Pedro Manuel Cerveira Andrade

REINFORCEMENT LEARNING FOR PREDICTIVE AIRCRAFT MAINTENANCE

UNIVERSIDADE D COIMBRA

1 2 9 0

1 2 9 0

UNIVERSIDADE D
COIMBRA

Pedro Manuel Cerveira Andrade

REINFORCEMENT LEARNING FOR PREDICTIVE AIRCRAFT MAINTENANCE

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems, advised by Professor Bernardete Ribeiro and co-advised by Professor Catarina Silva and presented to
Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2020

Faculty of Sciences and Technology

Department of Informatics Engineering

Reinforcement Learning for Predictive Aircraft Maintenance

Pedro Manuel Cerveira Andrade

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems advised by Prof. Bernardete Ribeiro and co-advised by Prof. Catarina Silva and presented to the
Faculty of Sciences and Technology / Department of Informatics Engineering..

September 2020

UNIVERSIDADE Ð COIMBRA

This page is intentionally left blank.

This research is supported by European Union’s Horizon 2020 programme under the ReMAP project, grant No 769288.

Abstract

The optimization of aircraft maintenance has been a focal point for airlines for many years, mainly due to the high costs associated with it. Currently, most airlines adopt preventive maintenance strategies to reduce the number of unexpected failures. However, some strategies have some flaws due to the possibility of over-maintenance and the inability to predict failures. The aim of the Real-time Condition-based Maintenance for Adaptive Aircraft Maintenance Planning (ReMAP) project is to use a predictive strategy, namely Condition-Based Maintenance (CBM), to deal with these problems. The idea is to monitor the condition of components with data gathered from sensors to compute Remaining Useful Life (RUL) values, which can be used in the maintenance decision process.

This work is part of the ReMAP project, and the goal is to use Reinforcement Learning (RL) to optimize the scheduling of aircraft maintenance. In the first stage, two formulations are proposed to optimize the scheduling of checks for a specified time horizon using the Deep Q-Learning (QL) algorithm. In the second stage, the Asynchronous Advantage Actor-Critic (A3C) algorithm is used to optimize a task packaging solution, while considering relevant aircraft maintenance factors, such as manpower resources and unscheduled maintenance. Moreover, a predictive task scheduling algorithm is proposed, using prognostic information to adapt the existing maintenance plan.

The proposed algorithms in both stages are tested and validated using real maintenance data from a fleet of 51 aircraft. The quality of the maintenance plan obtained is evaluated according to several Key Performance Indicators (KPI). The results are very positive and promising, which indicates the potential of RL to solve this type of problem.

Keywords

Aircraft Maintenance, Artificial Intelligence, Predictive Maintenance, Maintenance Scheduling, Reinforcement Learning

This page is intentionally left blank.

Resumo

A otimização da manutenção de aeronaves tem sido um ponto essencial para as companhias aéreas durante muitos anos, principalmente devido aos elevados custos que lhe estão associados. Atualmente, a maioria das companhias aéreas adotam estratégias de manutenção preventivas para reduzir o número de falhas inesperadas. Contudo, algumas destas estratégias apresentam alguns defeitos devido à possibilidade de realizar manutenção em excesso e à incapacidade de prever falhas. O projeto *Real-time Condition-based Maintenance for Adaptive Aircraft Maintenance Planning* (ReMAP) tem como objetivo usar uma estratégia preditiva, tal como a *Condition-Based Maintenance* (CBM), para lidar com estes problemas. A ideia é monitorizar a condição de componentes com dados recolhidos de sensores para calcular valores de *Remaining Useful Life* (RUL), que podem ser aplicados posteriormente no processo de decisão de manutenção.

Este trabalho está inserido no projeto ReMAP, tendo como objetivo o uso de *Reinforcement Learning* (RL) para otimizar o agendamento de manutenção para aeronaves. Na primeira etapa, são propostas duas formulações para otimizar o agendamento de *checks* para um horizonte temporal especifico usando o algoritmo *Deep Q-Learning* (QL). Na segunda etapa, o algoritmo *Asynchronous Advantage Actor-Critic* (A3C) é usado para otimizar uma solução de empacotamento de tarefas, considerando alguns fatores importantes na manutenção de aeronaves, tais como recursos humanos e manutenção inesperada. Além disso, um algoritmo de agendamento preditivo de tarefas é proposto, usando informação de prognóstico para adaptar o plano de manutenção existente.

Os algoritmos propostos para ambas as etapas são testados e validados com dados reais de manutenção de uma frota de 51 aeronaves. A qualidade do plano de manutenção obtido é avaliado de acordo com vários *Key Performance Indicators* (KPI). Os resultados são muito positivos e promissores, indicando o potencial de RL para resolver este tipo de problemas.

Palavras-Chave

Manutenção de Aeronaves, Inteligência Artificial, Manutenção Preditiva, Agendamento de Manutenção, Aprendizagem por Reforço

Contents

Acronyms

A3C Asynchronous Advantage Actor-Critic. v, 3, 14, 15, 20, 33, 34, 37–40, 44, 50, 57

ACARE Advisory Council for Aeronautics Research in Europe. 22

AS Aircraft Scheduler. 23, 30, 42, 51, 57

CBM Condition-Based Maintenance. v, 2, 3, 7, 15, 17, 18, 22, 33

CS Conflict Solver. 23, 25, 26, 29, 30, 51, 57

DMC Direct Maintenance Cost. 2

DP Dynamic Programming. xv, 53–55, 57

DY calendar days. 6, 24, 34, 67, 73

EASA European Aviation Safety Agency. 5

ESD Earliest Scheduled Day. 28

FAA Federal Aviation Administration. 6

FC Flight Cycles. xiii, 6, 24, 34, 41, 67–69, 73

FH Flight Hours. xiii, 6, 24, 34, 41–43, 49, 54, 55, 67–69, 73

IFHM Integrated Fleet Health Management. 7

KPI Key Performance Indicator. 41, 42, 50, 53

LLF Longest Length First. 28

LTSM Long Short-Term Memory. 20

MDP Markov Decision Process. 8, 9

MDST Maintenance Decision Support Tool. 3, 7, 15

ML Machine Learning. 3, 7, 8, 15, 22, 24

MPD Maintenance Planning Document. 5, 6

MRB Maintenance Review Board. 5

MRBR Maintenance Review Board Report. 5

MRO Maintenance, Repair and Overhaul. 2

NAA National Aviation Authority. 5

NN Neural Network. 13, 20, 24, 31, 41, 44

PPO Proximal Policy Optimization. 20

QL Q-Learning. v, 3, 12, 13, 15, 24, 25, 31, 41, 44

ReMAP Real-time Condition-based Maintenance for Adaptive Aircraft Maintenance Planning. v, 1–3, 5, 7, 15

RL Reinforcement Learning. v, 3, 5, 8–10, 12, 14, 15, 17, 19–22, 24–31, 36–42, 53–55, 57, 58

RNN Recurrent Neural Network. 17

RUL Remaining Useful Life. v, 2, 3, 7, 15, 17, 18, 22, 33, 38–40, 44, 48, 50, 51, 54, 56, 57

SLF Shortest Length First. 28

TD Temporal Difference. 14

TRPO Trust Region Policy Optimization. 20

TSD Tardiest Scheduled Day. 28

WP Work Package. 3

List of Figures

List of Tables

This page is intentionally left blank.

List of Algorithms

This page is intentionally left blank.

Nomenclature

s, s'	states
a	action
r	reward
$\mathbb{A}$	set of all possible actions
$\mathbb{S}$	set of all non-terminal states
t	discrete time step
S_t	state at t
A_t	action at t
R_t	reward at t
$r(s, a)$	reward of taking action a in state s
Gt	return (cumulative discounted reward) following t
π	policy
π_*	optimal policy
$\pi(s)$	action taken in s under policy π
$v_\pi(s)$	value of state s under policy π
$v_*(s)$	value of state s under the optimal policy
$q_\pi(s, a)$	value of taking action a in state s under policy π
$q_*(s, a)$	value of taking action a in state s under the optimal policy π
$Q(S_t, A_t)$	Q-value of taking action A in state S at t
γ	discount factor of future rewards
ϵ	exploration factor

This page is intentionally left blank.

Chapter 1

Introduction

This document presents a report of the work developed in the Master's Dissertation in Informatics Engineering presented in the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

The study is done in the context of the Real-time Condition-based Maintenance for Adaptive Aircraft Maintenance Planning (ReMAP) project (H2020 ReMAP) and follows the work done previously by Pedro Ferreira in his Master's Dissertation (Ferreira, 2019).

This chapter presents the context of the research in Section 1.1, the motivation in Section 1.2, and the global goals of the thesis along with an overview of the developed work in Section 1.3. Finally, the general structure of the document is explained in Section 1.4.

1.1 Context

According to the European Committee for Standardization, maintenance can be defined as the "combination of all technical, administrative and managerial actions during the life cycle of an item intended to retain it in, or restore it to a state in which it can perform the required function" (European Committee for Standardization, 2017). Over the last decades, as a result of the technological advancements, the equipment and systems in the various industry areas have become much more complex, which massively increases the number of possible failures. This increase in complexity has direct implications in maintenance, forcing the industry to adopt new strategies to achieve goals such as higher reliability, availability, and safety while attempting to reduce the cost. In the specific domain of aviation, where a failure in a critical system may have catastrophic effects, maintenance is of the utmost importance and is especially aimed at maximizing reliability.

There are three essential strategies of maintenance: corrective maintenance (also called reactive maintenance), preventive maintenance, and predictive maintenance. The first one is only performed when a failure occurs, thus it may be viewed as repair work. This strategy leads to high maintenance costs and huge risks, mainly when applied in critical systems. In preventive maintenance, the equipment or systems are maintained in predefined intervals of time. These intervals are, in most cases, defined by the manufacturers, the maintenance staff, or by legal factors and are usually based on the number of working hours or times of the equipment or system (Li et al., 2016b). This strategy aims at reducing the number of unexpected failures caused by deterioration and other factors during the life cycle of an equipment or system, thus reducing the overall cost of maintenance. Finally, predictive maintenance uses a combination of historical data regarding previous failures and real-time condition monitoring (Condition-Based Maintenance (CBM)), to predict when a failure is likely to occur and perform maintenance at an optimal time to prevent it.

Comparing with the other two strategies, predictive maintenance has some advantages, increasing reliability and availability, reducing costs and time spent with maintenance and extending the life of the equipment and systems (Wang et al., 2015). It also faces some challenges, such as the capability to deal with big industrial data and the level of accuracy associated with the equipment or system Remaining Useful Life (RUL) prediction, which can lead to unnecessary maintenance or unexpected failures if not accurate (Li et al., 2016b). Despite these challenges, with an industry continually evolving and the growing need for maintenance optimization, it is expected that predictive maintenance will become the go-to maintenance strategy for many companies and areas in the near future.

1.2 Motivation

One of the most critical aspects for any company is reliability. One day or, in some cases, even one hour of downtime can be financially catastrophic financially and damage the image of the company. An important factor in increasing reliability is a good product design, however, as products age, they will always deteriorate in real environments, causing failures or decreased efficiency. The solution is to perform maintenance and keep them at a good operational level (Jardine et al., 2006).

Maintenance costs represent around 10-20% of the total operating costs of an airline (Periyarselvam et al., 2013). Therefore it is imperative to improve the current existing procedures and look towards maintenance optimization. An aircraft maintenance cost analysis made by the Maintenance Cost Task Force (MCTF) from the International Air Transport Association (IATA) reinforces the financial impact of maintenance in the aviation industry. Globally, in 2016, airlines spent 67.6 billion dollars on Maintenance, Repair and Overhaul (MRO), while in 2017 the value increased to 76 billion dollars (IATA's Maintenance Cost Task Force, 2016, 2017). This cost is expected to continue to grow over the years as the systems and components tend to become more complex.

According to Luxhøj et al. (1997), 39% of the maintenance performed in the industry corresponds to unplanned activities. In the aviation domain, the number is typically 50-60% (Samaranayake, 2006). These numbers reveal a big problem for airlines since the cost associated with unplanned maintenance due to failures is much higher.

Direct Maintenance Cost (DMC) is one of the most significant contributors to maintenance costs, involving the labor and material directly used in maintenance. A few factors that influence DMC are the fleet size, the frequency of maintenance intervals, and the aircraft age and utilization (Periyarselvam et al., 2013). Therefore it is essential to design a maintenance plan considering these variables. In the specific case of maintenance intervals, if these are defined by the equipment manufacturers, then there is a good chance they are not optimal, due to the lack of knowledge and experience from those manufacturers in dealing with failures in a real environment.

Ideally, the maintenance plan should be designed with these factors in mind, aiming at performing maintenance at optimal times (when the equipment is reaching the end of its life cycle). The goal of ReMAP is precisely to generate maintenance plans close to this ideal scenario by replacing fixed-interval inspections with condition-based ones. The results are an estimated benefit of 700 million euros per year for the airlines, obtained mainly through maintenance cost reduction and increased aircraft availability.

1.3 Goals and Approaches

This internship is embedded in the Work Package 6 (WP6) of the ReMAP (H2020 ReMAP) project, entitled Maintenance Decision Support Tool (MDST), more specifically, in the third task of WP6, which aims at developing efficient Machine Learning (ML) models (with special focus on Reinforcement Learning) to deal with different equipment conditions, and apply predictive maintenance based on them.

Therefore, the main goal of this work is to develop a learning model, using Deep Reinforcement Learning (RL), that generates and optimizes maintenance plans for aircraft fleets for a specified time horizon. Another goal is to include information about health prognostics, such as RUL estimations, to cover different future health condition scenarios, allowing us to develop predictive maintenance plans.

The work done in this thesis is divided in two stages. In the first stage, an approach to schedule aircraft to maintenance slots is developed. Then, in the second stage, a task packaging approach is developed to assign a set of maintenance tasks of each aircraft to each of their scheduled slots, thus, creating a maintenance plan. This solution takes into account relevant maintenance factors, such as the available manpower and unscheduled maintenance ratios. Additionally, it is developed a predictive approach that uses RUL estimations for aircraft components to produce a new maintenance plan by rescheduling maintenance tasks to optimal times.

The baseline for the approaches developed in each stage consists on RL algorithms, namely, the Deep Q-Learning (QL) and Asynchronous Advantage Actor-Critic (A3C) algorithms. The models are applied to a dataset with real maintenance data from a fleet of 51 aircraft of different types (A319, A320, A321). These approaches will be discussed with detail later in this report, along with the obtained results and validation methods.

1.4 Document Structure

This document is structured in eight chapters.

Chapter 2 presents relevant background on related concepts and areas, which are important to understand the context of this work.

Chapter 3 contains the study and research of the state of the art with respect to the topics relevant in this work, such as CBM, maintenance scheduling, and RL.

Chapter 4 details the approaches proposed in the first stage of this work and the Deep QL algorithm used for the optimization.

In **Chapter 5**, the algorithms developed in the second stage are explained, including a task packaging solution, a predictive task scheduler, and the A3C algorithm used for optimization.

Chapter 6 presents the experiments and the results obtained for both stages of the work.

Chapter 7 describes the validation methods used for all the proposed approaches.

Finally, **Chapter 8** presents some conclusions from the developed work and provides an overview of future steps.

Chapter 2

Background

This chapter presents some important topics and gives an overview of some relevant areas in the context of this thesis.

We start by presenting some current aircraft maintenance practices and a few problem areas and challenges that airlines have to face every day in Section 2.1. Then, in Section 2.2, we give an overview of the Real-time Condition-based Maintenance for Adaptive Aircraft Maintenance Planning (ReMAP) project, presenting its goals, structure, parties involved, and the estimated benefits. Some key concepts of Reinforcement Learning (RL) are presented in Section 2.3, with a special focus on the strategies and algorithms used in this thesis. This chapter ends in Section 2.4 with a summary of the discussed topics.

2.1 Aircraft maintenance current practices

Aircraft maintenance is a complex process heavily dependent on the fleet's operating plan, due to the need of knowing in advance the ground times for each aircraft, in which maintenance can be performed. Therefore, four steps can be considered in the maintenance planning for an airline: (i) flight scheduling, (ii) fleet assignment, (iii) crew scheduling, and (iv) aircraft maintenance routing (Liang and Chaovalitwongse, 2009). In the flight scheduling phase, a flight schedule is built, typically some months in advance. The fleet assignment aims to optimize the distributions of aircraft to different flights, based on forecasts of passenger demand. Crew scheduling attempts to optimize the crew distribution over all flights. Finally, the maintenance routing consists of assigning each aircraft to a sequence of flights while ensuring that each one of them has multiple opportunities to perform maintenance during the time horizon of the plan (Liang and Chaovalitwongse, 2009).

The maintenance program of an aircraft has to go through several steps before being approved, as a consequence of all the standards and regulations in aviation. Initially, when a new aircraft is introduced, the manufacturer must specify the initial minimum maintenance requirements and include them in the Maintenance Review Board Report (MRBR), which is then presented to a Maintenance Review Board (MRB). The MRB is composed of members from the airline that is purchasing the equipment, the manufacturers, and the National Aviation Authority (NAA) of the respective country, and the goal is to complement the initial maintenance requirements (Ackert, 2010). Finally, the manufacturer gathers this information and develops the Maintenance Planning Document (MPD), which must be submitted to a certified international authority for approval. European Aviation Safety

Agency (EASA), in Europe, and Federal Aviation Administration (FAA), in the United States, are the top two international authorities and are responsible for the certifications and regulations in civil aviation.

The MPD contains the final maintenance requirements for the aircraft, which means that all maintenance tasks and respective intervals are described in this document. All these tasks are allocated in maintenance checks of various types and intervals. The intervals in which the checks occur are measured either in calendar days (DY), in Flight Hours (FH), or in Flight Cycles (FC). A FC represents a complete take-off and landing sequence. The number of different checks and their intervals depend on the airline and on the type of aircraft. However, in most cases, four checks are defined (Van den Bergh et al., 2013):

- **A-check:** is performed biweekly and lasts approximately 10 hours. This check is usually performed overnight.
- **B-check:** is performed every 300 to 600 FH and lasts between 10 to 24 hours.
- **C-check:** occurs once every year and lasts one to two weeks.
- **D-check:** is a structural check and is performed every 4 or 6 years, lasting for 3 weeks to 2 months.

In some airlines, B-checks do not exist. Instead, they are replaced by A-checks that assume their interval and duration. Furthermore, there is also a **transit check**, which is performed every time the airplane stops in between flights. This check involves visual inspection to verify if the Minimum Equipment List (MEL) is in good condition. The MEL corresponds to the mandatory list of equipment that allows a plane to fly (Van den Bergh et al., 2013).

C and D checks are considered heavy maintenance and are performed in a hangar or an overhaul facility. A and B checks are also usually done in a hangar or at the gates, while the transit check is part of what is generally called line maintenance. This type of maintenance occurs on the aircraft operating environment and is usually done in between flights, lasting 1 to 2 hours. The main goal is to perform minor maintenance tasks that got postponed, visual inspections, minor fixes and repairs, or tasks that have lower intervals than the hangar checks and, thus, have to be performed outside of them (for example, lubrication tasks).

When creating a maintenance plan, several decisions have to be made to try and optimize it. For instance, in the peak seasons, airlines try to avoid significant maintenance checks to allow most of the fleet to be ready to fly, and, in most airlines, the heavy maintenance work is interrupted on weekends and public holidays. Additionally, if an A or B check is planned to occur close to a heavy check (C or D), then the lighter check is canceled and merged to the heavier check. This decreases the ground time of the aircraft, consequently increasing availability and revenue (Cook and Tanner, 2008).

The personnel involved in aircraft maintenance needs to have a valid license and certificate. Usually one person can only be certified to work with a limited number of aircraft types, and can only perform a limited amount of maintenance tasks depending on his technical skills (Van den Bergh et al., 2013). Each maintenance task possesses a task card, containing general information such as the task interval, the aircraft type, the required technical skills, the materials needed, etc. The task card also includes information on the procedures involved in the execution of the task, which must be strictly followed by the workers.

There are some known problem areas regarding current maintenance practices, and airlines face new challenges every day. Two examples are related to inventory management (Asif,

2013) and hangar limitations, which result in a limited amount of maintenance checks per day, due to the available physical space and human resources (Van den Bergh et al., 2013). Another problem for the airline occurs when important and experienced personnel leave the company. The knowledge and experience from maintenance engineers and technicians are valuable to a company because they can easily identify problems just by sense, based on past experiences. Their replacement is not always easy (Li et al., 2016a).

However, the main challenge in aircraft maintenance is related to the associated uncertainty. Flight arrival delays, maintenance check duration, manpower, material availability, and unexpected failures are some examples of variables, with a certain uncertainty level, that are essential for maintenance planning.

2.2 ReMAP Project

The H2020 ReMAP (`https://h2020-remap.eu/`) project is a European project, which started in June of 2018 and will last four years. The goal is to create a solution for aircraft maintenance, the Integrated Fleet Health Management (IFHM) system, that aims at replacing fixed-interval inspections with condition-based interventions (H2020 ReMAP).

ReMAP seeks to be one of the first major contributors to start shifting from preventing maintenance to Condition-Based Maintenance (CBM), using health prognostics and diagnostics to calculate the Remaining Useful Life (RUL) of aircraft components, which will be ultimately used to develop maintenance plans. Another big focus of ReMAP is to address the regulatory barriers with the legislation in aviation maintenance, and perform a risk assessment of the proposed IFHM approach, to try moving towards the certification of CBM as a maintenance standard.

The process of ReMAP can be summarized in the following steps (H2020 ReMAP):

1. Leverage the existing aircraft sensors and develop new ones to monitor the system and structure conditions.
2. Gather the data from the sensors and perform preprocessing techniques on the data.
3. Use Machine Learning (ML) to translate the data into prognostics and diagnostics of the systems and structures.
4. Develop a Maintenance Decision Support Tool (MDST) that combines prognostics and diagnostics information with other maintenance required items, such as flight plans and resource availability, to develop an optimized maintenance plan.

The project has 13 partners involved, from 7 European countries, and it will be tested in more than 12 systems from two aircraft fleets. Those systems include Cabin Air Conditioning and Temperature Control System, Air Cycle Machine, Variable Frequency Starter Generator, and Wheels & Brakes.

ReMAP has several Work Packages (WP) with different roles in the project. This thesis is embedded in Work Package 6 (WP6), which aims at developing the MDST. The motivation of this WP is to, for the first time, include RUL estimations in a maintenance management solution.

The benefits of the proposed approach are estimated to be more than 700 million € per year. These results are due to a direct decrease in maintenance costs and unscheduled maintenance and increased aircraft availability (H2020 ReMAP).

2.3 Reinforcement Learning

RL is an area of ML that aims at making a series of subsequent decisions to reach a particular goal. An agent acts on a certain environment and receives a reward (positive or negative) depending on the action taken. The motivation is that the agent will continuously learn from his decisions and, over time, take actions that maximize the reward and achieve the specified goal. Besides the agent and the environment, there are other important elements related to RL (Sutton and Barto, 2018):

- **State:** a state is a representation of the environment. For example, if an agent is learning to play chess, the state could be an 8×8 matrix with information about the pieces.
- **Reward:** is a number that is sent to the agent each time it acts. This number allows the agent to identify good and bad actions, and the key goal of the agent is to maximize the total amount of reward received over time.
- **Value:** while the reward only identifies good immediate actions, the value allows the agent to identify good actions in the long run, because it takes into account the next states and their possible rewards. A state can yield a low immediate reward but have a high value if it is followed by states which give a higher reward (Sutton and Barto, 2018).
- **Policy:** the policy represents the agent's behavior in the environment. In other words, the policy is a function that takes a state as input and outputs either an action or a probability distribution of all actions in that state.
- **Model:** the model represents the inner working of the environment, allowing for example to make predictions or inferences about the next states and rewards. Since the model may not be available in some scenarios, it is possible to ignore it and rely on learning by trial-and-error. This strategy is called model-free RL.

In RL, the agent learns from his own decisions, which means that in order to achieve a big reward and, at the same time, avoid converging to a local minimum, he must perform the largest number of decisions possible. This is known as the **exploration-exploitation trade-off**, where the exploration corresponds to the agent choosing a random action, and the exploitation corresponds to the agent selecting the best action in the current state, based on his experience (Sutton and Barto, 2018). A widely used strategy to ensure a proper exploration of the search space is the ε-greedy. The variable ε represents the probability of choosing a random action and is usually initialized to 1 with a decay rate over time. This ensures high exploration at the beginning, and exploitation rising over time (Arulkumaran et al., 2017).

2.3.1 Markov Decision Process

Formally, RL can be described as a Markov Decision Process (MDP) (Arulkumaran et al., 2017). An MDP can be defined as a tuple (S, A, P, R, γ), where S represents the set of states, A is the action set, P is the state transition probability matrix, R is the reward function, and γ is the discount factor, representing the value of future rewards in comparison to immediate rewards. At each time step $t = 0, 1, 2, ...$ the RL agent interacts with the environment, choosing an action to take in the current state, S_t. After taking the selected

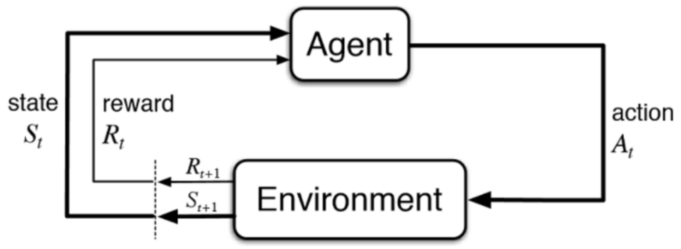


Figure 2.1: Interaction between the agent and the environment (from *Reinforcement Learning: An Introduction*, Sutton and Barto (2018))

action, the result comes one step later in the form of a numerical reward, R_{t+1}, and the agent moves to a new state, S_{t+1}. This interaction is represented in Figure 2.1.

As mentioned previously, the RL agent tries to maximize the cumulative reward over time. Thus, it is essential to have a balance between immediate and future rewards. Sometimes it may be best to choose the worst action in the present to achieve greater future rewards. There is also the concept of *discounting* that is used to give less relevance to rewards that occur far into the future. The goal of RL is to maximize the expected *return*, which represents the sum of discounted rewards over time, and can be computed with the following equation (Sutton and Barto, 2018):

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{2.1}$$

where $\gamma \in [0, 1]$ is the *discount factor*. If γ is equal to 0, only the immediate reward is considered. If γ is equal to 1, it means that all future rewards have the same importance when choosing the action to take in a given state.

When formulating an RL problem as an MDP, we have some flexibility to choose the basic elements, such as the state representation, the reward function, and the action set. It ultimately comes down to our interpretation of the problem, however, it is important to define a good reward function to ensure that the agent achieves the intended goals. Sutton and Barto (2018) give an interesting example of using RL in chess. While the state, represented by a matrix with the position of all pieces, and action set, represented by all available moves, would be straightforward to define, the reward function is more complicated. The natural idea is to give a positive reward for each piece captured, although, as the authors point out, this may lead to problems, since the agent might capture a big amount of pieces and achieve a good reward but still lose the game. A better solution is probably to give a reward only at the end of the game, which will be positive if the agent wins or negative otherwise.

A central concept in RL is the Markov property stating that: "the future is independent of the past given the present." In other words, the current state contains all the relevant information from the past, meaning that the transition from state S_t to state S_{t+1} is completely independent of past states. Therefore, the state transition probabilities can be defined with:

$$p(s', r|s, a) = Pr\{R_{t+1} = r,\, S_{t+1} = s' \,|\, S_t,\, A_t\}, \quad \forall\, s',\, s \in \mathbb{S},\, r \in \mathbb{R},\, a \in \mathbb{A} \tag{2.2}$$

With equation 2.2 it is possible to compute the expected reward for any state-action pair

(Sutton and Barto, 2018):

$$r(s, a) = \mathbb{E}[R_{t+1} \mid S_t = s,\ A_t = a] = \sum_{r \in \mathbb{R}} r \sum_{s' \in \mathbb{S}} p(s', r|s, a) \tag{2.3}$$

This information is essential to advance and to introduce the concept of **value** and **value functions**, which have a significant influence over the learning of the agent, and consequently, in achieving its goals.

2.3.2 Value Functions

In RL, the value functions are defined alongside a policy. As mentioned before, a policy π defines the behavior of the agent. In other words, a policy "is a mapping from each state, $s \in \mathbb{S}$, and action, $a \in \mathbb{A}$, to the probability $\pi(a|s)$ of taking action a when in state s" (Sutton and Barto, 2018). There are two types of value functions: **state-value function** and **action-value function**.

The state-value function defines "how good" a state really is. Formally, it is the *expected return* of starting in state s and following a policy π. This function is denoted v_π and can be defined by:

$$v_\pi(s) = \mathbb{E}_\pi\left[G_t \mid S_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right], \tag{2.4}$$

where $\mathbb{E}_\pi$ represents the expected value when following a policy π, and t is a time step.

The action-value function defines "how good" it is to take an action on a specific state. Formally, it is the *expected return* of starting in state s, taking action a and following a policy π. This function is denoted by q_π and can be defined by:

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right], \tag{2.5}$$

Both value functions 2.4 and 2.5 use the *expected return* to calculate the value, which considers the immediate reward and the discounted future rewards of taking an action in the current state. Thus, the *Bellman* equation assumes a key role in computing the value of a state, since it uses the value of the next states. This equation is defined by:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a)\left[r + \gamma v_\pi(s')\right], \tag{2.6}$$

where $p(s', r|s, a)$ is the probability of s' being the next state, and r being the reward if action a is taken in state s. This function specifies that information about the value is transferred back to a state from its successors, allowing to update the state and action values, and consequently lead to better decision making by the agent.

2.3.3 Optimal Policy

A policy is considered to be optimal if the expected return is greater or equal to the expected return of all other policies. If a policy π is optimal, then there is also an optimal state-value function and an optimal action-value function associated with that policy, called the *Bellman optimality equations* (Sutton and Barto, 2018).

Bellman's optimal state-value function, denoted v_*, is defined by:

$$v_*(s) = \max_{a \in \mathbb{A}} \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_*(s')\right] \tag{2.7}$$

Bellman's optimal action-value function, denoted q_* is defined by:

$$q_*(s, a) = \sum_{s',r} p(s', r|s, a) \left[r + \gamma \max_{a' \in \mathbb{A}} q_*(s', a')\right] \tag{2.8}$$

The Bellman optimality equations indicate that the value of a state, when following an optimal policy, is equal to the expected return for the best action to take in that state (Sutton and Barto, 2018).

Finally, we need to know the strategy to use the value function in the search for an optimal policy. There are two algorithms to achieve this goal: **policy iteration** and **value iteration**.

Policy iteration aims at continuously improving the policy until it converges to an optimal one. For this purpose two additional concepts are used: policy evaluation and policy improvement. The first one is used to evaluate a policy, and it is an iterative process of computing successive new state-value approximations using the Bellman equation 2.6 as an update rule:

$$v_{k+1}(s) = \sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_k(s')\right], \tag{2.9}$$

for all $s \in \mathbb{S}$, where k is the iteration step. This process usually starts with the state-values initialized to zero, and visits all the states in multiple iterations, updating them (using 2.9) until convergence. The policy improvement phase takes as input the state-values from the evaluation phase and finds a new policy by acting greedily, that is, choosing the best action to take in every state:

$$\pi'(s) = \arg\max_{a \in \mathbb{A}} \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s')\right], \tag{2.10}$$

for all $s \in \mathbb{S}$, where $\arg\max_a$ is the action, a, that maximizes the expression that follows it. Policy iteration works by using a sequence of policy evaluation and policy improvement multiple times until convergence. At that time the optimal policy is found.

Value iteration combines policy evaluation and policy improvement into a single operation. The goal is to overcome a drawback of policy iteration, related with the fact that the optimal policy is only obtained with exact convergence. Since each iteration requires updating the entire state space, it may take a long time to reach that convergence point. Value iteration is based on the following equation:

$$v_{k+1}(s) = \max_{a \in \mathbb{A}} \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_k(s')\right], \tag{2.11}$$

for all $s \in \mathbb{S}$. The iterative application of this equation through all states results in the optimal value function, v_*. The optimal policy can be easily obtained with v_*, by choosing to take the best action in each state.

2.3.4 Q-Learning

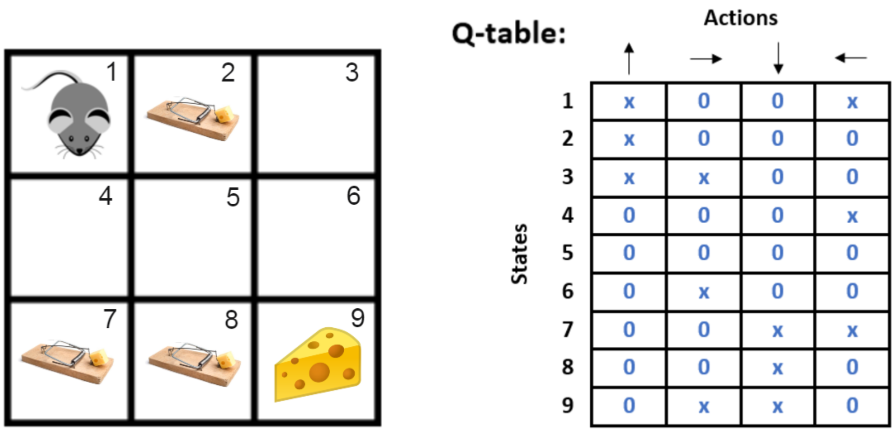
Q-Learning (QL) is a model-free RL algorithm with the goal of finding an optimal policy. This algorithm is called off-policy because it uses two different ones: a target policy, used to calculate the value functions and achieve optimality, and a behavior policy, used to control the agent behavior (Sutton and Barto, 2018). The target policy is an absolute greedy policy, and the behavior policy is usually a ε-greedy policy. QL uses a Q-table, which is, in most cases, defined by a matrix with the shape [*states*, *actions*]. Each number in the Q-table represents the value of taking an action, a, in a state, s, and it is named as the Q-value, Q(s,a). The RL agent, at each time step, observes the current state and chooses to perform the action with higher Q-value in that state. After acting, the agent receives a reward, that will be used to update the Q-table using the following equation, called the Q-function:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_{a \in A_{t+1}} Q(S_{t+1}, a) - Q(S_t, A_t) \right], \tag{2.12}$$

where $Q(S_t, A_t)$ is the Q-value of the state-action pair corresponding to state S and action A, α is a learning rate, R_{t+1} is the immediate reward received, γ is the discount factor, and $\max_{a \in A_{t+1}} Q(S_{t+1}, a)$ represents the estimated value of the next state, S_{t+a}, which corresponds to the value of taking the best action in that state.

In QL and RL in general, the path taken by the agent from the initial state to a final state is called an episode. The training of the agent is done in the course of multiple episodes.

To better understand how this algorithm works, let us look at a simple example. The agent is in an environment represented by a 3×3 grid, which indicates the existence of 9 possible states, and the goal is to reach the cheese while trying to avoid the traps. The reward function consists of a penalty of -1 each time the agent walks into a trap, and a positive reward of +1 if it manages to reach the cheese. There are four possible actions: go up, down, right, or left. Initially, we create the Q-table and initialize the Q-values to 0. In this case, let's consider moves that would get the agent out of the grid as impossible moves. Figure 2.2 shows the initial representation, along with the initial Q-table:



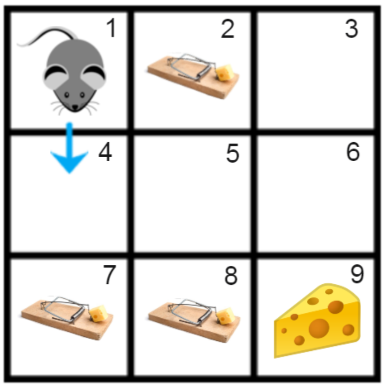
States	↑	→	↓	←
1	x	0	0	x
2	x	0	0	0
3	x	x	0	0
4	0	0	0	x
5	0	0	0	0
6	0	x	0	0
7	0	0	x	x
8	0	0	x	0
9	0	x	x	0

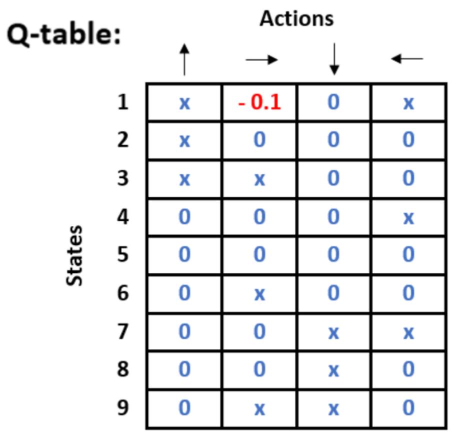
Figure 2.2: Environment representation and Q-table at time step 0.

Since all Q-values of the available actions on the state 1 (initial state) are equal to 0, the agent chooses randomly the action to take. Assuming it takes the action "go right", he will go to state 2. At this point, a reward of -1 is received, and the Q-table is updated by using equation 2.14. With the learning rate, $\alpha = 0.1$, and the discount factor, $\gamma = 0.95$:

$$Q(S = 1, A = \text{“}right''') = 0 + 0.1\,[-1 + 0.95 * 0 - 0] = -0.1 \tag{2.13}$$

After updating the Q-table, the next time the agent visits state 1, either in the same episode or in a new one, it will always choose the action “down” because it is the action with higher Q-value. This scenario is illustrated in Figure 2.3:





Q-table:

Actions

States	↑	→	↓	←
1	x	- 0.1	0	x
2	x	0	0	0
3	x	x	0	0
4	0	0	0	x
5	0	0	0	0
6	0	x	0	0
7	0	0	x	x
8	0	0	x	0
9	0	x	x	0

Figure 2.3: Environment representation and Q-table at the second visit of state 1.

This trivial example shows the learning process of the agent and the potential of this algorithm in achieving an optimal policy after multiple iterations. However, it is important to note that always taking the action with the biggest Q-value can result in achieving a local minimum, thus it is essential to introduce some exploration into the problem and ensure that all actions are tested a sufficient number of times (Glorennec, 2000). Also, this algorithm is not scalable, and for more complex problems and environments, it is not feasible to have a matrix representation of the Q-values. The solution is **Deep Q-Learning**.

A large and complex state space does not limit deep Q-Learning since it uses a Neural Network (NN) to handle the Q-values. While in QL, given a state S and an action A, we obtain the corresponding Q-value from the Q-table, in Deep QL, given a state S as input to a NN, we obtain approximations for the Q-values of every action in that state.

An important aspect of Deep Q-Learning is the use of experience replay for training the agent. Instead of training the agent with the sequence of experiences as they occur during the simulation, those experiences are saved in what is usually called the replay memory. After the agent acts, a random batch of experiences is sampled from the replay memory and used for training. This method increases the training efficiency because it trains multiple times with the same experiences and helps to overcome two problems of Deep Q-learning. One of them is the fact that as time passes, the agent tends to forget about past experiences. The second problem is the high temporal correlation that exists between two consecutive experiences, due to the fact that, most of the times, the state does not change much between two consecutive time steps (de Bruin et al., 2015).

2.3.5 Actor-critic methods

Actor-critic methods have two different memory structures to represent the policy independent of the value function: the actor and the critic (Sutton and Barto, 2018). The actor determines what action is taken in a given state by following a policy, while the critic evaluates the selected action by computing the Temporal Difference (TD) error:

$$\delta_t = R_{t+1} + \gamma V_t(S_{t+1}) - V(S_t) \tag{2.14}$$

This TD error is used for the learning of both the actor and the critic as shown in Figure 2.4. If it is positive, the selection of the same action by the actor should be encouraged in the future. If it is negative, the selection of that action should be discouraged.

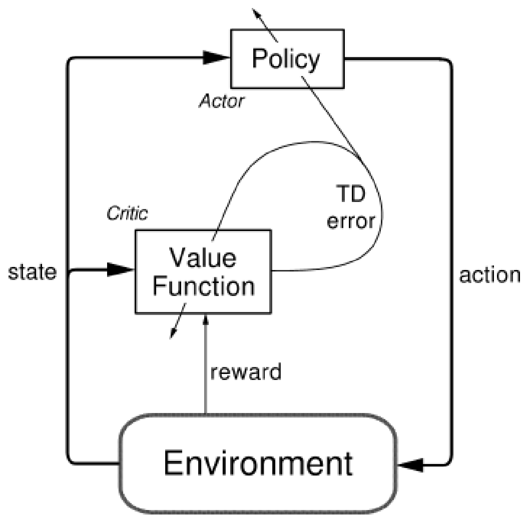


Figure 2.4: Actor-critic architecture (from Sutton and Barto (2018)).

Typically, RL algorithms use the sum of discounted rewards (G_t) to evaluate the quality of selecting an action. A different option, used in the **Advantage Actor-Critic (A2C)**, is to use the advantage function:

$$A(s,a) = Q^{\pi}(s,a) - V^{\pi}(s), \tag{2.15}$$

where $A(s,a)$ is the advantage of choosing action a in state s, $Q^{\pi}(s,a)$ is the expected reward of taking action a in state s and following policy π thereafter, and $V^{\pi}(s)$ is the expected reward of starting in state s and following policy π thereafter. By using the advantage value, the actor learns how much better it is to take a specific action compared to the others in each given state, thus, improving the learning process.

Another variant is the **Asynchronous Advantage Actor-Critic (A3C)** algorithm (Mnih et al., 2016), in which the key difference for A2C is the asynchronous part. In A3C, a set of agents interact with their own environment. The agents are trained in parallel and independently update a global network (hence "asynchronous"). After each update, they reset their parameters to those of the global network. An important benefit of this algorithm is the increased diversity in the training data, which allows for more efficient exploration and removes the need for an experience replay buffer.

2.4 Conclusion

Aircraft maintenance is a very complex challenge, mainly due to a large number of variables that must be considered, which generates a high uncertainty level. The development of a maintenance plan is also dependent on the fleet's operation plan, which consists of the flight scheduling and the routes of each aircraft for a time horizon. The combination of all these elements, along with the current preventive strategy, leads to sub-optimal decisions, making it difficult for maintenance planners to develop optimized plans, which results in substantial economic losses for the airline. The ReMAP project aims to overcome these issues by introducing CBM as a new maintenance strategy, in which the RUL of aircraft components is used in the maintenance decision process. A second important goal of ReMAP is to contribute towards the standardization of CBM in aircraft maintenance planning by addressing the existing regulatory barriers. The MDST of ReMAP aims at generating optimized maintenance plans using this new maintenance strategy by developing a learning model based on Reinforcement Learning.

RL is an area of ML that has been around for many years. An agent interacts with an environment by performing actions and receives numerical rewards based on the quality of the actions performed. The aim is to achieve a predefined goal by maximizing the sum of rewards. Instead of deciding what actions the agent must take, the idea is to allow it to discover the ones that yield greater rewards by trying them out and gather different experiences (Sutton and Barto, 2018). Two of the most well known RL algorithms were analyzed: the QL algorithm, which aims at finding an optimal policy, and the A3C algorithm, which uses an actor-critic architecture with several independent agents. Both of these algorithms are used in this work.

The concepts addressed in this chapter are essential to understand the context of this work and correspond to a baseline for the state of the art approaches, which are presented next in this thesis.

This page is intentionally left blank.

Chapter 3

State of the Art

An important step to better understand and get familiar with the methodologies and procedures that will be used is the study of the state of the art. This chapter presents the research and study of related work in multiple contexts.

Some Condition-Based Maintenance (CBM) approaches in different scenarios are presented in Section 3.1. Then, Section 3.2 presents maintenance scheduling problems applied in different areas with various optimization goals. Previous work using Reinforcement Learning (RL) is described next in Section 3.3, with a special focus on maintenance scheduling problems. Finally, Section 3.4 presents some conclusions from all the research on related work.

3.1 Condition-Based Maintenance

CBM is a maintenance strategy that aims to reduce unnecessary maintenance actions by monitoring the equipment condition using sensors of various types. With a reduction of preventive maintenance actions, the total cost associated with maintenance is also reduced. Other benefits of CBM include the increased availability, reduction of unexpected failures, and a higher security. There are three main steps involved with a CBM model: (i) data acquisition, (ii) data processing, and (iii) maintenance decision making (Jardine et al., 2006).

In an attempt to optimize the scheduling of maintenance in a fleet of twenty fighter aircraft, Li et al. (2016a) developed a maintenance model based on prognostic information, while taking into account real environment uncertainties, such as the maintenance duration and the incoming airplanes at a given time. The goal was to maximize aircraft availability and resource utilization by incorporating the remaining flying hours using probability distributions. Another study by Ahmad and Kamaruddin (2012) compares the practical challenges of implementing Time-based Maintenance (TBM) and CBM, concluding that the latter is more realistic since, in most cases, equipment show specific signs or conditions indicating that failure is going to occur.

Yam et al. (2001) developed an Intelligent Predictive Decision Support System (IPDSS) to evaluate the deterioration level of equipment and make decisions based on it. The authors take advantage of the learning ability associated with a Recurrent Neural Network (RNN) to predict the Remaining Useful Life (RUL) of equipment. The model was tested with critical equipment in a power plant, having achieved good predictive results, which allowed to perform maintenance actions at convenient times, to avoid major production losses as well as equipment failure, resulting in a maintenance cost reduction.

Koomsap et al. (2005) try to unify a machine's process control with CBM scheduling to increase production efficiency. The proposed architecture involves a CBM controller that has a lifetime estimator and a maintenance scheduler. The lifetime estimator takes the current operating parameters and the current machine conditions (based on multiple sensors) and outputs the RUL of that machine. This RUL value is then fed to the maintenance scheduler that, based on the machine conditions, will output new recommended operating parameters or recommend shutting down the machine for maintenance. This model is tested in a carbon dioxide laser system, with promising results in maintaining productivity at good levels.

Fischer et al. (2010) present a comparison between visual inspections, condition monitoring inspection, and online CBM, applied to wind turbine blades. They produce a Monte Carlo simulation to generate different scenarios over a finite time horizon. The evaluation of the different strategies is based on the total maintenance cost over the considered time period. Results indicate that both condition-based techniques are superior to visual inspections, which are done approximately once every year.

The related work presented in this section involves the application of various forms of CBM to solve different problems. In all of them, the authors have found benefits in using this maintenance strategy. Despite adding a level of uncertainty when compared to preventive strategies, due to it being condition-based, the resulting economic gains obtained with a higher maintenance efficiency, and the increased reliability obtained by predicting failures indicate the potential of CBM in dealing with maintenance scheduling problems.

3.2 Maintenance Scheduling

Feo and Bard (1989) presented a model to locate the best maintenance stations and build flight schedules that better meet maintenance demands. The authors use a two-phase heuristic to solve the model. In the first phase, a flight schedule for the entire fleet is generated over a small period of time, and the base locations are chosen in a way to minimize total cost. The idea was to choose locations with a higher number of aircraft that stayed overnight. This process is repeated N times. The second phase selects a K number of schedules from the previous phase and evaluates them at a greater time period. This approach was applied to real data from a fleet of Boeing 727, and the results proved that it was possible to substantially reduce costs by eliminating up to 5 of the 22 maintenance bases.

Nguyen and Bagajewicz (2008) developed a maintenance model to optimize preventive maintenance in a process plant, using a Monte Carlo simulation along with a Genetic Algorithm. The model takes into consideration three elements: the different types of failure for each equipment (different failures may have different cost and repair time), the equipment ranking according to the consequences of failure (used to assign maintenance priorities) and material and labor availability. The Monte Carlo simulation has an objective function corresponding to the total cost of maintenance plus economic loss (due to downtime or reduced efficiency), which is to be minimized. During the simulation, the failure times of each equipment are sampled based on their reliability (described by the mean time between failures), and the type of failure is sampled based on the probability of occurrences, which is obtained from historical data. The Monte Carlo simulation is optimized with a standard Genetic Algorithm.

With the goal of optimizing maintenance in Thailand airlines, specifically in relation to inventory management, Pleumpirom and Amornsawadwatana (2012) developed a multi-objective optimization model that aims at minimizing the lead time and the cost with material and components needed for maintenance, while maximizing its quality. The idea is that sometimes it is better to buy/loan materials or components from a supplier with a lower quality certificate to prevent an aircraft from being grounded awaiting the necessary resources to proceed with maintenance.

A study carried out by Başdere and Bilge (2014) aimed to develop a fast methodology to generate feasible maintenance routes for each aircraft in a fleet over a weekly planning horizon. The goal was to maximize the utilization of the remaining flying times of the fleet. This is equivalent to minimizing the unused flying times, which is directly related to cost. The authors formulate an Integer Linear Programming (ILP) model and use two methods to solve it: Compressed Annealing and branch-and-bound with different branching priorities to identify important variables.

Witteman (2019), in his master thesis, proposes a two-stage framework to solve the task packaging and allocation problem for an aircraft fleet. In the first stage, is determined the workforce to be allocated per day to each aircraft under maintenance. In the second stage, the task packaging and allocation problem is solved independently for each aircraft, with the tasks being allocated at the work shift level. Information about specific workforce skills and access panels are also taken into consideration. For both stages, an exact method is used and compared to an approximation algorithm based on bin packaging techniques. The methods are formulated using Mixed-Integer Linear Programming and the results indicate that the approximation algorithm runs up to 26 times faster than the exact method. The study was tested with data from 45 aircraft from a European airline, in which the maintenance tasks to be scheduled consisted of all the routine tasks required during A and C-checks.

The scheduling of aircraft maintenance is a complex problem, which involves multiple variables. As a result, most times, only a subset of these variables is considered to optimize the scheduling of maintenance. It ultimately depends on the optimization goal, which can be to increase efficiency and availability by improving resource utilization, optimize inventory management, or take into consideration other factors that optimize maintenance plans and bring benefits to the airline.

3.3 Reinforcement Learning: current approaches

Over the years, RL has been used with impressive results in the domain of the games. In Tesauro (1995) the author applies RL in the Backgammon game and was able to beat top professional players at the time. Szita and Lorincz (2006) use RL to play Tetris while Block et al. (2008) use it in Chess, both achieving good results. However, one of the most known applications is AlphaGo, a program capable of playing the board game Go. AlphaGo was able to beat various professional players using a combination of "value networks to evaluate board positions and policy networks to select moves" (Silver et al., 2016). The networks were trained with supervised learning using games from expert players and with RL. One year later, a new and superior version came out, named AlphaGo Zero (Silver et al., 2017). This time the training was based only on RL, using the moves done by AlphaGo. The results were even more impressive, with AlphaGo Zero beating the previous version by 100-0 and becoming the top player in the world.

Atari games are also heavily used to test the performance of RL agents. In this context, various RL approaches have been formulated. Mnih et al. (2015) developed a Deep Q-learning agent with experience replay, which was tested in a set of 49 games, achieving results comparable to professional game testers. Hasselt et al. (2016) formulated an approached that improved the results obtained by Mnih et al. (2015) using Double Q-learning. This approach has a new Neural Network (NN) (target-network) to estimate target values used to train the main network. The goal is to reduce the Q-value overestimation problems that are known to occur in Deep Q-learning. A second variation is proposed by Wang et al. (2016b), and it involves a Dueling Network Architecture. In this new architecture, the last fully connected layer of the network is divided into two: one representing the state value and another representing the advantage of taking each action. The idea was to learn the value function more efficiently since, at every update, all the action values are updated instead of only updating one action value.

Schaul et al. (2016) formulate a Prioritized Experience Replay, which aims at replaying the important experiences more frequently. This approach outperforms the standard experience replay Deep Q-learning in 41 of the 49 Atari games.

With a regular Deep Q-learning algorithm applied in an Atari game, the input consists of the last four game screens, which might not be enough to master games that require more distant knowledge. To tackle this problem, Hausknecht and Stone (2015) proposed Deep Recurrent Q-learning. This approach consisted of replacing the first fully connected layer of the network with a recurrent Long Short-Term Memory (LTSM).

Mnih et al. (2016) introduce the Asynchronous Advantage Actor-Critic (A3C) algorithm. Unlike traditional Deep Q-learning in which there is a single agent, in A3C, there are several agents, each one with a separate NN, and all of them contribute to a single global network. Since every agent acts on its own environment, the experiences are independent of each other, resulting in a better state-space exploration. The author tested this approach with the same set of Atari games, and the obtained results surpassed previous algorithms. Another important aspect of this method is the decrease in training time, which may be important in more complex scenarios.

Schulman et al. (2015) propose a new algorithm for policy optimization called Trust Region Policy Optimization (TRPO). The algorithm was used to learn tasks such as swimming, hopping, and walking in a robotic simulator, having outperformed prior generic algorithms, and proving to be robust in a wide variety of tasks. A policy gradient method called Proximal Policy Optimization (PPO) is presented in Schulman et al. (2017). It allows multiple epochs of stochastic gradient ascent updates per sample of data. This method keeps the reliability of the TRPO methods and, at the same time, is much more straightforward to implement.

Despite the achievements in the gaming domain over the years, RL also produced good results in various applications from different areas. For example, Bu et al. (2009) used RL in the autonomic configuration of online web systems. In the chemistry domain, Zhou et al. (2017) applied state-of-the-art RL algorithms to optimize chemical reactions. Zhang et al. (2018) produced an intelligent traffic light signal control with RL using a simulated environment and testing under various road networks and traffic flow rates. Huang et al. (2018) used Q-learning to detect anomalies in different datasets composed of labeled data.

3.3.1 Reinforcement Learning for scheduling of maintenance

Knowles et al. (2011) used Q-Learning in a maintenance scheduling problem to decide at each step if a maintenance job should be performed or not. They define some scenarios and do some simulations, which involve two elements: a plant-model, that provides information about the current state condition, and the RL module, that chooses to perform maintenance or do nothing based on the information received. The reward is directly related to the cost, which is higher in case of failure than in the case of preventive maintenance. Thus, at each decision step, the RL has to decide between performing maintenance and accepting the respective moderate cost or do nothing, risking a much bigger penalty if a failure occurs. Despite the obvious limitations in the proposed scenarios, namely the fact that the current state condition was only determined by the elapsed time since previous maintenance, not taking into account important factors such as the age and historical data, the RL agent produced good results, achieving convergence in all scenarios.

In the aviation domain, Mattila and Virtanen (2011) apply RL to optimize maintenance with fighter aircraft during conflict situations. The authors present two formulations to describe two different scenarios. The first formulation aims at maximizing aircraft availability by choosing when to start each maintenance activity. Each aircraft has a feasible maintenance window, and it must go to maintenance before the end of that window. The reward function involves a negative reward corresponding to the number of aircraft in maintenance. This formulation describes a situation where the number of aircraft available needs to be kept high for an extended period of time. The second formulation aims at keeping the fleet availability above a certain target level while maximizing the number of maintenance activities performed. In this formulation, the maintenance can be delayed to after the feasible maintenance window to preserve the level of availability. The reward function involves a positive reward each time maintenance is started and a negative reward if the availability decreases below the target level. This second formulation describes a scenario where the opponents are expected to make a move, being important to keep a high level of the fleet ready and in the best conditions possible. In both formulations the authors used RL to learn optimal policies, by using the Q-learning variants λ-SMART and SARSA with ε-greedy for exploration. The policies obtained for both formulations proved to be efficient.

Wang et al. (2016a) studied an approach to solve a maintenance problem in a flow line system composed of two series machines with a buffer in between. Since both machines deteriorate as they age, they have different condition states, which are defined by the yield level. The goal is to find a policy to perform preventive maintenance at optimal times. The authors formulate a semi-Markov decision process to model the system and use a multi-agent RL algorithm to solve it. Because the agents act independently from each other in the same environment, the local decisions taken by them need to be merged to achieve a global optimization goal, which corresponds to the cost. The proposed method was compared with other literature approaches, namely a sequential preventive maintenance algorithm and an independent RL algorithm. The obtained results for different costs and probabilities of failure proved that the proposed approach is superior to the other algorithms.

Barde et al. (2019) use on-policy first visit Monte Carlo to obtain the optimal replacement policy that minimizes the downtime of military trucks, composed of different types of components with random time-to-failure. The authors present four different strategies. The first strategy is based on corrective maintenance, meaning that each component is replaced at failure. The remaining three are based on preventive maintenance. The second strategy involves the addition of regular replacement intervals for each equipment (determined based on historical data). The third strategy adds a scheduled overhaul, where the whole system is replaced. The fourth strategy uses the concept that the time to replace components in the same physical area of the truck is lower than the time it takes to replace each one individually (due to the time wasted to open and close the panels of access to those components), thus when replacing a component, its neighbors are also replaced. The possible actions at a given state are to perform preventive maintenance or do nothing (for each truck). The reward function involves a penalty equal to the total downtime, which is bigger when replacing a failed component than replacing the same component with preventive maintenance. The obtained results show a reduction in the downtime of up to 36%.

3.4 Conclusion

Over the years, many approaches have been studied on how to improve and optimize maintenance scheduling in industry and specifically in the aviation domain, most of them with the goal of cost reduction in mind. Although, this is not as simple as it may sound due to the complexity level associated with this problem. The maintenance intervals for every task from each class of aircraft, along with material, spare parts, human resources, and the number of hangars available at any given time as well as general information about peak seasons, weekends, and public holidays are all variables that influence the maintenance plan, creating a difficult optimization problem.

CBM has been a focal point in many studies regarding maintenance optimization due to its achievements in reducing the total maintenance time and optimizing equipment usage, which leads to increased availability and reliability and, ultimately, to maintenance cost reduction. This strategy also poses some challenges, namely the need for large amounts of data, which may result in high initial costs if there is not already an infrastructure to support it, for example, with condition monitoring sensors (Ahmad and Kamaruddin, 2012). Another challenge is the uncertainty associated with the predicted RUL of the equipment, which must be properly dealt with (Li et al., 2016b). Despite these challenges, CBM approaches have already produced promising results, which is a good indicator that it can become the basis of maintenance scheduling in the future. In fact, the Advisory Council for Aeronautics Research in Europe (ACARE) estimates that by 2035 CBM will be accepted as the standard maintenance policy, and by 2050 all new aircraft and systems will be designed for CBM *(ACARE - Strategic Research and Innovation Agenda - Volume 2)*.

The study of state of the art regarding RL revealed that despite the multiple applications in the gaming department, this Machine Learning (ML) division could also be applied in other areas with good results, including in scheduling of maintenance. With RL, a three year maintenance plan can be optimized in a matter of minutes, while it can take days or even weeks for the maintenance planners and managers to develop an ideal program for the same time horizon. Also, with the basic functioning of RL we can explore a large part of the uncertainty space by encouraging the agent to do so, which allows the system to handle a substantial amount of possible future scenarios. Therefore, RL is a suitable approach to deal with the uncertainty, both related to the aircraft maintenance scheduling, and the health condition introduced by CBM.

Chapter 4

Scheduling Maintenance Checks

As mentioned previously, this work is divided in two stages. This chapter presents a detailed description of the problem and the proposed approaches for the first stage, with a special focus on the optimization goal and constraints. These constraints, along with other maintenance information for a set of aircraft, are part of the dataset used, which is explained in Section 4.1.

To solve this problem, a Deep Reinforcement Learning algorithm is used and described in Section 4.2. This algorithm serves as a baseline for the two formulations defined: a **Conflict Solver (CS)**, which is presented in Section 4.3, and an **Aircraft Scheduler (AS)**, which is presented in Section 4.4.

4.1 Problem description

The problem to solve consists of scheduling a set of maintenance checks for a time horizon of four years. This problem is composed of multiple constraints. For the A-checks:

- every A-check lasts one day;
- there is one A-check slot on Mondays and Thursdays and two slots on Tuesdays and Wednesdays;
- in public holidays, Fridays, and weekends there are zero A-check slots;
- it is possible to merge an A-check into a C-check to avoid the aircraft being grounded two times in a short period of time. This does not affect the duration of the C-check or the available A-check slots.

For the C-checks:

- the C-check duration varies between 5 and 23 days;
- there are three C-check slots every day except on weekends and public holidays, in which the C-check work is interrupted. Also, in some periods when the airline expects higher C-check requirements, there are 4 slots available;
- it is not possible to schedule C-checks during commercial peak seasons. These seasons consist of three weeks during Christmas and New Year period, two weeks in the Easter period, and the summer period between the 1st of June and the 30th of September;

- the start dates of two consecutive C-checks must be spaced three days apart, due to resource availability.

A dataset with real aircraft maintenance data is used. The dataset contains information from a fleet of 51 aircraft of different types (A319, A320, and A321). The information, for each aircraft, consists of the time elapsed since the previous A and C-check, the intervals in which the A and C-checks must occur, and the average daily utilization in terms of Flight Hours (FH) and Flight Cycles (FC). The combination of these variables allows us to compute the **due date** of each check, that corresponds to the limit date when the aircraft must be grounded for maintenance.

Every aircraft has an A-check interval of 120 calendar days (DY) (four months), 750 FH, and 750 FC. The C-check intervals are either 730 DY (two years), 7500 FH, and 5000 FC, for most aircraft, or 1096 DY (three years), 12000 FH, and 8000 FC, for a small number of aircraft. In case a maintenance check cannot be scheduled before its due date, there is a tolerance of 12 days, for A-checks, and 60 days, for C-checks, after the due date, to schedule it. Additional details regarding the dataset can be found in Appendix A.

The basic architecture of this check scheduling stage is illustrated in Figure 4.1.

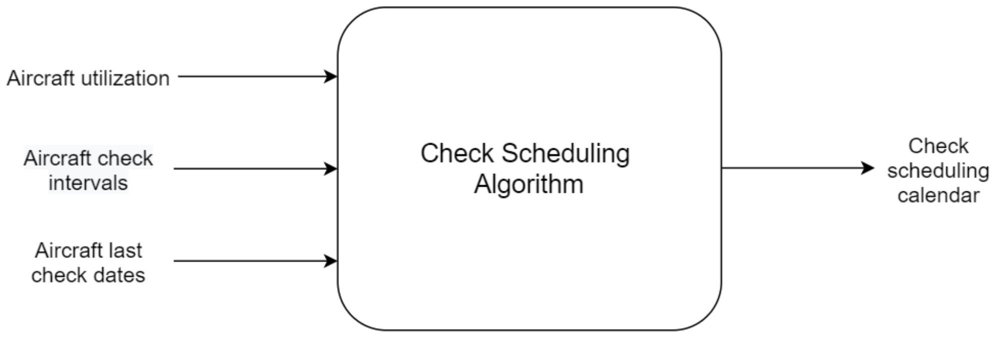

Aircraft utilization
Aircraft check intervals
Aircraft last check dates
Check Scheduling Algorithm
Check scheduling calendar

Figure 4.1: Check scheduling basic architecture.

To solve this problem, two formulations were developed, both solved with Deep Reinforcement Learning (RL), using the Deep Q-Learning (QL) algorithm. These elements are explained in detail in the following sections.

4.2 Deep Q-Learning

A Deep QL algorithm was used to solve the maintenance optimization problem described. Multiple combinations of hyper-parameters and Neural Network (NN) architectures were tested to achieve good results and will be discussed later.

Along with the base Machine Learning (ML) algorithm, some strategies have been used to increase its performance. Experience replay was used to enhance training efficiency and remove the correlation between subsequent decisions, and the ε-greedy strategy was used to ensure the proper exploration of the state-space.

In addition, the concept of Double QL is used. This method is proposed by Hasselt (2010) and aims at removing over-estimations in the action values. According to the author, these over-estimations occur because, in Deep QL, the same values are used to select an action and evaluate it. The solution decouples the selection and evaluation by using a second network, generally called the target network. The target network is used to estimate the action values while the online network is used to select the action greedily. Periodically the weights of the target network are updated by replacing them with the weights of the online network.

The pseudo-code for the Deep QL algorithm is presented in Algorithm 4.1. Initially, the replay memory and the online and target networks are initialized. At the beginning of each episode, the current state is set to the initial state. Then, for each step, an action is chosen either randomly or greedily with a certain probability. The chosen action is simulated, the reward and next state are collected and stored in the replay memory, and the current state advances to the next one. A minibatch is sampled from the replay memory and used to train the online network by computing the target value and respective loss. Finally, the target network can be updated to the online network. This update occurs periodically after a certain amount of steps.

Algorithm 4.1: Pseudo-code for the Deep Q-Learning algorithm

```
Initialize replay memory D;
Initialize the online network $Q^A$ with random weights;
Initialize the target network $Q^B$ with random weights;
for $episode = 1, M$ do
    $s_t = s_1$;
    for $t = 1, T$ do
        $a_t$ = random action with probability $\epsilon$, otherwise,
        $a_t = argmax_a Q^A(s_t, a)$;
        Simulate action $a_t$ and observe the reward $r_t$ and the next state $s_{t+1}$;
        Store the tuple $(s_t, a_t, r_t, s_{t+1})$ in D;
        $s_t = s_{t+1}$;
        Sample a minibatch of $(s_j, a_j, r_j, s_{j+1})$ from D to train $Q^A$;
        Compute the target value, $y_j = \begin{cases} r_j, \text{if the episode terminates at step j+1} \\ r_j + \lambda max_a Q^B(s_{j+1}, a), \text{otherwise} \end{cases}$;
        Calculate the loss using $(y_j - Q^A(s_j, a_j))^2$ and update the weights of $Q^A$;
        if $Update(Q^B)$ then
            Replace $Q^B$ weights with $Q^A$ weights;
        end
    end
end
```

Both formulations described in the following sections use this algorithm as a baseline. The only modifications made are related to the state representations, reward function, and the set of actions for the RL agent.

4.3 First formulation: Conflict Solver

The first formulation was adapted from Ferreira (2019), whose initial idea was to solve conflicts between maintenance checks. The intention comes from the fact that, in the real world, the manual approach to maintenance planning consists precisely of shifting checks in conflict to earlier slots (Deng et al., 2019). Our CS extends this work by introducing the constraints discussed previously, which were not being considered in Ferreira (2019).

The CS works as follows: in the beginning, all the A and C-checks for the time horizon considered are calculated by computing their due dates, and they are then scheduled, one by one, to the closest possible day to the due date. When scheduling each check, a conflict may arise. For instance, if an A-check is scheduled to a Monday and there is already one A-check occupying the slot available. In this case, the RL agent comes in and chooses one of the maintenance checks involved in the conflict, moving it back until the conflict is solved. The shifting of a maintenance check can cause yet another conflict, and if so, the same process is repeated (Figure 4.2). The algorithm ends when all maintenance checks are scheduled, and there are no conflicts in the calendar.

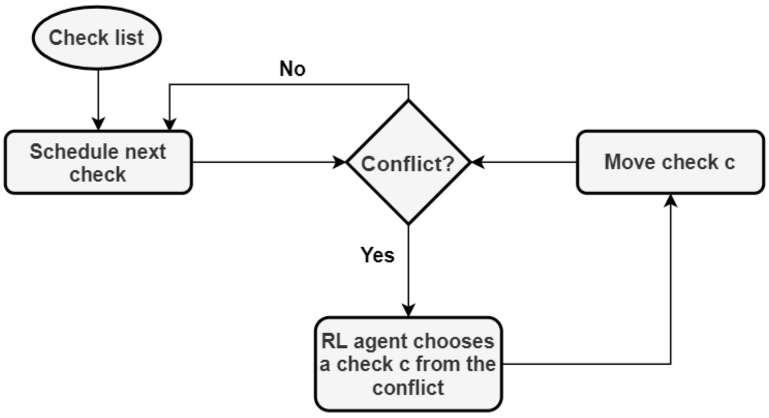


Figure 4.2: Steps of the Conflict Solver.

There is another important aspect to consider after moving a maintenance check. If there are any other checks of the same type that are already scheduled after the one being moved, those checks may also need to be moved back to ensure the maximum interval between checks is satisfied. For example, let's assume that an aircraft has two A-checks scheduled, and the interval between them is already the maximum interval allowed. Then, if the first A-check is shifted back due to a conflict with another aircraft, the second A-check also needs to be moved the same amount of days.

The pseudo-code for the CS is presented in Algorithm 4.2. Initially, the list of checks and an empty queue of conflicts are created. Then, for each check, schedule it to the closest possible day to its due date, check for new conflicts, and update the queue. While the queue is not empty, the RL agent chooses a check and moves it. The following checks for the same aircraft are also moved when the maximum interval is exceeded. Again, the queue is updated by checking for new conflicts.

4.3.1 Maintenance checks ordering

One of the most impacting factors in the outcome of the algorithm is the order in which the maintenance checks are scheduled. This order is important because it determines which checks will be in conflict first, influencing the path taken by the RL agent and the maintenance plan generated by his decisions.

To test this factor, two different methods to order the maintenance checks were applied to the CS:

- **Time-based ordering:** this method orders the checks by their due dates. In prac-

Algorithm 4.2: Pseudo-code for the Conflict Solver algorithm

```
Create the list of checks $C$;
Set $N$ = total number of checks;
Create an empty queue of conflicts $X$;
for $i = 1, N$ do
    Schedule $c_i$ as close as possible to its due date;
    Verify new conflicts and update $X$;
    while $X$ not empty do
        $c_j$ = check chosen by the RL agent;
        move $c_j$;
        move next checks for the same aircraft;
        Verify new conflicts and update $X$;
    end
end
```

tice, this means that in the exact moment a check is being scheduled, the maintenance calendar only has checks scheduled before or on its due date.

- **Aircraft-based ordering:** this method orders the checks by aircraft. In practice, this means that all checks from aircraft *j* must be scheduled before scheduling the first check of aircraft *j+1*.

When a check is being scheduled with the aircraft-based ordering method, the previously scheduled checks are well distributed on the calendar. This means that the probability of the new check generating a conflict is higher. Therefore, it is expected that, in the long term, this method will create more conflicts than time-based ordering.

4.3.2 Reinforcement Learning Environment

The simulated environment is a crucial element in an RL problem since it is responsible for establishing and evaluating the behavior of the agent. The main components to define in the RL environment are the scheduling function, the state representation, the reward function, and the action set.

Scheduling Function
As mentioned before, the scheduling function tries to schedule a maintenance check to its due date. If there is no maintenance work on that day (due to it being a public holiday, for example), then the check is scheduled before it, to the closest day that does not violate the constraint.

In a re-scheduling operation, the check is moved back to the closest day that ends the conflict. If there is no day available before the due date, then the tolerance is used. For instance, since an A-check lasts one day, when solving a conflict, the A-check only needs to be moved back one day to solve it (providing that the day before does not violate a constraint). This re-scheduling only solves the current conflict and may generate new ones in the new check date.

State Representation
In this formulation, since the RL agent selects one of the actions to move and solve a conflict, it is important that the state contains information about all the checks involved in that conflict, as well as the remaining checks that are not on the conflict but are scheduled

to the neighboring days. Therefore, the state is represented by a matrix, in which the columns represent the calendar days, the first *n* rows represent the checks in the conflict, and the row *n+1* represents the sum of the remaining neighboring checks. To decrease training time, only a partial calendar, composed of 100 days, is used to represent a state. A section of a state is presented in Figure 4.3. This section represents two A-checks scheduled to the same day and two additional A-checks outside the conflict.

$$\begin{array}{c|c|c|c|c|c|c|c}
 & d_i & d_{i+1} & d_{i+2} & d_{i+3} & d_{i+4} & d_{i+5} & \\
\hline
 & 0 & 0 & 0 & 1 & 0 & 0 & \\
\hline
\blacksquare\ \blacksquare\ \blacksquare & 0 & 0 & 0 & 1 & 0 & 0 & \blacksquare\ \blacksquare\ \blacksquare \\
\hline
 & 0 & 0 & 1 & 0 & 1 & 0 & \\
\hline
\end{array}$$

Figure 4.3: State representation of the Conflict Solver.

Reward Function
The reward function is presented in 4.1 and involves three different penalties:

$$R = \begin{cases} d - dd, & \text{if } d \leq dd \\ 10 * (dd - d), & \text{if } d > dd \\ -800, & \text{if } d = -1 \end{cases} \tag{4.1}$$

where *dd* corresponds to the check due date, and *d* corresponds to the scheduled day. The days in the calendar assume an integer sequence, with the first day corresponding to day 0. The first and second conditions punish the agent for each day that the check is scheduled before the due date and after the due date, respectively. This penalty is multiplied by 10 in the second condition to avoid the use of tolerance. Finally, the third condition inflicts a very high penalty in the scenario where a slot was not found before the due date or in the tolerance. The value of -800 ensures that the third condition always represents the highest penalty since the tolerance can be at most 60 days.

Action Set
The RL agent is responsible for selecting a check in a conflict to be re-scheduled. The action set to achieve consists of the following four actions:

- **Shortest Length First (SLF):** chooses the check with the lowest duration;
- **Longest Length First (LLF):** chooses the check with the highest duration;
- **Earliest Scheduled Day (ESD):** chooses the check with the earlier scheduled day;
- **Tardiest Scheduled Day (TSD):** chooses the check with the tardiest scheduled day;

Since in the constraints we know that an A-check only lasts for one day, this action set is only applicable to the C-checks. Thus, when the agent acts and selects an A-check, in practice, that check is chosen randomly.

4.4 Second formulation: Aircraft Scheduler

As mentioned before, the problem solved in Ferreira (2019) was defined without all the constraints. For instance, the public holidays, weekends, and peak seasons were not yet being considered. As more constraints started to be added into the problem, the CS became inefficient and started to produce worse results than those previously obtained without considering constraints. This behavior occurs because, with the introduction of days with no maintenance work, the number of conflicts rises, which results in many re-schedules and, in turn, creates even more conflicts.

Therefore, a second formulation is developed. The idea behind this formulation is to find an optimal sequence of aircraft to have its next check scheduled in the calendar. At each time step, the RL agent chooses an aircraft, and the next check for that aircraft is scheduled to the best available day, which does not create any conflict and does not violate any constraint. Since the position of a check after being scheduled on the calendar is final, the next due date for the aircraft is calculated immediately after scheduling a check, and the aircraft is ready to be selected again by the agent (Figure 4.4).

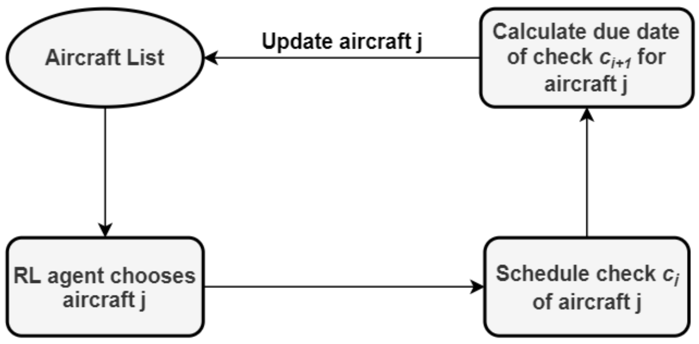

Aircraft List
Update aircraft j
Calculate due date of check c_{i+1} for aircraft j
RL agent chooses aircraft j
Schedule check c_i of aircraft j

Figure 4.4: Steps of the Aircraft Scheduler.

It is common for a lighter check to be merged to a close heavier check. This prevents the same aircraft from being grounded for maintenance twice in a short period. To accomplish this factor, the C-checks are scheduled first, which allows to verify if it is beneficial to perform a merging, based on the distance between the A-check and the closest C-check before it. When the next due date for the A-checks is beyond the time horizon considered, the aircraft becomes unavailable for the agent to choose. The algorithm ends when all aircraft are unavailable, meaning that all maintenance checks have been scheduled on the calendar.

The pseudo-code for the second formulation is presented in Algorithm 4.3. First, the aircraft list, total aircraft number, and time horizon are initialized. Also, the number of unavailable aircraft is set to 0. At each step, while the unavailable aircraft is lower than the total aircraft number, the RL agent chooses an aircraft. Then, if not all the C-checks for the chosen aircraft are scheduled, the next C-check is scheduled for the best possible day, and the following C due date is calculated. Otherwise, the next A-check is scheduled for the best possible day, and the following A due date is calculated. If the due date of the next A-check is higher than the time horizon, the aircraft is finished and the unavailable counter is increased by 1. Each step ends with the update of the aircraft in the global list.

Algorithm 4.3: Pseudo-code for the Aircraft Scheduler algorithm

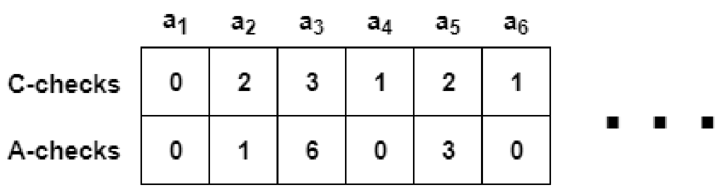
```
Initialize the aircraft list $A$;
$N$ = total number of aircraft;
$unavailable = 0$;
$H$ = time horizon;
while $unavailable < N$ do
    RL agent chooses an available aircraft $a_j$;
    if not all C-checks of $a_j$ are scheduled then
        Schedule the C-check of $a_j$ to the best possible day;
        Calculate the due date of the next C-check of $a_j$;
    else
        Schedule the A-check of $a_j$ to the best possible day;
        Calculate the due date of the next A-check of $a_j$;
    end
    if due date of the next A-check of $a_j > H$ then
        $unavailable = unavailable + 1$;
    end
    Update $a_j$ in the aircraft list;
end
```

4.4.1 Reinforcement Learning Environment

The RL simulated environment is also an important element to consider in this formulation. Similarly to the first formulation, the same 4 components are defined. The **reward function** is the same function from the CS since the agent goal is also the same. Therefore, the remaining three components will be the focus.

Scheduling Function
As mentioned before, the scheduling function in the second formulation aims to find the best slot available before the due date or, if not possible, after the due date by using the tolerance. While this behavior is similar to the scheduling function of formulation one, a key difference is that in this formulation, all constraints are taken into consideration, including the number of slots already being used. Thus, it is not possible to create any conflict when scheduling a maintenance check.

State Representation
In the AS, the goal of the agent is to choose an aircraft at each time step, which at the end of multiple episodes of training will result in one or more optimal aircraft scheduling sequences. Then, the state should contain information about the number of checks already scheduled for each aircraft, at that specific time step. To separate the A and C-checks, the state is represented with a matrix with two rows: the first one for C-checks and the second for A-checks. Each column represents a single aircraft. Thus, the total number of columns is equal to the total number of aircraft. An example of the initial portion of a state is represented in Figure 4.5.



	a1	a2	a3	a4	a5	a6
C-checks	0	2	3	1	2	1
A-checks	0	1	6	0	3	0

■ ■ ■

Figure 4.5: State representation of the Aircraft Scheduler.

Action Set
The RL agent chooses an aircraft to have its next check scheduled, at each time step. Therefore, the action set in this formulation consists of choosing each aircraft. If we have in total 51 aircraft, the agent has 51 possible actions. The action performed by the agent is chosen with an ϵ-greedy strategy, using the approximations of the Q-values for each action-state pair given by a NN, as explained in previous sections.

4.5 Conclusion

The problem that we are aiming to solve consists in the scheduling of aircraft maintenance checks for a time horizon of four years while taking into account several constraints. The goal is to optimize this process by scheduling maintenance checks as close as possible to their due dates, which increases the fleet availability and results in higher resource utilization.

We make use of a dataset containing real maintenance data, and propose two formulations to solve this problem. Both of them are based on the Deep QL algorithm. The first formulation aims at simulating the same methodology used by humans to develop maintenance plans, which consists of scheduling maintenance checks to their due date, or at least, to the closest day possible, and solving conflicts as they start to appear. The second formulation aims at finding a sequence of aircraft to be scheduled for maintenance by scheduling their next check to the best day available.

This page is intentionally left blank.

Chapter 5

Packaging Maintenance Tasks

This chapter presents a description of the work developed in the second stage of this thesis. The problem being solved is detailed in Section 5.1. In Section 5.2, the Asynchronous Advantage Actor-Critic (A3C) algorithm used for the optimization of the proposed approaches is presented. A task packaging approach is explained in Section 5.3, and a Condition-Based Maintenance (CBM) approach, which uses Remaining Useful Life (RUL) values in the maintenance decision process, is presented in Section 5.4. The chapter ends with a brief summary of the discussed topics in Section 5.5.

5.1 Problem Description

The problem to solve consists of packaging a set of maintenance tasks into predefined maintenance slots to produce a maintenance plan for an aircraft fleet. There are some constraints to consider:

- A task requires a specific skill to be performed.
- It is possible that a task can only be performed in a specific type of maintenance slot (for instance, only in a C-check).
- Each maintenance slot has a maximum number of hours allocated that depends on the number of aircraft under maintenance at that time.
- In each day there is a limited amount of manpower available for maintenance.

Another aspect to consider is the time to allocate for non-routine maintenance. Typically, when performing a maintenance task there is a certain amount of additional work that needs to be done. For instance, the detection of another malfunction or the need for more extensive testing of the equipment. Therefore, this unscheduled work also needs to be considered when packaging maintenance tasks.

The dataset used in this stage is composed of four types of data: general task information, manpower availability, aircraft utilization, and non-routine rates. The general task information contains several task attributes, such as the unique code, skill required, duration, interval, and the last date it was performed. The manpower data contains historical data of the number of maintenance technicians available on each day of 2017. The aircraft utilization data contains the average daily utilization of each aircraft in each month of the year. Finally, the non-routine rates data contains rates that need to be multiplied with the required man-hours of each task to have a more accurate representation of the actual work necessary to perform a task. Samples from the used data files can be found in Appendix A.

Similarly to the checks, the maintenance task intervals can be defined in three formats: DY, FH, and FC. This interval along with the last execution date of each task and the average utilization of the respective aircraft can be used to calculate the next due date for the task.

The aircraft fleet represented in this dataset is the same as the one from the first stage. Thus, the maintenance checks obtained with the previous check scheduling algorithm are used as an input for the task packaging algorithm.

The basic architecture of this task packaging stage is illustrated in Figure 5.1.

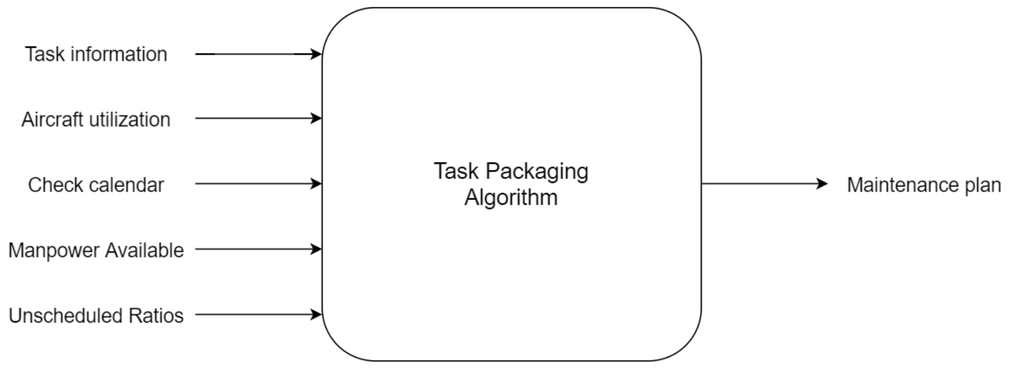


Figure 5.1: Task packaging basic architecture.

5.2 Asynchronous Advantage Actor-Critic

The A3Cs algorithm was used for the optimization of the developed task packaging approach. A single global network is used to represent the policy (critic), which must be followed by a set of actors working together in their own independent environment. Each actor has its own copy of the global network that is used to make decisions on the environment. When an actor reaches a terminal state, which corresponds to the end of an episode, it asynchronously updates the global network based on the experiences collected during the episode. Each experience contains information about the current and next state, action taken, and reward received. This information is used to compute the advantage values that are used for training. Before proceeding to the next episode, the actors also reset their copy of the network to the global one.

The pseudo-code for an actor thread of this algorithm is presented in Algorithm 5.1. First, the local network is initialized. At the beginning of each episode, the local network is synchronized with the global one, the current state is set to the initial state, and the step counter is set to 0. Then, at each step, until a terminal state is reached, an action is chosen and performed according to a certain policy, and the reward and next state are collected. After the episode ends, the respective counter is increased by 1, the sum of discounted rewards is computed and used to calculate advantage estimations. Finally, the global network is asynchronously updated with these advantages.

This algorithm is used to optimize the task packaging algorithms proposed in this chapter.

The parameters considered will be discussed with more detail later in the report.

Algorithm 5.1: Pseudo-code for an actor thread of the A3C algorithm

```
$\theta$ = global network;
$e$ = global episode count;
$M$ = max episodes;
Initialize local network $\theta'$;
while $e < M$ do
    synchronize local network: $\theta' = \theta$;
    set $s_t$ to initial state;
    set step counter $t = 0$;
    while $s_t$ non terminal do
        perform action $a_t$ according to policy $\pi(a_t|s_t;\theta')$;
        receive reward $r_t$ and state $s_{t+1}$;
        $t = t + 1$;
    end
    $M = M + 1$;
    set discounted rewards $R = 0$;
    for $i \in \{t-1, ..., 0\}$ do
        $R = r_i + \gamma R$;
    end
    calculate advantage estimations using $R$;
    perform asynchronous update of $\theta$;
end
```

5.3 Proposed approach

The developed approach for packaging maintenance tasks requires various types of input data, such as individual task data, manpower availability, aircraft utilization, and unscheduled ratios. Also, the list of checks obtained with the check scheduling algorithm is used to represent the maintenance slots needed to package the tasks. This list contains A and C-checks for the entire fleet for a time horizon of 4 years.

Before the actual packaging process begins, two steps need to be done. The first step is to calculate the maximum manpower allocated to each maintenance check. The input manpower availability contains historical data of the number of technicians available on each day of 2017. This data is used to calculate an estimation of the number of technicians in each weekday of each month. The manpower allocated to a maintenance check corresponds to the sum of the estimated technicians available on each day of the check. If more than one check is scheduled in a day, then the manpower on that day is divided equally between the checks. The second step is to cluster tasks that grant advantages if performed together in the same maintenance slot. The idea is to group tasks with the same interval, required skills, and more importantly, that were performed together in the past. This is an important phase that maintenance planners consider when deciding what tasks must be done in each maintenance slot since it is more efficient to perform tasks with equal or similar due dates and skill requirements. Although tasks are still packaged individually in our proposed approach, this clustering phase will allow us to assess how good it is to package a specific task in a slot, as well as the quality of the final maintenance plan.

Then, the packaging process begins. The idea is to find an optimal sequence of tasks to

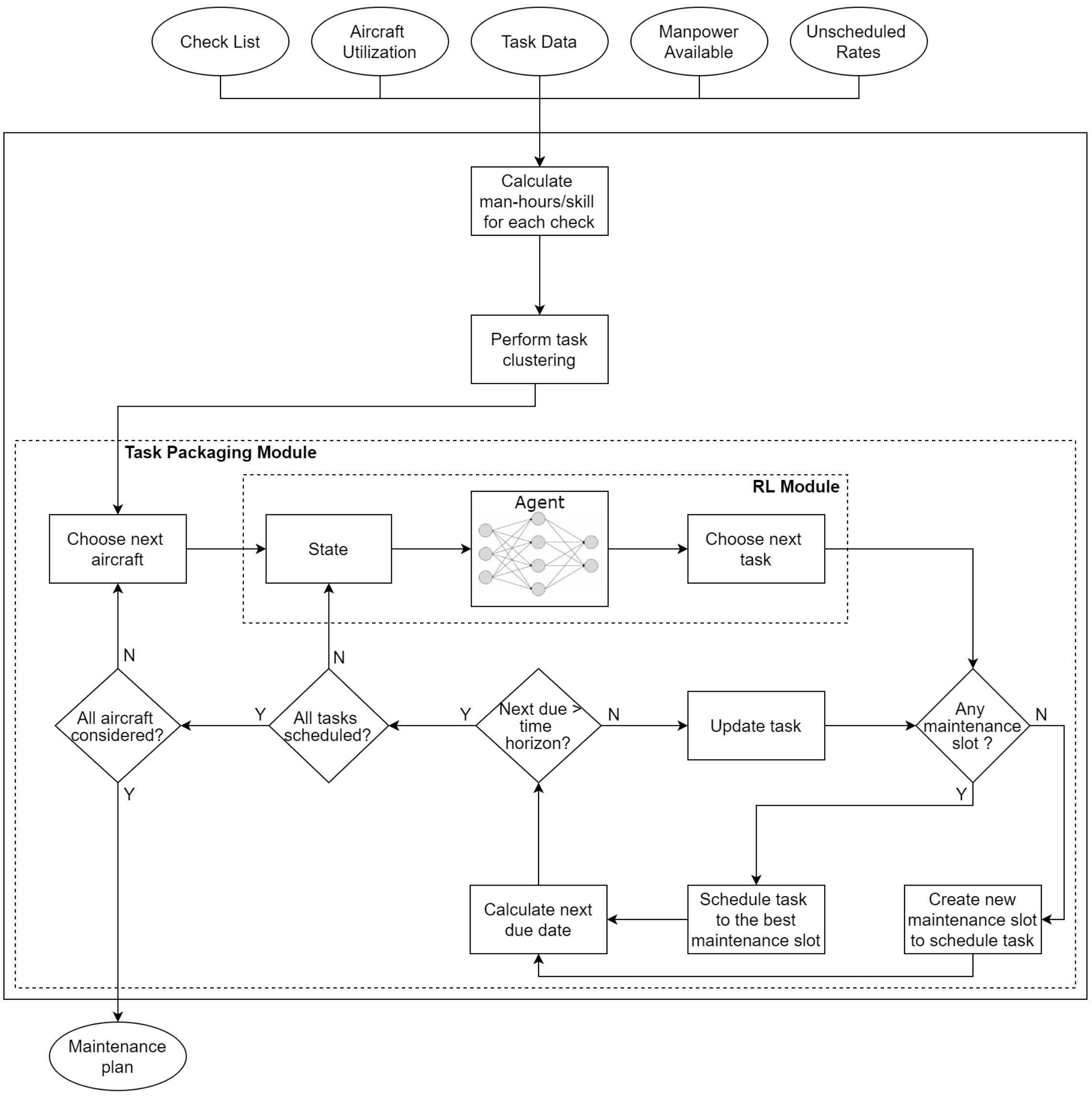


Figure 5.2: Task packaging algorithm flow model.

be packaged for each aircraft. Starting in the first aircraft, the RL agent selects a task that is not yet packaged. If there is at least one maintenance slot available, the task is scheduled to the best one, which corresponds to the one closest to its due date. If not, a new slot is created and the task is scheduled in it. Next, a new due date for the next execution of this same task is calculated and if this due date is lower than the horizon limit, the task is packaged again. When the due date is higher than the horizon limit, the task is successfully scheduled in the entire horizon and the RL agent chooses the next task. The algorithm ends when all tasks have been packaged for every aircraft. At that point, a maintenance plan is obtained, consisting of a set of maintenance slots with the package of tasks that need to be performed in them. This process is represented in Figure 5.2.

The creation of new maintenance slots is an important step in the algorithm. These new slots are created if there is no other available option for a task to be packaged. The reason for this to happen can either be because there are no slots scheduled before the next due date of the task, or because there is no slot with sufficient manpower left to accommodate the task. When this happens, a new maintenance slot is created in the closest working day before the task due date. It is important to note that the aim with these new slots is to represent smaller maintenance opportunities that are typically done at the line in between flights and usually only last for 1-4 hours.

In some situations, even if a task has a due date that is beyond the limit horizon of four years, it may only be possible to schedule it in a slot before that limit. To solve this problem, the actual maintenance plan was reduced to three years and the maintenance slots scheduled to the fourth year are considered to be fictitious checks, used to ensure the balance of the plan, specifically in the final months.

5.3.1 Reinforcement Learning Environment

As mentioned previously, the RL algorithm A3C is used to optimize the proposed packaging approach. The goal is to find an optimal sequence of tasks to be packaged for each aircraft. A crucial part is to define some elements of the RL environment: the state representation, the reward function, and the action set.

State Representation

The packaging of tasks is independent for each aircraft. Therefore, the representation of the environment contains several elements of the aircraft under consideration:

- The unique identifier of the aircraft.
- The list of tasks to be packaged.
- The list of maintenance slots scheduled for the aircraft during the entire horizon.

It is relevant to note that newly created slots during execution are continued to be added to the list of maintenance slots. Also, each slot has a list of the tasks that are already packaged into it (identified by their unique code).

Reward Function

The reward function used contains a penalty equivalent to the cost of packaging a task in a specific maintenance slot. This cost is defined by the following function:

$$Cost = task\ duration \times \frac{1}{task\ usage} \times \frac{1}{No.\ of\ tasks\ from\ same\ cluster} \tag{5.1}$$

The task duration represents the amount of time (in hours) needed to perform the task. The task usage represents how far the task was scheduled from its due date. The usage can assume values in the interval $]0, 1]$. A task usage of 1 means that the task was packaged to a maintenance slot occurring in its due date. The number of tasks from the same cluster represents how many tasks with the same cluster are scheduled in that maintenance slot. At the very minimum, this value is equal to 1 if the task is the only one from its cluster that was packaged in that slot.

Action Set

At each decision step, the RL agent chooses a task to be packaged. Therefore, the action set in this case consists of choosing each one of the tasks, meaning that the number of actions available to the agent is equal to the number of tasks of the aircraft.

5.4 Predictive Task Scheduling

With the proposed task packaging algorithm we obtain the fleet maintenance plan for a time horizon. Although, only the predefined intervals are used to compute the due date of tasks, which means that one of the goals of this work is still not satisfied: the predictive feature.

A predictive task scheduling approach is proposed to meet this goal by using the same A3C algorithm for optimization. Creating a new maintenance plan to deal with prognostic data is not viable. There are flights, crew assignments, and other maintenance details that were defined according to the initial plan and that cannot be changed. Therefore, the idea is to operate small modifications to the already defined maintenance plan in response to new prognostic information, such as new RUL values.

During this work we did not have access to real prognostic data of aircraft components or systems. Thus, the solution found was to produce our own synthetic data to be used in the predictive task scheduling algorithm. A simulation is performed to produce new RUL values using the maintenance plan obtained with the task packaging algorithm. In each iteration, a random day from the plan is chosen. Then, every task that is scheduled after that day has a certain probability of having its due date modified by a new RUL value. This value is calculated using the task interval and the observed remaining lifetime on the chosen day. The RUL values are also computed with the goal of having a more accurate representation of the real environment, in which drastic changes are much less likely than smaller ones. The final step is to calculate the new due dates of the tasks updated by a new RUL, which are then used to modify the plan. More details of this simulation are explained later in the report.

The predictive task scheduling approach takes as input the original maintenance plan and the updated task due dates obtained in the simulation. The first step is to verify for each updated task if it became invalid, that is, if the new due date is now smaller than the scheduled date. If so, the task is removed from its slot and set as an "open" task. Then, the rescheduling process begins. The next "open" task is chosen and all possible maintenance slots are obtained, that is, all slots that are scheduled between the last execution of the task and its due date. If there are no slots available, a new one is created in the closest working day of the due date and the task is scheduled to it. If there are one or more slots available, the RL agent chooses one of them to reschedule the task. In case the selected slot does not have enough allocated manpower, one or more tasks are removed from that slot to free sufficient resources for the scheduling of the new task. The removed tasks are set as "open" so that they can be rescheduled later. Every time a task is rescheduled, it is verified if its next execution is still within the maximum interval allowed. If not, this next instance of the task is removed from the corresponding slot and set as "open". The algorithm ends when all "open" tasks have been rescheduled. A new maintenance plan is obtained along with a history of changes made to the original one. This process is represented in Figure 5.3

In a situation that tasks need to be removed from a slot in order to free resources for the scheduling of another one, the goal should be to remove the smallest number of tasks possible. To accomplish this, it is computed the necessary amount of manpower to free and the task that requires the closest manpower to that value is removed. This process is repeated until enough resources are available.

The history of all changes made to the original plan provides useful information to understand each decision made. For each decision, it includes the task that was updated, the original slot in which it was packaged, its new slot, and the motive that led to this change.

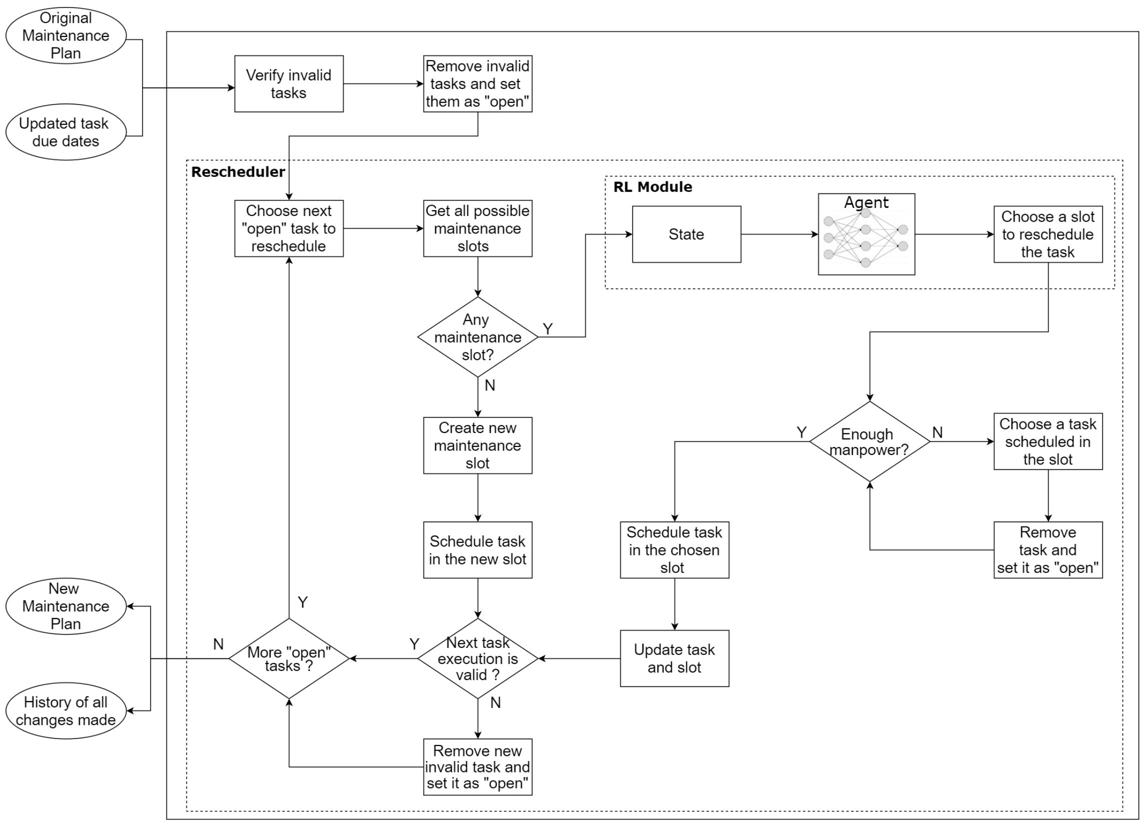

Original Maintenance Plan
Updated task due dates
Verify invalid tasks
Remove invalid tasks and set them as "open"
Rescheduler
Choose next "open" task to reschedule
Get all possible maintenance slots
Any maintenance slot?
Y
N
RL Module
State
Agent
Choose a slot to reschedule the task
Create new maintenance slot
Schedule task in the new slot
Enough manpower?
Y
N
Choose a task scheduled in the slot
Remove task and set it as "open"
Schedule task in the chosen slot
Update task and slot
New Maintenance Plan
History of all changes made
More "open" tasks ?
Y
N
Next task execution is valid ?
Y
N
Remove new invalid task and set it as "open"

Figure 5.3: Predictive task scheduling flow model.

There are three possible motives for a task to be rescheduled: (i) a new RUL obtained that caused the task to become invalid, (ii) the need to free resources in a slot to schedule another task, and (iii) the maximum interval to its previous execution is violated.

5.4.1 Reinforcement Learning Environment

The optimization of the predictive task scheduling algorithm is done with the same A3C algorithm discussed previously. In this case, the goal of the agents is to choose, at each decision step, a maintenance slot to reschedule a task. Again, the three elements necessary for the RL environment need to be defined.

State Representation

The representation of the environment includes several variables:

- The unique identifier of the aircraft.
- The “open” task being rescheduled.
- The list of possible maintenance slots to schedule the task.

Similarly to the task packaging solution, each slot has a list of tasks that are already packaged into it.

Reward Function

The reward function is the same as the one discussed in the task packaging algorithm, which uses the task cost as a penalty.

Action Set

At each decision step, the RL agent chooses a slot in which the current "open" task must be scheduled. There are three actions available to the agent:

- Choose the slot closest to the task due date.
- Choose the slot with the highest amount of manpower still available.
- Choose the slot which contains the higher number of tasks from the same cluster as the task being scheduled.

5.5 Conclusion

In this chapter, a solution to solve the packaging of aircraft maintenance tasks was proposed. The RL algorithm A3C is used to optimize this process, in which the goal is to find an optimal sequence of tasks to be packaged for each aircraft. This approach uses a dataset with real maintenance data from an aircraft fleet along with the list of checks produced by the maintenance check scheduling algorithm discussed previously. The output corresponds to the maintenance plan for the fleet, that is, the list of maintenance slots and packages of tasks to execute in each one.

Also, a predictive task scheduling algorithm is proposed. This solution takes into account prognostic information to update the original maintenance plan by rescheduling the necessary tasks. Since no real prognostic information was available, it was decided to develop a simulation in which new synthetic RUL values are generated. The same A3C algorithm is used for the optimization of the algorithm. The output corresponds to a new updated plan along with the history of changes made, which contains useful information to understand why certain decisions were made.

Chapter 6

Experiments and Results

In this chapter, the experiments and results obtained for all the proposed approaches are presented. The developed models are tested with maintenance data from 51 aircraft. These models were coded in Python 3.7 and two main libraries were used: the *OpenAI Gym* library to build the RL simulated environments, and *Tensorflow* for training the RL agents.

The solution parameters used along with results for the check scheduling stage are presented in Section 6.1. The same information is detailed in Section 6.2 for the task packaging stage. Finally, Section 6.3 presents some conclusions on this topic.

6.1 Check Scheduling Stage

The input for the check scheduling algorithm corresponds to the maintenance check dataset described previously. This input is in the form of an *Excel* file and contains, for each aircraft, the intervals, tolerance, duration, and previous execution dates of A and C-checks. It also includes information on the average daily utilization in FH and FC, for each day of the year, and the problem constraints, such as the weekends, public holidays, and commercial peak seasons for a time horizon of 6 years. The output of the system consists of a *JSON* file with information about all scheduled checks, particularly their starting and ending day, type of check (A or C), and the corresponding aircraft. A second output corresponds to an *Excel* with an easier visualization of the maintenance calendar. An example of these files can be visualized in Appendix B.

To evaluate the results, a set of Key Performance Indicators (KPI) is defined and consists of: the number of A and C-checks scheduled, the average FH for the airline, and the computation time. The average FH corresponds to the FH usage in between maintenance checks. For the first formulation, to compare the two different methods of ordering the maintenance checks, the average number of conflicts per episode was also considered as a Key Performance Indicator (KPI).

The NN and the hyper-parameters for the Deep QL algorithm were obtained using a grid search technique. For each parameter, several values were defined and individually tested to see the ones that produced better results. The final architecture chosen for the NN consists of a multilayer perceptron with two fully connected hidden layers, the first with 500 neurons and the second with 100 neurons, both using *sigmoid* as the activation function. The optimizer used is the *Adam optimizer* (Kingma and Ba, 2014). The remaining hyper-parameters are presented in Table 6.1.

Hyper-parameter	Value	Description
Episodes	500	Total number of training episodes
Max steps	10000	Maximum number of agent steps per episode
Replay memory size	100000	Size of the queue containing the agent experience
Batch size	32	Number of samples taken from replay memory
Discount factor	0.99	Discount factor of future rewards
Learning rate	0.00001	Learning rate used by the optimizer
Initial ε	1	Initial value for exploration
Final ε	0.01	Final value for exploration
Target update	10000	Step frequency to update the target network

Table 6.1: Hyper-parameters for the Deep Q-Learning algorithm.

First Formulation

The results obtained for the first formulation show that the simulation of the work done by maintenance planners in solving conflicts between checks is not efficient. The main reason is that, while the A-checks are all scheduled successfully without violating any constraint, the same does not apply for the C-checks. As the days available to perform C-checks decrease, mainly due to the summer peak season, the number of conflicts hugely increases, which leads to the agent being unable to schedule all C-checks. Therefore, the main focus of this first formulation is to verify the quality of the agent in scheduling A-checks. The time horizon for the planning is four years. The starting day is indicated in the dataset, and corresponds to 20-03-2019. A comparison between the two methods for ordering the checks concerning the predefined KPIs is presented in Table 6.2.

KPI	Time-based order	Aircraft-based order
A-checks avg. FH	716.5	715.2
Total A-checks	958	958
Avg. conflicts per episode	949	1287
Computation time (s)	1723	2142

Table 6.2: Results for the Conflict Solver.

As expected, the time-based order produces fewer conflicts per episode, and because of that, it is also faster. The quality of the results in terms of the average FH for the A-checks is very similar.

Second Formulation

In the second formulation, the RL agent can schedule both A and C-checks and generate a full maintenance calendar for the four years without violating any constraint, including all the C-check ones. Table 6.3 presents some results for the AS algorithm.

KPI	Time-based order
C-check avg. FH	7321.3
Total C-checks	89
A-checks avg. FH	716.5
Total A-checks	958
Computation time (s)	1070

Table 6.3: Results for the Aircraft Scheduler.

It is important to note that, while the majority of the fleet has a C-check interval of 7500 FH, there are 6 aircraft with a higher interval of 12000 FH, which increases the average usage. An alternative to better understand the quality of the scheduling is to calculate the average usage percentage, which corresponds to 92.1%.

The distribution of usages is shown in Figures 6.1, for the A-checks, and 6.2, for the C-checks.

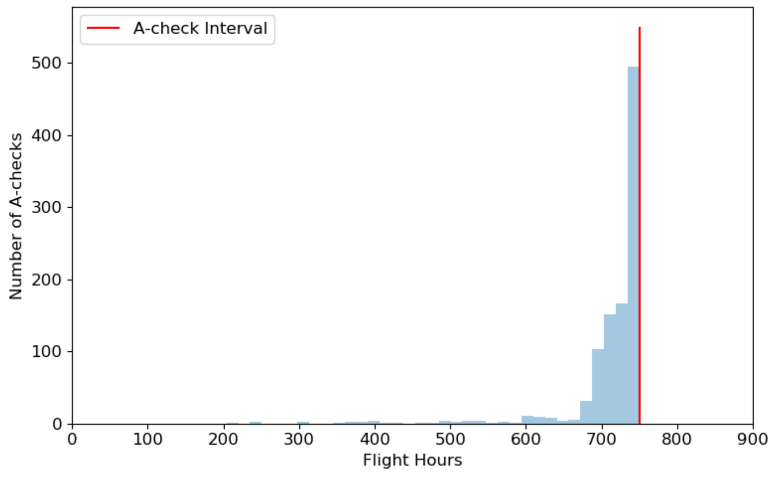


Figure 6.1: Distribution of the A-check usages.

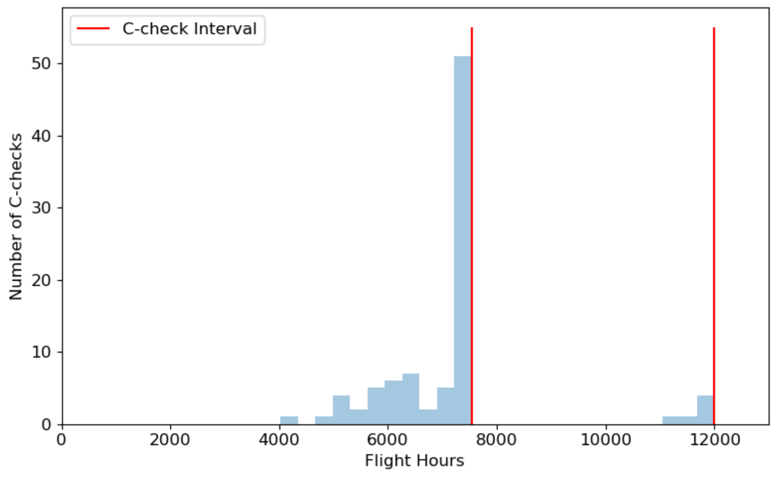


Figure 6.2: Distribution of the C-check usages.

The number of checks scheduled in each month is illustrated in Figure 6.3. This data shows that in the summer peak season (June to September) there are no C-checks scheduled. Also, in the first three months the total checks scheduled is slightly lower. This can be explained by the fact that the initial day of the planning is 20 of March.

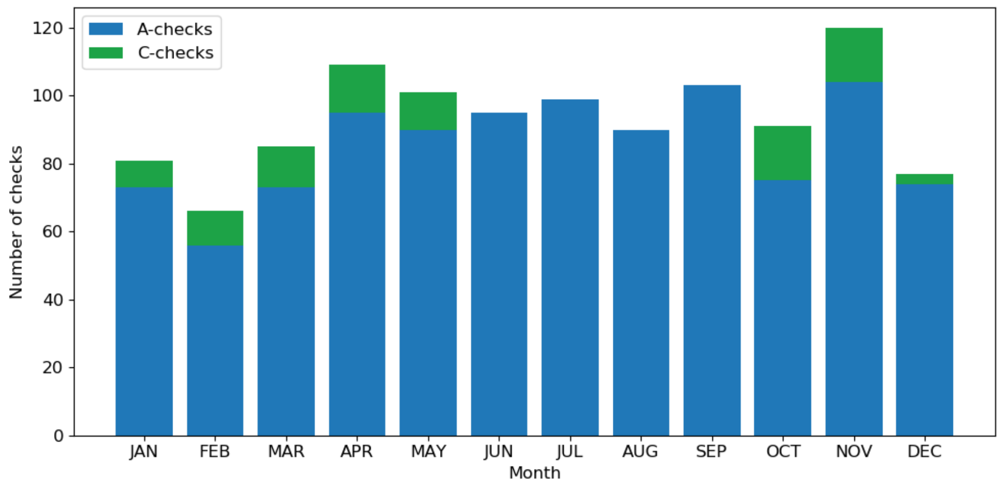


Figure 6.3: Number of checks scheduled in each month.

6.2 Task Packaging Stage

The task packaging algorithm takes as input several *Excel* files, containing information about the available slots for maintenance, aircraft average utilization, manpower available, unscheduled ratios, and general task data. The algorithm produces a 3-year maintenance plan consisting of an *Excel* file for every aircraft, with information about the set of tasks to be performed in each slot. A sample of the maintenance plan can be found in Appendix B.

After testing several combinations of parameters for the A3C algorithm, the final values that achieved better results were chosen and are presented in Table 6.4. The architecture of the global NN is similar to the one used in the Deep QL algorithm. It consists of a multilayer perceptron with two fully connected hidden layers, the first with 300 neurons and the second with 100 neurons. The *sigmoid* activation function is used in both layers with the *Adam* (Kingma and Ba, 2014) as the optimizer.

Hyper-parameter	**Value**	**Description**
Agents	4	Number of agents (threads)
Episodes	100	Total number of training episodes
Max steps	10000	Maximum number of agent steps per episode
Batch size	32	Number of samples taken from the agents experiences
Discount factor	0.99	Discount factor of future rewards
Learning rate	0.0001	Learning rate used by the optimizer

Table 6.4: Hyper-parameters for the A3C algorithm.

The elements of the maintenance plan produced by the task packaging algorithm, such as the manpower utilization per skill, number of tasks allocated, and the amount of RUL wasted can be visualized below. This information helps to assess the quality of the maintenance plan.

An important aspect of the algorithm is related to the manpower available per skill. There are eight skill types, each one with different requirements. The skills 1-4 represent 78% of the required manpower on average for each aircraft, because the majority of the tasks need one of those skills to be performed. The eight skill types and the percentage required for each one per aircraft are shown in Figure 6.4. It is also relevant to note that skills 6 and 7 are reserved for C-checks, meaning that tasks which need one of these skills must be packaged in a C-check.

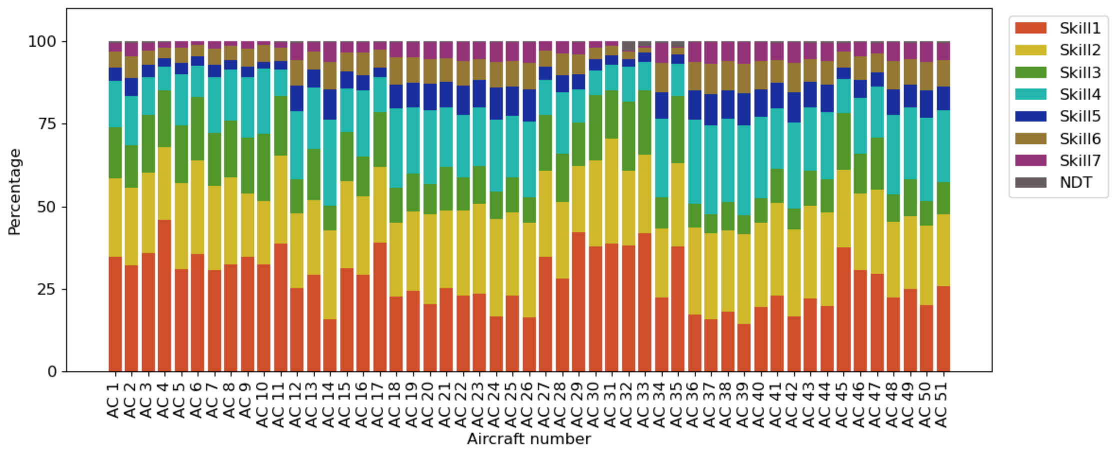


Figure 6.4: Percentage of manpower required per skill for each aircraft.

The manpower input file contains the number of technicians available per skill on each day. Because the duration of the tasks is expressed in hours, it is necessary to also convert the manpower value in hours. In the first phase, it was assumed that each technician works 8 hours per day. However, the results showed that the number of available hours unused in each slot was very high. An example of this issue is shown in Figure 6.5 for aircraft 26, in which the percentage of utilization per skill for every A-check is presented. The results indicate that even after all the tasks have been packaged there are a lot of unused hours in each slot, which suggests that the number of hours available is too high.

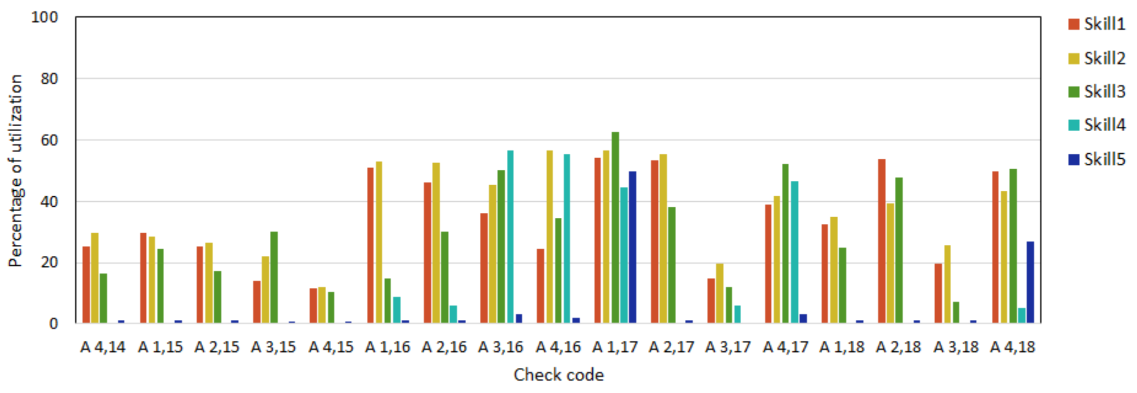


Figure 6.5: Percentage of manpower used per skill in each A-check of aircraft 26. Each technician represents 8 hours of work per day.

This issue has a relevant effect on the packaging algorithm because, with the high amount of available manpower, each slot can probably contain a lot more tasks than it would probably do in reality. To solve this problem, the number of hours that each technician represents was reduced until the amount of unused manpower was minimal. Figures 6.6 and 6.7 show the percentage of manpower used in each A-check of aircraft 26 in which a technician represents 6 and 4 hours of manpower, respectively. In the first case, the amount of unused manpower is still considerably high, whilst in the second one there are some checks close to maximum capacity.

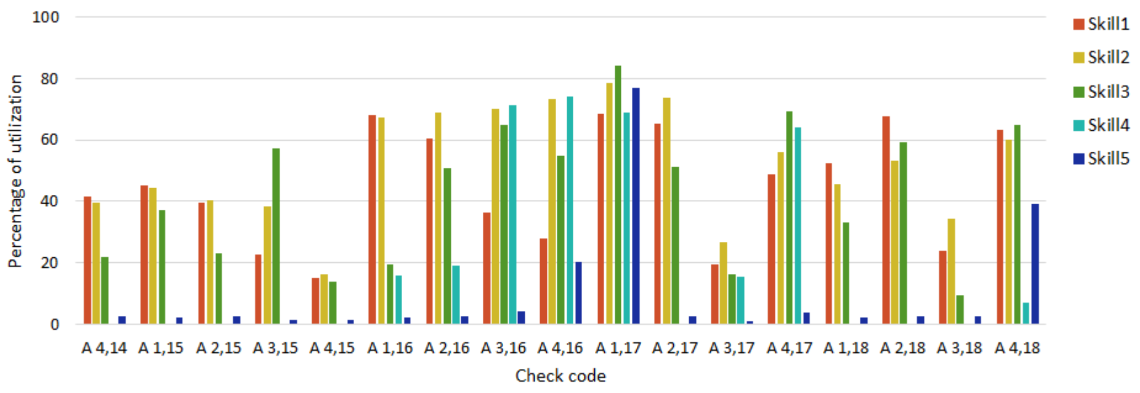


Figure 6.6: Percentage of manpower used per skill in each A-check of aircraft 26. Each technician represents 6 hours of work per day.

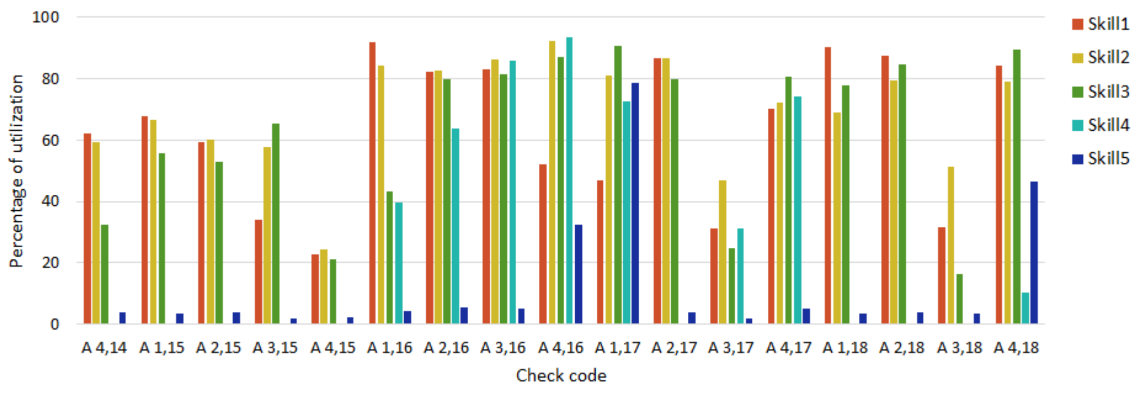


Figure 6.7: Percentage of manpower used per skill in each A-check of aircraft 26. Each technician represents 4 hours of work per day.

If the available manpower is reduced even more, most checks start to reach full capacity in some of the skills as illustrated in Figure 6.8, in which a technician represents 3 hours of manpower. This is not always good because while the manpower wasted is very low, the number of additional slots that are created to perform tasks that could not be packaged in the checks will grow. Therefore, the decision was to consider that each technician represents 4 hours of manpower.

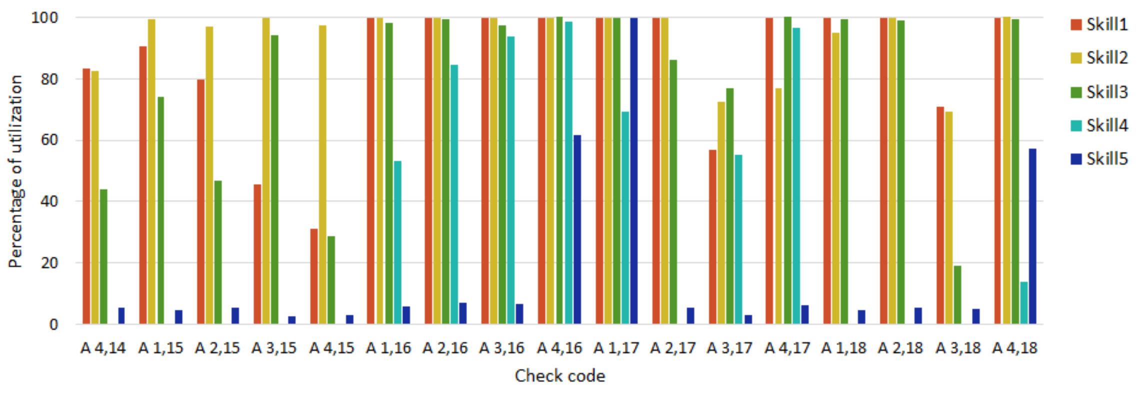


Figure 6.8: Percentage of manpower used per skill in each A-check of aircraft 26. Each technician represents 3 hours of work per day.

The tasks packaged in the predefined maintenance checks represent 90.1% of the total packaged tasks, while the remaining 9.9% is packaged in the smaller slots created during the execution of the algorithm. Figure 6.9 illustrates this data. There are three aircraft with a much smaller number of tasks (aircraft 32, 33, and 35). The reason is that all of them are phased-out early in the planning horizon, meaning that they will stop flying and being maintained.

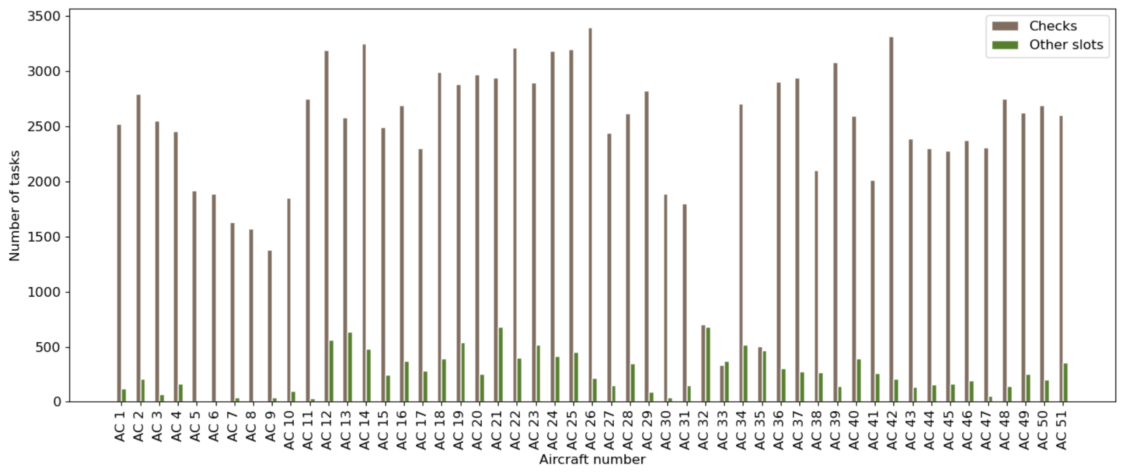


Figure 6.9: Total tasks scheduled by aircraft.

On average, from the tasks packaged in checks, 80.2% is packaged in A-checks and 19.8% is packaged in C-checks, which was already expected because of the intervals of both check types. Although, in most cases, C-checks are much heavier than A-checks, which is exemplified in Figure 6.10 for aircraft 26.

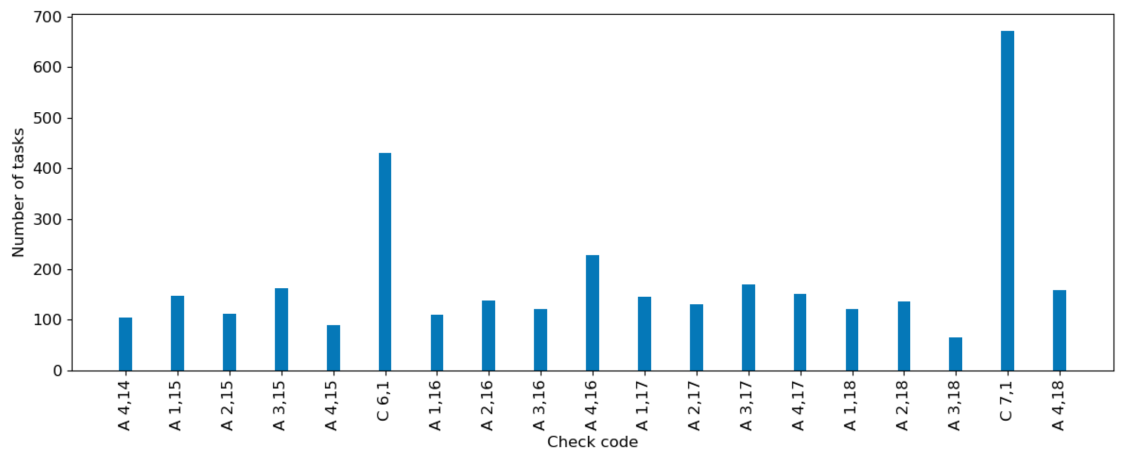


Figure 6.10: Number of tasks per check (aircraft 26).

The sum of task costs for each aircraft is represented in Figure 6.11. This data is heavily affected by the total tasks packaged. For that reason, the phased-out aircraft (32, 33, and 35) have a lower sum of costs than the remaining aircraft. A second element that contributes to the task cost is the amount of RUL wasted by not performing a task on its due date. Figure 6.12 presents the sum of RUL wasted for each aircraft.

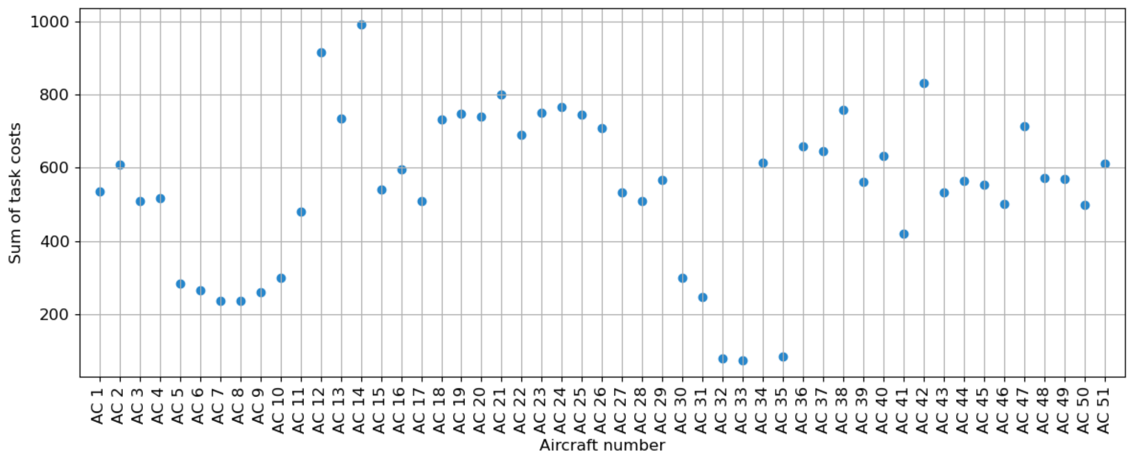


Figure 6.11: Sum of task costs per aircraft.

The last relevant information also related with the RUL wasted is the task usage, which indicates how much of the component lifetime was actually used. A task usage of 1 means that the task was packaged in a slot occurring in its due date. Figure 6.13 illustrates the distribution of task usage for the entire fleet.

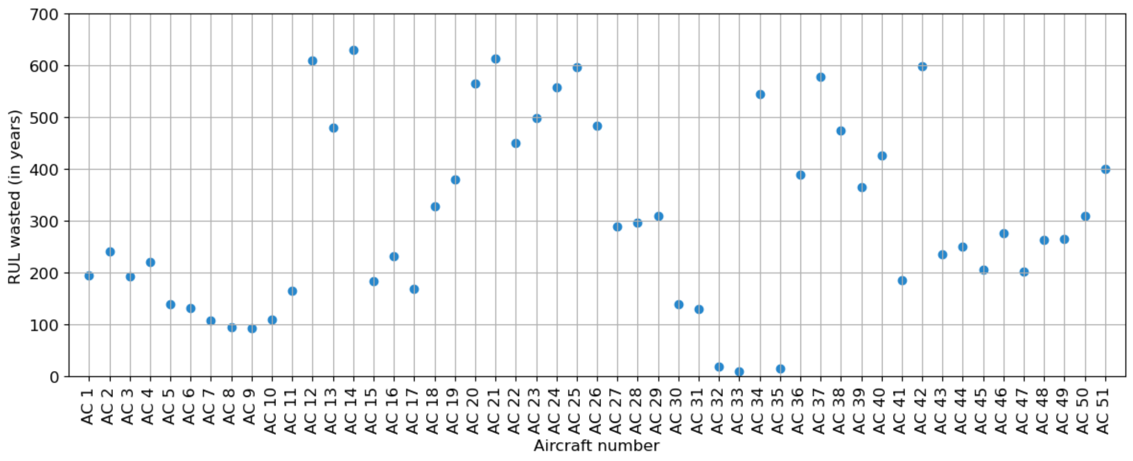


Figure 6.12: Amount of RUL wasted (in years) per aircraft.

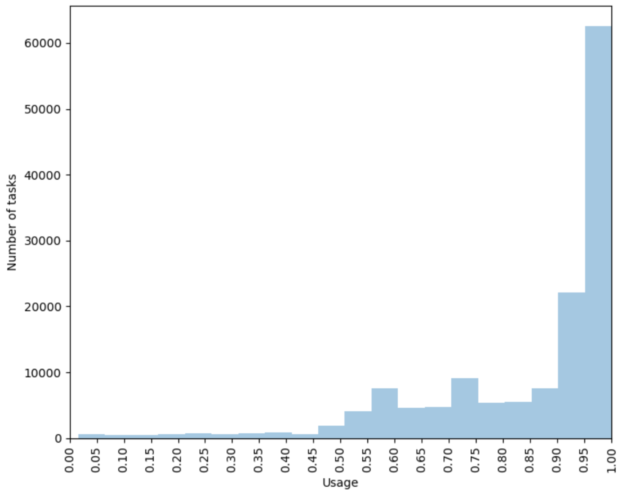


Figure 6.13: Distribution of task usages for the entire fleet.

To understand this usage data it is important to relate the task intervals with the predefined check intervals. Table 6.5 presents the percentage of tasks whose intervals (in FH) differ between certain ranges. As detailed in the table, half of the tasks have an interval between 750 and 800 FH. The A-check interval is also 750FH, meaning that half the tasks are performed at every A-check and have a usage close to 1 as illustrated in the usage data. Also, almost 20% of the tasks have an interval between 1000 and 1400 FH, which is less than 1500FH (the ideal interval to perform a task every two A-checks). This means that these tasks have usages ranging between 0.5 and 0.75 because, in most cases, they also have to be performed every A-check.

Interval range (FH)	**Percentage of tasks**
[750, 800]	50.2%
[1000, 1400]	19.6%
[1500, 3000]	15.9%
> 3000	14,3%

Table 6.5: Percentage of tasks per interval range (in FH).

6.2.1 Predictive task scheduling

The predictive task scheduling algorithm performs some necessary updates to the predefined 3-year maintenance plan in response to new RUL values. Due to the lack of real prognostic data, these values were synthetically produced in a simulation. At each iteration, one day from the 3-year plan is chosen as the new current day of the planning. Then, each task scheduled after this current day has a 1% chance of having its RUL updated. Because the average number of different tasks per aircraft is 994, each aircraft should have, on average, 10 new RUL values per iteration. To compute a new RUL, four levels are defined, which indicate the difference between the current lifetime value (calculated using aircraft utilization estimations) and the new one. In each level, the new RUL is calculated by multiplying the current lifetime by a random factor between two thresholds:

- Level 1: the new RUL is at most 0.1 times greater than the current estimated value.
- Level 2: the new RUL is at most 0.1 times less than the current estimated value.
- Level 3: the new RUL is between 0.1 and 0.2 times less than the current estimated value.
- Level 4: the new RUL is between 0.2 and 0.5 times less than the current estimated value.

These levels represent different degradation rates. Level 1 represents a situation in which the component has degraded less than expected. The other 3 levels represent a situation in which the component degraded more than expected, with increasing rates by each level.

The goal of this design is to have a representation of what happens in the real environment. Therefore, the defined levels have different probabilities of being chosen to compute a new RUL. These probabilities are 20% for level 1, 45% for level 2, 30% for level 3, and 5% for level 4. The idea is that drastic changes in the lifetime of a component are less likely than smaller ones.

The predictive task scheduling algorithm uses the same A3C algorithm discussed previously. Three scenarios were simulated to test every section of the plan, by fixing the chosen day of each iteration. In the first scenario, the initial day chosen is 1/7/2019. The second scenario starts at 1/7/2020, and the third scenario starts at 1/7/2021. In total, 50 iterations were performed for each scenario and the averages regarding some KPI are presented in Table 6.6. The cost of the plan corresponds to the sum of all task costs. Another indicator is related to the motive that led to a decision, that is, to the rescheduling of a task. As mentioned before, there are three possible motives: a new RUL obtained (motive 1), the need to free manpower resources in a slot (motive 2), and the violation of the maximum interval allowed for the last execution of the task (motive 3).

KPI	**Scenario 1**	**Scenario 2**	**Scenario 3**
Old plan cost	36 342	20 942	5849
New plan cost	36 221	20 904	5841
Total checks	715	412	117
New created slots	29	22	25
Decisions by motive 1	158	116	81
Decisions by motive 2	391	215	85
Decisions by motive 3	81	39	2

Table 6.6: Results for the predictive task scheduling algorithm.

The cost of the new plan in all scenarios is very similar to the cost of the old plan. Although, it is important to note that these costs represent only the sum of task costs. There are also new slots created that need to be considered to assess the overall quality of the plan. The number of decisions made, that also represent the number of tasks rescheduled, decreases from scenario 1 to 2, and from 2 to 3. This is expected since the number of slots and tasks decreases over the scenarios due to their initial day chosen in the simulation.

6.3 Conclusion

The conducted experiments for the check scheduling approach described in this chapter show different results for the two formulations. The CS failed to schedule all C-checks due to a large amount of constraints that were added. The AS, which was developed to overcome these problems, was able to schedule all A and C-checks with success. The check calendar obtained with this formulation has good results in some important metrics, such as the total number of scheduled checks and their average utilization.

In the task packaging stage, after running the algorithm for the first time, it became obvious that the available manpower per day was too high. Therefore, one of the first steps was to reduce it by decreasing the value of working hours that each technician represents. The obtained distribution of task usage shows that most of the tasks have a usage close to 1. This is relevant since a higher usage means that the amount of RUL wasted is low and consequently, fewer tasks are performed in the long run. A big contribution to this result is the fact that a high percentage of tasks have an interval close to the A-check interval.

A simulation was also developed to produce synthetic RUL data to be used in the predictive task scheduling algorithm, which aims at producing a new maintenance plan by performing a sequence of adjustments to the original one. Three scenarios were designed and simulated to test every section of the plan. Each one of them has a different starting day, corresponding to the middle of each year of the plan. The results indicate that, in all scenarios, the cost of the plan after the updates was slightly lower than the original one, which is a sign that the algorithm is able to produce a good alternative plan by performing the best possible adjustments.

Chapter 7

Validation

The methods for the validation of all the proposed approaches are presented in this chapter. The check scheduling algorithm is compared with a similar study in Section 7.1 and the corresponding results are presented. In Section 7.2, the task packaging algorithm is validated by ensuring that certain conditions on the produced maintenance plan remain true. Also, the predictive task scheduling approach is compared to a new method, that uses the packaging algorithm with the necessary adjustments to the initial conditions to build an entirely new plan. Finally, in Section 7.3, a summary of the results and main conclusions is presented.

7.1 Check Scheduling Stage

To evaluate the quality of the maintenance plans produced with the check scheduling algorithm and validate this solution, it is performed a comparison with the work done in Deng et al. (2019), in which the authors developed a practical Dynamic Programming (DP) methodology to solve the same problem using the same dataset. Some parameters had to be adjusted to match the DP approach, namely the start date of the planning, and the total aircraft considered, which was reduced from 51 to 45, with 6 aircraft not used. The time horizon considered is four years, and the generated plan is also compared with the one created by the airline. However, the A-check metrics were not used in this comparison because the airline only plans A-checks for the next year.

In Deng et al. (2019), additional KPIs are used to assess the quality of the results: tolerance events, which correspond to the number of checks that are scheduled using the tolerance, and the number of A-checks merged into C-checks. To determine when a merging can occur, it is defined that if the due date of an A-check is less than 20 days away from the end date of a previous C-check, then it is merged. Table 7.1 presents the comparison of the three solutions.

As the results suggest, the RL approach generates better maintenance plans than the other two. This is only possible because the developed DP based approach is not an exact method, mainly due to the very large outcome space that had to be reduced.

In regards to the C-checks, there is a clear improvement in the average usage, which leads to 2 fewer C-checks scheduled in the four years. This result has great significance when we know that each C-check represents several full days of ground time for the aircraft without any profit for the company. In the A-checks, the usage results for the airline are

KPI	Airline	DP	RL
Total C-checks	96	88	86
C-check Average FH	<6600	6615.2	7013.4
Tolerance Events	6	4	1
Total A-checks	895-920	877	879
A-check Average FH	—	717.6	716.8
Tolerance Events	—	0	0
Merged A-checks	—	18	15
Computation Time (s)	$\geq$ 3 Days	510.3	1023

Table 7.1: Results of the plans generated with the RL Aircraft Scheduler, compared with a DP approach and the airline approach.

not considered, but the total number of A-checks scheduled with RL and the average FH is very similar to the DP approach, which also helps validate the RL solution. Finally, the computation time, while being higher than the time taken in the DP approach, it is still nowhere near the time needed by the airline to develop the maintenance plan, which takes a minimum of three days.

7.2 Task Packaging Stage

No similar studies could be compared with the task packaging algorithm. Therefore, in order to verify that the obtained solution is valid, the following conditions need to be satisfied:

- All aircraft have all of their tasks scheduled for the 3-year plan.
- The newly created maintenance slots are all scheduled in working days.
- The required manpower per skill in each slot is not higher than its maximum allocated manpower.
- The required manpower per skill in each day is not higher than the maximum available manpower on that day.
- All tasks are packaged in a slot that starts before or on their due date.
- The maximum interval between two instances of the same task is always respected.
- The same task is not packaged twice in the same slot.
- Tasks that need to be performed in a specific slot type are only packaged in slots of that type.

After verifying that these conditions are all satisfied, the next step is to validate the predictive task scheduling algorithm. The maintenance plan obtained with this approach is compared to the plan produced by a second method, which uses the task packaging solution with the same initial conditions. This second method takes the simulation parameters as an additional input, namely the new RUL values and the new start day of the plan, and creates a new maintenance plan using the task packaging algorithm. The same three scenarios defined previously are used to test every section of the planning. Scenario 1 starts at 1/7/2019, scenario 2 starts at 1/7/2020, and scenario 3 starts at 1/7/2021. The results for both methods in each scenario are presented in Table 7.2.

Scenario	KPI	Predictive Task Scheduler	Full Task Packaging	Diff.
1	Plan Cost	36 221	31 109	-14.1%
	Number of slots	1578	1261	-20.1%
	Number of tasks	104 237	97 682	-6.3%
2	Plan Cost	20 904	18 354	-12.2%
	Number of slots	1032	860	-16.7%
	Number of tasks	70 924	67 521	-4.8%
3	Plan Cost	5841	5657	-3.2%
	Number of slots	260	245	-5.8%
	Number of tasks	14 630	14 201	-2.9%

Table 7.2: Results of the plans generated with the predictive task scheduling algorithm and with the full task packaging method.

It is expected that creating an entirely new plan from the start will produce better results than performing a series of small modifications to an already existing plan. The main reason is that every task rescheduled can originate new sub-optimal modifications, such as the removal of other tasks to free up resources. The results confirm this idea, with the method of performing a full task packaging having a lower plan cost, and fewer slots and tasks than the predictive task scheduling algorithm, which only produces small modifications to the existing plan. Although, in the third scenario the results between the two methods are very similar, which suggests that the predictive task scheduling approach can produce efficient plans with a shorter horizon.

It is relevant to note that, in reality, it would not be possible or viable to fully reset the current plan to a new one every time new prognostic information is available. There are flights already scheduled, material availability, manpower allocated, and other maintenance items that are already planned for the following weeks or even months, and that can not be changed. The time that it takes to do a full task packaging is also much higher than to perform just the necessary updates to the existing plan. Therefore, the predictive task scheduling algorithm could be used to produce an efficient plan for a shorter horizon, while the full task packaging solution could be useful to obtain a baseline plan for a larger period of time.

7.3 Conclusion

The check scheduling algorithm was validated by comparing it with another study done in Deng et al. (2019), in which the same problem is solved using a DP approach. The results of the developed RL approach are compared with the DP results and with aircraft estimations. The RL approach is the more efficient to schedule C-checks, having achieved a higher average FH usage and a reduction in the number of checks scheduled in the 4-year calendar. The results for the A-check scheduling are very similar to the DP approach and represent a great improvement over the airline method, which estimates a higher number of A-checks. This comparison helps to prove that the proposed check scheduling algorithm can be effective in solving this problem.

For the task packaging algorithm, no similar studies could be compared. Therefore, a set of conditions were defined to verify that the produced maintenance plan is valid. The predictive task scheduling algorithm was validated by comparing it with a second method. This new method corresponds to using the task packaging algorithm with modified initial conditions, namely a new plan start day, and the new RUL values, which are used to update the initial due dates of the corresponding tasks. The results of this comparison prove that the predictive task scheduling, which produces a series of modifications to the existing plan, can be almost as effective as doing a full task packaging for a shorter horizon.

Chapter 8

Conclusion

This thesis aimed at developing a learning model, using Deep Reinforcement Learning (RL), to generate and optimize a maintenance plan for an aircraft fleet.

The main goal, in the first stage, was to optimize the scheduling of aircraft maintenance checks for a specified time horizon. A Deep Q-Learning algorithm was used, and two formulations were developed to achieve this goal. In the first formulation, the Conflict Solver (CS), the aim is to solve conflicts between maintenance checks as they appear in the planning. At each decision step, the RL agent chooses one conflicting check to be rescheduled. In the second formulation, the Aircraft Scheduler (AS), the agent chooses an aircraft, at each decision step, and the next check for that aircraft is scheduled to the best date available. Both of these formulations were tested with a dataset containing real maintenance data for a set of aircraft, and a set of constraints. To validate the obtained solution, a comparison is performed with estimations made by the actual airline and with the results of a Dynamic Programming (DP) approach. Based on this comparison, we can conclude that the proposed RL approach is able to efficiently schedule aircraft maintenance checks.

In the second stage, a task packaging approach was developed to package a set of maintenance tasks of an aircraft in the respective maintenance slots. The Asynchronous Advantage Actor-Critic (A3C) algorithm was used to optimize the packaging process by having the agents choose the next task to be packaged. This approach was also tested with real maintenance data, which includes general task information, estimations of aircraft utilization, manpower availability, and unscheduled maintenance ratios. The maintenance checks produced in the first stage of this work are used as the maintenance slots available to package the tasks. Additionally, smaller maintenance slots are created if the tasks can not be packaged in the predefined checks. The cost of packaging a task in a slot combines its duration, usage, and the total number of tasks from the same cluster already packaged in that slot. This clustering step is done at the beginning by grouping tasks with the same interval, skill requirements, and that were performed together in the past.

Another contribution of this work was to use prognostic information in the maintenance decision process by developing a predictive task scheduling algorithm, which uses the same A3C algorithm for optimization. The goal is to perform a few modifications to the predefined 3-year maintenance plan in response to due date changes caused by new simulated Remaining Useful Life (RUL) values. This approach was compared to the method of doing a full task packaging and it was proved that both of them produce similar results for a shorter time horizon. Because it is not viable to create an entirely new maintenance plan each time new prognostic data is obtained, the predictive task scheduling algorithm can be used to perform only the necessary adjustments to the already existing plan for a shorter horizon. On the other hand, the task packaging algorithm can be used to periodically reset the plan and ensure a lower overall cost in the long term. Nevertheless, the results of both methods prove that RL can be an effective option in solving this maintenance optimization problem.

8.1 Future Work

Regarding future work, the proposed approaches can be improved by considering other maintenance variables. In reality, the maintenance plan is secondary when compared to the operation plan, which contains the flights scheduled for each aircraft. The scheduling of maintenance for an aircraft is always dependent on its operation and not the other way around. There are other task related elements such as the required access panels and material resources that could be used to produce a better and more accurate maintenance plan.

The developed predictive task scheduling algorithm can also be improved to update the maintenance plan according to new decisions or constraints proposed by maintenance engineers, in addition to the updates that originate from new prognostic information. For instance, it is common for a task to be anticipated or deferred to a different maintenance slot due to some event, such as an unexpected failure, the benefit of performing it together with another task that requires the opening of the same access panel, or any other convenience factor. The proposed approach could be extended to produce an updated maintenance plan with multiple types of additional constraints.

The results obtained with the RL based approaches applied to the aircraft maintenance scheduling problem were very promising, which encourages the continuation of the work in the future.

References

[ACARE - Strategic Research and Innovation Agenda - Volume 2] ACARE - Strategic Research and Innovation Agenda - Volume 2. – `https://www.acare4europe.org/acare-db?keys=CBM&field_cluster_enablers_text_value=&field_enabler_value=&field_capability_value=`. Accessed: 2020-04-18

[Ackert 2010] Ackert, Shannon P.: Basics of Aircraft Maintenance Programs for Financiers: Evaluation & Insights of Commercial Aircraft Maintenance Programs. (2010). – Retrieved from `http://www.aircraftmonitor.com/uploads/1/5/9/9/15993320/basics_of_aircraft_maintenance_programs_for_financiers___v1.pdf`. Accessed: 2020-08-10

[Ahmad and Kamaruddin 2012] Ahmad, Rosmaini ; Kamaruddin, Shahrul: An overview of time-based and condition-based maintenance in industrial application. In: *Computers and Industrial Engineering,* 63 (2012), Nr. 1, pp. 135–149

[Arulkumaran et al. 2017] Arulkumaran, Kai ; Deisenroth, Marc P. ; Brundage, Miles ; Bharath, Anil A.: Deep Reinforcement Learning: A Brief Survey. In: *IEEE Signal Processing Magazine,* 34 (2017), Nr. 6, pp. 26–38

[Asif 2013] Asif, Imran: How can airplane operators reduce maintenance cost. In: *Special Topic Paper for PhD. Program at Embry-Riddle Aeronautical University, USA* (2013)

[Barde et al. 2019] Barde, Stephane R. ; Yacout, Soumaya ; Shin, Hayong: Optimal preventive maintenance policy based on reinforcement learning of a fleet of military trucks. In: *Journal of Intelligent Manufacturing,* 30 (2019), Nr. 1, pp. 147–161

[Başdere and Bilge 2014] Başdere, Mehmet ; Bilge, Ümit: Operational aircraft maintenance routing problem with remaining time consideration. In: *European Journal of Operational Research,* 235 (2014), Nr. 1, pp. 315–328

[Block et al. 2008] Block, Marco ; Bader, Maro ; Tapia, Ernesto ; Ramírez, M. ; Gunnarsson, Ketill ; Cuevas, Erik ; Zaldivar, Daniel ; Rojas, Raúl: Using reinforcement learning in chess engines. In: *Research in Computing Science,* 35 (2008), pp. 31–40

[Bu et al. 2009] Bu, Xiangping ; Rao, Jia ; Xu, Cheng Z.: A reinforcement learning approach to online web systems auto-configuration. In: *Proceedings - International Conference on Distributed Computing Systems* (2009), pp. 2–11

[Cook and Tanner 2008] Cook, Andrew ; Tanner, Graham: Innovative Cooperative Actions of R&D in EUROCONTROL Programme CARE INO III: Dynamic Cost Indexing: Aircraft crewing – marginal delay costs. 2008. – Technical report [de Bruin et al. 2015] de Bruin, Tim ; Kober, Jens ; Tuyls, Karl ; Babuska, Robert: The importance of experience replay database composition in deep reinforcement learning. In: *Deep Reinforcement Learning Workshop, Advances in Neural Information Processing Systems (NIPS)*, 2015

[Deng et al. 2019] Deng, Qichen ; Santos, Bruno F. ; Curran, Richard: A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization. In: *European Journal of Operational Research,* 281 (2019), Nr. 2, pp. 256–273

[European Committee for Standardization 2017] European Committee for Standardization: Maintenance terminology, Brussels. (2017)

[Feo and Bard 1989] Feo, Thomas A. ; Bard, Jonathan F.: Flight Scheduling and Maintenance Base Planning. In: *Management Science,* 35 (1989), Nr. 12, pp. 1415–1432

[Ferreira 2019] Ferreira, Pedro: *Deep Reinforcement Learning for Condition Based Monitoring in Aircraft Planning*, University of Coimbra, Master Thesis, 2019

[Fischer et al. 2010] Fischer, Katharina ; Bertling Tjernberg, L ; Besnard, François ; Member, Student ; Bertling, Lina ; Member, Senior: An Approach for Condition-Based Maintenance Optimization Applied to Wind Turbine Blades. In: *IEEE Transactions on Sustainable Energy,* 1 (2010), Nr. 2, pp. 77–83

[Glorennec 2000] Glorennec, Pierre Y.: Reinforcement Learning: an Overview. In: *European Sym. on Intelligent Techniques*, 2000

[H2020 ReMAP] H2020 ReMAP: Real-time Condition-based Maintenance for Adaptive Aircraft Maintenance Planning. `https://h2020-remap.eu/`. Accessed: 2020-08-30.

[Hasselt 2010] Hasselt, Hado V.: Double Q-learning. In: *Advances in Neural Information Processing Systems 23.* Curran Associates, Inc., 2010, pp. 2613–2621

[Hasselt et al. 2016] Hasselt, Hado V. ; Guez, Arthur ; Silver, David: Double DQN. In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)* (2016), pp. 2094–2100

[Hausknecht and Stone 2015] Hausknecht, Matthew ; Stone, Peter: Deep recurrent q-learning for partially observable MDPs. In: *AAAI Fall Symposium - Technical Report,* FS-15-06 (2015), pp. 29–37

[Huang et al. 2018] Huang, Chengqiang ; Wu, Yulei ; Zuo, Yuan ; Pei, Ke ; Min, Geyong: Towards experienced anomaly detector through reinforcement learning. In: *32nd AAAI Conference on Artificial Intelligence, AAAI 2018* (2018), pp. 8087–8088

[IATA's Maintenance Cost Task Force 2016] IATA's Maintenance Cost Task Force: In: *Airline Maintenance Cost Executive Commentary* (2016)

[IATA's Maintenance Cost Task Force 2017] IATA's Maintenance Cost Task Force: In: *Airline Maintenance Cost Executive Commentary* (2017)

[Jardine et al. 2006] Jardine, Andrew K. ; Lin, Daming ; Banjevic, Dragan: A review on machinery diagnostics and prognostics implementing condition-based maintenance. In: *Mechanical Systems and Signal Processing,* 20 (2006), Nr. 7, pp. 1483–1510

[Kingma and Ba 2014] Kingma, Diederik ; Ba, Jimmy: Adam: A Method for Stochastic Optimization. In: *International Conference on Learning Representations* (2014)

[Knowles et al. 2011] Knowles, Michael ; Baglee, David ; Wermter, Stefan: Reinforcement learning for scheduling of maintenance. In: *Res. and Dev. in Intelligent Syst. XXVII: Incorporating Applications and Innovations in Intel. Sys. XVIII - AI 2010, 30th SGAI Int. Conf. on Innovative Techniques and Applications of Artificial Intel.* (2011), pp. 409–422

[Koomsap et al. 2005] Koomsap, P ; Shaikh, N I. ; Prabhu, V V. ; Koomsapy, P ; Shaikhz, N I. ; Prabhuz, V V.: Integrated process control and condition-based maintenance scheduler for distributed manufacturing control systems. In: *International Journal of Production Research,* 43 (2005), Nr. 8, pp. 1625–1641

[Li et al. 2016a] Li, Zhaojun ; Guo, Jian ; Zhou, Ruolin: Maintenance scheduling optimization based on reliability and prognostics information. In: *Proceedings - Annual Reliability and Maintainability Symposium* (2016)

[Li et al. 2016b] Li, Zhe ; Wang, Kesheng ; He, Yafei: Industry 4.0-Potentials for Predictive Maintenance. In: *International Workshop of Advanced Manufacturing and Automation (IWAMA 2016) Industry*, 2016

[Liang and Chaovalitwongse 2009] Liang, Zhe ; Chaovalitwongse, Wanpracha: The Aircraft Maintenance Routing Problem. In: *Operations Research,* 46 (2009), pp. 327–348

[Luxhøj et al. 1997] Luxhøj, James T. ; Riis, Jens O. ; Thorsteinsson, Uffe: Trends and Perspectives in Industrial Maintenance Management. In: *Journal of Manufacturing Systems,* 16 (1997), Nr. 6, pp. 437–453

[Mattila and Virtanen 2011] Mattila, Ville ; Virtanen, Kai: Scheduling fighter aircraft maintenance with reinforcement learning. In: *Proceedings - Winter Simulation Conference* (2011), pp. 2535–2546

[Mnih et al. 2016] Mnih, Volodymyr ; Badia, Adria P. ; Mirza, Mehdi ; Graves, Alex ; Lillicrap, Timothy ; Harley, Tim ; Silver, David ; Kavukcuoglu, Koray: Asynchronous Methods for Deep Reinforcement Learning. In: *Proceedings of The 33rd International Conference on Machine Learning,* 48 (2016), pp. 1928–1937

[Mnih et al. 2015] Mnih, Volodymyr ; Kavukcuoglu, Koray ; Silver, David ; Rusu, Andrei A. ; Veness, Joel ; Bellemare, Marc G. ; Graves, Alex ; Riedmiller, Martin ; Fidjeland, Andreas K. ; Ostrovski, Georg ; Petersen, Stig ; Beattie, Charles ; Sadik, Amir ; Antonoglou, Ioannis ; King, Helen ; Kumaran, Dharshan ; Wierstra, Daan ; Legg, Shane ; Hassabis, Demis: Human-level control through deep reinforcement learning. In: *Nature,* 518 (2015), Nr. 7540, pp. 529–533

[Nguyen and Bagajewicz 2008] Nguyen, Duy Q. ; Bagajewicz, Miguel: Optimization of preventive maintenance scheduling in processing plants. In: *Computer Aided Chemical Engineering,* 25 (2008), Nr. 2006, pp. 319–324

[Periyarselvam et al. 2013] Periyarselvam, U ; Tamilselvan, T ; Thilakan, S ; Shanmugaraja, M: Analysis on Costs for Aircraft Maintenance. In: *Research India Publications,* 3 (2013), Nr. 3, pp. 177–182

[Pleumpirom and Amornsawadwatana 2012] Pleumpirom, Yuttapong ; Amornsawadwatana, Sataporn: Multiobjective optimization of aircraft maintenance in Thailand using goal programming: A decision-support model. In: *Advances in Decision Sciences,* 2012 (2012)

[Samaranayake 2006] Samaranayake, Premaratne: Current practices and problem areas in aircraft maintenance planning and scheduling: interfaced/integrated system perspective. In: *Proceedings of the 7th Asia Pacific Industrial Engineering and Management Systems Conference* (2006)

[Schaul et al. 2016] Schaul, Tom ; Quan, John ; Antonoglou, Ioannis ; Silver, David: Prioritized experience replay. In: *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016

[Schulman et al. 2015] Schulman, John ; Levine, Sergey ; Abbeel, Pieter ; Jordan, Michael ; Moritz, Philipp: Trust Region Policy Optimization. In: *Proceedings of the 32nd International Conference on Machine Learning*, PMLR, 2015, pp. 1889–1897

[Schulman et al. 2017] Schulman, John ; Wolski, Filip ; Dhariwal, Prafulla ; Radford, Alec ; Klimov, Oleg: Proximal Policy Optimization Algorithms. *arXiv:1707.06347* (2017)

[Silver et al. 2016] Silver, David ; Huang, Aja ; Maddison, Chris J. ; Guez, Arthur ; Sifre, Laurent ; Van Den Driessche, George ; Schrittwieser, Julian ; Antonoglou, Ioannis ; Panneershelvam, Veda ; Lanctot, Marc ; Dieleman, Sander ; Grewe, Dominik ; Nham, John ; Kalchbrenner, Nal ; Sutskever, Ilya ; Lillicrap, Timothy ; Leach, Madeleine ; Kavukcuoglu, Koray ; Graepel, Thore ; Hassabis, Demis: Mastering the game of Go with deep neural networks and tree search. In: *Nature,* 529 (2016), Nr. 7587, pp. 484–489

[Silver et al. 2017] Silver, David ; Schrittwieser, Julian ; Simonyan, Karen ; Antonoglou, Ioannis ; Huang, Aja ; Guez, Arthur ; Hubert, Thomas ; Baker, Lucas ; Lai, Matthew ; Bolton, Adrian ; Chen, Yutian ; Lillicrap, Timothy ; Hui, Fan ; Sifre, Laurent ; Van Den Driessche, George ; Graepel, Thore ; Hassabis, Demis: Mastering the game of Go without human knowledge. In: *Nature,* 550 (2017), Nr. 7676, pp. 354–359

[Sutton and Barto 2018] Sutton, Richard S. ; Barto, Andrew G.: *Reinforcement Learning: An Introduction.* Second Edition. The MIT Press, 2018

[Szita and Lorincz 2006] Szita, István ; Lorincz, András: Learning tetris using the noisy cross-entropy method. In: *Neural Computation,* 18 (2006), Nr. 12, pp. 2936–2941

[Tesauro 1995] Tesauro, Gerald: Temporal Difference Learning and TD-Gammon. In: *Commun. ACM,* 38 (1995), Nr. 3, pp. 58–68

[Van den Bergh et al. 2013] Van den Bergh, Jorne ; De Bruecker, Philippe ; Beliën, Jeroen ; Peeters, Jonas: Aircraft maintenance operations: state of the art. In: *HUBrussel; Brussel,* 9 (2013)

[Wang et al. 2015] Wang, Ke S. ; Li, Zhe ; Braaten, Jørgen ; Yu, Quan: Interpretation and compensation of backlash error data in machine centers for intelligent predictive maintenance using ANNs. In: *Advances in Manufacturing,* 3 (2015), Nr. 2, pp. 97–104

[Wang et al. 2016a] Wang, Xiao ; Wang, Hongwei ; Qi, Chao: Multi-agent reinforcement learning based maintenance policy for a resource constrained flow line system. In: *Journal of Intelligent Manufacturing,* 27 (2016), Nr. 2, pp. 325–333

[Wang et al. 2016b] Wang, Ziyu ; Schaul, Tom ; Hessel, Matteo ; Van Hasselt, Hado ; Lanctot, Marc ; De Freitas, Nando: Dueling Network Architectures for Deep Reinforcement Learning. In: *Proceedings of the 33rd International Conference on*

International Conference on Machine Learning - Volume 48, 2016 (ICML'16), pp. 1995–2003

[Witteman 2019] Witteman, M. M. D.: *A practical maintenance task packaging model applicable to aircraft maintenance*, Delft University of Technology, Master Thesis, 2019

[Yam et al. 2001] Yam, R C M. ; Tse, P W. ; Li, L ; Tu, P: Intelligent Predictive Decision Support System for Condition-Based Maintenance. In: *International Journal of Advanced Manufactoring Technology,* 17 (2001), pp. 383–391

[Zhang et al. 2018] Zhang, Rusheng ; Ishikawa, Akihiro ; Wang, Wenli ; Striner, Benjamin ; Tonguz, Ozan: Intelligent Traffic Signal Control: Using Reinforcement Learning with Partial Detection. *arXiv:1807.01628* (2018)

[Zhou et al. 2017] Zhou, Zhenpeng ; Li, Xiaocheng ; Zare, Richard N.: Optimizing Chemical Reactions with Deep Reinforcement Learning. In: *ACS Central Science,* 3 (2017), Nr. 12, pp. 1337–1344

Appendices

This page is intentionally left blank.

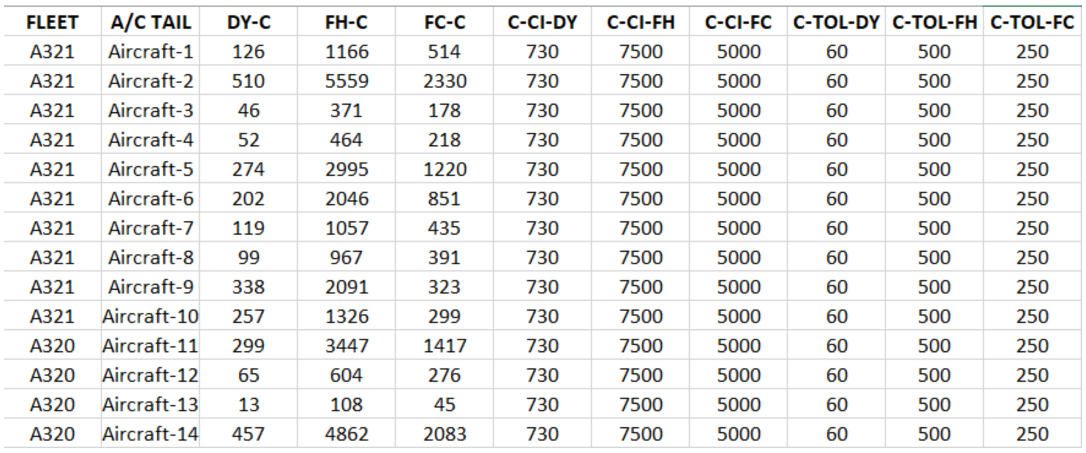
Appendix A

Maintenance Data

This chapter aims at detailing the real aircraft maintenance dataset by presenting samples, from the Excel files, that display the variables considered in the problem. The data files used in the check scheduling stage are presented in Section A.1, and the files used in the task packaging stage are presented in Section A.2.

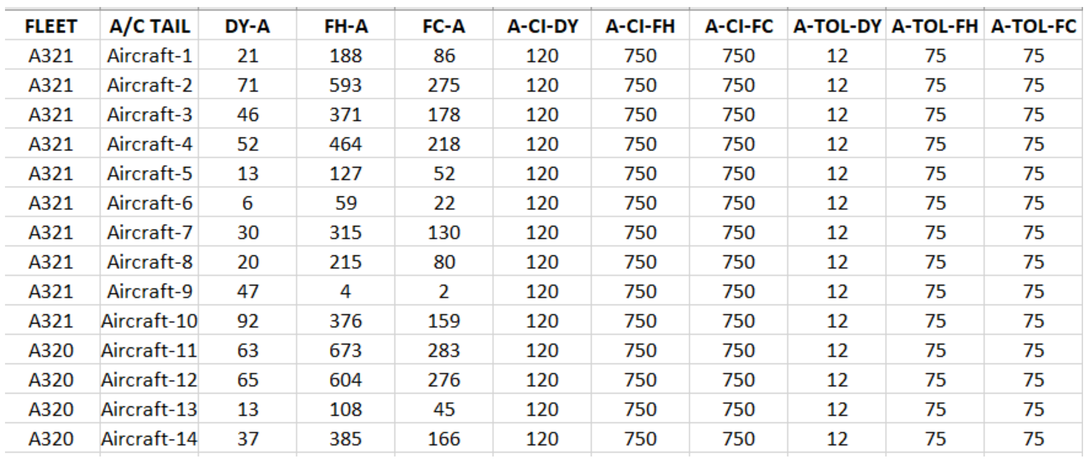
A.1 Check Scheduling Stage

The data used in the check scheduling stage includes general information about A and C-checks, estimations of aircraft utilization, and the periods corresponding to maintenance constraints. Figure A.1 presents information about the C-checks for a portion of the fleet. The column A/C TAIL represents the tail number of the airplane and serves as a unique identifier. The columns DY-C, FH-C, and FC-C represent the elapsed time since the previous C-check with respect to calendar days (DY), Flight Hours (FH), and Flight Cycles (FC), respectively. The columns C-CI-DY, C-CI-FH, and C-CI-FC are the check intervals while C-TOL-DY, C-TOL-FH, and C-TOL-FC are the tolerance allowed (again concerning the same three metrics). Figure A.2 presents the same information for the A-checks.



FLEET	A/C TAIL	DY-C	FH-C	FC-C	C-CI-DY	C-CI-FH	C-CI-FC	C-TOL-DY	C-TOL-FH	C-TOL-FC
A321	Aircraft-1	126	1166	514	730	7500	5000	60	500	250
A321	Aircraft-2	510	5559	2330	730	7500	5000	60	500	250
A321	Aircraft-3	46	371	178	730	7500	5000	60	500	250
A321	Aircraft-4	52	464	218	730	7500	5000	60	500	250
A321	Aircraft-5	274	2995	1220	730	7500	5000	60	500	250
A321	Aircraft-6	202	2046	851	730	7500	5000	60	500	250
A321	Aircraft-7	119	1057	435	730	7500	5000	60	500	250
A321	Aircraft-8	99	967	391	730	7500	5000	60	500	250
A321	Aircraft-9	338	2091	323	730	7500	5000	60	500	250
A321	Aircraft-10	257	1326	299	730	7500	5000	60	500	250
A320	Aircraft-11	299	3447	1417	730	7500	5000	60	500	250
A320	Aircraft-12	65	604	276	730	7500	5000	60	500	250
A320	Aircraft-13	13	108	45	730	7500	5000	60	500	250
A320	Aircraft-14	457	4862	2083	730	7500	5000	60	500	250

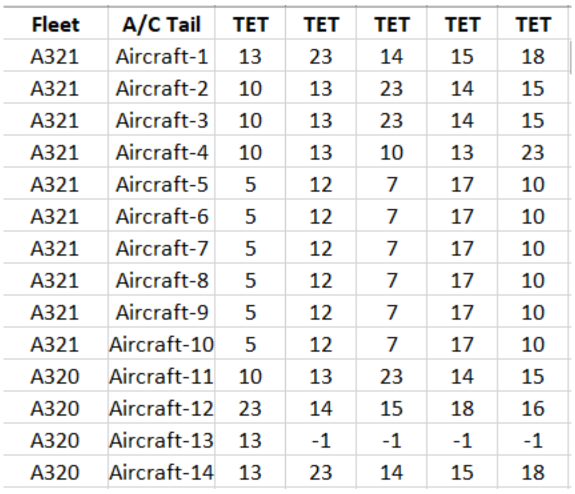
Figure A.1: Information about the C-checks.



FLEET	A/C TAIL	DY-A	FH-A	FC-A	A-CI-DY	A-CI-FH	A-CI-FC	A-TOL-DY	A-TOL-FH	A-TOL-FC
A321	Aircraft-1	21	188	86	120	750	750	12	75	75
A321	Aircraft-2	71	593	275	120	750	750	12	75	75
A321	Aircraft-3	46	371	178	120	750	750	12	75	75
A321	Aircraft-4	52	464	218	120	750	750	12	75	75
A321	Aircraft-5	13	127	52	120	750	750	12	75	75
A321	Aircraft-6	6	59	22	120	750	750	12	75	75
A321	Aircraft-7	30	315	130	120	750	750	12	75	75
A321	Aircraft-8	20	215	80	120	750	750	12	75	75
A321	Aircraft-9	47	4	2	120	750	750	12	75	75
A321	Aircraft-10	92	376	159	120	750	750	12	75	75
A320	Aircraft-11	63	673	283	120	750	750	12	75	75
A320	Aircraft-12	65	604	276	120	750	750	12	75	75
A320	Aircraft-13	13	108	45	120	750	750	12	75	75
A320	Aircraft-14	37	385	166	120	750	750	12	75	75

Figure A.2: Information about the A-checks.

The duration of 5 future C-checks is presented in Figure A.3, with 5 values of Total Estimated Time (TET). The value of -1 means that the aircraft is going to be phased-out, that is, it will stop flying. An important note is that the values correspond only to working days. On weekends and public holidays, the work is interrupted.

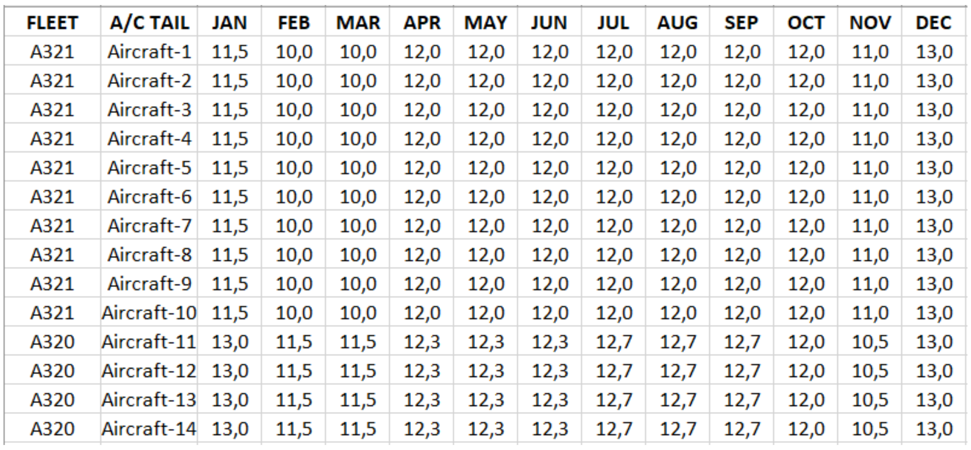


Fleet	A/C Tail	TET	TET	TET	TET	TET
A321	Aircraft-1	13	23	14	15	18
A321	Aircraft-2	10	13	23	14	15
A321	Aircraft-3	10	13	23	14	15
A321	Aircraft-4	10	13	10	13	23
A321	Aircraft-5	5	12	7	17	10
A321	Aircraft-6	5	12	7	17	10
A321	Aircraft-7	5	12	7	17	10
A321	Aircraft-8	5	12	7	17	10
A321	Aircraft-9	5	12	7	17	10
A321	Aircraft-10	5	12	7	17	10
A320	Aircraft-11	10	13	23	14	15
A320	Aircraft-12	23	14	15	18	16
A320	Aircraft-13	13	-1	-1	-1	-1
A320	Aircraft-14	13	23	14	15	18

Figure A.3: Estimated duration of 5 future C-checks.

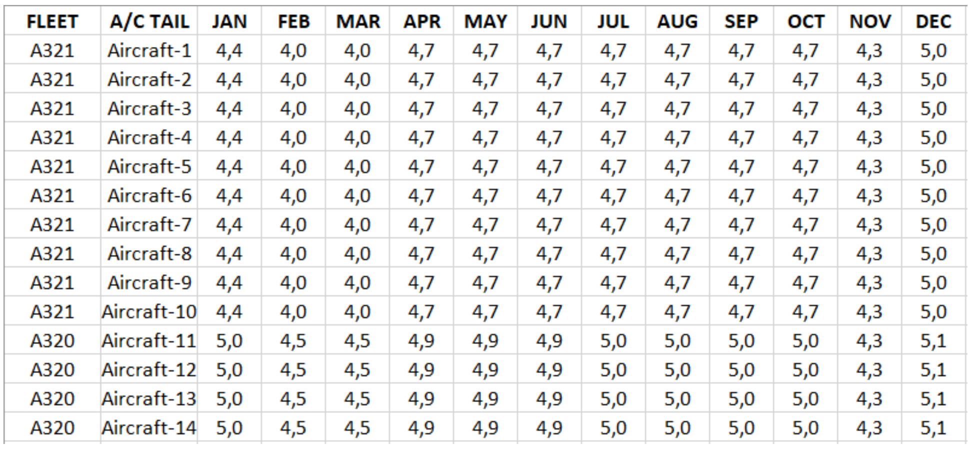
Figures A.4 and A.5 show the aircraft daily utilization in terms of FH and FC, respectively, for each month of the year.

Finally, Figure A.6 presents the periods when the C-checks are interrupted. These include 3 weeks during Christmas and New Year, 2 weeks during Easter, and the commercial peak seasons during the summer time.



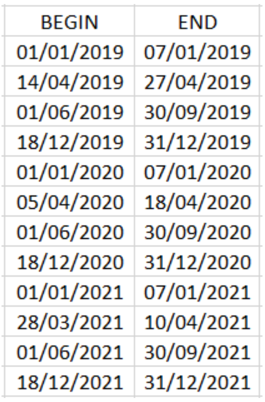
FLEET	A/C TAIL	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
A321	Aircraft-1	11,5	10,0	10,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	11,0	13,0
A321	Aircraft-2	11,5	10,0	10,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	11,0	13,0
A321	Aircraft-3	11,5	10,0	10,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	11,0	13,0
A321	Aircraft-4	11,5	10,0	10,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	11,0	13,0
A321	Aircraft-5	11,5	10,0	10,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	11,0	13,0
A321	Aircraft-6	11,5	10,0	10,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	11,0	13,0
A321	Aircraft-7	11,5	10,0	10,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	11,0	13,0
A321	Aircraft-8	11,5	10,0	10,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	11,0	13,0
A321	Aircraft-9	11,5	10,0	10,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	11,0	13,0
A321	Aircraft-10	11,5	10,0	10,0	12,0	12,0	12,0	12,0	12,0	12,0	12,0	11,0	13,0
A320	Aircraft-11	13,0	11,5	11,5	12,3	12,3	12,3	12,7	12,7	12,7	12,0	10,5	13,0
A320	Aircraft-12	13,0	11,5	11,5	12,3	12,3	12,3	12,7	12,7	12,7	12,0	10,5	13,0
A320	Aircraft-13	13,0	11,5	11,5	12,3	12,3	12,3	12,7	12,7	12,7	12,0	10,5	13,0
A320	Aircraft-14	13,0	11,5	11,5	12,3	12,3	12,3	12,7	12,7	12,7	12,0	10,5	13,0

Figure A.4: Aircraft daily utilization for each month (in FH).



FLEET	A/C TAIL	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
A321	Aircraft-1	4,4	4,0	4,0	4,7	4,7	4,7	4,7	4,7	4,7	4,7	4,3	5,0
A321	Aircraft-2	4,4	4,0	4,0	4,7	4,7	4,7	4,7	4,7	4,7	4,7	4,3	5,0
A321	Aircraft-3	4,4	4,0	4,0	4,7	4,7	4,7	4,7	4,7	4,7	4,7	4,3	5,0
A321	Aircraft-4	4,4	4,0	4,0	4,7	4,7	4,7	4,7	4,7	4,7	4,7	4,3	5,0
A321	Aircraft-5	4,4	4,0	4,0	4,7	4,7	4,7	4,7	4,7	4,7	4,7	4,3	5,0
A321	Aircraft-6	4,4	4,0	4,0	4,7	4,7	4,7	4,7	4,7	4,7	4,7	4,3	5,0
A321	Aircraft-7	4,4	4,0	4,0	4,7	4,7	4,7	4,7	4,7	4,7	4,7	4,3	5,0
A321	Aircraft-8	4,4	4,0	4,0	4,7	4,7	4,7	4,7	4,7	4,7	4,7	4,3	5,0
A321	Aircraft-9	4,4	4,0	4,0	4,7	4,7	4,7	4,7	4,7	4,7	4,7	4,3	5,0
A321	Aircraft-10	4,4	4,0	4,0	4,7	4,7	4,7	4,7	4,7	4,7	4,7	4,3	5,0
A320	Aircraft-11	5,0	4,5	4,5	4,9	4,9	4,9	5,0	5,0	5,0	5,0	4,3	5,1
A320	Aircraft-12	5,0	4,5	4,5	4,9	4,9	4,9	5,0	5,0	5,0	5,0	4,3	5,1
A320	Aircraft-13	5,0	4,5	4,5	4,9	4,9	4,9	5,0	5,0	5,0	5,0	4,3	5,1
A320	Aircraft-14	5,0	4,5	4,5	4,9	4,9	4,9	5,0	5,0	5,0	5,0	4,3	5,1

Figure A.5: Aircraft daily utilization for each month (in FC).

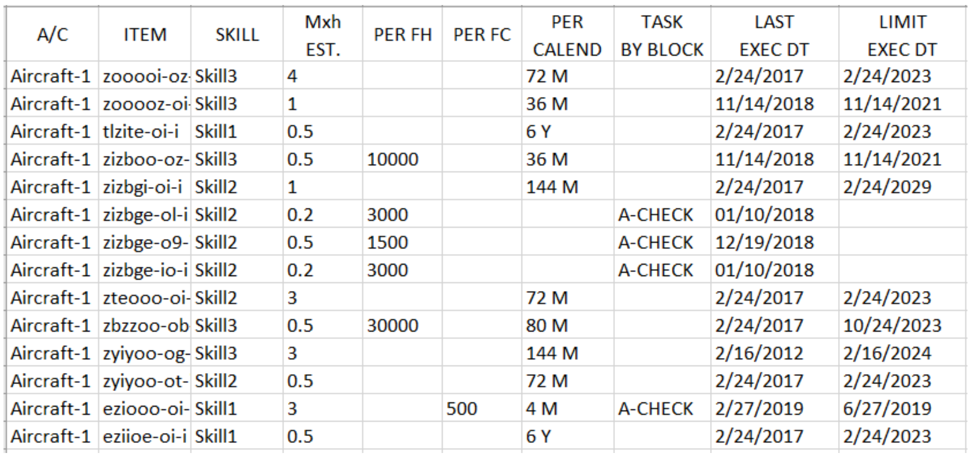


BEGIN	END
01/01/2019	07/01/2019
14/04/2019	27/04/2019
01/06/2019	30/09/2019
18/12/2019	31/12/2019
01/01/2020	07/01/2020
05/04/2020	18/04/2020
01/06/2020	30/09/2020
18/12/2020	31/12/2020
01/01/2021	07/01/2021
28/03/2021	10/04/2021
01/06/2021	30/09/2021
18/12/2021	31/12/2021

Figure A.6: Periods when the C-checks are interrupted.

A.2 Task Packaging Stage

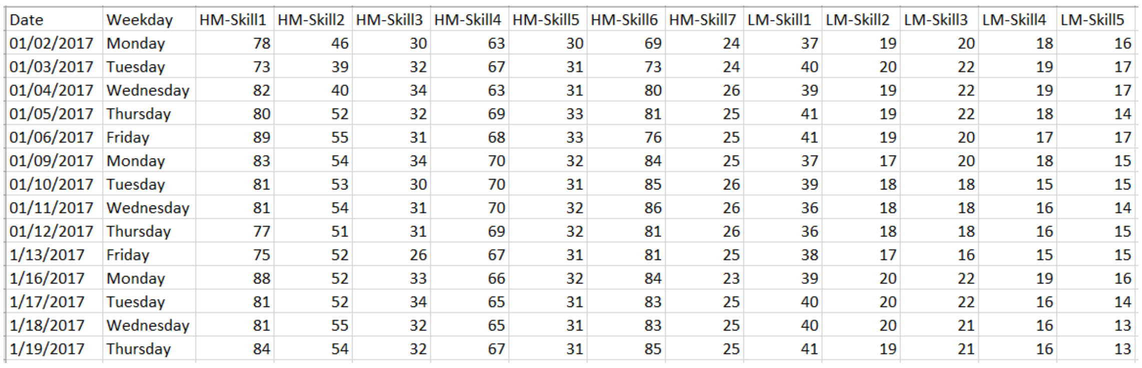
The data used in the task packaging stage includes general task information, manpower availability, unscheduled ratios, estimations of aircraft utilization, and the check calendar obtained in the check scheduling stage. Figure A.7 presents information about the maintenance tasks. Each line of the file corresponds to a single task. The column ITEM is the task code, SKILL is the primary skill required, and MxH EST. is the estimated time to execute the task (in hours). The columns PER FH, PER FC, and PER CALEND contain the interval of the tasks. TASK BY BLOCK indicates if the task needs to be performed in a specific check type. When this field is empty the task can be performed in any maintenance slot. The LAST EXEC DT column contains the task last execution date, and the LIMIT EXEC DT corresponds to the task due date. If the due date is empty, it can be computed using the last execution date and the task interval.



A/C	ITEM	SKILL	Mxh EST.	PER FH	PER FC	PER CALEND	TASK BY BLOCK	LAST EXEC DT	LIMIT EXEC DT
Aircraft-1	zooooi-oz	Skill3	4			72 M		2/24/2017	2/24/2023
Aircraft-1	zooooz-oi	Skill3	1			36 M		11/14/2018	11/14/2021
Aircraft-1	tlzite-oi-i	Skill1	0.5			6 Y		2/24/2017	2/24/2023
Aircraft-1	zizboo-oz-	Skill3	0.5	10000		36 M		11/14/2018	11/14/2021
Aircraft-1	zizbgi-oi-i	Skill2	1			144 M		2/24/2017	2/24/2029
Aircraft-1	zizbge-ol-i	Skill2	0.2	3000			A-CHECK	01/10/2018	
Aircraft-1	zizbge-o9-	Skill2	0.5	1500			A-CHECK	12/19/2018	
Aircraft-1	zizbge-io-i	Skill2	0.2	3000			A-CHECK	01/10/2018	
Aircraft-1	zteooo-oi-	Skill2	3			72 M		2/24/2017	2/24/2023
Aircraft-1	zbzzoo-ob	Skill3	0.5	30000		80 M		2/24/2017	10/24/2023
Aircraft-1	zyiyoo-og-	Skill3	3			144 M		2/16/2012	2/16/2024
Aircraft-1	zyiyoo-ot-	Skill2	0.5			72 M		2/24/2017	2/24/2023
Aircraft-1	eziooo-oi-	Skill1	3		500	4 M	A-CHECK	2/27/2019	6/27/2019
Aircraft-1	eziioe-oi-i	Skill1	0.5			6 Y		2/24/2017	2/24/2023

Figure A.7: Maintenance task information.

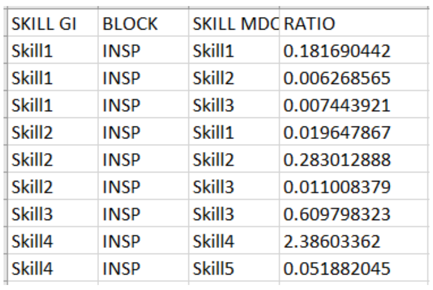
Figure A.8 presents historical data regarding the number of maintenance technicians available on each day of 2017 per skill type. The manpower with Heavy Maintenance (HM) skills can only be allocated to C-checks, while the manpower with Light Maintenance (LM) skills can be allocated to any other slot.



Date	Weekday	HM-Skill1	HM-Skill2	HM-Skill3	HM-Skill4	HM-Skill5	HM-Skill6	HM-Skill7	LM-Skill1	LM-Skill2	LM-Skill3	LM-Skill4	LM-Skill5
01/02/2017	Monday	78	46	30	63	30	69	24	37	19	20	18	16
01/03/2017	Tuesday	73	39	32	67	31	73	24	40	20	22	19	17
01/04/2017	Wednesday	82	40	34	63	31	80	26	39	19	22	19	17
01/05/2017	Thursday	80	52	32	69	33	81	25	41	19	22	18	14
01/06/2017	Friday	89	55	31	68	33	76	25	41	19	20	17	17
01/09/2017	Monday	83	54	34	70	32	84	25	37	17	20	18	15
01/10/2017	Tuesday	81	53	30	70	31	85	26	39	18	18	15	15
01/11/2017	Wednesday	81	54	31	70	32	86	26	36	18	18	16	14
01/12/2017	Thursday	77	51	31	69	32	81	26	36	18	18	16	15
1/13/2017	Friday	75	52	26	67	31	81	25	38	17	16	15	15
1/16/2017	Monday	88	52	33	66	32	84	23	39	20	22	19	16
1/17/2017	Tuesday	81	52	34	65	31	83	25	40	20	22	16	14
1/18/2017	Wednesday	81	55	32	65	31	83	25	40	20	21	16	13
1/19/2017	Thursday	84	54	32	67	31	85	25	41	19	21	16	13

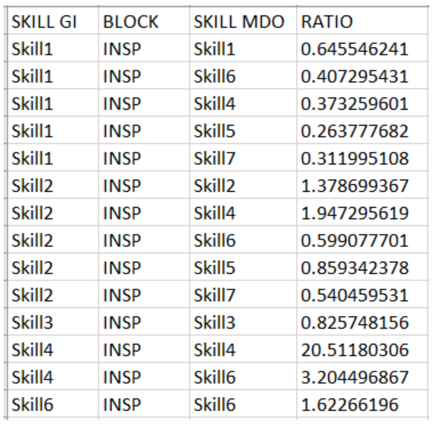
Figure A.8: Manpower availability per skill.

Figures A.9 and A.10 present the unscheduled maintenance ratios for A and C-checks, respectively. Each skill in the SKILL GI column can generate additional work from multiple skills in the SKILL MDO column. For example, if a 2-hour task with Skill1 as the required skill is packaged into an A-check, it will generate $2 \times 0.1816...$ hours of additional Skill1 work, and $2 \times 0.0062...$ hours of additional Skill2 work.



SKILL GI	BLOCK	SKILL MDO	RATIO
Skill1	INSP	Skill1	0.181690442
Skill1	INSP	Skill2	0.006268565
Skill1	INSP	Skill3	0.007443921
Skill2	INSP	Skill1	0.019647867
Skill2	INSP	Skill2	0.283012888
Skill2	INSP	Skill3	0.011008379
Skill3	INSP	Skill3	0.609798323
Skill4	INSP	Skill4	2.38603362
Skill4	INSP	Skill5	0.051882045

Figure A.9: A-check unscheduled ratios.



SKILL GI	BLOCK	SKILL MDO	RATIO
Skill1	INSP	Skill1	0.645546241
Skill1	INSP	Skill6	0.407295431
Skill1	INSP	Skill4	0.373259601
Skill1	INSP	Skill5	0.263777682
Skill1	INSP	Skill7	0.311995108
Skill2	INSP	Skill2	1.378699367
Skill2	INSP	Skill4	1.947295619
Skill2	INSP	Skill6	0.599077701
Skill2	INSP	Skill5	0.859342378
Skill2	INSP	Skill7	0.540459531
Skill3	INSP	Skill3	0.825748156
Skill4	INSP	Skill4	20.51180306
Skill4	INSP	Skill6	3.204496867
Skill6	INSP	Skill6	1.62266196

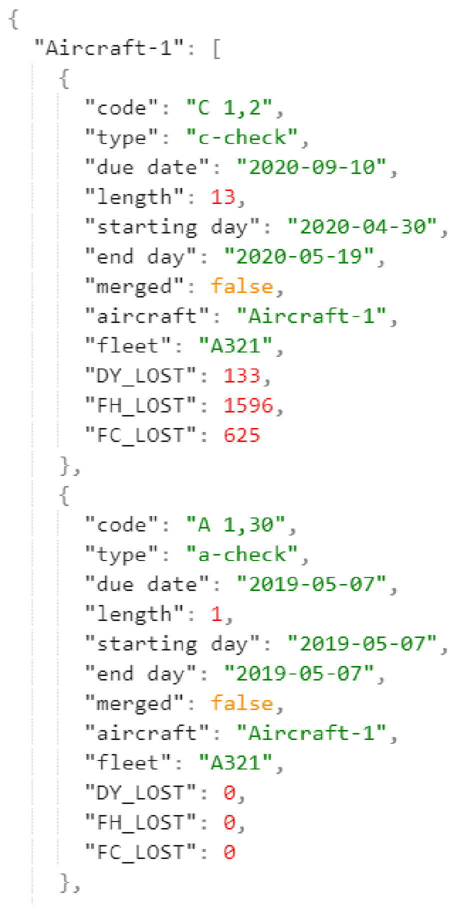
Figure A.10: C-check unscheduled ratios.

This page is intentionally left blank.

Appendix B

Algorithms Output Files

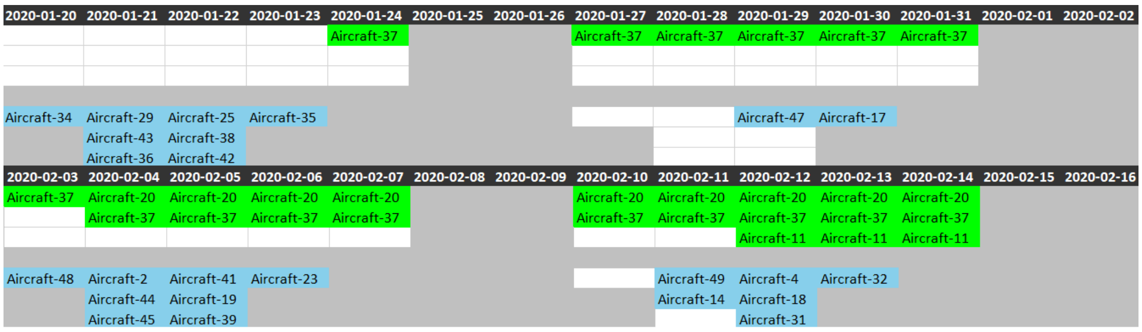
This chapter presents samples from the output files of the developed algorithms. The check scheduling algorithm generates a *JSON* file with relevant information about all scheduled checks, namely their code, type, due date, duration (in days), starting and ending day, a merged flag, aircraft information, and the amount of interval lost in terms of DY, FH, and FC.



```
{
  "Aircraft-1": [
    {
      "code": "C 1,2",
      "type": "c-check",
      "due date": "2020-09-10",
      "length": 13,
      "starting day": "2020-04-30",
      "end day": "2020-05-19",
      "merged": false,
      "aircraft": "Aircraft-1",
      "fleet": "A321",
      "DY_LOST": 133,
      "FH_LOST": 1596,
      "FC_LOST": 625
    },
    {
      "code": "A 1,30",
      "type": "a-check",
      "due date": "2019-05-07",
      "length": 1,
      "starting day": "2019-05-07",
      "end day": "2019-05-07",
      "merged": false,
      "aircraft": "Aircraft-1",
      "fleet": "A321",
      "DY_LOST": 0,
      "FH_LOST": 0,
      "FC_LOST": 0
    },
```

Figure B.1: Sample of the check scheduling *JSON* output file.

A second output of the check scheduling algorithm is an *Excel* file with an easier visualization of the check calendar. A sample of this file is illustrated in Figure B.2. The green cells correspond to C-checks, while the blue cells correspond to A-checks. The white cells correspond to the available slots that were not used, and the gray cells represent the constraints, such as the days of no maintenance work, and the days with limited slots available.

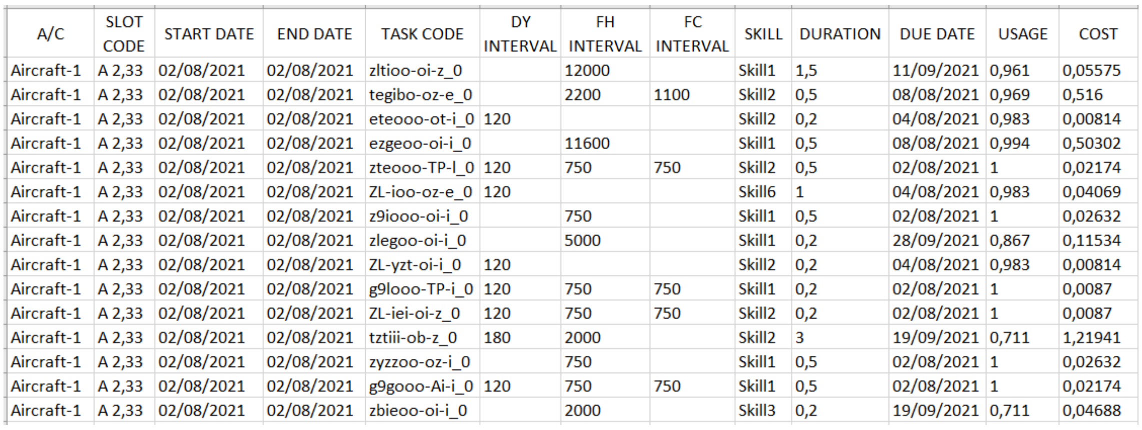


2020-01-20	2020-01-21	2020-01-22	2020-01-23	2020-01-24	2020-01-25	2020-01-26	2020-01-27	2020-01-28	2020-01-29	2020-01-30	2020-01-31	2020-02-01	2020-02-02
				Aircraft-37			Aircraft-37	Aircraft-37	Aircraft-37	Aircraft-37	Aircraft-37		
Aircraft-34	Aircraft-29	Aircraft-25	Aircraft-35						Aircraft-47	Aircraft-17			
	Aircraft-43	Aircraft-38											
	Aircraft-36	Aircraft-42											

2020-02-03	2020-02-04	2020-02-05	2020-02-06	2020-02-07	2020-02-08	2020-02-09	2020-02-10	2020-02-11	2020-02-12	2020-02-13	2020-02-14	2020-02-15	2020-02-16
Aircraft-37	Aircraft-20	Aircraft-20	Aircraft-20	Aircraft-20			Aircraft-20	Aircraft-20	Aircraft-20	Aircraft-20	Aircraft-20		
	Aircraft-37	Aircraft-37	Aircraft-37	Aircraft-37			Aircraft-37	Aircraft-37	Aircraft-37	Aircraft-37	Aircraft-37		
									Aircraft-11	Aircraft-11	Aircraft-11		
Aircraft-48	Aircraft-2	Aircraft-41	Aircraft-23					Aircraft-49	Aircraft-4	Aircraft-32			
	Aircraft-44	Aircraft-19						Aircraft-14	Aircraft-18				
	Aircraft-45	Aircraft-39							Aircraft-31				

Figure B.2: Sample of the check calendar output file.

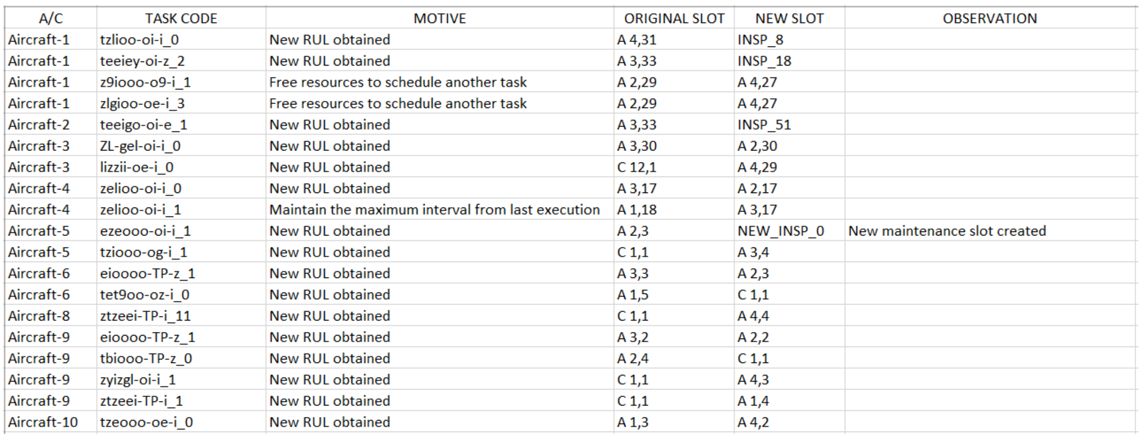
The algorithms developed in the task packaging stage produce a maintenance plan for the fleet consisting of an *Excel* file for every aircraft. Figure B.3 presents a sample of the maintenance plan for aircraft 1. Each line represents a maintenance task and contains information about the slot in which it was packaged, general task information, and the usage and cost obtained by packaging the task in that particular slot.



A/C	SLOT CODE	START DATE	END DATE	TASK CODE	DY INTERVAL	FH INTERVAL	FC INTERVAL	SKILL	DURATION	DUE DATE	USAGE	COST
Aircraft-1	A 2,33	02/08/2021	02/08/2021	zltioo-oi-z_0		12000		Skill1	1,5	11/09/2021	0,961	0,05575
Aircraft-1	A 2,33	02/08/2021	02/08/2021	tegibo-oz-e_0		2200	1100	Skill2	0,5	08/08/2021	0,969	0,516
Aircraft-1	A 2,33	02/08/2021	02/08/2021	eteooo-ot-i_0	120			Skill2	0,2	04/08/2021	0,983	0,00814
Aircraft-1	A 2,33	02/08/2021	02/08/2021	ezgeoo-oi-i_0		11600		Skill1	0,5	08/08/2021	0,994	0,50302
Aircraft-1	A 2,33	02/08/2021	02/08/2021	zteooo-TP-l_0	120	750	750	Skill2	0,5	02/08/2021	1	0,02174
Aircraft-1	A 2,33	02/08/2021	02/08/2021	ZL-ioo-oz-e_0	120			Skill6	1	04/08/2021	0,983	0,04069
Aircraft-1	A 2,33	02/08/2021	02/08/2021	z9iooo-oi-i_0		750		Skill1	0,5	02/08/2021	1	0,02632
Aircraft-1	A 2,33	02/08/2021	02/08/2021	zlegoo-oi-i_0		5000		Skill1	0,2	28/09/2021	0,867	0,11534
Aircraft-1	A 2,33	02/08/2021	02/08/2021	ZL-yzt-oi-i_0	120			Skill2	0,2	04/08/2021	0,983	0,00814
Aircraft-1	A 2,33	02/08/2021	02/08/2021	g9looo-TP-i_0	120	750	750	Skill1	0,2	02/08/2021	1	0,0087
Aircraft-1	A 2,33	02/08/2021	02/08/2021	ZL-iei-oi-z_0	120	750	750	Skill2	0,2	02/08/2021	1	0,0087
Aircraft-1	A 2,33	02/08/2021	02/08/2021	tztiii-ob-z_0	180	2000		Skill2	3	19/09/2021	0,711	1,21941
Aircraft-1	A 2,33	02/08/2021	02/08/2021	zyzzoo-oz-i_0		750		Skill1	0,5	02/08/2021	1	0,02632
Aircraft-1	A 2,33	02/08/2021	02/08/2021	g9gooo-Ai-i_0	120	750	750	Skill1	0,5	02/08/2021	1	0,02174
Aircraft-1	A 2,33	02/08/2021	02/08/2021	zbieoo-oi-i_0		2000		Skill3	0,2	19/09/2021	0,711	0,04688

Figure B.3: Sample of the maintenance plan for aircraft 1.

The predictive task scheduler also produces an *Excel* file with the history of all changes made to the original maintenance plan. Figure B.4 shows a sample of this file. Each line represents a modification made to the original plan, containing the code of the task that was rescheduled, the original slot and the new slot in which it was packaged, and the motive that led to the rescheduling. There is also an observation field to report some special events, such as the creation of a new maintenance slot.



A/C	TASK CODE	MOTIVE	ORIGINAL SLOT	NEW SLOT	OBSERVATION
Aircraft-1	tzlioo-oi-i_0	New RUL obtained	A 4,31	INSP_8	
Aircraft-1	teeiey-oi-z_2	New RUL obtained	A 3,33	INSP_18	
Aircraft-1	z9iooo-o9-i_1	Free resources to schedule another task	A 2,29	A 4,27	
Aircraft-1	zlgioo-oe-i_3	Free resources to schedule another task	A 2,29	A 4,27	
Aircraft-2	teeigo-oi-e_1	New RUL obtained	A 3,33	INSP_51	
Aircraft-3	ZL-gel-oi-i_0	New RUL obtained	A 3,30	A 2,30	
Aircraft-3	lizzii-oe-i_0	New RUL obtained	C 12,1	A 4,29	
Aircraft-4	zelioo-oi-i_0	New RUL obtained	A 3,17	A 2,17	
Aircraft-4	zelioo-oi-i_1	Maintain the maximum interval from last execution	A 1,18	A 3,17	
Aircraft-5	ezeooo-oi-i_1	New RUL obtained	A 2,3	NEW_INSP_0	New maintenance slot created
Aircraft-5	tziooo-og-i_1	New RUL obtained	C 1,1	A 3,4	
Aircraft-6	eioooo-TP-z_1	New RUL obtained	A 3,3	A 2,3	
Aircraft-6	tet9oo-oz-i_0	New RUL obtained	A 1,5	C 1,1	
Aircraft-8	ztzeei-TP-i_11	New RUL obtained	C 1,1	A 4,4	
Aircraft-9	eioooo-TP-z_1	New RUL obtained	A 3,2	A 2,2	
Aircraft-9	tbiooo-TP-z_0	New RUL obtained	A 2,4	C 1,1	
Aircraft-9	zyizgl-oi-i_1	New RUL obtained	C 1,1	A 4,3	
Aircraft-9	ztzeei-TP-i_1	New RUL obtained	C 1,1	A 1,4	
Aircraft-10	tzeooo-oe-i_0	New RUL obtained	A 1,3	A 4,2	

Figure B.4: History of changes made to the original maintenance plan.