propagates and how important delayed rewards are compared to immediate ones. Here, a discount factor of 0.995 means that the penalty remains significant for earlier steps. With many steps and long episodes, the unresolved conflict penalty helps, on top of the immediate resolution bonus, to send clear signals that solving conflicts is good.The reward hierarchy is then:

resolving a conflict > wait > manual cancel > auto cancel > delayed unresolved conflict penalty

4.3 Network Architecture

This section outlines the network architecture for Model 1 and Model 2. Both use identical DQN multilayer perceptron (MLP) architectures but differ in their input dimensions due to different state space representations described in Section 4.2.1.

Architecture Overview
Both models solve the same problem with the same action space. The difference is in how the state is encoded but the neural network processing is identical.

The architecture is as follows: an input layer with D dimensions depending on the model type as detailed in Section 4.2.1. Then three hidden layers consisting of hidden layer 1 with 256 neurons, hidden layer 2 with 512 neurons and hidden layer 3 with 256 neurons. Finally, the output layer which outputs Q-values for all actions in the action space, $\mathscr{A}$. The activation function for all hidden layers is the ReLU or Rectified Linear Unit function shown in Equation 4.24. The hidden layer outputs h_1, h_2, h_3 are described by Equation 4.25, Equation 4.26, and Equation 4.27. $\mathbf{W}_i$ are the weight matrices and $\mathbf{b}_i$ the bias vectors.

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{4.24}$$

$$\mathbf{h}_1 = \text{ReLU}(\mathbf{W}_1\mathbf{s} + \mathbf{b}_1) \tag{4.25}$$

$$\mathbf{h}_2 = \text{ReLU}(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2) \tag{4.26}$$

$$\mathbf{h}_3 = \text{ReLU}(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3) \tag{4.27}$$

The complete forward pass can be described by Equation 4.28, where θ represents the network parameters i.e. the weights and biases, and the subscript a denotes the Q-value for action a.

$$Q(s, a; \theta) = \mathbf{q}_a = [\mathbf{W}_4\text{ReLU}\left(\mathbf{W}_3\text{ReLU}\left(\mathbf{W}_2\text{ReLU}\left(\mathbf{W}_1\mathbf{s} + \mathbf{b}_1\right) + \mathbf{b}_2\right) + \mathbf{b}_3\right) + \mathbf{b}_4]_a \tag{4.28}$$

Action Masking and Action Selection Strategy with Epsilon Greedy
Before the action is selected, an action mask is applied to the action space described in Section 4.2.3. This is to prevent invalid actions (acting on departed flights, unavailable aircraft). The environment generates a binary mask, where 1 means the action i is valid, and 0 indicated that it is invalid:

$$Q_{mask}(s, a_i) = \begin{cases} Q(s, a_i) & \text{if } mask[i] = 1 \\ -\infty & \text{if } mask[i] = 0 \end{cases}$$

Masking is dynamic, meaning that it updates each time step based on flight departure status, aircraft availability, environment type restrictions.

The process of actually selecting an action is influenced by the exploration vs exploitation scheme. For this model the epsilon greedy policy in Equation 4.12, can be further expanded upon by perfecting the exploration strategy. Instead of only choosing random actions 100% of the time, the model uses a 40-30-30 exploration distribution across three strategies, ensuring an efficient balance between broad coverage across the action space, focussed action choices, and near-optimal exploration.

1. **40% Pure Random Exploration** Uniform sampling over all valid actions, which ensures broad coverage of action space
2. **30% Conflict-Guided Exploration**: Prioritises actions on conflicted flights to encourage conflict exposure.
3. **30% Q-Noise Exploration**: explores near-optimal actions, reducing the distribution mismatch with exploitation; adds Gaussian noise to Q-values.

4.4 Use Case Implementation

The following section details the training procedure for this ARP using as an algorithm the Deep Q-Network (DQN) with prioritized experience replay, and as a framework the stable-baselines3 DQN implementation. Section 4.4.2 provides an overview of the training process, and Section 4.4.3 provides a visual representation of the data flow per step. Finally, Section 4.4.1 describes the training data set for the experimental set-up of this work.

4.4.1 Training Data Description

The training takes place on a synthetically generated training data. It is specifically designed to tackle the aircraft recovery challenge. Each schedule equals one scenario and will be referred to as such. Each scenario consists of 3 aircraft and up to 17 flights per aircraft. Each scenario is completely unique and the flight times are randomly generated. A scenario is made up of one complete aircraft rotation and its corresponding flight legs. Aircraft unavailability periods also known as disruptions are also included. They can have varying probabilities, which later change across the recovery period. The amount of disruptions varies between 1 and 2 per schedule. Approximately one third of the scenarios have certain disruptions only, one third have stochastic as well as certain ones, and the remaining schedules only have stochastic ones. The controlled design of training data assures precise control over scenario complexity and allows for precise tuning for validation, scaling, or sensitivity analyses. Furthermore, it is crucial for experiment reproducibility and for the implementation and addition of stochastic unavailability periods. Next to the mentioned three aircraft, and seventeen flights, this work considers 182 scenarios.

4.4.2 Training Procedure

The goal of the decision maker, here the deep reinforcement learning (DRL) agent, is to move the overlapping flights away from the unavailability periods by moving them to different AC and/or delaying them. The decision variables are then the possible actions the agent can take at each time step, t, as described in Section 4.2.3.

The total amount of time steps is set in advance. Here, 300000 steps are considered. As mentioned, the data set comprises 182 scenarios. One scenario is tackled after the other and when all 182 have been tackled by the DRL agent, one episode has passed. Thus one episode equals one run-through of all scenarios present in the schedule in the data set. The cycle continues until the maximum amount of time steps has been reached. Figure 4.1 explains the relationship between scenarios and episodes. The model initialises for all environment types (proactive, reactive, myopic) in parallel and considers different seeds during the run to ensure reproducibility and measure variance.

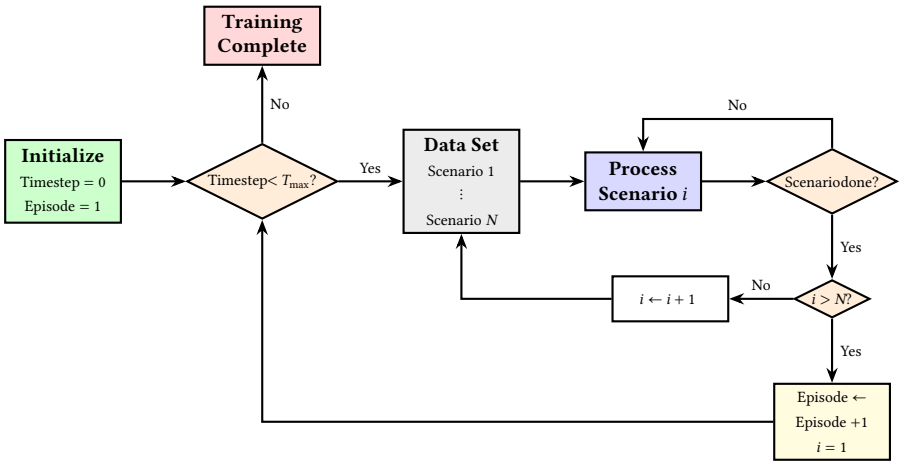


Figure 4.1: DQN training flowchart. The data set contains N scenarios. Each episode processes all scenarios sequentially. After all scenarios are processed, a new episode begins. Training continues until T_{max} timesteps. The replay buffer and network weights persist across episodes.

Per Scenario, decisions are made sequentially at discrete time steps in 1 hour intervals. If the time surpasses the start time of a conflicted flight and the agent fails to act on it, it is automatically cancelled by the environment. Hence actions must be taken before flights depart. The amount of timesteps it takes to solve one scenario is arbitrary and depends on the agent. The scenario is resolved if no conflicts remain and if all the unavailability periods have been resolved, i.e $p = 1$ or $p = 0$. Each next scenario resets to the initial state with the original flight schedules, the initial unavailability probabilities, and the reset conflict counters.

A detailed example of a step by step process of resolving a scenario is shown in Figure B.2 in Appendix B.

Important metrics are tracked, which will be used to critically analyse the hypotheses of Chapter 5. They differ for each step, scenario, and episode. When the total amount of steps have been reached and all seeds complete, the saved data is aggregated, smoothed and plotted. This includes computing statistics such as the mean and standard deviation of the total reward per episodes. Furthermore, the hyperparameters are detailed in Table A.1 in Appendix A. For example, the model is trained every eight steps, as training every step would be computationally costly and slow down environment interaction. However, training every 8 time steps provides a good balance between learning speed and computational efficiency. Furthermore, it takes around 8 steps to solve one schedule, which means training happens roughly after each scenario. The target network provides stable Q-value targets for learning and is updated every two hundred steps. Updating more frequently makes targets unstable. Every 200 steps provides a good balance between stability and adaptation speed.

The epsilon greedy policy starts with complete exploration rate of $\epsilon_{start} = 1.0$. It decays to $\epsilon_{min} = 0.05$ over 70% of total timesteps, hence reaching it at 210k steps. The exploration rate decay during training can be seen in Figure 4.2.

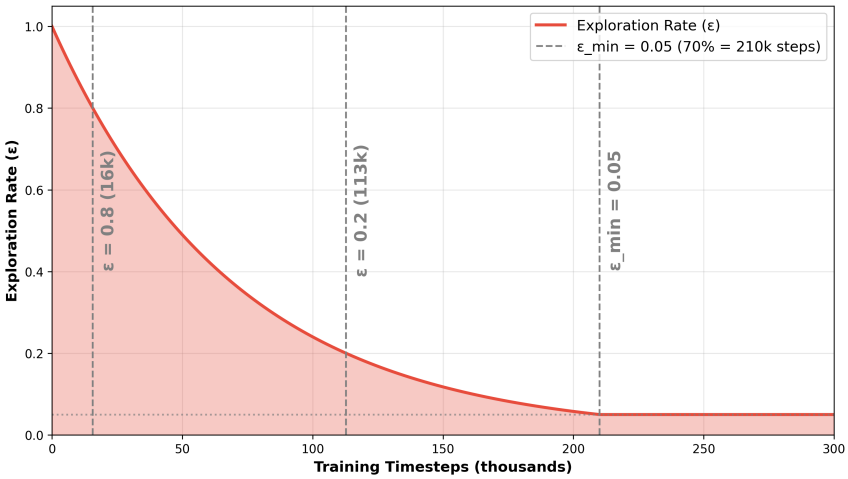


Figure 4.2: Exploration Rate Decay during Training

4.4.3 Data Flow per Step

Both models follow an identical data flow and training procedure, differing only in their input state representation. For Model 1 the input dimension is $D = 608$, representing a stacked observation with $K = 2$ frames and for Model 3 the input dimension $D = 1347$. The complete process is shown below: Figure 4.3 shows a block diagram of the model architecture for a single step. The corresponding Pseudo Code can be found in Listing A.1. The processes run in parallel for all environment types (reactive, proactive, myopic). First, the state, s_t, is constructed, followed by the forward pass $Q(s_t) = [Q(s_t, a_0), Q(s_t, a_1), ..., Q(s_t, a_{71})] \in \mathbb{R}^{72}$. Next the action mask is applied, $\longrightarrow Q_{masked}(s_t)$ and the action is selected via balanced exploration/exploitation. Then, the environment step is executed, ending with the reward allocations. Finally the experience is stored for the next step.

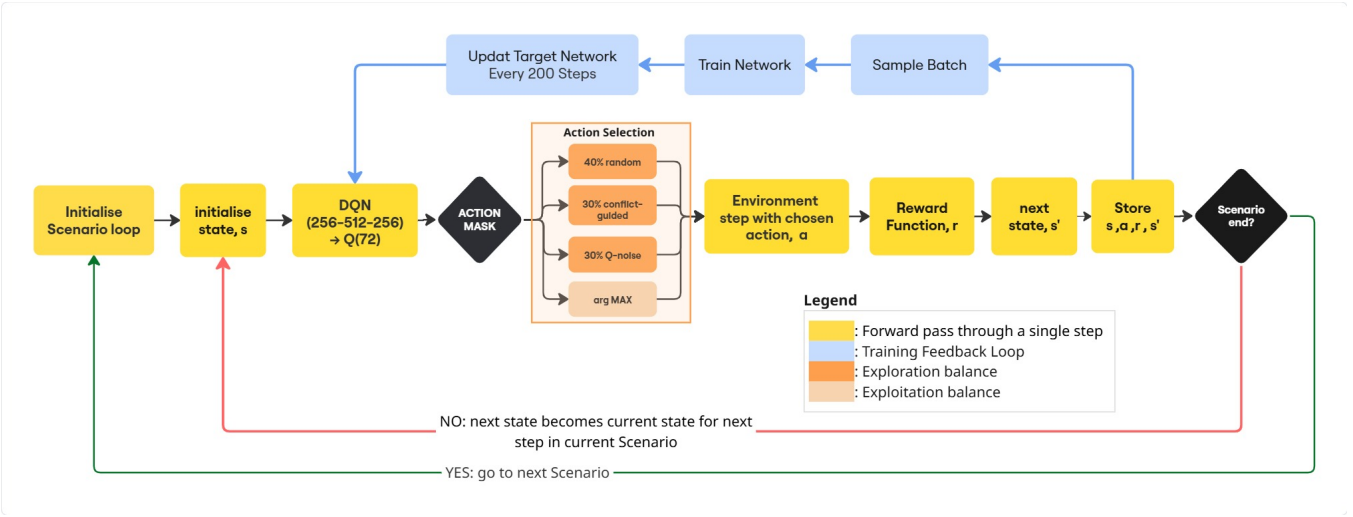


Figure 4.3: Model Network Architecture: Forward Pass for a Single Step. Parallel Process for all Environment Types i.e Reactive, Proactive, Myopic

5 Hypotheses

H1: State Space Formulation and Computational Efficiency

H1-Null Hypothesis: There is no significant difference in computational efficiency i.e. runtime between Model 1 and Model 2.

H1-Alternative Hypothesis: Model 2 has better computational efficiency because of its binary representation, resulting in faster runtime compared to Model 1.

Rationale: Model 2 mainly uses a sparse binary state space formulation of 1s and 0s with time intervals. This should theoretically be computationally more efficient for neural networks as it reduces the complexity thus simplifies matrix operations.

H2: Action Distribution & Strategy Preferences

- **H2-Null Hypothesis:** Model 1 and Model 2 will exhibit similar action distributions (inactions, cancellations, moves) and move success rates
- **H2-Alternative Hypothesis:** Model 2 will demonstrate a more efficient action distribution with fewer inactions, higher proportion of resolving moves, and lower cancellation rates.
- **Rationale:** Better state representation of Model 2 should lead to more informed decisions, resulting in less cancellations and more successful moves.

H3: Conflict Resolution

- **H3-Null Hypothesis:** There is no significant difference in conflict resolution rates between Model 1 and Model 2 across environment types.
- **H3-Alternative Hypothesis:** Model 2 will achieve higher conflict resolution rates due to more direct representation of conflicts in the state space and better action selection.
- **Rationale:** Model 2's flight-centric state space directly encodes conflict information. This thus makes it easier for the DRL to identify and resolve conflicts. The time interval column representation furthermore ensures efficient temporal indications

H4: Learning Performance and Reward Trends

- **H4-Null Hypothesis:** Both, Model 1 and Model 2 will exhibit positive learning trends across episodes with similar final reward values.
- **H4-Alternative Hypothesis:** Model 2 will have achieve a better performance with a steeper positive slope, higher reward values, and better convergence to an optimal policy.
- **Rationale:** Model 2's state space is optimal for a proactive DRL agent with stochastic information and thus leads to superior performance.

H5: Exploration vs Exploitation Trade-off

- **H5-Null Hypothesis:** Both models will show improved performance with decreasing epsilon, ϵ, reaching highest rewards when maximal exploitation is reached.
- **H5-Alternative Hypothesis:** Model 2 will demonstrate a better improvement from exploration to exploitation.
- **Rationale:** Successful learning should result in better performance when exploiting compared to random exploration.

H6: Strategy Performance Comparison: Proactive vs Reactive vs Myopic

- **H6-Null Hypothesis:** There is no significant difference in performance between proactive, reactive, and myopic environment types.
- **H6-Alternative Hypothesis:** Proactive environments achieve better performance than reactive/myopic ones across both models, with Model 2 having a larger performance gap between proactive and reactive/myopic strategies.
- **Rationale:** Extensive probabilistic information in proactive environments yields better risk assessment and thus early conflict resolution.

6 Results

This chapter describes the obtained results in a comparative manner for Models 1 and 2. First, Section 6.1 details the tracked metrics by the model and analyses the outcome regarding computation time. Furthermore, Section 6.2 describes how the actions are distributed, including primary actions, automatic and manual cancellations, and move success rates. Section 6.3 focusses on conflict resolution and section 6.4 analyses the learning trend and the effects of the reward function. Finally, section 6.5 dives into the behaviour during the exploitation and exploration phases.

6.1 Computation Time

During a run, the the model tracks several metrics, which are detailed in Table 6.1. They are stored and used for analysis. This also includes the computation time.

Table 6.1: Metrics Tracking

Metric Level	Metric Category	Description
Episode-Level	Episode Reward	Sum of all scenario rewards across episode
	Computation Time	Time for a complete run
	Conflict Resolution Rate	Percentage of conflicts successfully resolved
	Exploration vs. Exploitation	Performance explorations vs exploitation actions
Scenario-Level	Manual Cancellations	Number of manually cancelled flights
	Automatic Cancellations	Number of environment cancellations
	Tail Swaps	Number of tail swaps
	Tail Remains	Number of tail remains
	Inaction Count	Number inactions
	Initial Conflicts Resolved	Number resolved initial conflicts
	Zero Probability Disruptions	Disruptions that resolved to zero
Step-Level	Action Distribution	Action taken and selection reasoning
	Q-Value Statistics	Q-values of chosen action
	Reward Components	Penalties and rewards for each step
	Conflict Counts	Initial and current conflicts
	Probability Evolution	Evolution of disruption probabilities

Table 6.2 proves that the state space formulation of Model 2 results in much faster training computation times. For Model 1 the run time is 1.84 hours vs 1.06 for Model 2, resulting in a significant speed increase of 40.08%. Both models until the total amount of timesteps were reached. The proactive environment is the fastest, then the myopic, and finally the reactive one. This is logical as the proactive strategy undergoes no masking, which indicates that the amount of information available and the state space formulation impact each other. The dual matrix design combined with primarily binary matrix operations gives Model 2 the clear advantage even though it has a significantly higher input observation space (D = 1347 vs 608). Whereas the agent has to compute the time periods itself in Model 1 as it is only given start and end times, it directly identifies the periods in Model 2 thanks to the columns corresponding to the recovery period's time interval. Due to limited laptop capacity, a run with more than 300000 time steps was impossible when also tracing and logging an enormous amount of detailed step, scenario, and episode information.

Table 6.2: Mean runtime comparison between Model 1 and Model 2, averaged across multiple seeds

Env. Type	Model 1 Runtime (h)	Model 2 Runtime (h)	Difference (h)	Relative Diff. (%)
Proactive	1.84	1.06	-0.78	42.49
Reactive	1.82	1.20	-0.62	34.20
Myopic	1.82	1.09	-0.73	40.08

The consistent 33-43% relative difference, is reason enough to support the H1 Alternative, which predicted that the state space of Model 2 would yield superior run time results compared to Model 1. The outcome is relevant for ADM research in general as it is empirical evidence proving that sparse and primarily binary state space formulations decrease the training time in reinforcement learning applications. Regarding real world implementation, it is incredibly beneficial as faster training allows for larger-scale experiments and faster iteration in research and development if a computationally powerful enough computer is available, which is most often the case for larger institutions.

6.2 Action Distribution

The following analysis examines the action distribution pattern across Model 1 and Model 2 for all environment types, proactive, myopic, reactive. First, a comprehensive examination of the distribution of the three primary actions (inactions, cancel flight, move flight) in Section 6.2.1 shows how they add up to 100%. Next, in Section 6.2.2 the *move flight* actions are subdivided into those that managed to solve initial conflicts, and those that did not. Additionally, the unavailability periods that resolve to 0 are importantly mentioned. The split between automatic and manual cancellations is furthermore examined in Section 6.2.3.

6.2.1 Primary Action Distribution

Figure 6.1 shows the action distribution of the three primary actions, *inaction, move flight, cancel flight*, as well as the total amount of actions taken during the whole run. The observations made are:

- Model 1: Proactive environment yields least inactions, and most move actions
- Model 2: Proactive environment yields least inactions, and most move actions
- Across Models: Model 2's proactive environment performs best. Myopic environment of Model 1 outperforms that of Model 2.

Intuitively it makes sense that the proactive environment has the least inactions and thus the most move actions. The inaction percentage rate correlates with conflict entropy. When conflicts are unclear the agent tends to wait, whereas if they are clear the agent acts. Thus the explanation lies in how state space representation affects the agent's conflict detection ability. In Model 1 conflicts are represented across multiple columns. The state space shows the initial and current conflict counts e.g. "Aircraft 1 has 3 initial conflicts, currently 1 left. To deduce which flights are conflicted, the agent must cross reference the disruption's start and end times, the flight times, and possibly the conflict counts. In contrast, in Model 2 conflicts are directly visible through the matrix structure. Due to the binary time interval structure conflicts are explicit and it is clearer which flight overlaps with which unavailability period at which time interval. The agent has thus clearer conflict signals in Model 2 than in Model 1, making it more certain and resulting in more confident decisions. This results in less inactions in Model 2. Model 2's clear conflict representation explained above, also means that the agent waits less.

However, why do the reactive and the myopic environment of Model 1 show more move actions and less inactions than Model 2? For both myopic and reactive environments, information is limited and decisions have to be made quicker. One reason is that the clear state space representation and clear conflict signals of Model 2 might lead to overconfidence and encourages riskier actions such as cancellations and moves. The agent thinks it sees conflicts clearly and takes aggressive actions situations If these however then do not result in solved conflicts, the agent might retract quickly to a safer waiting strategy and thus inactions increase. Due to Model 1's uncertainty in Decision making, its cautiousness might be rewarded for the myopic/reactive environments. Furthermore, Model 2 has larger action space with complex masking which might mean that more time is spent on action masking and thus less time for decision making. On the other hand, Model 1 can, due to its smaller action space act faster.

In the end, it is certain that Model 1 and Model 2 perform better with more proactive information but Model 2, due to its alternate state representation, seems to benefit even more (51.0% vs 68.8% Moves). Although both benefit, the jump from reactive to proactive is much bigger for Model 2. Thus it can be stated that Model 2 shows a larger performance gap between proactive and reactive/myopic.

An important note to make is that although more move actions are taken than inactions or cancellations for the proactive environment, they are still barely above 50%. Inactions also being close to 50% or even more for the reactive environment suggests that the agent seems to find suboptimal policies. This flaw will be discussed further in Section 6.4.

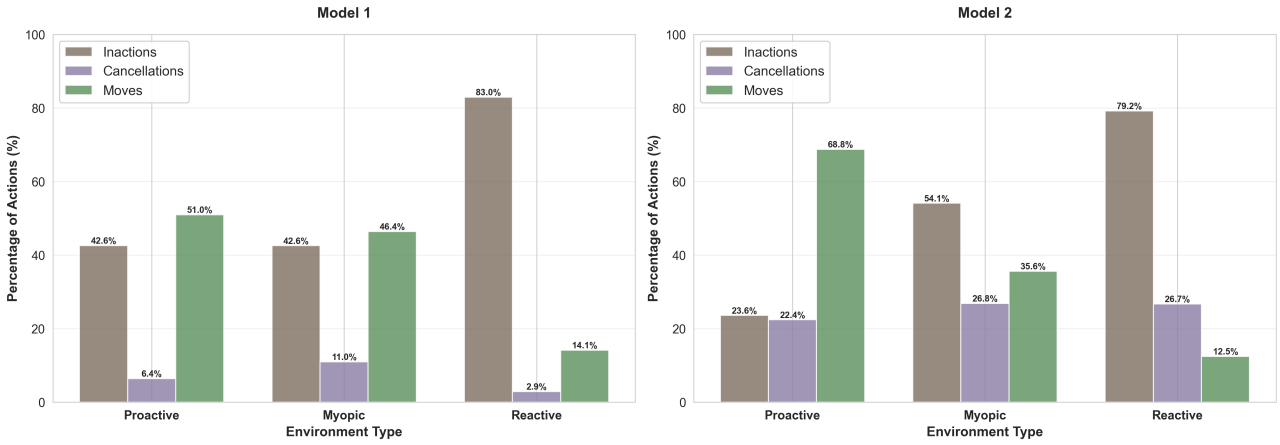


Figure 6.1: Primary Actions Distribution Comparison: Model 1 vs Model 2

6.2.2 Move Success Rates

Figure 6.2 considers the sum of all initial conflicts across all scenarios and all episodes. It depicts how these initial conflicts have been resolved. Were they resolved successfully by the agent? Did the probability of the unavailability periods resolve to $p = 0$ and did the conflict disappear? Or were overlapping flights cancelled by the agent or the environment and did the conflict remain unresolved? It was found that:

- Model 1: Highest proportion of resolving moves for proactive, then for myopic and reactive
- Model 2: Highest proportion of resolving moves for proactive, then for myopic and reactive
- Across Models: Model 2 is better for proactive, but Model 1 outperforms for myopic and reactive

The agent chooses actions based on the highest expected reward, which depends on the information availability, the state representation, and the timing. These decide not only whether the agent can see conflicts early but also if it can identify them correctly and if there is enough time to act.

The proactive environment performs best as it detects unavailability periods earlier giving it more time to act, thus having a higher chance of success. This can be summarised by Equation 6.1 describing the probability of successful move action, where $P_{\text{detect}}(t)$ is the probability of detecting conflict at time t, $P_{\text{valid}}(t)$ is the probability move is valid at time t, and $P_{\text{resolve}}(t)$ is the probability that the move resolves a conflict.

$$P_{\text{success}}(a_{\text{move}}|\text{env}) = \int_0^T P_{\text{detect}}(t) \cdot P_{\text{valid}}(t) \cdot P_{\text{resolve}}(t)\, dt \tag{6.1}$$

For each environment:

- **Proactive:** $P_{\text{detect}}(t) \approx 1$ for early t, leading to high $\int P_{\text{success}}$
- **Myopic:** $P_{\text{detect}}(t) \approx 0$ until $p = 1.0$, reducing time window
- **Reactive:** $P_{\text{detect}}(t) \approx 0$ until imminent, minimal time window

Note that a suspicious observation may be made, namely that the unavailability periods that ”Resolved to Zero”, i.e probability naturally went from >0 to 0.00, are more numerous for Model 2 than for Model 1. It is important to point out that whether probabilities go to 0.00 or 1.00 is random and the agent cannot control this outcome. The way it works is as follows: for each timestep the yet uncertain disruption probabilities evolve stochastically with a bias term that pushes values above 0.5 towards 1 and those below 0.5 towards 0. Once the start time of a scheduled uncertainty period is reached, the final probability is determined by a random dice roll weighted by the evolved probability i.e. the chance of the disruption actually occurring equals the probability value at that moment. The observed difference between Model 1 and Model 2 happens due to their distinct action sequence. Different actions trigger a different number of random function calls. Thus their unique actions affect the random number generator state differently. This consequently impacts the probability evolution for subsequent disruptions. Hence, the ”Resolved to Zero” difference does not show strategic superiority as it is a computational property rather than

a behavioural difference. Hence, it makes sense to focus entirely on ”Resolved by Agent” rates as done above as this is the only metric the agent can control and allows direct comparison of metrics that reflect agent capabilities.

Furthermore, similar to Section 6.2.1, the myopic and reactive environment of Model 1 outperform Model 2.

Although the proactive environment yields the most move actions, the unresolved conflicts dominate around half of all initial ones. This shows that the agent seems to learn a suboptimal policy and is not able to understand the input information as well as it should.

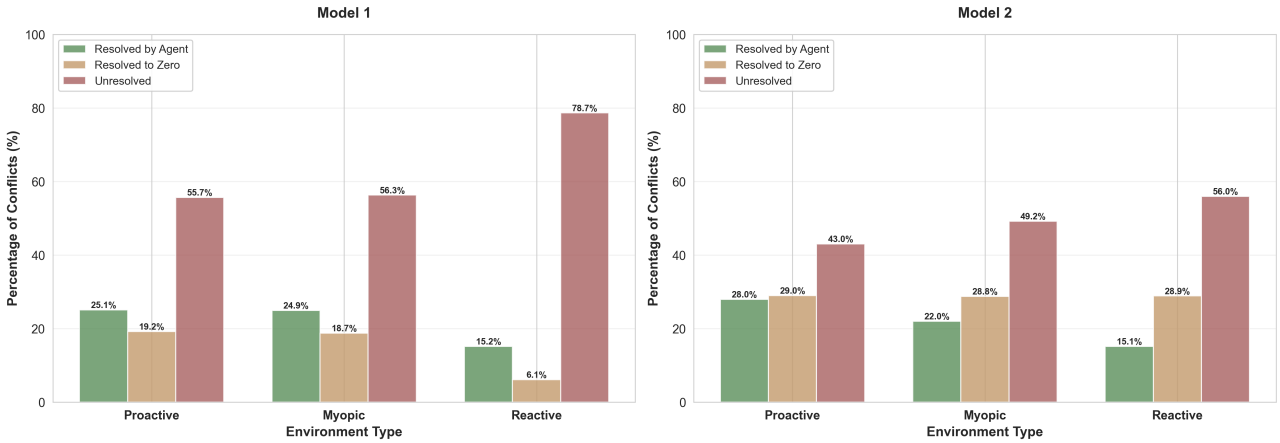


Figure 6.2: Initial Conflict Resolution Outcome: Model 1 vs Model 2

6.2.3 Automatic Cancellations

An automatic cancellation can happen in addition to a primary action. So if at each timestep one primary action is taken, an automatic cancellation can happen on top of it. This is because the key idea behind automatic cancellations is that the agent failed to act in time and thus the flight is cancelled by the environment and not by the DRL agent. Figure 6.3 shows the percentage of total primary actions that simultaneously also experienced an automatic cancellation, resulting in the observation that:

- Model 1: Higher automatic cancellation rates
- Model 2: Lower automatic cancellation rates

As the evaluation data volumes differ between models, data normalisation was done per 100,000 steps to enable fair comparison. Hence, Figure 6.4 depicts the normalised cancellation rate. This analysis reveal a striking contrast between the disruption management strategies of Models 1 and 2. Model 1 clearly exhibits a higher automatic cancellation rate across all environment types. For example does Model 1 have roughly 17 thousand automatic cancellations for the proactive strategy while Model 2 has 15 thousand. Conversely, Model 2 has a much higher number of manual cancellations as well as a total cancellations. This pattern persists across the proactive, myopic, and reactive environment, which suggests that there is a fundamental difference in learned behaviour.

The higher automatic cancellation rates in Model 1 underline that is follows a more passive approach. Combined with the higher inaction count seen in Figure 6.1, Model 1 seems to have developed a ’wait and see’ strategy for its disruption management. This results in more flight’s not being moved on time and thus needing to be automatically cancelled by the environment.

Model 2’s large manual cancellation rate highlights that it does not accept Model 1’s ’wait and see’ approach but rather proactively tries to identify those flights that are conflicted, deliberately cancelling them early. This can be considered a strategic sacrifice fapproach. Although it yields immediate penalties higher than those of the inactions used by Model 1, it has two key benefits. First, it prevents automatic cancellations by acting before the time reaches the flights. Secondly, it frees schedule space which can be used by the agent to move other flights to, which allows for more efficient conflict solving later on.

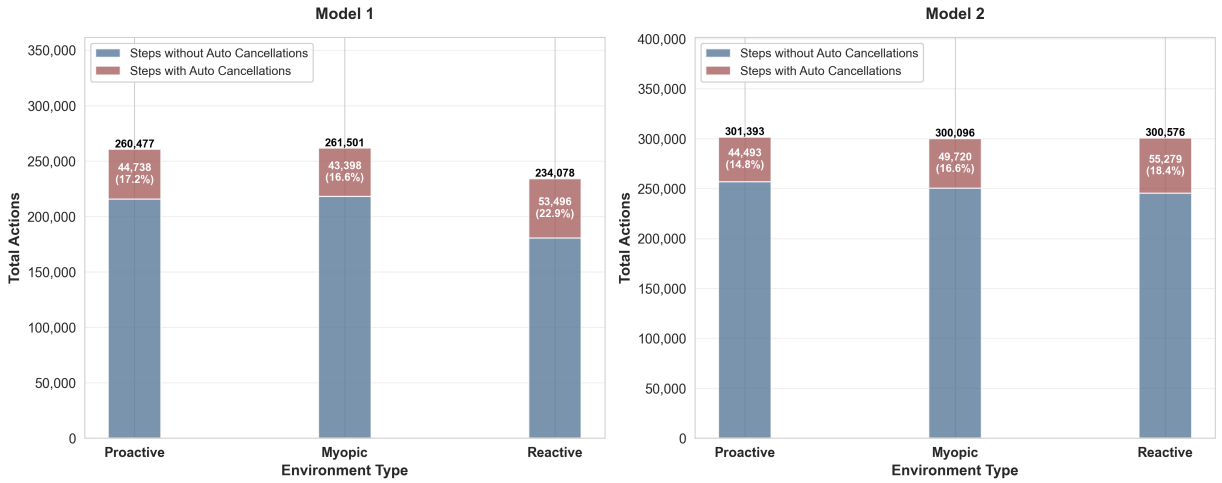


Figure 6.3: Model 1 vs Model 2: Automatic Cancellations Distribution over all Primary Actions

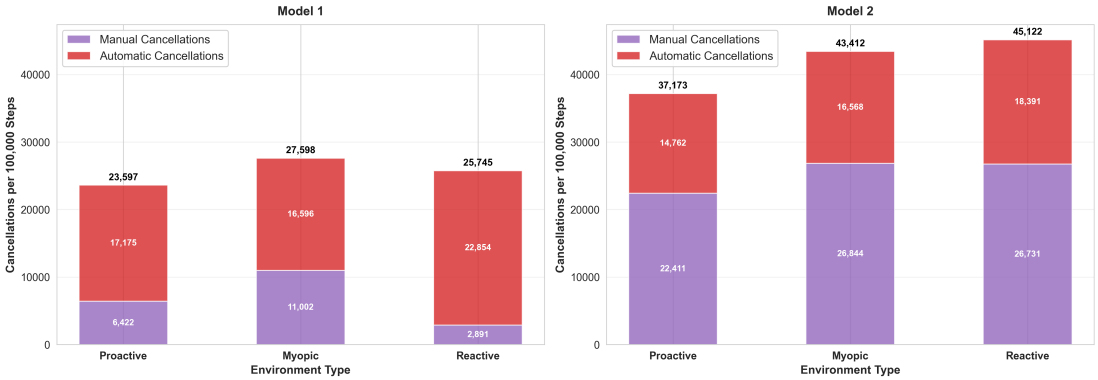


Figure 6.4: Model 1 vs Model 2: Total Normalised Cancellation Distribution: Manual vs Automatic

6.3 Conflict Resolution

Conflict resolution rates are crucial to investigate the performance of models 1 and 2 and their environment types. Table 6.3 details the metrics and outlines the findings per environment. To begin with, Model 2's proactive environment beats Model 1 in all metrics. It has a superior resolution rate for its proactive environment with 0.29% vs to 0.27% showing a 5.8% increase. It also has a higher mean final reward value with 39.38 vs 33.44 and a 17.8% increase, and a less steep reward slope of -0.40 vs. -0.48. The reward slope are the reward values per episode over time and will be further discussed in Section 6.4. This supports the findings in Section 6.2 and cements the H6 Alternative that stochastic (proactive) information benefits performance and that Model 2 better weaponises this early accessible information. For the myopic environment the tables turn and Model 1 scores better for the resolution rate (26% vs 21%), and the episode length. However, although the myopic strategy of Model 1 solves more conflicts, the final reward is higher for Model 2, underlining that during the whole run also a lot of inefficient actions have been taken resulting in higher penalties for individual actions. The opposite is true in reactive environments where the Model 2's slightly higher resolution rate is rivalled by Model 1's higher reward.

Finally, it is safe to say that Model 2 exhibits higher resolution rates for the proactive and reactive environments but not for its myopic one. This shows that resolution rate and final reward do not always align and it can be concluded that the conflict resolution metric is only one component of overall performance and must be considered as a puzzle piece in the bigger picture.

Table 6.3: Metrics Tracking

Environment	Metric	Model 1 Mean	Model 2 Mean	Difference	Relative Diff %
proactive	Resolution Rate	0.2738	0.2897	0.0159	5.8123
	Final Reward	33.44	39.38	5.94	17.77
	Reward Slope	-0.48	-0.40	0.08	15.84
reactive	Resolution Rate	0.1493	0.1540	0.0047	3.1282
	Final Reward	3.6482	2.9524	-0.6958	-19.0718
	Reward Slope	-0.37	-0.34	0.02	6.16
myopic	Resolution Rate	0.2558	0.2133	-0.0425	-16.6172
	Final Reward	12.61	17.87	5.26	41.68
	Reward Slope	-0.60	-0.49	0.10	16.99

6.4 Learning Trends & Reward function

Figure 6.5 depicts the reward over a complete run for the total number of steps. Immediately, the first observation that hits the eye is the negative slope that continues to decrease with increasing time steps. This drastically contradicts the H4 Alternative, which states that the expected curves should have positive slope in order to indicate positive learning experiences. The current result highlights a crucial shortcoming of reinforcement learning solution methods, namely that it is highly dependent on parameter tuning making it challenging to replicate across the research domain. This article at first aimed to build upon and improve the DRL approach to the ADM problem introduced by Becking (2024) but due to replicability challenges including scarcely documented original code and its parameters, the resulting negative slope persisted. However, although the reward behaviour may be categorised as a failed or unexpected result, the relative behaviour of the different environments, their action distributions, their conflict resolution, and the reward magnitudes are stable and worth interpreting. Despite this learning failure, valuable scientific comparisons can thus be made as already done in the previous sections.

Several reasons for the negative trend can still be hypothesised. First, the mainly negative reward structure may overwhelm the single positive bonus. However, extensive reward function redesign was done over the research period, from implementing solely positive rewards to solely negative ones to enormous reward values to small ones to normalised ones, but the curve never broke loose from its negative trend. This suggests that even though the reward structure makes sense, the agent was unable to establish a clear solution path in combination with the state space formulations. Even though the state space of Model 2 is easier to understand and yields higher resolution rates for the proactive environment, the information might still be too unclear for the DRL agent. Another reason is related to Figure 6.2. An overwhelming number of conflicts remain unresolved or resolve to 0, which rings alarm bells. This reflects the negative reward slope and that the agent cannot learn a good policy. It also hints at that the agent might not see enough examples of positive resolution cases. This then circles back to the initial synthetic data sets used which, due to the split between stochastic, deterministic and mixed scenarios all presenting different initial conflicts, might be too broad to be successful. The stochasticity of the unavailability periods could be another cause. If the agent is given rewards for solving conflicts but then the unavailability never occurs, it might get confused as to how to predict when a disruption will happen or will not happen. Finally, the agent could be overfitting to early scenarios where it learns good policies for initial individual scenarios but is unable to replicate them and thus fails to implement them during the exploitation segment. However, despite the numerous and systematic interventions, which furthermore included hyper parameters (Table A.1) tuning such as changing the learning rates by a factor of 10, altering gamma's, γ, range between 0.995 and 0.9999, and changing batch sizes, the slope held on to its downwards path.

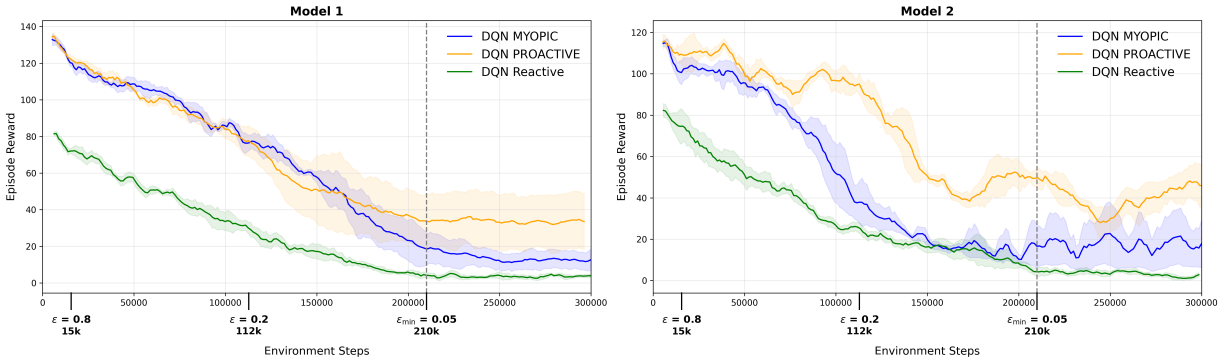


Figure 6.5: Episode Rewards for Model 1 and Model 2 with exploration rate, ϵ, indications

Table 6.4 highlights important observations from the reward trend figures. To begin with, the proactive environment yields the highest rewards over the first and last 50 time steps as well as the highest overall mean rewards for both models (67.2 for Model 1 and 70.5 for Model 2). This aligns with the final reward discussed in Section 6.3 and also reflects the findings of Figure 6.2 and Figure 6.1, where it was seen that the proactive strategy resolves the most conflicts and makes the most moves. Furthermore, all environment have a less steep slope in Model 2 than their respective counterparts in Model 1, e.g. for proactive -0.40 in Model 2 and -0.48 in Model 1. This is depicted visually by Figure 6.6. One might notice that the reward slope for the reactive environment is the least steep and conclude that it performs best but this is not the case. Combining the observed slope with the action distribution findings in Figure 6.1, the main primary action taken by the reactive strategy is made out to be inactions. It becomes thus apparent that the reactive environment likes to play it safe and does not take many aggressive nor adventurous actions as do the proactive and myopic environments. This assures a steady similar penalty for most steps resulting in the observed slope value. This proves again that a single metric cannot predict the overall performance of the DRL model. Each metric is a single component of a dependent dynamic system and thus has to be looked at in sufficient context.

Moreover, Model 2's proactive curve shows noticeable intermittent upward trends. This is underlined by the large positive window of 33.45% depicted in Table 6.4 showing the percentage of 30 episode intervals with positive slopes. In Model 1 it is only 8.90%. This difference suggests that Model 2's state space representation allows for brief reward improvements. A window length of 30 was chosen to reflect temporary strategy discovery rather than long term learning. The higher positive window value for Model'1 reactive environment compared to its proactive one suggests that late unavailability period visibility leads to occasionally good policies during a short episode span. The inconsistent pattern across environment types highlights that these improvements are only brief and not sustained as both models remain unable to break the spell of the ruling negative reward function.

Table 6.4: Reward trends and learning dynamics averaged across multiple seeds

Model	Environment	Mean Reward	First 50	Last 50	Slope	Improvement	Positive Windows (%)
Model 1	Proactive	67.20	116.01	33.60	-0.48	-82.42	8.90
	Reactive	25.96	63.04	3.57	-0.37	-59.47	12.45
	Myopic	61.58	115.21	12.82	-0.60	-102.40	6.45
Model 2	Proactive	70.49	108.49	39.26	-0.40	-69.23	33.45
	Reactive	25.63	62.96	3.12	-0.34	-59.84	8.30
	Myopic	46.05	101.56	17.74	-0.49	-83.82	22.30

Note: All values are averaged across two random seeds (232323 and 242424). Slope is computed via linear regression over all episodes. Improvement is defined as the mean reward over the last 50 episodes minus the first 50 episodes; negative values indicate performance degradation. Positive windows represent the percentage of 30-episode windows with positive local slopes.

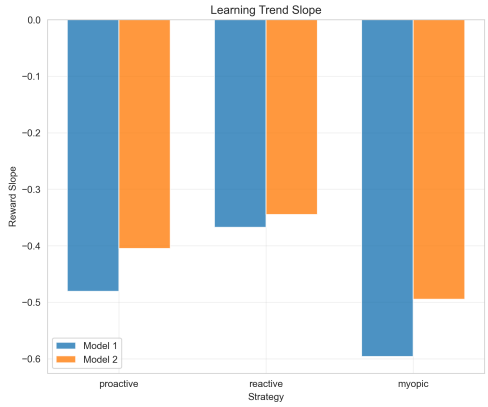


Figure 6.6: Slope of the Reward Curves portrayed in Figure 6.5

Additionally, Table 6.5 focusses on the mean reward differences between the proactive and the reactive/myopic environments. Model 1's proactive and myopic curves are much closer together with a separation of 5.62 compared to Model 2's curves with 24.44. The closeness of Model 1's curves stresses that its more compact state space leads to similar action strategies and results in similar policies, whereas Model 2 ends up with more individualised policies for each environment.

Table 6.5: Environment separation metrics averaged across multiple seeds

Model	Proactive Mean	Reactive Mean	Myopic Mean	P–R Sep	P–M Sep	P–R Slope Diff
Model 1	67.20	25.96	61.58	41.24	5.62	0.11
Model 2	70.49	25.63	46.05	44.86	24.44	0.06

Note: All values are averaged across two random seeds (232323 and 242424). Mean rewards are averaged over all training episodes. P–R Sep = Proactive Mean − Reactive Mean. P–M Sep = |Proactive Mean − Myopic Mean|. R–M Sep = |Reactive Mean − Myopic Mean|. P–R Slope Diff is the absolute difference between proactive and reactive learning slopes, computed after averaging slopes across seeds.

6.4.1 Metric Trends over Total Episodes

The reward structure is designed in such a way as to teach the agent that: **move flight** best > **inaction** okay for waiting (probability unresolved but no conflict), otherwise acceptable > **manual cancellation** bad > **automatic cancellation** worse

As the learning trend is however negative, this suggest that more negative rewards are accumulated towards the end of the run. These suspicions are confirmed in Figure 6.7 and Figure 6.8. In both figures, the attributes normally associated with good actions such as tail swaps and solving conflicts follow a similar pattern as the slopes in Figure 6.5 for all environments. On the other hand, the attributes associated with negative performance such as cancellations seem to go up significantly towards the end.

Solely concentrating on the environments, it can be seen that Model 2's proactive strategy is more successful everywhere: it has higher tail swaps, higher resolved conflicts, fewer inactions and fewer total cancellations compared to its other strategies. For Model 1, the gap between the proactive and myopic strategy is smaller. This further cements the separation analysis in Table 6.5.

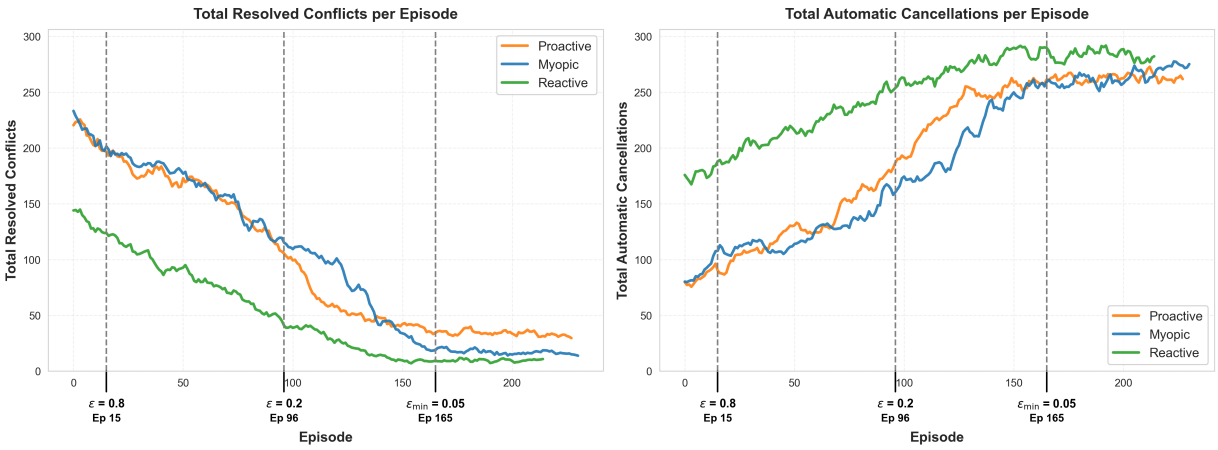


Figure 6.7: Model 1: Automatic Cancellations and Resolved Conflicts across episodes

Figure 6.8: Model 2: Automatic Cancellations and Resolved Conflicts across episodes

6.5 Exploration vs Exploitation

In the previous section, it became apparent that for both models the rewards at the start and throughout the exploration period of the epsilon greedy policy are significantly higher than during the exploitation period. In order to investigate this contradictory behaviour, this section explores Models 1 and 2 at different epsilon target values. The following target values are chosen for the analysis:

Table 6.6: Epsilon values (ϵ) reached at corresponding training episodes

	ϵ		
Model	0.80	0.20	0.05
Model 1	15	95	165
Model 2	14	94	166

Table 6.7 and Table 6.8 depict the average delayed flights, average resolved conflicts, average cancellations, and average tail swaps per scenario for each epsilon. The values confirm the critical paradox that the agent performs better during high exploration with $\epsilon = 0.8$ than during low exploration with $\epsilon = 0.05$. Delays are associated with good actions as it means that a flight was moved possibly able to solve a conflict. The delayed flights in Model 1 for

its proactive strategy go from 0.9 per scenario to 0.6 to 0.3, and the tail swaps from 0.8 to 0.4 to only 0.1 exhibiting a clear negative trend. Similar observations can be done for the remaining key performance indicators.

The reason why the key performance indicators yield more successful results during exploration is because or poor temporal credit assignment, and the 40%-30%-30% balance architecture discussed in Equation 4.3. The model is guided to act on conflicted flights during exploration. This results in positive resolution rewards. However, even with theses positive indicators, the model could not extract a meaningful converging policy. The cause for this is related to what was observed in Figure 6.2, namely that even though many conflicts resolved, more than 55% or more did not. Hence, good actions were present but occurred much less frequently than bad ones, causing the agent's decisions to be saturated by the the latter. The given resolution bonus rewards are sparse and the unresolved resolution penalties are temporarily far apart, making it quite difficult for the Q-values to associate the rewards to state action pairs.

During exploration the agent tries different things, being more aggressive and risky by relying on conflict information. But then a gradual Q-value collapse occurs as they become more and more miscalibrated due to a combination of rare conflict-resolving actions, many small negative rewards, and an ineffective state space model. The ability to decide what a good or bad action is lost and the model converges to a negative value.

Regarding both models, Model 1 seems to suffer more than Model 2 as it experiences a steeper decline, underlined by an approximate 80% drop vs. 50% drop in average number of resolved conflicts. Model 1's reactive strategy becomes almost completely inactive when exploration stops while Model 2 stays slightly more active. This means that Model 2's state-space provides better information to the agent, making it see clearer. However, it is still insufficient to yield a conclusive policy.

Finally, one can conclude that, when exploration stops, both models have learned to do nothing. Inactions are favoured over resolving actions effectively. Few tail swaps happen, few conflicts are being resolved, and flights end up being automatically cancelled.

Table 6.7: Model 1 Performance Comparison Across Epsilon Values

Strategy	Model 1 Averages per Scenario														
	Delays (flights)			Delays (min)			Swaps			Cancels			Resolved		
	0.8	0.2	0.05	0.8	0.2	0.05	0.8	0.2	0.05	0.8	0.2	0.05	0.8	0.2	0.05
Proactive	0.9	0.6	0.3	186.9	184.0	87.9	0.8	0.4	0.1	1.5	1.7	2.1	1.2	0.7	0.3
Myopic	0.9	0.6	0.1	217.5	152.0	29.9	0.8	0.4	0.1	1.5	2.1	2.6	1.2	0.7	0.2
Reactive	0.6	0.3	0.0	122.0	59.9	6.8	0.4	0.2	0.0	1.7	2.2	2.4	0.9	0.4	0.1

Table 6.8: Model 2 Performance Comparison Across Epsilon Values

Strategy	Model 2 Averages per Scenario														
	Delays (flights)			Delays (min)			Swaps			Cancels			Resolved		
	0.8	0.2	0.05	0.8	0.2	0.05	0.8	0.2	0.05	0.8	0.2	0.05	0.8	0.2	0.05
Proactive	3.9	3.7	1.8	596.3	565.5	283.1	2.1	1.9	0.9	2.1	2.2	3.1	1.0	0.9	0.6
Myopic	3.9	0.8	1.3	551.2	134.9	241.9	2.4	0.5	0.8	2.2	4.0	3.5	1.1	0.5	0.5
Reactive	1.6	0.7	0.2	242.3	120.8	58.3	0.9	0.4	0.1	2.6	3.4	3.6	0.8	0.4	0.3

7 Sensitivity Analysis, and Robustness

This chapter investigates the reward sensitivity in Section 7.1 by analysing the effect of disabling individual components. Furthermore, Section 7.2 discusses the robustness of hyperparameters.

7.1 Reward Sensitivity

In order to investigate and validate the effectiveness of the reward formulation and whether it contributes to the unexpected negative slope, a sensitivity analysis has been performed. For this, the reward components were disabled consecutively and then Model 2 was run for each configuration. High sensitivity would mean that the reward design works as the model is then sensitive to slight changes in its formulation. The result for the myopic environment type is seen in Figure 7.1. As the outcome for the remaining environment types are similar, they have been added to Appendix B.

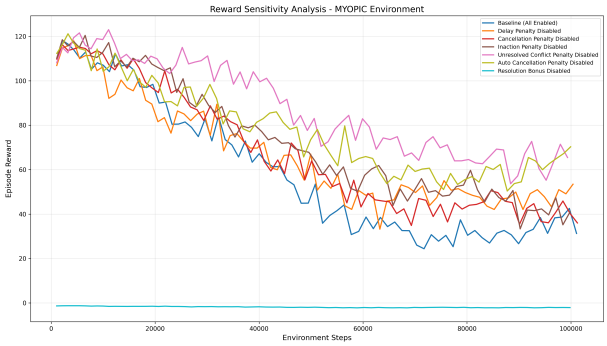


Figure 7.1: Reward Sensitivity Results for Model 2

Table 7.1: Impact of Penalty Configuration Changes on Myopic Strategy Performance

Penalty	Delay	Cancellation	Inaction	Unresolved Conflict	Auto Cancellation	Resolution Bonus
Myopic	+7.30	+8.11	+13.22	+27.90	+18.10	-62.86
Change	+12.0%	+13.3%	+21.7%	+45.8%	+29.7%	-103.1%

From the figure as well as from Table 7.1, it can be seen that the individual curves differ from the baseline resulting in different mean reward values. High change values, especially for the resolution bonus, indicate high sensitivity, suggesting that the agent indeed correctly interprets reward signals as it reacts to changes made to it. The high sensitivity highlights that the reward function is an important part of DRL and needs to be carefully designed. However, it also stresses that the reward design, once reward tuning and optimisation have been completed and validated, should be frozen in order to ensure stability of the model.

Furthermore, it is clearly seen that the learning trend, first observed in Figure 6.5, remains negative across all reward configurations. This proves that the agent fails to learn optimal policies despite reward modifications, suggesting that the cause is elsewhere. Most likely, the exploitation exploration or state space complexity fuel the negative slope.

7.2 Hyperparameter Sensitivity & Robustness

In order to further investigate the robustness and sensitivity of the presented DRL models, the learning rate, α, is analysed. Out of all parameters, α has been chosen as it is one of the most commonly adjusted in the domain of RL. Three variations of the learning parameter are considered:

- Baseline: $\alpha = 0.0015$
- Lower learning rate: $\alpha = 0.0008$
- Higher learning rate: $\alpha = 0.0025$

Table 7.2: Performance Comparison Across Learning Rates

Learning Rate	Proactive Mean Reward	Myopic Mean Reward	Reactive Mean Reward
Baseline (0.0015)	73.07	58.48	38.19
Low LR (0.0008)	79.18	77.76	26.15
High LR (0.0025)	70.26	71.25	23.97

The results in Table 7.2 show that rewards differ across learning rate, indicating that they are susceptible to learning parameter changes as values differ. This means that the results discussed earlier will most likely show different outcomes for different learning parameters. However, the performance ranking of the different environments is expected to remain similar, with the proactive environment generally outperforming the reactive and myopic ones. This highlights that the robustness of the DQN framework itself is quite low but the relative advantage of the environment strategies is preserved. Moreover, it can be observed that, although the gap between the proactive and myopic environments lowers with decreasing learning rate, the reactive strategy's outcome is always the worse. This highlights that the lack of stochastic information negatively impacts the DRL agent's performance.

8 Discussion & Limitations

This chapter discusses the outcomes, limitations, and considerations for future research. First, Section 8.1 evaluates the hypotheses formulated in Chapter 5 with respect to the results obtained in Chapter 6. Next, Section 8.2 highlights the existing limitations, and finally Section 8.3 underlines the implications of the research for future work.

8.1 Hypotheses Revisited

The results reveal significant differences between the models regarding their environment strategies, action preferences, and conflict resolution effectiveness. They highlight the importance of the availability of state-space information for a decision-making agent. Following the outcome of the use-case experiment, the hypotheses are now revisited and discussed.

To begin with, the H1 Alternative is **supported**. The hypothesis predicted that Model 2's sparse state-space formulation yields better computational efficiency. This is strongly underlined by the results from Table 6.3, showing a a steady reduction in the training time across all environments strategies. Despite having a larger observation state dimension cof 1347 vs 608, Model 2 completed the training in 1.06 hours vs 1.84 hours for Model 1. This favourable behaviour of Model 2 is because of its primarily binary state space matrix which allows for more efficient operations. Model 1's formulation on the other hand requires the agent to calculate duration times using continuous start and end values. The fact that Model 2 uses its columns as time intervals removes this computational step.

Next, the findings are nuanced for H2. Rather than complete contradictions, the H2 Alternative can be **partially accepted**. The hypothesis predicted that a better and more efficient action distribution with fewer inactions, more resolving moves, and fewer cancellations will be observed for Model 2. For the proactive environment, Model 2 achieves fewest inactions and the highest amount of move actions (68.8% vs 51% for Model 1). However, Model 2's advantages are often environment dependent. Although it excels with information and time (proactive), Model 1 performs better under constraints (myopic/reactive). Regarding the cancellation rates, Figure 6.4 highlights that Model 2 has more manual cancellations while having fewer automatic ones. The fact that Model 2 has higher overall reward than Model 1 even though it receives more manual cancellation penalties Figure 6.4, provides 2 key insights. Firstly, that Model 2 employs a proactive disruption management approach whereas Model 1 uses a 'wait and see' strategy, and secondly that this approach is more beneficial in the long term as the cancellation rate is compensated by more effective resolution for those flights that remain after cancellation. To conclude, this highlights that controlled losses can be superior compared to a more conservative approach. This is especially true when the conservative approach causes an increase risk for uncontrolled losses later on.

Likewise, the H3-Alternative is only **partially supported**. It assumes that Model 2's more straight forward conflict representation would result in better conflict resolution. While Model 2 indeed exhibits a 5.8% higher resolution

rate for the proactive (28.97% vs 27.38%), it does not do so for the myopic one. This finding suggests that the seemingly advantageous state-space formulation of Model 2 is not universal but rather environment dependent. It is most pronounced when temporal information is largely available as in the proactive case.

Importantly, a critical observation was made that inactions make up a large part of all primary actions and that, out of all conflicts, over 55% remain unresolved. This extremely high rate, regardless of model or environment type shows that the agent might adapt a suboptimal policy suggesting that it cannot make logical links between observation input and rewards.

Furthermore, Section 6.4 has shown undeniable truths proving that the H4 hypotheses has to be completely and unequivocally **rejected** as neither the H4 Null nor the H4 Alternative can be supported. Whereas the hypothesis predicts positive steeper learning slopes in either case, the results clearly show the opposite. Table 6.4 records negative reward slopes up to -0.60 for Model 1 and -0.49 for Model 2. Although Model 2 shows better performance than Model 1, the negative learning trend does not improve across environments, suggesting that the DRL agent learns suboptimal policies. However, the intermittent positive windows of positive slope values indicate that the agent finds positive strategies, which unfortunately cannot be sustained by either model.

Moreover, the findings discussed in Section 6.5 show that performance is better during exploration than exploitation for both models and across all environments. This thus strongly **contradicts** the H5 Alternative hypothesis and the expected behaviour of a well trained DRL agent. Table 6.7 and Table 6.8 show steady decreases for positive performance indicators; average tail swaps and delays per scenario go down with decreasing epsilon while cancellations go up. Model 1's proactive environment furthermore shows a 75% reduction in the average number of resolved conflicts per scenario, from 1.2 when epsilon = 0.8 to only 0.3 when epsilon = 0.05. Similarly, Model 2 shows a 40% drop for its proactive strategy, from 1.0 when epsilon = 0.8 to only 0.6 when epsilon = 0.05. Although Model 2 shows less severe degradation, it still follows the downward trend. The results further explain the pattern of the negative reward learning slope. This leads to the verdict that the DRL model's Q-values seem to be unable to capture the beneficial patterns from the initial exploration, thus failing later on.

The paradoxical finding that the models perform better during exploration highlights their movement towards passive behaviour i.e. learning to do nothing. For Model 1, the reactive strategy becomes more and more inactive with decreasing epsilon. Flights are not moved anymore and automatic cancellations take over.

The H6 Alternative predicted that the proactive environment would perform better than the myopic and reactive ones. Following the examined metrics such as the action distribution in Figure 6.1, the resolution manners in Figure 6.2, and the reward trends in Table 6.4, it was seen that the proactive strategy consistently outperformed the others. It is thus sound to **reject** the H6 Null and to **support** the H6 Alternative as all findings align with the latter. Proactive environments do indeed perform better. Furthermore, Model 2 demonstrates a larger performance gap, meaning that it makes better use of stochastic information. Model 2 showed a proactive-myopic separation of 24.44 compared to 5.62 only for Model 1. This aligns with the expectation that Model 2 provides better disruption visibility.

Additionally, Section 7.1 shows that the DRL model is highly susceptible to large changes. This indicates that the model is not yet ready for real life implementation as, even though a certain hyper parameter value might give excellent results for one problem, it might give a very different one for another. Airlines cannot rely on such instability. Furthermore, if a model is very sensitive and needs to be tuned often, the airline might need to hire data scientists to do so. This might increase the amount of long term hidden costs significantly.

8.2 Limitations

A significant limitation is the limited amount of GPU and RAM capacity of a standard laptop. This restricted computational capacity constrained the number of timesteps to 300000. Due to detailed metrics logging, running more than this generates a massive amount of data including detailed data for each episode, environment step, state, and action that the laptop cannot sustain. Furthermore, different environment types (proactive, myopic reactive) means that processes are running in parallel: 2 seeds × 3 environment types = 6 processes running simultaneously. The large replay buffer of 100000 transitions as well as large state vectors further hinder the PC's performance and push the hardware to its limits. This has the implication that the models could not achieve full convergence although consistent negative trends underlined that any additional training would not transform the slope to be positive.

Furthermore, the problem scope can be ameliorated. This research solely focusses on aircraft, which is only one of

ADM's three essential entities; aircraft, crew, and passenger. To make a realistic disruption management system all three should be included as decisions related to one entity can have significant effects on the others and vice versa.

Next, the training data could be improved upon. As described in Section 4.4.1, it was synthetically generated. While this allows for endless combinations of number of aircraft, flights and scenarios, as well as control, this work focused on one use-case only. More use-cases could be created and compared, improving the credibility of the outcome. Furthermore, the balance of deterministic, stochastic and mixed scenarios was opted for in order to make them balanced. However, this might not adequately represent reality.

In addition, while DQN is a good choice as RL algorithm for it provides a good baseline, other algorithms such as Proximal Policy Optimisation (PPO) might be able to tackle some of the identified limitations.

Another constraint are the state-space and reward function formulations. Although they were extensively researched for this work, their design remains a hurdle for the agent's performance. Negative learning trend remained for all tested configurations, but it remains possible that an alternate state space formulation will yield improved results.

8.3 Future Work

To begin with, the observations in Table 7.2 explain the difficulties regarding the replicability and reproducibility of another researcher's results. If certain hyper parameters were used, or an explicit reward function design was implemented but the corresponding coding resources had not been provided, it is unlikely that the exact same outcome will be achieved. For this work, an attempt to replicate Becking (2024)'s results was made but due to the just mentioned reasons it was unsuccessful, resulting in a different research path. This stresses the importance of collaborative research and clear research documentation for future research in ADM.

The discussed outcomes in Section 8.1 also have several implications on the usage of DRL for ADM in the aviation industry. First, it highlights a significant roadblock for the use of RL, namely that airlines' objectives are variable while the reward design is highly susceptible to any changes, thus needing very careful handling. Thus an important consideration for future research is that the reward function shall be considered a critical parameter. Its design should thus imperatively be part of a sensitivity analysis to ensure validation and calibration. In addition, the reward structure should, once tested and working, be locked in before any product deployment. If significant changes are done to the framework, the testing of the reward function should be revisited. Furthermore, the reward composition could be decomposed into immediate, intermediate and final rewards given thus at different times during a run. This could provide better and more frequent learning signals.

As the experiments were limited by computing power, it is urged that future research focusses on using a super-computer to better view the effects of the state space and reward function design on airline disruption problems. Thirdly, as this paper solely considers aircraft, it would be interesting to see the effect of crew and passengers on the model. Both entities significantly complicate the problem, hence expanding Model 2's state space formulation to fit the additional objectives would be a recommended.

Additionally, a heavier focus should remain on proactive disruption management as currently most papers take a reactive approach and only minimal research includes a proactive perspective such as this work and Becking (2024), Vos (2024), Lee et al. (2022), and Zang et al. (2024).

Alternative algorithmic approaches could be taken instead of the current DQN RL method. Algorithms such as PPO could help to find an optimal policy. Moreover, it is advised to further dive into the state-space formulation of Model 2 as it did yield positive outcomes and could have good effects applied to further testing and experimentation. This could be done by focusing on perhaps representing the flight schedule as a graph, enabling a more natural problem representation. One could also try to improve upon the temporal representation and divide the columns in Model 2 into even smaller time intervals. The conflict probability evolution could be worked out so that it is clearer to the agent which aircraft and which flights are conflicted.

Finally, the training methodology could be enhanced by starting with very simple scenarios and gradually increasing the difficulty. Furthermore, a specific direction that can be taken from this research is to differentiate between 'good'and 'bad' cancellations. Currently all cancellations are penalised. However, making this distinction could benefit and enhance not only the reward function design but also the overall performance.

9 Conclusion

The primary objective of this thesis was to optimise a reinforcement learning formulation for the ARP that minimises the effects of disruptions and conflicts. In order to tackle this objective, the work investigated the performance of two models, Model 1 and Model 2. Furthermore, a series of hypotheses regarding state space formulation, environment types, and the underlying learning dynamics of DRL agents have been addressed.

This research has achieved meaningful progress towards the objective while also identifying significant challenges and limitations. To begin with, the comparison between Model 1's dense continuous state space and Model 2's sparse dual matrix formulation successfully showed that state space design has a impactful effects on computation time, with a 40% increase for Model 2. This is vital for real-life airline operations and logistics. Furthermore, it demonstrated an improved resolution rate and higher confidence in decision making.

Next, the analysis focussed on comparing three environment types: proactive, myopic, and reactive. It was found that probabilistic information is indispensable and leads to significantly better outcomes. The proactive environment generally outperformed the other strategies. This furthermore showed that early conflict indication helps decision making. However, a significant and unexpected observation was made during the learning trend and reward analysis. Both models show a persistent negative slope for all environment types, highlighting the DRL agent's inability of finding an optimal policy. Properties such as cancellations, whose rewards should decrease over time, started to increase instead. An investigation of the sensitivity of the reward function revealed that the agent correctly interprets reward signals, underlining that it is not the cause of the negative trend. However, although it correctly identifies them, it is unable to successfully harvest them or establish a useful policy. This is furthermore underlined by the conspicuous observation that over 55% of initial conflicts remained unresolved, underling that current approaches fail to come up with effective resolution methods.

A critical finding is the paradox of the agent's behaviour during exploration versus exploitation. Whereas theoretically it was expected that the agent would perform better during high exploitation, when $\epsilon = 0.05$, it actually performed worse. This outcome coincides with that of the negative learning trend. The key performance indicators such a tail swaps showed a clear decrease across episodes. It underlines that once exploitation is reached, the model seems to be plagued by a Q-value collapse. The reason for this miscalibration is likely due to the saturation of smaller negative rewards which causes the model to be unable to distinguish between good and bad actions.

Moreover, a sensitivity and robustness analysis was performed, highlighting why the implementation of DRL for the aviation sector might be difficult. Model 2 showed high sensitivity to both, disabling critical reward components and hyperparameter changes, in this case the learning rate (α).

This work has highlighted the effectiveness of a sparser binary state space formulation as implemented in Model 2 for solving the ARP. However, the results also showed important limitations that impact the real life readiness of DRL for airline disruption management, such as computing power, problem scope, and algorithm choice.

In order to further investigate the ARP using a proactive DRL approach, several recommendation can be made for future research. The proactive environment type shall be further investigated in order to nurture its ability to positively impact the agent's strategy. Moreover, the problem scope may be enlarged by including passengers and crew. The current synthetically generated data can furthermore be partially traded for real life data in addition to considering bigger scenarios. Finally, different algorithms such as PPO can be investigated.

References

Aktürk, M. Selim et al. (2014). "Aircraft Rescheduling with Cruise Speed Control". In: *Operations Research* 62.4, pp. 829–845. DOI: 10.1287/opre.2014.1279.

Becking, Pieter (2024). "Airline Disruption Management: A proactive deep reinforcement learning approach for aircraft disruption recovery under uncertainty". Master's thesis, Department of Control and Operations, Delft University of Technology, Delft, The Netherlands.

Clausen, Jens et al. (2010). "Disruption management in the airline industry—Concepts, models and methods". In: *Computers & Operations Research* 37.5. Disruption Management, pp. 809–821. DOI: https://doi.org/10.1016/j.cor.2009.03.027.

Ding, Yida et al. (2023). "Towards efficient airline disruption recovery with reinforcement learning". In: *Transportation Research Part E: Logistics and Transportation Review* 179, p. 103295. DOI: https://doi.org/10.1016/j.tre.2023.103295.

Eggenberg, Niklaus et al. (2010). "Constraint-specific recovery network for solving airline recovery problems". In: *Computers & Operations Research* 37.6, pp. 1014–1026. DOI: https://doi.org/10.1016/j.cor.2009.08.006.

Haider, Imran et al. (2024). "Subnetwork prediction approach for aircraft schedule recovery". In: *Engineering Applications of Artificial Intelligence* 133, p. 108472. DOI: https://doi.org/10.1016/j.engappai.2024.108472.

Hassan, L. K. (Apr. 2019). "Aircraft Disruption Management: Increasing Performance with Machine Learning Predictions". Master's Thesis. Delft, Netherlands: Delft University of Technology.

Hassan, L.K. et al. (2021). "Airline disruption management: A literature review and practical challenges". In: *Computers & Operations Research* 127, p. 105137. DOI: https://doi.org/10.1016/j.cor.2020.105137.

Hondet, Gabriel et al. (Dec. 2018). "Airline Disruption Management with Aircraft Swapping and Reinforcement Learning". In: *SID 2018, 8th SESAR Innovation Days*. Salzburg, Austria.

Hu, Yuzhen et al. (2015). "Optimization of multi-fleet aircraft routing considering passenger transiting under airline disruption". In: *Computers & Industrial Engineering* 80, pp. 132–144. DOI: https://doi.org/10.1016/j.cie.2014.11.026.

Hu, Yuzhen et al. (2017). "Multiple objective solution approaches for aircraft rerouting under the disruption of multi-aircraft". In: *Expert Systems with Applications* 83, pp. 283–299. DOI: https://doi.org/10.1016/j.eswa.2017.04.031.

Huang, Zhouchun et al. (2022). "An iterative cost-driven copy generation approach for aircraft recovery problem". In: *European Journal of Operational Research* 301.1, pp. 334–348. DOI: https://doi.org/10.1016/j.ejor.2021.10.055.

IATA (Dec. 9, 2025). *Airline Profitability Stabilizes with Net Margin Expected in 2026*. Accessed: 2025-12-20. URL: https://www.iata.org/en/pressroom/2025-releases/2025-12-09-01/.

Jiang, Jianlin et al. (2025). "ADMM-based augmented Lagrangian methods for robust aircraft recovery problem considering connection time, resource capacity and maintenance flexibility". In: *Transportation Research Part E: Logistics and Transportation Review* 201, p. 104243. DOI: https://doi.org/10.1016/j.tre.2025.104243.

Kohl, Niklas et al. (2007). "Airline disruption management—Perspectives, experiences and outlook". In: *Journal of Air Transport Management* 13.3, pp. 149–162. DOI: https://doi.org/10.1016/j.jairtraman.2007.01.001.

Lee, Jane et al. (2020). "Dynamic Disruption Management in Airline Networks Under Airport Operating Uncertainty". In: *Transportation Science* 54.4, pp. 973–997. DOI: https://doi.org/10.1287/trsc.2020.0983.

Lee, Junhyeok et al. (2022). "A reinforcement learning approach for multi-fleet aircraft recovery under airline disruption". In: *Applied Soft Computing* 129, p. 109556. DOI: https://doi.org/10.1016/j.asoc.2022.109556.

Liang, Zhe et al. (2018). "A column generation-based heuristic for aircraft recovery problem with airport capacity constraints and maintenance flexibility". In: *Transportation Research Part B: Methodological* 113, pp. 70–90. DOI: https://doi.org/10.1016/j.trb.2018.05.007.

Lu, Jingxi and Xiongwen Qian (2025). "A Reinforcement Learning Approach for Initialization of Column Generation with Application to Aircraft Recovery Problem". In: Cited by: 0; All Open Access, Gold Open Access, pp. 9–13. DOI: 10.1145/3759179.3759181.

Peng, Yunfang et al. (2024). "A Study on Disrupted Flight Recovery Based on Logic-Based Benders Decomposition Method". In: *Aerospace* 11.5, p. 378. DOI: https://doi.org/10.3390/aerospace11050378.

Rashedi, Navid et al. (Jan. 2024). "A Machine Learning Approach for Solution Space Reduction in Aircraft Disruption Recovery". In: *SSRN Electronic Journal*. DOI: 10.2139/ssrn.4444548.

— (2025). "A machine learning approach for solution space reduction in aircraft disruption recovery". In: *European Journal of Operational Research* 323.1, pp. 297–308. DOI: https://doi.org/10.1016/j.ejor.2024.11.025.

Rhodes-Leader, Luke et al. (Aug. 2021). "A multi-fidelity modelling approach for airline disruption management using simulation". In: *Journal of the Operational Research Society* 73. DOI: 10.1080/01605682.2021.1971574.

Su, Yi et al. (2021). "Airline Disruption Management: A Review of Models and Solution Methods". In: *Engineering* 7.4, pp. 435–447. DOI: https://doi.org/10.1016/j.eng.2020.08.021.

Thengvall, Benjamin G. et al. (2000). "Balancing user preferences for aircraft schedule recovery during irregular operations". In: *IIE Transactions (Institute of Industrial Engineers)* 32.3. Cited by: 132, pp. 181–193. DOI: 10.1080/07408170008963891.

Vink, J. et al. (2020). "Dynamic aircraft recovery problem - An operational decision support framework". In: *Computers & Operations Research* 117, p. 104892. DOI: https://doi.org/10.1016/j.cor.2020.104892.

Vos, Hans-Wieger M. et al. (2015). "Aircraft Schedule Recovery Problem – A Dynamic Modeling Framework for Daily Operations". In: *Transportation Research Procedia* 10. 18th Euro Working Group on Transportation, EWGT 2015, 14-16 July 2015, Delft, The Netherlands, pp. 931–940. DOI: https://doi.org/10.1016/j.trpro.2015.09.047.

Vos, Willem (2024). "Anticipatory Airline Disruption Management: A model-based reinforcement learning approach to anticipatory aircraft recovery under disruption uncertainty". Master's thesis, Department of Control and Operations, Delft University of Technology, Delft, The Netherlands.

Wang, Qi et al. (2025). "Flight, aircraft, and crew integrated recovery policies for airlines - A deep reinforcement learning approach". In: *Transport Policy* 160, pp. 245–258. DOI: https://doi.org/10.1016/j.tranpol.2024.11.011.

Wu, Zhengtian et al. (2017). "Solving long haul airline disruption problem caused by groundings using a distributed fixed-point computational approach to integer programming". In: *Neurocomputing* 269, pp. 232–255. DOI: 10.1016/j.neucom.2017.02.091.

Xu, Haiwen and Songchen Han (Jan. 2016). "Weighted Time-Band Approximation Model for Flight Operations Recovery considering Simplex Group Cycle Approaches in China". In: *Mathematical Problems in Engineering* 2016, pp. 1–17. DOI: 10.1155/2016/3201490.

Zang, Haipei et al. (2024). "A proactive aircraft recovery approach based on airport spatiotemporal network supply and demand coordination". In: *Computers & Operations Research* 165, p. 106599. DOI: https://doi.org/10.1016/j.cor.2024.106599.

Zhang, Cheng (Aug. 2017). "Two-Stage Heuristic Algorithm for Aircraft Recovery Problem". In: *Discrete Dynamics in Nature and Society* 2017, pp. 1–12. DOI: 10.1155/2017/9575719.

Zhao, Ai et al. (2023). "A two-stage approach to aircraft recovery under uncertainty". In: *Journal of Air Transport Management* 111, p. 102421. DOI: https://doi.org/10.1016/j.jairtraman.2023.102421.

Żurek, Dominik et al. (2024). "RLEM: Deep Reinforcement Learning Ensemble Method for Aircraft Recovery Problem". In: *2024 IEEE International Conference on Big Data (BigData)*, pp. 2932–2938. DOI: 10.1109/BigData62323.2024.10826050.

A Algorithms

Table A.1: Training Hyperparameters for DQN Models

Parameter	Value
Learning Parameters	
Learning Rate (α)	0.0015
Discount Factor (γ)	0.995
Experience Replay	
Buffer Size	100,000
Learning Starts	1,000
Training Frequency	
Train Frequency	Every 8 steps
Target Network	
Update Interval	Every 200 steps
Update Method	Polyak (soft)
Exploration	
ϵ_{start}	1.0
ϵ_{min}	0.05
Decay Type	Exponential
Reach ϵ_{min} at	70% of timesteps
Exploration Distribution	40% random, 30% conflict-guided, 30% Q-noise

Listing A.1: Data flow across DQN network for both models (Single Step)

```
# Input: State representation (differs by model)
# Step 1: Forward Pass
Input: s (D,)  # D = 608 for Model 1, D = 1347 for Model 3
  ↓
Layer 1:  W @ s +  b    h (256,)
  ↓ ReLU
Layer 2:  W @  h +  b    h (512,)
  ↓ ReLU
Layer 3:  W @  h +  b    h (256,)
  ↓ ReLU
Output:  W @  h +  b   q (72,)

# Step 2: Action Selection (Balanced Exploration Strategy)
q = [Q(s,  a), Q(s,  a), ..., Q(s,   a)]
     = [2.3, -1.5, 5.7, 0.8, ..., 3.2]

action_mask = [1, 1, 0, 1, 0, ..., 1]  # Dynamic mask from environment
masked_q = [2.3, -1.5, -inf, 0.8, -inf, ..., 3.2]

epsilon = 0.5  # Exploration rate (decays over training)
if random() < epsilon:
    # Balanced exploration: 40% random, 30% conflict-guided, 30% Q-noise
    exploration_type = random()
    if exploration_type < 0.4:
        action = random_valid_action() # 40%: Pure random exploration
    elif exploration_type < 0.7:
        action = conflict_guided_action() # 30%: Conflict-guided exploration
    else: # 30%: Q-value exploration (noise on Q-values)
```

```
        noise = np.random.normal(0, 0.1, size=72)
        noisy_q = masked_q + noise
        action = argmax(noisy_q)
else:
    # Exploitation: best valid action
    action = argmax(masked_q)

# Step 3: Environment Step
env.step(action)   (s_next, reward, done, info)

# Step 4: Store Experience in Replay Buffer
replay_buffer.add(s, action, reward, s_next, done)

# Step 5: Training (if step % TRAIN_FREQ == 0 and step >= LEARNING_STARTS)
# Hyperparameters: TRAIN_FREQ=8, LEARNING_STARTS=1000, BATCH_SIZE=256
if step % 8 == 0 and step >= 1000:
    # Sample batch from prioritized replay buffer
    batch = [( s,  a,  r, s' ,  d), ..., (   s,    a,    r, s'   ,    d)]

    # Compute targets using target network
    targets = []
    for (s, a, r, s', d) in batch:
        if d:  # Terminal state
            target = r
        else:
            q_next = target_network(s')  # Forward pass through target network
            target = r + 0.995 * max(q_next)  # gamma = 0.995

    # Compute current Q-values using main network
    q_current = main_network(batch_states)  # (256, 72)
    q_selected = q_current[range(256), batch_actions]  # (256,)

    # Loss (Mean Squared Error)
    loss = mean((targets - q_selected)²)

    # Backpropagation
    loss.backward()
    optimizer.step()  # learning_rate = 0.001

# Step 6: Update Target Network (if step % TARGET_UPDATE_INTERVAL == 0)
# TARGET_UPDATE_INTERVAL = 200
if step % 200 == 0:
    target_network.load_state_dict(main_network.state_dict())
```

B Plots

B.1 Problem Description

Figure B.1 shows the results for a SCOPUS query with the following query:

(TITLE-ABS-KEY (”reinforcement learning” OR ”deep reinforcement learning” OR ”deep Q networks” OR ”DQN” OR ”machine learning”) AND TITLE-ABS-KEY (”airline disruption management” OR ”aircraft recovery” OR ”integrated recovery” OR ”airline recovery” OR ”disruption management” OR ”operations management” OR ”air transport management”) AND TITLE-ABS-KEY (”airline” OR ”aircraft”))

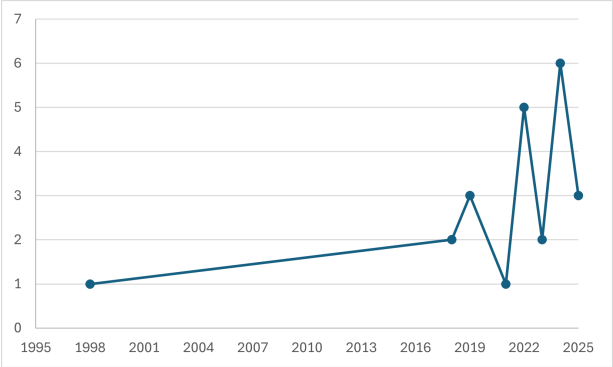


Figure B.1: Published papers for the SCOPUS query with a focus on RL

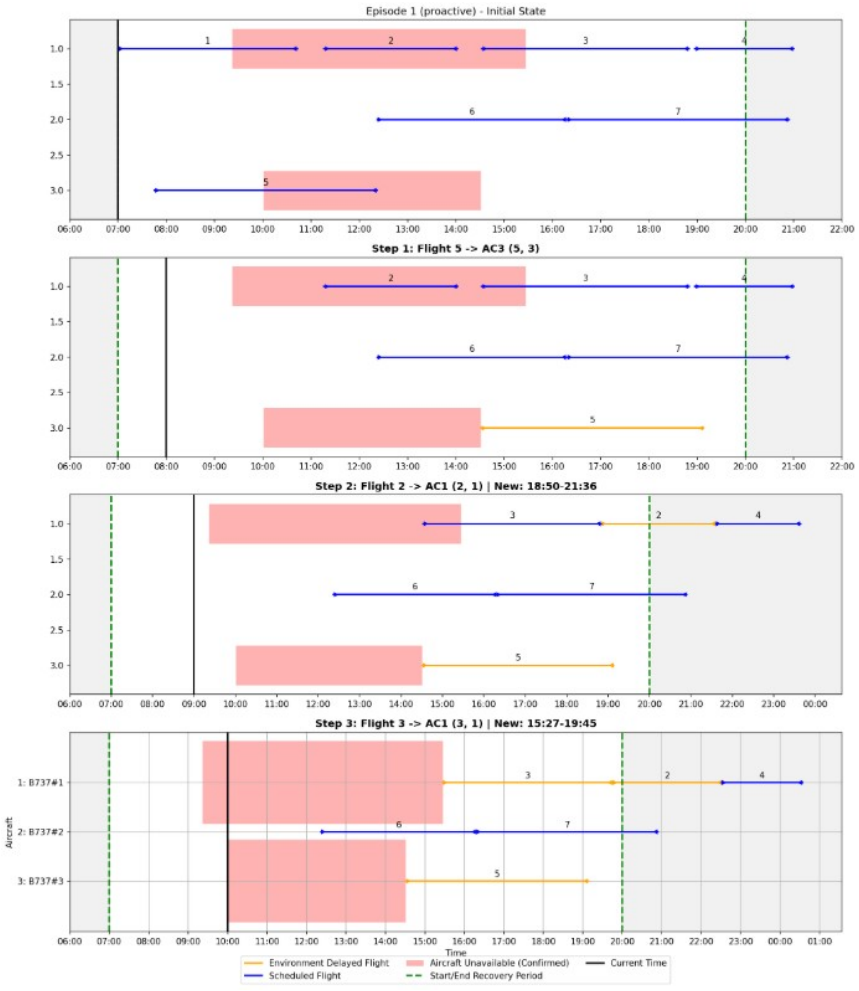


Figure B.2: Resolved Schedule: A step by Step Visualisation

B.2 Results

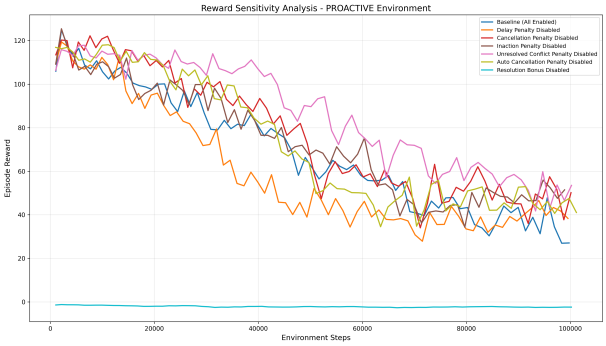


Figure B.3: Reward Sensitivity Results for Model 2 - Proactive

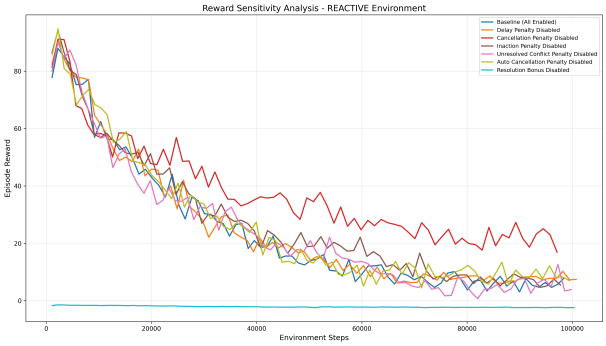


Figure B.4: Reward Sensitivity Results for Model 2 - Reactive

II. Literature Study

List of Figures

List of Tables

1

Introduction

2025 is expected to be the first year that passenger numbers will exceed the five billion mark, underlining the enormous demand for air travel (IATA 2024). The increase in popularity and accessibility of air transport has nurtured a rich economical sector, making airlines and their stakeholders a profit-oriented entity. Hence, in order to maximise efficiency and consequentially revenue, significant efforts go into creating an optimal flight schedule for aircraft, crew, and passengers. However, the schedule is often prone to disruptions such as airport closures, flight delays, mechanical problems, and the weather. This has led to the emergence of airline disruption management, an area in operational research that focuses on recovering the initial flight schedule while minimising an airline's costs, keeping profits as high as possible. How well an airline is able to respond to unexpected disruption events is crucial for its success as failing to efficiently recover the schedule could lead to passenger dissatisfaction and ultimately to high monetary losses.

Efforts to solve the recovery problem have introduced four different solution methodologies: exact optimisation methods, simple heuristics, hybrid and meta-heuristics, and reinforcement learning. The solution methods have brought about a trade-off between computation time (efficiency), and quality of solution (effectiveness) (Hassan et al. 2021). On the one hand, exact methods can find optimal solutions but take significantly longer than is often desired. On the other hand, meta-heuristics focus on finding a suboptimal solution, thus sacrificing optimality but achieving quicker convergence. Furthermore, hybrid methods, a mix of exact and heuristic solution methods, achieve a better balance but the computation time is often still too long. This is where AI and Machine Learning (ML) come in. In particular, reinforcement learning (RL) is a promising candidate for the aircraft recovery problem. It is capable to deal well with the above-mentioned trade-off conflict and hence will be the main tool used by this research to solve the recovery problem. RL is able to produce solutions much more efficiently than exact optimisation and meta-heuristics (Żurek et al. 2024). Furthermore, deep learning algorithms, in combination with RL, are extremely well equipped to deal with large amounts of data.

The base for this Master's Thesis is the MSc Thesis by Pieter Becking, completed in 2024 (Becking 2024). It focuses on a proactive deep reinforcement learning framework to solve the aircraft recovery problem (ARP). A proactive method contrasts the reactive approach often found in currently available research. Proactive disruption management is the anticipation of delays. It aims to identify and prevent unexpected events before they happen, or limit their impact once they do occur. It is crucial as even small schedule disruptions can have large effects down the line.

This paper aims to continue the development of the paper's solution methodology. The main focus thereby lies on improving reward function design to better capture the research objectives and obtain a more robust recovery schedule. Altercation of the reward functions shall minimise unnecessary recovery actions, especially inefficient flight cancellations, tail swaps, and flight delays. Furthermore, improvements shall be made regarding the introduction of uncertainty into the model. Stochastic and probabilistic methods shall be revised in order to improve real-world scenarios.

The literature report first outlines current practices and the problem statement in Chapter 2. Chapter 3 focuses on the literature review, Chapter 4 introduces crucial reinforcement learning (RL) principles, and Chapter 5 discusses the research proposal. The literature review intends to provide a comprehensive overview of airline disruption management and the entities that constitute the backbone of its sub-problems, i.e aircraft, crew, and passengers. It dives into the extensive solution methodologies and network representations. The research proposal analyses the existing research gaps, the research objectives, and the final research questions that this paper aims to answer.

Finally, the scope of the research is defined and a detailed Gantt Chart is provided to showcase the detailed structure and milestones of the thesis.

2

Problem Statement & Current Practices

The development of an initial flight schedule and the establishment of a recovery flight schedule in the case of sudden disruptions are critical components of airline operations management. These responsibilities fall upon the shoulders of the operation control centre (OCC), which monitors, coordinates, and executes the flight schedule. It furthermore handles the disruption management and makes the final call about which, and whether a recovery solution shall be implemented. It is important to note that, while many airlines are assisted by advanced computer systems and frameworks to process large amounts of data, the OCC, a human-led entity, makes the final choices. This research will focus solely on the development of recovery schedules for the aircraft recovery problem (ARP) and will take an already built synthetic flight schedule as a basis. Crew and passengers are left out.

To effectively manage airline disruption management in real-life, a recovery schedule should be generated in less than three minutes (Larsen et al. 2002). This requires a well balanced approach between computation time and solution quality. Deep reinforcement learning (DRL) is a good method in this context as it improves speed and scalability. In addition, deep neural networks (NN) can analyse large and complicated amounts data, and identify important patterns which would be impossible otherwise. The goal is to see if the proactive DRL framework developed outshines the reactive ones and if this can be improved upon by focussing on the state space formulation.

There exist many causes for disruptions and uncertainty in a system. These can have great effects on operations. Already a small delay in the initial airline schedule can lead to a ripple effect with significant consequences later on, such as a flight cancellation. This is referred to as reactionary delay. Kohl et al. (2007) identifies airport congestion, mechanical or maintenance problems, and weather events as possible disruptions. Furthermore, temporary airport closures hinder on-time performance, and taxiing in and out of the runway, as well as inefficiencies in air traffic flow management introduce additional delays into the schedule (Hondet et al. 2018). Hence, a minimal and unexpected event shall never be underestimated but rather be anticipated, underlying the necessity for a proactive reinforcement learning framework. This thesis will consider airport closures and aircraft unavailability as disruption causes.

Following disruption causes, schedule recovery actions are needed to mitigate them. Peng et al. (2024) identify four possible recovery actions: flight cancellation, flight delay, creating a flight, and reserving an aircraft. Furthermore, Vink et al. (2020) include cruise speed control and Hu et al. (2015) consider tail swaps and passenger itinerary changes. Papers on crew recovery additionally propose crew dead-heading, crew swaps and reserve crew. In real life operations, flight delays and tail swaps are the preferred choices as they are the most cost friendly. This MSc thesis will concentrate on flight cancellations, flight delays, and tail swaps as possible recovery actions.

Furthermore, there are two main network structures, point-to-point and hub-and-spoke. Most commercial airlines operate via a hub-and-spoke network as it is known to be the most-cost effective manner to link a big number of destinations (Kohl et al. 2007). Poin-to-point networks are often utilised by budget airlines (such as RyanAir and EasyJet) as they focus on destinations that are economically attractive for travellers but often without further connection possibilities.

Consequently, summarising the above mentioned points, this research will focus on solving the aircraft recovery problem using a proactive deep reinforcement learning approach. The possible recovery actions constitute cancelling a flight, delaying a flight, and tail swaps, whereas the disruption causes are airport closures. The network is presented in the form of a hub and spoke network.

3

Literature Review

Aircaft, crew, and passengers are the three essential entities of scheduling and schedule recovery. This chapter aims to outline a detailed but compact overview of all the important research covering or related to the schedule recovery problem. Although this literature review solely focuses on aircraft, it is important to note that there are many other transportation sectors that frequently need to recover from disruptions. Among these are the railway industry, the boat industry, and urban transportation such as busses (Kohl et al. 2007).

Section 3.1 gives a synopsis of crucial concepts such as network types, uncertainty modelling, and disruption causes. It furthermore provides a concise tabular overview of all papers. Section 3.2 introduces scheduling. Next, the different types of recovery problems are considered: Section 3.3 focuses on the the aircraft recovery problem (ARP), Section 3.4 on the aircraft and crew recovery problem, Section 3.5 on the aircraft and passenger recovery problem, and the integrated recovery problem is analysed in Section 3.6.

3.1. An Overview

The relevance of airline disruption management has led to a significant amount of scientific research. In real-life and real-time airline disruption management, aircraft, crew, and passengers are considered in chronological order and divided into three sub-problems: first, the aircraft recovery problem is solved, then the aircraft and crew recovery problem, and finally the aircraft, crew, and passenger recovery problem. However, most research papers only focus on one or two entities because the solution space and the problem complexity will otherwise increase too rapidly, rendering the computational power too little and the computational time too great.

In order to find scientific research papers, numerous SCOPUS searches have been done. The first query (see SCOPUS SEARCH 1) focusses on retrieving a broad range of papers. It has been adopted from Becking (2024). However, the query has been corrected as a mistake was noticed in Becking (2024)'s original formulation. The brackets around the query had been wrongly positioned i.e the query structure uses '(A AND B) OR C AND D' which SCOPUS misinterprets as ((A AND B) OR C) AND D, limiting the amount of papers found to 293. The correct search finds 429 scholarly articles using the following structure: (A AND B) OR (C AND D). Looking at Figure 3.1, it can be observed that airline disruption has steadily become a more popular research area.

SCOPUS SEARCH 1:
((TITLE-ABS-KEY (”airline recovery” OR ”aircraft recovery” OR ”crew recovery” OR ”passenger recovery” OR ”schedule recovery” OR ”integrated recovery”) AND TITLE-ABS-KEY (”airline” OR ”aircraft”)) OR (TITLE-ABS-KEY (”disruption management” OR ”irregular operations”) AND TITLE-ABS-KEY (”airline” OR ”aircraft”)))

SCOPUS SEARCH ORIGNIAL from Becking (2024):
(TITLE-ABS-KEY(”airline recovery” OR ”aircraft recovery” OR ”crew recovery” OR ”passenger recovery” OR ”schedule recovery” OR ”integrated recovery”) AND TITLE-ABS-KEY(”airline” OR ”aircraft”) OR TITLE-ABS-KEY(”disruption management” OR ”irregular operations”) AND TITLE-ABS-KEY(”airline” OR ”aircraft”))

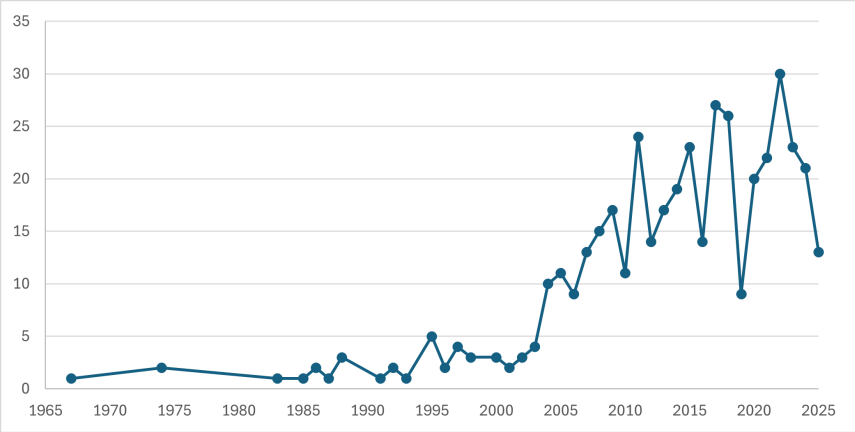


Figure 3.1: Published papers for SCOPUS query 1

The second SCOPUS search concentrates on finding papers more closely connected to the Thesis' objectives and is formulated as follows:

(TITLE-ABS-KEY (”reinforcement learning” OR ”deep reinforcement learning” OR ”deep Q networks” OR ”DQN” OR ”machine learning”) AND TITLE-ABS-KEY (”airline disruption management” OR ”aircraft recovery” OR ”integrated recovery” OR ”airline recovery” OR ”disruption management” OR ”operations management” OR ”air transport management”) AND TITLE-ABS-KEY (”airline” OR ”aircraft”))

It specifically focuses on literature related to reinforcement learning in operations research. Figure 3.2 demonstrates that, by making the search more specific, less results are obtained. Only 23 documents are found but a significant upward trend is seen. This is logical as AI and machine learning have become more relevant and influential across many industries, thus also in operations research.

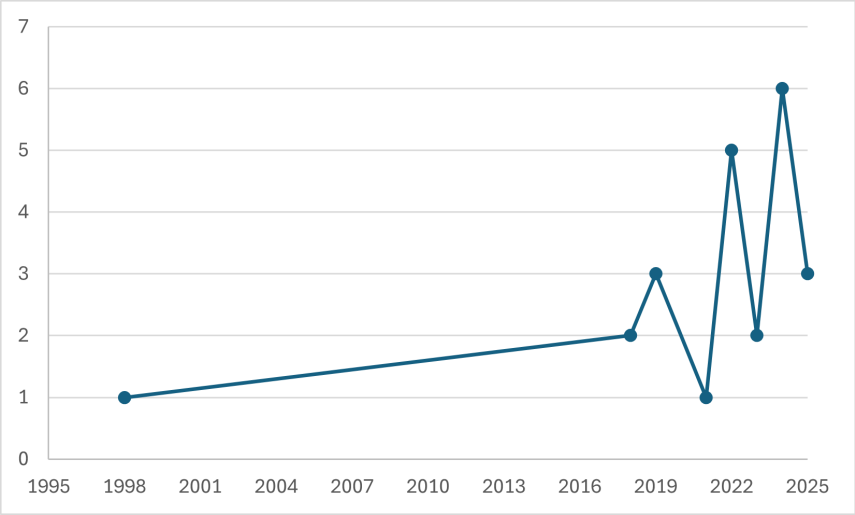


Figure 3.2: Published papers for SCOPUS query 2 with a focus on RL

3.1.1. Solution Methodologies

A trend can be observed regarding solution methodologies used to solve schedule recovery problems. They consist of either exact optimisation methods, meta-heuristics, hybrid heuristics, or reinforcement learning and machine learning techniques. Table 3.1 gives an overview of mentioned solution methods in published articles related to airline disruption management.

Table 3.1: Solution methodologies for airline disruption management

Category	Representative Methods and Algorithms
Exact Optimisation Methods	Integer Programming (MILP, IP) Conic Quadratic Programming Benders Decomposition Column Generation Rolling Horizon Frameworks Dynamic Programming LP based algorithms
Meta-Heuristics	Genetic Algorithms (GA) Simulated Annealing (SA) Greedy Randomized Adaptive Search Procedure (GRASP) Large Neighbourhood Search (LNS) Ant Colony Optimization (ACO) Tabu Search Multi-objective Genetic Algorithms (MOGA)
Hybrid Heuristics	GRASP with Ant Colony Optimization GRASP with simulated annealing Aircraft Selection combined with MILP LNS combined with Column Generation Sampling based Heuristic with MILP Decomposition based Dynamic Programming with MILP MILP guided with ML Proactive stochastic IP Column and Row Generation methods Dynamic programming with column generation
RL/Multi-Agent	Multi-Agent Systems Q-Learning RL with probabilistic delays Double Q-learning Reactive Deep RL (DRL) Proactive Deep RL (DRL) Markov Decision Process Approaches Symbiotic Simulation Frameworks (combined with heuristics) Advantage Actor-Critic (A2C)

3.1.2. Network Types

Most papers consider one of three different network types: the time-space network (also referred to as the time line network), the connection network, and the flight string network. The time-space and connection networks are both made up of nodes, flight arcs, and ground arcs. Regarding time space networks, each node represents an arrival or a departure at a certain point in time i.e. the aircraft's state. Arcs, also known as edges, are the activities between nodes or states i.e. maintenance, flights, waiting. Regarding connection networks, the nodes represent flight legs and arcs symbolise the transitions between flights for crew, passengers, and aircraft. Here, two nodes can be connected if both flight legs immediately follow each other (Hondet et al. 2018).

In contrast to time-space and connection networks, a flight string network, as employed by Rashedi et al. (2024), does not use nodes. Rather, each single flight string is predefined and represents a complete aircraft route, including flights and activities such as maintenance etc. Each flight string is a decision variable. This type of representation is useful as it foregoes the use of ground arcs and diminishes the network size (Rashedi et al. 2024). It is good for large scale and route-based recovery.

3.1.3. Uncertainty Modelling

The purpose of uncertainty modelling is to represent the randomness of disruptions by incorporating stochastic elements and probability distributions. In airline disruption management, disruption uncertainties focus on the likelihood of occurrence (the 'if?'), the timing (the 'when?'), and finally the severity (the 'how long?'). Uncertainty

modelling contrasts deterministic modelling, which treats disruptions as fixed event.

Probabilistic distributions are employed to introduce uncertainty into a model. Such distributions can be binary, Gaussian, uniform, exponential, etc. Monte Carlo simulations are also used to introduce probabilities.

Uncertainty modelling can be both fixed and dynamic. If start times and probability of occurrence are fixed, and uncertainty is used only to decide whether or not the disruption occurs at the time of sampling, it is referred to as a fixed approach. However, this undermines real-life applicability. In contrast dynamic forecasting updates the probability of occurrence with time, simulating the availability of more information closer to the start of the actual disruption. Ding et al. (2023) for example uses a dynamic probabilistic set to simulate disruptions.

Dynamic uncertainty modelling is necessary to make proactive decision making possible. This is underlined by Becking (2024) who introduces a DQN Proactive-U framework whose probabilities of disruptions are updated over time. For example at t = 0, the probability of flight 3 being delayed was 40% but at t = 6, it has risen to 60%.

3.2. Scheduling

Creating an initial schedule and recovering a schedule depend but also contrast each other. Scheduling is an essential prerequisite for the recovery problem as it forms the basis upon which the recovered timetable is built. However, whereas schedule recovery is a matter of seconds or minutes, scheduling itself is done weeks or months ahead of operations (Clausen et al. 2010).

Scheduling considers aircraft, crew, and passengers independently. First, a timetable is published. Then this is followed by fleet assignment, crew scheduling, and revenue management i.e modifications in prices and seat availability (Kohl et al. 2007). Finally, tail assignment and maintenance opportunities are added, and then the final timetable may be published. Regarding crew scheduling, Kohl et al. (2007) underlines that it can be further divided into crew pairing and crew rostering. An important note is that crews are often trained for a specific type of aircraft. Hence, crew scheduling is done per fleet (Su et al. 2021).

An effective and robust schedule that tremendously minimises disruptions, can be achieved by adding preventive self recovery measures. For example, Kohl et al. (2007) suggests adding extra slack in the schedule by using buffers alongside the minimum TAT for aircraft and crew, or having a stand by crew and aircraft (although very costly). Moreover, another option could be the assumption that cruise speed can be adaptable to the schedule if necessary. Most airlines do not operate their aircraft at max speed but rather at the most fuel efficient speed. However, while this may decrease the costs of possible delays, it does in turn increase the fuel burn. Becking (2024) states that slack can be added to a schedule by using strategies such as absorption robustness.

Another reason for the importance of implementing robust scheduling techniques is that they allow for better proactive disruption management later on (Hassan et al. 2021).

3.3. Aircraft Recovery

The ARP is most important, as the MSc Thesis report will concentrate on this part of the disruption management procedure. It was first tackled by Teodorović et al. (1984) who solved it by using a simple heuristic. Table 3.2 shows a collection of relevant literature focussing on the ARP. Next to each article's solution methodology and network type, the table further outlines the disruption types as well as the considered recovery actions.

How the ARP is modelled is crucial as it is the backbone in the code upon which the solution algorithm will run. Aircraft recovery is often modelled like a network routing problem and either presented as an arc-based model or a path-based model. Su et al. (2021) explains that arc-based models are built in a time-space network, first introduced by Thengvall et al. (2000). It consists of three types of nodes i.e. supply node, demand node, and intermediate node, and also three types of arcs i.e flight arcs, ground arcs, and copied flight arcs. Flight legs are modelled individually. Path-based models have been implemented by the works of Argüello et al. (1997), Eggenberg et al. (2010), Wu et al. (2017), Rosenberger et al. (2003), and Liang et al. (2018). Here, aircraft are assigned to a route instead of a single flight leg. They can handle more disruption situations because of the inclusion of side and legality constraints. However, this increases the problem-size significantly (Maher 2015). On the other hand, Becking (2024) and Vos (2024) do not represent their model as a network problem but rather use environment driven simulations, and function approximation. Figure 3.3 depicts a time-space network.

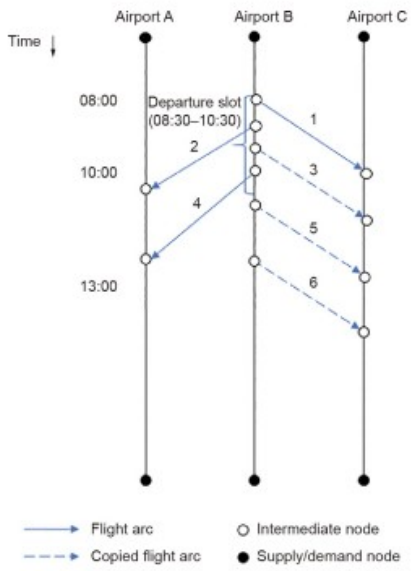


Figure 3.3: Time-space network example (Su et al. 2021)

Before the ARP can be solved, schedule disruptions need to be introduced into the model. Hondet et al. (2018) underlines how this is done. Regarding taxiing delays, a Gaussian distribution is used, for reactionary delays, which are delays that result from the ripple effect of earlier delays, the delay is exactly modelled, and for remaining delay sources an exponential distribution is used (Hondet et al. 2018).

Many works focus on exact optimisation solution methods (Wu et al. 2017; Peng et al. 2024; Aktürk et al. 2014) such as integer programming, dynamic programming, LP based algorithms or decomposition algorithms. For example, Aktürk et al. (2014) establishes the ARP as a conic quadratic mixed integer programming (exact method) and solves it using CPLEX, while considering tail swaps, flight delays, and cruise speed as recovery actions. Here 60 aircraft, 207 flights and a one day recovery period are considered.

However, difficult decision problems, or NP hard problems, require an immense computing power to obtain a merely satisfactory solution. Hence, other works instead focus on hybrid- or meta-heuristics such as benders decomposition (Petersen et al. 2012), thus settling for feasible, suboptimal solutions but with better computation times. Liang et al. (2018) solves the ARP using a hybrid heuristics method consisting of a path-based model combined with column generation. Here, the largest evaluated scenario includes 638 flights and 44 aircraft. It also considers airport capacity constraints and maintenance flexibility. Furthermore, Lin et al. (2018) reduces the complex constraints of the aircraft recovery problems into a simpler sequential decision problem. A fast VNS algorithm is used on a multi-fleet model. It shows quicker computation times and lower delays. From Table 3.2 it can be seen that meta- and hybrid heuristics are used more than exact methods. This popularity can be explained by the faster CPU times. Zang et al. (2024) proposes a hybrid heuristic approach combining linear programming with proactive DDBA.

Next to the exact and heuristic methods, reinforcement learning solution methodologies, which are agent-based models, have been on the rise. Hondet et al. (2018) for example employs a simple Q learning algorithm with aircraft swapping as its recovery action. However, no significant results were obtained. The first actual efficient experimental study using RL was done by Lee et al. (2022), who considers the aircraft reinforcement problem as a sequential decision-making operation. The paper combines RL with Q-learning algorithms and Double Q-learning algorithms in a simulated, artificial environment that represents a daily schedule. Temporary airport closures are chosen as the disruption cause and the problem involves multiple fleets and airlines, making it realistic. Both, Lee et al. (2022) and Liu et al. (2008) identify the minimisation of individual flight delays to less than 30 minutes as their primary objectives. Moreover, both also use the same flight schedule from a Taiwanese domestic airline with four airports, seventy flights, and seven aircraft per fleet from. Lee et al. (2022) does additional experiments on real life data from a South Korean airline with six airports, 20 aircraft, and 94 flights. DQL RL outperforms QL RL due to the overestimation bias that QL struggles with. Thus DQL is the most effective strategy.

Another attempt to solve the ARP was taken by Żurek et al. (2024), who used a deep reinforcement learning ensemble method (RLEM). It combines Deep Double Q-Learning (DDQL) with an Advantage Actor-Critic (A2C) approach. The experimental setup consists of two data sets: one, the same publicly available schedule as used by Liu et al. (2008) and Lee et al. (2022); second, a domestic Brazilian schedule with 26 airports, 105 flight, and 15 aircraft. The solution methodologies are contrasted to other approaches, including Q-learning, and WASP (without swapping).

Four studies take the initiative of a proactive approach and implement probabilistic modelling to anticipate future disruptions. First, Lee et al. (2020) uses a stochastic and dynamic queuing model as part of their proposed Stochastic Reactive and Proactive Disruption Management (SRPDM) framework. This approach anticipates hub airport delays. Secondly, Becking (2024) uses a proactive DRL approach. The model is compared to 3 others, a reactive heuristic, a reactive DRL agent, and a similar proactive DRL agent but without stochastic indications of possible future disruptions. Results show that the heuristic approach reaches the best solution, but that its computation time increases with problem size. DRL is the clear winner in terms of speed and scalability. Between the proactive and reactive DRL agents, the proactive approach is better as it reduces flight cancellations notably. Although uncertainty is introduced into the model when generating disruption, A drawback of this paper is that the uncertainty introduced into the framework only consists of randomising the occurrence of disruptions and not the duration of disruptions, which is static. Thirdly, Zang et al. (2024) proposes a proactive DDBA integer LP approach. However, as this is an exact method, it is limited by expansive computation times. Lastly, Vos (2024) introduces a model-based RL approach, formulated as a MDP. The aircraft recovery problem is solved by using approximate dynamic programming (ADP) in combination with Value Function Approximation (VFA). Vos employs fixed uncertainty modelling.

Certain papers, including Eggenberg et al. (2010), Thengvall et al. (2000), Vink et al. (2020), and Vos et al. (2015) use delay as a recovery action by introducing flight copies. These are built in arc copies that can be used if the initial arc in the schedule is not an option any more due to delays. However, flight copies present a significant research gap as the delay is often overestimated. There might be only a flight copy for 20 minutes and another one for 30 minutes. However, the actual delay might be only 15 minutes. Of course more options could be added but this has the drawback of drastically increasing the problem size. Huang et al. (2022) took the initiative to mitigate this problem. They did so by creating an algorithm that filters flight copies and assesses how critical their status is. However, flight copies were still in the solution.

Zhao et al. (2023) focuses on making their model realistic by introducing two sorts of uncertainty: first, the length of the disruption and secondly, the time stamp when more information becomes available. A rolling horizon approach is used to handle these uncertainties successfully.

Two papers include cruise speed control as a possible recovery action. First Lee et al. (2022) whose solution approach has been discussed above and secondly, Aktürk et al. (2014) who uses a conic quadratic optimisation approach to tackle the problem of non-linear fuel costs. The latter was the first to employ a mixed integer programming in combination to this. This model performed better than previous delay propagation solution methodologies.

A recent article, published by Rashedi et al. (2024) uses a machine learning approach during pre-processing to reduce the solution space. The actual recovery schedule however is still generated using an exact solution method, namely an exact MIP optimizer. This demonstrates that machine learning is being increasingly researched and included in the ARP.

Table 3.2: Overview of literature on the aircraft recovery problem

Paper	Network	Type	Solution Approach	Disruption Types				Recovery Actions					
				Flight Delay	Flight Canx	AC UA	Airport	Flight Delay	Flight Canx	Create Flight	Tail Swap	Reserve AC	Cruise Speed
Eggenberg et al. (2010)	Time Band	HH	Dynamic programming with column generation			✓		✓	✓		✓		
Gao et al. (2010)	Flight Strings	HH	GRASP with simulated annealing			✓	✓	✓	✓		✓		
Aktürk et al. (2014)	Time Space	EX	Conic quadratic MIP	✓				✓			✓		✓
Vos et al. (2015)	Time Space	MH	Aircraft selection heuristic with MILP	✓	✓	✓	✓	✓	✓		✓		
Xu and Han (2016)	Time Band	MH	Weighted time-band approximation with MILP			✓			✓				
Zhang (2017)	Connection	MH	Two-Stage heuristic for LOF reduction			✓	✓	✓	✓		✓		
Hu et al. (2017)	Connection	MH	neighbourhood search heuristic with ⊠-constraints			✓		✓	✓		✓		
Wu et al. (2017a)	Connection	EX	Distributed fixed-point integer programming			✓		✓	✓		✓		
Wu et al. (2017b)	Connection	EX	Distributed fixed-point integer programming	✓		✓	✓	✓	✓		✓		
Wu et al. (2017c)	Connection	EX	Distributed fixed-point integer programming				✓	✓	✓		✓		
Hondet et al. (2018)		RL	Q-Learning RL with probabilistic delays	✓							✓		
Lin and Zhong (2018)		MH	Sequential decision algorithm				✓	✓			✓		
Rhodes-Leader et al. (2018)	Time Space	O	Symbolic simulation with IP			✓		✓	✓		✓		
Liang et al. (2018)	Connection	HH	Column generation	✓		✓	✓	✓	✓		✓		
Hassan (2019)	Time Space	HH	MILP guided with machine learning	✓		✓	✓	✓	✓		✓		
Lee et al. (2020)	Time Space	HH	Proactive stochastic integer programming				✓		✓	✓		✓	✓
Wen et al. (2022)	Connection	HH	Column generation										
Lee et al. (2022)	Time Space	RL	Q-Learning and Double Q-learning				✓	✓			✓		
Huang et al. (2022)	Time Space	MH	Iterative Cost Driven-Copy Generation algorithm	✓	✓		✓	✓	✓		✓		
Rhodes-Leader et al. (2022)	Time Space	HH	deterministic MIP & Simulation			✓		✓	✓				
Zhao et al. (2023)	Time Space	HH	2-Step Multi-commodity network under uncertainty, RH				✓	✓	✓		✓		
Zang et al. (2024)	Time Space	HH	Proactive DDBA Integer Linear Programming	✓			✓	✓	✓		✓		
Peng et al. (2024)	Time Space	EX	Using LBBD to establish MIP	✓			✓	✓	✓	✓		✓	
Haider et al. (2024)	Time Space	HH	Multiple machine learning techniques			✓		✓	✓		✓	✓	
Zurek et al. (2024)		DRL	Deep Double Q-Learning (DDQL) & Advantage Actor-Critic (A2C)				✓	✓			✓		
Rashedi et al. (2025)	Time-Space	EX	ML preprocessing followed by exact MIP optimizer	✓	✓	✓		✓	✓		✓		

*AC UA: aircraft unavailability, Airport: Airport Closure, HH: Hybrid heuristic, EX: exact method, MH: meta heuristic, O: other, RL: reinforcement learning, MIP: mixed integer programming, GRASP:Greedy Randomized Adaptive Search Procedure, MILP: mixed integer linear programming, LOF: line of flights, DDBA: Decision-Decomposition-Based Algorithm, LBBD: Logic-Based Benders Decomposition

3.4. Aircraft and Crew Recovery

Regulations become a challenge when crew reallocation is added to the ARP. As humans are involved, strict labour rules from governing agencies must be followed. The goal us to reassign available crews with as low a cost as possible. The problem is often presented as a single fleet network (Su et al. 2021).

Hartong (2025) addresses the aircraft and crew recovery problem (ACRP) using a multi-objective Benders decomposition. Each aircraft tail and crew member is represented individually by using a multi-commodity flow network, comprising of arc and nodes. By focussing on individual crew members, much more flexibility is possible compared to traditional crew pairing methods. The nodes represent flights, maintenance opportunities, and transitions. The problem is divided into a master- and sup-problems which makes it possible to handle multiple objectives such as maximising on-time performance, minimising recovery costs, and minimising flight cancellations. Hartong (2025) furthermore uses pre-processing techniques to reduce the network size. The limitations of this paper mainly relate to scalability as the problem size grows exponentially with the number of aircraft. The second limitation relates to the fact that the paper describes a deterministic approach, which is not realistic. Hence, stochasticity should be

introduced in future research to promote uncertain and sudden disruption scenarios.

Table 3.3: Overview of literature on aircraft and crew recovery.

Paper	Network	Type	Solution Approach	Recovery actions								
				Flight Delay	Flight Canx	Create Flight	Tail Swap	Reserve AC	Cruise speed	Crew dead-heading	Crew swaps	Reserve crew
Zhang et al. (2015)	Time Space	MH	Two stage heuristic	✓	✓	✓	✓	✓		✓	✓	✓
Zhong et al. (2024)	Flight String	MH	Multi-objective swarm optimizer	✓	✓				✓		✓	✓
Hartong (2025)	MCFN	EX	multi-objective Benders decomposition, MILP	✓	✓		✓	✓				

*MH: meta heuristic, EX: exact methods, MILP: mixed integer linear programming

3.5. Aircraft and Passenger Recovery

The main costs to airlines due to disruptions to an initial schedule stems from the passengers (Su et al. 2021). Hence, passenger-focussed recovery is a promising research area.

Including passenger reassignment adds another layer of complexity to the problem. Marla et al. (2017) does so effectively, using conic quadratic formulations, part of non-linear optimisation, as the exact solution method. Furthermore, heuristic solution methodologies, including large neighbourhood search (LNS) and greedy randomised adaptive search procedure (GRASP), have been chosen to reduce computation time while still getting a feasible solution (Bisaillon et al. 2011; Hu et al. 2016) Vink et al. (2020) dynamically solves the aircraft and passenger recovery problem using a heuristic, which means that subsequent disruptions are handled based on the previous solutions (Lee et al. 2022).

In order to improve future research, a heavier focus should lie on passenger preferences. Data mining could be used to analyse this and further passenger information, which may ultimately lead to better optimisation processes (Su et al. 2021).

Table 3.4: Overview of literature on Aircraft and Passenger Recovery (AP)

Paper	**Network**	**Type**	**Solution Approach**	**Disruption Types**				**Recovery Actions**				
				Flight Delay	Flight Canx	AC UA	Airport	Flight Delay	Flight Canx	Tail Swap	Pax IC	Cruise Speed
Jafari et al. (2010)	Connection	EX	Rolling horizon time framework with MILP	✓		✓		✓	✓	✓	✓	
Zegordi and Hessamedin (2010)		MH	Ant Colony Optimization	✓		✓		✓	✓	✓		✓
Bisaillon et al. (2011)	Time Space	MH	Large Neighboorhood Search	✓		✓	✓	✓	✓	✓	✓	
Mansi et al. (2012)	Time Space	HH	Math heuristics	✓	✓	✓	✓	✓	✓	✓	✓	
Jozefowiez et al. (2013)	Connection	MH	Heuristic based on shortest path	✓	✓	✓	✓	✓	✓	✓	✓	
Sinclair et al. (2014)	Time Space	MH	Large neighbourhood search	✓	✓	✓	✓	✓	✓	✓	✓	
Hu et al. (2015)	Time Band	EX	Integer programming			✓		✓	✓	✓	✓	
Hu et al. (2016)	Time Space	MH	GRASP heuristic	✓		✓		✓	✓	✓	✓	
Sinclair et al. (2016)	Time Space	HH	LNS and Column Generation	✓	✓	✓	✓	✓	✓	✓	✓	
Zhang et al. (2016)	Time Space	MH	Sequential three stage heuristic	✓	✓	✓	✓	✓	✓	✓	✓	
Marla et al. (2017)	Time Space	EX	Rolling Horizon MILP	✓				✓	✓	✓		✓
Santos et al. (2017)		EX	Rolling horizon time framework with MILP	✓				✓				
Yang and Hu (2019)	Time Space	MH	Multi-Objective genetic algorithm			✓	✓	✓	✓	✓	✓	
Vink et al. (2020)	Time Space	HH	MILP combined with fleet selection heuristic	✓	✓	✓	✓	✓	✓	✓		✓
Hu et al. (2021)	Connection	HH	Multi-Directional Stochastic Variable Neighbourhood Search	✓	✓			✓	✓	✓	✓	
Yetimoğlu and Aktürk (2021)	Connection	HH	Math-heuristic			✓			✓	✓		✓
Sun et al. (2022)	Time Space	EX	CPLEX			✓		✓	✓	✓	✓	
Huang et al. (2023)	Flight String	HH	Machine learning w. column-and-row generation	✓		✓	✓					
Wandelt et al. (2023)	Connection	HH	heuristic VNS & MIP	✓	✓		✓	✓	✓	✓	✓	

*AC UA: aircraft unavailability, Airport: Airport Closure, HH: Hybrid heuristic, EX: exact method, MH: meta heuristic, MIP: mixed integer programming, GRASP:Greedy Randomized Adaptive Search Procedure, MILP: mixed integer linear programming,

3.6. Integrated Recovery

The integrated recovery problem is the most computationally intensive, as it combines all three entities, i.e. aircraft, crew, and passengers.

In 1997, Clarke (Clarke 1997) described several possible solution methodologies for the integrated airline recovery problem based on network flow theory. Table 3.5 underlines that the author represents the problem in the form of a time-space network. The solution methodologies are split between heuristic procedures and optimal approaches underlining the ever returning trade-off between the computational time and the optimality of the solution. The greedy heuristic solution approach employs the shortest path and out-of-killer (OFK) minimum cost flow algorithms, whereas the optimisation solution procedure focuses on solving an integer programming problem using column generation (CG) followed by a branch and bound approach (Clarke 1997). However, the paper does not provide proof or testing of the suggested methods.

The Descartes project, called to life in 2003, was a collaborative initiative by British Airways and the TU Denmark aiming to solve the integrated recovery problem. So-called dedicated solvers were used to tackle the sub-problems (aircraft, crew, and passenger recovery) independently of each other, before integrating the individual solutions. Considering all three entities individually is known as sequential solving. The independent or dedicated aircraft recovery (DAR) problem was modelled as a time-line network and solved using a local search heuristic (Kohl et al. 2007).

Similarly, Bratu et al. (2006) also use sequential solving. They suggest two models that both focus heavily on the passengers, using the OPL Studio tool and an OCC simulator for multi day scenarios (Bratu et al. 2006). However, later research steps away from sequential solving as it can possibly miss advantageous trade-offs between resources and instead focuses on simultaneous solving.

Petersen et al. (2012)'s modelling approach is path-based and splits the problem into a master problem which focusses on high level decisions i.e. which flights to cancel/delay, and sub-problems which are more detailed i.e. assignment of specific crew and aircraft. The paper describes Benders decomposition and CG as the chosen solution methods with the underlying reasoning that it deals well with complexity. Similarly to Petersen et al. (2012), Maher (2015) utilises a path-based model. The author solves the problem by column and row generation for real-time feasibility. The framework is tested on a point-to-point network with 262 flights and on a hub-and-spoke network with 441 flights. Both instances are solved in 427 seconds and in 400 seconds, respectively (Maher 2015). Although both, Petersen et al. (2012) and Maher (2015) incorporate passengers, they do not model the reassignment of them explicitly. Instead, they focus on penalising disruptions affecting passenger in the objective or cost function.

Similarly to (Clarke 1997), Bratu et al. (2006), and Maher (2015), exact optimisation solution methods have been proposed by Cadarso et al. (2022) and Arıkan et al. (2017). The former includes passenger behaviour preferences in their model and uses cruise speed as a decision variable. The method has been tested on a major European airline, concluding that the incorporation of passengers is crucial for better airline disruption recovery. The latter includes each disruption type as well as almost each recovery action. However, computation times were too long for real-life implementation.

Evler et al. (2021) extends the Resource-Constrained Project Scheduling Problem (RCPSP), formulating the integrated recovery problem as a MILP. It is solved using Gurobi within a rolling horizon framework The paper considers various recovery actions including stand reallocations, quick turnarounds, and transfer passenger prioritization. The paper identifies their use of an exact solution method as a limit of their research and suggest the use of a meta-heuristic approach to tackle large scale scenarios.

In 2023, Ding et al. (2023) proposed a DRL-bases variable neighbourhood search (VNS). Almost all disruption scenarios and recovery actions are considered as can be seen in Table 3.5, which underlines its possible real-world adaptability. The final results are compared to exact solvers, including CPLEX, and it is shown that only a 1.5% gap exists, concluding that the solution is very satisfactory. Additionally, computation time is significantly better than for the CPLEX method. Furthermore, RL has also been used by Wang et al. (2025). The authors propose an attention-based end-to-end deep reinforcement learning framework was proposed by and depict the problem as a Markov Decision Process (MDP). Compared to heuristic methods such as VNS, the authors confirmed that their DRL approach showed strong performance in computation time.

Table 3.5: Overview of literature on integrated recovery (Modified)

Paper	Network	Type	Solution Approach	Disruption Types				Recovery Actions							
				Flight Delay	Flight Canx	AC UA	Airport	Flight Delay	Flight Canx	Create Flight	Tail Swap	Reserve AC	Cruise Speed	Crew SwapReserve	Pax IC
Teodorović and Stojković (1995)	Connection	HH	Heuristic based on FIFO principle, & DP	✓		✓	✓								
Clarke (1997)	Time Space	EX, MH	Integer programming with feasibility constraints	✓	✓			✓	✓		✓				
Bratu and Barnhart (2006)	Time Space	EX	PDM and DPM models with OPL Studio	✓				✓	✓			✓			
Peterson et al. (2012)	Time Space	HH	Benders decomposition				✓	✓	✓		✓			✓	✓
Maher (2015)	Conneciton	EX	Column and row generation				✓	✓	✓	✓	✓		✓	✓	
Zhu et al. (2016)	Time Space	MH	Sampling-based algorithm framework				✓								
Arikan et al. (2017)	Connection	EX	Conic Quadratic MILP	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓
Evler et al.(2021)	Time Space	EX	RCPSP with MILP - rolling horizon	✓	✓	✓			✓	✓	✓	✓		✓	✓
Cadarso and Vaze (2022)	Time Space	EX	Passenger-centric MINP	✓	✓		✓				✓		✓		✓
Evler et al. (2022)			Rolling Horizon framework	✓		✓		✓	✓		✓				
Ding et al. (2023)	Flight Connection	RL	DRL-based variable neighborhood search	✓	✓		✓	✓	✓		✓		✓	✓	✓
Wang et al. (2025)	Flight Network	RL	Attention-based DRL	✓	✓	✓	✓	✓	✓				✓	✓	

**AC UA: aircraft unavailability, Airport: Airport Closure, HH: Hybrid heuristic, EX: exact method, MH: meta heuristic, RL: reinforcement learning, MILP: mixed integer linear programming, RCPSP: Resource-Constrained Project Scheduling Problem, PDM: Passenger Delay Metric, DPM: Disrupted Passenger Metric, FIFO: first in, first out, DP: dynamic programming

4

Principles of Reinforcement Learning

This chapter outlines important concepts of reinforcement learning (RL) and deep reinforcement learning (DRL). It analyses how RL has been used in operations research and more specifically in airline disruption management. Furthermore, it outlines the MDP (Markov Decision Process) and covers the importance of reward functions.

4.1. Basics of Reinforcement Learning

Reinforcement learning is a machine learning computational approach that trains agents by interacting with an environment and making decisions in it while in a specific state. The decisions are referred to as actions. To learn how to behave, the agent is rewarded for each action it takes and the environment and states are consequently updated. The goal of a RL model is to maximise the total cumulative reward over many executions and to maximise the total return. Return is the sum of discounted rewards (Lee et al. 2022). To decide what actions should be taken, the model defines so-called policies. A policy maps the probabilities of all possible actions that can be taken in the current state. A fitting analogy is to consider the agent as the OCC (airline operators) who can take decisions about whether to swap, cancel, or delay a flight. The environment would then be the disrupted fleet. A reinforcement learning framework can mathematically be represented by a finite Markov Decision Process (MDP).

Two important parameters in RL are the discount factor, λ, and the learning parameter, α. The discount factor denotes the range of sight of the agent. A bigger λ means that more delayed rewards are taken into account. In contrast, if it is zero the parameter only takes immediate rewards into account. The learning parameter, α, is needed if the transition between states is stochastic (Hondet et al. 2018).

In reinforcement learning, the trade-off between exploration and exploitation plays an important role. Exploitation focuses on choosing the most optimal action with the highest immediate reward, whereas exploration picks a random action to avoid local minima. Thus, the state space is explored to make sure that the optimal action is not falsely disguised as one. Multiple methods exist to deal with this trade-off, one of them being the k-armed bandit algorithm, which encourages a balance between exploration and exploitation using a mathematical expression.

4.2. Reinforcement Learning in Operations Research

There are several advantages of using RL for the ARP. To begin with, translating the real world as well as possible to a simulated environment is crucial for airlines. This involves assumptions and many conditions. RL can do this and is hence ideal for realistic scenarios. Secondly, due to the agent-based nature of RL, policies can be reused for different disruption causes, which can significantly advance the learning process. Thirdly, the reward functions, which are a fundamental part of RL, are modifiable which allows to comfortably shift the model's focus to different objectives. This makes it much more flexible than other operation research methods.

Hondet et al. (2018) aims to solve the ARP using Q-learning (QL), which is a simple RL algorithm that uses matrices and employs aircraft swapping as its recovery action. Compared to an exact optimisation method such as dynamic programming which needs to know the transition probabilities of states exactly, QL is satisfied with knowing the average probability. The paper identifies several research limits. Although the Q-learning algorithm has the advantage of being simple, it becomes increasingly inefficient and even unusable as the amount of states and possible actions increases (Hondet et al. 2018). It does not have the capability of dealing well with high dimensionality and continuous states. In order to go avoid this, the Q learning matrices can be replaced by multi-layered perceptrons (Hondet et al. 2018) as these can handle larger, continuous inputs.

Lee et al. (2022) performed the first experimental study efficiently solving the ARP using Q-learning as well as Double Q-learning (DQL) algorithms for RL. DQL multi agent RL approach helps immensely to overcome the challenges associated with scalability and complexity. Furthermore, Becking (2024), states that a benefit of DRL frameworks is, that uncertainty models can be integrated into it. This makes it possible for agents to learn policies that already take into account probabilistic events, which improves the model's reliability.

RL has been used in many other challenges in operations research such as the operational aircraft maintenance routing problem (OAMRP), the revenue management problem and the problem of demand and capacity imbalances. Ruan et al. (2021) solved the OAMRP by a Q-learing RL algorithm. Shihab et al. (2022) tackled the revenue management problem with a deep learning algorithm, involving stochastic demand and Kravaris et al. (2023) used a deep multi-agent reinforcement learning demand to handle imbalances in air traffic management. In addition, the latter used a deep deterministic policy gradient algorithm.

Regarding reward functions, (Hondet et al. 2018) minimises the final cost of delays. However, as the RL algorithm always maximizes (and not minimises) the expectation of rewards, the opposite value is taken. Equation 4.1 shows how Żurek et al. (2024) formulates their reward function for the ARP. Similar to Hondet et al. (2018), the delay should be minimised. In this particular case, if the average delay, ϕ_t is zero, the reward is one, if it is bigger than the previous time step, it is -1, if it is equal it is 0 and otherwise it is described by the Bellman equation.

$$R_t = \begin{cases} 1 & \text{if } \phi_t = 0 \\ -1 & \text{if } \phi_t > \phi_{t-1} \\ 0 & \text{if } \phi_t = \phi_{t-1} \\ \sum_{i=t+1}^{N} \frac{1}{\gamma^t \cdot R_i} & \text{otherwise} \end{cases} \tag{4.1}$$

Markov Decision Process (MDP) is the mathematical formulation of RL. Although both authors, Becking (2024) and Vos (2024), employ reinforcement learning, they each use different approaches that contrast implicit MDP vs explicit MDP, respectively. In the case of implicit MDP, the MDP is not defined ahead of time and the agent learns by trial and error through simulation, whereas for explicit MDP the components such as states, actions, transitions, are formally and mathematically defined beforehand. Regarding transitions, implicit MDP *learns* them by observing the effects of its actions, and explicit MDP *knows* the transition probabilities beforehand. Explicit modelling is common in model-based RL, whereas implicit MDP is common in model-free RL such as for example for DQN. Using an explicit formulation is beneficial as it is well interpretable and well suitable for analysis. However, it is limited when it comes to larger stochastic environments. The implicit is beneficial for flexibility and scalability but foregoes exact modelling, needs a lot of interactions, and is harder to interpret. As an analogy, implicit MDP is learning to play chess by playing it over and over again without any prior knowledge, whereas explicit MDP gets to know the rules of chess in advance and uses probability and logic to play.

The goal when solving the MDP is to find the optimal action-value function for the state and action pairs. When using exact optimisation methods such as dynamic programming, finding the transition probabilities becomes more difficult the more states are added and the more complicated the problem gets. This is known as the 'curse of modelling'(Lee et al. 2022). A big advantage of model-free algorithms such as QL for RL is that they do not need to calculate the exact transition probabilities. Q-learning thus approximates the action value function by using Bellman equations and temporal difference learning (Lee et al. 2022). The problem with QL is however that is tends to overestimate the action value function, known as overestimation bias. In order to bypass this effect, Lee et al. (2022) uses deep Q-learning (DQL).

5

Research Proposal

Chapter 3 has reviewed the available research of airline disruption management and Chapter 4 outlined the principles of reinforcement learning. Although the existing research spans widely, improvements can be made. Hence, this chapter identifies existing research gaps in Section 5.1 and outlines the research objectives of the MSc Thesis in Section 5.2. The research questions are considered in Section 5.3 and the scope and methodology are talked about in Section 5.4. Lastly, a Gantt Chart for general planning purposes is included in Section 5.5.

5.1. Research Gaps

In total, ten research gaps have been found in all the papers that have been read. They are defined below. Note that out of this selection, the ones relevant to this Master's Thesis are discussed in Section 5.2.

1. The need for more modern and advanced RL algorithms.
2. Generalisation of RL policies across different scenarios.
3. Performance of unnecessary tail swaps or flight cancellations, indicating a need for better reward functions.
4. Lack of integrated recovery solution methodologies, reducing real-world applicability.
5. Importance of preprocessing to reduce problem size and complexity.
6. Limited use of flight arc copies in current RL-based recovery frameworks.
7. Not taking an airline's on-time performance as the main objective.
8. Insufficient focus on dynamic and stochastic disruption management.
9. Need for faster computation times for operational viability.
10. Limited robustness of the final solving framework.
11. Limited scalability of the final solution framework.

To begin with, Hondet et al. (2018) identifies the need for more advanced RL algorithms in order to come closer to real-life scenarios. Hence, a simple Q learning algorithm could be substituted by a deep neural network or an Advantage Actor Critic algorithm (A3C). An upgrade to the QL approach is the DQL RL method employed by Lee et al. (2022). Although more precise, it also has its drawbacks. The developed policies are specific to the initial airline schedule and its routes. It cannot be reused for other schedules. Hence, generalising the policies to make them applicable to different schedules should be considered in future research. Furthermore, the algorithm performed unnecessary tail swaps. This risks increasing the airline's costs as crews have to be rearranged and other resources depleted. It is advised to revise the reward functions to include the additional costs of unnecessary swaps.

Six years after Hondet et al. (2018) advocated for advanced RL methods, Żurek et al. (2024) states that the usage of modern reinforcement learning techniques is still too small, identifying a reoccurring research gap. In addition, Żurek et al. (2024) underlines that when RL is used, more straightforward approaches are chosen over deep learning methods.

Many papers only consider one or two out of the three entities, aircraft, crew, and passenger. To make a realistic disruption management system all three should be included.

Su et al. (2021) underlines that integrated recovery research should focus more on the preprocessing of input data as this can positively reduce its size, which in turn diminishes the complexity.

Many papers' objective is to minimise the airline's cost. However, Lee et al. (2022) debates that the most relevant objective should be the measure of an airline's on-time performance instead. This influences the satisfaction of customers greatly and hence should become a bigger focus for future research. Lee et al. (2022) identifies another critical research gap, namely the use of flight arc copies as a recovery tool. The problem with this is that higher precision means more flight copies, which will increase the network size too much. However, if not enough copies are generated, the solution will not be satisfactory as crucial flight arcs might be missed. Although algorithms have been implemented to assess the importance of generated flight copies, it does not mitigate the usage of them completely, which would be better.

Additionally, a heavier focus should lie on proactive disruption management. Currently, only minimal research includes it, such as Becking (2024), Vos (2024), Lee et al. (2022), and Zang et al. (2024). Instead, most papers take a reactive approach, such as Wang et al. (2025), Wu et al. (2025), and Rashedi et al. (2024), not taking into consideration possible future disruptions to the schedule.

Dynamic and probabilistic airline disruption management contrasts the static and deterministic approach often taken. Hartong (2025), although implementing a deterministic model, recognises that this limits the scope significantly. Dynamic approaches will introduce stochasticity into the framework and improve real-life like scenarios. Furthermore, this will allow for sequential decision making, i.e. resolving a disruption right when new information becomes available. It also allows already made recovery decisions to be changed again if this improves the overall result and if there is enough time to do so.

Faster computation times are and will probably always be desired in the future. Slower times, as in Aktürk et al. (2014) hinder real life applicability. Improving upon this, makes the final recovery solver more attractive to airlines. More efficient algorithms could be used to increase the rate of convergence to a satisfactory solution.

5.2. Research Objectives

The base of this Master's Thesis is the MSc Thesis by Pieter Becking, completed in 2024 (Becking 2024). More specifically, the goal is to improve upon the paper's presented results. This can be done by focussing on the altercation of reward functions to minimise unnecessary recovery actions (flight delay, flight cancellations, tail swaps). Additionally, improvements shall be made regarding the introduction of uncertainty into the model. The robustness and scalability of the model must be considered too.

Now analysing the research gaps, and keeping the necessary improvements to Becking (2024) in mind, 4 out of 10 have been chosen for this Master's Thesis. Thus, primary research gaps that will be addressed are: 3, changing the reward function to prevent unnecessary tail swaps; 7, which aims to increase on-time performance and reduce delays as the primary objective; 8, which aims to integrate more probabilistic information into the model; 10, which hopes to increase robustness of the model.

The overall research objective can be formulated as follows:

Develop an enhanced proactive deep reinforcement learning framework for the aircraft recovery problem by focussing on the state space formulation and the reward function design

The compact above mentioned research objective can be split into multiple sub research objectives:

- **RO1:** Improve the reward function for the proactive deep RL approach to improve conflict resolution.
- **RO2:** Create two distinct state space formulations to analyse computation time differences.
- **RO3:** Evaluate the performance and efficiency of two distinct state space representations by focussing on action distribution, learning trends, resolution rate, and reward evolution.
- **RO4:** Revise the reward functions and increase their flexibility in order to be applicable to individual objectives.
- **RO5:** Focus on better integration of stochastic information to create a better dynamic and probabilistic disruption management approach.
- **RO6:** Investigate three DQN environment strategies with different probabilistic disruption properties to determine which is most effective.

- **RO7:** Focus on scalability and robustness (use sensitivity analysis) to increase real-world applicability.
- **RO8:** Validate the proactive deep RL framework, focussing on action distribution
- **RO9:** Analyse the effect of the new model on the exploitation vs exploration trade-off.

5.3. Research Questions

The overall research question can be formulated as follows:

How do different state space formulations and reward function design affect the results of a proactive deep reinforcement learning framework for the aircraft recovery problem?

The research questions are summarised below. RQ1-RQ10 align with the research objectives, whereas RQ11-RQ15 are new additions.

- **RQ1:** How can the reward function be modified or improved such that reward converges to an optimal value.
- **RQ2:** How can the reward function be changed to minimise flight cancellations and ensure a high conflict resolution rate?
- **RQ3:** How do different state space representations affect learning performance and computation time?
- **RQ4:** How does state space formulation affect action distribution and decision-making strategies?
- **RQ5:** How can probabilistic uncertainties be integrated better?
- **RQ6:** How do proactive, reactive, and myopic DQN strategies compare within and across state space formulations?
- **RQ7:** Does probabilistic information (proactive environment) lead to better performance than deterministic information (myopic) or no information (reactive)?
- **RQ8:** How can the scalability and robustness of the proactive deep RL framework be enhanced to increase its real-world applicability?

5.4. Scope & Methodology

Now that the research objectives and the research questions have been discussed, the scope and methodology can be defined. The former focuses on what is included and what is not included in the research, whereas the latter outlines how the research questions and objectives will be answered.

Scope

To begin with, the time frame for the recovery schedule is one day of operations. Further, the type of disruptions are considered. These are the same as for Becking (2024) and span aircraft non-availabilities and airport closures. The Thesis focuses on solving the aircraft recovery problem and thus, logically, the recovery resource are aircraft. The chosen recovery actions comprise flight delays, flight cancellations, and move actions. The technical extent of the research paper consists of reward function tuning, state space alterations, and uncertainty modelling. Planned modifications to the reward functions in Becking (2024) are the introduction of weighted delay penalties, costs for flight cancellations, and costs for inefficient tail swaps. The exploration vs exploitation trade-off of the framework will also be revised. Furthermore, Table 5.1 shows the key performance indicators that will be considered.

A synopsis of the scope is given here:

1. The simulated environment represents the flight schedule and comprises one day of operations.
2. The disruption causes are:
    - aircraft non-availabilities
    - airport closures
3. The recovery actions considered are:
    - Flight cancellations,
    - Tail swaps, and
    - Flight delays.

4. Delays are introduced into the model using probability distributions but are restricted to be below three hours, in accordance with passenger compensation regulations (European Commission 2004).
5. Only one fleet type is considered, and the turnaround time (TAT) is assumed identical for each flight.
6. Aircraft balance is enforced: the number of aircraft at each airport must be equivalent at the start and end of the day (Lee et al. 2022).
7. Reward functions can be modified by tuning parameter, and introducing cost-based weights.
8. The trade-off of exploitation vs exploration shall be revised.

Table 5.1: Key Performance Indicators (KPIs) used in the recovery model

KPI	Description
Cancellations	Total number of cancelled flights
Flight Delays	Total number of delayed flights
Tails Swaps	Total number of tail swaps
On-Time Performance (OTP)	Number of flights arriving within 30 or 0 minutes of scheduled time
Computation Time	Total computation time in seconds

Methodology

The methodology will include a mathematical MDP problem formulation. The research approach as mentioned previously is a proactive DRL framework. The advantages of using a DRL approach can be seen in Table 5.2.

DRL Advantage	Description
Continuous learning	DRL models can continuously update their knowledge to improve future predictions
Making proactive decisions	can take preventive actions before disruptions occur
Real time adaptation	can adjust its strategies when new data is available
Handles uncertainty well	DRL models can deal well with probabilistic information
Fast computation	DRL algorithms are often faster compared to traditional methods, which is important for real life applicability
Generalisation for unseen cases	can effectively handle unseen disruption scenarios
Good exploration	can explore different strategies before exploitation
Experience replay	can use past experiences to learn more efficiently
Fast decision making	can make near instantaneous decisions which is important for ADM

Table 5.2: Advantages of Deep Reinforcement Learning (DRL) for the ARP

In order to answer the research questions and focus on the objectives, the following steps need to be taken:

- **Step 1:** Get familiar with RL in Pytorch - learn how reward functions can be implemented and how they can be modified
- **Step 2:** Investigate the existing code of Becking (2024)
  - **Step 2a:** Set up the environment on own laptop & analyse, clean, and understand the code and framework
  - **Step 2b:** Make sure the exact method (EM) & the DRL proactive-U method run on own laptop
- **Step 3:** Extensive research on ways to alter reward functions
  - **Step 3a:** Identify the reward functions in the code, and inspect what modifications can be done
  - **Step 3b:** Modify the reward function and tune its parameters to satisfy the objectives

  - **Step 3c:** Train agent on small-scale
- **Step 4:** Extensive research and redefining of the state space formulation
  - **Step 4b:** Define different state space formulations. Create different designs if necessary and possible. Train agent on small-scale
- **Step 5:** Combine state space and reward alterations
- **Step 6:** Implement reward function design changes on large-scale problem
- **Step 7:** Implement state space modelling improvements on large-scale problem
- **Step 8:** Train and analyse the model performance
- **Step 10:** Analyse resolution rates, action distributions, learning trend
- **Step 11:** Investigate proactive, myopic, reactive environment types
- **Step 12:** Identify Limitations
- **Step 14:** Identify ways to make the model more robust

5.5. Planning and Resources

5.5.1. Data

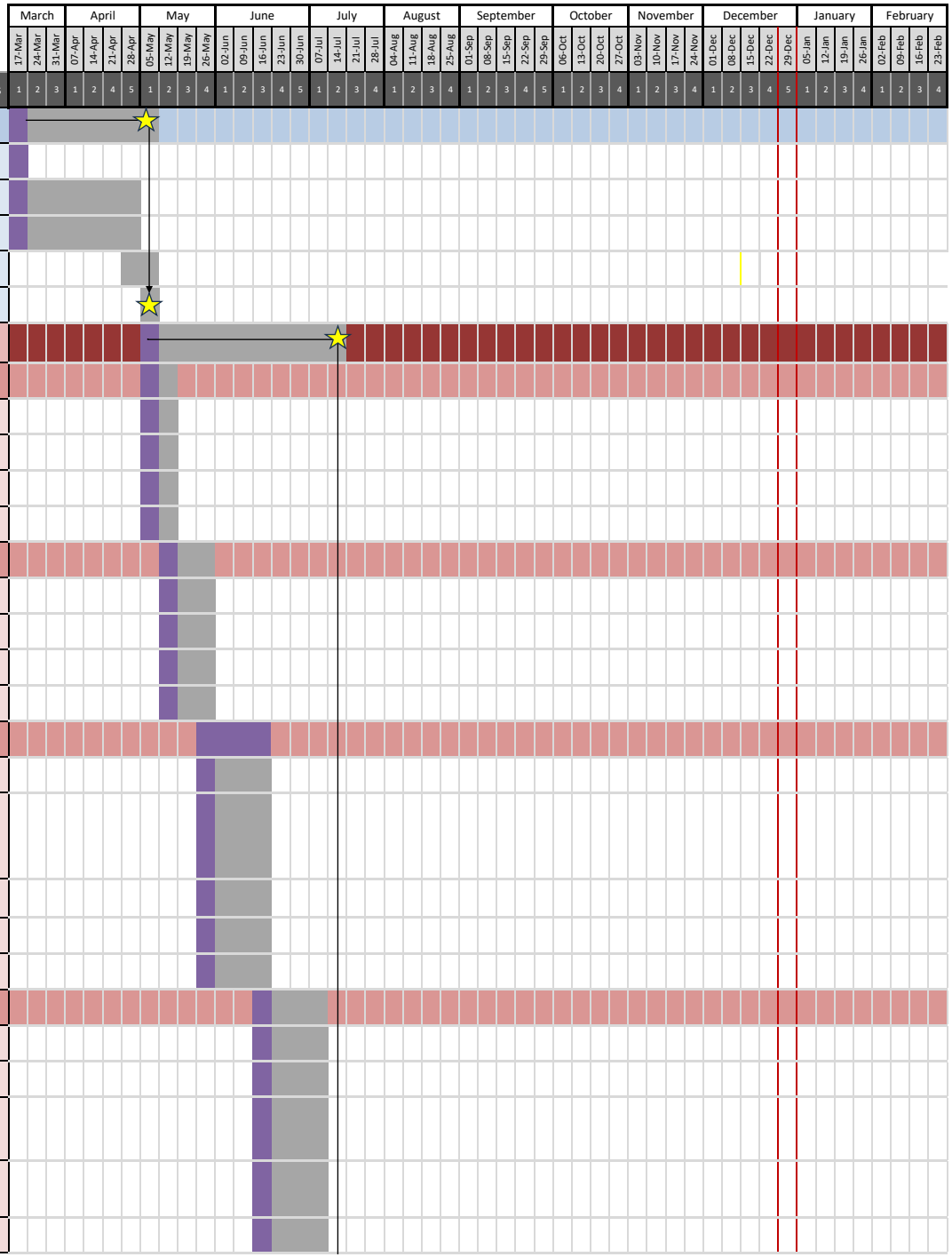
In order to modify and tune the reward functions, Becking (2024)'s code will be inspected and partly reused. Synthetic data will be generated to represent an initial airline schedule upon which the recovery schedule will be based. The code will be written in Python. Furthermore, for machine learning and (deep) reinforcement learning, StableBaseline3, and Pytorch and its libraries will be used.

5.5.2. Gantt Chart

The gantt chart below outlines the detailed planning for this MSc Thesis. It spans from the kick-off meeting in March till the Thesis defence in December.

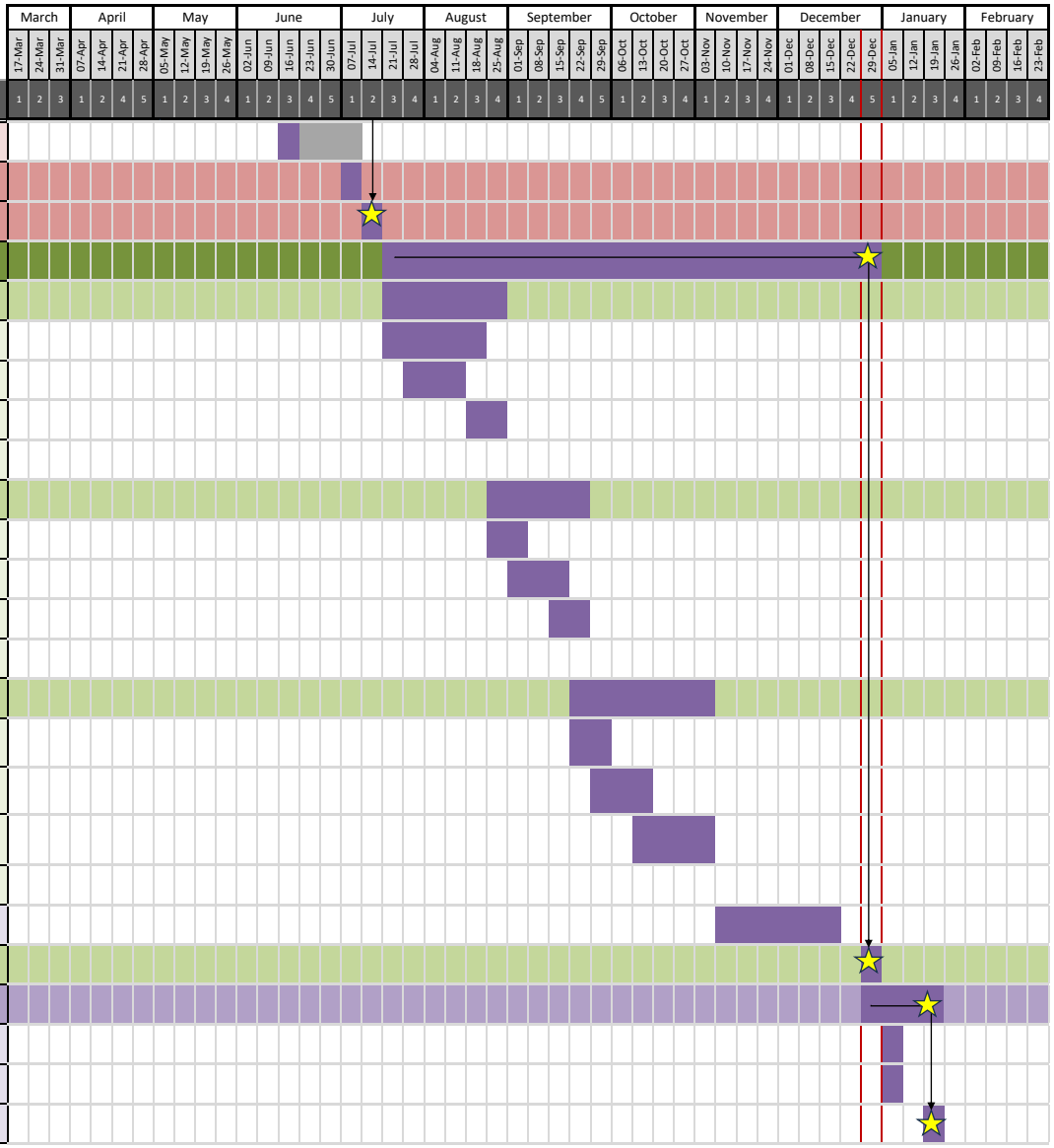
Display Week: 1

TASK	PROGRESS	START	END	DAYS
Literature Review and Research Definition	100%	20/3/25	5/5/25	47
Kick-off meeting WEEK 1	100%	20/3/25	20/3/25	
Draft Literature Review	100%	21/3/25	28/4/25	39
Draft Research Proposal	100%	21/3/25	28/4/25	39
Revised Literature Report Hand-In	100%	28/4/25	5/5/25	8
Research proposal review (Literature review for final report) WEEK 6	100%	5/5/25	5/5/25	1
Research Phase 1: Mid-Term (MT) - Experimental Setup, Reward Function Tuning & Probabilistic Formulations	100%	6/5/25	14/7/25	70
1. Getting familiar with RL in Pytorch	100%	6/5/25	13/5/25	8
Setting up Pytorch	100%	6/5/25	13/5/25	8
Experimenting with Pytorch	100%	6/5/25	13/5/25	8
Learn how reward functions work, how they can be modified	100%	6/5/25	13/5/25	8
Implement a small scale deep RL model to g+B19:B22et acquainted with it	100%	6/5/25	13/5/25	8
2. Analysis of existing Code and RL Environment	100%	13/5/25	27/5/25	15
Setting up the environment on own laptop	100%	13/5/25	27/5/25	15
Analyse, clean, and understand the code and framework	100%	13/5/25	27/5/25	15
Decide what to keep, what to change (e.g. state space, etc.)	100%	13/5/25	27/5/25	15
Make sure the EM & DRL proactive-U methods run	100%	13/5/25	27/5/25	15
3. Running existing code & creating aircraft schedules		28/5/25	18/6/25	22
Create own scenarios/datasets for training & testing.	100%	28/5/25	18/6/25	22
Create a baseline for comparison - possibly create multiple branches to compare the new SSF (state-space-formulation) & RFF (reward function formulation) with the og ones	100%	28/5/25	18/6/25	22
Be able to recreate Beckings figures for possible baseline comparisons	100%	28/5/25	18/6/25	
MT GOAL 1: create a baseline for comparison & create scenarios for testing/training	100%	28/5/25	18/6/25	22
NOTE: Testing will be adressed in Research Phase 2	100%	28/5/25	18/6/25	22
4. Research on the state-space formulation	100%	19/6/25	8/7/25	20
Research how the state-space is currently implemented	100%	19/6/25	8/7/25	20
Come up with a better formulation	100%	19/6/25	8/7/25	20
analyse all the methods & functions that use the state space matrix in any way and create pseudo code based on the new SSF	100%	19/6/25	8/7/25	20
MT GOAL 2: Find all methods connected to the SSF and write down pseudo code/ logic on how to change them using the new SSF	100%	19/6/25	8/7/25	20
NOTE: Testing will be adressed in Research Phase 2	100%	19/6/25	8/7/25	20



Display Week: 1

TASK	PROGRESS	START	END	DAYS
Prepare Mid-term	100%	19/6/25	8/7/25	20
Mid-term Deliverable	100%	8/7/25	8/7/25	1
Mid-term Review ***WEEK 17***	100%	17/7/25	17/7/25	1
Research Phase 2: Thesis Draft (TD) - SSF implementation & RFF		21/7/25	1/1/26	165
5. Implementation of own SSF (25 days for 20 functions - holidays reducted already)		21/7/25	29/8/25	26
Modify each fuction according to the new SSF (20 functions in total)		21/7/25	24/8/25	21
Vacation 2 Weeks Luxembourg/Switzerland		1/8/25	14/8/25	14
Compare the performance of the model to the baseline with the original SSF		24/8/25	29/8/25	6
TD GOAL 1: Finish implementing SSF and run it successfully				
6. Reward Function Modifications & Implementation		30/8/25	22/9/25	24
Decide how to change the code to obtain better rewards		30/8/25	5/9/25	7
Implement the changes into the code		6/9/25	18/9/25	13
Vacation 5 days Lithuania		18/9/25	22/9/25	5
TD GOAL 2: Implement new RFF				
7. Comparison of new SSF & RFF to baseline		22/9/25	9/11/25	49
Compare the branches: is there quicker conversion than the baseline? & Create the necessary figures & tables		22/9/25	2/10/25	11
Compare the branches: is there an improvement in action choices? & Create the necessary figures & tables		3/10/25	17/10/25	15
Compare the new RFF to the baseline: better rewards? & Create the necessary figures & tables		18/10/25	9/11/25	23
TD GOAL 3: Compare new SSF & RFF to baseline & create plots/tables				
Extension for Surgery in Denmark (week 33/45-week39/45)	0%	10/11/25	20/12/25	41
Submit Thesis Draft ***WEEK 40***	0%	1/1/26	1/1/26	1
Finalisation Phase		2/1/26	25/1/26	24
Greenlight Review ***WEEK 41***	0%	5/1/26	5/1/26	1
Submit final thesis	0%	10/1/26	10/1/26	1
Thesis defense ***WEEK 44***	0%	25/1/26	25/1/26	1



References

Aktürk, M. Selim et al. (2014). "Aircraft Rescheduling with Cruise Speed Control". In: *Operations Research* 62.4, pp. 829–845. DOI: 10.1287/opre.2014.1279.

Argüello, Michael F. et al. (1997). "A GRASP for aircraft routing in response to groundings and delays". In: *Journal of Combinatorial Optimization* 1.3, pp. 211–228. DOI: 10.1023/A:1009772208981.

Arıkan, Uğur et al. (2017). "Flight network-based approach for integrated airline recovery with cruise speed control". In: *Transportation Science* 51.4, pp. 1259–1287. DOI: 10.1287/trsc.2016.0716.

Becking, Pieter (2024). "Airline Disruption Management: A proactive deep reinforcement learning approach for aircraft disruption recovery under uncertainty". Master's thesis, Department of Control and Operations, Delft University of Technology, Delft, The Netherlands.

Bisaillon, S. et al. (2011). "A large neighbourhood search heuristic for the aircraft and passenger recovery problem". In: *4OR - A Quarterly Journal of Operations Research* 9.2, pp. 139–157. DOI: 10.1007/s10288-010-0145-5.

Bratu, Simina and Cynthia Barnhart (2006). "Flight operations recovery: New approaches considering passenger recovery". In: *Journal of Scheduling* 9.3, pp. 279–298. DOI: 10.1007/s10951-006-6781-0.

Cadarso, Luis and Vikrant Vaze (2022). "Passenger-Centric Integrated Airline Schedule and Aircraft Recovery". In: *Transportation Science* 57.3, pp. 813–837. DOI: https://doi.org/10.1287/trsc.2022.1174.

Clarke, Michael D. D. (1997). "An introduction to the airline recovery problem". Includes bibliographical references (leaves 90–99). M.S. thesis. Massachusetts Institute of Technology, Sloan School of Management.

Clausen, Jens et al. (2010). "Disruption management in the airline industry—Concepts, models and methods". In: *Computers & Operations Research* 37.5. Disruption Management, pp. 809–821. DOI: https://doi.org/10.1016/j.cor.2009.03.027.

Ding, Yida et al. (2023). "Towards efficient airline disruption recovery with reinforcement learning". In: *Transportation Research Part E: Logistics and Transportation Review* 179, p. 103295. DOI: https://doi.org/10.1016/j.tre.2023.103295.

Eggenberg, Niklaus et al. (2010). "Constraint-specific recovery network for solving airline recovery problems". In: *Computers & Operations Research* 37.6, pp. 1014–1026. DOI: https://doi.org/10.1016/j.cor.2009.08.006.

European Commission (2004). *Regulation (EC) No 261/2004 on compensation and assistance to passengers in the event of denied boarding and of cancellation or long delay of flights*. OJ L 46, 17.2.2004, pp. 1–8.

Evler, Jan et al. (2021). "Airline ground operations: Schedule recovery optimization approach with constrained resources". In: *Transportation Research Part C: Emerging Technologies* 128, p. 103129. DOI: https://doi.org/10.1016/j.trc.2021.103129.

Hartong, David A. (2025). "A multi-objective row-generation approach to the integrated aircraft and crew recovery problem". Master's thesis, Delft University of Technology, Delft, The Netherlands.

Hassan, L.K. et al. (2021). "Airline disruption management: A literature review and practical challenges". In: *Computers & Operations Research* 127, p. 105137. DOI: https://doi.org/10.1016/j.cor.2020.105137.

Hondet, Gabriel et al. (Dec. 2018). "Airline Disruption Management with Aircraft Swapping and Reinforcement Learning". In: *SID 2018, 8th SESAR Innovation Days*. Salzburg, Austria.

Hu, Yuzhen et al. (2015). "Optimization of multi-fleet aircraft routing considering passenger transiting under airline disruption". In: *Computers & Industrial Engineering* 80, pp. 132–144. DOI: https://doi.org/10.1016/j.cie.2014.11.026.

Hu, Yuzhen et al. (2016). "Integrated recovery of aircraft and passengers after airline operation disruption based on a GRASP algorithm". In: *Transportation Research Part E: Logistics and Transportation Review* 87, pp. 97–112. DOI: https://doi.org/10.1016/j.tre.2016.01.002.

Huang, Zhouchun et al. (2022). "An iterative cost-driven copy generation approach for aircraft recovery problem". In: *European Journal of Operational Research* 301.1, pp. 334–348. DOI: https://doi.org/10.1016/j.ejor.2021.10.055.

IATA (Dec. 10, 2024). *Strengthened Profitability Expected in 2025 Even as Supply Chain Issues Persist*. Accessed: 2025-04-24. URL: https://www.iata.org/en/pressroom/2024-releases/2024-12-10-01/.

Kohl, Niklas et al. (2007). "Airline disruption management—Perspectives, experiences and outlook". In: *Journal of Air Transport Management* 13.3, pp. 149–162. DOI: https://doi.org/10.1016/j.jairtraman.2007.01.001.

Kravaris, T. et al. (2023). "Explaining deep reinforcement learning decisions in complex multiagent settings: towards enabling automation in air traffic flow management". In: *Applied Intelligence* 53.3, pp. 4063–4098. DOI: https://doi.org/10.1007/s10489-022-03605-1.

Larsen, Jesper et al. (2002). "Disruption Management for an Airline - Rescheduling of aircraft". English. In: *Applications of Evolutionary Computing*. Ed. by S. Cagnoni et al. EvoWorkshops 2002 ; Conference date: 01-01-2002. Springer, pp. 315–324.

Lee, Jane et al. (2020). "Dynamic Disruption Management in Airline Networks Under Airport Operating Uncertainty". In: *Transportation Science* 54.4, pp. 973–997. DOI: https://doi.org/10.1287/trsc.2020.0983.

Lee, Junhyeok et al. (2022). "A reinforcement learning approach for multi-fleet aircraft recovery under airline disruption". In: *Applied Soft Computing* 129, p. 109556. DOI: https://doi.org/10.1016/j.asoc.2022.109556.

Liang, Zhe et al. (2018). "A column generation-based heuristic for aircraft recovery problem with airport capacity constraints and maintenance flexibility". In: *Transportation Research Part B: Methodological* 113, pp. 70–90. DOI: https://doi.org/10.1016/j.trb.2018.05.007.

Lin, Hongtao and Zhong Wang (2018). "Flight scheduling for airport closure based on sequential decision". In: *2018 4th International Conference on Information Management (ICIM)*, pp. 241–245. DOI: 10.1109/INFOMAN.2018.8392843.

Liu, Tung-Kuan et al. (2008). "Disruption Management of an Inequality-Based Multi-Fleet Airline Schedule by a Multi-Objective Genetic Algorithm". In: *Transportation Planning and Technology* 31.6, pp. 613–639. DOI: 10.1080/03081060802492652.

Maher, Stephen J. (2015). "Solving the Integrated Airline Recovery Problem Using Column-and-Row Generation". In: *Transportation Science* 50.1, pp. 216–239. DOI: 10.1287/trsc.2014.0552.

Marla, Lavanya et al. (2017). "Integrated disruption management and flight planning to trade off delays and fuel burn". In: *Transportation Science* 51.1, pp. 88–111. DOI: 10.1287/trsc.2015.0647.

Peng, Yunfang et al. (2024). "A Study on Disrupted Flight Recovery Based on Logic-Based Benders Decomposition Method". In: *Aerospace* 11.5, p. 378. DOI: https://doi.org/10.3390/aerospace11050378.

Petersen, Jon et al. (Nov. 2012). "An Optimization Approach to Airline Integrated Recovery". In: *Transportation Science* 46, pp. 482–500. DOI: 10.2307/23362876.

Rashedi, Navid et al. (Jan. 2024). "A Machine Learning Approach for Solution Space Reduction in Aircraft Disruption Recovery". In: *SSRN Electronic Journal*. DOI: 10.2139/ssrn.4444548.

Rosenberger, Jay M. et al. (2003). "Rerouting aircraft for airline recovery". In: *Transportation Science* 37.4, pp. 408–421. DOI: 10.1287/trsc.37.4.408.23271.

Ruan, J.H. et al. (2021). "A reinforcement learning-based algorithm for the aircraft maintenance routing problem". In: *Expert Systems with Applications* 169, p. 114399. DOI: https://doi.org/10.1016/j.eswa.2020.114399.

Shihab, S.A.M. and P. Wei (2022). "A deep reinforcement learning approach to seat inventory control for airline revenue management". In: *Journal of Revenue and Pricing Management* 21.3, pp. 183–199. DOI: https://doi.org/10.1057/s41272-021-00281-7.

Su, Yi et al. (2021). "Airline Disruption Management: A Review of Models and Solution Methods". In: *Engineering* 7.4, pp. 435–447. DOI: https://doi.org/10.1016/j.eng.2020.08.021.

Teodorović, Dušan and Slobodan Guberinić (1984). "Optimal dispatching strategy on an airline network after a schedule perturbation". In: *European Journal of Operational Research* 15.2, pp. 178–182. DOI: https://doi.org/10.1016/0377-2217(84)90207-8.

Thengvall, Benjamin G. et al. (2000). "Balancing user preferences for aircraft schedule recovery during irregular operations". In: *IIE Transactions (Institute of Industrial Engineers)* 32.3. Cited by: 132, pp. 181–193. DOI: 10.1080/07408170008963891.

Vink, J. et al. (2020). "Dynamic aircraft recovery problem - An operational decision support framework". In: *Computers & Operations Research* 117, p. 104892. DOI: https://doi.org/10.1016/j.cor.2020.104892.

Vos, Hans-Wieger M. et al. (2015). "Aircraft Schedule Recovery Problem – A Dynamic Modeling Framework for Daily Operations". In: *Transportation Research Procedia* 10. 18th Euro Working Group on Transportation, EWGT 2015, 14-16 July 2015, Delft, The Netherlands, pp. 931–940. DOI: https://doi.org/10.1016/j.trpro.2015.09.047.

Vos, Willem (2024). "Anticipatory Airline Disruption Management: A model-based reinforcement learning approach to anticipatory aircraft recovery under disruption uncertainty". Master's thesis, Department of Control and Operations, Delft University of Technology, Delft, The Netherlands.

Wang, Qi et al. (2025). "Flight, aircraft, and crew integrated recovery policies for airlines - A deep reinforcement learning approach". In: *Transport Policy* 160, pp. 245–258. DOI: https://doi.org/10.1016/j.tranpol.2024.11.011.

Wu, Shuai et al. (2025). "Airline recovery problem under disruptions: A review". In: *Computers & Operations Research* 175, p. 106915. DOI: https://doi.org/10.1016/j.cor.2024.106915.

Wu, Zhengtian et al. (2017). "Solving long haul airline disruption problem caused by groundings using a distributed fixed-point computational approach to integer programming". In: *Neurocomputing* 269, pp. 232–255. DOI: https://doi.org/10.1016/j.neucom.2017.02.091.

Zang, Haipei et al. (2024). "A proactive aircraft recovery approach based on airport spatiotemporal network supply and demand coordination". In: *Computers & Operations Research* 165, p. 106599. DOI: https://doi.org/10.1016/j.cor.2024.106599.

Zhao, Ai et al. (2023). "A two-stage approach to aircraft recovery under uncertainty". In: *Journal of Air Transport Management* 111, p. 102421. DOI: https://doi.org/10.1016/j.jairtraman.2023.102421.

Żurek, Dominik et al. (2024). "RLEM: Deep Reinforcement Learning Ensemble Method for Aircraft Recovery Problem". In: *2024 IEEE International Conference on Big Data (BigData)*, pp. 2932–2938. DOI: 10.1109/BigData62323.2024.10826050.