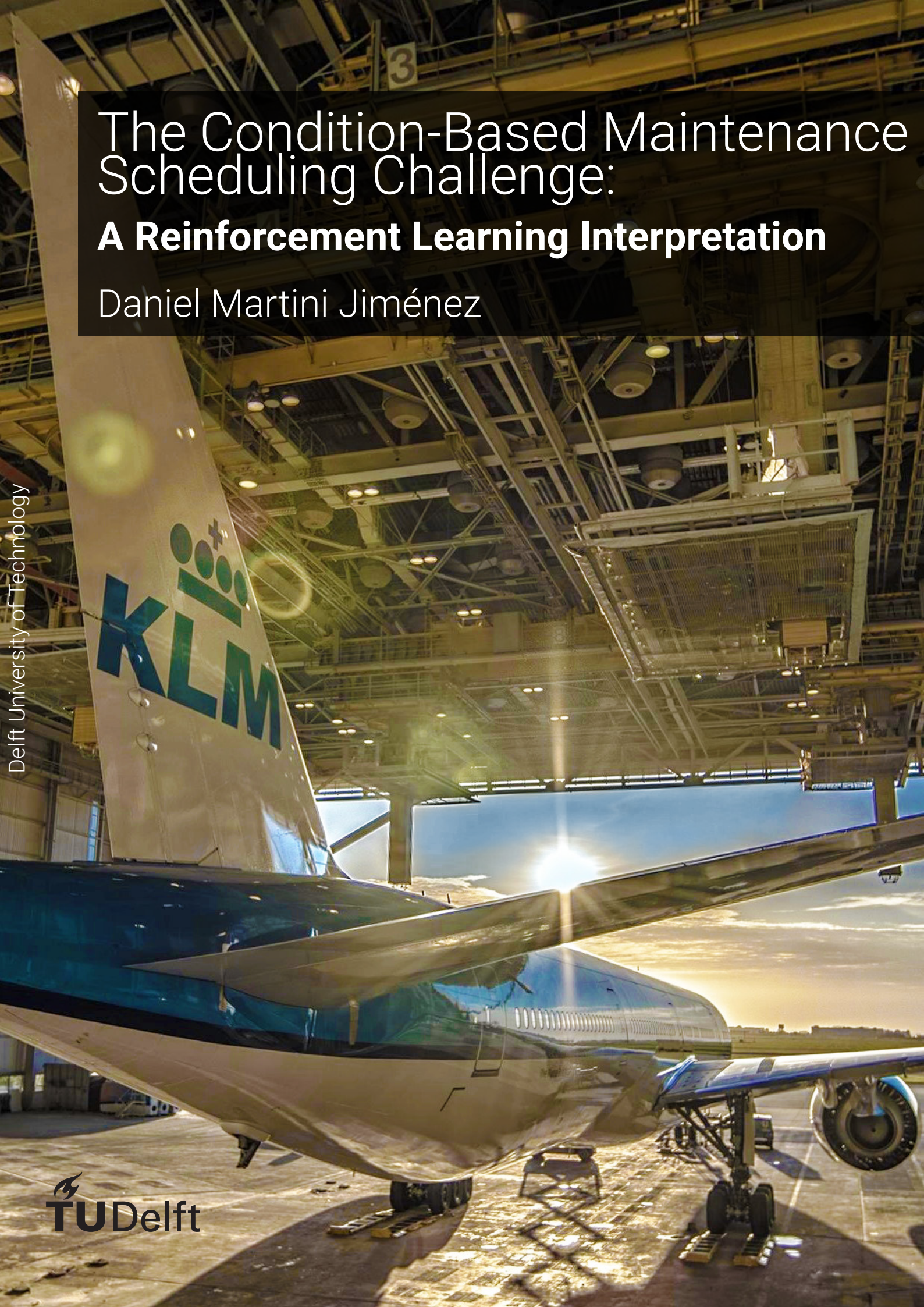
The Condition-Based Maintenance Scheduling Challenge:

A Reinforcement Learning Interpretation

Daniel Martini Jiménez



Delft University of Technology

TUDelft

This page was intentionally left blank

The Condition-Based Maintenance Scheduling Challenge:

A Reinforcement Learning Interpretation

by

Daniel Martini Jiménez

To obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday April 15, 2021

Student number:	4448650	
Thesis committee:	Member	Department
Chair	Dr. G. C. H. E. de Croon	TU Delft - Control & Simulation
Examiner	Dr. A. Bombelli	TU Delft - Air Transport & Operations
Supervisor TU Delft	Dr. B. F. Lopes dos Santos	TU Delft - Air Transport & Operations
Supervisor KLM	Ir. F. C. Freeman	KLM Royal Dutch Airlines

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

TU Delft

This page was intentionally left blank

Acknowledgements

This report is the final deliverable for my Master of Science thesis on condition-based maintenance scheduling. The work is the result of a fruitful collaboration with TU Delft and KLM Royal Dutch Airlines, made possible by the European Horizons 2020's ReMAP project. During the past year, I have worked on developing a maintenance scheduling framework that enables condition-based strategies through the power of reinforcement learning. The joint effort of KLM and TU Delft, allowed me to produce a realistic simulation of airline maintenance operations that are scheduled by an automated tool. The objective of the project is to introduce a new maintenance decision-support system that reflects the benefit of having a continuous monitoring of the aircraft systems thanks to the prognostics and health management (PHM) technology, i.e. sensors on board the aircraft. I am grateful for having been able to study such a challenging topic that is set to revolutionize the future of the airline business. The completion of this work would not have been possible without certain people, whom I would like to thank.

Firstly, I would like to thank my TU Delft supervisor Bruno for his continuous support, the constructive feedback and day to day coaching throughout these months. His dedication to research and teaching are exemplary, and his insatiable motivation and out of the box ideas have never stopped impressing me.

I am thankful to my KLM supervisor Floris, without whom I would not have been able to capture the reality of maintenance operations. I have vivid memories of the weekly conversations while sharing frustrations about the maintenance data and the stochastic simulation parameters. I thank him for the many detailed discussions on airline maintenance strategies and for shedding light on my analysis.

A special word of thanks goes to Erik-Jan van Kampen, Qichen Deng and Iordanis Tseremoglou for listening to my vision of reinforcement learning in the operations research domain, as well as validating my methodology every time I was in doubt. Their input has been very valuable for the construction of my model.

Last but not least, I am grateful to my friends and family for the continued trust they placed in me. On a special note, I would like to thank my parents and my two sisters for their unconditional support. Their encouragement and belief in my capabilities during these years have been essential for my engagement with my academic career, and their lessons have allowed me to develop personally and professionally. There are no words that could express my gratitude for everything they have sacrificed to make me become the person I am today. To them, I dedicate this thesis.

Daniel Martini Jiménez
Delft, March 2021

Contents

List of Figures

List of Tables

List of Abbreviations

A2C	Advantage Actor-Critic
A3C	Asynchronous Advantage Actor-Critic
ADP	Approximate Dynamic Programming
AMP	Aircraft Maintenance Program
AMR	Aircraft Maintenance Routing
AMSP	Airline Maintenance Scheduling Problem
AOG	Aircraft on Ground
ARP	Aircraft Rotation Problem
CBM	Condition-based Maintenance
CFA	Cost Function Approximation
CO	Combinatorial Optimization
DDPG	Deep Deterministic Policy Gradient
DET	Detailed Inspection
DIS	Discard
DP	Dynamic Programming
DQN	Deep Q-Network
DY	Calendar Days
FC	Flight Cycles
FH	Flight Hours
FNC	Functional Check
GVI	General Visual Inspection
HMV	Heavy Maintenance Visit
IATA	International Air Transport Association
LOF	Line of Flight
LUB	Lubrication
MC	Monte Carlo
MDP	Markov Decision Process
MEL	Minimum Equipment List
MILP	Mixed Integer Linear Programming

ML	Machine Learning
MO	Maintenance Opportunity
MPD	Maintenance Planning Document
MRB	Maintenance Review Board
MRO	Maintenance Repair and Overhaul
MSG	Maintenance Steering Group
NCO	Neural Combinatorial Optimization
NN	Neural Network
OPC	Operational Check
PER	Prioritized Experience Replay
PFA	Policy Function Approximation
PHM	Prognostic Health and Management
RH	Rolling Horizon
RL	Reinforcement Learning
RST	Restoration
RUL	Remaining Useful Life
SGD	Stochastic Gradient Descent
SVC	Servicing
TA	Tail Assignment
TAT	Turn-around-time
TD	Temporal Difference
VC	Visual Check
VFA	Value Function Approximation

.

I

Scientific Paper

The Condition-Based Maintenance Scheduling Challenge: A Reinforcement Learning Interpretation

Daniel Martini Jiménez*

Delft University of Technology, Delft, The Netherlands

Abstract

Condition-based maintenance (CBM) is emerging in the airline industry as a revolutionary concept that could potentially increase the efficiency of aircraft operations. The adoption of CBM strategies is not yet widespread due to the stringent requirements imposed by aviation authorities and the fact that the prognostic and health management (PHM) technology is still in its infancy. Along with the developing technology, CBM requires a performance evaluation in terms of aircraft maintenance schedule optimization. This paper proposes a novel reinforcement learning model to solve the airline maintenance scheduling problem subject to prognostics uncertainty. A Deep Q-Learning model is trained to schedule A-check routine tasks. The approach preserves the dichotomy of the maintenance problem by scheduling both interval-based clusters of tasks and condition-based tasks monitored with synthetic prognostics. Maintenance, repair, and overhaul (MRO) data from a major European airline was used to construct a realistic simulation and a model tailored to their operations. We explore a wide range of scenarios with varied numbers of tasks scheduled with a CBM policy, as well as different magnitudes of uncertainty in order to enable a viable maintenance strategy. Compared to traditional maintenance policies, the results demonstrate that the implementation of CBM reduces the fleet ground time and improves the task's interval utilization when assessing the uncertainty involved in prognostics.

Keywords: Condition-based maintenance, Airline maintenance operations, Operations research, Stochastic optimization, Reinforcement learning

---

1 Introduction

Over the years, aviation safety and operational efficiency have become key factors in remaining competitive in the airline industry. Maintenance is performed regularly in accordance with strict requirements imposed by international airworthiness authorities and aircraft manufacturers. While preventive maintenance is largely responsible for the outstanding safety and reliability of today's aviation, its effectiveness is limited by statistical generalizations and the lack of consideration for specific operating conditions (Daily and Peterson, 2017). As a result, some systems are replaced long before their potential due date, and others may fail prior to their assigned maintenance slot. Total maintenance

*Author of this article, supervised by Dr. ir. B. F. Lopes dos Santos and ir. F. C. Freeman

expenditure corresponds to approximately 10% of airlines' direct operating cost (Lagos et al., 2020). In 2018, the global maintenance, repair and overhaul (MRO) costs were valued at 69 billion USD, and, based on estimates by IATA's Maintenance Cost Technical Group (2019), a 4.1% increase per annum is expected, reaching an estimated market size of 103 billion USD by the year 2028. It is evident that stakeholders cannot afford to ignore the potential benefit of being at the cutting edge of technology with predictive maintenance.

Condition-based maintenance (CBM) is a predictive strategy designed to transform real-time aircraft data into actionable intelligence. Its use helps avoid unnecessary ground times and can replace periodic-based maintenance checks with a continuous monitoring of the aircraft thanks to prognostic and health management (PHM) technology. Although some authors define CBM as a preliminary step before predictive maintenance, this work includes the notion of prognostics and the prediction of a future due date as part of the CBM concept. The implementation of CBM strategies in the airline industry consists of two key elements: (1) the use of advanced analytics to understand the driving factors of performance, monitor aircraft systems, and predict remaining useful life (RUL), and (2) the fielded evaluation of PHM technology as it matures (Saxena et al., 2008). While the RUL models are still emerging and being researched, little effort has been made to assess CBM strategies in terms of aircraft maintenance schedule optimization (Sprong et al., 2020).

Background

The majority of researchers that include maintenance in the scope of operations optimization have focused on the aircraft routing problem (ARP) and the tail assignment (TA). The former is concerned with the creation of lines of flight, while the latter assigns these *routings* to individual aircraft. Frequently, these problems are solved in combination with maintenance considerations, in order to respect the maintenance schedule that requires the aircraft to be present at the hangar location. Since the early efforts of Feo and Bard (1989) a vast array of literature has formulated the ARP with a time-space network. Several other studies have focused on operational aircraft maintenance such as Afsar et al. (2006), Eltoukhy et al. (2017) and Lagos et al. (2020), yet none of them deals with the creation of a maintenance schedule but rather assumes that available planning exists to create the aircraft routings. It is noteworthy that these problems are solved with an operational horizon that normally has a time-window of 3-4 days. Instead, the planning horizon that is investigated in this research extends it to 8-12 weeks for A-checks. Lastly, Deng et al. (2020) is the only work that considers the check scheduling problem for long-term strategic decisions. In their work, a look-ahead dynamic optimization model is developed to solve the aircraft maintenance check scheduling problem in a 4-year time window. In order to reduce the size of the problem, the aircraft are scheduled by maintenance urgency based on the earliest due date. One of the major concerns that has arisen from this work is the computational time of dynamic programming (DP) methods in a stochastic environment.

The problem dimensionality can be circumvented with reinforcement learning (RL) or approximate dynamic programming (ADP). These frameworks arise as elegant theories to reach near-optimal solutions that make the problem computationally tractable. RL has seen successful applications in the fields of control theory and robotics (Sutton and Barto, 1998; Kober et al., 2013; Silver et al., 2017), while ADP has emerged more recently with similar algorithmic strategies to solve problems in the operation research domain. Powell (2011) addresses a wide range of ADP policies in the context of

dynamic resource management. Nonetheless, the applications of RL or ADP to scheduling problems are not common in the literature, and they are mostly found in the cloud computing domain (Solozabal et al., 2020; Tong et al., 2020; Liang et al., 2020).

The lack of automated planning tools for maintenance scheduling arises from the fact that it is a long-term problem in a dynamic environment, where factors such as aircraft utilization and task demand are continuously changing. Moreover, the literature mainly focuses on the operational horizon by solving the aircraft routing problem with maintenance constraints. Therefore, there is no flexibility in the maintenance planning and a pre-existent schedule is necessary in most cases. Two main challenges are yet to be addressed with regard to the airline maintenance scheduling problem (AMSP): (1) the model's exponential growth in size as the planning window increases, and (2) the omission of stochastic factors that result in the optimization of a static environment with a deterministic approach. Both challenges indicate that standard optimization techniques would not be able to cope with the size of the problem and produce useful results for a stochastic environment. In this work, Deep Q-learning, a well-known RL algorithm, is proposed as a suitable algorithmic strategy based on the discrete nature of the problem, and the successful applications across different industries (Mnih et al., 2015; Waschneck et al., 2018; Sun and Tan, 2019; Tong et al., 2020).

Paper Contribution

This study focuses on the assessment of CBM policies in the AMSP by contributing to the deployment of a scheduling model in a transitional period from preventive to predictive maintenance. We simulate synthetic prognostics for a limited number of aircraft systems using real data from a preventive maintenance case. Furthermore, we leverage a reinforcement learning framework to address the challenges that such a dynamic environment poses. Four main contributions to airline maintenance research arise from this work: (1) the AMSP has been extended with a CBM policy, and formulated as a sequential Markov decision process (MDP) that integrates a stochastic sampling strategy to simulate RUL prognostics; (2) a deep reinforcement learning algorithm is trained to learn a dynamic scheduling policy; (3) diverse transition strategies based on task clustering and varied frequency of maintenance opportunities have been tested to explore strategic decisions that airlines should take to accommodate CBM policies; (4) lastly, a wide range of scenarios is proposed to compare the scheduling performance to the uncertainty levels of PHM technology. To the author's knowledge, this is the first attempt at using a reinforcement learning methodology to create a data-driven solution for the AMSP subject to prognostic uncertainty.

Report Structure

The remainder of this paper is organized as follows. Section 2 gives an overview of the airline maintenance planning problem. In Section 3, the Markov decision process (MDP) formulation of the scheduling problem is presented. Section 4 explains the methodology developed to simulate the problem, from the clustering of individual tasks and the prognostic generation, to the scheduling algorithm. Section 5 deals with the experimental set-up required to analyze the maintenance scheduling framework and the methodology proposed. The results are described in Section 6, whilst Section 7 highlights their relevance and implications. Finally, the conclusion and recommendations are outlined in Section 8.

2 Problem Definition

The airline maintenance planning process involves two different paradigms. The first one, also known as *task-packaging*, consists of grouping routine tasks based on different features. The second one is the airline maintenance scheduling problem (AMSP) that consists of assigning each block of tasks to a maintenance opportunity. We extend this problem by introducing a CBM strategy in a portion of the maintenance program. Since most of the literature has focused on a limited number of systems monitored by prognostics and RUL predictive models will not become widely available for all aircraft tasks, this work assumes an early adoption period where the CBM concept is applied for a limited number of tasks. The remaining aircraft tasks are scheduled with a traditional letter-check structure. Therefore, we address two categories of events within the aircraft maintenance program: routine and CBM tasks. The first ones are clustered during the task-packaging phase in maintenance checks, and they must be performed periodically based on a usage interval. On the other hand, CBM tasks are monitored individually during the scheduling process by means of prognostics. Thus, we define a *transition strategy* that combines traditional letter checks, predefined in the *task-packaging* problem, with a task-based methodology for CBM tasks. The scheduling model allocates A-checks to maintenance opportunities, and inserts CBM tasks into the scheduled A-checks based on the RUL prognostics. In the remainder of this section, an overview of the maintenance planning process and the challenges faced by the airline industry to implement CBM strategies is described.

2.1 Maintenance Planning

There are three main categories of airline maintenance checks based on their periodicity and duration (Yan et al., 2011). The first, *long-term*, includes heavy maintenance tasks that require an overhaul lasting longer than 10 days. They are also called level C and D checks. The *mid-term* maintenance plans are set monthly on the long-term plan. They incorporate both level A and B checks. Lastly, *short-term* plans produce schedule adjustments due to incidents. Line maintenance is an additional short-term maintenance activity performed at the gate of the airport. The only method approved by airworthiness authorities is the Maintenance Steering Group-3 (MSG-3), which develops a set of tasks for specific functional failures (Shannon and Ackert, 2010). The airplane manufacturer also releases the maintenance planning document (MPD) that contains a list of the mandatory requirements to develop a customized maintenance program (Kinnison, 2004).

Traditional maintenance policies opt to cluster a series of routine tasks, based on the resources required, in large blocks, also called letter-checks (A,B,C,D checks). They use a combination of time-driven intervals based on calendar days (DY), and usage-driven intervals based on the cumulative aircraft flight hours (FH) and flight cycles (FC). This methodology has declined in recent years in favor of progressive maintenance checking. Modern maintenance planning approaches adopt a task-driven solution where tasks are bundled together in smaller clusters such that the resulting check can be performed overnight while the aircraft is not in use (Muchiri and Smit, 2009). This is done to reduce aircraft downtime, to have a flexible grouping of tasks, and to optimize the schedule. On the other hand, the planning complexity is highly increased, and the reduced slot capacity may represent a risk when considering safety buffer times to allocate ad hoc maintenance tasks (Shannon and Ackert, 2010).

Maintenance tasks can be classified in three categories based on their scheduling nature: *routine, non-routine, and unscheduled* (Kinnison, 2004). Routine tasks occur periodically at intervals defined by the authorities. They are clustered in the letter-checks based on a hierarchy defined in the MPD. Generally, a large number of routine tasks are dephased from the intervals of these checks, in order to perform the tasks before due time, which leads to inefficiencies (Witteman et al., 2021). Furthermore, non-routine tasks are those that vary from check to check and are usually triggered by a routine task such as an inspection or a functional check. Lastly, unscheduled tasks originate from unexpected failures or events, such as faults reported by pilots, bird strikes and reactive maintenance tasks.

In this work, we focus exclusively on the *A-check routine* maintenance program for the following reasons. Firstly, C-checks are excluded as they are scheduled much farther in advanced than A-checks. Since C-checks are generally long overhauls, it is preferred to schedule them during low-demand periods, and in sequence for the whole fleet. Moreover, this allows leveraging the learning curve effect of the engineers involved in the maintenance checks. Therefore, there is little room for improvement in terms of aircraft maintenance schedule optimization. Secondly, the non-routine maintenance program and the unscheduled maintenance events have been excluded because they have an operational planning horizon. Therefore, they make use of additional opportunities such as line maintenance or last-minute slots. In this way, the scope of the problem is reduced to the tactical horizon, and the focus is on the benefits of a CBM transition strategy for the *A-check routine* maintenance program.

2.2 Condition-Based Maintenance

The transition to condition-based maintenance (CBM) combines preventive and predictive strategies that aim to reduce maintenance costs while providing services with improved quality and reliability (Hölzel et al., 2012). *Preventive maintenance* strategies will remain necessary for an effective transition, while the industry accommodates a CBM concept. Therefore, the transition maintenance strategy that we propose includes both interval-based checks and individual CBM tasks. The interval-based checks adhere to a preventive methodology, while the CBM tasks monitored with RUL prognostics belong to the predictive maintenance concept.

Through continuous aircraft monitoring, it becomes possible to perform repairs before damage occurs, and maintenance can be scheduled only when necessary. The prognostic and health management (PHM) technologies, i.e. sensors on board the aircraft, are employed to determine the optimal time to allocate the tasks and avoid any waste of RUL. The integration of PHM techniques with maintenance decisions represents the basis of the CBM concept (Vianna and Yoneyama, 2018). It requires the aircraft to be completely wired and equipped with sensors to monitor the real-time health of the fleet. To date, various airline operators have implemented certain aspects of CBM in their operations that have dramatically increased the level of electronic integration in aircraft systems (Teal and Sorensen, 2001). However, none have taken full advantage of the CBM concept because of the lack of fully equipped aircraft with health monitoring sensors, strict air safety requirements, and the lack of digitalization in the maintenance operations domain. In order to foster the implementation of CBM strategies, some hybrid approaches have been proposed in the research community where traditional scheduled maintenance is used for safety-critical tasks, and CBM is used for non-critical systems (Dong et al., 2019; ReMAP H2020). We also define a hybrid approach using real data from a preventive maintenance case. Since

the focus of this research is exclusively on routine maintenance tasks, we include a limited number of these tasks in the scope of CBM, and we simulate synthetic prognostics to monitor their due date. The objective is to create a scheduling framework that allows the allocation of aircraft maintenance checks, as well as the insertion of CBM tasks in the same maintenance slot.

CBM strategies can be formalized with two types of actions: interval task (de-)escalation and task substitution (Hölzel et al., 2014). The former extends or reduces the original interval of the task based on condition data from sensors. Since interval limitations imposed by authorities are very conservative, it is likely that CBM escalation will safely lengthen the service life of monitored systems (Vandawaker et al., 2015). Similarly, task substitution replaces an inspection or a functional check with a sensor that defers a *non-routine* task until a threshold is triggered. In both cases, the CBM tasks monitored by PHM technology no longer form part of a routine block. However, task substitution results in the removal of inspections and functional checks from the maintenance program, while in the case of (de-)escalation the tasks are kept in the maintenance program but performed only when the prognostic alerts a failure. Since the proposed approach only considers routine maintenance tasks, non-routine tasks triggered by substitution are not included in the scope of the problem. Based on the cost-benefit analysis presented by Hölzel et al. (2014), an improvement in terms of labor hours is expected due to a reduction in unnecessary preventive maintenance and improved system utilization. Moreover, we extend their single-aircraft analysis to a fleet of 16 long-haul aircraft in order to evaluate the planning and scheduling performance where several aircraft compete for limited maintenance resources.

3 Problem Formulation

The mathematical formulation of the airline maintenance scheduling problem (AMSP) depends on the choice of optimization model, the planning horizon, and the uncertainty of the factors involved. The proposed formulation is inspired by the works of Deng et al. (2020) and Lagos et al. (2020) that employ a dynamic programming approach and a look-ahead estimation of the next states with the objective of maximizing aircraft utilization. We also leverage the approximate dynamic programming (ADP) framework presented in the work of Powell and Topaloglu (2006). Their formulation recognizes two key elements in any dynamic resource allocation problem: the resources and the demand. In the AMSP, resources are identified as the maintenance opportunities predefined in the slot availability calendar, while the demand corresponds to the *scheduling units*, also called routine blocks and CBM tasks, that need to be allocated. Both types of scheduling units are modeled in the exact same way. However, routine blocks have a due date dictated by an interval, while the due dates of CBM tasks are dependent on RUL prognostics. The formulation makes use of the following five main assumptions.

1. **Aircraft utilization is known and constant.** Aircraft routings are usually only known three to five days before operations. Therefore, to produce monthly A-check schedules, a statistical analysis is performed to calculate the average flight hours and cycles per day.
2. **The amount of weekly slots is constant and there are sufficient labor hours**. We assume that the size of the A-checks will be similar and the CBM tasks can be accommodated within the scheduled maintenance slot.
3. **Inventory management is not considered.** Tools and parts are assumed to be always

available to perform scheduled maintenance tasks.

4. **CBM tasks can only be allocated within a routine block**. In practice, an individual task could also be performed in line maintenance or additional slots besides A-checks.
5. **Additional tasks can be performed with a buffer-time**. Eventual non-routine tasks and unscheduled maintenance events can be accommodated within the A-checks.

The goal of the problem is to select a maintenance slot date for every aircraft in order to maximize fleet utilization. The problem is set up in a sequential fashion such that at every every timestep t, the scheduling unit with highest priority (AC_t) is selected based on the earliest due date. The problem is deemed solved when the model schedules the last routine block or CBM task on the time horizon date (TH). Every time an aircraft is selected either a routine block (A-check) or a CBM task can drive the earliest due date selection. In the first case, a routine block needs to be allocated to a maintenance slot, while in the second case, a CBM task is inserted in one of the A-checks previously scheduled. The problem formulation uses the notation presented below. The choice of reinforcement learning (RL) or approximate dynamic programming (ADP) requires formulating the problem as a sequential decision making process, most commonly as a Markov Decision Process (MDP). An MDP assumes the presence of the state-space, the action-space, the reward function and the transition function. In the remainder of this section a clear overview of each element is provided.

Sets

$\mathcal{K}$	:	sets of aircraft scheduling units (A-checks and CBM tasks) for the complete fleet
$\mathcal{A}$	:	sets of discrete scheduling actions
$\mathcal{M}$	:	sets of maintenance opportunities

Parameters

TH	:	Time horizon of the maintenance scheduling simulation
I^k_{FH}	:	Flight hours interval of scheduling unit k
I^k_{FC}	:	Flight cycles interval of scheduling unit k
I^k_{DY}	:	Calendar based interval of scheduling unit k
δ^t_{fh}	:	Daily flown hours at timestep t
δ^t_{fc}	:	Daily flown cycles at timestep t

Variables

AC_t	:	Aircraft scheduling unit with highest priority selected at time t
SD^k_t	:	Scheduled date selected by a_t for aircraft k
DD^k_t	:	Due date of scheduling unit k at timestep t
PE^k_t	:	Previous maintenance execution date of scheduling unit k at timestep t
M^k_a	:	Amount of available slots on sub-interval a for scheduling unit k
MD^k_a	:	Date of latest maintenance slot on sub-interval a for scheduling unit k
RD^k_t	:	Reference date to update all environment parameters of scheduling unit k
ΔFH^k_t	:	Simulated flight hours at timestep t for scheduling unit k
ΔFC^k_t	:	Simulated flight cycles at timestep t for scheduling unit k
ΔDY^k_t	:	Simulated calendar days at timestep t for scheduling unit k
FH^k_t	:	Remaining flight hours at timestep t for scheduling unit k
FC^k_t	:	Remaining flight cycles at timestep t for scheduling unit k
DY^k_t	:	Remaining calendar days at timestep t for scheduling unit k

X_t^k	:	Remaining useful life in calendar days at timestep t for scheduling unit k
I_t^k	:	Minimum interval in calendar days at timestep t for scheduling unit k
ω_t^k	:	Predicted due date at timestep t for scheduling unit k

Decision variable

a_t	:	Interval utilization of scheduling unit AC_t at timestep t

3.1 State Space

The maintenance planning simulation occurs in a sequential process by selecting the most critical *scheduling unit*, either a block or a CBM task, depending on which has the earliest upcoming due date. Each item or *scheduling unit* has a time window that goes from the previous execution to the next due date, dictated by either an interval or a prognostic. This time window is discretized in $|\mathcal{A}|$ sub-intervals, as shown in Figure 1, to analyze the available resources in each of them, as well as the demand for those resources. The state vector takes as a reference the time window of the most critical item to calculate the relevant environment features. A total of six features is employed to capture the scheduling environment condition through which the model can learn the effect of its decisions:

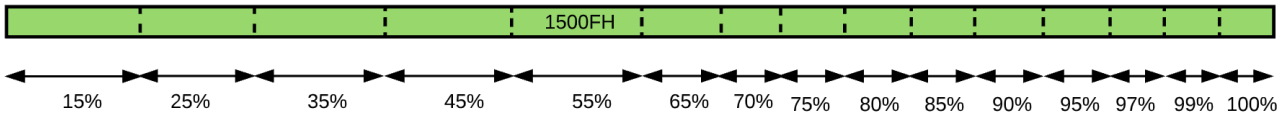


Figure 1: Scheduling unit discretization with an example time window of 1500 flight hours

- $D_{t,a}$: demand or number of scheduling units competing for a slot in the sub-interval "a". If there is no slot available, the value of this feature becomes null.
- $R_{t,a}$: resources or number of the available slots to the demand $D_{t,a}$ competing for the slot in sub-interval "a" of the most critical unit AC_t at time t.
- $lh_{t,a}$ is the average estimated reward for the competing aircraft computed with a look-ahead function, assuming that the slot in sub-interval "a" is occupied by AC_t. The objective of this feature is to capture the performance consequences for the rest of the fleet when a slot is being considered for the most critical scheduling unit. The function works with a greedy policy: every time a slot in sub-interval "a" is considered, it is removed from the available resources. The function assigns the latest possible slot to each aircraft in their priority order, and calculates the cost of allocating the remaining resources to the competing aircraft.
- $h_{t,a}$ indicates the time index of the slot in the simulation horizon, such that the system is aware of the slot position in time. This feature represents the relative position of the maintenance date with respect to the simulation end date.
- $p_{t,a}$ is the probability of failure at each sub-interval "a" based on the RUL prognostics of a CBM task. In the case of routine blocks, the problem is deterministic and this feature is nullified.
- $b_{t,a}$ is a binary variable that indicates whether the scheduling unit AC_t is a routine block or a CBM task. This feature takes the same value for all $a \in \mathcal{A}$

At every iteration the features are calculated for each discretized sub-interval $a \in \mathcal{A}$, and the resulting vector becomes a column of the state as shown in Equations (1)-(2). The final state given in Equation 3 has a size of $|\mathcal{A}| \times 6$.

$$R_t = [R_{t,a}]_{a\in\mathcal{A}}, \quad D_t = [D_{t,a}]_{a\in\mathcal{A}}, \quad lh_t = [lh_{t,a}]_{a\in\mathcal{A}} \tag{1}$$

$$h_t = [h_{t,a}]_{a\in\mathcal{A}}, \quad p_t = [p_{t,a}]_{a\in\mathcal{A}}, \quad b_t = [b_{t,a}]_{a\in\mathcal{A}} \tag{2}$$

$$S_t = (R_t, D_t, lh_t, h_t, p_t, b_t) \tag{3}$$

3.2 Action Space

The action $a_t \in \mathcal{A}$ controls the scheduling environment by determining the aircraft utilization rate before entering maintenance. Thus, the action space is given by the discretized sub-intervals of the scheduling unit, plus the additional possibility of having an aircraft on ground (AOG). In the latter case, the aircraft is forced to be grounded and wait for a maintenance opportunity. Therefore, the complete action-space with the discretization scheme shown in Figure 1 becomes:

$$\mathcal{A} = \left\{15\%, 25\%, 35\%, ..., 95\%, 97\%, 99\%, 100\%, AOG\right\} \tag{4}$$

Let Figure 2 be a visualization of the maintenance scheduling environment, with the available slots on top, and the scheduling unit intervals of different aircraft on the bottom. The interval of $AC1$ on top of the list is the most critical as it has the earliest due date. Moreover, a total of five slots, indicated with a red "X", are available during the time window of this scheduling unit. However, only four actions are feasible based on the discretization scheme of the interval. For clarity purposes, the four feasible actions have been highlighted with a striped pattern. The second feasible action contains two slots, and if it were selected, the latest slot would be assigned to the most critical scheduling unit.

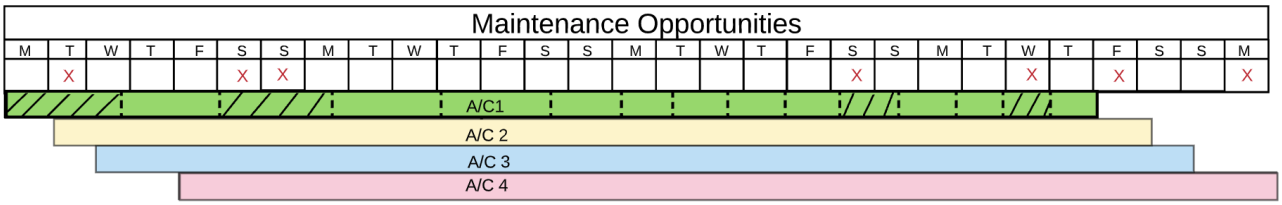


Figure 2: Maintenance Scheduling Environment

There are two types of scenarios that can occur. When a routine block has the earliest due date, the maintenance opportunities are given based on hangar availability. On the other hand, when a CBM task needs to be allocated, the opportunities are determined by the slots occupied with a routine block of the same aircraft tail. In this way, CBM tasks are inserted in the same slot as the A-checks. Ideally, the maintenance scheduling units should be allocated as close as possible to their due date in order to maximize aircraft utilization. Nevertheless, an action is feasible only if a maintenance opportunity is overlapping with the respective sub-interval, as shown in Figure 2 with the striped pattern. Thus, the feasible region $\mathcal{A}_t$ at time t is expressed as follows:

$$\mathcal{A}_t = \Bigg\{ a_i : \quad PE_t^{AC_t} + a_{i-1} \cdot I_t^{AC_t} \leq MD_i^{AC_t} \leq PE_t^{AC_t} + a_i \cdot I_t^{AC_t} \quad \forall i \in \mathcal{A} \tag{5}$$

$$1 \leq M_i^{AC_t} \qquad \forall i \in \mathcal{A} \Bigg\} \tag{6}$$

Constraints 5 ensure the maintenance slot date is in the desired sub-interval $[a_{i-1}, a_i]$, which is calculated based on the previous execution date ($PE_t^{AC_t}$) and the interval requirement ($I_t^{AC_t}$) of the maintenance scheduling unit. Likewise, constraints 6 consider action a_i to be feasible only if at least one maintenance opportunity is present at the time the aircraft is utilized by a_i percentage. Lastly, Equation 7 shows the feasible action space of the example presented in Figure 2.

$$\mathcal{A}_t = \Big\{ 15\%, 35\%, 90\%, 99\%, AOG \Big\} \tag{7}$$

3.3 Reward Function

In reinforcement learning, the sum of the rewards received over an entire episode determines the final objective function value, which is referred to as the cumulative reward. The evaluation of the action a_t, given the state S_t and the next state S_{t+1}, is assessed through the rewards received at each timestep. In this way, the model is able to extract information from the intermediate steps between the start and the end of an episode. We interpret the reward as a cost function that needs to be minimized by the model applied to the problem such that the fleet utilization is maximized.

The operational impact of the actions represents the interval limit consumed by an aircraft before entering the maintenance hangar. The *linear* sum of the interval utilization values would not lead to the maximization of the whole fleet utilization because some aircraft might have extremely high rates, while others might be utilized much less. Since the objective function would not reflect this behavior, an alternative approach is chosen in order to maximize the fleet utilization by means of the cumulative reward function. A logarithmic transformation of the interval utilization can still represent the cost of individual actions but, more importantly, the minimization of the cumulative reward would lead to the maximization of the overall fleet utilization. Equation 8 shows the cost or reward function shape, where the constant K takes value 1 for CBM tasks, while the K value of routine blocks is doubled in order to outline the hierarchy of a task-cluster. The cost function is shaped such that it is minimized when all the actions are as close as possible to 95%, as the airline strives to have a 5% margin due to the variability in aircraft utilization. Lastly, the action that is penalized most heavily corresponds to an *aircraft on ground* (AOG) or missed maintenance opportunity. This option has been included for feasibility purposes, such that if all maintenance opportunities are occupied or the due date of a task occurs before the prescribed maintenance date, a large penalty is returned in order to respect the scheduling unit's interval.

$$R_t(S_t, a_t) = \begin{cases} K \ln\left(\frac{100}{a_t}\right) & \text{if } a_t \leq 95\% \\ K \ln\left(\frac{100}{2 \cdot 95 - a_t}\right) & \text{if } 95\% < a_t \leq 100\% \\ 2K \ln\left(100\right) & \text{if } a_t \text{ is AOG} \end{cases} \tag{8}$$

3.4 Transition Dynamics

The transition function dictates the dynamics of the environment from state S_t to state S_{t+1}. It is noteworthy that, at each timestep, the agent processes the aircraft routine block or CBM task with the highest priority and decides to which opportunity it should be allocated. The complete set of steps involved in the simulation process are outlined as follows:

1. Simulate the fleet utilization
2. Update remaining useful life (RUL) for all aircraft
3. Calculate the next due date of all scheduling units
4. Select the most urgent scheduling unit based on the earliest due date
5. Process features for the selected scheduling unit
6. Select a scheduling action
7. Update environment attributes
8. Repeat the previous steps until the end of the horizon

Every time a scheduling unit AC_t is being processed in state S_t, a scheduling action a_t^* is chosen based on the policy of the reinforcement learning model. Subsequently, the maintenance resources are updated by subtracting the slot that has just been allocated (Equation 9), and the scheduled date variable ($SD_t^{AC_t}$) becomes the date of the chosen slot (Equation 10).

$$M_{a_t^*}^k = M_{a_t^*}^k - 1 \qquad \forall k \in \mathcal{K} \tag{9}$$

$$SD_t^{AC_t} = MD_{a_t^*}^{AC_t} \tag{10}$$

Once the current aircraft has been scheduled, the whole fleet utilization is simulated for a number of days that goes from the previous reference date RD_t^k until the selected scheduled date $SD_t^{AC_t}$. The simulated days are converted to flight hours and flight cycles according to daily ratios ($\delta_{fh}^t, \delta_{fc}^t$) that are estimated based on the calendar month at timestep t for each scheduling unit k (Equation 11). These parameters are used to update the remaining calendar days (CY), flight hours (FH), and flight cycles (FC) of the whole fleet in Equation 12. However, the remaining utilization parameters of the scheduling unit AC_t, that has just been allocated, are re-initiated to their original interval limitations, as shown in Equation 13. A limitation of this approach is the assumption of complete aircraft utilization information. Therefore, it does not include future uncertainty related to the utilization variability, for this reason a 95% target utilization was defined.

$$\begin{bmatrix} \Delta DY_t^k \\ \Delta FH_t^k \\ \Delta FC_t^k \end{bmatrix} = \begin{bmatrix} SD_t^{AC_t} - RD_t^k \\ \Delta DY_t^k \cdot \delta_{fh}^t \\ \Delta DY_t^k \cdot \delta_{fc}^t \end{bmatrix} \qquad \forall k \in \mathcal{K} \tag{11}$$

$$\begin{bmatrix} DY_{t+1}^k \\ FH_{t+1}^k \\ FC_{t+1}^k \end{bmatrix} = \begin{bmatrix} DY_t^k \\ FH_t^k \\ FC_t^k \end{bmatrix} - \begin{bmatrix} \Delta DY_t^k \\ \Delta FH_t^k \\ \Delta FC_t^k \end{bmatrix} \qquad \forall k \in \mathcal{K} \tag{12}$$

$$\begin{bmatrix} DY_{t+1}^{AC_t} \\ FH_{t+1}^{AC_t} \\ FC_{t+1}^{AC_t} \end{bmatrix} = \begin{bmatrix} I_{DY}^{AC_t} \\ I_{FH}^{AC_t} \\ I_{FC}^{AC_t} \end{bmatrix} \tag{13}$$

The remaining useful life variables (X_t^k) are updated with the minimum driving requirement either in DY, FH or FC in Equation 14. Then, the routine block due dates are updated based on the remaining interval days, while in the case of a CBM task a prognostic ω_t^k dictates the scheduling unit due date, as shown in Equation 15. The transition dynamics are slightly different when a CBM task needs to be allocated. Since it is necessary to simulate the prognostic multiple times with varying uncertainty, the CBM task scheduling occurs in multiple steps. At every iteration, the model updates the reference date to the following day ($RD_t^k + 1$) in order to simulate a chronological sequence of actions until the next available maintenance opportunity. On the other hand, the reference date for the routine blocks is directly updated to the selected scheduled date (Equation 16). In this way, the prognosticated due date (ω_t^k) is re-sampled with less uncertainty since the model gets closer to the end of the interval. The CBM task is considered to be allocated only when the reference date (RD_{t+1}^k) is the same as the scheduled date ($SD_t^{AC_t}$). The reward of the intermediary steps in the CBM task allocation process is set to zero until the scheduling date is reached. Furthermore, the previous maintenance execution of AC_t and the minimum interval for every unit in $\mathcal{K}$, are updated in Equations (17)-(18). Lastly, the scheduling units are positioned in descending order, based on the updated due dates, and the next most urgent one is selected (Equation 19).

$$X_t^k = \min \left(\begin{bmatrix} DY_t^k \\ FH_t^k / \delta_{fh}^{t,k} \\ FC_t^k / \delta_{fc}^{t,k} \end{bmatrix} \right) \qquad \forall k \in \mathcal{K} \tag{14}$$

$$DD_{t+1}^k = \begin{cases} RD_t^k + X_t^k & \text{if } k \text{ is a routine block} \\ \omega_t^k & \text{if } k \text{ is a CBM task} \end{cases} \qquad \forall k \in \mathcal{K} \tag{15}$$

$$RD_{t+1}^k = \begin{cases} SD_t^{AC_t} & \text{if } k \text{ is a routine block} \\ RD_t^k + 1 & \text{if } k \text{ is a CBM task} \end{cases} \qquad \forall k \in \mathcal{K} \tag{16}$$

$$PE_{t+1}^{AC_t} = SD_t^{AC_t} \tag{17}$$

$$I_{t+1}^k = DD_{t+1}^k - PE_{t+1}^k \qquad \forall k \in \mathcal{K} \tag{18}$$

$$AC_{t+1} = \arg\min_k \left(\left[DD_{t+1}^k \right]_{k \in \mathcal{K}} \right) \tag{19}$$

Figure 3 shows the transition dynamics that allow the agent to move from slot to slot. At every timestep, a new scheduling unit AC_t is selected based on its due date. The action chosen determines the opportunity to which the respective scheduling unit is allocated, as well as the simulated days of fleet utilization. In the case of S_{t+3}, the aircraft selected cannot be scheduled after the reference date. For this reason, the sequence of actions takes a step backwards in the schedule. It is noteworthy that the timestep index indicates the chronological order in which the actions are taken. The actual date and time is determined by the date of the maintenance slot that is selected with every action.

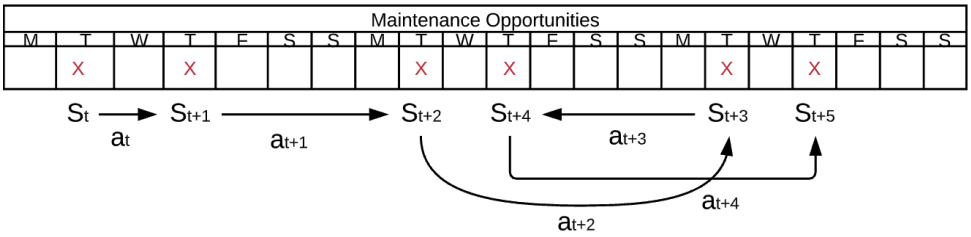


Figure 3: Transition Dynamics

4 Methodology

To optimize the maintenance task allocation and increase aircraft utilization, the scheduling framework developed in this work devises three modules. The first is the clustering algorithm required to create the routine blocks. The second module simulates the synthetic prognostics of the (de-)escalated CBM tasks. Lastly, the third module allocates maintenance opportunities to both routine blocks and CBM tasks with a reinforcement learning (RL) agent. The scheduling process is sequential. First, a routine block or A-check must be allocated to an available slot. Then, the CBM tasks corresponding to the same aircraft tail can be inserted in maintenance opportunities defined by A-checks.

4.1 Routine Block Clustering

The benefit of clustering tasks in routine blocks is the reduction of the problem size. In fact, there is an analogy between the maintenance scheduling problem and a puzzle. The more pieces there are, the more complicate the problem becomes. There is no doubt that a highly segmented maintenance strategy could be more optimal from a mathematical perspective. However, scheduling maintenance implies a disturbance to operations because the aircraft has to be grounded for a certain period of time. Moreover, from a planning perspective, overhead costs related to hangar, engineers, and tool availability must also be considered. In recent years, airlines have begun to explore policies that increase the number of checks and reduce the ground time. This enables overnight checks and increases aircraft availability (Muchiri and Smit, 2009). These strategies foster the integration of CBM in the airline industry. In fact, if the aircraft is grounded more times for shorter periods, there are more maintenance opportunities that could be used to perform individual tasks monitored by prognostics. To address this hypothesis, we consider different maintenance block cycle policies that vary the frequency of the routine blocks, and the respective maintenance elapsed time.

Figure 4 shows the logic behind different maintenance block cycle policies and the clustering algorithm. The cycle on the left considers a maintenance policy with 1500 flight hours between every check, while the cycle on the right employs a policy spaced by 750 flight hours. Therefore, twice as many blocks appear in the same horizon when the maintenance frequency is increased. Nevertheless, the duration of these routine blocks is halved to maintain a similar amount of labor hours between the two policies. Additionally, two other maintenance policies with 500 and 375 flight hours between blocks are evaluated to observe scheduling performance in such scenarios. A calendar based estimation for these policies would lead to the grounding of aircraft every three months (1500FH) to every 3 weeks (375FH). The implementation of these policies requires ad adaptation of the available slots schedule, used as input for the RL model. The available slots are subdivided in multiple opportunities when maintenance

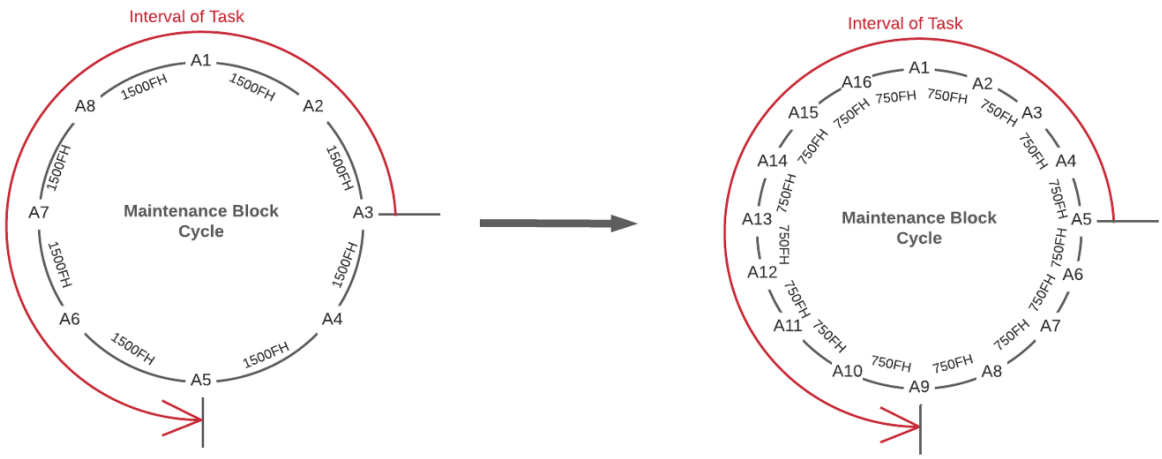


Figure 4: Clustering Strategy

frequency is increased. Thus, a 50% decrease in interval flight hours between the blocks (from 1500FH to 750FH), leads to the creation of two maintenance slots with half of the original labor capacity, such that one slot becomes a grounding opportunity for two aircraft. Similarly, when the flight hours are reduced to 500FH and 375FH, one maintenance slot in the original slot calendar becomes a slot for 3 and 4 aircraft, respectively.

The traditional maintenance policy of the reference airline clusters all the routine tasks with a 1500FH policy. Their task-packaging strategy involves several expert criteria that are used to produce an operational maintenance block cycle. Among others, the interval requirement, the labor hours, the skill of the licensed personnel, the aircraft zone, the inventory items, and other task inter-dependencies are considered in the task-packaging process (Pereira and Ashok Babu, 2016; Ozkol and Senturk, 2017). There are two main reasons for which the clustering algorithm is employed to create new maintenance task-packages or routine blocks. Firstly, different maintenance block policies than the one proposed by the reference airline are explored, for which the interval between blocks is varied. Secondly, a portion of the routine tasks is removed and monitored individually by means of synthetic prognostics. Therefore, the composition of routine blocks changes and a new clustering methodology is required. For the sake of simplicity, tasks have been clustered only based on interval and slot capacity considerations in this research. The following parameters have been defined to formulate the clustering MILP.

Sets

J : The set of routine tasks for an aircraft

B : The set of blocks based on the maintenance policy configuration

P_{ij} : The set of blocks that have a distance from block i that is less or equal than I_j

Parameters

I_j : The interval of task j

D_j : The duration of task j

LH : The capacity of labor hours in each slot

Decision Variable

x_{ij} : Task j is assigned to block i

The objective function of the clustering algorithm is to minimize the number of times a task is repeated in the maintenance block cycle (Equation 20). The coverage constraints in Equation 21 ensure that all the maintenance tasks are selected at least once in the maintenance block cycle. Constraints 22 prevent a task from being assigned to a block past its interval limitation. The equation makes sure that if task j is allocated in block i, it appears one or more times in the previous blocks P_{ij}. The reader is referred to the policy on the left of Figure 4 for a graphical definition of set P_{ij}. If the model considers placing a task in block A3, the same task should appear at least once in the blocks covered by the red line in order to respect the interval limitation. In the depicted case, P_{ij} is composed by blocks $A5, A6, A7, A8, A1$, and $A2$. Moreover, constraints 23 ensure that the number of tasks allocated in a certain block is constrained by the maximum labor hour capacity. A major limitation of this formulation is the lack of consideration for a homogeneous workload distribution over the maintenance blocks. Lastly, constraints 24 indicate that the decision variable is a binary variable.

$$\textbf{minimize} \sum_{j \in J} \sum_{i \in B} x_{ij} \tag{20}$$

$$\sum_{i \in B} x_{ij} \geq 1 \qquad \forall j \in J \tag{21}$$

$$x_{ij} - \sum_{p \in P_{ij}} x_{pj} \leq 0 \qquad \forall i \in B, \forall j \in J \tag{22}$$

$$\sum_{j \in J} x_{ij} \cdot D_j \leq LH \qquad \forall i \in B \tag{23}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in B, \forall j \in J \tag{24}$$

4.2 Synthetic Prognostic Generation

CBM requires health monitoring data to diagnose the condition of the aircraft, and predict the remaining useful life (RUL) of different systems, components and structures. Legacy aircraft are still not fitted with many of these sensors and RUL predictive models are still emerging in the PHM industry. Moreover, every system should have a tailored predictive model which is not available for this research. Therefore, a synthetic prognostic module has been designed to simulate the CBM task interval escalation. It is essential to understand that there will be always uncertainty related to a predicted due date. Saxena et al. (2008) stress the need of considering the impact of uncertainty in order for the prognostic to be useful. For this reason, we produce probabilistic interval estimations of the due date that we draw from a normal distribution. Based on the works of Sankararaman et al. (2013) and Elattar et al. (2016) it is clear that the RUL prediction models are mainly based on log-normal, Weibull or Gaussian distributions. Although the probability distribution is likely to be asymmetric, a normal distribution is assumed because this study generalizes over multiple sub-systems and different failure possibilities. The main goal of this work is to schedule routine checks and CBM tasks while considering RUL uncertainty dynamically.

Figure 5 has been included to guide the reader through the prognostic simulation process. Assuming a routine task with an original interval of 1500 flight hours can be escalated by $E\%$, it becomes a CBM task. The risk parameter r indicates the maximum date up to which the CBM task can be

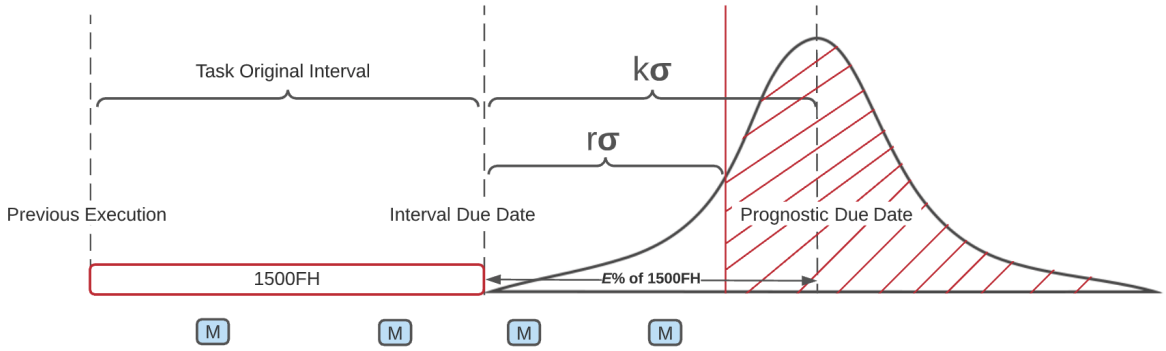


Figure 5: Synthetic prognostic simulation

scheduled. Past the risk limit, represented by the red vertical bar in Figure 5, there are no more maintenance opportunities identified in blue at the bottom of the figure as the failure probability is deemed too high and the task may fail prior to the allocated slot. Furthermore, the parameter k determines the standard deviation or the uncertainty of the RUL prognostics. The uncertainty σ is calculated dividing the escalated interval by k such that the 99.7% confidence interval (3σ) is contained within a specific uncertainty limit. For example, if $k = 6$, then the corresponding uncertainty is 50% because the 3σ interval is equal to half of the $k\sigma$ one. The uncertainty value represents the accuracy that an eventual RUL predictive model can offer. Therefore, as parameter k increases, the uncertainty σ becomes smaller. Furthermore, the initial due date expectation or the mean of the distribution is determined in Equation 25 such that the prognostic is centered around the escalation date, and the initial uncertainty is calculated with respect to the original interval due date in Equation 26. Lastly, the predicted due date (ω_t) is drawn from the normal distribution shown in Equation 27.

$$\mu_t = DD_t^{AC_t} + E\% \cdot X_t^{AC_t} \tag{25}$$

$$\sigma_t = \left(\mu_t - DD_t^{AC_t}\right)/k \tag{26}$$

$$\omega_t \sim \mathcal{N}(\mu_t, \sigma_t) \tag{27}$$

Throughout the schedule generation process, the prognostics are simulated dynamically with less uncertainty as the algorithm gets closer in time to the predicted due date. Elattar et al. (2016) defines this concept as "the shrinking of the required accuracy margin when the algorithm approaches the end of life". This means that algorithm performance must improve with time as more data becomes available, being the best just before the end of life. For this reason, we assume that the confidence interval of the prediction decreases linearly in a Gaussian uncertainty propagation model. Every time the RUL prognostics are re-sampled, the uncertainty is decreased proportionally to the previous one based on the time difference between each update. Furthermore, the new prognostic is assumed to be centered around the previously sampled due date. In this way, the performance of the prediction evolves with time, being less confident at the beginning of the evaluation. As the predicted failure date approaches in time, the prediction uncertainty decreases and the prognostic is re-evaluated. Therefore, the dynamic distribution is sampled again based on the following parameters:

$$\mu_{t+1} = \omega_t \tag{28}$$

$$\sigma_{t+1} = \left(\omega_t - RD_t^{AC_t}\right)/k \tag{29}$$

The CBM task interval goes from the *Previous Execution* up to the *Prognostic Due Date* (see Figure 5). The main difference between the interval of a block and a CBM task is that the latter continuously changes based on the sampled due date. However, the discretization of the interval explained in section 3 is carried out in the exact same manner for both types of scheduling units. Lastly, the feature $p_{t,a}$ is determined by the cumulative probability of the failure occurring before the date of the observed maintenance slot, as shown in Equation 30 where P is the cumulative probability of the normal distribution $\mathcal{N}(\mu_t, \sigma_t)$.

$$p_{t,a} = F(MD_a^{AC_t}) = P(\mu_t \leq MD_a^{AC_t}) \tag{30}$$

4.3 Deep Reinforcement Learning

The goal of reinforcement learning is to develop a policy for sequential decision problems, by optimizing the cumulative discounted reward signal in Equation 31. The discount factor $\gamma \in [0, 1]$ assigns the importance of future rewards with an exponential factor with respect to the current timestep, such that the future returns are considered to be of less value than current rewards. The choice of discount factor requires careful consideration about the relevance of future rewards for the current state. It is important to consider the effect of a scheduling action on the short-term because it may compromise opportunities for the rest of the fleet, as well as in the long-term because a scheduling action influences the start of the following interval.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{31}$$

4.3.1 Q-Learning

The AMSP is solved with Q-Learning which is one of the most widely known reinforcement learning algorithms. The Q-function, also called action-value function, is the expectation of the discounted reward sum given a certain action a_t in a state S_t. The challenge of Q-Learning is to estimate the future discounted sum of rewards for a state-action pair. By taking advantage of the problem's recursive nature, the objective function can be expressed in the form of the Bellman equation (Sutton and Barto, 1998). From Equation 32 it can be observed that the Q-function is employed to provide an estimate of the cumulative discounted reward. The optimal policy π^* of the Q-learning model can be deduced by minimizing the Q-function over the feasible region as expressed in Equation 33. A policy $\pi : S_t \rightarrow a_t$, is defined as a function that maps a state S_t into an action a_t.

$$Q^{\pi}(S_t, a_t) = \mathbb{E}^{\pi}(G_t|S_t, a_t) = \mathbb{E}^{\pi}\left(R_{t+1} + \gamma \cdot Q^{\pi}(S_{t+1}, a_{t+1})\right) \tag{32}$$

$$\pi^*(S_t) = \underset{a_t \in \mathcal{A}_t}{\arg\min}\, Q^*(S_t, a_t) \tag{33}$$

In a Markov decision process (MDP), each sequence contains information of a different state. This information is referred to as the agent experience and it is stored at every timestep with a tuple $< s, a, r, s' >$ that represents the state, the action, the reward and the next state of each transition. The Q-value of each state-action pair is computed by means of Equation 34, where α is the learning rate of the model.

$$Q^{new}(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \min_{a'} Q\left(s', a'\right) - Q(s,a) \right] \tag{34}$$

4.3.2 The Deep Q-Network

The state-space of the AMSP is extremely large and the solution space is non-convex. The dimensionality problem can be circumvented with a neural network as a non-linear function approximator that learns a dynamic scheduling policy. Moreover, it is not required to sweep across all the state-action pairs to achieve convergence. In this way, the neural network facilitates forward dynamic programming (Powell, 2011): it replaces the computational burden of looping through each possible state-action pair, with the statistical problem of estimating their value.

A Deep Q-Network (DQN) is a multi-layered neural network with weights θ. The input of the neural network is the state S_t and it has a fixed output layer with $|\mathcal{A}|$ neurons, each representing the $Q(S_t, a_t, \theta)$ value from which it can be extracted the optimal policy. There are two main components in a DQN (Mnih et al., 2015). The first one is the target network with weights θ^- which is a delayed copy of the online network. The purpose of the target network is to update the loss function visible in Equation 35. It is noteworthy, that the weights of the target network remain fixed during τ steps to improve the learning stability, such that the online network can be calibrated in the direction of a stationary target. The second element of this algorithm is the experience replay buffer $\mathcal{R}$, which is a memory where the agent stores the observed experiences. Then, at each timestep the agent randomly samples a batch of experiences to train the neural network. During the training phase, the weights θ of the online neural network are optimized with a stochastic gradient descent (SGD) algorithm to minimize the mean squared error of the loss function with respect to the weights θ of the online network (Hasselt et al., 2016).

$$L(\theta) = \mathbb{E}_{<s,a,r,s'>\sim\mathcal{U}(\mathcal{R})} \left(r + \gamma \min_{a'} \underbrace{Q(s', a'; \theta^-)}_{\text{Target network}} - \underbrace{Q(s, a; \theta)}_{\text{Online network}} \right) \tag{35}$$

4.3.3 Exploration vs Exploitation Dilemma

In large state-space environments, the agent needs to weigh the possibility of exploring other regions of the search space during the training, and thus giving up short-term rewards. To overcome this paradigm, we employ a widely known policy to train the agent, also called ϵ-greedy (Sutton and Barto, 1998). The agent follows the greedy strategy with probability $1 - \epsilon$ and selects a random action with probability ϵ. Furthermore, the exploration rate ϵ is decayed over time. As the learning begins to converge, the behavioral policy shifts toward an exploiting behavior while keeping a reduced exploration rate of ϵ=1%. Q-learning is classified as an off-policy algorithm because it learns a greedy

policy to minimize the cost value of its actions, however, the behavioral policy is dictated by an ϵ-greedy strategy. In other words, the target can be computed without considering how the experience is generated.

4.3.4 Training Strategy

Figure 6 depicts a representation of the reinforcement learning concept. The agent observes the state from the environment that contains information about the upcoming due dates, the maintenance opportunities and the RUL prognostics. Furthermore, the DQN produces an estimation of the Q-value for every state-action pair, and the agent selects the one with the minimum cost. In order to respect the slot availability constraints, the **argmin** operation is allowed only across the feasible region $\mathcal{A}_t$. Literature refers to this variant of the algorithm as **constrained** Deep Q-Learning (Kalweit et al., 2020). It is also possible to highly penalize the agent when choosing an unfeasible action. However, experiments conducted for this work demonstrated that the training convergence could be achieved with a smaller amount of episodes, and the scheduling performance returned better results when the agent action-space was constrained. Once the policy has been evaluated, the scheduling action is applied to the environment and the respective reward is used to optimize the DQN weights. In model-free learning, the transition function is implicit in the update. This means that the Q-function is updated iteratively based on the new experiences observed, without actually learning the underlying transition dynamics. Although the assumption of a *fully observable* process is required by the MDP, it can be argued that the state is not Markovian due to inherent uncertainties involved in the AMSP. Nevertheless, we assume that prognostics are sufficiently reliable to schedule the aircraft before their actual due date. As a result, the model can adapt to stochastic situations and the rewards received will adjust the Q-function based on the agent's experience.

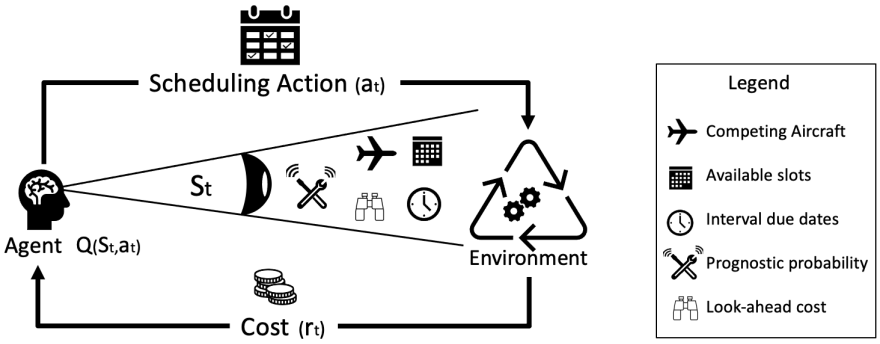


Figure 6: Agent-Environment Interaction

5 Experimental set-up

5.1 CBM scenarios

The complete Aircraft Maintenance Program (AMP) contains 1086 tasks that an aircraft should undergo during its lifecycle. The maintenance plan considered in this research contains only 186 routine tasks based on the A-check data provided by the reference airline. This corresponds to a small fraction

(17%) of the total AMP. Each task has a different interval specification and an estimated duration to be considered when clustering tasks into blocks. Moreover, two case studies have been designed based on a moderate and an optimistic scenario of CBM integration. They are benchmarked against the baseline case where all the routine tasks are considered to be part of a maintenance block. Under the MSG-3 approach, a series of task groups are identified for air-frame systems (Kinnison, 2004). These groups have been reported in Table 2 with a detailed overview of the percentage of tasks that have been substituted or escalated in each of the three cases. Functional Checks (FNC) and Operational Checks (OPC) are heavily dependent on the acquisition of condition-data, therefore, a higher action rate has been prescribed for those task groups. The remaining tasks that are not monitored by CBM are clustered in different blocks based on the maintenance block cycle chosen. The reference airline's policy groups the A-check tasks in a sequence of 24 blocks spaced by 1500 flight hours. In order to reap the benefits of CBM, three other policies are defined with interval limitations of 750, 500 and 375 flight hours, respectively. During the second and third case, 10% and 25% of the LUB, RST, and DIS task groups are escalated, and the remaining task groups are substituted with similar action rates. This corresponds to 0, 4 and 14 tasks being escalated per aircraft at every case, while the number of substitutions is 0, 18 and 43, respectively. Thus, the maximum number of escalatable tasks is 14, while the most tasks that are substituted is 43. In total, this corresponds to 57 tasks in the scope of CBM. During the result analysis, these tasks are closely monitored to understand the scheduling performance when they are part of routine block or part of the CBM implementation.

Table 2: CBM Scenarios

Task Group	**Nr.**	**CBM Action**	**Case 1 [%]**	**Case 2 [%]**	**Case 3 [%]**
General Visual Inspection (GVI)	41	Task substitution	0	10	25
Visual Check (VC)	9	Task substitution	0	10	25
Functional Check (FNC)	18	Task substitution	0	25	50
Detailed Inspection (DET)	21	Task substitution	0	10	25
Servicing (SVC)	8	Interval Escalation	0	10	25
Lubrication (LUB)	18	Interval Escalation	0	10	25
Restoration (RST)	14	Interval Escalation	0	10	25
Discard (DIS)	23	Interval Escalation	0	10	25
Operational Check (OPC)	34	Task substitution	0	25	50

5.2 PHM uncertainty and interval escalation

The stochastic version of the AMSP considers prognostics to perform task interval escalation. The escalation value represents the factor by which the task interval can be extended if monitored with PHM. Even though the interval of a task can be extended, the uncertainty of the prediction plays an essential role in the performance of the scheduling model, as it indicates how much the actual due date can vary with respect to the estimated due date. Any airline would prefer to schedule with conservative limits in order to ensure safety and reliability over postponing maintenance. To approach this issue, the escalation parameter is varied throughout the simulations in order to gain understanding on the benefit of task escalation with respect to the interval utilization. On the other hand, it is of paramount importance to consider the level of uncertainty that PHM technology can offer. The uncertainty value is expressed such that the 99.7% confidence interval (3σ) of the prognostic distribution is contained

within $\pm U\%$. From Figure 5 is possible to visualize the interval escalation ($E\%$) and the uncertainty interval ($k\sigma$) parameters. In the following tables, the values of both parameters varied throughout the sensitivity analysis are outlined. In Table 3 the uncertainty is presented as a function of parameter k, and as percentage variation with respect to the prediction distance in time. For clarification an example has been reported in the last column for the resulting variation in the case the RUL prediction is estimated to be in 90 days. Let $k = 3$, then the corresponding uncertainty is 100% because the 3σ interval is equal to the $k\sigma$ one. An uncertainty of 100% entails that the due date can vary $\pm 100\%$ of the distance $k\sigma$ which is the equivalent to the distance between the current time and the estimated due date. In the case of a predicted due date in 90 days, an uncertainty variation of 100% denotes a possible change of ± 90 days. Similarly, if the uncertainty would be 10%, the actual due date could vary ± 9 days with respect to the predicted one. The interval escalation values presented in Table 4 are based on the hypothesis tested by Hölzel et al. (2014).

Table 3: Uncertainty variation of prognostic

k	Uncertainty (U%)	Variation [DY]
3	100.0	90
3.5	85.7	77
4.5	66.7	60
10	30.0	27
20	15.0	13.5
30	10.0	9
100	3.0	2.7

Table 4: Interval Escalation variation

Escalation (E%)
25.0
50.0
75.0
100.0
125.0

5.3 DQN configuration

In general, neural networks can be described as parametric functions that need to be calibrated during the training process. Nevertheless, some parameters are established beforehand in order to initialize the model. The scheduling units have been discretized in sub-intervals of 1% in order to have a fine mesh of all the possible slots. Consequently, the neural network input layer has a size of 100 neurons. The model is composed by three sequential hidden layers, comprising 100 neurons each with a *swish* activation function (Ramachandran et al., 2017). The activation function converts the input of the neurons with a non-linear transformation. Instead, the final layer is designed with a linear activation function because the output represents the Q-value of the next-state and it should converge to the discounted reward value. Moreover, the output layer is sized with 101 neurons corresponding to the discretized sub-intervals and the AOG action. The learning rate, the exploration decay and the discount factor were selected upon a careful sensitivity analysis. The discount factor is an essential component as a very low value tends to create a greedy behavior that compromises the future fleet opportunities, while a very high discount factor does not provide relevant information to the agent. Since, the maintenance scheduling problem is solved with a long-term horizon, the decisions far in the future are not relevant to actions taken at the beginning of the simulation. The optimal model performance is achieved when discounting future rewards by a 0.5 factor. Lastly, the Q-Learning model has been trained over a period of 100 episodes in order to observe convergence. A complete list of the relevant DQN parameters has been reported in Table 5.

Table 5: DQN Hyperparameters

Parameter	Value
Learning rate (α)	0.0001
Discount factor (γ)	0.5
Initial exploration rate (ϵ_0)	1.0
Final exploration rate (ϵ_T)	0.01
Exploration rate decay ($d\epsilon/dt$)	0.9
Target delay (τ)	10
Batch size	32
Hidden layers	3
Dense size (neurons)	100
Training episodes (T)	100

6 Results

The case study investigates the maintenance schedule of a narrow-body fleet, composed by 16 aircraft, with a planning horizon of 12 months. The slot opportunities are assumed to be available twice a week for a duration of 24 hours. Normally, these resources can only be used by a single aircraft, based on the airline policy that grounds each aircraft every 1500 flight hours. We assume that one slot can be used by more aircraft that share the labor hours capacity when the flight hour interval policy is reduced. Furthermore, the flight schedule data of the previous 5 years has been used to estimate the aircraft utilization, and the last maintenance execution of each aircraft is used to initialize the problem. The resulting schedule is analyzed in order to recommend the adaptive maintenance policy that maximizes the benefits of a CBM strategy. The quality of the maintenance scheduling policy is evaluated by the interval utilization, the labor hours, and the aircraft availability. The case study results focus on a scenario with a 100% interval escalation and an uncertainty of 15%, based on a PHM accuracy of $\pm$13.5 days for a due date predicted in 90 days time. A sensitivity analysis is included to address the effects of PHM uncertainty and interval escalation.

6.1 Training performance

The DQN training performance is monitored to ensure that the agent is able to converge towards the optimal policy and the cumulative cost function is minimized. In order to reduce the computational efforts required to train the DQL agent, the learning is performed across a reduced instance of eight months for each case. During the initial training phase, an exploration period takes place where sub-optimal decisions are selected multiple times, and the agent learns the impact of each action with respect to the state information. The training evolution is characterized by fluctuations in the cumulative rewards due to the DQN calibration process in a stochastic environment. To assess the training performance, the cumulative rewards are smoothed out using a *simple moving average* (SMA) over 10 samples. Figure 7 shows the SMA of the Deep Q-Learning algorithm with different scheduling policies for Case 3, which includes the most CBM tasks. The vertical axis of the training evolution figure is scaled with a logarithmic transformation based on the shape of the reward function in order to capture the resolution of the rewards at each phase. The buffered areas around the SMA curve represent one standard deviation of the 10 samples included in the calculation. From the figure, it is

possible to visualize that the four policies do not converge to the same steady-state value. However, this behavior does not represent their relative performance. As the flight hour interval decreases, the aircraft maintenance events become shorter and more frequent. Hence, a larger number of scheduling units is present, and a larger number of actions needs to be executed in one episode. Moreover, the increased availability of maintenance opportunities allows the agent to schedule the CBM tasks closer to the end of their interval which affects the reward obtained at each step. Therefore, the main takeaway from the training observation is the agent's ability to minimize the cost function in each of the stochastic maintenance scheduling environments. Lastly, the variation of the SMA, visible in the buffered areas, is larger when reducing the flight hours in the maintenance block cycle. This phenomena occurs because there are more maintenance opportunities to explore, and the due date can be postponed more often based on the prognostics re-evaluation.

Figure 7: SMA of the cumulative rewards over training episodes for Case 3 (25% CBM)

6.2 Maintenance scheduling policies

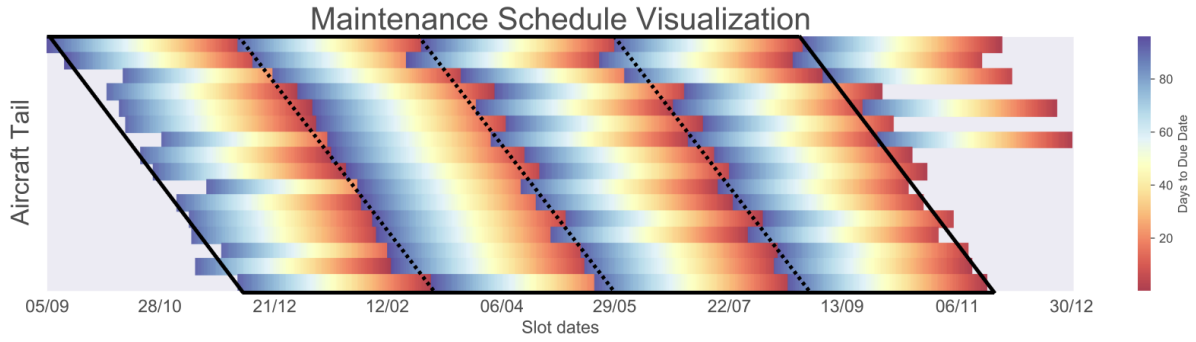
The routine blocks configuration is determined by the task clustering algorithm. The MILP clustering formulation is addressed using the commercial software package *Gurobi* with a computational time limit of two hours on an Intel Core i5 3.1GHz laptop with 16GB ram. The 186 routine tasks considered in the scope of this research are either bundled in a maintenance block or they are individually monitored with prognostics. Table 6 provides the reduction of routine task repetitions for each maintenance block policy and for each CBM case with respect to the airline policy. Case 1 with 1500FH has the same configuration as the airline current practices. Although there should not be any improvement since the same configuration parameters are being used, a reduction of 1.77% tasks repetitions is reported due to the fact that labor skills and task inter-dependencies were not considered in the clustering algorithm.

Moreover, as the interval reduces between each maintenance cycle, the improvement increases thanks to the availability of more maintenance opportunities. Lastly, when more CBM tasks are considered, the portion of routine tasks to be clustered decreases. In Case 3, task repetitions are reduced by more than 50% because a quarter of the aircraft maintenance program (AMP) shifts from a preventive policy (routine blocks) to a condition-based one (CBM tasks).

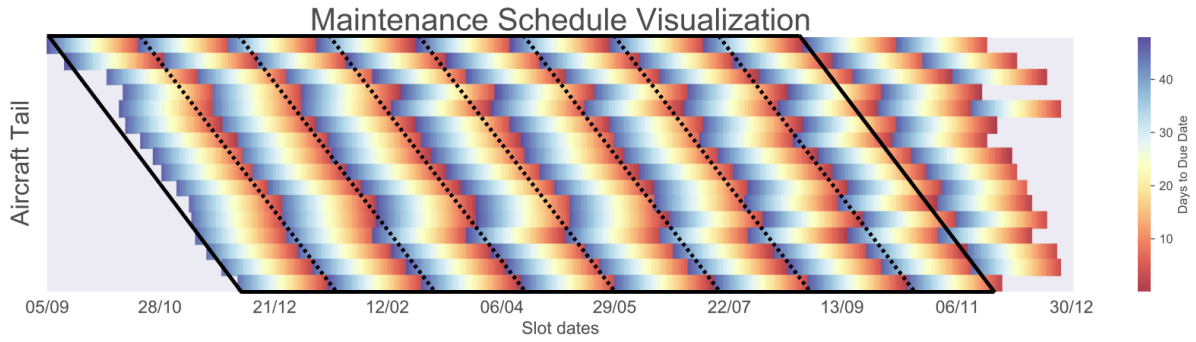
Table 6: Improvement of the clustering MILP with respect to the airline policy

Nr. Blocks	Interval (FH)	Case 1 [%]	Case 2 [%]	Case 3 [%]
24	1500	1.77	24.01	53.23
48	750	9.31	31.51	56.94
72	500	11.64	32.16	57.41
96	375	10.26	32.67	57.50

The number of blocks indicates the required maintenance executions to complete the maintenance block cycle. Nevertheless, in the 12 months simulation horizon, only four *full-checks* out of the 24 can be scheduled based on the minimum interval of 1500 FH (approximately three months) established by the airline policy. We define one *full-check* as the execution of one maintenance block in the 1500FH policy, as two blocks in the 750FH policy, as three blocks in the 500FH policy, and as four blocks in the 375FH policy. This is done such that a similar time interval and labor workload are considered for every policy. It is noteworthy, that policies which are more segmented divide one check over multiple blocks, which implies that the start of the next check will happen earlier than in the lower-frequency policies. To minimize the effect of this final boundary condition, we consider a horizon of 12 months per aircraft, starting from the initial condition of the problem.



(a) 1500FH Policy



(b) 750FH Policy

Figure 8: Remaining days to due date and horizon in scope of the results (black contour)

A visualization of the investigated horizon and the maintenance schedule is presented in Figure 8 for two different policies in Case 3. The black contour indicates the 12 month time horizon per aircraft in the scope of our analysis. The boundary condition of the problem are shown in dark gray. The fleet is always initialized with the same condition for both policies, as shown on the left of the plot. On the other side, the simulation is halted when one of the aircraft reaches the end of the simulation horizon.

Even though the schedule is simulated until the end of the year, only the schedule within the 12 months horizon in the black contour is considered for the analysis. The heatmap depicts the remaining days to the due date for every aircraft, so each row indicates the maintenance cycle of a specific tail. When the color changes from red to blue, a new due date is calculated from that point in time, which means that the aircraft has been scheduled for maintenance. For clarity purposes, the scheduled dates are indicated with the dotted parallel lines. Each of the parallel lines indicates that one maintenance grounding has been scheduled for the complete fleet. In Figure 8a, four clear sections are delimited by the dotted lines. They indicate that each aircraft undergoes four maintenance groundings. Instead, eight sections are encountered in Figure 8b. This corresponds in both cases to the completion of four *full-checks*, however, in the first case only four groundings opportunities are encountered, while for the latter case they are doubled.

(a) 1500FH Policy

(b) 750FH Policy

(c) 500FH Policy

(d) 375FH Policy

Figure 9: DQN Action distribution - Interval Utilization [%]

The distribution of the actions chosen by the DQN model is presented in Figure 9 for the four maintenance policies applied to Case 3. The agent chooses the interval utilization percentage of a scheduling unit (routine block or CBM task) at every time step. Additionally, the possibility of missing a maintenance opportunity is labeled as "Missed". However, it is never chosen throughout the episodes tested. It is noteworthy, that a segmented maintenance policy generates a shift towards a higher interval uti-

lization in the action distribution. Initially, two peaks are evident in Figure 9a that correspond to the CBM tasks and the routine blocks utilization, respectively. The DQN struggles to achieve high interval utilization for the CBM tasks in the 1500FH policy because the maintenance opportunities are spaced by too much time. Nevertheless, the two peaks become closer when the policy interval decreases to 750FH (Figure 9b). This behavior takes place because the grounding opportunities are more abundant. Therefore, a higher interval utilization can be achieved thanks to the availability of more maintenance opportunities closer to the due date, and the re-evaluation of prognostics over multiple opportunities. The segmentation in the maintenance policy increases the interval utilization until almost all the CBM tasks and routine blocks are scheduled between 80%-100% as shown in Figure 9c and Figure 9d.

6.3 Key performance indicators

While the interval policy effect is clearly visible from the action distribution of the DQN, the model performance cannot be analyzed only based on the utilization of the individual scheduling units. Table 7 provides an overview of the key performance indicators (KPI's) per aircraft during the time horizon of 12 months (365 days). The results of the scheduling algorithm are based on 100 Monte Carlo simulations for each scenario or row of the table. The first two columns indicate the scenario configuration in terms of CBM action rate and interval policy. The third column includes the average *full-checks* executed. The policies with smaller flight hour intervals perform slightly more *full-checks* due to the final boundary condition effect: a higher maintenance frequency inevitably leads to the execution of part of the following check at the end of the horizon. For this reason, the average labor hour per check has been included in the ninth column of Table 7. Clearly, the introduction of CBM and a segmented policy lead to a reduction of the required labor hours per check.

Table 7: KPI's per aircraft with 100% interval escalation and 15% uncertainty (365 days horizon)[1]

Case	Policy	Checks	Subs.	Esc.	Util.[%]	Avail.[DY]	Labor[hrs]	Labor[hrs/check]	Comp.[min]
1	1500	4.00	150.00	48.00	-	361.24	302.48	75.62	2.31
1	750	4.00	139.00	43.00	-	361.34	289.74	72.43	4.57
1	500	4.15	141.50	45.69	-	361.29	296.52	71.45	7.00
1	375	4.25	140.00	46.00	-	361.26	286.37	67.38	9.66
2	1500	4.00	79.00	43.31	107.16	362.34	217.93	54.48	3.65
2	750	4.00	71.00	36.00	143.67	362.43	212.43	53.11	7.81
2	500	4.15	70.75	37.54	147.22	362.35	218.00	52.58	12.16
2	375	4.25	67.00	34.87	162.20	362.34	219.05	51.54	17.74
3	1500	4.00	0.00	24.73	113.61	363.09	161.36	40.34	11.55
3	750	4.00	0.00	18.29	148.68	363.26	149.17	37.29	24.15
3	500	4.15	0.00	17.00	160.15	363.24	150.34	36.26	40.27
3	375	4.25	0.00	16.94	168.71	363.32	143.50	33.77	54.78

The fourth column of Table 7 (Subs.) keeps track of the repetitions of the 43 aircraft tasks that are in the scope of CBM substitution. Instead, the fifth column (Esc.) considers the task repetitions of the 14 tasks in the scope of interval escalation. Initially, all tasks are bundled in one maintenance block

[1]Subs.: Tasks repetitions in the scope of CBM substitution
Esc.: Tasks repetitions in the scope of CBM escalation
Util.[%]: Original interval utilization of the escalated tasks
Avail.[DY]: Aircraft Availability in days
Comp.[min]: Computation time in minutes

for Case 1. Then, based on the configuration of each case, some substitution tasks are removed from the AMP, while the escalation tasks are scheduled based on the RUL prognostics. The reduction of task repetitions in the case of substitution occurs because they are not part of the AMP anymore. While for the escalated tasks, the reduction of task repetitions is attributed to the fact that they can be scheduled later than their original interval. The mean utilization (Util.) of the tasks monitored with RUL prognostics is reported in column five of Table 7. It is noteworthy, that in Case 1 there are zero CBM tasks, so the respective rows have been intentionally left blank. The mean utilization is very dependent on the flight hour interval, being higher when the policy is more segmented.

Even though the performance of individual scheduling units is increased, the amount of escalation tasks that has been considered does not produce a significant reduction in the total labor time. Considering the fact that approximately 75 labor hours can be performed in one day, in the best case, the ground time reduction due to interval escalation is lower than half a day per aircraft. The aircraft availability has been computed based on the ground time reduction in order to provide an estimate of the benefits for the airline network. On average, in the two CBM scenarios, the availability is increased by approximately 1.0 and 1.9 days, respectively. Although two days are not sufficient to justify a change in the flight schedule, it can be considered a realistic improvement. Initially, the reference airline A-check routine program comprises a maximum ground time of 300 labor hours or 4 maintenance days per year. The conclusion drawn from this analysis points to the fact that the introduction of CBM in 10% of the AMP can reduce labor time by 27% with respect to traditional policies, while a 25% CBM action rate decreases the aircraft labor hours by 48%. The reason why the decrease in labor hours is not proportional to the action rate lies on the interval of the CBM tasks. Initially, shorter intervals are chosen for the CBM tasks. Nevertheless, as the action rate increases, more tasks with larger intervals are included which do not have the same impact on the scheduling performance, since they have less occurrences in the studied horizon. Furthermore, the amount of tasks considered is too little to produce a significant improvement in the overall aircraft availability. Similarly, the increase of system utilization between the four interval policies, leads to a reduction of tasks that is too small to produce major changes in the aircraft availability.

Lastly, the computational times (Comp.) for each case have been reported in the last column of Table 7. This provides an estimate of how the problem complexity evolves as the number of CBM tasks is increased. More importantly, the computational time is affected by a greater factor when the flight hours are reduced. The shorter interval policies are able to re-evaluate prognostics with less uncertainty as the agent approaches the due date, thanks to the increased availability of maintenance opportunities. Though the interval utilization increases, this comes at the cost of longer computational times required to produce a maintenance schedule.

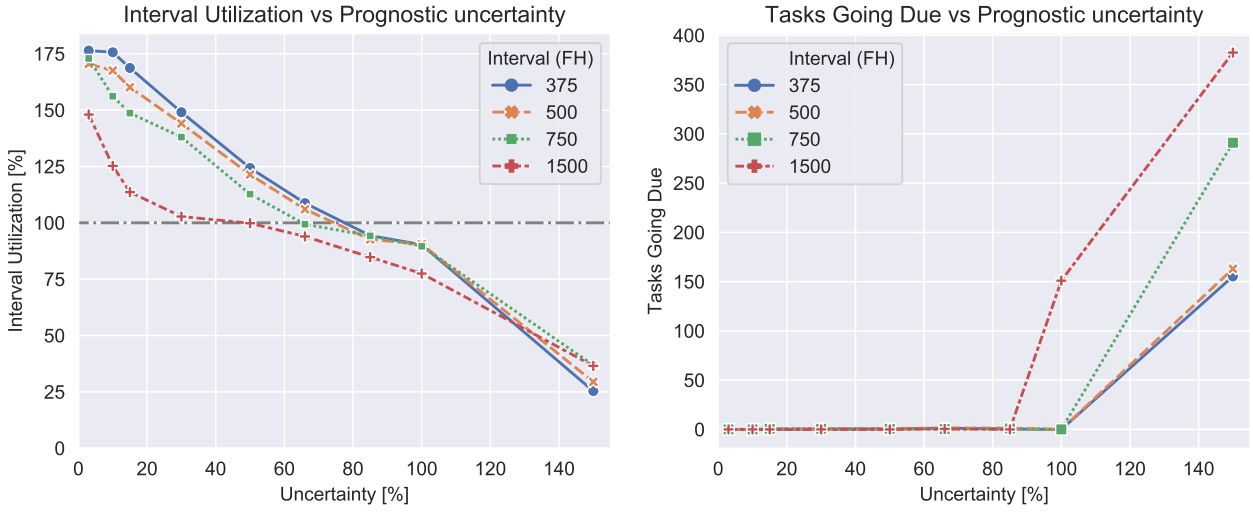
6.4 Sensitivity Analysis

6.4.1 Effect of PHM uncertainty

Among other challenges, the deployment of CBM is heavily dependent on the degree of accuracy that PHM technology can offer. For this reason, we study the effect of PHM uncertainty on the interval utilization of the escalated task. The impact of uncertainty variation for an interval escalation of

100% is summarized in Figure 10. The uncertainty accommodation has a parabolic relation with the utilization of escalated systems if there are not sufficient maintenance opportunities, as visible from the 1500FH policy in Figure 10a. The other maintenance policies accommodate the uncertainty variation with a linear trend because uncertainty effects are absorbed due to re-evaluation of prognostics as the agent gets closer in time to the due date. Moreover, any average interval utilization below the 100% line is considered non-beneficial. In theory, an interval escalation of 100% could double the original task interval, however, when the uncertainty is too large a conservative decision is made by the algorithm in order to preserve safety. The point where the system utilization curves cross the horizontal line can be considered as the minimum uncertainty level to consider the implementation of CBM policies.

Figure 10b clearly shows that the DQN keeps the number of *tasks going due* to zero up to an uncertainty level of 85%. When the uncertainty is increased to 100%, the 1500FH policy is forced to have an aircraft on ground (AOG) due to missed maintenance opportunities. The simulation has been extended to uncertainty levels up to 150% for verification purposes. This demonstrates that as the confidence interval decreases, the algorithm is forced to miss almost all the task monitored by prognostics. It also suggests, from this synthetic case, that if the RUL estimations are reasonably reliable, the DQN can avoid missing maintenance opportunities. This is an inherent trade-off between the risk level up to which the agent is allowed to schedule a CBM task and the AOG penalties. In this work, we have focus on a conservative scenario that minimizes risk in order to preserve safety which will be one of the key factors questioned by the airworthiness authorities during the implementation of CBM policies.



(a) CBM Interval Utilization

(b) Tasks going due

Figure 10: PHM uncertainty variation results of Case 3 with a 100% interval escalation

6.4.2 Effect of interval escalation

The influence of the interval escalation is assessed in Figure 11. A linear behavior is appreciated for an uncertainty level of 15% with respect to the interval utilization of the escalated tasks. As with the previous results, a clear offset is present between the four maintenance policies that are benchmarked in this analysis. When the interval escalation increases, the predicted due date is pushed further in time and the prognostic curve is also stretched to be centered around the new prediction. Thus,

the utilization slope does not conserve a one-to-one benefit with the escalation values because when escalation increases, the RUL uncertainty also grows. When there is more uncertainty involved in the prediction, the maintenance opportunities are selected with a more conservative approach in order to guarantee that any failure would not occur before the prescribed maintenance date. In actual practice, the escalation value is dependent on the specific operating conditions and the individual aircraft systems. This work generalizes over multiple subsystems and assumes it is possible to extend the task intervals based on the experiments studied by Hölzel et al. (2014), and the relaxation of the conservative interval requirements established by the MSG-3 (Vandawaker et al., 2015).

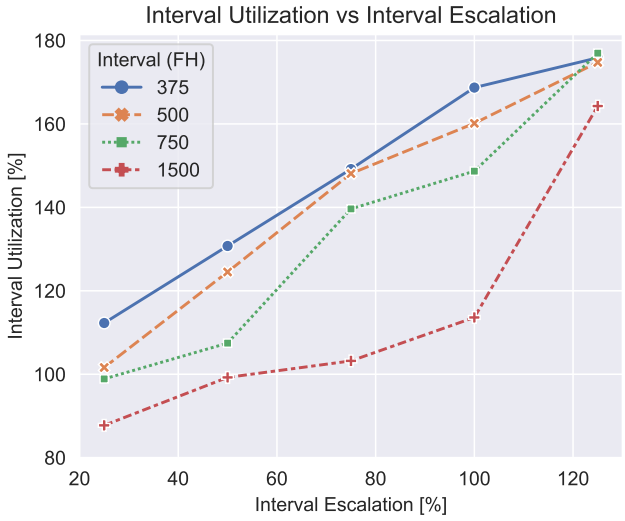


Figure 11: Interval Escalation effect with a 15% prognostic uncertainty (Case 3)

6.5 Cost-benefit analysis

The introduction of CBM in the airline maintenance strategy does not come at a low cost. In order to make a cost-benefit analysis, the required investment is compared with the cost reduction in labor hours. Since the labor hour reduction and the aircraft availability are approximately the same for the four interval policies, the average values for the three CBM cases are considered for this analysis. In Case 2 (10% CBM), the average labor hour reduction is 85.63 hours and the respective aircraft availability increase is 1 day. When the CBM action rate is increased up to 25% (Case 3), the average labor reduction becomes 151.39 hours, while the aircraft availability is increased by 1.9 days.

The implementation costs can be subdivided in sensor certification and installation. The certification procedure requires a one-time payment that on average costs €200,000 per sensor, while the installation is budgeted at €25,000 per aircraft based on the opinion of the reference airline MRO experts. Given these assumptions, the 16-aircraft fleet requires an overall investment of €600,000 per sensor. The labor cost savings per aircraft have been calculated based on the ground time reduction with respect to the baseline case assuming a rate of €100 per man hour. Table 8 shows that the investment cost can only be partially covered by the labor cost reduction. Based on an aircraft life cycle of 30 years, only 31.14% and 21.25% of the investment cost can be absorbed for the two CBM cases. Even if the maintenance could be reduced to zero hours, the yearly cost reduction would only be €30,000 per aircraft or €480,000 for the whole fleet. Thus, the labor hour reduction cannot absorb the CBM

investment cost in order to have a profitable maintenance strategy. Nevertheless, the maintenance cost should also include overhead, delay, slot, and the revenue loss due to aircraft downtime (Saltoğlu et al., 2016). For a more complete cost-benefit analysis of CBM strategies the reader is referred to the work of Vlamings et al. (2020). Their analysis demonstrates that the main source of income from a CBM policy should focus on the additional profit generated by an increased aircraft availability. We extend the cost-benefit analysis assuming a €11,000 aircraft profit per day based on an educated guess of the reference airline[2]. The respective profit increase leads to an absorption of the investment cost of 71.14% and 50.58% during an aircraft life cycle of 30 years.

Table 8: Cost-benefit analysis

Case	Tasks	Investment [M€]	Labor [k€/year]	Profit[k€/year]	Absorption [%]
10% CBM	22	13.2	137.00	176.00	71.14
25% CBM	57	34.2	242.22	334.40	50.58

7 Discussion

The implementation of CBM decreases the labor time thanks to the combined effect of CBM actions: substitution and escalation. In Case 2 a reduction of 27% of the labor hours is observed, while in Case 3 it is further reduced by 48%. Consequently, the aircraft availability is increased by 1 and 1.9 days, respectively. Given the reduced A-check program, that contains only 186 tasks, the results are promising. However, the aircraft availability improvement is too small to eventually produce major changes in the airline network and increase passenger revenue. Nonetheless, the marginal improvement in availability could be accounted as an added safety margin during disruptions in order to recover delays and cancellations. The conclusion drawn points to the fact that CBM cannot increase the fleet availability with a reduced maintenance plan that includes only A-check routine tasks. For the studied program, it is estimated that, on average, each aircraft requires 300 labor hours of maintenance that correspond to only 4 days on the ground. Even though the ground time is halved, an extended maintenance program needs to be considered in the scope of the research, including non-routine and unscheduled tasks, in order to improve the aircraft availability KPI.

Furthermore, we show that prognostics uncertainty is managed more efficiently when maintenance opportunities are increased. In this way, a re-evaluation of the predicted due date occurs multiple times over the interval of a task. In the event that there are not sufficient maintenance opportunities, the scheduling algorithm is forced to allocate the task much earlier than the due date to avoid the risk of missing the prescribed maintenance slot. The four maintenance policies that have been explored in this work (1500FH, 750FH, 500FH, 375FH) have the objective of exclusively improving the escalated task's interval utilization. It has been observed that the relation between interval utilization and the prognostic uncertainty of the escalated tasks follows a parabolic trend when the aircraft is grounded every 1500FH. Instead, when the maintenance opportunities are doubled with a 750FH policy, the trend becomes linear and the overall interval utilization per aircraft is increased. The results show that the original interval utilization in Case 3 is increased from 113.6% to 148.7% when reducing the flight hours

[2]The profit has been calculated with an assumed 10% operating margin and €110,000 aircraft revenue per day

from 1500 to 750. Further segmentation of the maintenance strategy provides a marginal improvement up to 168.7% utilization of the original interval. Thanks to the increased interval utilization, the DQN manages to schedule fewer tasks in the same time horizon. The four policies that have been explored reduce the task repetitions of the escalated tasks by 48.5%, 57.5%, 62.8% and 63.2% .

Despite the attractive performance that can be achieved with CBM, the number of escalated tasks in the scope of this research is too small to produce tangible benefits for the airline. In fact, the most optimal case reported a reduction of 29 tasks per aircraft in one year, which corresponds to approximately half a day of ground time. The potential of task interval escalation can be unlocked only when a larger portion of maintenance tasks is considered. In this research, the schedule is composed of interval-based blocks, and individual CBM tasks that are escalated by monitoring RUL prognostics. However, due to the fact that the aircraft needs to be grounded the same number of times because of fixed-interval checks, the labor time is dictated by the routine blocks whose composition is determined prior to the scheduling process. Therefore, the airline should not shift to a highly segmented maintenance policy, while the number of escalated tasks is too small and routine maintenance blocks are imposed in the maintenance planning. Nevertheless, if in future scenarios a CBM task-based strategy should be considered, aircraft availability could be significantly improved when the maintenance opportunities are doubled as in the 750FH policy.

A major concern regarding CBM implementation is the substantial monetary investment required from the MRO stakeholders. In order to absorb these costs, it is not possible to rely only on the labor cost savings and the profit generated from extended aircraft availability. The number of labor hours required to execute 186 tasks is not sufficient to justify an investment of €13.2 M or €34.2M to install 22 and 57 sensors, respectively. For an aircraft life cycle of 30 years, the cost absorption is 71.14% in Case 1, while in Case 2 it decreases to 50.58%. Even though the cost is not fully absorbed, two main conclusions are drawn from this analysis. Firstly, not all tasks are worth a CBM investment. In fact, the reference airline should focus on the tasks with shorter intervals because they have a higher impact on the maintenance life cycle. Alternatively, a larger simulation horizon is required to understand the impact of tasks with larger intervals. Secondly, the certification cost could have a lower influence in the case of a larger fleet due to economies of scale.

Lastly, the problem investigated was successfully solved with a Deep Q-Learning algorithm. During the tuning of the neural network the discount factor had a minor influence on the performance of the agent, meaning that the RL policy is close to being myopic because the objective of the airline is always to schedule as late as possible. The potential RL benefits are more noticeable in highly constrained environments with limited resources. Therefore, there are other RL interpretations that could be more meaningful for the AMSP. A task-based approach would generate a far more challenging problem where the objective of the agent would not only be to select a scheduling slot, but to group a combination of CBM tasks and preventive routine tasks subject to capacity constraints. This also entails grouping tasks as the opportunities arise. In this way, the policy gets rid of static block-structures, and would be able to schedule maintenance work-packages more efficiently. The resulting schedule could potentially postpone the maintenance dates instead of grounding the aircraft periodically with a fixed interval policy.

8 Conclusions

This paper presents a Deep Q-Learning (DQL) model to solve the airline maintenance scheduling problem (AMSP) subject to prognostic uncertainty. We explore a wide range of maintenance policies to accommodate a transition to condition-based maintenance (CBM) in airline practices. To evaluate the DQL model, a simulation was performed using real data of a preventive maintenance case from a major European airline. Four main conclusions are drawn from the results of the case study that are key to interpreting the hypotheses addressed in this work.

The DQL methodology can aid maintenance planners to promptly adapt the aircraft schedules to changes in the predicted remaining useful life (RUL). The upcoming month schedule can be updated within 1-5 minutes, depending on the segmentation of the interval policy. Contrarily to deterministic optimization techniques, reinforcement learning (RL) develops a solution in a sequential fashion where the state is evaluated at each step. In the AMSP, the interval due date of CBM tasks is continuously sampled based on a Gaussian propagation model that allows prognostics uncertainty to be taken into account. Furthermore, this work demonstrates that DQL is well-suited to learn a dynamic policy based on the resources available, the maintenance demand, and the RUL prognostics.

Secondly, the implementation of CBM in the preventive maintenance program decreases the required labor hours, while maintaining high levels of reliability. By applying substitution, specific tasks are removed from the aircraft maintenance program (AMP), while interval escalation postpones the due dates of other routine tasks based on RUL predictions. When CBM is implemented in a quarter of the AMP, the combined effect of substitution and interval escalation reduces the overall task executions leading to half of the initial labor workload. Despite the promising results, aircraft availability is increased by only two days per year due to the exclusive focus on the routine A-check program.

The third hypothesis analyzes the impact of prognostics uncertainty on the scheduling performance of the airline. This problem was addressed by increasing the segmentation of the aircraft maintenance cycle, such that there are more grounding opportunities that allow the re-evaluation of prognostics as the due date approaches. The results demonstrate a significant increase in the utilization of tasks monitored with prognostics. Therefore, a segmented maintenance policy will be necessary in future adoptions of CBM strategies in order to manage the uncertainty involved in RUL prognostics. Nevertheless, only a minor improvement is obtained in aircraft availability thanks to policy segmentation as only a reduced portion of the routine tasks is considered in the scope of interval escalation.

Finally, the profitability of CBM strategies has also been evaluated. The investment costs for the certification and the installation of PHM technology cannot be absorbed exclusively with the reduction of labor hours and the profit generated from extended aircraft availability. In order to justify the investment costs, a complete AMP should be tested to address the influence of CBM within the non-routine and unscheduled maintenance events. More importantly, the limited number of aircraft tasks monitored by PHM needs to be selected upon a careful consideration of the labor hours, the interval requirements, and the impact on the non-routine program. Lastly, the fleet size plays an important role in the absorption of fixed costs, such as the certification cost. For future endeavors, we recommend an initial implementation of CBM in a reduced portion of a large fleet, in order to leverage economies of scale when extending the maintenance policy.

In future work, it would be interesting to explore the combined effect of different sources of uncertainty such as RUL prognostics, aircraft daily utilization and maintenance elapsed time. This could be combined with an extension of the maintenance program in order to include non-routine and unscheduled tasks. In this way, a complete cost-benefit analysis could be investigated, and the investment costs could be absorbed by other maintenance events apart from preventive routine tasks. Changing the maintenance strategy to a task-based approach with slot capacity constraints could also be considered to uncover new synergies between tasks and RUL prognostics. This could possibly solve inefficiencies encountered during the preliminary clustering of routine tasks into blocks, as well as unlocking the potential of a predictive strategy by postponing the maintenance dates. Although these implementations would greatly affect the computational time of RL algorithms, they represent extensions to improve the AMSP performance and capture previously unreachable stochastic complexities.

Aknowledgements

This research work is part of ReMAP project, which received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769288. We would like to express our gratitude to the associated ReMAP partners for providing their aircraft maintenance data and validation of the proposed methodology and its practical relevance. For more information please visit https://h2020-remap.eu/

References

Afsar, H. M., Espinouse, M., and Penz, B. (2006). A Two-step Heuristic to Build Flight and Maintenance Planning in a Rolling-horizon. In *2006 International Conference on Service Systems and Service Management*, volume 2, pages 1251–1256.

Daily, J. and Peterson, J. (2017). Predictive Maintenance: How Big Data Analysis Can Improve Maintenance. In Richter, K. and Walther, J., editors, *Supply Chain Integration Challenges in Commercial Aerospace: A Comprehensive Perspective on the Aviation Value Chain*, pages 267–278. Springer International Publishing, Cham. URL.

Deng, Q., Santos, B. F., and Curran, R. (2020). A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization. *European Journal of Operational Research*, 281(2):256–273. URL.

Dong, T., Haftka, R., and Kim, N. (2019). Advantages of Condition-Based Maintenance over Scheduled Maintenance using Structural Health Monitoring System.

Elattar, H. M., Elminir, H. K., and Riad, A. M. (2016). Prognostics: a literature review. *Complex & Intelligent Systems*, 2(2):125–154. URL.

Eltoukhy, A., Chan, F., Chung, S.-H., and Qu, T. (2017). *Scenario-based Stochastic Framework for Operational Aircraft Maintenance Routing Problem.*

Feo, T. A. and Bard, J. F. (1989). Flight Scheduling and Maintenance Base Planning. *Management Science*, 35(12):1415–1432.

Hasselt, H. V., Guez, A., and Silver, D. (2016). Deep Reinforcement Learning with Double Q-Learning. In *AAAI Conference on Artificial Intelligence,*.

Hölzel, N., Schilling, T., and Gollnick, V. (2014). An Aircraft Lifecycle Approach for the Cost-Benefit Analysis of Prognostics and Condition-based Maintenance based on Discrete Event Simulation. In *PHM 2014 - Proceedings of the Annual Conference of the Prognostics and Health Management Society 2014*.

Hölzel, N., Schröder, C., Schilling, T., and Gollnick, V. (2012). A Maintenance Packaging and Scheduling Optimization Method for Future Aircraft. In *6th International Meeting for Aviation Product Support Processes (IMAPP)*. URL.

IATA's Maintenance Cost Technical Group (2019). Airline Maintenance Cost Executive Commentary. Technical report.

Kalweit, G., Huegle, M., Werling, M., and Boedecker, J. (2020). Deep Constrained Q-learning. URL.

Kinnison, H. (2004). *Aviation Maintenance Management*. McGraw-Hill Education. URL.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274. URL.

Lagos, C., Delgado, F., and Klapp, M. A. (2020). Dynamic Optimization for Airline Maintenance Operations. *Transportation Science*, 54:998–1015.

Liang, S., Yang, Z., Jin, F., and Chen, Y. (2020). Data Centers Job Scheduling with Deep Reinforcement Learning. In Lauw, H. W., Wong, R. C.-W., Ntoulas, A., Lim, E.-P., Ng, S.-K., and Pan, S. J., editors, *Advances in Knowledge Discovery and Data Mining*, pages 906–917, Cham. Springer International Publishing.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Muchiri, A. and Smit, K. G. (2009). Application of Maintenance Interval De-Escalation in Base Maintenance Planning Optimization. *Enterprise Risk Management*, 1.

Ozkol, I. and Senturk, C. (2017). The effects of the use of single task-oriented maintenance concept and more accurate letter check alternatives on the reduction of scheduled maintenance downtime of aircraft. In *2017 8th International Conference on Mechanical and Aerospace Engineering (ICMAE)*, pages 67–74.

Pereira, M. A. and Ashok Babu, J. (2016). Information Support Tool for Aircraft Maintenance Task Planning. *International Advanced Research Journal in Science, Engineering and Technology*, 3(2).

Powell, W. B. (2011). *Approximate dynamic programming : solving the curses of dimensionality*. Wiley.

Powell, W. B. and Topaloglu, H. (2006). Approximate Dynamic Programming for Large-Scale Resource Allocation Problems. In *Models, Methods, and Applications for Innovative Decision Making*, pages 123–147. INFORMS.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for Activation Functions. URL.

ReMAP H2020. Website. Accessed on 23/10/2020. URL.

Saltoğlu, R., Humaira, N., and İnalhan, G. (2016). Aircraft Scheduled Airframe Maintenance and Downtime Integrated Cost Model. *Advances in Operations Research*, 2016:2576825. URL.

Sankararaman, S., Daigle, M., Saxena, A., and Goebel, K. (2013). Analytical algorithms to quantify the uncertainty in remaining useful life prediction. In *2013 IEEE Aerospace Conference*, pages 1–11.

Saxena, A., Celaya, J., Balaban, E., Goebel, K., Saha, B., Saha, S., and Schwabacher, M. (2008). Metrics for evaluating performance of prognostic techniques. In *2008 International Conference on Prognostics and Health Management*, pages 1–17.

Shannon, M. and Ackert, P. (2010). Basics of Aircraft Maintenance Programs for Financiers. Evaluation & Insights of Commercial Aircraft Maintenance Programs. Technical report.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359. URL.

Solozabal, R., Ceberio, J., Sanchoyerto, A., Zabala, L., Blanco, B., and Liberal, F. (2020). Virtual Network Function Placement Optimization With Deep Reinforcement Learning. *IEEE Journal on Selected Areas in Communications*, 38(2):292–303.

Sprong, J. P., Jiang, X., and Polinder, H. (2020). Deployment of Prognostics to Optimize Aircraft Maintenance – A Literature Review. *Journal of International Business Research and Marketing*, 5:26–37.

Sun, Y. and Tan, W. (2019). A trust-aware task allocation method using deep q-learning for uncertain mobile crowdsourcing. *Human-centric Computing and Information Sciences*, 9(1).

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, first edition.

Teal, C. and Sorensen, D. (2001). Condition based maintenance [aircraft wiring]. In *20th DASC. 20th Digital Avionics Systems Conference (Cat. No.01CH37219)*, volume 1, pages 1–3.

Tong, Z., Chen, H., Deng, X., Li, K., and Li, K. (2020). A scheduling scheme in the cloud computing environment using deep Q-learning. *Information Sciences*, 512:1170–1191.

Vandawaker, R. M., Jacques, D. R., and Freels, J. (2015). Impact of prognostic uncertainty in system health monitoring. *International Journal of Prognostics and Health Management*, 6.

Vianna, W. O. L. and Yoneyama, T. (2018). Predictive Maintenance Optimization for Aircraft Redundant Systems Subjected to Multiple Wear Profiles. *IEEE Systems Journal*, 12(2):1170–1181.

Vlamings, B., Verhagen, W., and Freeman, F. (2020). Assessing the Impact of Condition-Based Maintenance as a Function of the Variation in Prognostics Performance Levels. Technical report.

Waschneck, B., Reichstaller, A., Belzner, L., Altenmüller, T., Bauernhansl, T., Knapp, A., and Kyek, A. (2018). Optimization of global production scheduling with deep reinforcement learning. *Procedia CIRP*, 72:1264–1269. URL.

Witteman, M., Deng, Q., and Santos, B. F. (2021). A bin packing approach to solve the aircraft maintenance task allocation problem. *European Journal of Operational Research*.

Yan, S., Hsiao, F. Y., Guo, J., and Chen, Y. C. (2011). Effective aircraft maintenance schedule adjustment following incidents. *Transportation Planning and Technology*, 34(8):727–745.

II

Literature Study
previously graded under AE4020

1 Introduction

Throughout the years, aviation safety and operational efficiency have become a major concern to remain competitive in the airline industry. Maintenance is performed regularly in accordance to the international air safety requirements and the aircraft's manufacturer. It is a necessary process closely interconnected with the flight scheduling problem, especially because the aircraft under repair will remain in the hangar unable to fly passengers. Ten percent of flight delays and cancellations are currently caused by unscheduled maintenance events, costing the global airline industry an estimated 8 billion USD. For U.S. airlines alone, this translates to over 60,000 delays and cancellations a year that could be prevented with a more efficient maintenance planning program (Daily and Peterson, 2017).

There are three main categories of airline maintenance that can be distinguished: long-term, mid-term and short-term (Yan et al., 2011). The first one includes heavy maintenance tasks that require an overhaul longer than 10 days, they are also called level C and D checks. Mid-term maintenance plans are set monthly on the long-term plan. They incorporate both level A and B checks. Lastly, short-term plans produce schedule adjustments due to incidents. It is noteworthy that these checks are all performed in the hangar facility. Nonetheless, line maintenance is an additional category performed at the gate of the airport. There is not a unique and systematic approach to perform maintenance, and every airline has different guidelines to solve the problem. In fact, occasionally, different category checks are merged based on the slot opportunities and the aircraft's operational condition.

To date, the airline planning process is primarily manually driven due to the difficulty in solving the re-scheduling airline models. In general, there is still a big gap between the reality faced by airlines and the resolution that has been offered by an optimization-based environment (Clausen et al., 2010). Nevertheless, the manual planning is becoming unpractical due to a more dynamic environment in which both costs and the complexity of the airplanes keep rising (Van Den Bergh et al., 2013). The aircraft data volumes are growing exponentially in the airline industry, mainly because of the new sensors that could be used to provide an early signal of performance. The new aircraft generations (e.g. Airbus A350) produce 50 times more data than the older ones (e.g. Airbus A320), but organizations still struggle to apply this flood of information in their operations optimization (Daily and Peterson, 2017).

Condition-based maintenance is a strategy designed to transform real-time aircraft data into actionable intelligence to avoid unnecessary ground times and replacing periodic-based strategies. The approach to realize the digital future of the aviation maintenance is twofold. First, it is essential to deploy the use of advanced analytics to understand the driving factors of future performance, monitor the individual aircraft parts and predict remaining useful life (RUL). The second element revolves around how to plan maintenance and optimize schedules with a flexible framework enabled through the power of predictive analytics. The ultimate objective of this research project is focused on the latter, and can be stated as follows:

Develop a maintenance scheduling methodology with an approximate dynamic programming approach to enable condition-based maintenance in a fast and adaptive manner, and to deliver a near-optimal maintenance schedule.

The temporal succession of the maintenance operations gives the problem a dynamic nature. At each step, the available slots change based on the previous decision, constraining further the solution space for the future. Therefore, sequential optimization is the approach chosen, while being aware of the future availability constraints. Through approximate estimations of the fleet condition and predictive analytics, it is possible to

create such a model. To cope with the nature of the environment and the ever-changing operations, approximate dynamic programming is proposed as a valid methodology to optimize the maintenance scheduling process.

1.1. Research framework

The maintenance schedule indicates the slot opportunities available to the fleet and how to allocate a series of tasks. There are two different problems to be identified. The first one, also called task-packaging, consist on grouping tasks in blocks according to their periodicity. The second one is to assign each block of tasks to a maintenance opportunity. Additionally, non-routine maintenance tasks are incorporated to the existing blocks based on predictions, together with the reactive maintenance tasks, detected during inspections. Therefore, the challenge consists on allocating the correct tasks on each block and assigning them to a maintenance opportunity. Further extensions should include the additional challenge of stochastic programming with elements such as the maintenance elapsed time and the aircraft utilization, that will directly affect the task due date. Additional line maintenance opportunities, which are only known in the operational horizon, can also be used to extend the scope of this research.

1.1.1. Scope definition

This section is used to outline the questions that this research work aims to answer. They do not establish the plan to reach the ultimate goal, but highlight the steps that are considered relevant to expand in the first phase of this research project. The main research question to be answered in order to fulfill the research objective is the following:

What approximate dynamic programming approach can enable adaptive maintenance task allocation, and how should the task-packaging be connected with the maintenance scheduling problem?

In order to achieve the objective and answer the main research question, the following four core sub-questions have been identified and broken down further:

1. What are the current practices in airline maintenance scheduling (AMS)?
   (a) What are the airworthiness requirements related to maintenance?
   (b) What is the economic impact of maintenance in the airline industry?
   (c) Can maintenance scheduling be divided in multiple phases?
   (d) What are the task features considered when creating maintenance work packages?
   (e) Which simplifications to the AMS problem are acceptable within the industry practice?
2. How can a CBM context be exploited during maintenance task packaging and allocation?
   (a) Can a CBM strategy be applied while keeping a traditional block check structure?
   (b) Is it possible to create smaller work-packages instead of fixed periodic blocks?
   (c) How does the size of the maintenance work-packages affect the schedule optimization?
   (d) How much access synergy can be obtained by packaging tasks prior to scheduling?
   (e) Does the condition monitoring technology exist to allow a transition to a fully automatic and digital maintenance planning tool?
3. What are the current models for airline maintenance scheduling?
   (a) What models have already been implemented in the airline industry?
   (b) Are maintenance task packaging and allocation considered in the optimization models?
   (c) What are the limitations of the current strategies?
   (d) What assumptions can be made to the maintenance scheduling model?
   (e) What industries face similar problems?

4. Can approximate optimization techniques produce near-optimal maintenance schedules?
    (a) What approximation models can be used?
    (b) How can historical task allocation data be used to structure the optimization algorithm?
    (c) What are the block and task features that provide better results?
    (d) How should the quality-time trade-off be assessed during the validation phase?
    (e) What are the most relevant key performance indicators to measure the AMS performance?
    (f) How can the robustness of the framework be quantified and tested?
    (g) What validation models can be used to validate the scheduling algorithm?
    (h) What are the limitations of the proposed scheduling method?

1.2. Outline of the report

The report is organized in the following fashion, such that all the research questions are investigated in the order presented. Chapter 2 gives an overview of the maintenance planning process and a historical perspective of the maintenance practices. In Chapter 3, the CBM challenge is presented to construct an adaptive schedule with a large planning horizon such that when the aircraft condition indicators enter the hazard threshold, the maintenance checks are scheduled accordingly. An overview of the state of the art scheduling models and solution techniques is discussed in Chapter 4. The limitations and the gap of literature regarding the planning horizon and the scope of this project is also presented here. Motivated by the problem's computational intractability, the methods of Approximate Dynamic Programming and Reinforcement Learning are introduced in Chapter 5. Their potential application could solve scheduling problems of stochastic complexity previously unreachable. Finally, Chapter 6 concludes the literature review with an outlook on the future research project.

2
Maintenance Planning

The development of an airline maintenance schedule (AMS) is a complicated process that considers economic, legal, and technical factors. The ultimate goal is to achieve a flight schedule that is compatible with airworthiness regulations and the airline's policies. The AMS problem deals with the construction of a schedule that minimizes maintenance costs, disruption to airline operations, and resource usage, while satisfying safety regulations imposed by civil authorities (Sriram and Haghani, 2003). Among others, maintenance resources include hangar facilities geographically dispersed through the network, limited inventory tools, specialized technicians, and spare parts (Sanchez et al., 2020). The main challenge is to allocate these resources in a cost-effective fashion. The first research question regarding *the current practices in airline maintenance scheduling* is tackled in this chapter.

2.1. Economic impact of airline maintenance

According to a recent study by Lagos et al. (2020), maintenance operations correspond to 10% of the airline's direct operating cost (DOC). This represents a 12.5 billion USD industry considering only the 10 largest US airlines. Within this cost, 50% of the expenses are attributed to wages of the highly skilled workforce. For this reason it is possible to relate the amount of workhours on a task with its cost, as Lagos et al. (2020) and Witteman (2019) suggest. Apart from the cost of the maintenance activity, there is an additional loss of revenue due to network disruptions caused by unexpected maintenance events. A study by Cook et al. (2004) has shown that the annual budget spent in delay due to maintenance amounts to 40% of the total maintenance burden. Finally, there are additional soft costs due to passenger dissatisfaction that may lead to a significant loss of the market share in future operations.

The report made by IATA's Maintenance Cost Tasks Force (2018) presents a rough estimate of $ 324M that the average airline sustained on maintenance operations during the 2017 Financial Year (FY). This corresponds to an average of $ 3.6M per aircraft or $ 1,200 per flying hour. According to General Electric (Daily and Peterson, 2017), in 2011 commercial jet airplanes were in the air for 50 million hours. This translates into a $ 60 billion annual maintenance bill. Engine maintenance alone accounts for 43% of the total, or 25 billion. This means that commercial jet engine maintenance costs can be reduced by 250 million for every 1% improvement in engine maintenance efficiency due to condition-based maintenance. McFadden and Worrells (2012) estimated that by 2020 the global Maintenance, Repair and Overhaul (MRO) spending would reach $ 65 billion. Even a minimal improvement in the operational efficiency would lead to large savings, and a higher aircraft utilization could be achieved to earn extra revenue. It becomes clear that anyone involved in the airline business cannot afford to ignore the potential benefit from optimizing the maintenance planning process.

2.2. Maintenance Program Development

The current approach to develop a maintenance plan in the aviation industry stems from the latest Maintenance Steering Group (MSG-3) program developed by the Air Transport Association of America in 1980 (Pontecorvo, 1984). The main characteristic of this technique is the "top down" approach to conduct failure analysis at the highest manageable level rather than at the component level like in the previous versions. The MSG-3 logic assesses how the failure affects operations, mainly looking at tasks done for safety reasons and for economic aspects (Shannon and Ackert, 2010). It is also renowned as a *task-oriented* approach, as it develops a set of specific tasks selected for a given functional failure. Therefore, it groups tasks in a way that is

more efficient for the airlines. It matches the check type with the operational requirement, rather than carrying out a maintenance process based on the analysis of each aircraft unit (Kinnison, 2004). There are three main categories that can be highlighted within the tasks developed by the MSG-3: airframe systems tasks, structural item tasks and zonal tasks.

The tasks selected in the MSG are published by the manufacturer in a document approved by the airworthiness authorities: the maintenance review board (MRB) report. This document outlines the minimum scheduled maintenance requirements to develop a program for the operators. Additionally, the manufacturer also releases the MPD that contains all the MRB requirements plus maintenance task information to aid in planning such as grouping by letter check, access panel, man-hour requirement, interval, etc.

2.2.1. Maintenance task categories

Maintenance task can be classified as scheduled or unscheduled tasks (Pereira and Ashok Babu, 2016). The former refers to maintenance activities that occur periodically in intervals defined by the authorities. In Kinnison (2004) they are also defined as *routine tasks*, which are basically the ones specified in the MRB and MPD documents. Another category are the *variable routine* tasks, which are those that vary from check to check. These tasks include deferred items from previous checks, airworthiness directives and one-off maintenance actions. In general, the time required to accomplish these tasks is known, so these items are similar to the routine tasks for planning purposes.

A considerable challenge for the airline industry is to design a schedule that includes *non-routine* or unscheduled tasks. This category encloses items that arise out of inspections, as well as faults reported by pilots, systems that are maintained reactively, i.e. only after the failure occurs, and other events (e.g. bird strikes). Aircraft are allowed to carry forward a limited amount of unscheduled maintenance while remaining serviceable. Even if some failures do not affect the immediate continuation of operations, others are included in the minimum equipment list (MEL). The MEL is based on the aircraft equipment, level of redundancy and systems. In case of an unexpected failure, the aircraft can be dispatched only if it meets the MEL-conditions. Otherwise, it must be grounded for safety reasons (Obadimu et al., 2020). It is only possible to estimate the number of non-routine tasks and their duration depends on multiple factors. Therefore, a buffer time is considered when planning routine checks that accounts for non-routine tasks. In most cases, it even doubles or even triplicates the expected duration of the scheduled check.

2.3. Maintenance Checks

On average, a normal aircraft receives around 12 hours of line maintenance per week (Qantas, 2016). This happens "around the world and around the clock", whenever a transit check is performed. Examples of tasks performed at the apron are inspecting the wheels, the brakes and fluids in the hydraulic systems, as well as checking when the sensors alert the need of running repairs.

Every eight to ten weeks the filters are replaced and thorough inspections are carried during the so called A-checks. On a medium-haul passenger aircraft, these checks can take up to a full day. The B-checks are similar but involve different tasks that consider moving parts, such as the ailerons and horizontal stabilizers, as well as the lubrication of key systems (Sriram and Haghani, 2003). Kinnison (2004) mentions that some airlines incorporate these tasks directly into the line maintenance. They are also known by the expression "less-than-A-check tasks". The scheduling method is chosen by the airline, however, tasks easily get deferred day after day due to demanding flight schedules and the aircraft risks to be grounded in case of exceeding utilization limits.

The C-checks includes mostly heavy-maintenance tasks lasting approximately three weeks. The frequency of these tasks is once every two years or every 2,500-3,000 flight hours. Moreover, the total workforce required varies between 2,000-4,000 man hours. The interior of the aircraft is dismantled to closely analyze all the systems and any hidden structural damage. A typical C-check task is the upgrade of the cabin seats and interiors.

Finally the D-check is the last and most comprehensive activity. During this check, the entire plane is dissected and, after maintenance, each system is put back together. It can take up to 6 weeks, and for this reason it is performed only once every six to twelve years, for the newer generation of aircraft. This check involves the analysis of the aircraft skin and the complete removal of the landing gear, among other systems. The cost can

Table 2.1: Airline Maintenance Check Schedule Example (Kinnison, 2004)

	747-400	747-200/300	DC-10-30	A300B4	F50
A check	Every 600 FH	Every 500 FH or 7 weeks	Every 465 FH or 9 weeks In 3 parts (A1,A2,A3)	Every 385 FH or 11 weeks In 4 parts (A1,A2,A3,A4)	Every 650 FH or 4 months
B check	Every 1200 FH In 2 parts (B1,B2)	Every 1000 FH In 2 parts (B1,B2)	None	None	Every 1300 FH or 8 months
C check	Every 5000 FH or 18 months In 2 parts (C1,C2)	Every 4650 FH or 24 months	Every 4500 FH or 20 months In 2 parts (C1,C2)	Every 3000 FH or 18 months In 2 parts (C1,C2)	Every 4000 FH or 25 months In 2 parts (C1,C2)
D/HMV check	Every 25K FH or 6 years	Every 20K FH or 5 years	Every 20K FH or 6 years	Every 12K FH or 4 years	Every 12K FH or 6 years In 2 parts (H1,H2)

sum up to several million and an average of 10,000-50,000 labor hours, but the aircraft comes out completely refurbished.

Engine overhauls are considered heavy-maintenance. Depending on the engine type, the cost oscillates between $ 500,000 and $ 5M every 4,500-24,000 flight hours. They represent the most expensive segment of the airline maintenance activities (IATA's Maintenance Cost Tasks Force, 2018). McFadden and Worrells (2012) states that after the deregulation act of airlines (1978), due to the fierce competition, outsourcing of the maintenance activities has been a viable and attractive option for many airlines . New leasing and buying contracts are frequent, where the manufacturer takes responsibility of the engine checks, reducing the in-house workshops.

These indicative ciphers give an estimation of what airlines invest in maintenance. The actual spending, however, varies per airline depending on fleet type, aircraft utilization, outsourcing of the activities, etc. Table 2.1 summarizes airworthiness intervals related with the checks that have been discussed above. It is noteworthy that nowadays most airlines only have A and C checks categories. The other tasks have been distributed over these blocks.

2.3.1. Block and Phased Checks

Nowadays, it is more common to divide these higher checks into segmented clusters or phases. The main reasons are to reduce aircraft downtime, to have a flexible grouping of tasks, and to optimize the schedule. On the other hand, the planning complexity is highly increased, the slot capacity may represent a risk when considering non-routine tasks.

Traditional maintenance strategies opt to cluster a series of tasks in blocks based on the resources required (A,B,C,D checks). This methodology has declined in recent years, in favor of progressive maintenance checking. Modern maintenance planning approaches adopt a task-driven solution, where tasks are bundled together, so that the resulting check can be performed overnight while the aircraft is not in use (Muchiri and Smit, 2009). Rather than having long aircraft downtimes and sporadic manpower requirements as in block checks, phased checks increase the aircraft availability and introduce flexibility in the way tasks are grouped. In light of these recent advancements, Papakostas et al. (2010) adheres to this methodology by developing a line maintenance decision support that schedules tasks sequentially per each aircraft in the fleet.

Phased checks identify smaller packages within the already existing ones in order to perform the tasks with more frequency but requiring shorter maintenance event duration. For instance, a typical C check that is performed once a year may be broken down in 4 elements (C1,C2,C3,C4) executed every 3 months or in 12 parts (C1,C2,...,C12), one performed each month (Kinnison, 2004). On the other hand, the scheduling becomes more cumbersome with the phased checks as there are shorter work-packages, but they become more numerous. Finally, it is noteworthy that not all checks can be carried out progressively. For the heavy maintenance tasks (D checks) the aircraft must still be taken apart or undergo a major overhaul (Ruther et al., 2016). In 1988, after the Aloha Airlines Flight 243 incident, the US National Transportation Safety Board (NTSB) concluded that a *highly segmented* structural inspection precludes a comprehensive assessment of the aircraft's operating condition (Shannon and Ackert, 2010).

2.4. Maintenance Strategies

Fossier and Robic (2017) recognize three categories of airline maintenance strategies. In order to give a broader overview, the approach presented is combined with the work of Tinga (2013) that includes a much more exhaustive maintenance classification. The maintenance categories scheme adapted to the airline industry can be seen in Figure C.3. The first category is a completely **reactive** methodology that replaces a system after the fault has been detected. This strategy is mainly used for non-critical system failures that do no restrict aircraft operations. The advantage of a reactive maintenance policy is that the service lifetime of parts and components is fully utilized, thus no remaining lifetime is spoiled.

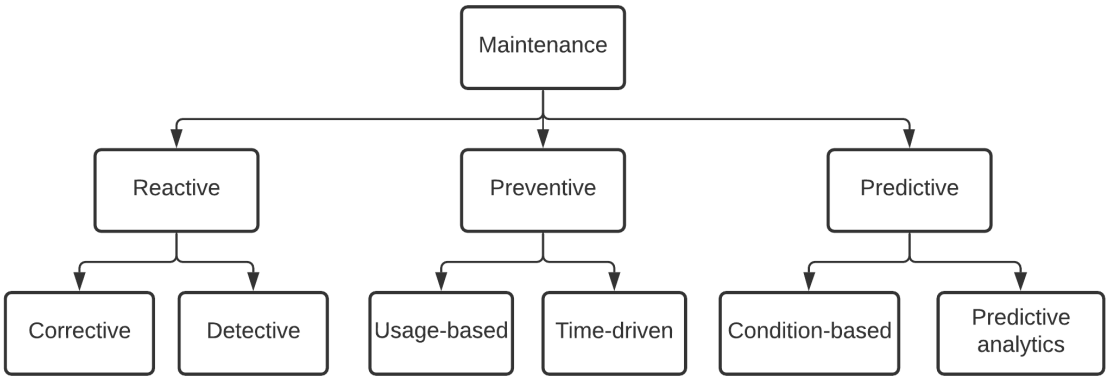


Figure 2.1: Overview and classification of maintenance policies. Adapted from Tinga (2013)

Preventive maintenance strives to repair a system before the failure occurs. The obvious disadvantage is that the optimal time to perform a task is hard to determine. It is possible to subdivide this category in static time-driven intervals, e.g. calendar based (DY); or usage-based intervals, e.g. flight hours (FH) and flight cycles (FC). It is common practice to combine preventive maintenance with opportunistic policies to benefit from clustering tasks and economies of scale while scheduling.

Lastly, **predictive** strategies have the objective of performing maintenance only when necessary. It is a great approach to reduce the frequency of preventive maintenance and the rate of failures leading to reactive maintenance. Thus, improving system availability and reducing maintenance costs. Predictive maintenance analyzes metrics, mainly sensor data, that characterize the real-time health of the aircraft element (condition-based), as well as predicting when it is more likely to fail in the future (predictive analytics). There is a vast overlap between the terms predictive and condition-based maintenance, sometimes they are used as synonyms in the industry; others condition-based is intended only as the real-time monitoring, while predictive includes the outlook over the future lifetime of the aircraft system. In this review the distinction is not of high relevance. The main interest lies on the impact of scheduling when predicting the maintenance due date by monitoring the aircraft condition.

2.5. Disruption Management

In general, aircraft recovery models have a partial view of maintenance operations when solving disruptions. The aircraft routing problem determines the sequence of flights covered by each aircraft in the fleet. This problem is limited to keeping the maintenance constraints intact during the time window considered. Due to the sequential approach in the airline operation control center, a maintenance plan is firstly developed and the flight routings are built accordingly. Barnhart and Smith (2012) states that the fact that maintenance schedule is regarded as a hard constraint, restricts other potential valuable solutions such as:

- Swapping maintenance of aircraft to a new overnight location.
- Swapping maintenance from an aircraft where it can be legally postponed to a maintenance-critical aircraft.

- Prematurely executing maintenance to assign future opportunities to maintenance-critical aircraft.

Shen and Yao (2015) stresses the need of a dynamic rescheduling system in flexible flow shop problems, especially to have the ability of coping with disruptions in schedule planning. In fact, it would not only help to allocate tasks, it would also create robust schedules that are able to account for the uncertainty of non-routine tasks. In this way the chances of having disruptions are decreased when it regards maintenance factors. Another valuable point that is made in Barnhart et al. (2003), is the fact that airline operations optimization focuses a lot on increasing aircraft utilization and reducing slack time as much as possible. Although in theory it is economically beneficial, in practice it can translate into less robustness and increased costs. To address this issue researchers have begun to investigate the paradigm of *realized and unplanned* costs. Among others, Chiraphadhanakul and Barnhart (2013) and Lan et al. (2006) stand out for designing robust flight schedules by introducing slack time judiciously in order to hinder the effect of possible disruptions.

2.6. Discussion

In this chapter a general study of the maintenance planning in the industry has been presented, along with the benefits and drawbacks of different maintenance strategies. The conclusion of the economic impact of airline maintenance points to possible savings in the order of hundreds of millions of dollars a year for a small improvement in maintenance plans. More importantly, it can prevent disruptions due to unscheduled maintenance that costs 8 billion USD a year to the global airline industry. One of the main takeaways is that maintenance practices are conservative and strict in order to ensure air safety requirements. Most checks are performed in blocks of tasks, triggered by opportunistic policies that want to avoid groundings as much as possible. On the other hand, phased blocks dissect traditional checks into smaller work-packages that require the aircraft to be grounded more often but the systems can be utilized more efficiently. To foster these strategies, there is a general trend in research to shift from preventive to predictive maintenance in order to reduce costs while keeping high levels of reliability.

Maintenance plans are developed in different stages based on their category. The heavy checks (C and D) are planned in the long term and they are decoupled from the other shorter-term plans. It is important to strategically position these checks in the aircraft timeline to avoid an overhaul during a high peak period or the summer. Mid-term checks (A and B) are more flexible and can be scheduled less than a month in advanced. If the slot availability and the aircraft condition are favorable, they can be merged with higher hierarchy checks. Lastly, line maintenance is planned with a week in advanced concurrently with short-term plans.

3
Condition-Based Maintenance

The transition to condition-based maintenance (CBM) combines preventive and predictive strategies that aim to reduce maintenance costs, yet providing services with an improved quality and reliability. *Preventive maintenance* strategies are used nowadays to execute recurrent maintenance tasks, and will remain necessary for an effective transition. It both serves as a fail-safe for predictive maintenance and to globally assess the status of a system in case of false-positive predictions. Furthermore, *predictive maintenance* can reduce disruptions through condition-based information. Maintenance can be performed only when necessary and it becomes possible to identify damage before failure occurs. The prognostic and health monitoring systems are employed to decide when is the most optimal time to allocate the task and avoid waste of remaining useful life (RUL). This chapter revolves around the second research question that studies the *disruptive impact of CBM in the airline industry*.

3.1. Prognostic and Health Management (PHM)

The integration of PHM techniques with maintenance decision represents the basis of the CBM concept (Vianna and Yoneyama, 2018). Hölzel et al. (2012b) present the disruptive potential of PHM systems to reduce both, *operational interruptions* caused by unscheduled maintenance tasks and *downtimes* due to unnecessary preventive maintenance. Recurrent inspections that could be monitored with sensor data are set to become an obsolete practice in the near future. In theory, a drastic reduction of maintenance events could be reached in future operations. Holzel alleges that "in the ideal case - which probably will never occur - there will be no more scheduled maintenance programs. This means a complete shift from preventive to a condition- or prognostic based maintenance strategy". Significant advances have been reached from sensor integration to scheduling optimization, but several challenges are yet to be solved. In order to initiate a transition phase, the deployment of on-board monitoring systems is of paramount importance.

Literature suggests that only a half of the total work and parts required by heavy-maintenance checks can be planned (Kulkarni et al., 2017). The remaining half remains unscheduled, and is identified during the inspections. Hence, any of the maintenance tasks may cause a potential delay if unexpected requirements come out of inspections. The expected completion time for the same check may vary substantially from one case to another, and the inherently unpredictable nature of these events can also have wider implications on budgeting, inventory management, and maintenance capacity planning (Samaranayake and Kiridena, 2012). As part of a risk management tool, a safety buffer is always considered in the planning of maintenance, even though it decreases efficiency. Moreover, the specialized workforce is highly paid, thus idle time should be kept as low as possible to avoid cost spillage. Clearly, this represents an additional incentive to foster the development of CBM, such that the scheduling can be able to include tasks before the system fails.

3.2. Task-oriented Strategy

Condition-based maintenance can be considered a task-oriented strategy because it monitors individual tasks based on the aircraft condition. Therefore, CBM strategies can be best applied with a phased check structure. In this way a more flexible framework to schedule tasks close to their due date is provided. As explained before, the block check is a large list of tasks, that are grouped together and repeated periodically. The innovation of phased checks is that it gets rid of static templates and the individual tasks are assigned to the most optimal maintenance opportunity. Hölzel et al. (2012a) proposes an aircraft life cycle cost-benefit model that follows this task-oriented approach. The validity of the maintenance schedule is constrained by

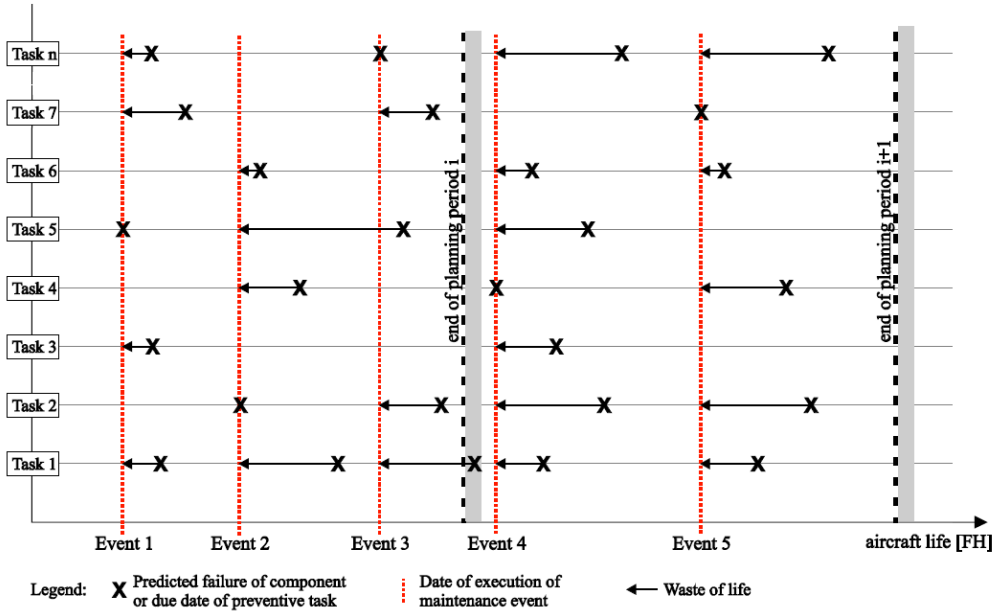


Figure 3.1: CBM Maintenance Planning (Hölzel et al., 2014)

the available labor force, the availability of slots and the certification of the hangars. Such a problem is classified as NP-hard with a bin-packing structure. In order to solve the problem in polynomial time, a common approach is to employ heuristic depth-first search.

In a subsequent publication (Hölzel et al., 2014), the authors depicted graphically the holistic approach, visible in Figure 3.1. The planning horizon is divided in time windows in order to reduce the size of the problem. The activities included in a certain time window must be scheduled in the maintenance events of that planning period. Based on the predicted failure dates, the model assigns tasks minimizing the overall cost, as a result the waste of life of each task is also minimized. Moreover, the authors test a series of heuristics to allocate first the tasks with higher priority. Interestingly, the model showed that the best results were achieved when sorting the tasks by cost in descending order. Finally, even if at first it seems logic to perform the task as late as possible, in some cases the costs of using an additional maintenance opportunity, including the loss of revenue due to aircraft unavailability, are higher than the cost associated with the wasted RUL of a system.

A study by Ozkol and Senturk (2017) highlights the economic benefits of having a flexible single task-oriented maintenance concept over a classical rigid letter check system. The main goal is to increase the aircraft utilization by reducing the scheduled maintenance downtime, based on the idea that any ground time can be seen as an opportunity to perform maintenance. The paper benchmarks their policy against a classic maintenance strategy on a A340 fleet considering A, C and D maintenance tasks over a planning period of 10 years. The results show cost savings above 4 million USD and an increased aircraft utilization by 72 days.

3.3. Task-Packaging

A standard maintenance plan includes more than two thousands maintenance tasks which need to be executed for each aircraft in the fleet. A series of criterion that can be used to examine individual tasks and their packaging is reported below (Ozkol and Senturk, 2017).

- Maintenance task source
- Man-hour requirement
- Skill code and licensed personnel requirement (mechanic, avionics, etc.)
- Relationship of the maintenance task with other tasks
- Planning requirements of task card

- Reference status of maintenance task
- Inventory items required for the completion of the task, i.e. materials and tools

Pereira and Ashok Babu (2016) identifies some factors of the maintenance packaging strategies used nowadays. The location of the task, is very sensitive because if the same panel has to be open for multiple systems, it is likely that the task related to those (sub)systems will be placed in the same work-package. The paper uses the access panel as a design criteria for the tasks that can be performed individually or in combination with other. The opening of the access panel is time consuming and the minimization of it's access reduces wear and tear of the panels and the fasteners that bind it to the aircraft surface.

It is of paramount importance to create task work-packages according to the slot's capacity in order use resources efficiently. Moreover, the skill set of the engineers increases every time the same task is performed, which increases time efficiency. Therefore, it is logical to have the same check scheduled in sequence (on different aircraft) and assume that progressively its execution time speeds up with respect to the previous one. The engineers learning curve effect can be an interesting topic to study in order to understand how this factor can be included in the scheduling process.

3.4. EC-H2020: ReMAP project

The European Union project ReMAP H2020 aims to reduce maintenance cost by reducing interval-based tasks such as standard routine inspections, operational and functional checks. Moreover, when possible task escalation is considered in order to postpone tasks and fully utilize the RUL of the maintained systems. Finally, failure prevention is achieved with predictive maintenance and a proactive scheduling. In the remaining of this review, the focus will be shifted to the scheduling optimization that enable condition-based maintenance policies.
In order to test the validity of the CBM strategies, the project considers dozens of systems that are not maintenance-critical. The concept is to demonstrate the reliability and the possibility of implementing CBM prognostic and scheduling for most aircraft system . Even though the technology is not ready yet, as there are no specific sensors that monitor the health of all aircraft systems, it is possible to elaborate a proof of concept. The simulation, verification and validation can be achieved with the diagnostics and prognostics of non-critical systems, currently maintained reactively. The aim is to initiate a transition from fixed intervals and recurrent inspections, to a dynamic and adaptive maintenance plan that is continuously being monitored by PHM sensors and a data-driven technology.

3.5. Discussion

The long-term cost-benefit analysis of CBM is one of the most attractive aspects of this strategy. It is set to be the future of airline maintenance because of the digital transition it supposes, and the advantages of a data-driven technology. The sensors ought to detect damages in real-time that might not be visible to a human operator, enabling the damage assessment to be performed as frequently as needed. To understand the true advantage of CBM, a comprehensive maintenance plan must be developed to capture the inter-dependencies within a fleet. The economic benefits per aircraft can be determined only when a new maintenance plan can be elaborated and the incoming streams of data are employed efficiently in the schedule optimization . This is because several aircraft compete for limited maintenance resources, leading to less efficient solutions than performing maintenance as soon as a threshold is triggered.

At the same time, CBM strategies requires the aircraft to be completely wired and equipped with sensors to monitor the real-time health of the fleet. To date, various airline operators have implemented certain aspects of CBM in their operations, that have dramatically increased the level of electronic integration in aircraft's systems (Teal and Sorensen, 2001). However, none have taken full advantage of the CBM concept because of the lack of fully equipped aircraft with health monitoring sensors, the strict air safety requirements, and the lack of digitalization in the maintenance operations domain. Despite the initial capital investments required to build the foundations of CBM, some hybrid approaches have been proposed in the research community where traditional scheduled maintenance is used for critical structures and condition-based maintenance for non-critical systems (Dong et al., 2019, ReMAP H2020).

4
The Aircraft Maintenance Problem

The airline scheduling problem contains several stages with different planning horizons. The first one is the flight schedule design, followed by the fleet assignment which selects a fleet type to match the forecast demand with the capacity offered in every leg. Furthermore, during the tactical stage the crew scheduling problem and the aircraft rotation problem (ARP) are solved. The latter is aslso known as the aircraft maintenance routing because generates sequences of flights that are called routings or line of flights (LOF's) (Kabbani and Patty, 1992). The scheduling concludes with the tail assignment (TA) in the operational stage, when the LOF's are assigned to a specific aircraft. Frequently, the ARP and TA problems are solved in combination with maintenance considerations, as indicated in Figure 4.1, that shows the typical stages of airline scheduling problem. In this chapter, the answer to the third sub question, about *what are the current models for maintenance scheduling*, is elaborated and a complete discussion is presented at the end.

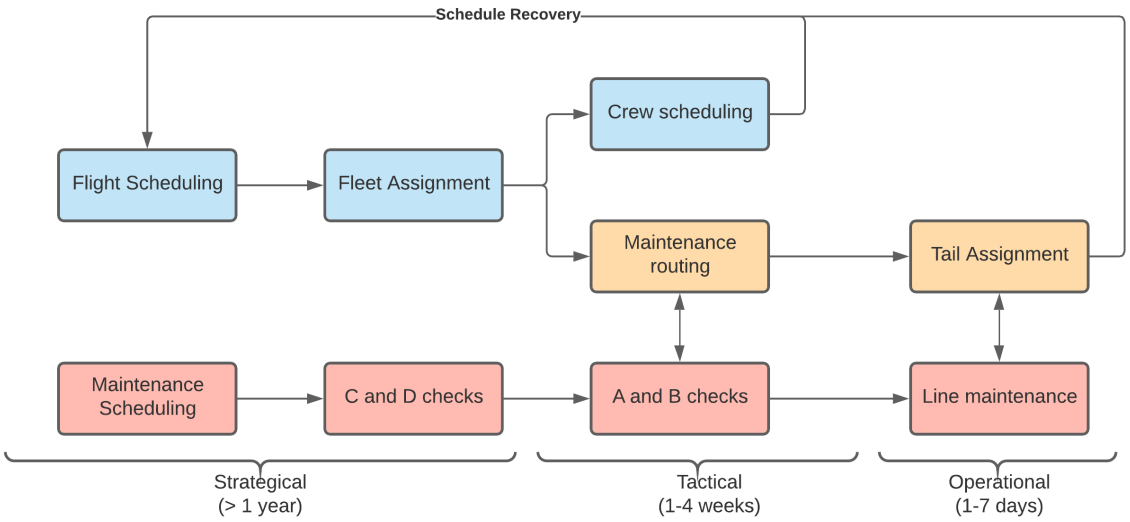


Figure 4.1: Stages of the airline scheduling problem. Adapted from Lagos et al. (2020)

4.1. Aircraft Maintenance Routing

Feo and Bard (1989) propose a model to set-up a network with the minimum number of A-check maintenance base locations and respect a 4-day maintenance interval. The formulation is a closed-loop network that tries to minimize cost. The multicommodity flow problem with integer restrictions is too large to be solved with linear programming. One possibility is relaxing the integrality constraint, but the result would be of marginal value. For this reason, the authors opt for a two-phase heuristc. Firstly, the problem is decomposed by fleet type because there are very few economies of scale factors associated with running a maintenance base for heterogeneous fleets. In fact the main cost is attributed solely to the inventory. The second simplification is the elimination of the space and labor capacity constraints that is rarely binding.

Hane et al. (1995) formulates a daily fleet assignment considering up to eleven fleet and 2500 flight legs with a time-space network. Their work is extended by Clarke et al. (1996) with maintenance and crew considerations. The mixed integer program (MIP) is solved in a similar fashion using a combination of dual-steepest edge simplex and a branch and bound method. Gopalan and Talluri (1998) study the aircraft maintenance routing problem to introduce small transit checks every three days and balance check, that have considerable leeway in scheduling frequency. The model developed is referred under the term "static infinite horizon" to indicate that the solution is a line of flight that should repeat consistently in the schedule. The network is constructed such that it only contains overnight maintenance stations and fixed line-of-flights (LOF). Then, they solve the maintenance routing by swapping flights or even fleet, mainly respecting the three-day maintenance requirement to form a feasible solution.

Clarke et al. (1997) formulates the aircraft rotation problem with specified locations, durations and intervals of line maintenance checks (less than 5 hours long). The solution methodology combines Langrangian Relaxation and subgradient optimization. This approach may yield to feasible, but not provably optimal, solutions. In fact in Martin (1999) this method is described to be useful to calculate the upper and lower bounds of a MIP, specially in minimization problems where dropping constraints can only lead to a smaller objective value.

Sarac et al. (2006) solves the operational aircraft maintenance routing problem with a time horizon of just one day. Even though the planning problem that is being researched in this thesis is very different, this study has been included because it considers the flying hours in the objective function, which is a major concern in long-term maintenance planning. Thus, it is one of the few works that aims to maximize aircraft utilization subject to the resource constraints (available man-hours and maintenance slots). The route generation is prioritized per aircraft in ascending order of their legal remaining flying hours. The author uses an innovative branch-and-price algorithm formulated with a route-based network including flight and aircraft coverage constraints, as well as the labor force hours and the slot availability.

Around the same time of the previous publication, Afsar et al. (2006) proposes a rolling horizon heuristic to solve the maintenance planning problem for an airline with 200 aircraft. The objective is to assign flights to aircraft while still respecting the maintenance plan. The planning is produced for a 40-weeks period with a sliding time window of one week. The heuristic first addresses critical aircraft that must undergo maintenance in the planning week and subsequently manages the non-critical aircraft load smoothing with simulated annealing (Afsar et al., 2009). Contrarily to other works in the field of the aircraft routing problem, the minimization of flight assignment cost is not considered in this model but the maximization of the aircraft utilization. The strong suit of this approach is the speed of the algorithm which is able to provide the flight planning of a week instance with 2500 flight under 7 minutes. However, the critical aircraft number is fixed rather than being dynamically labeled in function of the utilization.

Aircraft maintenance routing (AMR) is closely related to the tail assignment (TA) problem. For this reason, Liang et al. (2015) combine both problems to produce robust schedules and minimize expected propagated delays. The robustness is achieved by inserting buffer time in the routings that are more likely to be disrupted. The TA is treated as an extension of the AMR, where each aircraft is placed in a *subnetwork*, which it is understood as a multi-commodity flow problem formulation. The solution approach proposes a two-stage column generation algorithm: the master problem is a MIP solved with CPLEX while the pricing (sub)problems are tackled with a heuristic. The (sub)problems are used to create line of flights with expected delay propagation. The heuristic chooses a (delayed) flight option such that the LOF's follow an optimal buffer allocation pattern based on minimal propagation delay studies.

Safaei and Jardine (2018) designed a multi-commodity network flow model with generalized maintenance constraints that ensure that enough maintenance opportunities are present within the aircraft routes. The author claims to solve the maintenance routing problem considering the full range of maintenance requirements of individual aircraft over a planning horizon of one week. The model minimizes the maintenance misalignment between the opportunities and the workload due for each aircraft. The solution is produced with LINGO 14.0 which uses a branch-and-bound method to solve a MIP.

In Sanchez et al. (2020) the maintenance planning approach for a 30-day schedule is split into three steps. The first one consist in defining the maintenance opportunities (MO's) when the aircraft is on ground, if possible by large turn-around-times (TAT), otherwise through dedicated ground times. If the maintenance opportunities do not generate a feasible schedule, a tail assignment algorithm is executed to allocate the

remaining checks. Finally, to preserve tractability and improve quality of solutions, the final step is divided in two stages. The first stage solves conflicts by reassigning maintenance to the next opportunity, while the second stage makes the timeline more granular to improve the resource allocation. The paper uses Gurobi optimzer to solve the models and minimize the violation of regulations. Thereby the resources, regulations and flying time are not hard constraints but are implemented with soft costs. A multi-workshop test case with 16000 flights and 529 aircraft is solved under 2 hours.

4.2. Aircraft Schedule Recovery

Although there is an extensive literature that deals with the classic aircraft routing problem, there is little work on the integration with maintenance schedules. Maher (2016) is one of the few papers that formulates an integrated airline recovery methodology. The author proposes a column and row generation framework in order to reduce the size of the problem. Maintenance planning is enforced by ensuring maintenance critical aircraft terminate at the base locations. A similar approach is undertaken by Ruther et al. (2016) with a branch and price solution method for the *integrated* aircraft routing problem. The routings are generated for one week using a static rolling horizon approach that partitions the optimization in two problems. The author recommends this approach for low-cost airlines which generally operate a point-to-point network and are more open to re-engineering their business process. Contrarily, hub-and-spoke carriers alleviate the effect of *sequential* optimization by having multiple crew and aircraft swapping opportunities at their hubs.

The work of Cordeau et al. (2001) uses Bender's decomposition to solve the same problem, both the Benders master problem and the primal subproblem are solved using column generation. The model only considers routine checks performed every 3 to 4 days with a duration of 6 to 8 hours. Other types of maintenance are not considered because their duration is too long and depends on the availability of maintenance facilities. A three-phase approach is used to tackle the problem: the first phase relaxes all the integrality constraints, during the second they are only added to the master problem and in the third phase, integrality constraints are implemented in the subproblem as well. They report study cases with over 500 flight legs but is not clear if the approach is computationally feasible since the CPU time exceeds 20 hours on hub-spoke networks with many crew-bases.

Heuristic approaches have been widely studied to solve large-scale problems to reach computationally tractability. Even though they are not guaranteed to reach the global optimum, the quality-time trade off is overcome with near-optimal solutions and feasible solving times. Different strategies exist to only explore the promising and realistic actions in the neighborhood of expanded nodes. There have been many publications incorporating heuristic approaches to solve the airline recovery problem. Notable among these, is the work of Yang (2007) who solves the airline recovery problem using Tabu Search, a meta-heuristic search method that uses memory structures to explore decision trees and find near-optimal solutions. The study explores the synergy that exists between tabu search and a multi-commodity flow network. The resulting hybrid model combines a route based method and a time-space network, that showed a favorable performance in terms of speed and objective function value.

4.3. Aircraft Maintenance Scheduling

Sriram and Haghani (2003) consider a 7-day planning horizon assuming a cyclic schedule for an heterogeneous fleet with maintenance constraints. The maintenance check assigned is dependent on the aircraft utilization, thus on the flight sequence that is determined optimal. The scope of the problem is limited by the planning horizon which allows only A and B checks to be considered. Moreover, this is one of the few papers that considers aircraft re-assignment over an optimal maintenance opportunity, even though the aircraft was already assigned. Finally, unexpected maintenance requirements such as unscheduled and reactive maintenance tasks are not considered. However, this is also the case for the previous models presented throughout this chapter. The solution approach uses a heuristic technique that combines a depth first search with a random search. The smaller instances (60 flights and 75 airports) are solved within 5 minutes and then are benchmarked against CPLEX. The result show a gap optimality under 3% with a CPU time reduced with a factor over 200 for the largest cases.

Steiner (2006) presents a heuristic method for maintenance scheduling with the objective of minimizing maintenance actions and evenly distribute among the fleet the capacity requirements and flying hours. In

their model there are four constraint relaxation which involve the quarterly flying hour requirement, the maximum flying hours per week, the shifting tolerances of tasks and the maintenance capacity. In addition, external capacity can be purchased to overcome temporal shortages. The computational period to produce a maintenance plan with a time span of five years varies from 5 to 15 minutes with real test-instances of a large fleet. Even though the time seems surprisingly efficient a clear solution method is not provided after the model formulation.

The MSc thesis by Lotten (2018) claims to achieve a significant reduction in the search space of maintenance task allocation problem by applying an oriented orthogonal bin-packaging algorithm. The task workpackages are not optimally scheduled but the instance size is reduced heuristically by clustering tasks in function of the due date and re-solving the bin-packing with a commercial software. The work of Witteman (2019) approaches the problem in a similar fashion, using an online bin-packing algorithm that allocates task based on the priority of their interval. The approach is twofold, the problem is first solved for a tactical stage and then for an operational horizon. In the first one the resources (man-hours) are distributed over the maintenance opportunities for the whole fleet, decoupling the problem per aircraft. The second stage, allocates the task individually to the aircraft, including additional unscheduled maintenance tasks that may arise.

Deng et al. (2020) developed a look-ahead dynamic optimization model to solve the aircraft maintenance scheduling problem by minimizing the unused flight hours for each aircraft interval. The problem size is determined by a 4-year planning horizon and an heterogeneous fleet of more than 40 aircraft. Two types of phased checks are considered in the model which theoretically should involve all the tasks provided in the MPD. The state space is characterized by three types of attributes. The first one concerns those parameters that have a direct impact on the next decision, the second type refers to attributes of the maintenance check intervals and tolerances, the last type of attributes depend on exogenous information such as the aircraft utilization and the check duration. It is important to note that this last type of attributes does not include stochasticity because the exogenous information is sampled only once with look-up tables and estimates according to historical performance. Therefore the model developed is deterministic. The solution methodology uses forward induction to estimate the value function at each state, otherwise it would be impossible to solve the model for the whole planning horizon due to its size. In order to reduce the size of the problem the aircraft are ordered by maintenance urgency based on the earliest deadline. A *thrifty algorithm* is used to check the *workability* of a state by making sure no aircraft will have to be grounded by supposing the worst case scenario, which makes use of all available slots. Finally, the state space is reduced by aggregation, thus states with similar properties are clustered into a single one, and discretized according to the mean fleet utilization with respect to the checks. The policy developed is a greedy one that exploits all the iterations, i.e. there is no learning process thus the policy can only be applied to a specific scheduling scenario embedded in the formulation.

4.4. Discussion

Most of the literature focus on the aircraft routing problem (ARP) with maintenance considerations using a multi-commodity network flow model with hard constraints. The solutions propose a cyclic rotation with planned maintenance at the end of every routing (Jayaraj et al., 2020). In doing so, the aircraft utilization becomes secondary, and the turn around times are chosen beforehand. Moreover, conflicts are common and a revision to solve discrepancies in the schedule is often requested from the Airline Operation Control Center (AOCC). This also originates due to the lack of communication between the AOCC and the Maintenance Operation Center (MOC) when creating the schedules (Papakostas et al., 2010). Samaranayake and Kiridena (2012) identifies that one of the main limitations of maintenance planning is the lack of integration between aircraft scheduling and resource allocation. The main takeaway of this chapter is that there is barely no work on the maintenance allocation problem. The models found in literature take a preliminary maintenance allocation schedule which is imposed in the operational horizon when solving the ARP.

Over 30 year ago, Feo and Bard (1989) stated that "few optimization models existed for solving maintenance related problems". Even though, over the course of the years, an increasing interest has risen in the field, maintenance schedule optimization remains a great challenge for airlines. It is evident from the analyzed literature that the challenge to face is not the problem formulation but rather the solving technique. The fairly recent literature suggests that the actual tail assignment process in airlines is not fully optimized (Sarac et al., 2006). The imposed hard constraints due to maintenance narrow the routing problem to one-way policies

that do not allow airlines to use the full potential of their schedule, since feasibility is favored over optimization in order to achieve tractability (Boere, 1977). As a direct consequence, airlines incur many opportunity costs, including increased premature maintenance, inefficient usage of maintenance opportunities, unnecessary grounding of aircraft, maintenance demand fluctuation and imbalanced fleet utilization (Safaei and Jardine, 2018).

Furthermore, it is noteworthy that the planning horizon considered in the routing problem does not go beyond 7-days. It has been observed that the full range of maintenance requirements is omitted in the formulation, and only the most frequent maintenance type is considered. Table 4.1 summarizes the key findings of this chapter by classifying the main papers in function of the maintenance type, the methodology used, the objective function and the time-window investigated. This research aims to fill the gap with the literature studied by attempting the formulation of a long-term schedule with A and C checks. The long-term task allocation is a strategic problem, thus the aircraft location and the routings are not available for verification until a week before operations. Finally, all the approaches offer a deterministic solution which neglects the inherent stochastic component of airline maintenance. The few studies that are operational in nature fail to consider the uncertainty of legal remaining flying hours and resources at the maintenance bases.

Table 4.1: Classification of previous works in airline maintenance optimization

Model config.	Maint. type	Methodology notes	Objective function	Horizon	Ref. paper
Aircraft maintenance routing	A-check	Multi-commodity flow network with integer constraints	Min. maintenance costs	1 week	Feo and Bard (1989)
	A-check	Lagrangian relaxation and subgradient optimization	Max. through value of connections	3-5 days	Clarke et al. (1997)
	A-check	Three-stage static infinite-horizon model	Min. routing costs	3 days	Gopalan and Talluri (1998)
	A-check	Branch-and-price with heuristic column generation [1]	Min. unused legal flying hours	1 day	Sarac et al. (2006)
	A-check	Rolling horizon and simulated annealing	Min. unused legal flying hours	1 week	Afsar et al. (2006)
	A-check	Two-stage column generation [1]	Min. expected propagated delay costs	1 week	Liang et al. (2015)
	A-check	Multi-commodity flow network with integer constraints [1]	Min. routing costs	1 week	Safaei and Jardine (2018)
	A-check	Value-function approximation and rolling horizon (ADP)	Max. flying hours	30 days	Lagos et al. (2020)
	A\C-check	Two-stage algorithm to solve AMS and TA [1]	Min violation of maintenance regulations	30 days	Sanchez et al. (2020)
Aircraft schedule recovery	Daily check	Benders decomposition with column generation	Min. routings and pairings costs	1-3 days	Cordeau et al. (2001)
	Daily check	Hybrid multi-commodity flow model with Tabu search	Min delay, cancellation and swap costs	1 day	Yang (2007)
	Daily check	Column and row generation[1]	Min. delay, cancellation and crew costs	1 day	Maher (2016)
	Daily check	Branch and price algorithm [1]	Min. routes and pairings costs	4 days	Ruther et al. (2016)
Aircraft maintenance scheduling	A\C-check	Depth Search and random search[1]	Min. maintenance and re-assignment costs	1 week	Sriram and Haghani (2003)
	A\C-check	Heuristic algorithm	Min. maintenance actions	5 years	Steiner (2006)
	A\C-check	Stochastic look-ahead policy (ADP)	Max aircraft flying hours	6 months	Looker et al. (2017)
	A\C-check	Bin-packaging algorithm	Min. maintenance and re-assignment costs	4 years	Witteman (2019)
	A\C-check	Deterministic look-ahead policy (DP)	Min. unused legal flying hours	4 years	Deng et al. (2020)

[1] Solved by global optimization solver, e.g. CPLEX, LINGO, Gurobi.

5
Dynamic Programming Approaches

Artificial intelligence has revolutionized multiple fields of engineering with big data and novel optimization techniques. It is a strategy that does not require hand-engineering and is able to build large parametric approximators to solve combinatorial problems by means of trial and error in simulations (Bengio et al., 2020). The methodology researched in this chapter is referred to as *Approximate Dynamic Programming* (ADP) in the operations research community, and *Reinforcement Learning* (RL) in the field of control theory. The fundamental idea is to let an agent evaluate decisions through rewards, allowing him to iterate over the optimization environment and train an optimal policy model.

The common ground of the previous approaches is solving highly-dimensional problems related to scheduling and task allocation. Airline maintenance operations are inherently stochastic because they include unexpected events such as disruptions and unpredictable failure modes. Unscheduled maintenance has a regular occurrence in airlines. This clearly outlines the limitations of deterministic models to provide useful guidance in the schedule planning process. In this chapter, the need of a stochastic model is highlighted and a series of algorithmic strategies are presented based on their suitability to the research topic. The discussion's ultimate objective is answering the fourth and last research question: *what ADP methods can produce near-optimal solutions for maintenance scheduling?*

5.1. Classic Dynamic Programming

The concept of dynamic programming was first coined by Bellman (1972) that wanted to describe a mathematical optimization approach ('programming') used to model a time-varying environment ('dynamic'). During the training process the agent learns to define a **policy** $\pi : S \rightarrow A$, that maps a state S into an action A. The Markov Decision Process (MDP) is used to formalized the dynamic programming model (Busoniu et al., 2010). Starting from the fundamentals, an MDP respects the Markov Property, this is also called the memoryless property. It means that the *transition probability* to the next state is merely dependent on the present state and all the previous information can be discarded (Alagoz et al., 2010). The **Markov transition function** is also known as the system model. Normally, textbooks on dynamic programming assume that the transition probability is given, but in many problems, it can be very hard to compute (Iram, 2020). Another vital element is the **reward system**: a value for each decision associated with reaching a goal. The goal is to maximize the gain G_t which is the sum of all the rewards discounted by $\gamma \in [0,1)$ with an exponential factor respect to the current timestep, as shown in equation 5.1.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{5.1}$$

The optimal policy is calculated incrementally and iteratively by satisfying the the Bellman optimality equation shown in 5.2 as the state-value function and in 5.3 as the action-value function. The action-value function returned by the controller indicates the value associated to each action for a given state. Iterative policy evaluation is used to train the agent and eventually achieve the optimal status. The one-step transition uses the old value of the policy evaluation and the estimation of future rewards to update the current policy. This kind of operation is known by *full backup* because the iterations are based on all the possible next states rather than on a sample next state. The update is performed as a sweep across all the states, and the order in which this is done influences the rate of convergence.

$$v_*(s) = \max_{a_{t+1}} \quad \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] = \max_{a_{t+1}} \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s') \tag{5.2}$$

or

$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma q_*(S_{t+1}, a') | S_t = s, A_t = a] = \sum_{s',r} p(s', r|s, a) \cdot [r + \gamma \max_{a'} q_*(s', a') \tag{5.3}$$

Dynamic Programming iteratively solves the value-function by stepping backwards in time. The reason why it has such a scheme is because the value function of the current state at t_s needs the contribution of the final state and the ones in between. Therefore, it steps back in time from the final state to the current one in order to estimate the value function at the current point. Within the DP models two different structures can be identified. First, some problems have a specific end-time or they are modeled over a **finite horizon** to reach a decision at the current time instance. A class of problems attaining to this description are those that require reaching some goal in the future (but not at a particular point in time). The second typology, called **infinite horizon** problems, covers those where the governing exogenous information process do not vary over time.

There are two algorithmic strategies to solve DP models. Value iteration is perhaps the most widely used because of the simplicity of its implementation. Basically, it updates the value function at each iteration and then extracts a policy given the new estimate. By contrast, the second approach, policy iteration, picks a random policy and then determines the steady-state value of being in each state given the policy. Not surprisingly, the second method converges faster in terms of the number of iterations, but it takes longer per iteration.

An application of classic DP in the airline industry is presented in Moudani and Mora-Camino (2000) to solve the fleet assignment model (FAM). The FAM is first solved using dynamic programming, and the associated maintenance scheduling problem is tackled separately with a greedy heuristic that inserts one aircraft at a time in the maintenance slots, previously defined. The FAM is solved with a dynamic programming approach for two reasons. The first one is that DP is efficient to treat combinatorial optimization in a dynamic environment, which in this case is dictated by the temporal nature of the operations. Secondly, because of the recursive structure of the algorithm, it is suited to cope with perturbations from nominal conditions, such as delayed and canceled flights, and the deviations can be used in the feedback loop to iterate the solution.

The key idea of Dynamic Programming is dividing the problem into states and **nesting smaller decision problems inside a larger decision** (Dreyfus, 2002). If n and m indicate the number of states and actions, a DP method takes a number of computational operations that is less than some polynomial function of n and m. A DP method is guaranteed to find an optimal policy in polynomial time even though the total number of policies is m^n (Sutton and Barto, 1998). Other deterministic approaches can be used to solve similar problems but their size explodes before DP methods; estimates rise up to a factor of 100. Therefore, for the largest problems, only DP methods might be feasible.

5.2. The need for approximation

Classic DP is an important theoretical concept but almost impossible to use because it requires a complete model of the operations and because of the computational intractability. Approximate models simplify the structure of the search three and allow the use of heuristic approaches that lead to near-optimal solutions. The focus of this chapter is on the approximate models that are introduced to make large optimization models computationally viable. Powell (2011) outlines that the main difference between approximate and classic dynamic programming is how the value function is computed. The algorithmic strategy proposed for ADP involves a series of important notions that should be noted down:

- *Forward dynamic programming* is used to avoid the problem of looping through all the states to estimate the value function, but it requires a one-step transition matrix.
- *Forward induction* only updates visited states but it requires to have an estimation of the values of states that *might* be visited. It replaces the computational burden of looping through each possible state, with the statistical problem of estimating their value.
- States that are not relevant to reach optimality can be avoided but it is also possible to never visit a state that would have created a good solution.

The effect of computing the next state value by stepping forward has two consequences. Firstly, it is impossible to explore the whole state space so the exploration versus exploitation dilemma is faced. The agent needs

to weigh the possibility of exploring other regions of the search space during the optimization, and thus giving up short-term rewards, against focusing on high-reward action and potentially giving up valuable regions of the environment. Secondly, the long-term performance must be maximized using only the feedback of the one-step performance. This is challenging because actions taken in the present affect the future decision space and rewards, but the immediate rewards provide only local information and exclude these long-term effects. This paradigm is also known as the *delayed reward* problem (Watkins, 1989).
Stochastic optimization problems are studied in many settings, as a result different communities have developed algorithmic strategies lacking a common modeling framework. Markov decision processes, stochastic programming, stochastic search, simulation optimization, reinforcement learning, approximate dynamic programming and optimal control deal with sequential optimization problems. Policy iteration may be presented in different flavors, thanks to the diversity in which a policy can be expressed (Powell, 2011). The strategies used to solve the (stochastic) optimization can be grouped around five fundamental classes of **policies** (Powell et al., 2012):

1. **Policy function approximations** (PFA) are analytic functions that return an action given a state, without solving an optimization problem. Thus, the action is computed from an a-priori knowledge-base that could be built out of a set of rules or estimations. A clear example is associating each action with a threshold of a measurable quantity or computing the action based on the average of given parameters.

2. **Value function approximation** (VFA) is a policy determined by an explicit function $V_t^{\pi,n-1}(S_t^x)$ that indicates the optimal action to take. A VFA provides an estimation of the long-term value of a state. Mathematically it is expressed as the expected gain given the current state s. The value-function can be decomposed in *imminent rewards*, that evaluate the present gains, and *discounted rewards* that estimate the value of future states. The standard definition of the value function is given in equation 5.4, this is also called the Bellman optimality equation.

$$V_t(S_t) = \max_{x_t \in \mathcal{X}(S_t)} (\underbrace{C_t(S_t, x_t)}_{\text{Imminent reward}} + \underbrace{\gamma \mathbb{E}\{V_{t+1}(S_{t+1}|S_t)\}}_{\text{Discounted rewards}}) \tag{5.4}$$

3. **Myopic policies** can be used for stochastic systems influenced by several parameters. This is a relatively easy policy to implement. It has a greedy behavior that consists on maximizing the imminent rewards without considering the future evolution of the trajectories. The mathematical expression is exactly the same as the previous one with a discount factor equal to zero.

4. **Cost function approximations** (CFA) modify myopic policies that rarely achieve optimal solutions. The key idea is to slightly modify the cost function with additional basis functions or parametric approximations over a single time period.

5. **Look-ahead policies** optimize over more than one time step into the future, for the purpose of making better decisions now. The most common version is to approximate the future deterministically and solve the deterministic optimization problem. The main difference between look-ahead and value-function approximation is the necessity to solve the future periods in order to make a decision, while the latter uses an estimation. Therefore, look-ahead can be considered a backward dynamic programming method.

5.3. Stochastic modeling in ADP

In the context of airline maintenance there are variables that are inherently dominated by uncertainty. The most obvious is the interval definition of the task. Since most of them are based upon flying hours (FH) and flight cycles (FC), it is required to estimated the aircraft utilization to set a fixed task due date. Moreover, the task duration might also vary due to the possibility of encountering non-routine tasks during inspections. Finally, the CBM predictions add another layer of complexity and stochasticity. The remaining useful life (RUL) prognostics are continuously being updated and produce new task lists to be incorporated in the schedule before the due date.

One of the main advantages of ADP is the facility to model stochastic variables in the formulation. Ideally any problem could be solved deterministically by calculating the expectation in Equation 5.4. However, most of

the time it turns out that the solution space is too large to converge with standard optimization techniques. The solution procedure can take two different perspectives in this case. One possibility consists on using value functions, scenario trees approximations or any parametric model that can be used to measure the impact of a decision now on the future. Nevertheless, it is common practice in the industry to use deterministic approximations on the future in order to create an easier problem formulation, but which is criticized for ignoring uncertainty and being computationally very demanding. The deterministic models of airlines use fixed flight times to schedule aircraft and crew, energy companies use deterministic predictions of wind and solar loads, retailers rely on deterministic estimates of demand to plan inventories, and so on. These models are modified in order to handle uncertainty. For instance, a supply chain management problem can handle uncertainty through buffer stocks, an energy company might include reserve capacity, and an airline scheduling model might handle stochastic delays using schedule slack.

It can be concluded that there are two ways to capture the inevitable uncertainty. The first one uses statistical distributions to model the range of possibilities (including slack), and solve samples of the model deterministically. This technique is also called *sampled approximation* because the objective function tries to maximize the *sample average approximation* instead of the expectation in Equation 5.4. The second method includes the stochasticity in the modeling. Therefore, for the same state-action combination, the response of the model varies based on the *exogenous information*, by this term we refer to external information that is uncertain until the next action is taken. These approaches are referred to as *adaptive learning models*, they use iterative methods to solve sequences of relatively easy problems, with the hope that the algorithms will converge to the solution of the original problem. All sequential learning algorithms, for any stochastic optimization problem, can ultimately be reduced to a sequential decision problem, otherwise known as a *dynamic program* (Powell, 2020). In the remainder of this chapter the algorithms presented will either model stochastic variables explicitly, or sample stochastic trajectories solved by deterministic models.

5.4. The curse of dimensionality

It is well-known that solving the Bellman optimality equation (5.4) suffers from the curse of dimensionality associated with high-dimensional problems that may lead to a computational intractable structure (Powell and Topaloglu, 2006). Unfortunately, there are three curses of dimensionality that arise from the size of the state, the outcome, and the action space. In this section a very elegant theory is presented for solving stochastic, dynamic programs when it is possible to make some reasonably limiting assumptions that allow forward induction. The following examples include different categories of value function approximations (VFA) to tackle **the first curse of dimensionality**. Their implementation provides estimate for the value of future states without having to sweep across all the possible ones.

(a) **Look-up table**. A policy that can be conveyed in an implicit tabular form where each action can be associated with a state in the form: $x_t^n = X_t^\pi(S_t^n)$. This entails that for every discrete state there will be a tabular entry that maps into the discrete action or state-value.

(b) **Parametric function**. This approximation can be utilized in policies where the decision to be optimized is a quantity that depends on other observable variables. It is very common to write it in linear form based on a regression model.

(c) **Neural networks**. This is a successful framework to express a general statistical expression which has to be trained to learn which action should be taken in a given state. They are considered a sophisticated version of parametric methods.

(d) **Non-parametric functions** estimate the value function based on a series of observations rather than using parametric features with a preset behavior. They represent a flexible approach which can be convenient in functions that have relationships with the marginal value of resources in the state-space. A common example is Gaussian kernel regression and support vector machines.

The key message of this section is that the classical Bellman equation (5.4) can be rewritten replacing the expectation ($\mathbb{E}$) with an approximate function for the next state $\bar{V}_t(\cdot)$, as shown in Equation 5.5. The algorithmic strategy works as follows, at each iteration n, a path ω^n is sampled and a post-decision state $S_{t-1}^{x,n}$ will follow after taking a decision x_{t-1}. By dropping the expectation, it is possible to extend the scope of the optimization to stochastic problems, where each transition is influenced by some exogenous information that is

unknown until the next action is taken. To clarify the reader, the same notation used in Powell and Topaloglu (2006) is introduced for the pre- and post-decision state vector, respectively: S_t and S_t^x. Thus the sequence of state decisions and information would be $(S_0, x_0, S_0^{x_0}, W_1, S_1, x_1, S_1^{x_1}, ..., W_T, S_T, x_T, S_T^{x_T})$, where W_{t+1} is the exogenous information that becomes available at time $t+1$ after making a decision x_t. This involves stochastic information regarding the number of resources and demand related to an attribute vector. Similarly, for the problem studied, the aircraft utilization, the duration of the task and the new task demand after making a decision could be treated as exogenous information. These changes lead to a new equation to be solved, visible in Equation 5.5, where $\hat{v}_t^n$ is a sample realization of the value of being in state S_t^n.

$$\hat{v}_t^n = \max_{x_t \in \mathscr{X}(S_t^n, W_t^n)} \{C_t(S_t^n, x_t) + \gamma \bar{V}_t^{n-1}(S_t^{n,x})\} \tag{5.5}$$

From here, it is possible to update the value function approximation using a relation depending on the the previous iteration and the new value. Multiple approaches exist ranging from simple linear relations with a careful choice of steps to non-linear or piecewise-linear functions (George et al., 2008). The pre-decision state at iteration n can be computed with $S_t^n = S^{M,W}(S_{t-1}^x, W_t(\omega^n))$. From the pre-decision state S_t^n, the feasible region $\mathscr{X}_t^n$ can be computed. Instead,the post-decision state is computed using $S_t^{x,n} = S^{M,x}(S_t^n, x_t)$. The problem is solved for a particular realization of new exogenous information by computing a single decision x_t. In doing so, **the second curse of dimensionality** due to the outcome space is tackled.

The third curse of dimensionality refers to the action space. In other words, it becomes relevant only when the action space is so large that the *max* operation in Equation 5.5 is too complicate. In order to tackle this problem a specific value-function can be used to reduce the size of the feasible region defined by $\mathscr{X}(S_t^n, W_t^n)$. Other techniques such as discretization and aggregation of states are also employed to achieve the same objective. Finally, another possibility consist on considering online approaches that limit the search three expansion or even reducing the problem size using approaches such as the rolling horizon. This concludes the curse of dimensionality also widely cited as the "Achilles heel" of dynamic programming.

5.5. Applications of ADP in Literature

Hereafter, a series of ADP techniques that have been implemented in literature and were considered to be highly related to the task allocation problem, are presented and thoroughly discussed. Moreover, it is possible to grasp what industries aside from the airline maintenance repair and overhaul (MRO) are affected by analogous paradigms.

5.5.1. Dynamic fleet management

The work of Simão et al. (2008) is one of the most influential and relevant publications to ADP. The model developed is meant to manage the fleet of a large truck company in the US. The assignment of truck loads to drivers is done via an innovative ADP model with the goal of *resembling the current network of the company as much as possible*. This has been highlighted to diversify this work with the optimization purpose of this research. To do so, the mathematical formulation ties a series of attributes for each load and driver to the state definition, e.g. home location of the driver, duty hours, travel hours, etc. The model uses the attributes as part of the state and creates an objective function to minimize the cost of operations. The author justifies the use of a linear approximate value function with the fact that the focus of this project was to understand the type of driver that should be assigned to each truck load, and not to optimize the network. Therefore, data regarding the next state is available to construct the linear function. Finally, a direct mapping with assigning maintenance tasks to multiple aircraft can be seen with the distribution of truck-loads over drivers. Most of the papers on ADP focus on single-machine scheduling which significantly reduces the size of their models. Therefore, this method could be used to have an initial formulation of the research problem even though the proposed value function cannot be used for optimization.

5.5.2. Inventory Optimization

In Simão and Powell (2009) a novel model is proposed to optimize the maintenance parts inventory of different cost and frequency demand. The state definition of this approach is composed by two elements. The first one indicates the number of components for each attribute defined, i.e. location, type, weight, production

lead-time, age, etc. While this captures the state of the inventory being managed, the second element contains information about the uncertainty involved in the process (exogenous information). This encapsulates parameters directly related to the likelihood of failure, such as age of the fleet, average lifetime of the parts and failure distributions. Moreover, the author hovers on the computational intensity of discrete resource allocation problems, that can be intractable if solved with a classic deterministic approach. The value function approximation is achieved via the marginal value of having a certain component available as a function of the attributes. In the case that the attribute is not relevant for the estimation, a myopic strategy is used where the impact of decisions on the future is ignored. Unless there is exogenous information, like the availability of CBM data, the value-function approximation will have to be computed iteratively using dual-variables or derivatives estimated with the evolution of the state value-function.

As part of the ADP strategy the problem is decomposed into smaller planning horizons, this yields a sequential optimization which is then solved chronologically. Even though the size of the state space is extremely reduced, the risk of diverging from the optimality can be dangerous. For instance, in maintenance scheduling this could risk postponing tasks until failure occurs or the interval is exceeded. Sometimes, with the requirement of backtracking to previous options that may not be feasible anymore.

5.5.3. A Vehicle Routing Problem Example

The vehicle routing problem deals with the optimal assignment of a series of transportation services that must be picked up and delivered in different locations by a fleet with a series of compatibility and capacity constraints, sometimes the visit time-windows are also specified (Goel and Gruhn, 2008). This is a classical combinatorial optimization problem that can be extended by considering a dynamic setting with incoming stochastic information. For instance, in Baradaran et al. (2019) the cost of the edges is subject to volatile variables such as travel time rather than a fixed distance, the demand also changes over time and needs to be fulfilled in specified periods.

Ulmer (2020) proposes a novel ADP approach to solve these problems. The author alleviates the trade-off between online and offline approaches by combining them into one single method. On one hand, a value function approximation (VFA) is used as an offline method to capture the decision and transition space. On the other, a rolling horizon algorithm (RH) is employed to have a full detail of the state space due to aggregation. The strategy is called VFA-based limited horizon rollout algorithm (V-LHRA). The algorihtm procedure is depicted in Figure 5.1 in order to ease the steps of the discussion. The first circle to the left represents a post-decision state S_k^x. At every state, a decision is made using the approximate Bellman equation (Eq.5.5) with a sampled online transition ω_i^k that is simulated m times (vertical dimension in the figure). This process is a variant of a Monte Carlo tree search. Each sample transition is simulated until the rollout horizon is reached at $k+h$ (horizontal dimension in the figure). The value for each simulation is the sum of accumulated rewards plus the VFA value of the last simulated post-decision variable S_{k+h}^x, as shown in the right hand side of the figure. The approximate value for each post-decision state is averaged out of the m simulations for each S_k^x. The presented approach outperforms state-of-the-art benchmark policies while additionally reducing the required online calculation time drastically for the case sets studied in the paper.

A different approach is undertaken by Nazari et al. (2018) to model the vehicle routing problem as an MDP. However, an exciting extension is introduced: to improve the policy, a neural network is used to make decisions. The placement decisions are made step-by-step, such that the agent is able to learn from intermediary states, instead of using an online roll-out approach for every training instance. It is noteworthy that the model does not infers decisions from a single input as it happens in end-to-end learning.

5.5.4. Aircraft Maintenance Optimization

Stemming from Powell's ADP knowledge, Lagos et al. (2020) develops a stochastic and dynamic model to solve the *operational* maintenance scheduling problem with tail assignment. The formulation allows to enforce an aircraft on ground (AOG) when the aircraft critical task date expires. In the paper, tasks get disclosed dynamically over the 30-day planning horizon with a predictive anticipation to resemble real-life operations. The model is solved with Approximate Dynamic Programming in order to overcome the curse of dimensionality. The cardinality of the state space grows exponentially with the number of tasks and fleet size, as the expectation expression has many terms depending on the number of tasks per day.

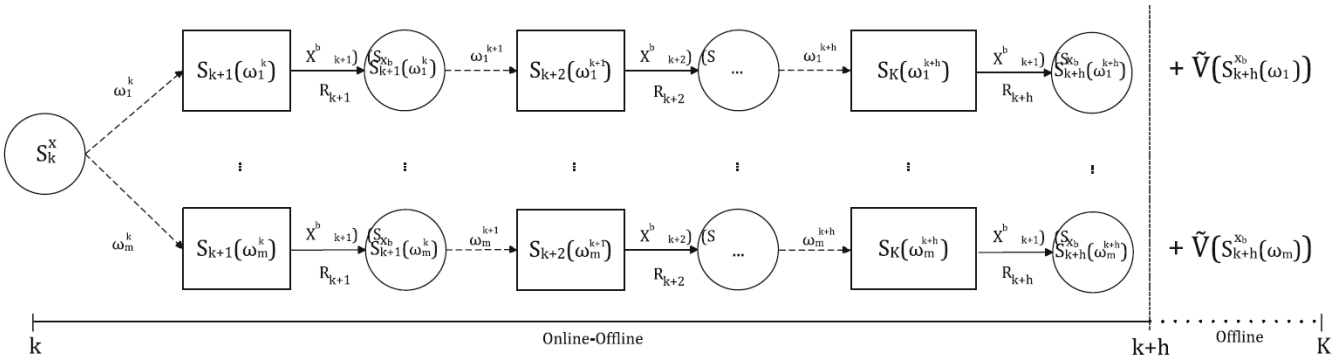


Figure 5.1: VFA limited horizon rollout algorithm (V-LHRA) structure (Ulmer, 2020)

The state at time t is defined by the ground time for each aircraft available for maintenance operations and by the task status (disclosed or pending). The decision variables capture the tail assignment to a line of flight, the maintenance assignment to an aircraft, and finally what tasks are executed on each maintenance day. The constraints that have been implemented in the model range from flight and maintenance coverage to capacity and resource restrictions. Finally, the objective function tries to minimize the cost of AOG's and expired maintenance tasks.

The author uses a parametric value-function approximator (VFA) and a Rolling Horizon (RH) policy to approach the problem. This is a very similar hybrid approach that complements offline and online methods as the one proposed in Ulmer (2020). The offline approximations are based on a number of features that are weighted with a learning technique called Approximate Value Iteration (AVI) and linear regression, while the online recursion uses a truncated horizon of one week. It is noteworthy, that one of the most useful features to calibrate the VFA is the workload of pending tasks with a few days left before the due date. The best solutions are returned when the tasks requiring less resources are scheduled first, in order to increase the utilization of maintenance residual capacity.

Looker et al. (2017) proposes an ADP model to address the maintenance allocation problem of a military aircraft fleet. The objective function is structure to maximize the aircraft utilization in flying hours and the maintenance throughput. The decision variables include the allocation flight hours and maintenance slots to an aircraft. Moreover, the state space is composed by a continuous variable that indicate the accrued flying hours and a binary variable that activates if an aircraft requires maintenance. The unscheduled maintenance is modeled as an exogenous stochastic variable that represents the daily amount of new unscheduled maintenance generated randomly from a lognormal probability distribution. It includes both time between failures and time to repair (duration). However, in the ReMAP project, the non-routine task demand should be generated based on the CBM prognostic data. The model is designed to discover policies that dynamically adapt to random events to deliver optimal fleet serviceability. The policy considered is a deterministic look-ahead that depends on a discount factor and a fixed time horizon. More specifically, the computational experiment produces a schedule plan for 180 days for a 12-aircraft fleet with a sliding time-window of 1 to 14 days.

5.6. Model-free vs Model-based

An approximate dynamic programming approach is twofold, depending on how the model is set up it is possible to have either a model-based or a model-free scheme (Powell, 2011, Sutton and Barto, 1998). In the former case the agent has *full observability* over the environment and the problem can be defined as a *Markov Decision Process* (MDP). Here the agent learns a model of the environment thanks to the explicit transition function, and is able to predict the next state and reward value. Otherwise, when the agent has *partial observability* over the environment, one refers to this case as a *Partially Observable Markov Decision Process* (POMDP) or model-free learning (Singh et al., 1994). In that case, the agent requires to remember the past observations in a memory storage, which is then used for training. A clear understanding of the problem is required to determine which of the two is more apt to solve a particular combinatorial optimization problem.

Some systems are so complex that mathematical models are not able to represent them. However, it might be possible to observe behaviors directly. Such applications arise in operational settings where a model is running in production, establishing the outcomes observation and state transitions from physical processes rather than mathematical equations. A simple example, is a tic-tac-toe algorithm that does not have a model of the opponent. In the field of dynamic programming, *model-free* refers to those systems that lack of an explicit transition function. Thus, an MDP cannot be defined, and an exogenous process is assumed to be available to generate observations and outcome of the system response (Powell, 2011). They are typically more flexible and thus more common in deep reinforcement learning but they require more samples in the learning process (Jin et al., 2018). On the other hand, *model-based* algorithms are highly dependent on the ability of representing the environment transition dynamics, typically expressed as a decision tree which expands exponentially as it branches out in time. Instead model-free is based on sampling the episodes and replaying a buffer memory during training (Polydoros and Nalpantidis, 2017). Therefore, model-free methods are considered more suitable for high-dimensional problems such as the airline maintenance scheduling (AMS).

Sutton and Barto (1998) divides the model-free approaches in Monte Carlo and temporal difference (TD-learning). The first one does not update the estimates of a state-value until the following state is visited. Then, it uses that value as a target. This procedure is similar to the approaches that have been previously presented in this chapter, where each iteration takes a sample path and is solved deterministically. In the following sections, more emphasis will be placed on temporal difference approaches (TD-learning) because they perform *bootstrapping*, a technique that becomes relevant in high-dimensional optimization problems. During bootstrapping the state-value is updated based on *estimates* of successor state-values without the need to visit them first. Once the agent learns how to behave, it may generalize the model it has learned to other unseen problems.

5.7. Q-Learning

Q-Learning was first introduced by Watkins and Dayan (1992) in the early 90's. The algorithm presents one policy used to decide which action to evaluate in the training procedure, and a separate policy, which is continuously improved, is used to choose the action in the future. It is an off-policy algorithms because it learns the value of an optimal policy independent of the agent's actions (Odonkor and Lewis, 2018). The algorithm makes no attempt to learn the underlying dynamics of the environment, in the case of this research is the maintenance scheduling update of resources, hence it can be labeled as model-free. Knowles et al. (2011) is one of the few papers that has already tried to approach the CBM problem with a Reinforcement Learning strategy. The paper is very indicative of the problem that has been formulated in this research. A crucial point highlighted is that maintenance scheduling can be optimized via a reinforcement learning algorithm because it is structured as a long-term optimization over a series of short-term decisions. One of the main gaps in this research is the lack of stochasticity: a deterministic distribution of failures is assumed and the model is trained accordingly. Moreover there is no diversification between maintenance task categories and the costs of repairing and performing maintenance are unique. An interesting recommendation is the usage of function approximators for scaling the problem size in terms of states. The paper uses a classic Q-Learning algorithm which discretizes every possible state.

5.7.1. Deep Q-Learning

The dimensionality problem can be circumvented with a neural network. In this model the input size has to be determined, but not the scale. Therefore, having a functionality to capture the optimal policy of the problem could be very efficient if the right features are extracted to improve it (Mnih et al., 2013). Equation 5.6 shows how the value function is updated with the Bellman optimality equation based on the online network parameters θ, and the target network parameters θ^{-}. The target neural network is copied from the online network every τ steps because this delay improves performance and reduces bias in the Q-value estimation. This approach, called Deep Q-Learning, has been successfully applied in literature in problems ranging from optimal control to resource allocation. Mnih et al. (2015) received overwhelming attention from the reinforcement learning community after developing a Deep Q-network able to surpass performance scores on 49 different Atari games, without altering the model hyperparameters. Their work was followed by the application of Deep Q-Learning in other domains. Among others Waschneck et al. (2018) stands out for their dispatching

rule model that optimizes the schedule for production control. Recently, Dai et al. (2017) proposed a Deep Q-learning architecture to learn greedy policies over graphs for a diverse range of combinatorial optimization problems.

$$Q(s_t, a_t;\theta) = (1-\alpha)\ \underbrace{Q(s_t, a_t;\theta)}_{\text{Online network}} + \alpha \left(R_t + \gamma \max_{a_{t+1}} \underbrace{Q(s_{t+1}, a_{t+1};\theta^-)}_{\text{Target network}} \right) \tag{5.6}$$

A powerful attribute of Q-learning is that it is able to create a direct mapping between features of the state and the optimal action. Luo (2020) clarifies this concept when solving the flexible job shop scheduling problem with Deep Q-Networks. In his work, seven generic state features are extracted to represent the production states and learn the most suitable action. The model processes the continuous state feature as an input to the neural network and is trained to output the state-action value of each dispatching rule, also called the Q-value in this algorithm. Classical approaches would instead assume static manufacturing environment, thus producing deterministic solutions that are not able to consider variations that inevitably affect the solution in today's complex and varying operations.

To conclude, it is important to emphasize that Deep Q-Learning solves problems related to a large number of states, but it still struggles in high dimensional action spaces. The reason being that finding an optimal decision, first requires to estimate the Q-function for all possible actions.

5.7.2. Double Q-Learning

The standard deep Q-Learning algorithm is known to overestimates the action-values due to the **max()** operator in the Bellman Equation. The risk of using overestimated values does not affect only locally the training but it is resented on other states due to the discounted influence of future rewards. The root cause of higher output values in the last neural network layer can be attributed to noise and an insufficiently flexible function approximation. To prevent this from happening, Hasselt et al. (2016) propose to use an alternative approach to calculate the Q-value of the next state-action pair.

The standard DQN approach uses the target network to both, select and evaluate an action. Instead, in double DQN the online network is used to compute the best action and the target network to evaluate it. This is mathematically expressed in Equation 5.7 where the resulting update function can be seen. It is noteworthy, how the first set of weights, θ, is employed to select the action via the **argmax()** operator, while the second set of weights, θ^-, is used to fairly evaluate the value of this policy.

$$Q(s_t, a_t;\theta) = (1-\alpha)\, Q(s_t, a_t;\theta) + \alpha \left(R_t + \underbrace{\gamma\, Q\left(s_{t+1}, \arg\max_{a_{t+1}} \{ \underbrace{Q(s_{t+1}, a_{t+1};\theta)}_{\text{Online network}} \}; \theta^- \right)}_{\text{Target network}} \right) \tag{5.7}$$

5.7.3. Dueling Q-Learning

The dueling architecture (Wang et al., 2016) is a simple concept that includes an extra layer of neurons in a DQN. This layer explicitly separates the representation of state-value and action advantages. The two streams are aggregate in the output layer to produce an estimate of the state-action function. Figure 5.2 contains a clear representation of the new layer that is added in the DQN structure.
The connection between the state-value and the action advantages can be demonstrated with Equation 5.8. Intuitively, it follows that V measures how good it is to be in a particular state s. The advantage function A gives a relative measure of the importance of each action. Finally, the set of $Q-$values, which is the result of the output layer, provides the value of choosing a specific action when being in state s.

$$Q(s,a) = V(s) + A(s,a) \tag{5.8}$$

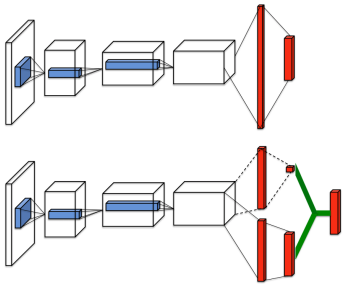


Figure 5.2: Standard DQN (above) and Dueling DQN structure (below) (Wang et al., 2016)

Prioritized Experience Replay
An additional feature that may boost the performance of a DQN is prioritizing valuable experience during the training. The experience an agent gains during the simulation is stored in a replay memory. In a standard approach a uniform sampling occurs to select the steps that are fed to the neural network update. However, Schaul et al. (2016) enhances the performance of the agent by replaying important transitions more frequently, and therefore learn more efficiently. This idea can be generalized by including a priority for each step the agent takes. A feasible approach is to use the magnitude of the temporal difference (TD) error in the Q-update, between the target the and the online network, as priority function, such that the smaller the error the higher priority should be given to that step.

5.8. Actor-Critic

At the beginning of this chapter a clear division was made between value-based and policy-based approaches to classify solution methodologies in dynamic programming. Actor-critic methods combine both approaches into a single one by having two networks: the actor and the critic. The critic uses temporal difference to learn a value function. The actor is updated in an approximate gradient direction, based on information provided by the critic (Q-value network), to produce an optimal policy (Konda and Tsitsiklis, 2003).

The actor-critic methodology can be seen as a learning framework rather than a single algorithmic strategy. There are two main variants that have been particularly successful in literature: advantage actor-critic (A2C) and deep deterministic policy gradient (DDPG). The first one relies on Equation 5.8 where the actor policy calculates the advantages of each action with respect to a baseline. The baseline is calculated by the critic in the form of the state-value function (Mnih et al., 2016). It is an online policy method because as the behavior of the agent changes the updates occur independently of past estimates, so there is not a replay buffer memory as it happens in Deep Q-Learning.

Lillicrap et al. (2016) introduce deep deterministic policy gradient (DDPG) as an extension of DQN on continuous state-action domains. The architecture consists of two main networks, the actor and the critic, and two time-delayed copies of these which are called the target networks. It is an off-policy method, like DQN, that uses a replay buffer to sample experience and update neural network parameters. The value network (critic) is updated similarly as done in Q-learning. The updated Q value is obtained by iterating over the Bellman equation. However, in DDPG, the actor updates occur with the next-state Q values in function of the target value network and target policy network. To calculate the policy network loss function, the objective function is derived with respect to the policy parameters or the actor network weights. Liu et al. (2020) apply DDPG to solve a job shop scheduling problem (JSSP). The problem is formalized as a series of jobs that need to be processed on a limited number of machines and processing time, each time an action is taken a job is dispatched. The state is represented by a matrix that includes features such as process time, job compatibility with each machine and completion of the job. The proposed model comprises actor network and critic network, both including convolution layers and fully connected layers. The results are benchmarked against Google's OR-Library with an accuracy of 91% on deterministic problems and 83% on stochastic environments.

5.9. Neural Combinatorial Optimization

Bello et al. (2019) is among the first works to use machine learning in combinatorial optimization. In his paper

he proposes *Neural Combinatorial Optimization* (NCO) approaches to solve large-scale problem. Classical supervised frameworks are not applicable due to the lack of optimal labels. However, if combined with a reinforcement learning algorithm a feedback reward can be designed to steer an agent in the learning process. The interplay between machine learning (ML) and combinatorial optimization (CO) is discussed in Bengio et al. (2020). In the article, the author mentions the problem of sparse rewards because it hinders the agent's behavioral improvement if no positive rewards are obtained. Therefore, special attention has to be conveyed when creating a simulation-environment to provide learning opportunities. When ML is used to approximate functions and enough data is provided, a policy can be learned by imitation through the expected behavior which is embedded in the data labels. In case of optimization, a new policy needs to be discovered, by matching the reward structure with the optimization objective, the goal of the learning agent becomes to solve the problem, without assuming any prior expected knowledge. Bengio et al. (2020) complete their discussion by considering augmentations of the CO problem with ML and MILP, for further information the reader is referred to their paper.

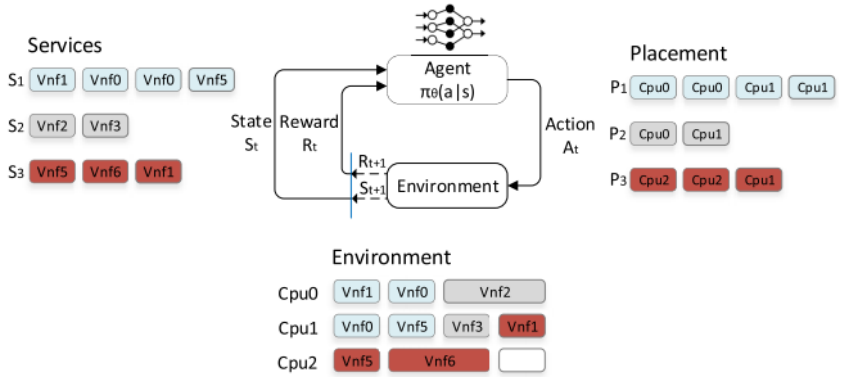


Figure 5.3: Reinforcement learning action-reward feedback loop (Solozabal et al., 2020)

The work of Solozabal et al. (2020) is reasonably related to the task allocation problem. This paper solves a bin-packaging problem using a Recurrent Neural Network as value-function approximator. The problem considered is cloud computing: a series of network services that need to be optimally placed in a set of host servers. Along the same line, the maintenance task allocation problem consider a series of tasks that need to be optimally placed on a set of maintenance opportunities. While on the first problem capacity is dictated by computing power, storage and connection capabilities, on the maintenance problem the main resource considered are the man-hours per task. Moreover, there is an additional layer of complexity given by the interval deadlines, which in the problem of Solozabal could be regarded as a constraint that dictates which network services can be hosted on each CPU. In very simplistic terms, the bin-packing problem can be seen as a "Tetris" game, however the final goal is not to construct a puzzle but a schedule that takes the least possible resources. Figure 5.3 clearly depicts the reinforcement learning process. As visible, the agent's action is a sequence with equal length as the input that indicates where each service is going to be placed. The agent uses NCO to infer a policy $\pi_\theta(p^s|s)$, where θ are the weights of the neural network, and the policy is able to map an observation of services, s, to an optimal placement, p^s. Thereby, the model proposed has an end-to-end learning structure. NCO does not consider constraints within its formulation. Thus, dealing with constraint dissatisfaction is essential to have an objective function that captures enough information to infer a competitive policy. In this work, the author chooses to further extend the objective function using Lagrange relaxation technique to include the constraints. The Policy Gradients method suffers from local convergence. Throughout the learning process, the weights of the neurons are calibrated in the direction of the gradient of the chosen objective function. Since the solution space is non-convex, this method is prone to converging to sub-optimal minimums. In order to escapes these zones, exploitation techniques such as *entropy regularization* are used in the training process.

5.10. Discussion

Approximate dynamic programming is a wide umbrella that encloses different policy behaviors. The reality is that with rare exceptions, all the approximate classes of policies are almost guaranteed to be suboptimal (Powell, 2014). This is the cost of dealing with a stochastic optimization model. Nonetheless, it is possible to argue that each class of policy offers features that could match specific classes of problems. PFA's are best suited for low dimensional action spaces and the behavior to be expected is fairly obvious. CFA's can be attractive for deterministic models that are not able to account for the uncertainty invovled in the environment. Moreover, look-ahead policies can be useful for time-dependent problems and especially if there is a stochastic prediction involved. Lastly, VFA's are applicable in highly dimensional problems and when the future state can be estimated, to communicate the marginal value of being in a state. Therefore, more attention has been drawn to the latter, given the availability of data, multidimensional resource allocation problems have been successfully treated with VFA in literature. Hybrid approaches are also a valuable option in order to capture interactions that might be hard to estimate otherwise.

One of the main points to be addressed is the difference between model-free and model-based approaches. In model-free methods, the agent's policy is optimized in a simulation environment by a trial-and-error. Instead, model-based methods develop an internal model of the transition dynamics to derive the optimal policy. Thus, the policies are updated using the model, and then the optimal policy is applied in the environment. A transition state function is necessary to update the environment attributes that define the state when a new decision is taken. In the case that the dynamics of the underlying environment are too complex and require large decision trees to be formalized, model-based schemes can be biased towards their internal modeling, while model-free can learn a mapping from state to actions based on the correct observable features. Therefore, a model-free approach seems to be a more natural choice at first sight.

One of the main advantages of using reinforcement learning is that is effective in a high dimensional and continuous state space. As seen in most of the control applications, it is possible to extrapolate the properties related to a state, model them in the optimization with a continuous domain and convert them back into a discrete action with a mapping function. Furthermore, reinforcement learning it is best suited to learn stochastic policies since the training process requires a large amount of samples to learn the optimal policy with respect to a series of (stochastic) attributes and observations. On the other side, disadvantages are also present when approximations are considered: the local convergence is highly dependent on what approximation functions are chosen. For instance, a neural network uses stochastic gradient descent which means that neuron weights are trained in the direction of local minimum that can become difficult to escape without a diversification strategy.

Another appealing design choice is end-to-end deep learning. However, those reinforcement learning approaches have the risk to be inaccessible processes. As in the Neural Combinatorial Optimization case, the agent learns to infer the policy at once from a single input source. In practice, it becomes cumbersome to fine-tune an approach with a very large state-space which is also not transparent. Moreover, placements are naturally made in step-by-step decisions, so an agent should be given the possibility to learn from the intermediary processes when allocating maintenance tasks. Thus, an end-to-end strategy should be avoided.

Finally, it is noteworthy that most works focus on specific techniques around reducing the model size to make it computationally tractable. Among others, the rolling horizon stands out for drastically reducing the computational efforts by solving smaller time-window instances. It is also important to include state aggregation and discretization, especially when considering a stochastic optimization setting. Moreover, the forward induction or approximation is an essential ingredient to reduce the optimization iterations. The last approximation methodology that should be considered is the decoupling of the problem into several parallel stages to go from a multi-machine allocation to a single-machine scheduling problem.

6
Conclusion

This literature review explores the airline maintenance scheduling problem from a bottom-up perspective. The ultimate goal of this work is to develop an adaptive schedule with a task-resolution, rather than a rigid check-letter scheme. A plethora of literature is critically reviewed to address the challenges that condition-based maintenance (CBM) brings along, and to enable a dynamic decision-making framework.

The vast majority of the research in airline maintenance focuses on the operational horizon by solving the aircraft maintenance routing problem. The coupling of the tail assignment with the maintenance scheduling problem leads to a large model, whose size quickly becomes absurd with the number of flights, tasks, aircraft and time-windows. Thus, only the most frequent checks are considered, and schedules are assumed to have a cyclical pattern. The optimization efforts revolve around approximations techniques and simplifications of the problem to make it computationally tractable. In most cases, the approaches developed are deterministic, thus a clear limitation arises due to the fact that the inherently stochastic nature of operations is disregarded. The novelty of this research is the study of large horizon scheduling problems with a dynamic fashion. The modeling of the stochastic variables is mainly identified in the aircraft utilization, the maintenance elapsed time, and the demand of non-routine tasks. The last one is especially important, because one of the biggest challenges of CBM is to deploy a technology able to predict failure and dynamically schedule the tasks related to the systems affected.

Therefore, approximate dynamic programming is selected as a valid methodology. The reasoning behind this choice can be broken down in three steps. Firstly, dynamic programming allows to solve a large combinatorial optimization by nesting smaller decision problems. Hence, the sequential nature of the problem is captured. Secondly, the stochasticity can be introduced using exogenous information in the state space representation. Lastly, the synergies between machine learning and dynamic programming make this methodology the best candidate to solve the stochastic maintenance task allocation problem. In order to narrow down the vast landscape of algorithmic possibilities, the most promising approaches have been decomposed in four branches: (1) value-function approximations are introduced to achieve forward induction, and range from neural networks to simple look-up tables; (2) the receding horizon approach can be used to tackle the large planning time window; (3) the decoupling of the fleet problem into several single-aircraft scheduling problems; and (4) state aggregation and discretization are used to avoid sweeping through states that may not produce beneficial solutions.

Although the purpose of this literature review is purely exploratory, we believe that the proposed methodologies foster the advancement of airline maintenance planning tools, to solve scheduling problems of stochastic complexity previously unreachable.

III

Research Methodologies
previously graded under AE4010

Executive Summary

The long-term maintenance planning is an interesting, yet unsolved, problem for the airline industry, but also for the operations research community. Preliminary maintenance schedules are created years ahead in the planning-horizon phase, especially the packaging of maintenance tasks occurs at this stage, and long overhaul checks are established. However, in the operational-horizon phase, non-routine tasks arise as unexpected events, and can produce expensive changes in the airline network. Condition-based maintenance (CBM) strategies aim to bridge the gap between these two realities that normally are treated separately. Despite of the potential economic benefit in optimizing the maintenance schedule, the challenge has received limited attention regarding the planning horizon perspective. This research advocates for pushing further the integration of machine learning techniques in the stochastic maintenance task allocation problem. An extensive review of the state-of-the-art methodologies is presented, the limitations of current approaches are addressed, and the value of stochastic dynamic optimization is conveyed. The outcome of the project will be a maintenance scheduling tool designed to capture unexpected events, such as aircraft utilization and random failures, that hopefully outperforms the current performance of the European airline investigated. More broadly, the results will be of interest to airlines, airworthiness institutions and MRO operators, enabling the next generation of maintenance schedules based on a data-driven technology.

1 Introduction

Throughout the years, aviation safety and operational efficiency have become a major concern to remain competitive in the airline industry. Maintenance is performed regularly in accordance to the international air safety requirements and the aircraft's manufacturer. It is a necessary process closely interconnected with the flight scheduling problem, especially because the aircraft under repair will remain in the hangar unable to fly passengers. Ten percent of flight delays and cancellations are currently caused by unscheduled maintenance events, costing the global airline industry an estimated 8 billion USD (Daily and Peterson, 2017). Maintenance operations expenditures correspond to 10% of the airline's direct operating cost, any marginal improvement can result in a significant benefit for the airline's profit (Lagos et al., 2020).

Condition-based maintenance is a strategy designed to transform real-time aircraft data into actionable intelligence to avoid unnecessary ground times and replacing periodic-based maintenance strategies. The approach to realize the digital future of the aviation maintenance is twofold. First, it is essential to deploy the use of advanced analytics to understand the driving factors of future performance, monitor the individual aircraft parts and predict remaining useful life (RUL). The second element revolves around how to plan maintenance and optimize schedules in a dynamic environment. The objective of this thesis focuses on the latter, by proposing and thoroughly evaluating an adaptive scheduling framework that enables maintenance tasks allocation in a dynamic environment. The framework will represent a substantial contribution to both the ADP systems and airline maintenance scheduling literature, bridging the gap of adaptive optimization in stochastic environments. Capturing the dynamism and planning constraints at hand poses novel challenges in ADP, and requires substantially extending state of the art optimization techniques from literature.

This research proposal is organized in the following fashion. Section 2 presents a summary of the state of the art in airline maintenance scheduling, in order to guide the reader from the vast array of literature to the research questions and main objective in section 3. In section 4, the theoretical background of adaptive learning methods and their application in a context of airline maintenance scheduling are described. Section 5 deals with the experimental set-up required to develop a maintenance scheduling platform that integrates the data sources with the methodology proposed. The expected results are described in Section 6, which also highlights the relevance and implications of the project. In Section 7 the logistics of the project are tackled and supported with a detailed planning. Finally, Section 8 concludes the project plan with a reflection on all aspects of the project plan.

2
Literature Review

The airline maintenance scheduling (AMS) problem involves two different paradigms. The first one, also called task-packaging, consists on grouping tasks in blocks according to their periodicity, required skills and duration. The second one is to assign each block of tasks to a maintenance opportunity. Further extensions of the problem should include the additional challenge of stochastic optimization with elements such as the maintenance elapsed time and the aircraft utilization, that will directly affect the task due date (Deng et al., 2020). Additional line maintenance opportunities, which are only known in the operational horizon, can also be used to extend the scope of this research.

2.1. Maintenance Planning

There are three main categories of airline maintenance based on their periodicity and duration (Yan et al., 2011). The first one, *long-term*, includes heavy maintenance tasks that require an overhaul longer than 10 days. They are also called level C and D checks. The *mid-term* maintenance plans are set monthly on the long-term plan. They incorporate both level A and B checks. Lastly, *short-term* plans produce schedule adjustments due to incidents. It is noteworthy that these checks are all performed in the hangar facility. Nonetheless, line maintenance is an additional category performed at the gate of the airport. There is not a unique and systematic approach to perform maintenance, and every airline has different guidelines to solve the problem. In fact, occasionally, different category checks are merged based on the slot opportunities and the aircraft's operational condition.

Traditional maintenance strategies opt to cluster a series of tasks in blocks based on the resources required (A,B,C,D checks). This methodology has declined in recent years, in favor of progressive maintenance checking. Modern maintenance planning approaches adopt a task-driven solution, where tasks are bundled together, such that the resulting check can be performed overnight while the aircraft is not in use (Muchiri and Smit, 2009). Ozkol and Senturk (2017) developed maintenance task packages for a fleet with over two thousands tasks based on the task source, the personnel requirement, dependency of tasks, and inventory items. Task-packaging is an important aspect of phased checks in order to increase the aircraft availability and introduce flexibility in the way tasks are grouped, rather than having long aircraft downtimes and sporadic manpower requirements as in block checks. Nowadays, it is more common to divide these higher checks into segmented clusters or phases. The main reasons are to reduce aircraft downtime, to have a flexible grouping of tasks, and to optimize the schedule. On the other hand, the planning complexity is highly increased, and the reduced slot capacity may represent a risk when considering non-routine tasks (Shannon and Ackert, 2010).

2.2. Condition-based maintenance

The transition to condition-based maintenance (CBM) combines preventive and predictive strategies that aim to reduce maintenance costs, yet providing services with an improved quality and reliability. Furthermore, disruptions due to maintenance faults are reduced, and more likely to be foreseen without causing unexpected aircraft downtimes. Through condition-based information, maintenance can be performed only when necessary and it becomes possible to identify and perform repairs before damage occurs. The prognostic and health monitoring (PHM) systems are employed to decide when is the most optimal time to allocate the task and avoid waste of remaining useful life (RUL). *Preventive maintenance* strategies are used nowadays to execute recurrent maintenance tasks, and will remain necessary for an effective transition. It both serves as a fail-safe for predictive maintenance and to globally assess the status of a system in case of false-positive

predictions. They use a combination of time-driven intervals, based on calendar days, and usage-driven intervals, based on the cumulative aircraft flight hours and cycles (Kinnison, 2004).

The integration of PHM techniques with maintenance decision represents the basis of the CBM concept (Vianna and Yoneyama, 2018). Hölzel et al. (2012b) present the disruptive potential of PHM systems to reduce both, *operational interruptions* caused by unscheduled maintenance tasks and *downtimes* due to unnecessary preventive maintenance. Literature suggests that only a half of the total tasks required by heavy-maintenance can be planned, the remaining half remains unscheduled until the inspections (Kulkarni et al., 2017).

Condition-based maintenance (CBM) strategies requires the aircraft to be completely wired and equipped with sensors to monitor the real-time health of the fleet. To date, various airline operators have implemented certain aspects of CBM in their operations, that have dramatically increased the level of electronic integration in aircraft's systems (Teal and Sorensen, 2001). However, none have taken full advantage of the CBM concept because of the lack of fully equipped aircraft with health monitoring sensors, the strict air safety requirements, and the lack of digitalization in the maintenance operations domain. In order to foster the implementation of CBM strategies in the industry, some hybrid approaches have been proposed in the research community where traditional scheduled maintenance is used for critical structures and condition-based maintenance for non-critical systems (Dong et al., 2019, ReMAP H2020).

2.3. Airline Maintenance Optimization

The majority of researchers that include maintenance in the scope of the optimization have focused on the aircraft routing problem (ARP) and the tail assignment (TA). The first one is concerned with the creation of lines of flight, while the second one assigns these *routings* to individual aircraft. Frequently, these problems are solved in combination with maintenance considerations, in order to respect the maintenance schedule that requires the aircraft to be present at the hangar location. Since the early efforts of Feo and Bard (1989) a vast array of literature has formulated the ARP with a time-space network. Hane et al. (1995), Clarke et al. (1996), Clarke et al. (1997) and Gopalan and Talluri (1998) extend their formulation with maintenance and crew considerations. It is noteworthy that the problem is solved with an operational horizon that normally has a time-window of 3-4 days. Instead, the planning horizon that is investigated in this research extends to 3-8 weeks for mid-term checks until 1-2 years for the long-term strategical checks.

Sarac et al. (2006) is one of the few works that aims to maximize aircraft utilization subject to the resource constraints (available man-hours and maintenance slots). The author uses an innovative branch-and-price algorithm formulated with a route-based network including flight and aircraft coverage constraints, as well as the labor force hours and the slot availability. Along similar lines, Safaei and Jardine (2018) designed a multi-commodity network flow model with generalized maintenance constraints. The longest aircraft maintenance routing horizon is proposed by Sanchez et al. (2020) with a 30-day schedule. Moreover, there is more flexibility as any ground time represents a feasible opportunity. However, the author explores a different problem that concerns line maintenance instead of hangar checks.

Sriram and Haghani (2003) consider a 7-day planning horizon assuming a cyclic flight schedule for an heterogeneous fleet with maintenance constraints. The maintenance checks are dependent on the aircraft utilization, thus also on the flight sequence. The scope of the problem is limited by the planning horizon which allows only A and B checks to be considered. Steiner (2006) presents a heuristic method for maintenance scheduling with the objective of minimizing maintenance actions and evenly distribute among the fleet the slot capacity availabl and flying hours. Around the same time, Afsar et al. (2006) proposes a rolling horizon heuristic to solve the maintenance planning problem for an airline with 200 aircraft. The planning is produced for a ten-weeks period with a sliding time window of one week. The heuristic first addresses critical aircraft that must undergo maintenance in the planning week and subsequently manages the non-critical aircraft load smoothing with simulated annealing (Afsar et al., 2009).

Lastly, Deng et al. (2020) developed a look-ahead dynamic optimization model to solve the aircraft maintenance scheduling problem of a European airline. The problem size is determined by a 4-year planning horizon and an heterogeneous fleet of more than 40 aircraft. Two types of phased checks are considered in the model which theoretically should involve all the tasks provided in the maintenance planning document (MPD). In order to reduce the size of the problem the aircraft are ordered by maintenance urgency based on the earliest deadline. A *thrifty algorithm* is used to check the *workability* of a state by making sure that no

aircraft will have to be grounded in the worst case scenario, which makes use of all available slots.

2.4. Approximate Dynamic Programming approaches

The key idea of Dynamic Programming (DP) is dividing the problem into states and *nesting smaller decision problems inside a larger decision* (Dreyfus, 2002). The Markov Decision Process (MDP) is used to formalized the dynamic programming model in DP (Busoniu et al., 2010). Even though classic DP is an important theoretical concept, it is almost impossible to use in large combinatorial problems. Therefore, to make a problem computationally tractable, approximate models introduce elegant theories to reach near-optimal solutions. The algorithmic strategy proposed by Approximate Dynamic Prpgramming (ADP) involves *forward induction*. This optimization technique only updates visited states, but it requires to have an estimation of the values of states that *might* be visited. It replaces the computational burden of looping backwards through each possible state to obtain the exact value in classic DP, with the statistical problem of *estimating* the state-value. This is also known as the first curse of dimensionality that arises due to the state space. Powell et al. (2012) defines the *value function approximation* (VFA) as a policy determined by an explicit function that indicates the optimal action to take when the state space is too large. Mathematically it is expressed as the expected gain given the current state s. The value-function can be decomposed in *imminent rewards* and *discounted rewards*, that estimate the value of future states (Littman et al., 2013). Value function approximations can be presented in different flavors, this terminology include simple look-up tables to parametric functions, like a linear regression or a neural network, as well as non-parametric functions, like Gaussian kernel regressions (Powell and Topaloglu, 2006).

There are two ways to solve stochastic dynamic problems. The first one uses statistical distributions to model the range of possibilities (including slack), and solve samples of the model deterministically. This technique is also called *sampled approximation* because the objective function tries to maximize the *sample average approximation* instead of the expectation in the Bellman equation (Bellman, 1972). The second method includes the stochasticity in the modeling. Therefore, for the same state-action combination, the response of the model varies based on the *exogenous information*, by this term we refer to external information that is uncertain until the next action is taken. These approaches are referred to as *adaptive-learning models*, they use iterative methods to solve sequences of relatively easy problems, with the hope that the algorithms will converge to the solution of the original problem. All sequential learning algorithms, for any stochastic optimization problem, can ultimately be reduced to a sequential decision problem, otherwise known as a *dynamic program* (Powell, 2011).

In order to narrow down the vast landscape of algorithmic possibilities, the most promising approaches have been decomposed in four branches: (1) value-function approximations; (2) the receding horizon approach can be used to tackle the large planning time window; (3) the decoupling of the fleet problem into several single-aircraft scheduling problems; (4) state aggregation and discretization are used to avoid sweeping through states that may not produce beneficial solutions. The first one, *VFA*, has been widely studied in the works of Nazari et al. (2018), Simão and Powell (2009), Simão et al. (2008) to solve a dynamic fleet management, an inventory optimization and a stochastic vehicle routing problem respectively. *The rolling horizon* has been implemented with Monte Carlo iterations in the work of Lagos et al. (2020), Ulmer (2020) and Deng et al. (2020). *The decoupling of the fleet problem* into parallel or sequential stages is proposed based on the airline strategical diversity that exists between different aircraft subtypes and check types. *State aggregation and discretization* are general approaches used to reduce the computational requirements of a model (Deng et al., 2020, Powell and Topaloglu, 2006). Finally, within the reinforcement learning community the approaches that stand out are *model free* due to the difficulty in learning the large dynamics of maintenance operations, that would be required for a model-based algorithm. The most attractive approaches that have been identified are deep Q-learning (Mnih et al., 2015), actor-critic (Mnih et al., 2016), and neural combinatorial optimization (Bengio et al., 2020).

2.5. Synthesis, relevance and positioning of project

In this literature review, the airline maintenance scheduling problem has been presented alongside the models that include maintenance considerations. The clear lack of automated planning tools for maintenance scheduling arises from the fact that it is a long-term problem in a dynamic environment where factors such as the aircraft utilization, the maintenance elapsed time and the non-routine task demand are continuously

changing. Moreover, the literature mainly focuses on the operational horizon by solving the aircraft routing problem with maintenance constraints. Therefore, there is no flexibility in the maintenance scheduling and a pre-existent maintenance plan is necessary. The main takeaway of this chapter is that there is barely no work on the maintenance allocation problem, however, the literature models can be adapted and extended. It is evident from the analyzed literature that two main challenges are yet to be addressed: (1) the models grow in exponential size as the planning window increases, and (2) stochastic factors are not included in the models, so the optimization takes a static environment and a deterministic approach. Therefore, it is deduced that both reasons indicate that standard optimization techniques, like commercial solvers, would not be able to cope with the size of the problem and produce useful results for a dynamic environment. In order to include the inherent uncertainty of maintenance operations, the state-of-the-art scheduling approaches are reviewed and considered for the scope of this research. Based on the sequential nature of dynamic programming and its synergy with machine learning, adaptive-learning methods are proposed as a suitable algorithmic strategy to solve the problem addressed in this thesis project.

3
Research Objectives and Questions

The main research objective of this thesis is the following:

Develop a maintenance scheduling methodology with an approximate dynamic programming approach to enable condition-based maintenance in a fast and adaptive manner, and to deliver a near-optimal maintenance schedule.

The main research question to be answered in order to fulfill the research objective is stated below.

What approximate dynamic programming approach can enable adaptive maintenance scheduling, and how should the stochastic elements be modeled in the optimization?

In order to achieve the objective and answer the main research question, the SMART framework is chosen to formulate specific, measurable, achievable, realistic and time-specific project objectives. The following four sub-questions are identified and broken down further:

1. What are the current practices in airline maintenance scheduling (AMS)?
   (a) What are the airworthiness requirements related to maintenance?
   (b) What is the economic impact of maintenance in the airline industry?
   (c) Can maintenance scheduling be divided in multiple phases?
   (d) What are the task features considered when creating maintenance work packages?
   (e) Which simplifications to the AMS problem are acceptable within the industry practice?
2. How can a CBM context be exploited during maintenance task packaging and allocation?
   (a) Can a CBM strategy be applied while keeping a traditional block check structure?
   (b) Is it possible to create smaller work-packages instead of fixed periodic blocks?
   (c) How does the size of the maintenance work-packages affect the schedule optimization?
   (d) How much access synergy can be obtained by packaging tasks prior to scheduling?
   (e) Does the condition monitoring technology exist to allow a transition to a fully automatic and digital maintenance planning tool?
3. What are the current models for airline maintenance scheduling?
   (a) What models have already been implemented in the airline industry?
   (b) Are maintenance task packaging and allocation considered in the optimization models?
   (c) What are the limitations of the current strategies?
   (d) What assumptions can be made to the maintenance scheduling model?
   (e) What industries face similar problems?
4. Can approximate optimization techniques produce near-optimal maintenance schedules?
   (a) What approximation models can be used?
   (b) How can historical task allocation data be used to structure the optimization algorithm?
   (c) What are the block and task features that provide better results?
   (d) How should the quality-time trade-off be assessed during the validation phase?
   (e) What are the most relevant key performance indicators to measure the AMS performance?
   (f) How can the robustness of the framework be quantified and tested?
   (g) What validation models can be used to validate the scheduling algorithm?
   (h) What are the limitations of the proposed scheduling method?

4
Methodology

In order to answer the research questions in an exhaustive manner, the methodology has been structured based on the research objective and questions. These steps are fundamental to establish an ADP framework that will be tested in a real case scenario of a European airline that is providing the data for this research project.

4.1. Model of the environment and operating setting

It is of paramount importance to develop an environment that encloses the airline operations dynamics. The first and third research questions are investigated in order to accurately model the maintenance operations and adapt existing airline maintenance models to an ADP environment. As described in the literature, a dynamic programming approach is modeled as a Markov Decision Process (MDP) that have four essential components (Busoniu et al., 2010). The first one is the *transition function*, that establishes how each action changes the state of the environment. Therefore, it updates the resources (available slots) and the demand (aircraft competition), as well as the next due date of the rescheduling problem. Secondly, *the state* needs to be defined based on observable features that the agent shall be able to assess. Eventually, the agent will learn the most optimal policy based on this information received from the environment. For this reason, it is important to mathematically formulate the aircraft competition, the resources available, the future impact on the rescheduling tasks, and the availability of maintenance slots. In order to respect the *Markov* or *memoryless* property, the optimal policy should be merely dependent on the present state and all the previous ones can be disregarded (Alagoz et al., 2010). Thirdly, the *reward function* needs to be shaped. Each action that the scheduling agent takes, follows a local reward. It is important to transmit sufficient information to the agent in order to learn the value of being in a state. Lastly, the fourth element is the *action space* that defines the possible actions that the agent can take at each step based on the operational constraints. Due to the nature of the problem each action will be related to a scheduling possibility and it will be discrete. Since, a model-free approach has been chosen, the agent will not be aware of how each element is computed but it will simply observe the state that should contain all the relevant information to make a decision.

4.2. Model of the scheduling agent

The agent function is to learn an optimal policy that is able to map a state into a scheduling action. The knowledge-base of the agent is improved through iterations over the environment such that the internal function is calibrated to respond with an optimal action at every step. Therefore, the fourth research question, regarding the approximate optimization model, is intrinsically related to the agent behavior. A key aspect is to develop an agent that is able to interact with the environment. The sequential nature of the problem should be captured in order to take into account the weight of an action in the future. This means that both, the reward function and the state features need to be shaped based on the agent learning performance. Ideally, it should be possible to design a modular agent that is able to test different algorithmic strategies such as deep Q-learning, actor-critic and Monte Carlo iteration. Finally, the agent should be able to adapt to new environments it has never seen before. Therefore, it is quintessential to explore different scenarios during the training. The agent should learn the optimal behavior for every possible situation rather than only a specific fleet condition or maintenance slot configuration.

4.3. Stochastic modeling

Initially, the model developed will be deterministic in order to create a simpler case study and test the ADP frameworks. Therefore, the aircraft utilization will be fixed, and the due dates of each task will be the same at every iteration. Moreover, the non-routine tasks will not be considered explicitly, but it is assumed they can be included with a buffer time at every slot. Lastly, the unexpected failures will be excluded from the scope of the deterministic model such that due dates will be only dependent on the aircraft utilization.

The stochastic parameters that will be implemented in later stages are the aircraft utilization and the non-routine tasks. The first one can be modeled as a random variable sampled at each step of an episode based on historical information. The aircraft utilization will mainly affect the due date of a maintenance block. The second parameter is the non-routine task demand which can be modeled based on unexpected failures and maintenance elapsed time. Thus, it affects both due dates and tasks included in each block.

4.4. Task-packaging integration

The last step of the methodology proposed investigates a maintenance policy that goes beyond the traditional operational setting. In other words, it assumes a future scenario where the maintenance slot calendar is not designed to fit equal blocks in a periodic pattern. Therefore, smaller work-packages need to be designed in order to exploit more the interval of each task. Since the task with the earliest due date defines the block interval, it follows that the smaller the maintenance blocks, the more flexible becomes the scheduling framework. In order to evaluate possible task-packaging policies the guidance of the airline will be essential to produce realistic groups of tasks and maintenance slot calendars. Ideally, the task-packaging can be solved with simple rules that slightly alter the existing work packages. The resulting configurations will be tested with the ADP framework, and evaluated based on several key performance indicators such as the interval utilization, the number of groundings and the rescheduling instances of a task.

5
Experimental Set-up

The research project is conducted under the supervision and guidance of TU Delft and a European Airline. The airline facilitated maintenance operations data in order to develop a realistic framework with operational restrictions. The data necessary to develop a maintenance plan involves three different sources. The first one refers to the maintenance slot availability that indicates the opportunities for a fleet. Secondly, the maintenance planning document is essential to have a realistic number of tasks and a block configuration. Thirdly, the daily aircraft utilization is necessary to calculate estimations of the next due date and constrain the task intervals. Moreover, the airline provided historical task allocation data, to have a baseline for the optimization, and flight schedules, in order to leverage turn-around-times as line maintenance opportunities.

The environment features include the maintenance blocks to be performed, the fleet condition, and the maintenance slot availability. The agent will perform scheduling actions that will return a reward, the next state and the status of the episode (ongoing or finished). At every step, the environment is updated based on the aircraft utilization data and the tasks are ordered based on their priority and urgency. Figure 5.1 has been included to visualize the modules, and the general sequencing of a maintenance scheduling framework. The figure shows a general reinforcement learning structure adapted to the research topic, and it is possible to distinguish deterministic (blue) from stochastic (red) modules. The flowchart has been designed based on the methodological steps described in the previous section.

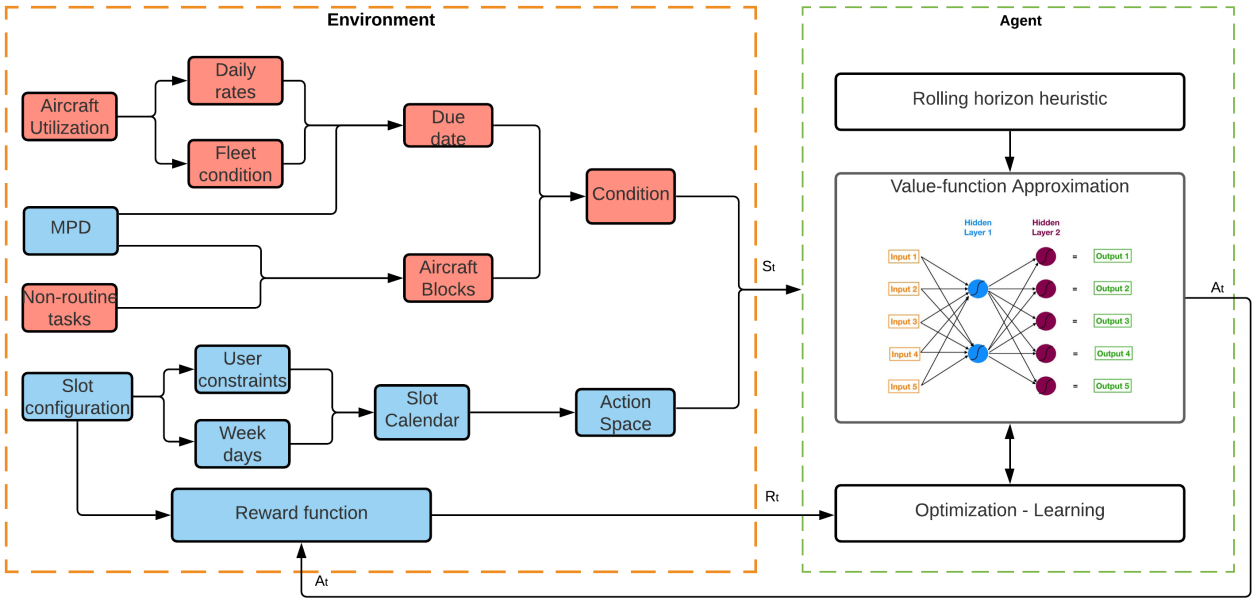


Figure 5.1: Experimental set-up of the reinforcement learning framework

The approximate dynamic framework of maintenance task allocation will be developed in Python 3.7. The main reason is because it is a programming language widely used for scientific research. Moreover, it is possible to exploit several built-in libraries that range from machine learning to optimization. The most attractive

libraries are *keras* to develop neural networks as parametric value-functions, and *openAI* that is used as inspiration to create a reinforcement learning environment. Furthermore, the simulations will be executed on a MacBook Pro machine with a 3.1 GHz Intel i5 dual-core processor and 16GB of RAM. All the data previously mentioned is downloaded from the WebDrive and stored locally in the computer. The main drawback is the limited computational power of the machine, however, it could be possible to test the final training performance on the TU Delft server that is directly linked with the WebDrive that the airline uses to share the data.

6
Results, Outcome and Relevance

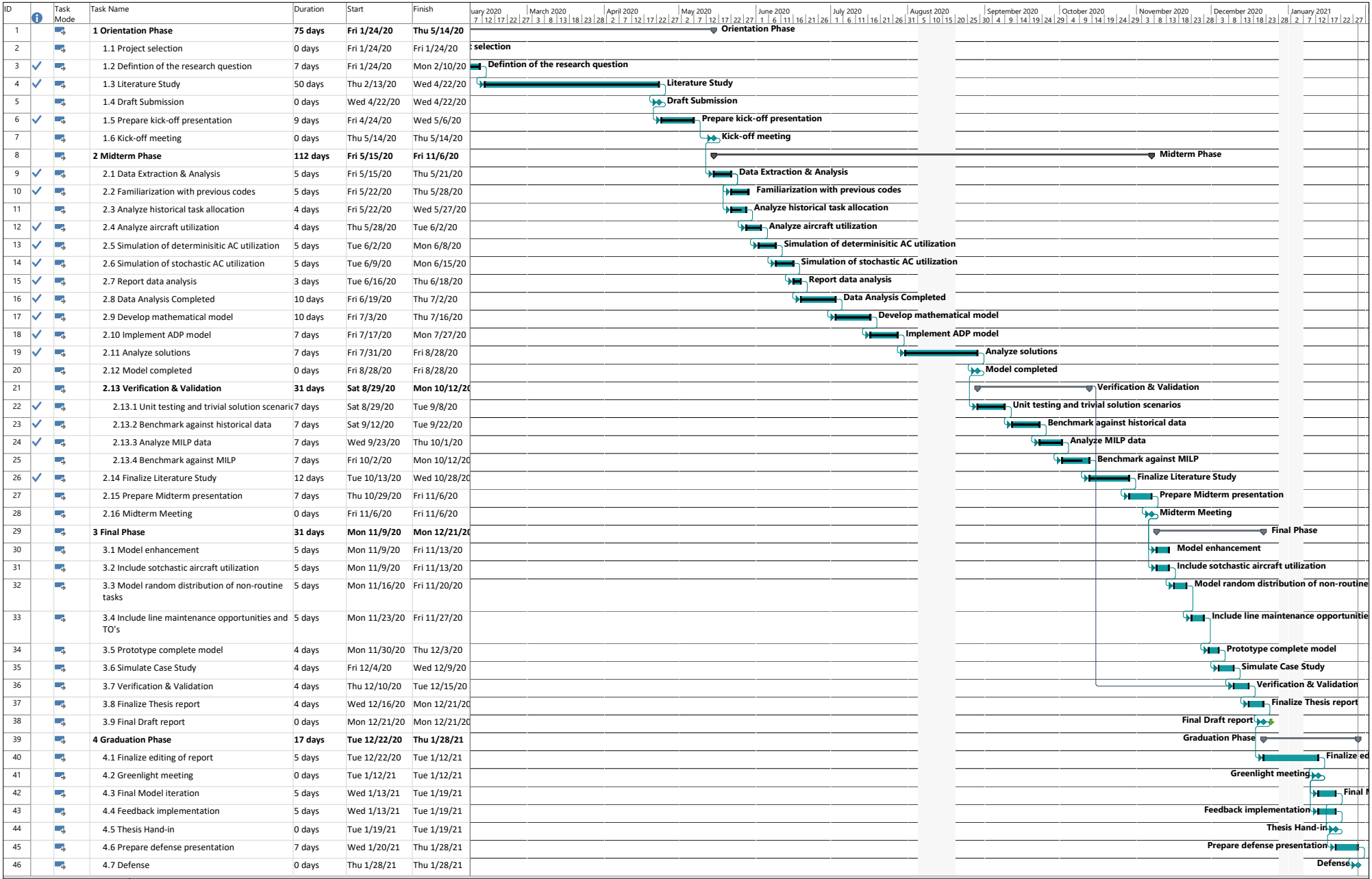
The analysis of the scheduling result is a core part of the project captured by research questions 2 and 4. The first question investigates the possibility of changing the task-packaging schedule of the airline with alternative configurations, while the second one refers to the optimization results achieved with an ADP approach. The outcome of the research project will be a novel decision framework for maintenance scheduling able to deal with stochastic variables, with the main goal of enabling a condition-based maintenance strategy. To the best of our knowledge, it is the first attempt to bridge the gap between long-term maintenance planning with the operational horizon, as well as the creation of a stochastic optimization for maintenance planning. The results can also be used by the European airline, to which the operations model has been tailored, to address the feasibility of a maintenance plan under certainty, as well as re-adapting the schedule due to unexpected events.

Verification and validation tests are necessary in order to ensure soundness of the results, establish the relevance of the research and its intersection with the maintenance planning and the ADP literature. Unit level test are essential to ensure that the individual blocks are working properly. For instance the due date calculation will be compared for known cases, the interval and aircraft utilization constraints will be computed for an input maintenance task and the performance of the agent will be tested with different cost functions. The verification at a system level will involve (1) robustness tests to ensure that small schedule changes lead to similar allocation results, (2) greedy tests to avoid an agent that oversees the importance of future performance, and (3) reduced scenarios with trivial solutions in which the optimal allocation is known. The validation will be performed based on the historical allocation, however, this comparison will provide only a limited understanding of the ADP performance due to flexibility that maintenance controllers have in practice and the maintenance plan changes over large horizons. Therefore, a second validation strategy has been proposed. The airline tool to schedule A-checks will be compared under the same conditions of the ADP framework. Even though only A-checks are considered in the scope, the validation result will obtain insights on the optimization performance of an ADP model compared to a commercial solver.

7
Project Planning and Gantt Chart

In the following page, a Gantt chart has been included to illustrate a complete schedule of the work to be conducted. The main steps described in the methodology have been broken down into several tasks and modules. Moreover, an estimate is made regarding the time required for the completion of each task. To date, most of the tasks up the midterm have been completed as indicated by the progress bars. These work-packages constitute the bulk of the thesis project, but several other steps such as revisions, iterations, report writing, and key meetings are also important parts of the project. The Gantt chart starts after the literature study with the kick-off meeting (14/05/2020) but it is still possible to look at the orientation phase activities on the leftmost column. The core modeling began after the kick-off meeting, which is the date on which the airline data became accessible after signing the non-disclosure agreement.

The project is organized in four main blocks. During the orientation phase, the literature study is performed together with the formulation of the research questions. In the midterm phase the data analysis and the model are developed, as well as the verification and validation of the deterministic model. It is noteworthy that most of the activities have been designed based on the research questions formulated in Section 3 and their chronological dependency. During the final phase, the model is reiterated based on the feedback received in the Midterm meeting (09/11/2020). Moreover, the aim is to enhance the model with the addition of stochastic variables such as aircraft utilization, non-routine tasks and variable task-packages. The graduation phase concludes the project with the submission of the draft report and the green light meeting (12/01/2021) before the last date (28/01/2021), when the defense will take place.



ID	Task Mode	Task Name	Duration	Start	Finish
1		**1 Orientation Phase**	**75 days**	**Fri 1/24/20**	**Thu 5/14/20**
2		1.1 Project selection	0 days	Fri 1/24/20	Fri 1/24/20
3		1.2 Defintion of the research question	7 days	Fri 1/24/20	Mon 2/10/20
4		1.3 Literature Study	50 days	Thu 2/13/20	Wed 4/22/20
5		1.4 Draft Submission	0 days	Wed 4/22/20	Wed 4/22/20
6		1.5 Prepare kick-off presentation	9 days	Fri 4/24/20	Wed 5/6/20
7		1.6 Kick-off meeting	0 days	Thu 5/14/20	Thu 5/14/20
8		**2 Midterm Phase**	**112 days**	**Fri 5/15/20**	**Fri 11/6/20**
9		2.1 Data Extraction & Analysis	5 days	Fri 5/15/20	Thu 5/21/20
10		2.2 Familiarization with previous codes	5 days	Fri 5/22/20	Thu 5/28/20
11		2.3 Analyze historical task allocation	4 days	Fri 5/22/20	Wed 5/27/20
12		2.4 Analyze aircraft utilization	4 days	Thu 5/28/20	Tue 6/2/20
13		2.5 Simulation of determinisitic AC utilization	5 days	Tue 6/2/20	Mon 6/8/20
14		2.6 Simulation of stochastic AC utilization	5 days	Tue 6/9/20	Mon 6/15/20
15		2.7 Report data analysis	3 days	Tue 6/16/20	Thu 6/18/20
16		2.8 Data Analysis Completed	10 days	Fri 6/19/20	Thu 7/2/20
17		2.9 Develop mathematical model	10 days	Fri 7/3/20	Thu 7/16/20
18		2.10 Implement ADP model	7 days	Fri 7/17/20	Mon 7/27/20
19		2.11 Analyze solutions	7 days	Fri 7/31/20	Fri 8/28/20
20		2.12 Model completed	0 days	Fri 8/28/20	Fri 8/28/20
21		**2.13 Verification & Validation**	**31 days**	**Sat 8/29/20**	**Mon 10/12/20**
22		2.13.1 Unit testing and trivial solution scenario	7 days	Sat 8/29/20	Tue 9/8/20
23		2.13.2 Benchmark against historical data	7 days	Sat 9/12/20	Tue 9/22/20
24		2.13.3 Analyze MILP data	7 days	Wed 9/23/20	Thu 10/1/20
25		2.13.4 Benchmark against MILP	7 days	Fri 10/2/20	Mon 10/12/20
26		2.14 Finalize Literature Study	12 days	Tue 10/13/20	Wed 10/28/20
27		2.15 Prepare Midterm presentation	7 days	Thu 10/29/20	Fri 11/6/20
28		2.16 Midterm Meeting	0 days	Fri 11/6/20	Fri 11/6/20
29		**3 Final Phase**	**31 days**	**Mon 11/9/20**	**Mon 12/21/20**
30		3.1 Model enhancement	5 days	Mon 11/9/20	Fri 11/13/20
31		3.2 Include sotchastic aircraft utilization	5 days	Mon 11/9/20	Fri 11/13/20
32		3.3 Model random distribution of non-routine tasks	5 days	Mon 11/16/20	Fri 11/20/20
33		3.4 Include line maintenance opportunities and TO's	5 days	Mon 11/23/20	Fri 11/27/20
34		3.5 Prototype complete model	4 days	Mon 11/30/20	Thu 12/3/20
35		3.6 Simulate Case Study	4 days	Fri 12/4/20	Wed 12/9/20
36		3.7 Verification & Validation	4 days	Thu 12/10/20	Tue 12/15/20
37		3.8 Finalize Thesis report	4 days	Wed 12/16/20	Mon 12/21/20
38		3.9 Final Draft report	0 days	Mon 12/21/20	Mon 12/21/20
39		**4 Graduation Phase**	**17 days**	**Tue 12/22/20**	**Thu 1/28/21**
40		4.1 Finalize editing of report	5 days	Tue 12/22/20	Tue 1/12/21
41		4.2 Greenlight meeting	0 days	Tue 1/12/21	Tue 1/12/21
42		4.3 Final Model iteration	5 days	Wed 1/13/21	Tue 1/19/21
43		4.4 Feedback implementation	5 days	Wed 1/13/21	Tue 1/19/21
44		4.5 Thesis Hand-in	0 days	Tue 1/19/21	Tue 1/19/21
45		4.6 Prepare defense presentation	7 days	Wed 1/20/21	Thu 1/28/21
46		4.7 Defense	0 days	Thu 1/28/21	Thu 1/28/21

Project: Thesis_ProjectPlan_Dan
Date: Thu 3/11/21

Task | Split | Milestone | Summary | Project Summary | Inactive Task | Inactive Milestone | Inactive Summary | Manual Task | Duration-only | Manual Summary Rollup | Manual Summary | Start-only | Finish-only | External Tasks | External Milestone | Deadline | Progress | Manual Progress

8
Conclusions

Approximate dynamic programming approaches have the potential to unlock the benefits of condition-based maintenance policies. In order to capitalize on these benefits, airlines face a number of hurdles due to the lack of sensor technology and the unattempted optimization of maintenance schedules. The challenge considered in this research project is the latter. The literature study revealed that maintenance scheduling is a problem studied with short planning horizons and deterministic models. The need of stochastic optimization stems from the uncertainty involved in airline operations and the availability of large streams of maintenance data. To overcome this barrier, adaptive-learning methods have been identified as a promising algorithmic framework.

Several research questions have been formulated and a methodology was defined to address them, and fulfill the research objectives. The methodology considers four main modules. Firstly, the environment module encloses everything related to the simulation of airline maintenance operations. Secondly, the reinforcement learning agent model is created to optimize the maintenance blocks allocation. The third module requires the implementation of stochastic variables and a simulation with unexpected events. Lastly, the fourth module considers a modification to the maintenance work-packages in order to explore alternative configurations that may result in a better performance. The duration of the thesis project will be of approximately 9 months, excluding the summer and Christmas holidays, with the final defense at the end of January 2021. The majority of resources have been allocated to developing a reinforcement learning framework and thoroughly analyzing the maintenance scheduling optimization performance. The outcome of the project will be a maintenance scheduling tool designed to capture unexpected events, such as aircraft utilization and random failures, that hopefully outperforms the current performance of the airline investigated. More broadly, the results will be of interest to airlines, airworthiness institutions and MRO operators, enabling the generation of maintenance schedules with a data-driven technology.

IV

Supporting Work

1
Airline Maintenance Planning

The airline maintenance panning problem consists of two paradigms. The first one is the task-packaging problem that is related to the creation of maintenance work-packages. The second paradigm is the airline maintenance scheduling problem (AMSP), where the predefined maintenance blocks or work-packages are allocated to a limited set of maintenance opportunities. Modern airlines have several thousands of parts, systems and components that require the aircraft to undergo maintenance periodically after a specific usage interval, defined in either flight hours (FH), flight cycles (FC), or calendar days (DY) (Witteman et al., 2021). With the introduction of condition-based maintenance (CBM), some of the tasks of the aircraft maintenance program (AMP) are monitored by prognostic and health management (PHM) technology, i.e. sensors on board of the aircraft. Therefore, the notion of fixed interval is replaced by a probabilistic distribution of the task due date over the maintenance lifecycle. The task-packaging problem is tackled in a methodology that combines routine blocks predefined with a MILP clustering model and individual CBM tasks. The latter are continuously monitored with prognostics and must be allocated together within the routine blocks, also called A-checks, before their predicted due date. Figure 1.1 shows the conceptualization of the approach defined to solve the AMSP. In this chapter an overview of the reduced AMP considered for this work is provided. Moreover, the transition to condition-based maintenance (CBM) is presented in order to enable the next-generation of airline maintenance operations. Lastly, the CBM experimental scenarios and the routine block clustering policies are explained in more detail. The reinforcement learning model that solves the AMSP is presented in the following chapter.

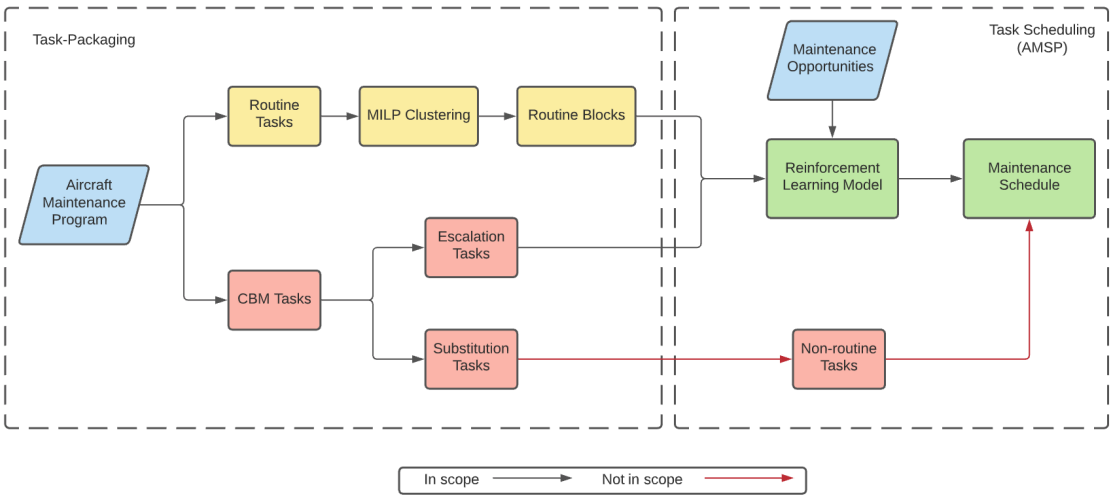


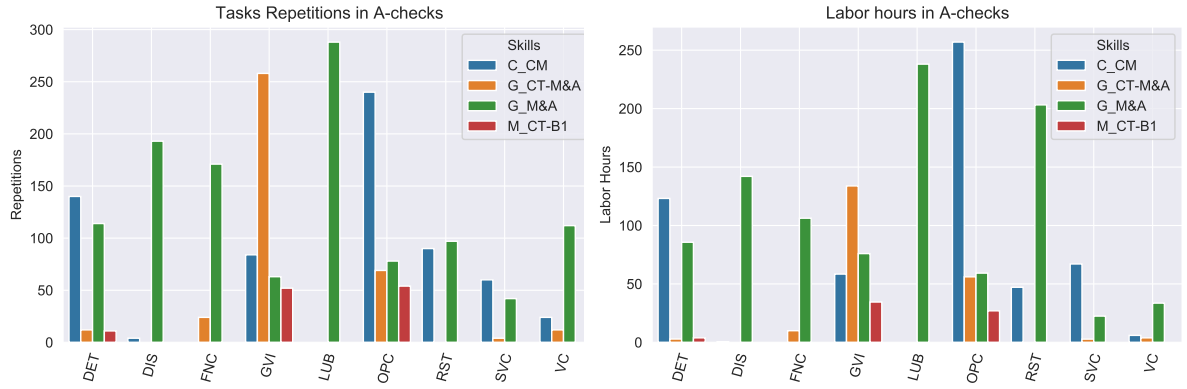
Figure 1.1: Approach overview

1.1. Aircraft maintenance program

The complete routine aircraft maintenance program contains 1086 tasks. These tasks belong to different check categories based on their interval limitation, the aircraft zone, the labor hours, the skills required, and other task inter-dependencies (Ozkol and Senturk, 2017). The focus of this research is centered around 186 routine tasks belonging to the A-check program. Therefore only 17% of the complete AMP is in the scope of this research. The C-check program is excluded because due to strategical reasons they are scheduled much

farther in advanced than A-checks. Since C-checks require a lot more labor hours, it is preferred to schedule them during low-demand periods and in sequence for the whole fleet, in order to leverage the learning curve effect of the engineers. Furthermore, smaller than A-checks and the non-routine maintenance program have been excluded because they have an operational planning horizon. Therefore, they make use of additional opportunities such as line maintenance or last-minute slots. For this reason, in Figure 1.1 non-routine tasks arising from substituted tasks have not been considered in the scope of this work. Since this research focuses on the tactical maintenance planning process, only A-checks have been considered in the scope of the CBM implementation. For a complete overview of the AMP used for this work, the reader is referred to Appendix A where all the tasks have been classified in their respective group with the interval limit specifications.

The A-check program defined by the reference European airline comprises a total of 24 maintenance blocks spaced by 1500 FH, 600 FC, and 120 DY. Therefore, it determines the aircraft maintenance lifecycle for approximately 8 years. The 24-block maintenance program comprises a total of 2296 task executions per aircraft. In Figure 1.2 the original A-check program has been taken as a reference to illustrate the different skills required per task group. On the left (Figure 1.2a) the total number of tasks repetitions has been reported with a classification per skill and task group, while on the right (Figure 1.2b) the amount of labor hours is visible. It is deduced that the skills that are required the most for the completion of the 24-block program are *G_M&A* and *C_CM*. Nonetheless, skills have not been considered for the creation of maintenance blocks when modifying the original A-check program. In order to simplify the problem, it has been assumed that the work-packages can be created exclusively on interval-based requirements and slot capacity constraints. This assumption should lead to different clustering results than the airline, however, it is safe to assume that their influence will be minor since, generally, maintenance work-packages have an homogeneous distribution of resources. Therefore, the A-checks composition could be modified in later stages to include additional operational requirements.



(a) Task Repetitions of original A-check Program diversified per skill and task group

(b) Labor hours of original A-check Program diversified per skill and task group

Figure 1.2: Overview of routine tasks in original A-check program

1.2. Daily aircraft utilization

The task intervals can be defined either in flight hours (FH), flight cycles (FC) or calendar days (DY). It is of paramount importance to account for the daily aircraft utilization because the first two limitations are usage based. Moreover, the fleet investigated is composed by 16 Boeing 787 which usually fly long haul routes. Consequently, the flight hour interval tends to be the most constraining factor for maintenance requirements. The average monthly utilization of five years has been considered to perform a statistical analysis over the complete fleet, as visible from Figure 1.3. The average daily rates are 16 flight hours and 1.85 cycles per aircraft. A seasonal pattern is observable with higher utilization rates in the summer and the Christmas periods. This pattern is captured in the simulation by calculating the task due dates with the respective aircraft utilization at each month. An additional remark is the generalization of equal daily rates for the complete fleet which gives place to another assumption: every aircraft is utilized by a constant monthly rate. In reality, each aircraft is utilized by a different amount of flight hours and flight cycles depending on the individual routings.

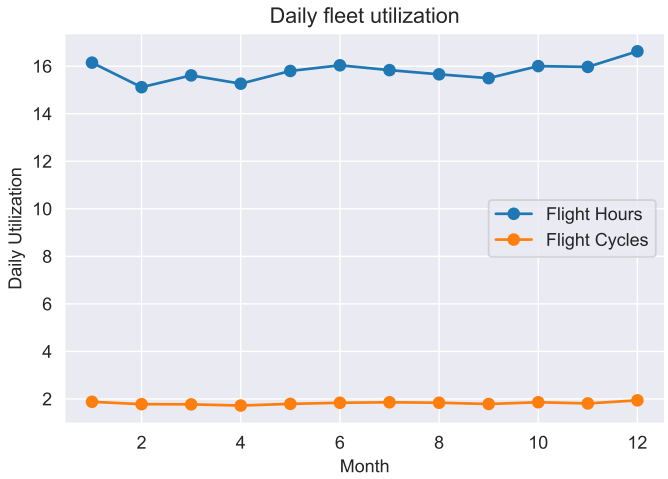


Figure 1.3: Daily aircraft utilization

1.3. The transition to CBM

Nowadays aircraft maintenance programs are mainly constituted by preventive and corrective tasks (Hölzel et al., 2012a). While the first ones are foreseeable and easy to plan, the actual due date of the tasks is dictated by a conservative interval limit to preserve safety. On the other hand, corrective maintenance tasks make full use of the life of a system but are more difficult to plan due to the short-term notice. The use of prognostics brings the benefit of extending the useful life of preventive tasks, and the ability of forecasting the due date of a corrective maintenance task before the respective failure occurs. The installation of PHM technology in the future generation of aircraft could reduce scheduled maintenance, and in turn increase the aircraft availability and utilization.

A large share of the maintenance activities in the AMP are expected to become obsolete and unnecessary thanks to the availability of aircraft condition data and remaining useful life (RUL) prognostics. Hölzel et al. (2012b) describes this maintenance strategy as prognosis- and task-based . However, there is not a clear symbiosis between the CBM task-based approach and the traditional letter-check program. Moreover, not all tasks will be monitored with sensors, and preventive maintenance strategies will remain a core part of the maintenance program in the transition period. Therefore, this work proposes a strategy that is able to incorporate an early adoption of PHM technology in the current maintenance planning practices, while keeping a letter-check structure. To overcome this barrier, a portion of the routine maintenance tasks are assumed to be monitored with a task-based approach; while the remaining routine tasks are clustered into maintenance blocks that are repeated in a predefined sequence based on fixed-interval requirements.

Table 1.1: CBM Scenarios

Task Group	**Nr.**	**CBM Action**	**Case 1 [%]**	**Case 2 [%]**	**Case 3 [%]**
General Visual Inspection (GVI)	41	Task substitution	0	10	25
Visual Check (VC)	9	Task Substitution	0	10	25
Detailed Inspection (DET)	21	Task Substitution	0	10	25
Functional Check (FNC)	18	Task Substitution	0	25	50
Operational Check (OPC)	34	Task Substitution	0	25	50
Servicing (SVC)	8	Interval Escalation	0	10	25
Lubrication (LUB)	18	Interval Escalation	0	10	25
Restoration (RST)	14	Interval Escalation	0	10	25
Discard (DIS)	23	Interval Escalation	0	10	25

The CBM task-based approach is twofold. Some tasks will follow a task interval **escalation** based on RUL prognostics, while the other part will be **substituted**. In the latter case, tasks such as inspections or visual

checks are completely removed from the routine maintenance program, and instead a sensor replaces those tasks that eventually will trigger a non-routine task. The experiments proposed in this work evaluate three different cases where the CBM action rate is varied as reported in Table 1.1. The baseline case only contains routine tasks, while the second and third case have a 10% and 25% CBM action rate. Each task group is affected by either escalation or substitution based on the nature of the tasks. For instance, an inspection can only lead to substitution, while lubrication or restoring tasks are affected by interval escalation. A higher action rate was applied to the Functional Checks and Operational Checks as they are heavily dependent on the acquisition of condition-data.

Table 1.2 and Table 1.3 provide a more detailed overview of the specific tasks that have been considered in the two CBM cases. The tasks have been classified per task group and their respective interval in either flight hours, flight cycles or calendar days. It is observable than in the 10% CBM case, the majority of tasks have 1500 FH interval. On the other hand, in the 25% CBM case a larger portion of tasks is included. Therefore, tasks that occur with less frequency have been considered in the CBM scope. The average task interval of Case 2 is estimated to be every 103.7 days, while for Case 3 is 148.1 days. Moreover, it is expected that the benefit of CBM is more relevant for tasks that occur with more frequency because their executions can be reduced by a larger share than the others.

Table 1.2: Case 2 (10% CBM) Tasks classified per group and interval duration

Task Group	**CBM Action**	**FH**	**FC**	**DY**	**Nr. Tasks**	**Labor [hrs]**
Operational Check	Substitution	1500	-	-	8	9.8
Detailed Inspection	Substitution	1500	-	-	2	1.5
Functional Check	Substitution	1500	-	-	1	0.4
General Visual Inspection	Substitution	1500	-	-	4	4.5
Functional Check	Substitution	2000	-	-	3	2.1
Discard	Escalation	1500	-	-	1	0.4
Restoration	Escalation	1500	-	-	1	0.2
Lubrication	Escalation	2000	-	-	1	1.4
Discard	Escalation	3000	-	-	1	1.5

Table 1.3: Case 3 (25% CBM) Tasks classified per group and interval duration

Task Group	**CBM Action**	**FH**	**FC**	**DY**	**Nr. Tasks**	**Labor [hrs]**
Operational Check	Substitution	1500	-	-	9	11.5
Detailed Inspection	Substitution	1500	-	-	5	4.0
Functional Check	Substitution	1500	-	-	1	0.4
General Visual Inspection	Substitution	1500	-	-	4	4.5
Visual Check	Substitution	1500	-	-	2	1.1
General Visual Inspection	Substitution	-	600	120	2	1.4
Functional Check	Substitution	2000	-	-	4	2.3
General Visual Inspection	Substitution	-	-	180	4	1.6
Operational Check	Substitution	-	-	180	3	1.8
Operational Check	Substitution	3000	-	-	5	1.5
Functional Check	Substitution	-	2000	360	3	1.7
Functional Check	Substitution	6000	-	-	1	0.5
Discard	Escalation	1500	-	-	1	0.4
Servicing	Escalation	1500	-	-	2	2.6
Restoration	Escalation	1500	-	-	3	2.4
Lubrication	Escalation	2000	-	-	1	1.4
Lubrication	Escalation	-	800	150	3	3.7
Discard	Escalation	3000	-	-	1	1.5
Discard	Escalation	4000	-	-	3	2.4

Figure 1.4 provides an overview of the influence of CBM on the original AMP. Since the maintenance program is different for every interval policy and CBM case, the original 24-block program with 2296 tasks repetitions has been used to have a general reference framework. The figure shows the portion of tasks repetitions affected by the implementation of a CBM concept with the three action rates proposed in the experiment set-up. In the first case, all the tasks are considered as part of a routine block. In the second case (10% CBM), the portion of substituted tasks corresponds to 18.8% of the complete maintenance tasks repetitions, while the escalated tasks only to the 3.7%. Since the fraction of the escalated tasks is so small, the scheduling performance should be lightly affected by RUL prognostics. The third case (25%CBM) raises the portion of substituted tasks to 39.5%, while the tasks executions that will be escalated correspond to 12.5% of the total.

Figure 1.4: Task Repetitions of original A-check Program classified with CBM action

1.4. Interval policies exploration

In order to support and integrate prognostics in an optimal way, a flexible maintenance planning process is considered to accommodate individual maintenance tasks that follow a CBM task-based strategy. Since a CBM task can only be allocated in the same slot as a routine block, the routine blocks are clustered with different interval policies to explore the benefit of having a varied frequency of maintenance opportunities. The analysis starts with the standard airline approach that spaces blocks by 1500FH and grounds the aircraft for a total duration of 24 hours. Then, the check frequency is increased and the maintenance slot duration is proportionally reduced. In this way, the CBM tasks dispose of more block opportunities where to be allocated. This hypothesis should lead towards a more flexible maintenance strategy that integrates CBM more optimally. A complete overview of the block clustering policies has been included in Table 1.4. It is noteworthy that the majority of tasks that are part of a routine block tend to be dephased from the interval of the routine block. In other words, most tasks are allocated to a block that is scheduled long before their due date, which leads to inefficiencies in the scheduling process. The explored interval policies are presented along with the average gap optimality of the three cases (0%, 10%, and 25% CBM) returned by *Gurobi* solver based on a computational time limit of 2 hours per case on an Intel Core i5 3.1GHz laptop with 16GB ram.

Table 1.4: Interval policies explored

Interval Policy			**N. Blocks**	**Ground time [hrs]**	**Gap optimality[%]**
FH	**FC**	**DY**			
1500	600	120	24	24	2.95
750	300	60	48	12	13.07
500	200	40	72	8	23.14
375	150	30	96	6	30.59

2
Reinforcement Learning

In recent years, reinforcement learning (RL) has gradually evolved to one of the most active research areas in machine learning, artificial intelligence, and neural networks (Sutton and Barto, 1998). Contrarily to classical machine learning approaches, RL does not require a dataset with labeled input and output for each element. Rather than inferring a pattern, RL is able to interact with an environment through a reward structure. In this way, an agent is able to explore the action-space and discover which actions yield the highest reward. Figure 3.3 represents the canonical agent-environment feedback loop by means of a Markov Decision Process (MDP). At every timestep, the agent observes a state S_t of the environment and it takes an action based on the internal policy $\pi(a|s)$. The objective of the agent is to calibrate the policy function based on the rewards received after performing each action. In the airline maintenance scheduling problem (AMSP), a policy is interpreted as selecting the best maintenance opportunity for an aircraft, given the fleet condition and the available resources. In this chapter, a general overview of the RL strategies is provided. Furthermore, the application of the chosen algorithm, Deep Q-Learning (DQL), is contextualized in the AMSP. Lastly, a benchmark with other well known RL models is provided to motivate further the choice of DQL.

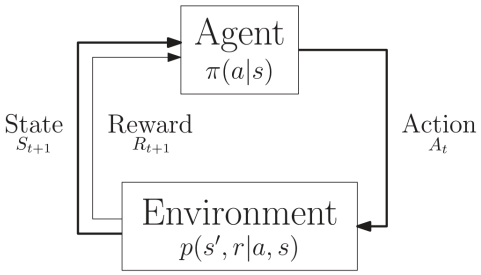


Figure 2.1: Markov Decision Process. Adapted from Sutton and Barto (1998)

2.1. Learning Strategies

The iterative nature of the problem allows multiple suitable approaches to develop the optimal policy $\pi(a|s)$ that maps a state s into an action a. Primarily, a distinction must be made regarding the search type and the methodology to update the agent policy. In this section the main algorithmic strategies to solve a dynamic program have been outlined.

2.1.1. Model-based vs Model-free

The reinforcement learning model is strictly related to the observable information at every state. The approach can be classified either as model-based or as a model-free scheme (Powell, 2011, Sutton and Barto, 1998). In the former case the agent learns a model of the environment thanks to an explicit transition function, and is able to predict the next state and reward value. Otherwise, when the transition function is implicit, model-free learning is employed to learn a dynamic policy. In that case, the agent requires to remember the past observations in a memory storage, which is then used for training. A clear understanding of the problem is required to determine which of the two is more apt to solve a particular combinatorial optimization problem.

Some systems are so complex that mathematical models are not able to represent them. However, it might be possible to observe behaviors directly. Such applications arise in operational settings where a model is running in production, establishing the outcomes observation and state transitions from physical processes rather than mathematical equations. Moreover, in many real world problems where uncertainty is considered, the assumption of a fully observable state cannot be maintained. In these cases, the agent has *partial observability* over the environment and the problem is be defined as a *Partially Observable Markov Decision Process* (POMDP) (Singh et al., 1994). In the field of dynamic programming, *model-free* refers to those systems that lack of an explicit transition function. Thus, a transition function cannot be learned, and an exogenous process is assumed to be available to generate observations and outcome of the system response (Powell, 2011). They are typically more flexible and thus more common in deep reinforcement learning but they require more samples in the learning process (Jin et al., 2018). On the other hand, *model-based* algorithms are highly dependent on the ability of representing the environment transition dynamics, typically expressed as a decision tree which expands exponentially as it branches out in time. Instead, model-free relies on sampling the experience gathered throughout the episodes and replaying a buffer memory during training (Polydoros and Nalpantidis, 2017). Therefore, model-free methods are considered more suitable for high-dimensional problems such as the AMSP.

2.1.2. Temporal Difference vs Monte Carlo

Sutton and Barto (1998) divides the model-free approaches in Monte Carlo and temporal difference (TD-learning). The first one requires to explore a full-sequence of actions to determine the value of a state. As shown on the right of Figure 2.2, the Monte Carlo tree search needs to arrive to the end of the episode in order to calculate the gain or sum of discounted rewards shown in Equation 2.1. On the other hand, temporal difference approaches perform *bootstrapping*, a technique that becomes relevant in high-dimensional optimization problems. During bootstrapping the state-value is updated based on *estimates* of successor state-values without the need to visit them first. The most obvious advantage of TD-learning is the ability to learn from each transition regardless of what subsequent actions are taken. Therefore, it is not required to wait until the end of the episode to obtain the state-value. However, the agent must provide an estimate of the value of the learned policy. Hybrid approaches combine a Monte tree search up to a certain level, after which they perform bootstrapping for the remaining states by means of a discounted reward approximation.

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

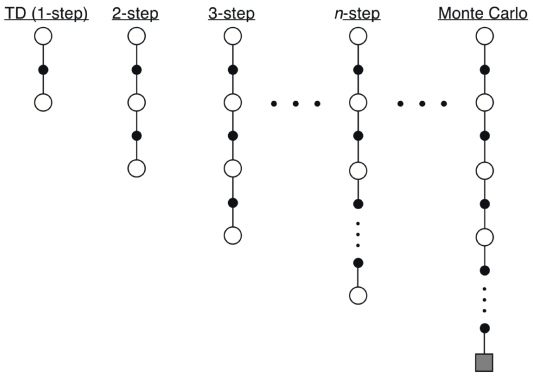


Figure 2.2: Temporal Difference vs Monte Carlo update (Sutton and Barto, 1998)

2.1.3. Value-based vs Policy-based

In reinforcement learning a major distinction is made between deterministic and stochastic polices. Value-based methods provide an algorithmic strategy to compute the value of next states. The policy is determined by choosing the actions that leads to the state with the highest value. Therefore, value-based methods are deterministic approaches. They leverage the Bellman Equation to calculate incrementally and iteratively the state-value function (Equation 2.2) or the action-value function (Equation 2.3). The policy is extracted by taking the action that maximizes one of the two functions.

$$v_*(s) = \max_{a_{t+1}} \quad \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] = \max_{a_{t+1}} \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s') \tag{2.2}$$

or

$$q_*(s, a) = \mathbb{E}[R_{t+1} + \gamma q_*(S_{t+1}, a') | S_t = s, A_t = a] = \sum_{s',r} p(s', r|s, a) \cdot [r + \gamma \max_{a'} q_*(s', a') \tag{2.3}$$

On the other hand, stochastic policies are formalized as the probability distribution of selecting a certain action with the objective of maximizing future rewards. The goal of policy estimation methods is to maximize a performance function $J(\pi_\theta)$ that corresponds to the sum of future rewards as shown in Equation 2.4, where π_θ is the learned policy with parameters θ. The maximization of the objective function is achieved by policy gradient algorithms that update the parameters θ in the direction of the performance gradient, shown in Equation 2.5. For a complete derivation of the policy gradient theorem the reader is referred to Silver et al. (2017).

$$J(\pi_\theta) = \mathbb{E}_{s,a\sim\pi_\theta}[r(s, a)] = \sum r_t(s, a) = G_t \tag{2.4}$$

$$\nabla J(\pi_\theta) = \mathbb{E}_{s,a\sim\pi_\theta}[\nabla_\theta log \pi_\theta(a|s) \cdot G_t] \tag{2.5}$$

2.2. Q-Learning

Value-based methods are widely used because of the simplicity of its implementation. These approaches suffer the dimensionality of large action spaces and are better suited for discrete actions. Since the scheduling problem is discrete by nature, a Q-learning algorithm has been devised to solve the AMSP. The algorithm was firstly developed by Watkins (1989) in a tabular form. A look-up table was used to store every possible state-action pair, which is iteratively updated based on the agent experience. The Q-value of each state-action pair is computed by means of Equation 2.6 at each timestep, where α is the learning rate of the model. The update is performed with a 1-step TD learning methodology because the discounted reward sum of the following states is contained in the Q-value.

$$Q^{new}(S_t, a_t) \leftarrow Q(S_t, a_t) + \alpha \left[R_{t+1} + \gamma \min_{a_{t+1}} Q(S_{t+1}, a_{t+1}) - Q(S_t, a_t) \right] \tag{2.6}$$

The algorithm makes use of an ϵ-greedy policy to decide which action to evaluate in the training procedure. The underlying principle of this policy chooses a random action with probability ϵ, and the rest of the times it uses a separate policy, which is continuously improved, to pick the action that maximizes the Q-value. It is an off-policy algorithm because it learns the value of an optimal policy independent of the agent's actions (Odonkor and Lewis, 2018). The algorithm makes no attempt to learn the underlying dynamics of the environment. In the case of this research, this refers to the transition function used to update the slot resources and the fleet utilization. Hence, Q-Learning can be labeled as value-based model-free methodology.

2.3. Deep Q-Learning

The Q-Learning tabular version developed by Watkins (1989) presents a major limitation because it requires an exhaustive representation of all possible state-action pairs. The state-space dimensionality of Q-Learning can be circumvented with a neural network. The input size has to be determined, but not the scale. Therefore, having a functionality to capture the Q-value allows the algorithm to be a lot more efficient (Mnih et al., 2013). This approach, called Deep Q-Learning, has been successfully applied in literature in problems ranging from optimal control to resource allocation. An artificial neural network (ANN) is constructed by multiple layers that contain a series of neurons. Each of the neurons act as a coefficient of the parametric model that has value θ_i and consist of a non-linear transformation function. The sequence of these transformations leads to learning different levels of abstraction from which the optimal policy is inferred (François-Lavet et al., 2018). The mathematical operation that takes place in each hidden layer of an ANN is a simple matrix multiplication shown in Equation 2.7, where h are the output values of the hidden layer, A is the activation function or non-linear transformation, and b a bias term.

$$h = A(\theta \cdot x + b) \tag{2.7}$$

The input of the neural network is the state S_t and it has a fixed output layer with $|\mathscr{A}|$ neurons, each representing the $Q(S_t, a_t, \theta)$ value from which it can be extracted the optimal policy. There are two main components in the original DQN proposed by Mnih et al. (2015). The first one is the target network with weights θ^- which is a delayed copy of the online network, such that $\theta^- \leftarrow \theta$ every τ steps. It is noteworthy, that the weights of the target network remain fixed during τ steps to improve the learning stability. In this way the online network is able to chase a stationary target value during τ steps. The purpose of the target network is to update the loss function. Equation 2.8 shows how the value function is updated with the Bellman optimality equation based on the online network parameters θ, and the target network parameters θ^-.

$$Q(s_t, a_t; \theta) = (1-\alpha)\ \underbrace{Q(s_t, a_t; \theta)}_{\text{Online network}} + \alpha \left(R_t + \gamma \min_{a_{t+1}} \underbrace{Q(s_{t+1}, a_{t+1}; \theta^-)}_{\text{Target network}} \right) \tag{2.8}$$

The second element of this algorithm is the experience replay buffer $\mathscr{R}$, which is a memory where the agent stores the observed experiences. Then, at each timestep the agent randomly samples a batch of experiences to train the neural network. During the training phase, the weights θ of the online neural network are optimized with a stochastic gradient descent (SGD) algorithm to minimize the mean squared error of the loss function with respect to the weights θ of the online network (Hasselt et al., 2016). The complete algorithm has been outlined below, in order to provide a clear sequence of the steps occurring in the training loop of the Deep Q-Learning agent.

Algorithm 1 Deep Q-Learning

Initialize DQN with random weights θ
Initialize target network with DQN weights θ
for each $episode$ **do**
  Reset Environment
  while not done **do**
    get state s
    calculate action-values $Q(s, a, \theta) \quad \forall a \in \mathscr{A}$
    select best action $a = \text{argmin}_a Q(s, a, \theta)$
    receive reward r
    transition to next state s'
    store experience $< s, a, r, s' >$ in $\mathscr{R}$
    for batch in BatchSize **do**
      sample experience $< s, a, r, s' > \sim \mathscr{U}(\mathscr{R})$
      if s' is terminal **then**
        target = r
      else
        target = $r + \gamma \min Q(s', a', \theta^-)$
      end if
      calculate loss $L_\theta = (Q(s, a, \theta) - \text{target})^2$
      update DQN $\theta_{i+1} \leftarrow \theta_i + \alpha \frac{\partial L(\theta_i)}{\partial \theta_i}$
    end for
  end while
  if τ is 10 **then**
    update target network $\theta^-_{i+1} \leftarrow \theta_{i+1}$
    $\tau_{i+1} \leftarrow 0$
  else
    $\tau_{i+1} \leftarrow \tau_i + 1$
  end if
end for

The weights of the neural network define the behavioral policy of the deep reinforcement learning agent. They are updated with backpropagation via the SGD algorithm. The fundamental principle is to update the weights in the direction of the loss function gradient as shown in Equation 2.9. The backpropagation process is given in Equation 2.10 with the loss function gradient of the Deep Q-Learning model.

$$\theta_{i+1} \leftarrow \theta_i + \alpha \frac{\partial L(\theta_i)}{\partial \theta_i} \tag{2.9}$$

$$\theta_{i+1} \leftarrow \theta_i + \alpha \left(r + \gamma Q(s', a', \theta_i^-) - Q(s, a, \theta_i)\right) \cdot \nabla_{\theta_i} Q(s, a, \theta_i) \tag{2.10}$$

2.3.1. State Space

The state space is the mathematical formalization of the maintenance scheduling environment. It provides the agent with six observable features of the environment based on which the optimal decision is taken. The Markov property, also known as the memory-less property, assumes that the optimal action is merely dependent on the current state (Alagoz et al., 2010). Therefore, the agent should not require any past state information to reach the optimal scheduling action . To satisfy this requirement, a number of features have been provided to the agent for every possible allocation slot that capture the essence of maintenance operations. Although the assumption of a *fully observable* state is required by the MDP, it can be arguably discussed that the state is not Markovian due to inherent uncertainties involved in the AMSP subject to prognostics uncertainty. Nevertheless, we assume that prognostics are sufficiently reliable to schedule the aircraft before their actual due date. In this way, Deep Q-Learning is applicable to a stochastic environment.

The model works in a sequential fashion by scheduling at every timestep the aircraft routine block or CBM task with earliest due date. The interval of the most critical scheduling unit is discretized in 100 units at every timestep. Then, the state features are calculated *for each of the discretized sub-intervals*, if a slot is present. Otherwise, the value of the features is null when there are no slots available. Thus, the state vector is composed by 100 rows that correspond to the discretization levels of the scheduling unit, and six columns that represent the features of the environment. In the remainder of this section a detailed explanation of each feature is provided. Lastly, a reduced scenario example is employed to demonstrate a sample calculation of the state vector.

Demand ($D_{t,a}$)
The demand is defined as the number of aircraft competing for a slot. Although the complete fleet is continuously competing for every slot, only the aircraft that have been utilized by more than 70% of their interval are considered as part of the demand for a slot. The feature takes 70% interval utilization as a baseline since any aircraft scheduled below this value is considered to have a poor performance.

Resources ($R_{t,a}$)
The resources are the remaining maintenance opportunities available to the competing aircraft, i.e. the scheduling units demand. Therefore, this feature provides an overview of the alternative options to the fleet when a specific slot is being considered.

Look-ahead function ($lh_{t,a}$)
The look-ahead function provides an estimation of the competing aircraft's average reward when a slot is encountered. Let Figure 2.3 be a reduced version of the AMSP environment. The most urgent scheduling unit is on top of the aircraft list and highlighted in green. The agent has four possible actions available based on the discretization level shown in the figure. The first option includes two possible slots. In this case, the latest slot which is on Friday is considered for the allocation.

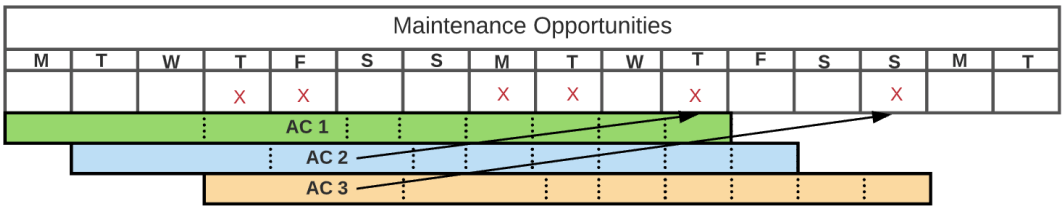


Figure 2.3: Look-ahead mechanism

The look-ahead reward provides an overview of the consequences to the rest of the fleet if $AC1$ is allocated to that slot on Friday. The working logic of the look-ahead function is to allocate each aircraft, in the priority order shown, to the latest available slot. Even if the optimal policy might not be dictated by a myopic policy such as this one, this methodology provides the average reward per aircraft and it is a way to capture the consequences of an action for the rest of the fleet. Moreover, this look-ahead mechanism reflects at each state if a competing aircraft might be grounded because there are not sufficient slots; or if a future aircraft will have to be allocated to a much earlier slot than its due date because of sub-optimal allocations previously made.

Time-index ($h_{t,a}$)
The time index variable indicates the position with respect to the simulation horizon of each discretized unit. Therefore, if a specific slot is available 6 months after the start of the simulation, and the simulation horizon is 12 months in total, the time-index feature will be equal to 0.5

Prognostic probability ($p_{t,a}$)
The prognostic probability is included in the state vector only when a CBM task is considered. The due date of CBM tasks is dependent on the prognostic curve. Thus, this feature is included to capture that a slot with a high probability of failure should be avoided. The agent should try to re-evaluate a prognostic at the next iteration when it is closer in time to the due date such that less uncertainty is involved in the prediction.

Task-Block feature ($b_{t,a}$)
The task-block feature is a binary variable that indicates the type of scheduling unit with earliest due date. This might be either a *routine block* that needs to be allocated to a free slot, or a *CBM task* that needs to be allocated in a slot where a routine block of the same aircraft was previously scheduled.

Example state-space calculation
The reduced environment proposed in Figure 2.3 is used as reference to calculate the state vector features. Since the time-index, the prognostic probability and the task-block feature are quite straight forward, the focus of this example is on the first three features: the resources, the demand, and the look-ahead reward.

The scheduling unit $AC1$ has the earliest due date. For this reason, it is selected as the most critical one. Moreover, it is observable how the block intervals have been discretized in 8 different units, only for this reduced scenario. In reality, the vector has 100 rows but for the sake of simplicity it has been reduced to 8 rows for this example. The reader can assume that the first row spans from 0-25% interval utilization, the second row from 25-40%, and the remaining rows go from 40%-100% interval utilization in steps of 10%. Equation 2.11 shows the corresponding state-vector of the reduced scenario.

$$S_t = \left[\begin{array}{cccccc} D_{t,a} & R_{t,a} & lh_{t,a} & h_{t,a} & p_{t,a} & b_{t,a} \\ \hline 0 & 0 & 0 & - & - & - \\ 1 & 5 & lh_{t,40} & - & - & - \\ 0 & 0 & 0 & - & - & - \\ 0 & 0 & 0 & - & - & - \\ 1 & 5 & lh_{t,70} & - & - & - \\ 2 & 5 & lh_{t,80} & - & - & - \\ 0 & 0 & 0 & - & - & - \\ 3 & 6 & lh_{t,100} & - & - & - \end{array}\right] \tag{2.11}$$

The first two columns indicate the amount of scheduling units competing at each sub-interval, and the total resources available to those aircraft. It is noteworthy that in row 6, corresponding to the 80% interval utilization of $AC1$, there are only two aircraft competing. The additional aircraft is AC2 because this slot on Tuesday is at 70% of its interval. $AC3$ is not included because this slot is at 50% of its interval. The second column corresponds to the number of slots available during the interval of the competing scheduling units. Lastly, the third column is calculated with the logarithmic reward function explained in Part I of this work. The four lh values are worked out in the following equations. There are three terms corresponding to the reward of AC1, AC2 and AC3, respectively. Every time the look-ahead considers two elements: (1) the reward of AC1 when assigned in the slot corresponding to the row of the vector; (2) the reward of the remaining two aircraft based on a myopic policy as shown in Figure 2.3. In the first three equations, AC 2 is assigned to the latest slot

available at 90% of its interval. In the last case ($lh_{t,100}$), AC 1 is assigned to the slot on the second Thursday, and therefore AC 2 is moved to an earlier slot at 70% of its interval.

$$lh_{t,40} = \frac{1}{3} \cdot \left(\ln\left(\frac{100}{40}\right) + \ln\left(\frac{100}{90}\right) + \ln\left(\frac{100}{100}\right) \right) \tag{2.12}$$

$$lh_{t,70} = \frac{1}{3} \cdot \left(\ln\left(\frac{100}{70}\right) + \ln\left(\frac{100}{90}\right) + \ln\left(\frac{100}{100}\right) \right) \tag{2.13}$$

$$lh_{t,80} = \frac{1}{3} \cdot \left(\ln\left(\frac{100}{80}\right) + \ln\left(\frac{100}{90}\right) + \ln\left(\frac{100}{100}\right) \right) \tag{2.14}$$

$$lh_{t,100} = \frac{1}{3} \cdot \left(\ln\left(\frac{100}{100}\right) + \ln\left(\frac{100}{70}\right) + \ln\left(\frac{100}{100}\right) \right) \tag{2.15}$$

Lastly, it is of paramount importance to consider that the gradient of the Q-value depends on the state vector. Therefore, the normalization of the state vector is performed to stabilize the learning curve. In this way, the gradient of the neural network weights becomes smaller, and it should be easier to optimize the reinforcement learning policy. The normalization of the features is performed to bound each value in the interval [0,1]. The demand and the resources are normalized with respect to the fleet size. The look-ahead results are normalized with respect to the highest average result of each step. Lastly, the time-index, the prognostic probability, and the task-block feature are already bounded in the desired interval.

2.3.2. Action Space

The action space represents the interval utilization rate at which the most critical aircraft is scheduled for maintenance. The neural network produces 100 values that correspond to the aircraft utilization percentage between 1%-100%. Nonetheless, not all actions are possible to take, but only those where a slot is present. Otherwise, the agent would be allocating an aircraft to a day where there is no slot available. In Equation 2.11 the feasible action-space can be understood as the rows that are not null in the state. Furthermore, the action space of the Deep Q-learning model requires a fixed size because it determines the size of the ANN output layer. Even if the DQL model calculates the action-values for the null rows of the state, the policy is forced to only pick one of sub-intervals that contain a maintenance opportunity.

Moreover, the agent has an additional action that is always available. This option corresponds to an aircraft on ground (AOG), and it should be taken only if no slots are available to the model in order to highly penalize the agent when an aircraft is left with no other opportunities. For the problem investigated, this results in an action-space with a size of 101 elements, and the following notation:

$$\mathscr{A} = \left\{ 1\%, \ldots . 98\%, 99\%, 100\%, \text{AOG} \right\} \tag{2.16}$$

2.3.3. DQN Architecture

The DQN architecture is designed in function of the state-space and the action-space of the AMSP. The state-space defines the size of the input layer, while the action-space determines the size of the output layer. A clear visualization of the DQN structure is shown in Figure 2.4 alongside the shape of each layer and the parameters that have to be optimized in the model. The model is composed by a total of five layers. The first one is labeled as the input layer and has the same size of the state vector and, for this reason there are no model parameters because it simply receives an input from the RL environment. The following two layers, also called dense layers, have 100 neurons each, and their output shape depends on the previous layer. The first dense layer has 700 parameters corresponding to 6 features times 100 weights plus 100 biases of each neuron ($6 \cdot 100 + 100 = 700$). Each neuron provides an abstract feature, therefore in the second layer there are $100 \cdot 100 + 100 = 10100$ parameters. Moreover, the flatten layer simply makes a one-dimensional vector out of the previous layer. Lastly, the output layer has a size of 101 neurons, exactly the same as the action-space. The total parameters of this layer are $10000 \cdot 101 + 101 = 1010101$.

The neural network weights are initialized with a normal distribution in order to reduce bias. Moreover, the activation function for each layer is called *swish* (Ramachandran et al., 2017). Instead, the last two layers are designed with a linear activation function because the DQN output represents the Q-value of the next-state and it should converge to the discounted reward value. The learning rate, the exploration decay and

the discount factor were selected upon a careful sensitivity analysis. For a complete overview of the neural network tuning and the optimal model configuration the reader is referred to chapter 4 of this report.

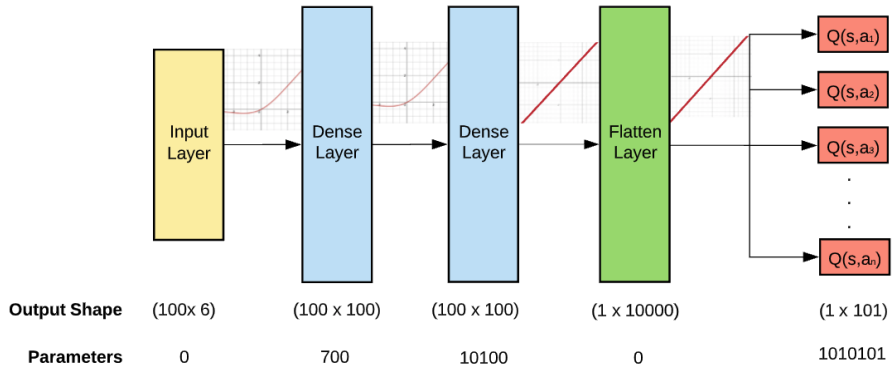


Figure 2.4: DQN Architecture

2.4. Training strategy

The interaction of the Deep Q-Learning model with the scheduling environment is explained in more detail in this section. The classic reinforcement learning feedback loop is tailored to the AMSP problem by means of the process shown in Figure 2.5. Initially, the environment is reset with the last maintenance execution dates and the complete list of tasks and routine blocks that form part of the aircraft maintenance program. The due dates of each aircraft are calculated based on the incremental utilization that is simulated at each timestep, and the scheduling unit with earliest due date is selected as the most urgent one.

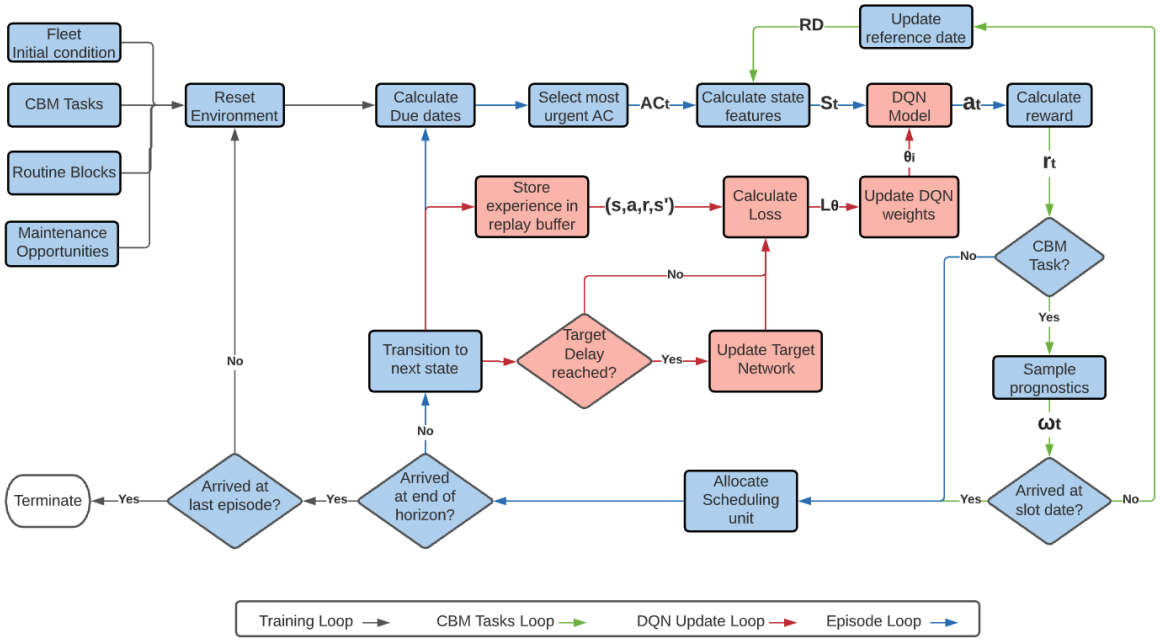


Figure 2.5: Training strategy of Deep Q-Learning model in the AMSP

In the following step, the state vector is calculated and fed as input to the DQL model. The DQN outputs a series of Q-values that represent the discounted cost sum of the available discretized actions to the agent. The best action is selected by taking the minimum Q-value in order to minimize the cost function. Furthermore, in the case that the model is processing a routine block, the scheduling unit is directly allocated. Alternatively, when a CBM task is considered, an additional loop is introduced in the agent-environment interaction that updates the state based on a new RUL prognostic. In this way, it is possible to synchronize the DQN decision model with the Gaussian propagation of the prognostic uncertainty. Once the scheduling unit is allocated, the transition to the next state occurs and the DQN update loop initiates. The blocks highlighted in red indicate the steps required to calibrate the neural network. Firstly, the target network is updated every 10 steps in order to ensure learning stability. Then, the experience observed at every agent-environment interaction step is sampled from a finite buffer memory. The sampling occurs in batches of 32 steps, where the stored experiences are fed as a tuple, with the form $< s, a, r, s' >$, and the loss function is calculated in order to minimize the deviation between the online and the target networks. The episode loop concludes when the simulation horizon is reached, while the training loop terminates after a sequence of 100 episodes.

2.5. Agents benchmark

Reinforcement learning provides a general framework to solve a specific problem as a Markov Decision Process. The agent-environment interaction can be adapted to any algorithmic strategy. This section explores other two well-known RL algorithms that were considered during the literature study of this work in Part II. Deep deterministic policy gradient (DDPG) belong to the policy-based methods, and asynchronous advantage actor-critic (A3C) is considered to bridge both worlds, policy-based and value-based, by means of two networks that improve each other. In general, these approaches are preferred to overcome the dimensionality of the action-space, however, they have a high variance and may converge to a local minimum. In Figure 2.6 the three agents have been benchmarked in the deterministic problem version that only schedules aircraft routine blocks. The DQN is the algorithm that behaves best in the AMSP. Moreover, the advantage of the other two algorithms is not beneficial for this case because the action space is relatively small: only 101 actions are available to the agent. Furthermore, A3C is particularly conservative when choosing a slot and prefers to schedule maintenance earlier which causes it to converge to a local minima. The DDPG algorithm has a comparable performance to the DQN, however, it also creates a schedule where the fleet is utilized slightly less. Lastly, DDPG is best suited in continuous action spaces due to the gradient calculation of the performance function. Instead in this problem, the policy is discretized over a limited set of available actions.

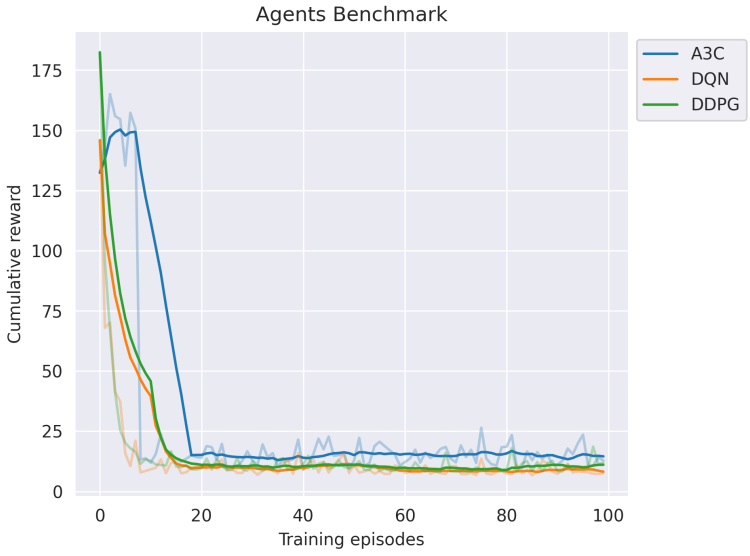


Figure 2.6: RL Agents benchmark

3
Verification & Validation

Verification and validation are necessary procedures in order to ensure soundness of the results, and establish the relevance of the methodology. In this chapter, the most important unit tests are outlined in Section 3.1, then, the system tests are presented in Section 3.2. Lastly, in section 3.3 the reinforcement learning model is validated by comparing the scheduling results of a deterministic scenario with the airline's A-check scheduling tool.

3.1. Unit tests

Unit level test are essential to ensure that the individual simulation blocks are working properly. This verification procedure involves three main assessments. Firstly, the discretization of the scheduling intervals is analyzed. For this test, the DQL agent is configured with different state and action space dimensions to ensure that when a finer mesh is utilized, the rewards are closer to the actual utilization rate. The second test is related to the state vector calculation. The calculation is verified by using the initial states of different scenarios, and the features are compared to the analytical state calculation of the initial condition. Lastly, the number of scheduled checks per aircraft is divided by the simulation horizon in order to ensure that the number of maintenance events is in line with the interval limitations of each policy.

3.2. System tests

The system tests represent the bulk work of the model verification. In this section, the model response is observed and the scheduling policy changes are analyzed.

3.2.1. Utilization alteration

The due date calculation is dependent on the utilization of the aircraft fleet. To ensure that the due dates are calculated accordingly to the flown hours and cycles, the utilization is altered to fixed values. The model grounds more aircraft when the utilization is higher because there are not sufficient slots where to allocate the aircraft. On the other hand, when the utilization is reduced to extremely low values, the DQN does not schedule any aircraft as all the due dates appear to occur after the end of the simulation horizon.

3.2.2. Resources alteration

The maintenance scheduling problem is highly affected by the resources available or the maintenance opportunities. If there are sufficient slots, the optimal solution could be found by applying a greedy policy algorithm that selects the latest option for each aircraft. Therefore, the resource alteration test removes slots in order to observe that aircraft are grounded when the slots are not available to the agent. Similarly, the agent achieves 100% utilization when there are unrealistic amounts of slots available every day.

3.2.3. Greedy algorithm comparison

One of the key reasons to use a reinforcement learning approach is the high adaptability to other environments. The approximate dynamic framework offers a high versatility by exploring the actions available. In order to demonstrate the viability of Deep Q-learning, a reduced scenario with a time horizon of 6 months has been created in which a total of 18 aircraft blocks need to be allocated with a 1500 flight hour interval policy. The case study analyzed does not contain any CBM tasks for this verification test due to computational resources required to train the models, and the fact that CBM adds an additional layer of complexity to the problem. Therefore, it assumed that the differences analyzed in this verification test would be more accen-

tuated in a scenario with CBM tasks. The performance of the algorithm is compared with a myopic policy that chooses the best option for every aircraft without considering other factors such as the future impact on the fleet. This agent is referred to as the *greedy* agent and it does not use any value function approximation since its decisions are merely based on the imminent cost value of the action-space at time t. In Table 3.1, the results of the first verification test have been reported. The first scenario is set-up with two available slots per week. It is observable how the greedy policy works very well in environments with a lot of available slots as every aircraft has an available option close to their respective due date. In fact, the highest interval utilization is reached with the Greedy agent. However, the DQN with 0.5 discount factor finds the best solution because the objective is to schedule all aircraft blocks as close as possible to 95% interval utilization.

Table 3.1: Verification test comparing Greedy vs DQN Agent with 2 slots per week

Parameter	**Greedy Agent**	**DQN Agent**			
Dicsount factor	-	0.1	0.5	0.7	0.95
Obj. Function	5.34	5.37	5.25	5.84	7.57
AOG's	0	0	0	0	0
Utilization [%]	95.29	95.15	94.74	94.33	93.05

Figure 3.1 has been included to visualize the distribution of the aircraft block utilization. The DQN performance with 0.5 discount factor outperforms the others because the spread of the utilization is smaller and a better allocation management is achieved by scheduling all the blocks as close as possible to 95%.

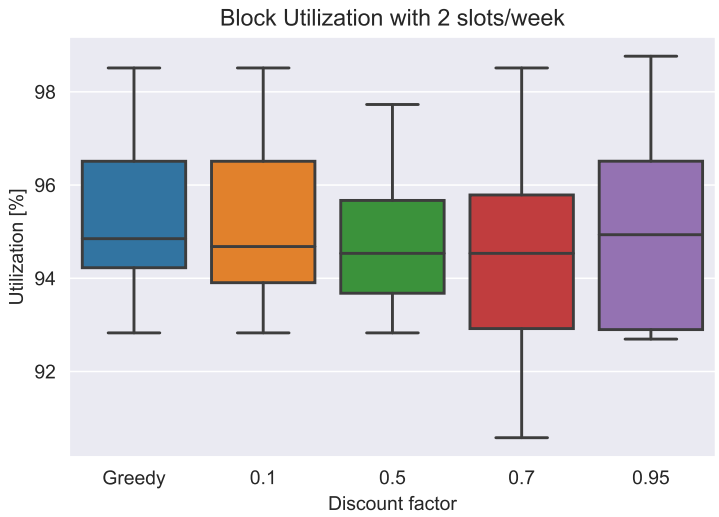


Figure 3.1: Verification Test with 2 slots per week for different discount factors

The second verification test is more constrained in the solution space because only one slot per week is available. The greedy algorithm manages to have a relatively high average block utilization but it comes at the cost of some aircraft in the fleet that are scheduled at 17%. The greedy policy schedules all aircraft as late as possible instead of scheduling them earlier in order to leave better options to rest of the fleet. For this reason the results reported in Table 3.2 demonstrate that a greedy policy does not perform better than the DQN agent. Once again, when the rewards are discounted by 0.5, the DQN manages to schedule aircraft in order to have the lowest interval utilization at 39%. For this reason, the overall cost is the lowest for the DQN with 0.5 discount factor.

Table 3.2: Verification test comparing Greedy vs DQN Agent with 1 slot per week

Parameter	Greedy Agent	DQN Agent			
Dicsount factor	-	0.1	0.5	0.7	0.95
Obj. Function	40.96	39.4	27.8	75.41	51.9
AOG's	0	0	0	0	0
Utilization [%]	86.65	86.81	86.52	76.27	81.31

In Figure 3.2 the action distribution of the second verification test is shown. The agent with 0.5 discount factor performs much better than the other as it manages to have an average utilization of 86.52 % with a relatively contained variation. From this analysis it is deemed that the RL agent is able to understand the state space representation and it does not learn a simple greedy policy. Moreover, it is observable that the greedy agent achieves a performance that is close to optimal in scenarios with a large amount of slots. Nonetheless, the solution quality is not competitive enough in constrained action-spaces. Therefore, it can be concluded that reinforcement learning present a much more robust solution and it can be used as a valid algorithmic strategy. Lastly, when the DQN is discounted with a low value such as 0.1, the performance of the agent is very similar to the greedy agent because it learns to approximate the imminent rewards without considering future states. Instead, if the discount factor rises above 0.5, the future rewards estimation does not provide relevant information to the agent, and it starts behaving sub-optimally. This additional verification analysis demonstrates the usefulness of reinforcement learning in maintenance scheduling, and the importance of discounting rewards in a way that it provides relevant information about the competing aircraft and the available resources.

Figure 3.2: Verification Test with 1 slot per week for different discount factors

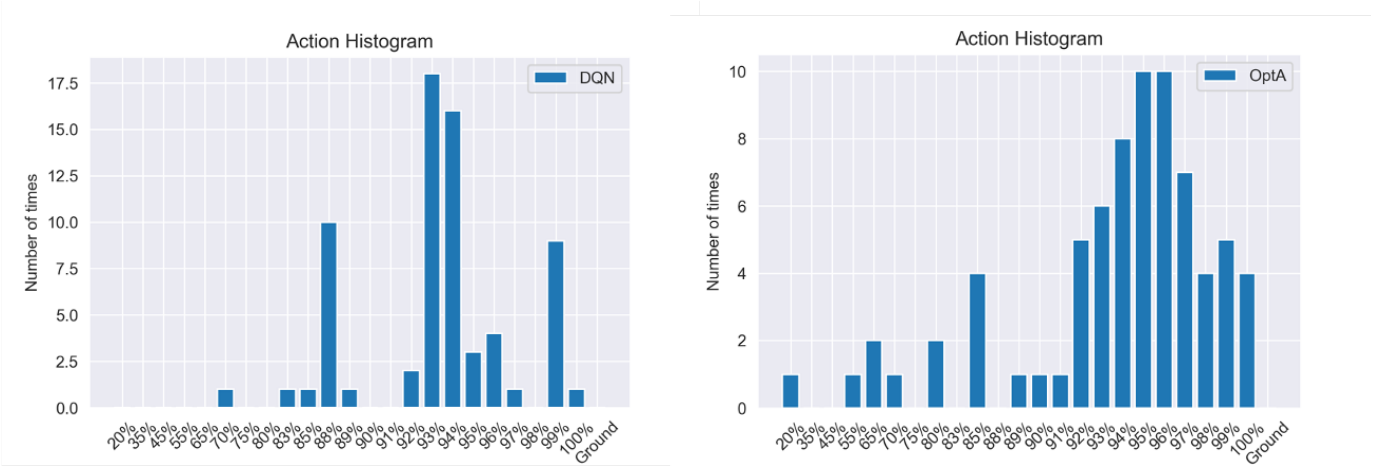
3.3. Validation

The validation strategy of the scheduling tool is a necessary aspect of this research, required to benchmark the solution quality and understand whether the reinforcement learning algorithm is able to produce an near-optimal scheduling policy. Since, the CBM simulation has never been attempted before, the tool can be validated only for a deterministic case. Therefore, the validation of a scenario with routine blocks is deemed sufficient for the implementation of a DQL model in the AMSP. The proposed validation strategy employs the airline tool called *OptA-Check*, that schedules A-checks under the same conditions of the DQL model.

OptA-Check is a mixed-integer linear program (MILP) tool that schedules the A-checks of a specific fleet. The validation excericse is carried out with a maintenance block policy of 1500FH as outlined in the Maintenance Planning Document (MDP). The Boeing 737 fleet, composed by 52 aircraft, is considered in the scope of the problem because it provides a more challenging scenario; mainly because the fleet is larger, and the scenario

considers three types of user constraints. Furthermore, the validation exercise proved that the adaptability of the RL algorithm to user-constraints is well-suited in an industry context. The user-constraints are explained in the following list:

- **Slot reservation**: a specific slot is assigned to a tail number
- **Slot cancellation**: specific slots are canceled due to an event, such as holidays or engineers unavailability
- **Merging with C-checks**: if a C-Check of a specific tail is scheduled during the simulated horizon, it is highly recommended to merge the A-check in order to ground the aircraft the least possible times. To implement this constraint, the action-space of the DQL was increased to 102 neurons, where the additional output corresponds to the merge action. The reward of merging an A-check into a C-Check is costed as half of the best interval utilization action.



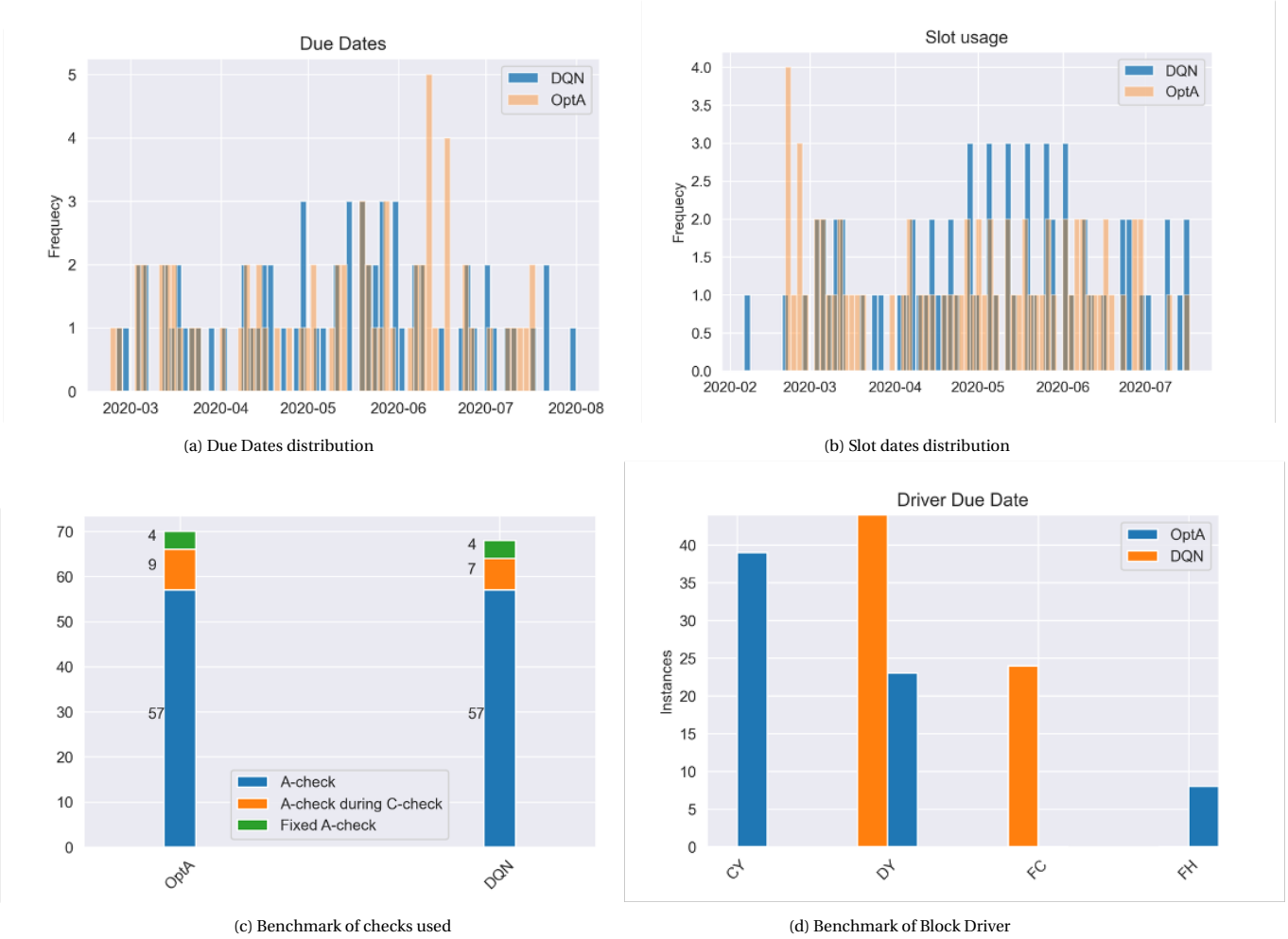
(a) DQN Average Interval Utilization: 93.5%

(b) OptA Average Interval Utilization: 91.5%

Figure 3.3: Action distribution in validation scenario

The action distribution of the two algorithms can be seen in Figure 3.3. The performance of the Deep Q-Network is slightly better than OptA-Check and it is more centered around an interval utilization of 95%. The average interval utilization of the DQN is 93.5%, while OptA-Check has an average performance of 91.5%. Although theoretically in a deterministic scenario it would be impossible to have a better performance than a MILP model, the improvement is attributed to different assumptions and simulation dynamics. For instance, the DQN assumes a constant flight hour distribution between each update of the fleet, and the MILP calculates maintenance days as idle time for each aircraft. In Figure 3.4a and Figure 3.4b, the due dates events and the slots utilized by the two algorithms are superimposed. Two observations are noted from this analysis: (1) the DQN dates are slightly delayed with respect to the OptA-check decisions, and (2) in 72% of the cases the same slots were selected, while the remaining times a slot on a different date was chosen.

Furthermore, it can be observed from Figure 3.4c that the type of slots used by the two algorithms is almost the same. The only difference is that the DQN does not merge two extra A-checks in a C-check due to boundary conditions. In other words, the DQN stops the simulation earlier than OptA check because the maintenance could be scheduled in a slot that is not given in the current horizon. Lastly, in Figure 3.4d the block interval drivers are depicted. Most of the times the block driving factors are determined by the calendar days or year limitation (DY, CY) which can be considered as the same. However, the DQN simulation is also constrained by the flight cycles, while OptA check is affected by the FH interval. This divergence occurs due to slightly different utilization parameters that have been estimated from historical data for the DQN case. Overall, the performance is quite satisfactory, and the simulation parameters such as drivers, due dates and used slots indicate that the DQN policy produces a schedule similar to the airline in-house tool.



(a) Due Dates distribution

(b) Slot dates distribution

(c) Benchmark of checks used

(d) Benchmark of Block Driver

Figure 3.4: Benchmark Validation Scenario

Finally, Table 3.3 contains a summary of the validation results. The block interval utilization is highly comparable for the two models. The 2% difference can be justified because the DQN is more flexible when calculating due dates and makes use of different transition dynamics than *Opt-A* in order to calculate aircraft utilization. The objective function values align with the difference in interval utilization based on the logarithmic shape of the reward function, and the fact that the DQN model schedules less tasks when is close to the final boundary condition.

Table 3.3: Validation of DQN vs Opt-A Check for the A-check plan of B737 fleet

Model	**Utilization[%]**	**Obj. Function**	**Blocks Scheduled**	**A-Check Slots**	**C-Check Slots**	**User Constrained Slots**
DQN	93.5	55	68	57	7	4
Opt-A	91.5	73	70	57	9	4

4
Sensitivity Analysis

It is of paramount importance to investigate the influence of the model parameters and the input variables on the scheduling performance. In this chapter, a detailed analysis is presented to tune the DQN configuration. Firstly, the learning convergence and the final training performance are observed to make an educated choice of neural network hyperparameters. Then, the maintenance scheduling results are analyzed by changing some of the most relevant input variables in the CBM scenarios.

4.1. DQN hyperparameter tuning

4.1.1. Greediness sensitivity

The greediness determines the balance between exploration and exploitation during the training. Initially, the neural network weights are assigned randomly. As a result, the RL model behaves accordingly during the first episodes, which leads to sub-optimal decisions. The initial exploration rate of the environment is set to 100%, such that it is able to search the feasible action-space region. As the learning progresses, the agent understands what is the impact of the actions it takes, based on the state it observes and the reward it receives. At this point, the exploration rate is slowly decayed such that the internal value-function knowledge is exploited with a greedy policy by means of the **argmin** operator. The implementation of this strategy, also known as $\epsilon-$greedy policy, occurs by decreasing the probability that the agent acts randomly over time, and instead a greedy policy is used to decide the next action. The analysis performed in this section keeps the same starting and finishing ϵ to 1 and 0.01, respectively. However, the decay is changed at every episode with a different rate until the minimum exploration value is reached. The resulting sensitivity analysis can be seen in Figure 4.1 where the simple moving average (SMA) over ten samples of the cumulative rewards has been plotted against the training episodes.

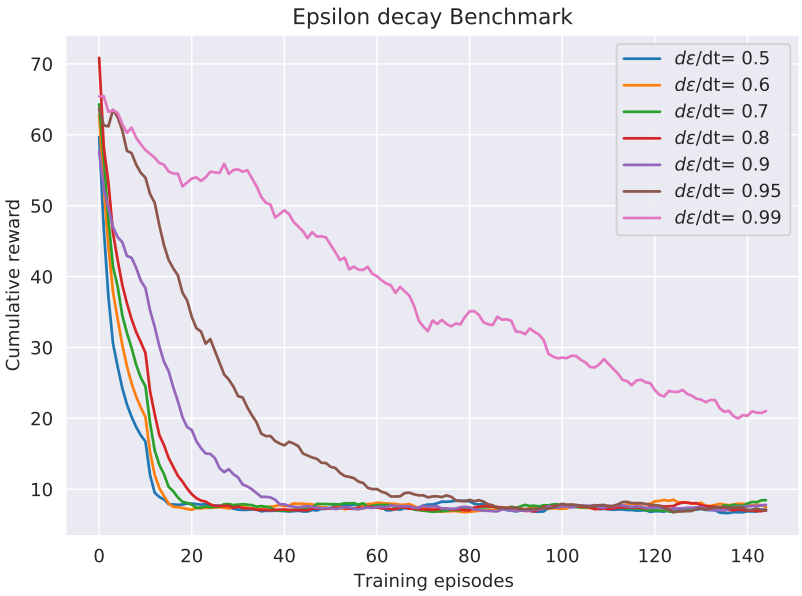


Figure 4.1: Exploration decay sensitivity

In general, the exploration rate is not highly influential on the model performance but mainly on the convergence rate. Thus, in most of the cases, the DQN agent is able to learn very similar policies. The optimal exploration decay was calibrated with a value of 0.9 since it manages to explore a wide portion of the action space, and it achieves the lowest cumulative cost at the end of the training phase.

4.1.2. Learning rate sensitivity

The learning rate indicates the importance of new experiences in the calibration of the model.The neural network is defined by the coefficients of each neuron, also known as weights. Powell (2011) states that the choice of the step-size or learning rate is one of the most important decisions to ensure convergence in the learning process. Therefore, a sensitivity analysis has been performed in order to analyze the influence of the backproagation in the model and determine the most optimal learning rate. Two main conclusions can be drawn from Figure 4.2. Firstly, when the learning rate is too high, as in the case of $lr = 1e-2$, the NN does not manage to learn the optimal policy because every update changes the model too much and the gradient is never calibrated in the optimal direction. On the other hand, when the learning rate is increased above $1e-5$ the convergence process is too slow and the agent learning is not sufficient to produce an optimal schedule. Therefore, the best choice is found for $lr \in [1e-5, 1e-3]$. In fact, the performance is almost equal for these cases. Based on the sensitivity results, the policy observed was slightly better for $lr = 1e-4$ which was deemed as the most optimal choice for the final model.

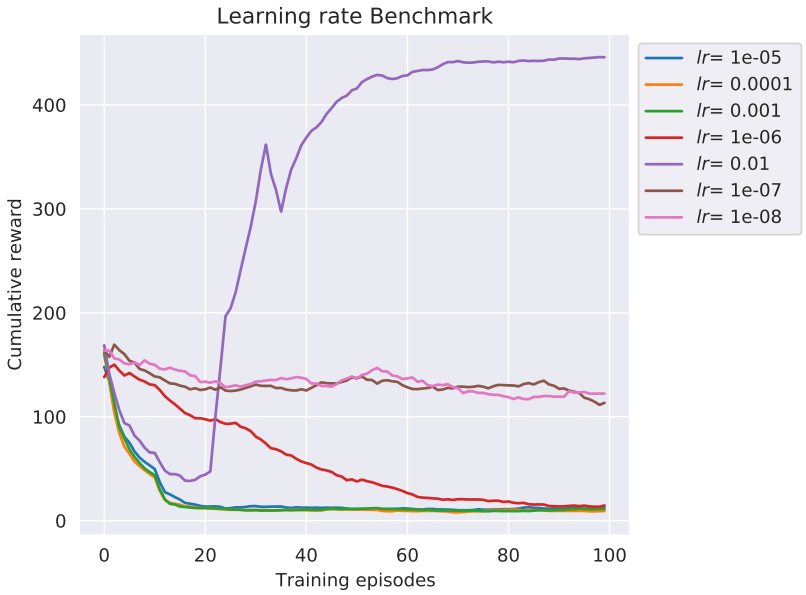


Figure 4.2: Learning rate sensitivity

4.1.3. Activation function sensitivity

The activation function is a crucial component of a neural network. It has a major role in the ability of the model to converge and learn significant (abstract) features from the scheduling environment. The underlying principle is fairly simple: it serves as a transformation function to map the input of the neuron with a non-linear function. It is also possible to use a linear mapping but then a NN would produce a model that is comparable to a parametric linear regression. Neural networks use a technique called backpropagation to train the model, which places an increased computational strain on the activation function, and its derivative. The sensitivity analysis considers four activation functions, that often have been applied in the deep reinforcement learning domain. The mathematical expressions of *Sigmoid, Relu, Leaky Relu, and Swish* are given in the equations below, where x represents the input of the neurons. The first two are widely known in the artificial intelligence community, while the last two have arisen more recently as a modification of the other.

$$Sigmoid = \frac{1}{1+e^{-x}} \tag{4.1}$$

$$Relu = \max(0, x) \tag{4.2}$$

$$LeakyRelu = \max(0.1x, x) \tag{4.3}$$

$$Swish = x \cdot Sigmoid = \frac{x}{1+e^{-x}} \tag{4.4}$$

In Figure 4.3 the activation functions have been applied to the DQN model and their training performance is observed. It is noteworthy how the first two functions do not cope well with the minimization of the cumulative cost. This behavior arises as a consequence of the NN weights calibration that is not able to converge. It is assumed that the minimization of the loss function encounters negative inputs in the neuron values that in general are not properly interpreted with the *Sigmoid and the Relu* activation. The best function is *Swish*, since it achieves convergences earlier than the *LeakyRelu*, and it outputs a lower objective function at the end of the training period.

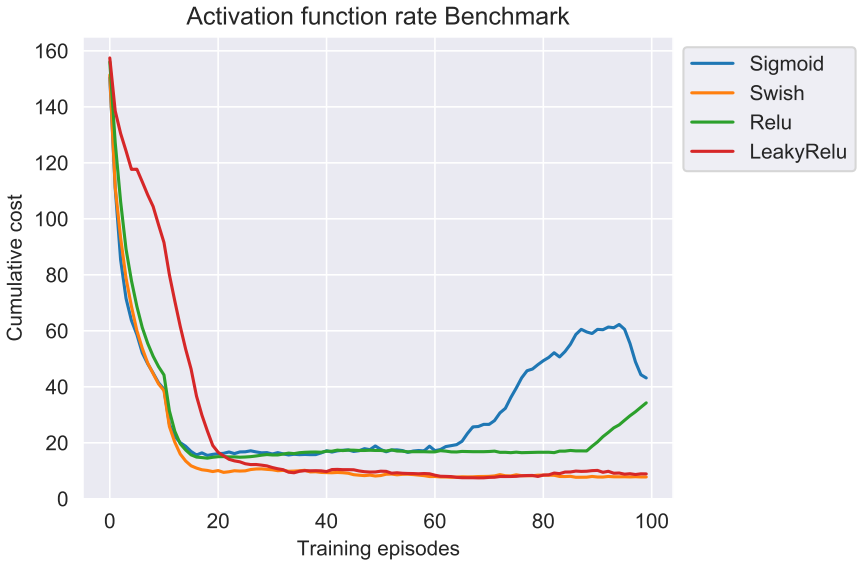


Figure 4.3: Activation function sensitivity

4.1.4. DQN structure benchmark

Several modifications have been explored to adjust the DQN architecture in order to improve training performance. The literature study proposes three other structures: Double DQN, Dueling DQN, and prioritized experience replay (PER). The neural networks modifications have been investigated individually assuming their effect could be linearly superimposed. In Figure 4.4 the four DQN structures have been benchmarked. Firstly, the dueling structure does not achieve any learning. Moreover, the prioritized experience replay (PER) initially converges towards a higher aircraft utilization but the learning is destabilized, causing the model to drift away from the optimal policy. This behavior is attributed to the fact that PER was designed to give priority to those experiences that produced a small deviation in the loss function. However, the agent needs to rely on the other experiences as well to learn the optimal scheduling policy. Lastly, the double DQN does not influence a lot the learning behavior because it is best employed when the Q-values are overestimated. Since the problem studied in this research is not affected by future rewards as much as other RL problems, the DQN predictions are lower. Therefore, the conclusion drawn from this analysis points to the fact that the DQN modifications suggested in literature do not provide an increase in performance for the maintenance scheduling problem.

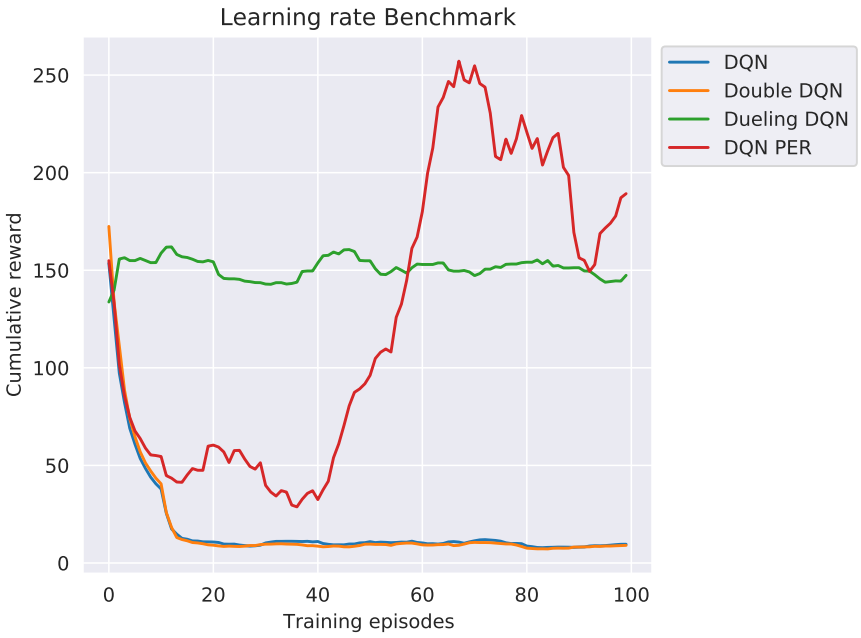


Figure 4.4: DQN structure benchmark

4.1.5. DQN final configuration

The optimal DQN configuration is derived from the sensitivity analysis and the verification tests. In Table 4.1 the complete set of hyperparameters has been reported. The discount factor has been determined from the verification test conducted in chapter 3. Moreover, the learning rate, the decay rate, and the activation function were established during the sensitivity analysis. Lastly, the batch size, buffer memory size, the hidden layers, the target delay, and the dense layer size were chosen on a trial and error basis.

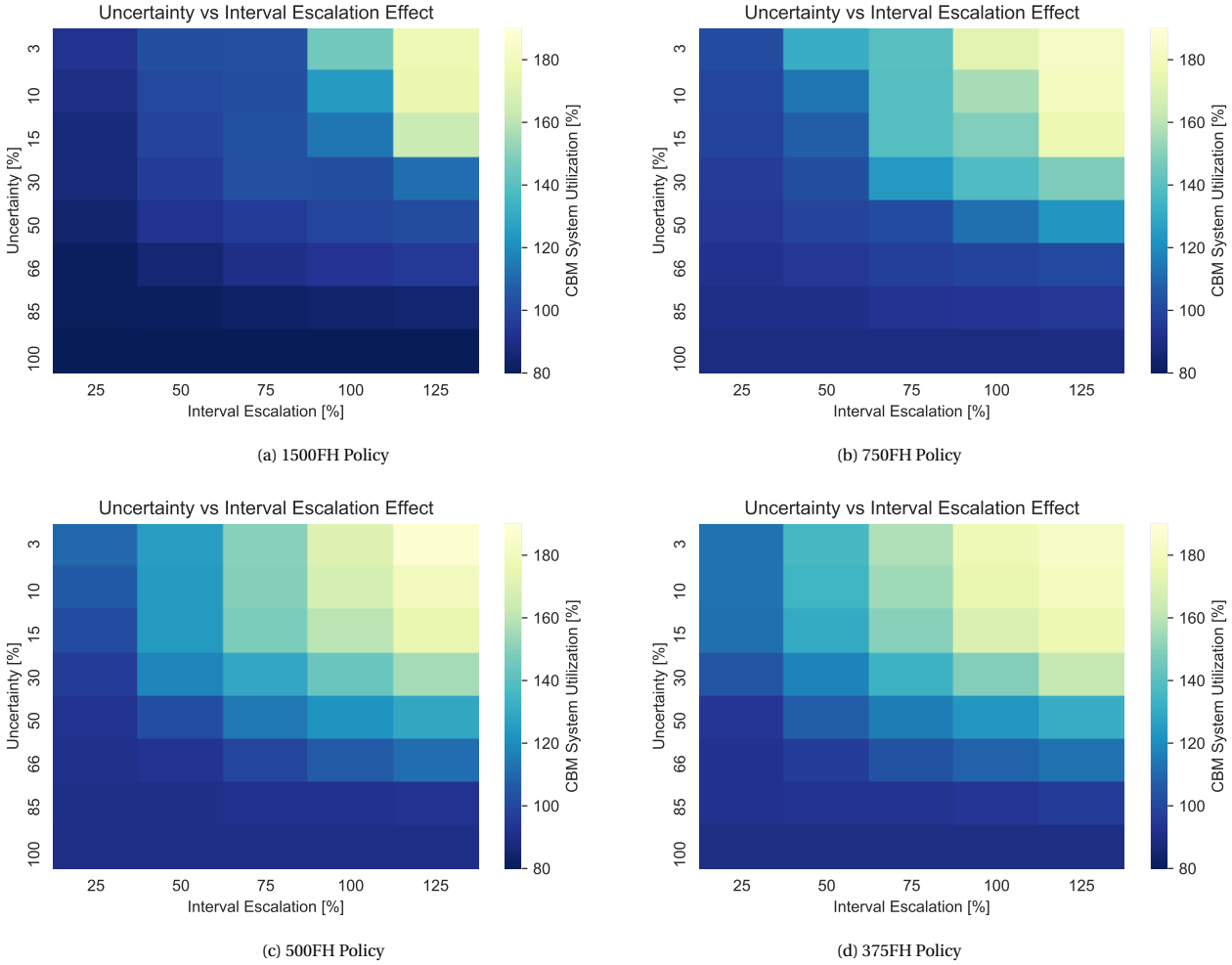
Table 4.1: DQN configuration

Parameter	**Value**
Learning rate (α)	0.0001
Discount factor (γ)	0.5
Initial exploration rate (ϵ_0)	1.0
Final exploration rate (ϵ_T)	0.01
Exploration rate decay ($d\epsilon/dt$)	0.9
Target delay (τ)	10
Batch size	32
Buffer memory size ($\|\mathscr{R}\|$)	64
Hidden layers	3
Dense size (neurons)	100
Output size (neurons)	101
Training episodes (T)	100
Activation function	swish
DQN structure	DQN

4.2. CBM sensitivity

4.2.1. PHM uncertainty and interval escalation

The results presented in Part I of this work focused on a specific case with 100% interval escalation and 15% PHM uncertainty. Though a preliminary sensitivity analysis was already presented in the scientific article, the purpose of this section is to provide additional information on the results of other scenarios with varied CBM parameters. The sensitivity analysis presented in Figure 4.5 shows the combined effect of escalation and uncertainty on the utilization of escalated tasks. The analysis investigates the results of Case 3 (25% CBM) because a larger number of escalated tasks is considered. However, a similar effect could be observed in Case 2 (10% CBM). A clear trend is appreciated from the figures: the utilization of escalated systems increases from the bottom left corner towards the upper right corner. This behavior is expected as lower uncertainties and higher interval escalation should lead to an overall higher interval utilization. It is noteworthy that the 1500FH policy presents lower values than the rest because, in this case, uncertainty affects the scheduling performance parabolically. Furthermore, as the interval escalation increases, the CBM utilization converges more and more for the policies analyzed. Therefore, a highly segmented policy is more useful whenever there is uncertainty involved in the maintenance planning process. However, this effect dissipates as the interval escalation increases. For instance, the top right corner (3% uncertainty and 125% escalation) of the four plots presents values that are much closer than in any other regions of the heatmap. Nonetheless, an uncertainty level of only 3% is too optimistic and not likely to occur. The conclusion drawn from this sensitivity analysis, is that interval segmentation always benefits the utilization of tasks monitored by prognostics.



(a) 1500FH Policy

(b) 750FH Policy

(c) 500FH Policy

(d) 375FH Policy

Figure 4.5: Interval escalation and PHM uncertainty sensitivity analysis (Case 3 - 25% CBM)

In the tables below, four additional cases have been reported where the complete key performance indicators can be analyzed. The scenarios selected include the following combinations of escalation-uncertainty percentages: 100-3, 100-85, 25-15, and 125-15. In the first two cases, shown in Table 4.2 and Table 4.3, the effect of uncertainty is analyzed more accurately, while in Table 4.4 and Table 4.5 the effect of interval escalation is considered. The key performance indicators reported in the tables are the following:

Case	:	Experimental case that determines CBM action rate
Policy	:	Interval policy in flight hours (1500, 750, 500, 375)
Checks	:	One full check corresponds to one A-check for the 1500FH policy, two A-checks for the 750FH policy, and so on
Tasks	:	Total amount of tasks scheduled. This includes routine tasks, substituted tasks, and escalated tasks based on RUL prognostics
Sub.	:	Repetitions of tasks in the scope of substitution (43 tasks in total)
Esc.	:	Repetitions of tasks in the scope of interval escalation (14 tasks in total)
Util.[%]	:	Average interval utilization of escalated tasks monitored with RUL prognostics
Util. (all)[%]	:	Average interval utilization of 14 tasks in the scope of CBM interval escalation and 43 tasks in scope of CBM substitution
Avail.[DY]	:	Average aircraft availability in days
Labor[hrs]	:	Average labor hours per aircraft
Labor[hrs/check]	:	Average labor hours per aircraft check
Comp[min]	:	Computational time of the instance in minutes

An increase in the interval utilization of escalated tasks is visible when decreasing uncertainty and increasing escalation. Nevertheless, the scheduling performance is dictated by other factors such as the aircraft availability, the labor hours, and the total number of scheduled tasks. In the four cases, it is clear that the policies with shorter intervals perform more maintenance checks. However, this is an inevitable consequence of calculating the KPI's over a period of 12 months for each aircraft. It is expected that policies that have a higher maintenance frequency will start already the first part of the next check before the lower frequency policies.

It is noteworthy that the KPI's of Case 1 (0% CBM) are fixed for the all the tables because this sensitivity analysis is only varying CBM parameters. When the uncertainty is increased from Table 4.2 to Table 4.3, the number of tasks repetitions also increases for the escalated systems. This effect is a direct consequence of not being able to postpone the task due dates, as visible from the sixth column of the tables where the mean utilization of escalated systems (Util.[%]) is reported. Moreover, the hour difference of the labor per check KPI is smaller for the high frequency policies, since they can deal with uncertainty more efficiently. The aircraft availability is lightly affected in the sensitivity analysis because it is mainly driven by the reduction of substitution tasks. Although substitution is not affected by escalation nor uncertainty, in future extensions of this work, the consideration of non-routine tasks originating from substituted tasks should influence the overall model performance.

Table 4.2: Scheduling results with 3% Uncertainty and 100% Interval Escalation

Case	Policy	Check	Tasks	Sub.	Esc.	Util.[%]	Util.(all)[%]	Avail.[DY]	Labor [hrs.]	Labor[hrs/check]	Comp.[min]
1	1500	4.00	383.00	150.00	48.00	-	78.92	361.24	302.48	75.62	2.31
1	750	4.00	365.00	139.00	43.00	-	82.05	361.34	289.74	72.43	4.57
1	500	4.19	368.12	141.50	45.69	-	75.69	361.29	296.52	70.81	7.00
1	375	4.25	351.00	140.00	46.00	-	77.47	361.44	286.37	67.38	9.66
2	1500	4.00	301.04	79.00	41.04	144.09	77.40	362.34	216.77	54.19	3.84
2	750	4.00	287.36	71.00	35.36	156.83	82.75	362.44	212.15	53.04	7.17
2	500	4.13	297.69	70.61	37.66	139.71	77.19	362.35	217.68	52.66	10.79
2	375	4.25	285.07	67.00	35.07	155.40	81.33	362.34	219.29	51.60	15.24
3	1500	4.00	201.04	0.00	19.04	149.77	149.77	363.19	154.63	38.66	13.19
3	750	4.00	190.07	0.00	17.07	173.17	173.17	363.28	147.83	36.96	23.10
3	500	4.15	196.84	0.00	16.65	170.45	170.45	363.24	149.83	36.14	34.92
3	375	4.25	185.92	0.00	16.92	175.39	175.39	363.32	143.48	33.76	47.41

Table 4.3: Scheduling results with 85% Uncertainty and 100% Interval Escalation

Case	Policy	Check	Tasks	Sub.	Esc.	Util.[%]	Util.(all)[%]	Avail.[DY]	Labor [hrs.]	Labor[hrs/check]	Comp.[min]
1	1500	4.00	383.00	150.00	48.00	-	78.92	361.24	302.48	75.62	2.31
1	750	4.00	365.00	139.00	43.00	-	82.05	361.34	289.74	72.43	4.57
1	500	4.19	368.12	141.50	45.69	-	75.69	361.29	296.52	70.81	7.00
1	375	4.25	351.00	140.00	46.00	-	77.47	361.44	286.37	67.38	9.66
2	1500	4.00	307.67	79.00	47.67	88.27	74.86	362.25	222.92	55.73	3.71
2	750	4.00	293.04	71.00	41.04	96.46	79.92	362.34	216.42	54.10	8.18
2	500	4.14	303.04	70.66	42.82	92.69	75.21	362.28	221.56	53.54	12.05
2	375	4.25	289.61	67.00	39.61	97.03	78.67	362.25	222.44	52.34	16.16
3	1500	4.00	224.19	0.00	42.19	84.77	84.77	362.75	186.98	46.75	12.60
3	750	4.00	210.12	0.00	37.12	94.22	94.22	362.93	173.31	43.33	29.02
3	500	4.15	216.94	0.00	36.75	92.63	92.63	362.90	175.34	42.29	38.55
3	375	4.25	204.47	0.00	35.47	94.36	94.36	363.01	166.95	39.28	54.61

The interval escalation mainly influences the average utilization of escalated systems. In Table 4.4 the due dates are postponed more often than in Table 4.5. Therefore, a higher interval escalation value, leads to a larger interval utilization and fewer task repetitions. Overall, the influence on the labor hours and the aircraft availability is smaller for an interval escalation change with respect to a change in PHM uncertainty. This effect also occurs because interval escalation affects the model performance linearly, while uncertainty can have a parabolic relation with the interval utilization if the prognostics are not re-evaluated on time.

Table 4.4: Scheduling results with 15% Uncertainty and 125% Interval Escalation

Case	Policy	Check	Tasks	Sub.	Esc.	Util.[%]	Util.(all)[%]	Avail.[DY]	Labor [hrs.]	Labor[hrs/check]	Comp.[min]
1	1500	4.00	383.00	150.00	48.00	-	78.92	361.24	302.48	75.62	2.31
1	750	4.00	365.00	139.00	43.00	-	82.05	361.34	289.74	72.43	4.57
1	500	4.19	368.12	141.50	45.69	-	75.69	361.29	296.52	70.81	7.00
1	375	4.25	351.00	140.00	46.00	-	77.47	361.44	286.37	67.38	9.66
2	1500	4.00	300.19	79.00	40.19	157.16	77.60	362.34	216.06	54.02	3.87
2	750	4.00	287.04	71.00	35.04	149.54	82.03	362.44	211.64	52.91	7.36
2	500	4.13	297.62	70.57	37.73	128.38	76.56	362.36	217.56	52.67	11.12
2	375	4.25	285.02	67.00	35.02	136.60	80.18	362.34	219.07	51.55	15.15
3	1500	4.00	199.41	0.00	17.41	164.29	164.29	363.21	152.62	38.16	15.48
3	750	4.00	189.52	0.00	16.52	176.96	176.96	363.29	147.05	36.76	27.49
3	500	4.14	195.47	0.00	15.38	174.75	174.75	363.26	148.16	35.75	40.77
3	375	4.25	184.37	0.00	15.37	175.86	175.86	363.35	141.51	33.30	53.43

Table 4.5: Scheduling results with 15% Uncertainty and 25% Interval Escalation

Case	Policy	Check	Tasks	Sub.	Esc.	Util.[%]	Util.(all)[%]	Avail.[DY]	Labor [hrs.]	Labor[hrs/check]	Comp.[min]
1	1500	4.00	383.00	150.00	48.00	-	78.92	361.24	302.48	75.62	2.31
1	750	4.00	365.00	139.00	43.00	-	82.05	361.34	289.74	72.43	4.57
1	500	4.19	368.12	141.50	45.69	-	75.69	361.29	296.52	70.81	7.00
1	375	4.25	351.00	140.00	46.00	-	77.47	361.44	286.37	67.38	9.66
2	1500	4.00	307.49	79.00	47.49	88.27	74.84	362.26	222.65	55.66	3.75
2	750	4.00	292.12	71.00	40.12	98.40	79.91	362.34	215.05	53.76	7.75
2	500	4.15	301.67	70.75	41.11	102.12	75.76	362.30	220.36	53.15	11.83
2	375	4.25	288.00	67.00	38.00	110.76	79.57	362.29	221.52	52.12	16.25
3	1500	4.00	222.62	0.00	40.62	87.75	87.75	362.79	183.97	45.99	12.22
3	750	4.00	209.09	0.00	36.09	98.88	98.88	362.96	171.80	42.95	28.79
3	500	4.14	212.66	0.00	33.03	101.62	101.62	362.97	170.41	41.19	42.16
3	375	4.25	199.02	0.00	30.02	112.28	112.28	363.09	160.57	37.78	62.58

4.2.2. Fleet size sensitivity

One of the main issues remarked during the cost-benefit analysis (CBA) was the importance of considering the fleet size in the scope of the CBM implementation. By increasing the number of aircraft, the investment cost increases as there are more sensors that need to be installed. Nevertheless, the profit value due to extra fleet availability and the labor cost reduction lead to larger savings. Lastly, the sensor's certification costs can be absorbed more easily due to economies of scale. The CBA sensitivity is performed based on a linear extrapolation of the investment cost, the labor savings, and the profit due to extended availability of one aircraft. The absorption of the investment cost is calculated by means of Equation 4.5, where AC indicates the fleet size.

$$\text{Absorption}[\%] = \frac{\text{New Income}}{\text{Investment}} = \frac{(\text{Labor Savings} + \text{Profit}) \cdot AC}{\text{Installation} \cdot AC + \text{Certification}} \tag{4.5}$$

Figure 4.6 demonstrates that the cost absorption improves with a logarithmic trend. Based on Equation 4.5 an horizontal asymptote is expected as the fleet size increases. The original fleet size of 16 aircraft corresponds to the first point of the graph. Therefore, the current fleet size poses an additional barrier for the profitability of the CBM strategy. Moreover, the investment cost required to implement CBM in 25% of the AMP is not compensated with a larger fleet. Since the tasks included in the 25% CBM case have larger intervals, only a few executions of those tasks are realized in a time horizon of one year. To have a profitable maintenance strategy, CBM tasks need to be selected based on their impact on the investigated time horizon, as well as their influence on the non-routine maintenance program.

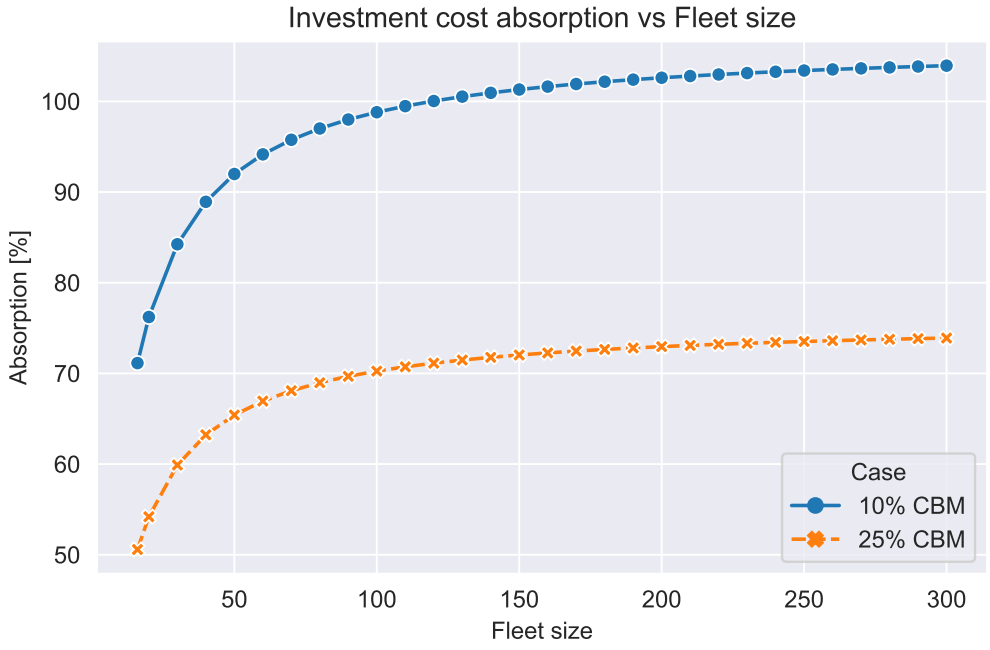


Figure 4.6: Investment cost absorption vs Fleet size

Bibliography

Afsar, H. M., Espinouse, M., and Penz, B. (2006). A Two-step Heuristic to Build Flight and Maintenance Planning in a Rolling-horizon. In *2006 International Conference on Service Systems and Service Management*, volume 2, pages 1251–1256.

Afsar, M. H., Marie-Laure, E., and Bernard, P. (2009). Building flight planning for an airline company under maintenance constraints. *Journal of Quality in Maintenance Engineering*, 15(4):430–443. URL.

Alagoz, O., Hsu, H., Schaefer, A. J., and Roberts, M. S. (2010). Markov decision processes: A tool for sequential decision making under uncertainty. *Medical Decision Making*, 30(4):474–483.

Baradaran, V., Shafaei, A., and Hosseinian, A. H. (2019). Stochastic vehicle routing problem with heterogeneous vehicles and multiple prioritized time windows: Mathematical modeling and solution approach. *Computers & Industrial Engineering*, 131:187–199. URL.

Barnhart, C., Belobaba, P., and Odoni, A. R. (2003). Applications of operations research in the air transport industry. *Transportation Science*, 37(4):368–391.

Barnhart, C. and Smith, B. (2012). *Quantitative Problem Solving Methods in the Airline Industry*, volume 169 of *International Series in Operations Research & Management Science*. Springer US, Boston, MA. URL.

Bellman, R. E. (1972). *Dynamic Programming*. Academic Press, Inc., New York, USA, first edition.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. (2019). Neural combinatorial optimization with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings*. URL.

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

Boere, N. J. (1977). Air Canada Saves with Aircraft Maintenance Scheduling. *Interfaces*, 7(3):1–13.

Busoniu, L., De Schutter, B., and Babuska, R. (2010). Approximate Dynamic Programming and Reinforcement Learning. In *Studies in Computational Intelligence*, volume 281, pages 3–44.

Chiraphadhanakul, V. and Barnhart, C. (2013). Robust flight schedules through slack re-allocation. *EURO Journal on Transportation and Logistics*, 2(4):277–306. URL.

Clarke, L., Johnson, E., Nemhauser, G., and Zhu, Z. (1997). The aircraft rotation problem. *Annals of Operations Research*, 69(0):33–46. URL.

Clarke, L. W., Hane, C. A., Johnson, E. L., and Nemhauser, G. L. (1996). Maintenance and crew considerations in fleet assignment. *Transportation Science*, 30(3):249–260.

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

Cook, A., Tanner, G., and Anderson, S. (2004). Evaluating the true cost to airlines of one minute of airborne or ground delay: final report. Technical report, EUROCONTROL, Brussels, Belgium. URL.

Cordeau, J.-F., Stojkovic, G., Soumis, F., and Desrosiers, J. (2001). Benders Decomposition for Simultaneous Aircraft Routing and Crew Scheduling. *Transportation Science*, 35:375–388.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., and Song, L. (2017). Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, volume 2017-December, pages 6349–6359. URL.

Daily, J. and Peterson, J. (2017). Predictive Maintenance: How Big Data Analysis Can Improve Maintenance. In Richter, K. and Walther, J., editors, *Supply Chain Integration Challenges in Commercial Aerospace: A Comprehensive Perspective on the Aviation Value Chain*, pages 267–278. Springer International Publishing, Cham. URL.

Deng, Q., Santos, B. F., and Curran, R. (2020). A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization. *European Journal of Operational Research*, 281(2):256–273. URL.

Dong, T., Haftka, R., and Kim, N. (2019). Advantages of Condition-Based Maintenance over Scheduled Maintenance using Structural Health Monitoring System.

Dreyfus, S. (2002). Richard Bellman on the birth of dynamic programming. *Operations Research*, 50(1):48–51.

Feo, T. A. and Bard, J. F. (1989). Flight Scheduling and Maintenance Base Planning. *Management Science*, 35(12):1415–1432.

Fossier, S. and Robic, P. (2017). Maintenance of complex systems — From preventive to predictive. In *2017 12th International Conference on Live Maintenance (ICOLIM)*, pages 1–6.

François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., and Pineau, J. (2018). *An Introduction to Deep Reinforcement Learning*. now. URL.

George, A., Powell, W. B., and Kulkarni, S. R. (2008). Value Function Approximation using Multiple Aggregation for Multiattribute Resource Management. *Journal of Machine Learning Research*, 9(68):2079–2111. URL.

Goel, A. and Gruhn, V. (2008). A General Vehicle Routing Problem. *European Journal of Operational Research*, 191(3):650–660.

Gopalan, R. and Talluri, K. T. (1998). The Aircraft Maintenance Routing Problem. *Operations Research*, 46(2):260–271. URL.

Hane, C. A., Barnhart, C., Johnson, E. L., Marsten, R. E., Nemhauser, G. L., and Sigismondi, G. (1995). The fleet assignment problem: Solving a large-scale integer program. *Mathematical Programming*, 70(1):211–232. URL.

Hasselt, H. V., Guez, A., and Silver, D. (2016). Deep Reinforcement Learning with Double Q-Learning. In *AAAI Conference on Artificial Intelligence,*.

Hölzel, N., Schilling, T., and Gollnick, V. (2014). An Aircraft Lifecycle Approach for the Cost-Benefit Analysis of Prognostics and Condition-based Maintenance based on Discrete Event Simulation. In *PHM 2014 - Proceedings of the Annual Conference of the Prognostics and Health Management Society 2014*.

Hölzel, N., Schilling, T., Neuheuser, T., and Gollnick, V. (2012a). System Analysis of Prognostics and Health Management Systems for Future Transport Aircraft. In *28th Congress of the International Council of the Aeronautical Sciences 2012, ICAS 2012*, volume 6.

Hölzel, N., Schröder, C., Schilling, T., and Gollnick, V. (2012b). A Maintenance Packaging and Scheduling Optimization Method for Future Aircraft. In *6th International Meeting for Aviation Product Support Processes (IMAPP)*. URL.

IATA's Maintenance Cost Tasks Force (2018). Airline maintenance executive cost commentary. Accessed on 12/11/2020. URL.

Iram, P. (2020). Algorithms of approximate dynamic programming for hydro scheduling. *E3S Web of Conferences*, 144.

Jayaraj, A., Sridharan, R., and Panicker, V. V. (2020). Dynamic tail re-assignment model for optimal line-of-flight breakages. *Sādhanā*, 45(1):16. URL.

Jin, C., Allen-Zhu, Z., Bubeck, S., and Jordan, M. I. (2018). Is Q-learning Provably Efficient? *32nd Conference on Neural Information Processing Systems*.

Kabbani, N. M. and Patty, B. W. (1992). Aircraft routing at American Airlines. *Proceedings of the AGIFORS symposium*.

Kinnison, H. (2004). *Aviation Maintenance Management*. McGraw-Hill Education. URL.

Knowles, M., Baglee, D., and Wermter, S. (2011). Reinforcement Learning for Scheduling of Maintenance. In Bramer, M., Petridis, M., and Hopgood, A., editors, *Research and Development in Intelligent Systems XXVII*, pages 409–422, London. Springer London.

Konda, V. R. and Tsitsiklis, J. N. (2003). On actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166.

Kulkarni, A., Yadav, D., and nikraz, h. (2017). Aircraft maintenance checks using critical chain project path. *Aircraft Engineering and Aerospace Technology*, 89:0.

Lagos, C., Delgado, F., and Klapp, M. A. (2020). Dynamic Optimization for Airline Maintenance Operations. *Transportation Science*, 54:998–1015.

Lan, S., Clarke, J. P., and Barnhart, C. (2006). Planning for robust airline operations: Optimizing aircraft routings and flight departure times to minimize passenger disruptions. *Transportation Science*, 40(1):15–28.

Liang, Z., Feng, Y., Zhang, X., Wu, T., and Chaovalitwongse, W. A. (2015). Robust weekly aircraft maintenance routing problem and the extension to the tail assignment problem. *Transportation Research Part B: Methodological*, 78:238–259. URL.

Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971.

Littman, M., Dean, T., and Kaelbling, L. (2013). On the Complexity of Solving Markov Decision Problems.

Liu, C.-L., Chang, C.-C., and Tseng, C.-J. (2020). Actor-Critic Deep Reinforcement Learning for Solving Job Shop Scheduling Problems. *IEEE Access*, PP:1.

Looker, J. R., Mak-Hau, V., and Marlow, D. O. (2017). Optimal policies for aircraft fleet management in the presence of unscheduled maintenance. In *22nd International Congress on Modelling and Simulation*, Hobart, Tasmania, Australia. URL.

Lotten, D. (2018). Scheduling Planned Hangar Maintenance. Technical report, Amsterdam Vrije Universitetit, Amsterdam. URL.

Luo, S. (2020). Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Applied Soft Computing*, 91:106208. URL.

Maher, S. J. (2016). Solving the integrated airline recovery problem using column-and-row generation. *Transportation Science*, 50(1):216–239.

Martin, R. K. (1999). *Large Scale Linear and Integer Optimization: A Unified Approach*. Springer US.

McFadden, M. and Worrells, D. (2012). Global Outsourcing of Aircraft Maintenance. *Journal of Aviation Technology and Engineering*, 1.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. In *33rd International Conference on Machine Learning, ICML*, pages 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. (2013). Playing Atari with Deep Reinforcement Learning. *ArXiv*, abs/1312.5602. URL.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Moudani, W. E. and Mora-Camino, F. (2000). A dynamic approach for aircraft assignment and maintenance scheduling by airlines. *Journal of Air Transport Management*, 6(4):233–237. URL.

Muchiri, A. and Smit, K. G. (2009). Application of Maintenance Interval De-Escalation in Base Maintenance Planning Optimization. *Enterprise Risk Management*, 1.

Nazari, M., Oroojlooy, A., Snyder, L., and Takac, M. (2018). Reinforcement Learning for Solving the Vehicle Routing Problem. *Advances in Neural Information Processing Systems 31*, pages 9839–9849.

Obadimu, S. O., Karanikas, N., and Kourousis, K. I. (2020). Development of the minimum equipment list: Current practice and the need for standardisation. *MDPI Aerospace*, 7(1).

Odonkor, P. and Lewis, K. (2018). *Control of Shared Energy Storage Assets Within Building Clusters Using Reinforcement Learning*.

Ozkol, I. and Senturk, C. (2017). The effects of the use of single task-oriented maintenance concept and more accurate letter check alternatives on the reduction of scheduled maintenance downtime of aircraft. In *2017 8th International Conference on Mechanical and Aerospace Engineering (ICMAE)*, pages 67–74.

Papakostas, N., Papachatzakis, P., Xanthakis, E., Mourtzis, D., and Chryssolouris, G. (2010). An approach to operational aircraft maintenance planning. *Decision Support Systems*, pages 604–612.

Pereira, M. A. and Ashok Babu, J. (2016). Information Support Tool for Aircraft Maintenance Task Planning. *International Advanced Research Journal in Science, Engineering and Technology*, 3(2).

Polydoros, A. and Nalpantidis, L. (2017). Survey of Model-Based Reinforcement Learning: Applications on Robotics. *Journal of Intelligent & Robotic Systems*, 86:153.

Pontecorvo, J. A. (1984). MSG-3–A Method For Maintenance Program Planning. In *Aerospace Congress and Exposition*. SAE International. URL.

Powell, W. B. (2011). *Approximate dynamic programming : solving the curses of dimensionality*. Wiley.

Powell, W. B. (2014). Clearing the Jungle of Stochastic Optimization. In *Bridging Data and Decisions*, pages 109–137. INFORMS.

Powell, W. B. (2020). *Reinforcement Learning and Stochastic Optimization: A unified framework*. John Wiley & Sons. URL.

Powell, W. B., Simao, H. P., and Bouzaiene-Ayari, B. (2012). Approximate dynamic programming in transportation and logistics: a unified framework. *EURO Journal on Transportation and Logistics*, 1(3):237–284. URL.

Powell, W. B. and Topaloglu, H. (2006). Approximate Dynamic Programming for Large-Scale Resource Allocation Problems. In *Models, Methods, and Applications for Innovative Decision Making*, pages 123–147. INFORMS.

Qantas (2016). The A, C and D of aircraft maintenance. Website. Accessed on 10/05/2020. URL.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for Activation Functions. URL.

ReMAP H2020. Website. Accessed on 23/10/2020. URL.

Ruther, S., Boland, N., Engineer, F., and Evans, I. (2016). Integrated Aircraft Routing, Crew Pairing, and Tail Assignment: Branch-and-Price with Many Pricing Problems. *Transportation Science*, 51.

Safaei, N. and Jardine, A. K. S. (2018). Aircraft routing with generalized maintenance constraints. *Omega*, 80:111–122. URL.

Samaranayake, P. and Kiridena, S. (2012). Aircraft maintenance planning and scheduling: An integrated framework. *Journal of Quality in Maintenance Engineering*, 18(4):432–453.

Sanchez, D. T., Boyacı, B., and Zografos, K. G. (2020). An optimisation framework for airline fleet maintenance scheduling with tail assignment considerations. *Transportation Research Part B: Methodological*, 133:142–164. URL.

Sarac, A., Batta, R., and Rump, C. M. (2006). A branch-and-price approach for operational aircraft maintenance routing. *European Journal of Operational Research*, 175(3):1850–1869. URL.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). Prioritized experience replay. In *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*. URL.

Shannon, M. and Ackert, P. (2010). Basics of Aircraft Maintenance Programs for Financiers. Evaluation & Insights of Commercial Aircraft Maintenance Programs. Technical report.

Shen, X.-N. and Yao, X. (2015). Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. *Information Sciences*, 298:198–224. URL.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359. URL.

Simão, H. and Powell, W. (2009). Approximate dynamic programming for management of high-value spare parts. *Journal of Manufacturing Technology Management*, 20:147–160.

Simão, H. P., Day, J., George, A. P., Gifford, T., Nienow, J., and Powell, W. B. (2008). An Approximate Dynamic Programming Algorithm for Large-Scale Fleet Management: A Case Application. *Transportation Science*, 43(2):178–197. URL.

Singh, S. P., Jaakkola, T., and Jordan, M. I. (1994). Learning Without State-Estimation in Partially Observable Markovian Decision Processes. In Cohen, W. W. and Hirsh, H., editors, *Machine Learning Proceedings 1994*, pages 284–292. Morgan Kaufmann, San Francisco (CA). URL.

Solozabal, R., Ceberio, J., Sanchoyerto, A., Zabala, L., Blanco, B., and Liberal, F. (2020). Virtual Network Function Placement Optimization With Deep Reinforcement Learning. *IEEE Journal on Selected Areas in Communications*, 38(2):292–303.

Sriram, C. and Haghani, A. (2003). An optimization model for aircraft maintenance scheduling and reassignment. *Transportation Research Part A: Policy and Practice*, 37(1):29–48. URL.

Steiner, A. (2006). A Heuristic Method for Aircraft Maintenance Scheduling under Various Constraints. *6 th Swiss Transport Research Conference*. URL.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, first edition.

Teal, C. and Sorensen, D. (2001). Condition based maintenance [aircraft wiring]. In *20th DASC. 20th Digital Avionics Systems Conference (Cat. No.01CH37219)*, volume 1, pages 1–3.

Tinga, T. (2013). *Principles of loads and failure mechanisms. Applications in maintenance, reliability and design*. Springer Series in Reliability Engineering. Springer.

Ulmer, M. W. (2020). Horizontal combinations of online and offline approximate dynamic programming for stochastic dynamic vehicle routing. *Central European Journal of Operations Research*, 28(1):279–308.

Van Den Bergh, J., De Bruecker, P., Beliën, J., and Peeters, J. (2013). Aircraft maintenance operations: state of the art. Technical report, KU Leuven, Leuven.

Vianna, W. O. L. and Yoneyama, T. (2018). Predictive Maintenance Optimization for Aircraft Redundant Systems Subjected to Multiple Wear Profiles. *IEEE Systems Journal*, 12(2):1170–1181.

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Frcitas, N. (2016). Dueling Network Architectures for Deep Reinforcement Learning. In *33rd International Conference on Machine Learning, ICML 2016*, volume 4, pages 2939–2947. URL.

Waschneck, B., Reichstaller, A., Belzner, L., Altenmüller, T., Bauernhansl, T., Knapp, A., and Kyek, A. (2018). Optimization of global production scheduling with deep reinforcement learning. *Procedia CIRP*, 72:1264–1269. URL.

Watkins, C. (1989). *Learning From Delayed Rewards*. PhD thesis, University of Cambridge, England.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292. URL.

Witteman, M., Deng, Q., and Santos, B. F. (2021). A bin packing approach to solve the aircraft maintenance task allocation problem. *European Journal of Operational Research*.

Witteman, M. M. D. (2019). A practical maintenance task packaging model applicable to aircraft maintenance. Technical report, Delft University of Technology. URL.

Yan, S., Hsiao, F. Y., Guo, J., and Chen, Y. C. (2011). Effective aircraft maintenance schedule adjustment following incidents. *Transportation Planning and Technology*, 34(8):727–745.

Yang, M. (2007). *Using advanced tabu search techniques to solve airline disruption management problems*. PhD thesis, University of Texas. URL.

A

Aircraft Maintenance Program Tasks

Table A.1: Routine tasks of A-Check AMP classified per task group and interval duration

Task Group	FH	FC	DY	Nr. Tasks	Labor [hrs]
Detailed Inspection	1500	-	-	7	5.35
Detailed Inspection	-	-	120	1	1
Detailed Inspection	2000	-	360	1	0.5
Detailed Inspection	3000	-	-	1	0.33
Detailed Inspection	-	-	365	1	1.4
Detailed Inspection	7500	-	-	1	0.13
Detailed Inspection	-	1000	-	2	4.46
Detailed Inspection	-	4000	730	1	0.27
Detailed Inspection	-	-	730	1	0.5
Detailed Inspection	12000	-	-	5	1.81
Discard	1500	-	-	1	0.35
Discard	3000	-	-	1	1.5
Discard	4000	-	-	8	5.11
Discard	6000	-	-	4	3.57
Discard	8000	-	-	3	2.1
Discard	8000	3650	-	1	1.4
Discard	-	-	720	1	0.35
Discard	-	-	730	1	0.13
Discard	12000	-	-	1	0.7
Discard	12000	3650	-	2	2.8
Functional Check	1500	-	-	1	0.35
Functional Check	2000	-	-	4	2.32
Functional Check	-	2000	360	3	1.7
Functional Check	6000	-	-	4	2.4
Functional Check	8000	-	-	3	2.19
Functional Check	12000	-	-	3	3.7
General Visual Inspection	1500	-	-	4	4.53
General Visual Inspection	-	600	120	2	1.38
General Visual Inspection	-	-	180	6	2.44
General Visual Inspection	3000	-	-	1	0.27
General Visual Inspection	4000	-	-	4	2.24
General Visual Inspection	6000	-	-	12	10.7
General Visual Inspection	-	-	730	1	0.35
General Visual Inspection	12000	-	-	5	1.15
General Visual Inspection	12000	6000	1095	6	1.07
Lubrication	2000	-	-	1	1.4
Lubrication	-	800	150	8	6.48
Lubrication	-	1000	180	1	0.27
Lubrication	-	2000	360	1	0.48
Lubrication	6000	-	540	4	3.8

Lubrication	-	2000	540	2	2.4
Lubrication	-	-	730	1	0.35
Operational Check	1500	-	-	9	11.53
Operational Check	-	-	180	3	1.75
Operational Check	3000	-	-	6	2.01
Operational Check	3500	-	-	1	0.33
Operational Check	-	-	365	1	1.4
Operational Check	6000	-	-	2	1.74
Operational Check	8000	-	-	4	1.57
Operational Check	-	-	730	1	0.27
Operational Check	12000	6000	1095	2	2.8
Operational Check	12000	-	-	5	3.97
Restoration	1500	-	-	5	8.65
Restoration	3000	-	-	3	2.02
Restoration	6000	-	-	3	2.06
Restoration	-	1000	-	1	0.27
Restoration	9000	-	-	1	0.08
Restoration	12000	-	-	1	1.4
Servicing	1500	-	-	3	2.98
Servicing	3000	-	-	1	0.33
Servicing	6000	-	540	2	1.65
Servicing	6000	-	-	1	0.7
Servicing	-	-	730	1	0.7
Visual Check	1500	-	-	3	1.21
Visual Check	2000	-	-	2	0.26
Visual Check	-	600	-	2	0.54
Visual Check	-	-	360	1	0.42
Visual Check	-	-	730	1	0.13

B
Extended Block Clustering Model

An extension of the block clustering models has been included in this appendix. The new objective function (Equation B.1) minimizes the task repetitions by giving priority to the tasks that occur more frequently, so those that have a shorter interval limitation. Furthermore, it considers capacity constraints per skill (Equation B.4), instead of having a general capacity limitation per block. However, this model was not employed for two main reasons. Firstly, the reinforcement learning algorithm was designed without considering labor hours, such that the required skills would always be available on the maintenance date. Secondly, this extended model did not manage to find a feasible solution in a reasonable time. Lastly, the clustering model could be further extended by implementing zonal and task inter-dependencies requirements.

Sets

J : The set of routine tasks for an aircraft
B : The set of blocks based on the maintenance policy configuration
K : The set of labor skills
P_{ij} : The set of blocks that have a distance from block i that is less or equal than I_j

Parameters

I_j : The interval of task j
D_j^k : The labor hours requirement of skill k for task j
LH^k : The capacity of labor hours in each block for skill k

Decision Variable

x_{ij} : Task j is assigned to block i

$$\textbf{minimize} \sum_{j \in J} \sum_{i \in B} x_{ij} / I_j \tag{B.1}$$

$$\sum_{i \in B} x_{ij} \geq 1 \qquad \forall j \in J \tag{B.2}$$

$$x_{ij} - \sum_{p \in P_{ij}} x_{pj} \leq 0 \qquad \forall i \in B, \quad \forall j \in J \tag{B.3}$$

$$\sum_{j \in J} x_{ij} \cdot D_j^k \leq LH^k \qquad \forall i \in B, \quad \forall k \in K \tag{B.4}$$

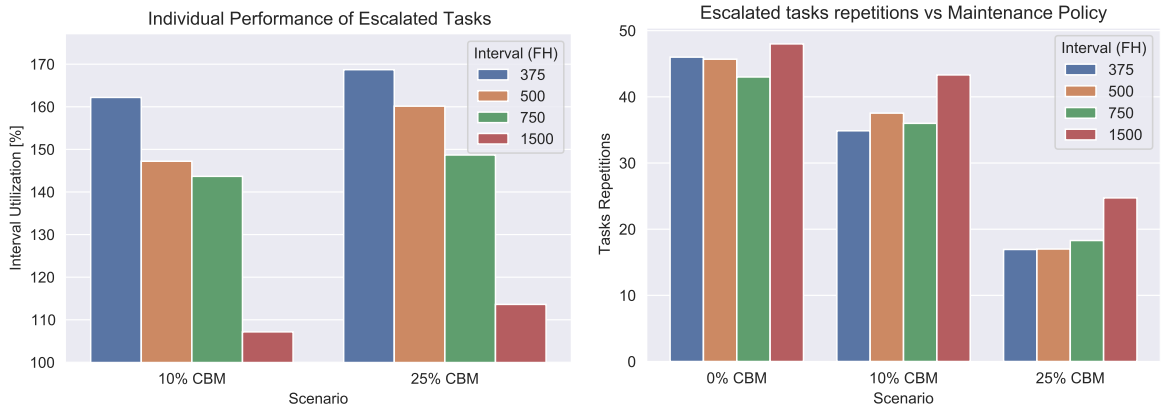
$$x_{ij} \in \{0,1\} \qquad \forall i \in B, \quad \forall j \in J \tag{B.5}$$

C
Additional Scheduling Data

The results presented in the scientific article can be visualized by means of plots for the individual KPI's rather than in a table format. In this appendix the final results for the 15% uncertainty and 100% interval escalation case are presented by means of several bar charts in order to visualize the relative performance between policies and CBM cases. Each plot contains the average results per aircraft based on a 12 months horizon and 100 Monte Carlo simulations.

Table C.1: Scheduling results with 100% interval escalation and 15% uncertainty (365 days horizon)

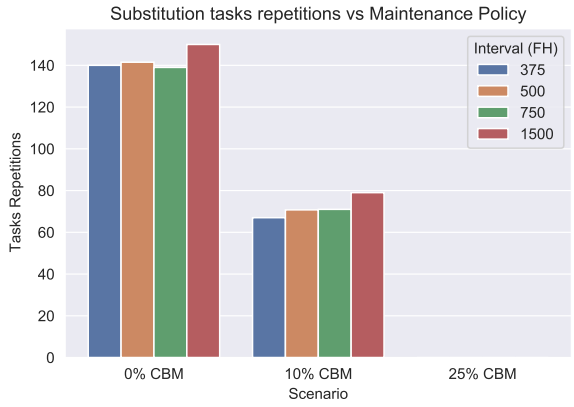
Case	Policy	Check	Tasks	Sub.	Esc.	Util.[%]	Util.(all)[%]	Avail.[DY]	Labor [hrs.]	Labor[hrs/check]	Comp.[min]
1	1500	4.00	383.00	150.00	48.00	-	78.92	361.24	302.48	75.62	2.31
1	750	4.00	365.00	139.00	43.00	-	82.05	361.34	289.74	72.43	4.57
1	500	4.19	368.12	141.50	45.69	-	75.69	361.29	296.52	70.81	7.00
1	375	4.25	351.00	140.00	46.00	-	77.47	361.44	286.37	67.38	9.66
2	1500	4.00	303.31	79.00	43.31	107.16	75.82	362.34	217.93	54.48	3.65
2	750	4.00	288.00	71.00	36.00	143.67	82.27	362.43	212.43	53.11	7.81
2	500	4.15	298.10	70.75	37.54	147.22	77.49	362.35	218.00	52.58	12.16
2	375	4.25	284.87	67.00	34.87	162.20	81.58	362.34	219.05	51.54	17.64
3	1500	4.00	206.73	0.00	24.73	113.61	113.61	363.09	161.36	40.34	11.49
3	750	4.00	191.29	0.00	18.29	148.68	148.68	363.26	149.17	37.29	24.15
3	500	4.15	197.19	0.00	17.00	160.15	160.15	363.24	150.34	36.26	40.27
3	375	4.25	185.94	0.00	16.94	168.71	168.71	363.32	143.50	33.77	54.78



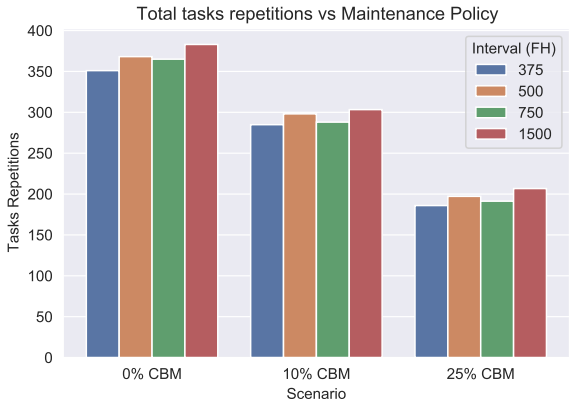
(a) Utilization of tasks monitored by RUL prognostics

(b) Escalated tasks repetitions

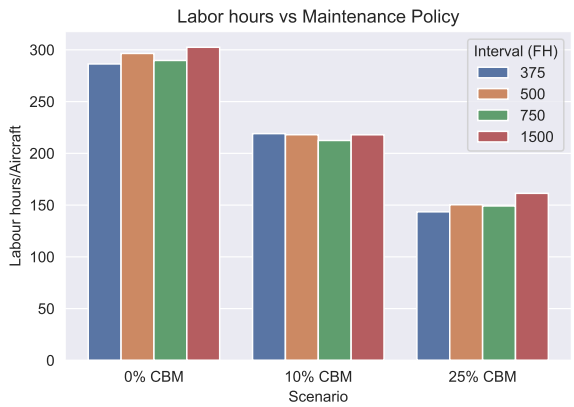
Figure C.1: Escalated tasks results



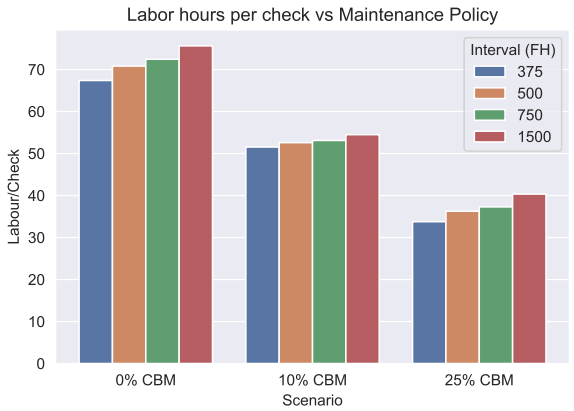
(a) Substitution tasks repetitions



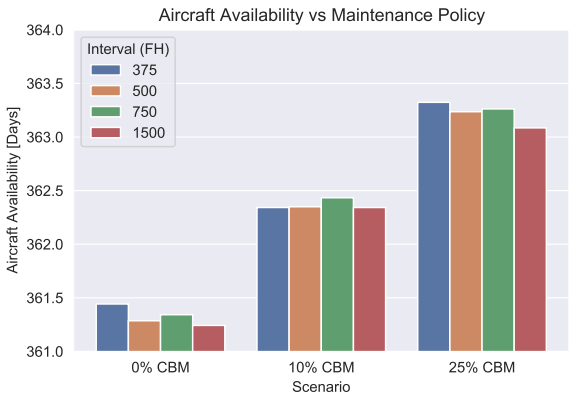
(b) Total Tasks repetitions



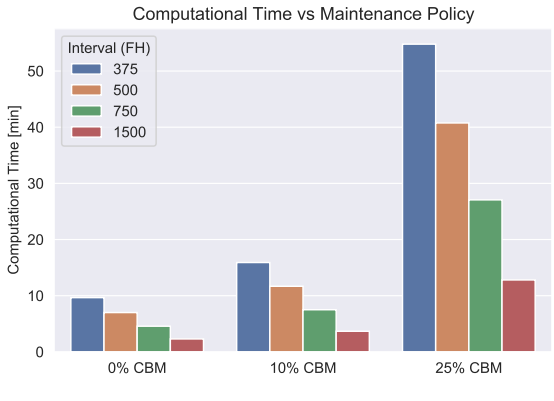
(c) Maintenance Labor hours



(d) Maintenance Labor hours per cycle

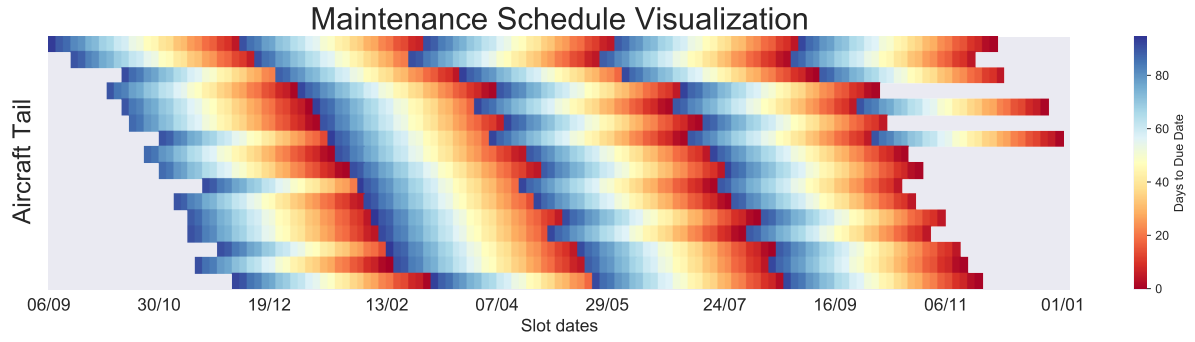


(e) Aircraft Availability

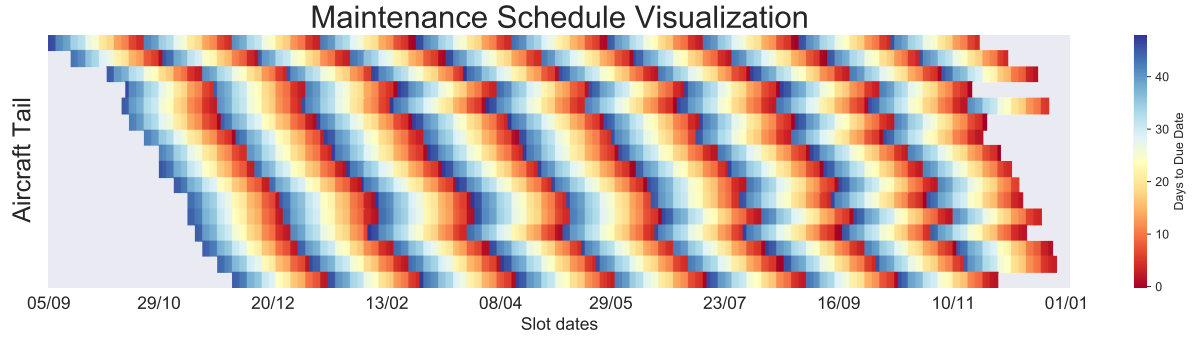


(f) Computational time per scenario

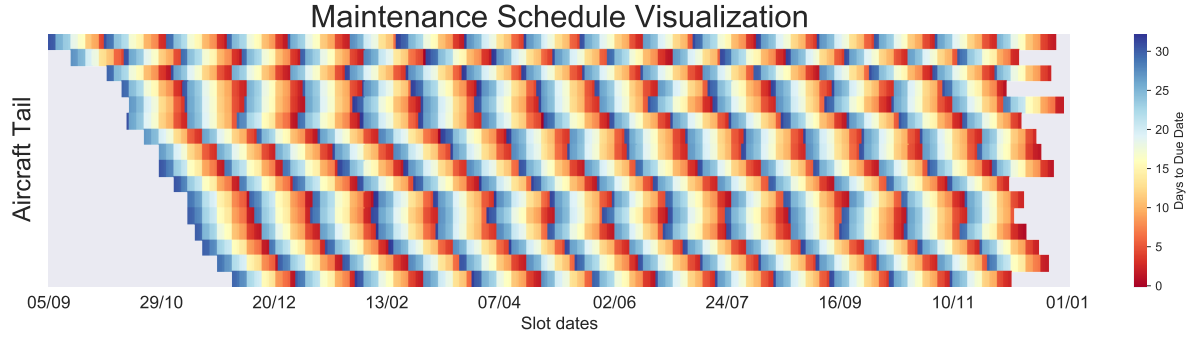
Figure C.2: Visualization of results with 15% uncertainty and 100% interval escalation



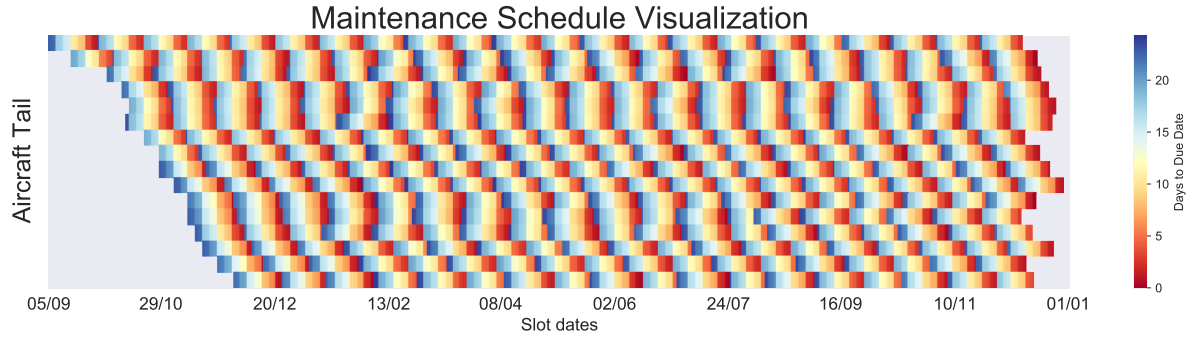
(a) 1500FH Policy



(b) 750FH Policy



(c) 500FH Policy



(d) 375FH Policy

Figure C.3: Maintenance schedule visualization example for the explored interval policies