A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization

Deng, Qichen; Santos, Bruno F.; Curran, Richard

DOI
10.1016/j.ejor.2019.08.025

Publication date
2020

Document Version
Final published version

Published in
European Journal of Operational Research

Citation (APA)
Deng, Q., Santos, B. F., & Curran, R. (2020). A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization. *European Journal of Operational Research*, *281*(2), 256-273. https://doi.org/10.1016/j.ejor.2019.08.025

European Journal of Operational Research 281 (2020) 256–273

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing, Transportation and Logistics

A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization

Qichen Deng*, Bruno F. Santos, Richard Curran

Section of Air Transport and Operations Delft University of Technology, The Netherlands

ARTICLE INFO

Article history:
Received 6 November 2018
Accepted 12 August 2019
Available online 16 August 2019

Keywords:
Scheduling
Aircraft maintenance
Dynamic programming
Forward induction

ABSTRACT

This paper presents a practical dynamic programming based methodology to optimize the long-term maintenance check schedule for a fleet of heterogeneous aircraft. It is the first time that the long-term aircraft maintenance check schedule is optimized, integrating different check types in a single schedule solution. The proposed methodology aims at minimizing the wasted interval between checks. By achieving this goal, one is also reducing the number of checks over time, increasing aircraft availability and, therefore, reducing maintenance costs, while respecting safety regulations. The model formulation takes aircraft type, status, maintenance capacity, and other operational constraints into consideration. We also validate and demonstrate the proposed methodology using fleet maintenance data from a European airline. The outcomes show that, when compared with the current practice, the number of maintenance checks can be reduced by around 7% over a period of 4 years, while computation time is less than 15 minutes. This could result in saving worth \$1.1M–\$3.4M in maintenance costs for a fleet of about 40 aircraft and generating more than \$9.8M of revenue due to higher aircraft availability.

1. Introduction

Aircraft maintenance is the overhaul, repair, inspection or modification of an aircraft or aircraft systems, components and structures in an airworthy condition (Minister of Justice, 2012). Regular maintenance prevents aircraft components and systems failures during operations. It takes place when an aircraft undergoes certain flight hours, flight cycles or calendar months. There are three major types of maintenance: A-check, B-check[1], C-check and D-check. A typical A-check includes inspection of the interior or exterior of the airplane with selected areas opened (e.g., checking and servicing oil, filter replacement and lubrication) (Ackert, 2010); they are performed approximately every 2–3 months. The C-check requires thorough inspection of individual systems and components for serviceability and function; it is planned within an interval of 18–24 months. The D-check (a.k.a Structural Check) uncovers the airframe, supporting structure and wings for inspection of most structurally significant items; it is carried out every 6–10 years. Many airlines merge D-check into C-check and label it as a heavy C-check. During a C-check or D-check, the aircraft has to be grounded for several weeks and removed from the revenue schedule. This paper for the first time optimizes the long-term integrated A- and C-check scheduling problem. We call this problem the aircraft maintenance check scheduling problem, or for short the AMCS problem.

Scheduling the maintenance inspection for a large heterogeneous fleet is generally a demanding and complex problem. In practice, the aircraft maintenance schedules are usually prepared according to the experience of maintenance operators. The main problem associated with such planning approach is that it is time consuming and it can result in poor solutions. For a large fleet, the maintenance operators need to spend several days or weeks planning the A- and C-checks one after another according to individual aircraft inspection interval and maintenance resource of the airline. If conflict A- or C-checks occur, the maintenance operator needs to adjust the schedule, constantly moving checks to earlier or later time slots until a feasible schedule is found. Limited by the manual planning approach, the goal is usually to find a feasible maintenance schedule for a fleet instead of an optimal one (Buskirk et al., 2002). As a result, the traditional manual maintenance planning approach inevitably decreases the aircraft utilization and leads to more A- and C-checks in the long term, increasing aircraft maintenance cost. Therefore, an optimized long-term maintenance schedule reduces the number of maintenance checks and increases aircraft availability, the saving derived from efficient maintenance planning can be very substantial.

* Corresponding author.
E-mail address: q.deng@tudelft.nl (Q. Deng).

[1] B-checks are rarely mentioned in practice. The tasks that could be included in B-checks are commonly incorporated into successive A-checks.

https://doi.org/10.1016/j.ejor.2019.08.025
0377-2217/© 2019 Elsevier B.V. All rights reserved.

Nowadays, airlines are laying increasing emphasis on improving their aircraft availability and planning their maintenance in a more efficient way. Aircraft maintenance represents one of the main direct operating costs and plays an important role on the balance sheet of an airline. According to Horder (2003), 11% of the total cost of an airline goes to aircraft maintenance. In 2014, this was equivalent to $295M on average per year per airline (IATA's Maintenance Cost Task Force, 2015). The long-term economical and operational benefits of adopting a more efficient approach are clear; a typical C-check of A320 family may cost $150k–$350k (Ackert, 2010), an A-check costs around $10k–$15k, while an additional day on operation may represent $75k–$120k of commercial revenue (depending on the utilization level of aircraft). However, the challenge is that A- and C-check scheduling problems are two correlated combinatorial problems. The decision of scheduling or not scheduling A-/C-check on an aircraft today impacts on the utilization of the aircraft onwards and, therefore, on the need to perform A-/C-checks in the future. This kind of problem is hard to solve and is often addressed by heuristics or algorithms (Steiner, 2006).

In this paper we propose a dynamic programming (DP) based method to solve the AMCS problem. The main contribution of this work can be summarized in the following three aspects:

1. Methodology:
    - An innovative and tractable DP-based model formulation is presented, suitable to solve real-life, large scale scheduling problems.
    - A thrifty algorithm is used to infer future implications of an action taken in the current time stage.
2. Practicality
    - The optimization takes the inspection interval of different check types and detailed operations into consideration.
    - It takes less than 15 minutes to optimize the 4-year A- and C-check schedule for more than 40 aircraft, rather than days or weeks.
3. Application
    - For the first time, the long-term AMCS problem is formulated and optimized by a single algorithm.
    - The formulation is flexible and other maintenance events can be easily included in the proposed model, such as landing gear maintenance or cabin modification.

The outline of this paper is as follows: Section 2 reviews the literature about aircraft maintenance planning and solution techniques for scheduling problems. The aircraft maintenance requirements, constraints and AMCS problem formulation, for aircraft A- and C-check scheduling, are presented in Section 3. The DP based methodology for A- and C-check scheduling optimization is discussed in detail in Section 4. Section 5 describes the case study from a European airline. The last section summarizes the research with concluding remarks and gives an outlook to future work.

2. Literature overview

Aircraft maintenance scheduling has been relying on the manual planning approach for many years. Since the introduction of commercialized wide-body aircraft in the early 1970s, the maintenance scheduling has become increasingly difficult due to the emphasis on efficiency and lack of an accurate and timely maintenance scheduling tool. It usually took several weeks for planning personnel to create a maintenance schedule (Boere, 1977). Air Canada was aware of this issue and developed in the 1970's an aircraft maintenance operations simulation model (AMOS) to improve maintenance efficiency and reduce labor and material cost (Boere, 1977). The AMOS tool formulated the aircraft maintenance scheduling as a discrete integer programming problem. According to the author, the problem constraints in AMOS included manpower, public holidays and summer time (when no maintenance was allowed), contractual maintenance duties for other airlines, reliability and required inspection intervals from the maintenance planning document (MPD). Several assumptions had been made in AMOS: each maintenance event ties up only one hangar/slot; the minimum time unit is one calendar day; aircraft is aged by daily flight hours; maintenance events can be postponed from the desired due date within a certain tolerance. Although AMOS works for both C-check and A-check scheduling, with the objective of minimizing total unused flight hours between two successive C-checks, a priority-based simulation heuristic is used to produce (good) feasible solutions. The author claims that neither mixed-integer programming or dynamic programming is deemed suitable to Air Canada's environment. Furthermore, due to rapidly changing of aircraft utilization and other unforeseen events, the author did not see the value of using computational power to find an optimal solution that could rapidly become obsolete. Therefore, a simulation-based approach was adopted in AMOS, in which the user is the one that chooses the best solution. The process is very similar to the manual planning approach, shifting conflict checks to earlier time slots until a feasible solution is found, except that a lower bound of utilization was implemented to prevent scheduling checks too often. Still, the main contribution of AMOS was to propose a systematic maintenance scheduling approach that could reduce the time required to generate a 5-year plan from 3 weeks to a few hours (Boere, 1977).

Despite the limitations in AMOS, this is together with Etschmaier and Franke (1969) and Bauer-Stämpfli (1971) the only available reference devoted to long-term aircraft maintenance scheduling. The long-term planning still has not been adequately studied. In fact, most research works about the aircraft maintenance topic focus on the short-term aircraft maintenance routing. That is, ensuring that each aircraft is assigned to a sequence of flight legs (a routing) that allows the aircraft to undergo daily checks, which are needed every two to four days (Belobaba, Odoni, & Barnhart, 2009). The main reason is that aircraft A-/C-checks have intervals of several months/years and the benefits of an optimal schedule are only visible in the long term. Airlines usually have higher urgency to monitor and optimize short-term activities, such as aircraft routing and routine aircraft inspections, from which they can rapidly see tangible cost savings and profits.

Feo and Bard (1989) is one of the first works to address the aircraft maintenance routing problem. It primarily focuses on the flight schedule design and incorporates the maintenance requirement as part of the constraints. A homogeneous fleet and fixed time intervals between maintenance checks were considered for simplicity. Only A-checks are considered in this work, since C-checks are spaced at relatively large time intervals. The planned flight schedule minimizes the total maintenance cost and also determines the maintenance base for the aircraft which start and end at the same city. The problem is formulated as a min-cost, multi-commodity flow network with integer constraints. Column generation is applied to obtain an optimized solution. Although the main purpose of Feo and Bard (1989) is to design a flight schedule, it is considered as a significant step in maintenance planning. Several authors have followed this path and continued the research on aircraft maintenance routing, such as Moudani and Mora-Camino (2000), Papakostas, Papachatzakis, Xanthakis, Mourtzis, and Chryssolouris (2010), Başdere and Bilge (2014) and Liang, Feng, Zhang, Wu, and Chaovalitwongse (2015). For example, Moudani and Mora-Camino (2000) proposed a hybrid dynamic programming (DP) which recursively searched for the best maintenance schedule, followed by a greedy algorithm to solve the sequential maintenance schedule problem. This approach was developed specifically as an on-line fleet operations management decision support system, focused on providing improved daily aircraft assignment solutions based on a given aircraft maintenance check schedule.

Sriram and Haghani (2003) is one rare example where the focus has been shifted from flight schedule design to maintenance scheduling. The authors proposed a mixed random search and depth first search heuristic to minimize the total costs of A- and B-checks and inappropriate aircraft assignments. Different from other research works which emphasize flight schedule design and consider aircraft maintenance as constraint, the flight schedule is given as input and, the goal is to determine when, where and what type of maintenance check an aircraft should undergo. Although the main focus is on maintenance scheduling, this research is still considered as an extension of Feo and Bard (1989), the C-check scheduling has not been considered since long-term flight schedules are unknown.

Instead of scheduling letter checks (A-, C- and structural/D-check), an alternative that reduces maintenance cost and generates profits in the short-term is aircraft maintenance task scheduling. This recent approach reverses the conventional top-down stereotyped planning. It schedules tasks individually, which gives flexibility to maintenance operators to execute the tasks at the most appropriate time (Kinnison & Siddiqui, 2012; Senturk, Kavsaoglu, & Nikbay, 2010). The task-oriented planning concept and its application are illustrated in Senturk and Ozkol (2018). The case study claims that more than $4M can be saved over 72 days compared with the rigid letter checks. However, among the works concerning task-oriented planning approach, there is little information about the influence of fleet size, maintenance capacity and algorithm computation time. Since an aircraft can have about 2000–3000 maintenance tasks, the practicality of applying a task-oriented approach on planning a long-term maintenance schedule for a large fleet remains questionable.

In general, the literature in long-term maintenance scheduling is limited, when planning aircraft maintenance checks, researchers often resort to the solution techniques from more general scheduling problems. A list of objective functions, models and optimization methods of scheduling problems are summarized by Duffuaa and Al-Sultan (1997). Since scheduling problems usually involve integer decision variables and linear constraints, the most common approaches to such mixed-integer linear programming problems are heuristics, which rely heavily on comemrcial solver such as CPLEX (Go, Kim, & Lee, 2013; Kiefera, Schildeb, & Doerner, 2018); the other alternative is dynamic programming (DP). DP was proposed in the 1950s by Bellman, referring specifically to nesting smaller decision problems inside larger decisions (Dreyffus, 2002). It divides a large and complicated problem into stages and states. The smaller sub-problems contained within each state are solved faster than the initial problem and the optimal solution can be retrieved by examining the solutions from all sub-problems. DP was initially applied on single-machine production (Bomberger, 1966; Gascon & Leachman, 1988; Lawler, 1990) or single-machine maintenance scheduling problems (Graves & Lee, 1999). For example, the work from Bomberger (1966), which minimizes the total cost of producing different items from a single machine, is considered to be one of the first to motivate application of DP on scheduling problems, although it assumes that only one unit can be produced at a time and the demand rate of units is constant.

As the development of DP, the application has been gradually extended from scheduling of single machine to multiple machines. Most of the DP applications on multiple machines are related to power generation. One of the examples can be found in Pereira, Campodónico, and Kelmam (1998). It presented a study to optimize the cost of multiple reservoir systems over a long period considering the uncertainties of water inflow and equipment outage. Even though piecewise linear functions are introduced in the solution process to estimate the future operation cost (this avoids recursively computing the actual future operating cost), the application is still limited to low-dimensional problems, namely, a small number of reservoirs (Asamov, Salas, & Powell, 2016). When the number of reservoirs increases, the number of decisions, i.e., how much water should be kept in each reservoir, increases exponentially.

Several conclusions can be drawn from the review of literature. First of all, Boere (1977) is the only available reference for long-term A- and C-check scheduling although no optimization technique is implemented. Secondly, the long-term A- and C-check scheduling forms a typical large-scale combinatorial problem, but there is no standard approach or exact algorithm for such a problem type. Thirdly, aircraft A- and C-check scheduling on a fleet aggregate level is analogous to multiple unit scheduling, which can be treated with similar formulation and solution techniques. Fourthly, DP is capable of dealing with small-scale mixed-integer/combinatorial problems, but the classic DP is not applicable to large-scale problems. In the rest of the paper, we will present a DP based methodology to tackle the long-term aircraft A- and C-check scheduling optimization.

3. Problem formulation

In this section we formulate the AMCS problem, adopting the DP framework and the nomenclature presented in Appendix A. We start with an introduction of inspection intervals (3.1), followed by a list of assumptions (3.2). After that we explain the maintenance capacity and some common operational constraints (3.3). The formulation of the AMCS problem is then described, divided into decision space (3.4), definition of state (3.5), state transition (3.6), constraints formulation (3.7) and the objective function (3.8). The final subsection summarizes the optimization model formulation.

3.1. Maintenance inspection interval

In aviation industry, aircraft are aged by daily utilization with respect to 3 different usage parameters, calendar day (DY), flight hours (FH) and flight cycles (FC). One DY is a full 24 hours period; FH refers to the elapsed time between wheel lift off and touch down; and a FC is defined by a complete take-off and landing sequence. The inspection interval reflects to the maximum usage parameters allowed in operation. For example, the maintenance planning document (MDP) of the AIRBUS A320 family (AIRBUS, 2017) defines that an C-check interval corresponds to 730 DY, 7500 FH or 5000 FC; and 120 DY, 750 FH or 750 FC for the A-check.

After an A-/C-check the corresponding three usage parameters are set to 0 and a maintenance cycle is concluded. These maintenance cycles are associated with labels, referring to different task packages (i.e., A1, A2, A3, ... for the A-check and C1, C2, ... for the C-check). The A-check program is commonly divided into 4 cycles, in which A1 has similar task packages as that of A5, while A2 has similar task packages as that of A6, and so forth. The C-check program has 12 cycles and consists of continuous C-checks, whereby every three check (i.e., C3, C6, C9, ...) is a heavier check incorporating tasks from the D-check.

The aircraft MPD also includes an inspection interval tolerance. This tolerance allows operators to fit the maintenance schedule around maintenance capacity and operations constraints, as well as operations demand. However, in the case that tolerance is used in one maintenance cycle, the amount of DY, FH and FC used from the tolerance need to be deducted from the maximum usage parameter values for the next cycle. This guarantees that the maximum usage parameters are verified in the long term. The inspection interval tolerance should not be included as a planning option but it is commonly used in practice to accommodate deviations from the initial schedule.

Although having different usage parameters, the A-check and C-check scheduling problems are dependent on each other due to two reasons: the first is that when an aircraft is performing a C-check (or A-check), it will be grounded and the A-check (or C-check) usage parameters are not altered (i.e., the daily utilization of these parameters is equal to 0). The second reason is that, depending on the usage parameters for the A-check, it could be beneficial for the airline to merge the A-check within a C-check. This has the advantage of performing the A-checks without necessarily increasing the C-check duration and without using an A-check slot. On the other hand, to anticipate an A-check, merging an A-check within a C-check will increase the number of A-checks in the long-term.

3.2. Assumptions

Boere (1977) defined a list of *major conditions* for maintenance event scheduling, based on aircraft maintenance practice. In this paper, we adopt the first six of these assumptions (A.1–A.6) and add two more (A.7–A.8) necessary to define our approach. A.1–A.7 by far are commonly used among airlines. A.8 is added due to the fact that airlines do not have their flight schedule for future 4–5 years, thus flexible aircraft routing is assumed for long-term maintenance scheduling.

- A.1 Minimum time unit of the aircraft maintenance schedule is 1 DY.
- A.2 Aircraft ages by DY, daily FH and FC. The daily utilization, as well as the commercial peak seasons, can be estimated per aircraft according to historical data.
- A.3 Each A-/C-check ties up only one hangar (slot) for its total duration.
- A.4 A-/C-check priority is defined according to the rule of "earliest deadline first", namely, aircraft which has earlier A-/C-check deadline is given higher A-/C-check priority, respectively.
- A.5 However, when looking at one particular aircraft, C-check has higher priority than A-check.
- A.6 A-check can be merged in C-check, which will not affect the C-check duration or existing A-check slots.
- A.7 The duration of an A-/C-check per check label can be estimated according to historical data or can be specified by airline.
- A.8 There is flexibility in aircraft routing to accommodate the A-/C-check and the geographical location of the hangars does not have to be specified.

The last assumption is based on the fact that the AMCS is a type of strategic problem. The aircraft routings are still not known, since they are only defined a couple of weeks before operations. For this same reason, the location of the aircraft at the time of the maintenance checks is unknown. Therefore, the geographical location of the A-/C-checks is not considered in this work. Nevertheless, the formulation presented can be easily adapted to incorporate different locations of the hangars and constraints regarding the allocation of a given fleet to specific hangar locations.

3.3. Maintenance capacity and operational constraints

In this research work, we only consider one single maintenance location with multiple A-/C- check hangars, meaning that all aircraft will undergo A-/C-check in the main hub of the airline. Although later on in the formulation, we will see that the location of hangar can be easily incorporated in the constraints of the DP framework, we opt not to take it into consideration. The main reason is that the AMCS is a type of strategic problem, no long-term aircraft rotations or flight schedules are given for verification or validation. Therefore, multiple locations of performing A-/C-checks would become redundant and only complicate the formulation.

In the AMCS problem, the A-/C-check capacity can either be expressed as person-hour available during a working day or, equivalently, as the maximum parallel A-/C-check allowed per working day (defined as "time slots" or just "slots"). The capacity for each check type is not always constant over time. Airlines and maintenance, repair and overhaul (MRO) service providers usually have operational constraints that influence the maintenance capacity per day. For instance, during commercial peak season (e.g., the period of New Year, Easter, summer and Christmas) it is desirable for the airline to operate with the maximum fleet. Performing C-checks will lead to high commercial revenue loss and it is also common to have reduced or no checks during weekends and public holidays due to higher labor costs. A final example is the case where some maintenance capacity is pre-allocated to third-party aircraft and therefore cannot be considered in the airline maintenance schedule.

In this paper, we define the capacity $M_{h,t}^k$ for hangar h:

$$M_{h,t}^k = \begin{cases} 1 & \text{if hangar } h \text{ is available on day } t \text{ for type } k \text{ slots} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

This parameter has to be defined per day per hangar for the entire time horizon, reflecting capacity variations between peak season and off-peak season and between weekends and normal working days, according to the airline policy. The capacity $M_{h,t}^k$ for hangar h can also be set equal to zero if hangar h is reserved for a specific maintenance event, such as performing a landing gear change on an aircraft of the fleet or for a type k check of an third-party aircraft.

3.4. Decision space

An action x_t of day t is to perform A-checks or C-checks, or do nothing:

$$x_t = \left\{ \left\{ \chi_{i,t}^k \right\}_{i=1}^N \right\}_{k \in \{\text{A, C}\}} \tag{2}$$

where each $\chi_{i,t}^k$ is a binary decision variables in which:

$$\chi_{i,t}^k = \begin{cases} 1 & \text{a type } k \text{ check for aircraft } i \text{ is planned to start} \\ & \quad \text{at time } t \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

3.5. Definition of state

A state vector s_t is defined by the set of attributes that influence our decisions and the available maintenance slots of each check type:

$$s_t = \left\{ \left\{ a_{i,t}^{\text{A}} \right\}_{i=1}^N, \ \left\{ a_{i,t}^{\text{C}} \right\}_{i=1}^N \right\} \tag{4}$$

where, each attribute set $a_{i,t}^k$ contains the information of aircraft i on day t, with respect to check type k:

$$a_{i,t}^k = \{\underbrace{M_t^k,\ z_{i,t}^k,\ \delta_{i,t}^k,\ \eta_{i,t}^k,}_{\text{Type 1 } \left(a_{i,t}^{(1),k}\right)} \underbrace{\text{DY}_{i,t}^k, \text{FH}_{i,t}^k, \text{FC}_{i,t}^k, \epsilon_{i,t}^{k\text{-DY}}, \epsilon_{i,t}^{k\text{-FH}}, \epsilon_{i,t}^{k\text{-FC}}, \theta_{i,t}^k, y_{i,t}^k,}_{\text{Type 2 } \left(a_{i,t}^{(2),k}\right)}$$
$$\underbrace{L_i^k\left(y_{i,t}^k\right), \text{fh}_{i,t}, \text{fc}_{i,t}}_{\text{Type 3 } \left(a_{i,t}^{(3),k}\right)}\} \tag{5}$$

These attributes are described in Appendix A and discussed in the next subsection. They can be divided into three types, as showed in Table 1.

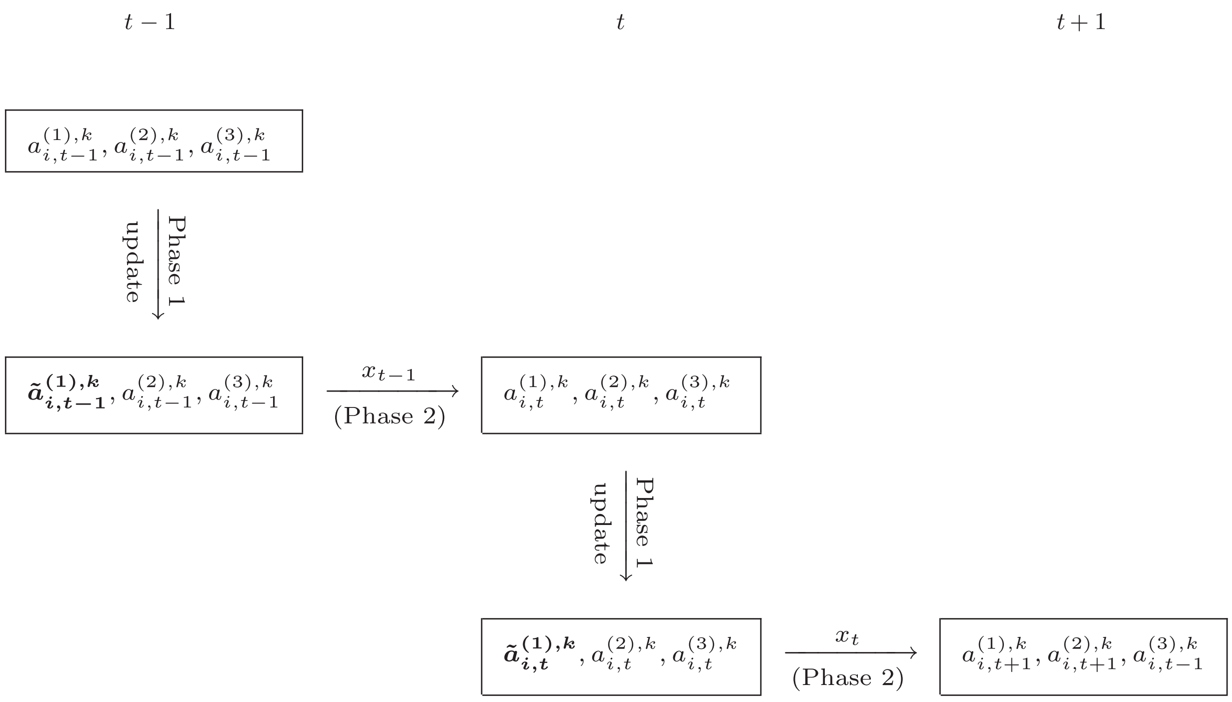

$t-1$
t
$t+1$
$a^{(1),k}_{i,t-1}, a^{(2),k}_{i,t-1}, a^{(3),k}_{i,t-1}$
update Phase 1
$\tilde{a}^{(1),k}_{i,t-1}, a^{(2),k}_{i,t-1}, a^{(3),k}_{i,t-1}$
x_{t-1} (Phase 2)
$a^{(1),k}_{i,t}, a^{(2),k}_{i,t}, a^{(3),k}_{i,t}$
update Phase 1
$\tilde{a}^{(1),k}_{i,t}, a^{(2),k}_{i,t}, a^{(3),k}_{i,t}$
x_t (Phase 2)
$a^{(1),k}_{i,t+1}, a^{(2),k}_{i,t+1}, a^{(3),k}_{i,t-1}$

Fig. 1. A two-phase attribute update mechanism: Phase 1 updates the set of pre-decision Type 1 attribute $a^{(1),k}_{i,t}$ to $\tilde{a}^{(1),k}_{i,t}$ before making any action; after an action x_t is made, Phase 2 updates $\tilde{a}^{(1),k}_{i,t}$, $a^{(2),k}_{i,t}$ and $a^{(3),k}_{i,t}$ to $a^{(1),k}_{i,t+1}$, $a^{(2),k}_{i,t+1}$ and $a^{(3),k}_{i,t+1}$.

Table 1
Different types of attribute within a state s_t.

Type 1 $a^{(1),k}_{i,t}$	Attributes at time t that impact the action x_t and are modified only when a check starts or ends, or when an aircraft is grounded
Type 2 $a^{(2),k}_{i,t}$	Attributes at time t that are updated every time based on their value at time $t-1$
Type 3 $a^{(3),k}_{i,t}$	Attributes at time t that depend on exogenous information

3.6. State transition

The transition between states in subsequent time steps depends on the actions taken. This can be described by a *state transition function* in which the state s_{t+1} is defined as a function of the initial state s_t and the action x_t chosen in state s_t:

$$\begin{cases} x_t = \mathcal{X}^{\pi}(s_t) \\ s_{t+1} = \mathcal{S}^{\mathcal{X}}(s_t, x_t) \end{cases} \quad \text{for} \quad t = t_0,\ t_0+1,\ \dots,\ T \tag{6}$$

where $\mathcal{X}^{\pi}(s_t)$ generates actions based on s_t according to A- and C-check hangar capacities at day t, M^{A}_t and M^{C}_t. The state transition function $\mathcal{S}^{\mathcal{X}}(s_t, x_t)$ describes how the state vector is updated and expresses the fact that an action taken at time t influences the future maintenance activities and capacities. A history of such process, including the sequence of actions and evolution of state, can be represented as:

$$\left(s_{t_0},\ x_{t_0},\ s_{t_0+1},\ x_{t_0+1},\ s_{t_0+2},\ \dots,\ s_{t-1},\ x_{t-1},\ s_t,\ \dots,\ s_T,\ x_T,\ s_{T+1}\right) \tag{7}$$

The main purpose of state transition is to renew the attributes over the time horizon. The attributes are updated in two phases: pre-decision (Phase 1) and post-decision (Phase 2). The goal of the pre-decision phase is to update the hangar capacity and aircraft availability for time t, before any decision is made. This provides the information about how many hangars can be used to perform A-/C-checks and which aircraft is available for operation. In Phase 1, only Type 1 attribute ($a^{(1),k}_{i,t}$) is updated and the resulting attributes ($\tilde{a}^{(1),k}_{i,t}$) from the pre-decision update is defined as pre-decision attributes. On the other hand, the goal of the post-decision phase is to update aircraft usage parameters, according to the action x_t that made on day t. All 3 type attributes will be updated in Phase 2 and we call the subsequent attributes from Phase 2 update post-decision attributes. Since the attributes of a state are divided into three types (Table 1), the transition of each type of attributes is presented separately in the following sub-sections.

3.6.1. Update of Type 1 attributes

Phase 1, which we called the pre-decision phase (Fig. 1), only updates the Type 1 attributes $a^{(1),k}_{i,t}$. In this phase, we check if at time t is the end day for an ongoing aircraft check. The new vector within the pre-decision attributes of type 1 is $\tilde{a}^{(1),k}_{i,t} = \left\{\tilde{M}^k_t,\ \tilde{z}^k_{i,t},\ \tilde{\delta}^k_{i,t},\ \tilde{\eta}^k_{i,t}\right\}$. The pre-decision update is triggered by verifying if the end date ($z^k_{i,t}$) of an ongoing check is equal to $t-1$ (i.e., if $z^k_{i,t} = t-1$), for any aircraft in the fleet:

$$\tilde{z}^k_{i,t} = \begin{cases} 0 & \text{if } z^k_{i,t} = t-1 \\ [5pt] z^k_{i,t} & \text{otherwise} \end{cases} \tag{8}$$

At the same time, we update $\delta^k_{i,t}$ to $\tilde{\delta}^k_{i,t}$:

$$\tilde{\delta}^k_{i,t} = \begin{cases} 0 & \text{if } z^k_{i,t} = t-1 \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

If the end date of a type k check for an aircraft i is larger than the current calendar day t, it means that there is an aircraft check occurring. And the hangar capacity needs to be updated for time t accordingly:

$$\tilde{M}^k_t = \sum_h M^k_{h,t} - \sum_{i=1}^{N} \tilde{\delta}^k_{i,t} \tag{10}$$

where $M^k_{h,t}$ is the maintenance capacity per hangar h at time t. The value of $\tilde{\eta}^k_{i,t}$ is initialized using $\eta^k_{i,t}$, namely, $\tilde{\eta}^k_{i,t} = \eta^k_{i,t}$.

In Phase 2, or the post-decision phase (see Fig. 1), the action x_t is taken into account to update Type 1 attributes. For all aircraft that start type k check on day t ($\chi^k_{i,t} = 1$), the values of $\delta^k_{i,t}$ and $z^k_{i,t}$ need to be updated. The $z^k_{i,t}$ is updated according to:

$$z^k_{i,t+1} = \begin{cases} t + L^k_i\left(y^k_{i,t}\right) & \text{if } \chi^k_{i,t} = 1 \\ \tilde{z}^k_{i,t} & \text{otherwise} \end{cases} \tag{11}$$

where $L^k_i\left(y^k_{i,t}\right)$ is the elapse time for maintenance type k, with label $y^k_{i,t}$. Following this update, the values of $\delta^k_{i,t}$ can also be renewed:

$$\delta^k_{i,t+1} = \begin{cases} 0 & \text{if } \chi^k_{i,t} = 1 \\ \tilde{\delta}^k_{i,t} & \text{otherwise} \end{cases} \tag{12}$$

Still in the post-decision stage, in some special cases, aircraft reach their inspection intervals and no maintenance check capacity is available. In these undesirable situations, the aircraft needs to be grounded and put out of operations, waiting for the next maintenance opportunity. This happens if the usages parameters for time $t+1$ of any aircraft is larger than the respective inspection interval. We first compute the expected usage parameters of $t+1$ as follows:

$$\Delta \mathrm{DY}^k_{i,t+1} = \underbrace{\left(\mathrm{DY}^k_{i,t} + 1\right)}_{\mathrm{DY}^k_{i,t+1}} - \underbrace{\left[I^i_{k\text{-DY}} + \left(1 - \theta^k_{i,t}\right)e^i_{k\text{-DY}} - \epsilon^{k\text{-DY}}_{i,t}\right]}_{\text{Actual DY Interval of Type } k \text{ Check}} \tag{13}$$

$$\Delta \Psi^k_{i,t+1} = \underbrace{\left(\Psi^k_{i,t} + \psi_{i,t}\right)}_{\text{usage parameters of } t+1} - \underbrace{\left[I^i_{k\text{-}\Psi} + \left(1 - \theta^k_{i,t}\right)e^i_{k\text{-}\Psi} - \epsilon^{k\text{-}\Psi}_{i,t}\right]}_{\text{Actual Interval of } \Psi \ (\Psi \in \{\mathrm{FH}, \mathrm{FC}\}) \text{ of Type } k \text{ Check}} \tag{14}$$

where we use $\Psi^k_{i,t+1} \in \{\mathrm{FH}^k_{i,t+1}, \mathrm{FC}^k_{i,t+1}\}$, $\Psi \in \{\mathrm{FH}, \mathrm{FC}\}$ and $\psi^k_{i,t+1} \in \{\mathrm{fh}^k_{i,t+1}, \mathrm{fc}^k_{i,t+1}\}$ for convenience. We separate DY from other usage parameters because its utilization update is different from FH or FC. $\mathrm{DY}^k_{i,t}$, $\mathrm{FH}^k_{i,t}$ and $\mathrm{FC}^k_{i,t}$ are the cumulative DY, FH and FC since previous type k check till day t; $I^i_{k\text{-DY}}$, $I^i_{k\text{-FH}}$ and $I^i_{k\text{-FC}}$ refer to the standard interval of type k check; $(1-\theta^k_{i,t})e^i_{k\text{-DY}}$, $(1-\theta^k_{i,t})e^i_{k\text{-FH}}$ and $(1-\theta^k_{i,t})e^i_{k\text{-FC}}$ represent the respective tolerance that can be added to the standard interval; and $\epsilon^{k\text{-DY}}_{i,t}$, $\epsilon^{k\text{-FH}}_{i,t}$ and $\epsilon^{k\text{-FC}}_{i,t}$ represent the amount of DY, FH and FC tolerance used in previous type k check (and that needs to be deducted from the coming maintenance check according to the aircraft MPD).

The aircraft is then grounded if any of these previous delta values is greater than 0 and no maintenance check is being performed on the aircraft:

$$\eta^k_{i,t+1} = \begin{cases} 1 & \chi^k_{i,t} = 0, \max\left\{\Delta\mathrm{DY}^k_{i,t+1}, \Delta\mathrm{FH}^k_{i,t+1}, \Delta\mathrm{FC}^k_{i,t+1}\right\} > 0 \\ \tilde{\eta}^k_{i,t} & \text{otherwise} \end{cases} \tag{15}$$

3.6.2. Update of Type 2 attributes

Once the action of day t is known, the update of Type 2 attributes is trivial. The aircraft usage parameters are updated according to the following equations:

$$\mathrm{DY}^k_{i,t+1} = \left(1 - \delta^k_{i,t}\right)\left(\mathrm{DY}^k_{i,t} + 1\right) \tag{16}$$

$$\Psi^k_{i,t+1} = \left(1 - \delta^k_{i,t}\right)\left[\Psi^k_{i,t} + \left(1 - \delta^{\{\mathrm{A,C}\}\backslash k}_{i,t}\right)\psi_{i,t}\right] \tag{17}$$

where $\delta^{\{\mathrm{A,C}\}\backslash k}_{i,t}$ means if $k = \mathrm{A}$ then $\delta^{\{\mathrm{A,C}\}\backslash k}_{i,t} = \delta^{\mathrm{C}}_{i,t}$ and vice versa. The parameters are reset to 0 if a maintenance check of type k was scheduled in the previous time step (i.e., $\delta^k_{i,t} = 1$). Otherwise, the parameters are either increased by the average daily aging of the aircraft or kept constant, if a maintenance of the type other than k is scheduled (i.e., $\delta^{\{\mathrm{A,C}\}\backslash k}_{i,t} = 1$).

The update of Type 2 attributes also includes renewing the tolerance usage variables for each maintenance check type k:

$$\epsilon^{k\text{-DY}}_{i,t+1} = \begin{cases} \max\left\{0, \Psi^k_{i,t} - I^i_{k\text{-DY}}\right\} & \text{if } \chi^k_{i,t} = 1 \\ \epsilon^{k\text{-DY}}_{i,t} & \text{otherwise} \end{cases} \tag{18}$$

$$\epsilon^{k\text{-}\Psi}_{i,t+1} = \begin{cases} \max\left\{0, \Psi^k_{i,t} - I^i_{k\text{-}\Psi}\right\} & \text{if } \chi^k_{i,t} = 1 \\ \epsilon^{k\text{-}\Psi}_{i,t} & \text{otherwise} \end{cases} \tag{19}$$

where $\Psi \in \{\mathrm{FH}, \mathrm{FC}\}$. (18) and (19) indicate that the status of tolerance usage of a type k check is the same as the day before if there is no type k check allocated on day t. On the contrary, if a type k check is scheduled before all usage parameters reach maximum, then no tolerance is used and $\epsilon^{k\text{-DY}}_{i,t+1}/\epsilon^{k\text{-FH}}_{i,t+1}/\epsilon^{k\text{-FC}}_{i,t+1}$ are set to 0. If an aircraft has to operate over the limit of a type k check, the corresponding $\epsilon^{k\text{-DY}}_{i,t+1}/\epsilon^{k\text{-FH}}_{i,t+1}/\epsilon^{k\text{-FC}}_{i,t+1}$ are updated according to the difference between the cumulative DY/FH/FC and type k check interval. As a result, the tolerance usage indicators will be renewed:

$$\theta^k_{i,t+1} = \begin{cases} 1 & \text{if } \max\left\{\epsilon^{k\text{-DY}}_{i,t+1}, \epsilon^{k\text{-FH}}_{i,t+1}, \epsilon^{k\text{-FC}}_{i,t+1}\right\} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

After an action is evaluated, the maintenance labels for both type k checks are updated consequently. The maintenance labels of an aircraft i are updated to the next label using the following equation:

$$y^k_{i,t+1} = \begin{cases} y^k_{i,t} + 1 & \text{if } \chi^k_{i,t} = 1 \\ y^k_{i,t} & \text{otherwise} \end{cases} \tag{21}$$

3.6.3. Update of Type 3 attributes

The Type 3 attributes are exogenous variables that are updated according to lookup tables, or provided by an airline, or estimated according to historical data of airline. They refer to:

- $L^k_i\left(y^k_{i,t+1}\right)$ is the elapsed time specified by airline.
- $\mathrm{fh}^k_{i,t+1}$ is estimated according to historical aircraft FH.
- $\mathrm{fc}^k_{i,t+1}$ is estimated according to historical aircraft FC.

3.7. Constraints formulation

There are two types of constraints in the A- and C-check scheduling optimization: A-/C-check intervals and A-/C-check operational constraints. The A-/C-checks are usually scheduled before the corresponding usage parameters reach maximum. That not being possible, in practice the airline can make use of the interval tolerance. The extra DY/FH/FC used from tolerance must be compensated in the next type k check, as mentioned in Section 3.6.1. This can be described as follows, for each check k, aircraft i, and time t:

$$\mathrm{DY}^k_{i,t+1} \le I^i_{k\text{-DY}} + \left(1 - \theta^k_{i,t}\right)e^i_{k\text{-DY}} - \epsilon^{k\text{-DY}}_{i,t} \tag{22}$$

$$\Psi^k_{i,t+1} \le I^i_{k\text{-}\Psi} + \left(1 - \theta^k_{i,t}\right)e^i_{k\text{-}\Psi} - \epsilon^{k\text{-}\Psi}_{i,t} \tag{23}$$

where $\Psi \in \{\mathrm{FH}, \mathrm{FC}\}$; the first term of the right-hand side of each inequality refers to the standard check interval, the second terms adds the tolerance interval, and the last term subtracts the tolerance used in the previous check of the same type.

Before instigating any action, we need to verify whether or not there are sufficient slots for a type k check in a hangar during the entire maintenance elapse time $L_i^k(y_{i,t}^k)$, for all aircraft and hangars available:

$$\chi_{i,t}^k \le \frac{\sum_{\tau=t}^{t+L_i^k(y_{i,t}^k)} M_{h,\tau}^k}{L_i^k\left(y_{i,t}^k\right)}, \quad k \in \{\text{A}, \ \text{C}\}, \quad t \in [t_0, T] \tag{24}$$

The operational constraints are required to guarantee that the number of A-/C-checks performed in parallel per day do not exceed the hangar capacity, namely:

$$\sum_{i=1}^{N} \delta_{i,t}^k \le \sum_h M_{h,t}^k, \quad k \in \{\text{A}, \ \text{C}\}, \quad t \in [t_0, T] \tag{25}$$

Some airlines require a minimum number of days (d_k) between the start dates of two type k checks preparing the maintenance resources, such as tools, manpower, aircraft spare parts and to avoid parallel peaks of workload at the hangar, meaning that:

- If $d_k > 0$, there can be at most 1 aircraft starts a type k check at time t.
- If $d_k > 0$ and there is a type k check starting at t, no type k check is allowed to start in $[t, t + d_k)$

The requirement of start date can be translated in the following equations for all time t:

$$\sum_{i=1}^{N} \chi_{i,t}^k \le \begin{cases} 1 & \text{if } d_k > 0 \text{ and } \sum_{i=1}^{N} \chi_{i,\tau}^k = 0, \ \forall \tau \in [t - d_k, t) \\ M_t^k & \text{otherwise} \end{cases} \tag{26}$$

Note that we use a generic indicator h to represent an A-/C-check hangar in this paper, based on the assumption A8. If one wants to consider multiple locations of perform the aircraft A-/C-check, each hangar h would have to be associated with a location l_h and the decision variable $\delta_{i,t}^k$ will be replaced by $\delta_{i,t}^{l_h,k}$.

3.8. Objective function

When scheduling aircraft maintenance activities, the most common objectives are minimization of costs (Moudani & Mora-Camino, 2000; Sriram & Haghani, 2003) or minimization of the unused flight hours (FH) (Başdere & Bilge, 2014; Boere, 1977). In this work we consider the second objective. The cost minimization objective was not considered for three main reasons:

- The available maintenance cost data is unreliable and hard to associate to a specific maintenance check.
- Maintenance checks are mandatory and the total maintenance costs of an airline can only be reduced if the number of aircraft checks over time is also reduced.
- One day of an aircraft out of operations is more costly than the daily cost of a maintenance check.

Therefore, minimization of the unused FH, and consequently, in the long term, the reduction of the number of aircraft checks and days on the ground, is considered to be the best objective for the AMCS problem. For an aircraft i, the value of unused FH in a day t is equal to the summation of the FH loss due to an A- or a C-check scheduled for that day:

$$\chi_{i,t}^{\text{A}}\left(I_{\text{A-FH}}^i - \text{FH}_{i,t-1}^{\text{A}}\right) + \chi_{i,t}^{\text{C}}\left(I_{\text{C-FH}}^i - \text{FH}_{i,t-1}^{\text{C}}\right) \tag{27}$$

The *contribution function* of FH loss on day t is calculated by:

$$C_t(s_t, x_t) = \sum_{k \in \{\text{A,C}\}} \sum_{i=1}^{N} \left[\chi_{i,t}^k\left(I_{k\text{-FH}}^i - \text{FH}_{i,t}^k\right) + \left(1 - \chi_{i,t}^k\right) P_a \theta_{i,t}^k + P_d \eta_{i,t}^k\right] \tag{28}$$

where the first term on the right hand side reflects the unused FH of aircraft i, the second term is a penalty for aircraft i using an interval tolerance, and the third term is a penalty for having an aircraft on the ground without doing maintenance.

The penalty P_a is introduced due to the fact that the use of tolerance needs to be communicated and approved by the local civil aviation authorities. Therefore, tolerance should not be considered at a scheduling stage or, if inevitable, it should be used as little as possible. The second penalty is introduced to reflect the cost of having an aircraft on the ground waiting for a maintenance slot. This results in very high costs and it should always be avoided, unless it proves to be unfeasible otherwise. For that reason, the value of P_d should always be of a very large magnitude. Our objective is then to minimize the sum of the total contributions for all states visited during the time horizon, discounted by a factor γ. That is, we search for the optimal AMCS policy (π) that minimizes the contribution of our scheduling decisions over the time horizon $T - t_0$:

$$\min_{\pi} \ \mathbb{E}\left\{\sum_{t=t_0}^{T} \gamma^{t-t_0} C_t(s_t, \mathcal{X}^{\pi}(s_t))\right\} \tag{29}$$

where $\mathcal{X}^{\pi}(s_t)$ is the optimal scheduling policy function.

3.9. Optimization model

After the introduction of state transition, constraints and objective function, the optimization problem can be described by the following:

$$\min_{\pi} \ \mathbb{E}\left\{\sum_{t=t_0}^{T} \gamma^{t-t_0} C_t(s_t, \mathcal{X}^{\pi}(s_t))\right\} \tag{30}$$

subject to:

Constraints $(11) - (26)$

The optimal scheduling policy over the time horizon T can be found by recursively computing the Bellman's equation:

$$V_t(s_t) = \min_{x_t} \left\{C_t(s_t, x_t) + \gamma \sum_{s_{t+1}} p\left(s_{t+1} \middle| s_t, x_t\right) V_{t+1}(s_{t+1})\right\} \tag{31}$$

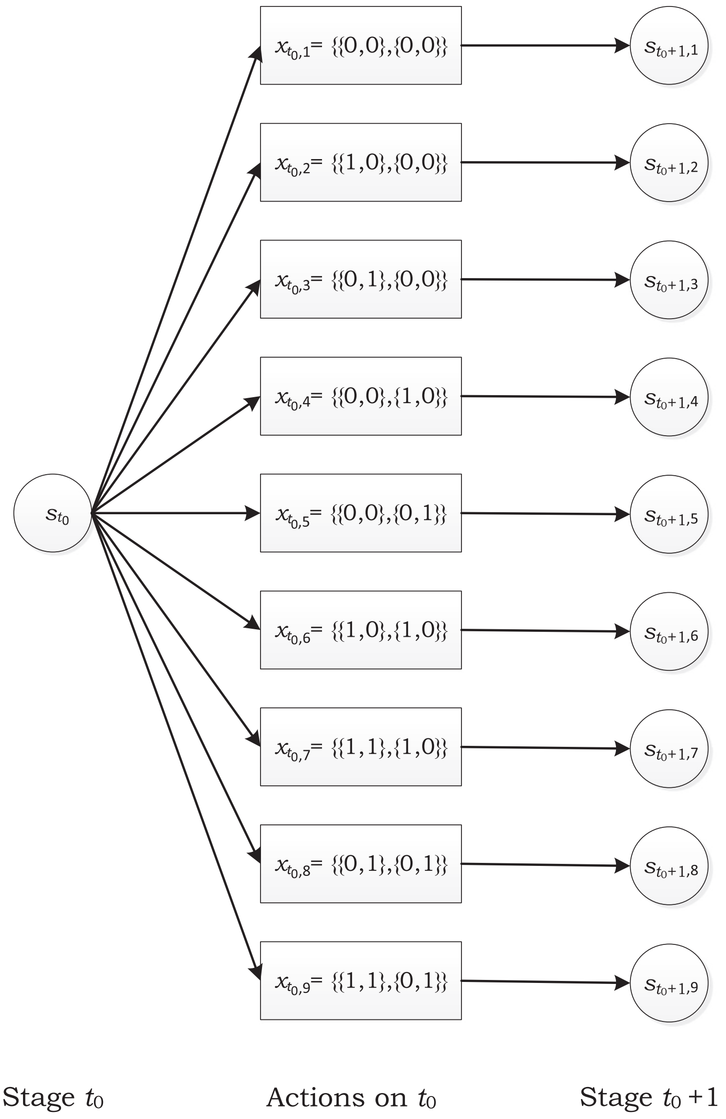
where $s_{t+1} = \mathcal{S}^X(s_t, x_t) = \mathcal{S}^X(s_t, \mathcal{X}^{\pi}(s_t))$ and $p\left(s_{t+1} \middle| s_t\right)$ is the probability of transitioning from state s_t to state s_{t+1}. The Bellman's equation expresses the value of being at each state S_t.

4. Methodology

The AMCS problem has a structure that follows the Markov Decision Process (MDP). Like any other MDP, it can be solved using dynamic programming (DP). The AMCS problem can be divided into stages, each stage referring to one calendar day (indexed by t). For each stage, all feasible actions x_t from a state s_t need to be evaluated and the optimal one x_t^* can be eventually identified. For illustration purpose, we use Fig. 2 to depict an example of state transition from stage t_0 to stage $t_0 + 1$ (deterministic). In this case, s_{t_0} is the initial state. There are two aircraft, 1 A-check slot and 1 C-check slot on stage t_0. The action vector x_t has the following structure:

$$x_t = \left\{ \underbrace{\{0, 0\}}_{\text{aircraft 1}}, \underbrace{\{0, 0\}}_{\text{aircraft 2}} \right\} \tag{32}$$

For each aircraft in (32), the first number indicates the action of A-check and the second number is for C-check. If an A-/C-check starts, the corresponding number is 1, and 0 otherwise. This gives 9 possible actions on stage t_0:



$x_{t_0,1}$= {{0,0},{0,0}}
$x_{t_0,2}$= {{1,0},{0,0}}
$x_{t_0,3}$= {{0,1},{0,0}}
$x_{t_0,4}$= {{0,0},{1,0}}
$x_{t_0,5}$= {{0,0},{0,1}}
$x_{t_0,6}$= {{1,0},{1,0}}
$x_{t_0,7}$= {{1,1},{1,0}}
$x_{t_0,8}$= {{0,1},{0,1}}
$x_{t_0,9}$= {{1,1},{0,1}}
s_{t_0}
$s_{t_0+1,1}$ … $s_{t_0+1,9}$
Stage t_0
Actions on t_0
Stage t_0+1

Fig. 2. An example of deterministic state transition from stage t_0 to stage t_0+1. $\{x_{t_0,i}\}$ is the set of possible actions and $\{s_{t_0+1,i}\}$ is the set of resulting states. In deterministic case, the transition probability is 1, thus there is only one resulting state per action. Similarly from stage t_0+1 on-wards, each $s_{t_0+1,i}$ has the same number of actions as s_{t_0}.

1) no A-check or C-check: $x_{t_0,1} = \{\{0,0\},\{0,0\}\}$
2) A-check on aircraft 1 but no C-check: $x_{t_0,2} = \{\{1,0\},\{0,0\}\}$
3) A-check on aircraft 2 but no C-check: $x_{t_0,3} = \{\{0,1\},\{0,0\}\}$
4) no A-check but C-check on aircraft 1: $x_{t_0,4} = \{\{0,0\},\{1,0\}\}$
5) no A-check but C-check on aircraft 2: $x_{t_0,5} = \{\{0,0\},\{0,1\}\}$
6) merge A- into C-check for aircraft 1 but no A-check on aircraft 2: $x_{t_0,6} = \{\{1,0\},\{1,0\}\}$
7) merge A- into C-check for aircraft 1 and A-check on aircraft 2: $x_{t_0,7} = \{\{1,1\},\{1,0\}\}$
8) no A-check on aircraft 1 but merge A- into C-check for aircraft 2: $x_{t_0,8} = \{\{0,1\},\{0,1\}\}$
9) A-check on aircraft 1 and merge A- into C-check for aircraft 2: $x_{t_0,9} = \{\{1,1\},\{1,0\}\}$

It can be observed that 9 possible actions lead to 9 states on stage t_0+1, even for an example of 1 A-check slot and 1 C-check slot (here we assumed that an A-check can be merged into a C-check, in deterministic case there is only 1 resulting state per action). This process repeats as the state transition proceeds, i.e., each state $s_{t_0+1,i}$ has 9 different actions and therefore will have 9 outcome states. As we move forward, the number of states within a stage will grow exponentially.

The value associated with each action can be computed using Bellman's optimality Eqs. (31). However, solving these for all possible actions is not trivial due to three challenges: the size of the multi-dimensional action vector x_t; the size of the multi-dimensional state vector s_t; and the very large outcome space. These challenges are well known as the "curse of dimensionality" (Powell, 2011). It is easy to understand these challenges if we analyze our AMCS problem, as the state vector s_t is a tuple that contains the states of N aircraft, and each aircraft has 28 attributes with respect to the A- and C-check. If one wants to use discretization for each attribute, e.g. to l levels, the total number of levels to access will be $l^{28}\times N$, just for a single stage. This requires a large amount of computer memory and also makes it difficult to trace decisions backwards. In terms of actions, for each time stage t and capacities M_t^{A} (A-check) and M_t^{C} (C-check), we would have:

$$\sum_{m_c=0}^{M_t^{\mathrm{C}}}\sum_{m_a=0}^{M_t^{\mathrm{A}}}\left[\frac{N!}{(N-m_a)!\times m_a!}\times\frac{N!}{(N-m_c)!\times m_c!}\right] \tag{33}$$

possible actions and, if no optimal final state is given, there will be:

$$\prod_{t=t_0}^{T} \sum_{m_c=0}^{M_t^C} \sum_{m_a=0}^{M_t^A} \left[\frac{N!}{(N-m_a)! \times m_a!} \times \frac{N!}{(N-m_c)! \times m_c!} \right] \tag{34}$$

possible outcomes for the last stage. This means that even for a small case with 10 aircraft and one daily slot available for each check type, we would have 121 possible actions on the first day and more than 1.7 million possible sequences of actions just after three days.

In classic dynamic programming, computing $V_{t+1}(s_{t+1})$ requires $V_{t+2}(s_{t+2})$, and in order to obtain $V_{t+2}(s_{t+2})$, the $V_{t+3}(s_{t+3})$ has to be computed and so forth, until t reaching the final stage T. The aforementioned solution process is called backward induction that, if the final state is not known, easily becomes intractable even for the small example of 10 aircraft. This forces us to treat the AMCS problem in a different way, adopting a *forward induction* approach which moves from the initial planning stage towards the future.

In this section we propose a forward induction DP based methodology to solve the AMCS problem. We begin with a brief introduction to the forward induction concept in Section 4.1. After that we describe a priority solution that is proposed in dealing with the multi-dimensional action vector (Section 4.2). Section 4.3 presents a *Thrifty Algorithm* for A- and C-check scheduling, which estimates the implications of an action at the current stage on the remaining planning horizon. Section 4.4 presents the discretization and aggregation approach adapted to implement the algorithm. The last subsection (Subsection 4.5) includes an algorithm complexity analysis.

4.1. Forward induction

Since the final state of our AMCS problem is not known, we propose to use a forward induction approach. Forward induction is the process of reasoning forward in time, determining a sequence of optimal actions from an initial state till the end of the time horizon. This comes from the observation that the shortest path from an initial node s_{t_0} to an end node s_{T+1} is equal to the shortest path from the end node to the initial node (Hoppe, 2018). That is, determining the optimal solution for the forward shortest path problem is the same as determining the optimal solution for the backward shortest path problem, as computed by the backward induction approach. The idea from the forward induction approach is to move forward in time, continuously computing the shortest path between the initial node s_{t_0} and the current node being tested s_t. This process is repeated until one has determined the best action for every stage in the time horizon.

Although the forward induction approach would solve the problem of not knowing the final state of our problem, due to the large number of intermediate states between the initial stage and the final stage of planning horizon, this approach is still inefficient for AMCS in terms of computation time and storage. For this reason, we incorporate forward induction with three additional components:

- *An A- and C-check priorities definition solution*
- *A Thrifty Algorithm for A- and C-check scheduling*
- *A discretization and state aggregation strategy.*

These components are labeled as blocks **A**, **B** and **C**, respectively, in the work flow diagram of forward induction (Fig. 3). The first component is used to deal with the multi-dimensional action vector; the second component is used to estimate the consequences of an action on future time steps; the third components is designed to reduce the outcome space to a manageable size. These three components are explained in the following subsections.

4.2. Defining A-/C-check priority

Given a state s_t, we generate an action based on s_t:

$$x_t = \mathcal{X}^{\pi}(s_t) \tag{35}$$

Note that x_t can be performing m_C C-check and m_A A-check (m_C and m_A depend on corresponding hangar capacity), meaning that the action affects the status of multiple aircraft. This leads to a combination of $\frac{(N!)^2}{m_C!(N-m_C)!m_A!(N-m_A)!}$ aircraft selection and therefore, $\frac{(N!)^2}{m_C!(N-m_C)!m_A!(N-m_A)!}$ outcome states.

We observe that the number of states can quickly explode due to such a multi-dimensional action vector (being an action on multiple items). One common solution to this challenge is to assign priorities to each aircraft and in our case, we define A- and C-check priorities (Block A in Fig. 3) according to the rule of "earliest deadline first". However, in order to know the deadline of an A-/C-check, we need to first compute the *remaining utilization*. Since there are 3 usage parameters for each check type in the AMCS problem, this gives 3 different remaining utilization with respect to DY, FH and FC, while a type k check should be scheduled before any of the remaining utilization goes to 0.

This way, we define the aircraft *remaining utilization* by the fewest days to the next A-/C-check:

$$R_{i,t}^{k} = \min\left\{R_{i,t}^{k\text{-DY}},\quad R_{i,t}^{k\text{-FH}},\quad R_{i,t}^{k\text{-FC}}\right\} \tag{36}$$

The $R_{i,t}^{k\text{-DY}}$, $R_{i,t}^{k\text{-FH}}$ and $R_{i,t}^{k\text{-FC}}$ refer to the remaining operation days with respect to each usage parameter and associated interval specified by the MPD:

$$R_{i,t}^{k\text{-DY}} = \max\ \arg_{r\in\mathbb{N}}\left\{r \le I_{k\text{-DY}}^{i} - \epsilon_{i,t}^{k\text{-DY}} - \mathrm{DY}_{i,t}^{k}\right\} \tag{37}$$

$$R_{i,t}^{k\text{-FH}} = \max\ \arg_{r\in\mathbb{N}}\left\{\sum_{\tau=t}^{t+r} \mathrm{fh}_{i,\tau} \le I_{k\text{-FH}}^{i} - \epsilon_{i,t}^{k\text{-FH}} - \mathrm{FH}_{i,t}^{k}\right\} \tag{38}$$

$$R_{i,t}^{k\text{-FC}} = \max\ \arg_{r\in\mathbb{N}}\left\{\sum_{\tau=t}^{t+r} \mathrm{fc}_{i,\tau} \le I_{k\text{-FC}}^{i} - \epsilon_{i,t}^{k\text{-FC}} - \mathrm{FC}_{i,t}^{k}\right\} \tag{39}$$

where $\mathbb{N}$ is the set of natural numbers for $k \in \{$A, C$\}$. At any given time t, the remaining utilizations are sorted in ascending order:

$$\tilde{R}_{1,t}^{k},\ \tilde{R}_{2,t}^{k},\ \tilde{R}_{3,t}^{k},\ \ldots,\ \tilde{R}_{N,t}^{k} \qquad \tilde{R}_{i,t}^{k} \le \tilde{R}_{i+1,t}^{k},\ \tilde{R}_{i,t}^{k} \in \left\{R_{i,t}^{k}\right\}_{i=1}^{N} \tag{40}$$

The aircraft are sent to maintenance check according to this sorted list while aircraft with a lower remaining utilization are given a higher check priority. Since the C-check is more restrictive and demanding in terms of resources, it has higher priority than an A-check. In addition to the available slots and maintenance elapsed time of the check type, we set the following rules for making A- and C-check decisions:

(i) No type k check should be scheduled if there is no available hangar for type k check on day t.

(ii) An aircraft i is allocated a type k check only if its remaining operation days is lower than the threshold ($R_{i,t}^{k} \le \overline{R}_k$) and there are available slots for type k check ($\exists h,\ M_{h,\tau}^{k} > 0$ for $\forall \tau \in [t, t + L_i^k(y_{i,t}^k) - 1]$)

(iii) If the number of type k check slots is sufficient, the aircraft that has lowest remaining utilization $\tilde{R}_{1,t}^{k} = \min\{R_{i,t}^{k}\}$ has highest priority of type k check.

(iv) The A-check of aircraft i can be incorporated into C-check if $\delta_{i,t}^{C} = 1$ and $L_i^A(y_{i,t}^A) < L_i^C(y_{i,t}^C)$.

(v) If aircraft i has a higher type k check priority than aircraft j ($\tilde{R}_{i,t}^{k} < \tilde{R}_{j,t}^{k}$) but the remaining slots of type k check are only sufficient to accommodate a type k check for aircraft j rather than i, swap the priorities of type k check between aircraft i and j.

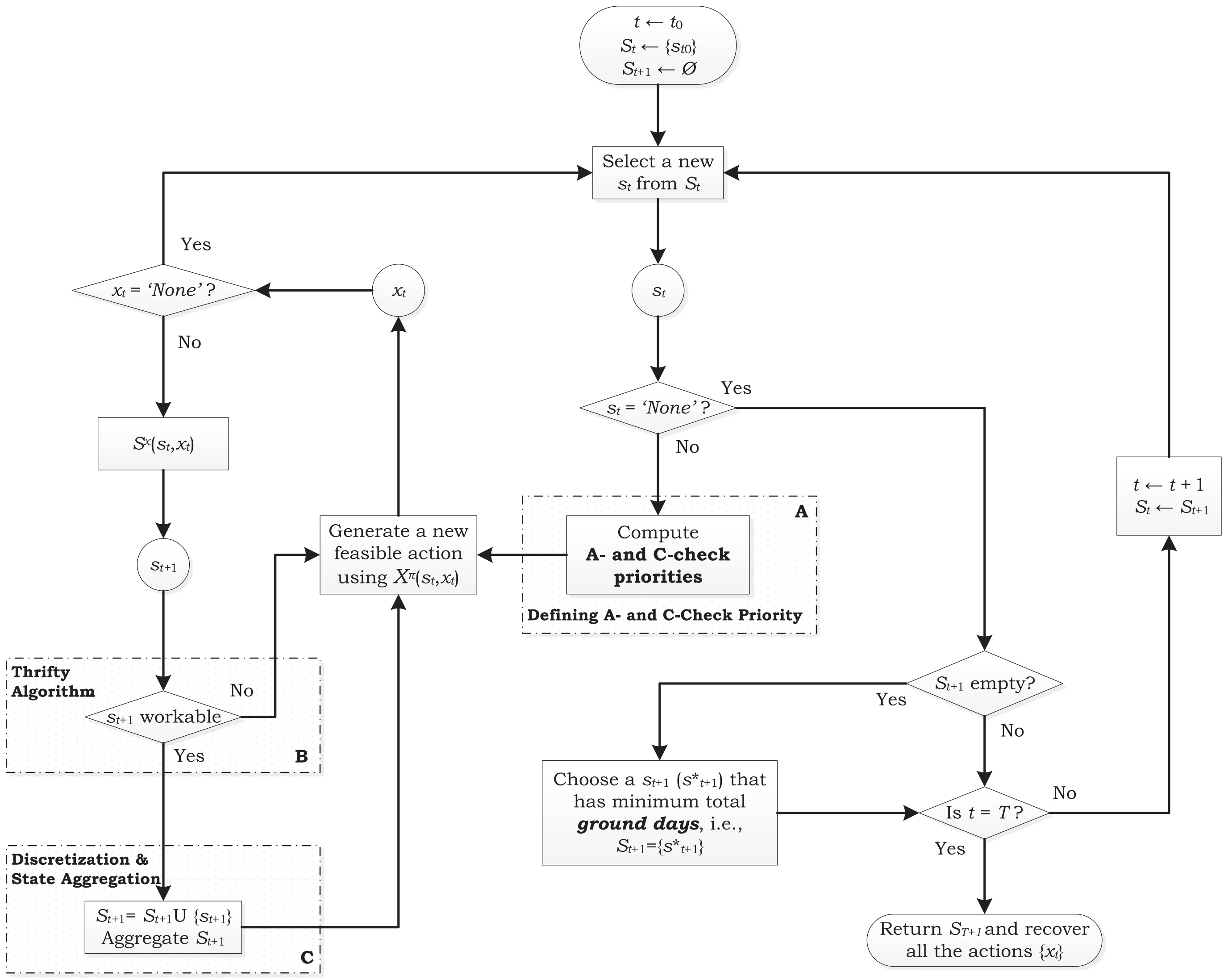

$t \leftarrow t_0$
$S_t \leftarrow \{s_{t0}\}$
$S_{t+1} \leftarrow \emptyset$
Select a new s_t from S_t
s_t
s_t = 'None'?
Yes
No
Compute A- and C-check priorities
A
Defining A- and C-Check Priority
Generate a new feasible action using $X^\pi(s_t,x_t)$
x_t
x_t = 'None'?
Yes
No
$S^x(s_t,x_t)$
s_{t+1}
Thrifty Algorithm
s_{t+1} workable
No
Yes
B
Discretization & State Aggregation
S_{t+1}= S_{t+1}U $\{s_{t+1}\}$
Aggregate S_{t+1}
C
S_{t+1} empty?
Yes
No
Choose a s_{t+1} (s^*_{t+1}) that has minimum total ground days, i.e., S_{t+1}=$\{s^*_{t+1}\}$
Is $t = T$?
No
Yes
$t \leftarrow t+1$
$S_t \leftarrow S_{t+1}$
Return S_{T+1} and recover all the actions $\{x_t\}$

Fig. 3. Work flow of the proposed DP based methodology. Three main components are incorporated, labeled as blocks A, B and C. In block A, we *define the A- and C-check priority* for each aircraft in order to reduce the number of actions. In block B we use a *Thrifty Algorithm* to infer future implications of an action and in block C we use *discretization and state aggregation* to overcome the challenge of multi-dimensional state vectors. The term *ground days* refers to the days when an aircraft is grounded and waiting for a A-/C-check slot. After the forward induction, a final state S_{t+1} is returned and a sequence of actions x_{t_0}, x_{t_0+1}, ..., x_T is generated.

After assigning A- and C-check priorities to each aircraft, the combination of aircraft selection for maintenance, as well as the number of outcome states, is reduced from $\frac{(N!)^2}{m_C!(N-m_C)!m_A!(N-m_A)!}$ to 1.

4.3. Thrifty Algorithm for A- and C-check scheduling

Even after reducing the size of outcome space to one action per state, the number of final states n_{act}^{T+1} is very tremendous for a large T, we need to further trim the outcome space so that forward induction is tractable.

After an action is performed, we place ourselves at state s_{t+1}, where $s_{t+1} = \mathcal{S}^{\mathcal{X}}(s_t, x_t)$. Many s_{t+1} states may have the status that some aircraft will have to be grounded to wait for a A-/C-check slot in a future stage τ ($\tau > t+1$). Apparently this is what airlines to avoid (unless they have no better option), due to the very high cost of parking aircraft on the ground. Therefore, we propose to only consider the actions that lead to a *workable* state. We describe *workability* of s_{t+1} by using the following function:

$$g(s_{t+1}) = \sum_{k\in\{A,C\}} \sum_{i=1}^{N} \sum_{\tau=t+1}^{T} \eta_{i,\tau}^{k} \tag{41}$$

A state s_{t+1}, resulting from being at state s_t and taking action x_t, is said to be *workable* if there exists a sequence of actions x_{t+1}, ..., x_T such that no aircraft has to wait on the ground for an A- or C-check between $t+1$ and T. That is:

$$g(s_{t+1}) = 0 \tag{42}$$

In our approach, we use a *Thrifty Algorithm* to check the *workability* of future states (Block B in Fig. 3). That is, for each possible action x_t and resulting state s_{t+1}, we use an algorithm to check if a sequence of actions exist that guarantee (42). If that is the case, we consider s_{t+1} to be *workable* and otherwise we assume that x_t does not lead to such a *workable* state.

The term *Thrifty* refers to the concept that we make use of all available slots, allocating A- and C-check to aircraft whenever there is a maintenance opportunity. The *Thrifty Algorithm* serves the purpose of checking the workability of a state s_t. For convenience, in the rest of the paper, we refer to running the *Thrifty Algorithm* to check whether (42) holds when mentioning "checking *workability*".

4.4. Discretization and state aggregation

After moving one stage ahead in time for a set of *workable* states s_t, several *workable* s_{t+1} states are produced from a combination of s_t and x_t. We use S_{t+1} to denote the set of *workable* s_{t+1}:

$$S_{t+1} = \left\{ s_{t+1} \middle| s_{t+1} \text{ workable} \right\} \tag{43}$$

Although the *Thrifty Algorithm* can help reduce the outcome space to a certain extent by only keeping the *workable* s_{t+1}, the number of *workable* s_{t+1} is still not bounded; meaning that the number of *workable* states may still grow exponentially. This increases the difficulty of saving all *workable* s_{t+1} and tracing the actions backwards, especially if we move several stages ahead. In order to prevent the explosion of *workable* states, we need to restrain the number of *workable* s_{t+1}, from the first stage t_0 to final stage T. That is, giving an upper bound to the number of *workable* s_{t+1} so that it will not increase exponentially with increasing t. For such purpose we resort to *discretization and state aggregation*.

Discretization is the process of transferring continuous models or variables into discrete counterparts. *State aggregation* refers to collecting or clustering the states that have the same properties into a group. Here we use 'properties' to differentiate state attributes, which are the features that at a fleet level, such as mean utilization of fleet (A- or C-check) or standard deviation of fleet utilization (A- or C-check). We divide the outcome space (a set of *workable* s_{t+1}) into several disjunct space regions, where each space region is characterized by a unique tuple of values of some state properties. For the states that clustered in the same space region (having the same tuple of state properties), only one single state will be selected to represent such a space region and considered in forward induction for the next stage.

Such *discretization and state aggregation* provides an upper bound to the outcome space, since the number of *workable* s_{t+1} is determined by the number of space regions. One way of collecting *workable* states is to discretize the AMCS problem according to the mean utilization of the fleet with respect to A- and C-checks, and then categorize the *workable* states according to the values of the features:

$$\overline{u}^{\text{A}}_{t+1} = \frac{1}{N}\sum_{i=1}^{N} u^{\text{A}}_{i,t+1}, \quad \overline{u}^{\text{C}}_{t+1} = \frac{1}{N}\sum_{i=1}^{N} u^{\text{C}}_{i,t+1} \tag{44}$$

where $u^k_{i,t+1}$ is the utilization of aircraft i with respect to check type k ($k \in \{\text{A, C}\}$). We define individual utilization $u^k_{i,t}$ as the maximum of the ratios between the current value of the usage parameters and their respective maximum values, according to the MPD:

$$u^k_{i,t+1} = \max\left\{ \frac{\text{DY}^k_{i,t+1}}{I^i_{k\text{-DY}} - \epsilon^{k\text{-DY}}_{i,t+1}}, \frac{\text{FH}^k_{i,t+1}}{I^i_{k\text{-FH}} - \epsilon^{k\text{-FH}}_{i,t+1}}, \frac{\text{FC}^k_{i,t+1}}{I^i_{k\text{-FC}} - \epsilon^{k\text{-FC}}_{i,t+1}} \right\} \tag{45}$$

for $k \in \{\text{A, C}\}$. For each check type k, we also give upper bound $U^k_{\max}$ and lower bound $U^k_{\min}$ to restrict the outcome space region to be discretized. This significantly improves algorithm efficiency and reduces required computer memory when optimizing A- and C-check schedules for a large fleet. For instance, if a fleet has about 200 aircraft, performing A- or C-check on an aircraft will only impact the overall fleet utilization slightly. In such case, $U^k_{\max}$ and $U^k_{\min}$ can be chosen close to $\bar{u}^k_{t_0}$.

Since tolerance is not allowed in planning unless no feasible schedule can be found, the mean utilization of a fleet normally ranges between 0 and 1 in practice ($U^k_{\max} = 1$ and $U^k_{\min} = 0$), a discretization increment $\Delta u = 0.1$ yields 11^2 space regions in 1 stage, while an increment of $\Delta u = 0.01$ increases the number of space regions in 1 stage increase to 101^2.

Using $\overline{u}^{\text{A}}_{t+1}$ and $\overline{u}^{\text{C}}_{t+1}$ to categorize the set of *workable* s_{t+1} enables us to cover the state properties of both the A- and C-check, with each pair $(\overline{u}^{\text{A}}_{t+1}, \overline{u}^{\text{C}}_{t+1})$ corresponding to one outcome space region. For each workable state s_{t+1}, we compute the mean fleet utilization with respect to the A- and C-check, $\bar{u}^{\text{A}}_{t+1}$ and $\bar{u}^{\text{C}}_{t+1}$, from (44) and (45). These two features will be further rounded according to the number of decimal points chosen from Δu. For example, if $\Delta u = 0.1$, $\bar{u}^{\text{C}}_{t+1} = 0.345$ and $\bar{u}^{\text{A}}_{t+1} = 0.678$, then $\bar{u}^{\text{C}}_{t+1}$ can be rounded to 0.3 and $\bar{u}^{\text{A}}_{t+1}$ to 0.7. After that, we compute the cumulative contribution from state s_{t_0} to a specific *workable* state s_{t+1}:

$$J_{t+1,\bar{u}^{\text{A}}_{t+1},\bar{u}^{\text{C}}_{t+1}}(s_{t+1}) = \begin{cases} J_{t,\bar{u}^{\text{A}}_t,\bar{u}^{\text{C}}_t}(s_t) + C_t(s_t, x_t) & t > t_0 \\ C_{t_0}(s_{t_0}, x_{t_0}) & t = t_0 \end{cases} \tag{46}$$

where $s_{t+1} = \mathcal{S}^{\mathcal{X}}(s_t, x_t)$, $C_\tau(s_\tau, x_\tau)$ refers to the contribution function in (28). If a given space region has no state within it, a cumulative contribution value of infinity ∞ is assumed for that space region.

During forward induction, there will be several workable states that can be grouped into the same outcome space region because of identical $\bar{u}^{\text{A}}_{t+1}$ and $\bar{u}^{\text{C}}_{t+1}$ after rounding. Then, an aggregation procedure is followed: the state with the lowest cumulative contribution is selected as the representative of its outcome space region, while all others are discarded:

$$s^*_{t+1,\bar{u}^{\text{A}}_{t+1},\bar{u}^{\text{C}}_{t+1}} = \underset{s}{\text{argmin}} \left\{ J_{t+1,\bar{u}^{\text{A}}_{t+1},\bar{u}^{\text{C}}_{t+1}}(s) \right\} \tag{47}$$

In the worst case, no s_t has a subsequent workable s_{t+1}, that is, $g(s_{t+1}) = g(\mathcal{S}^{\mathcal{X}}(s_t, x_t)) > 0$ for all s_t and $x_t = \mathcal{X}^\pi(s_t)$. In such a circumstance, we select only one $\hat{s}_{t+1}$ according to:

$$\hat{s}_{t+1} = \underset{s,\bar{u}^{\text{A}}_t,\bar{u}^{\text{C}}_t}{\text{argmin}} \left\{ J_{t,\bar{u}^{\text{A}}_t,\bar{u}^{\text{C}}_t}(s) \right\} \tag{48}$$

$$S_{t+1} = \left\{ \hat{s}_{t+1} \right\} \tag{49}$$

where the right hand side of (48) means choosing the state s among all outcome space regions $(\overline{u}^{\text{A}}_{t+1}, \overline{u}^{\text{C}}_{t+1})$. The forward induction then continues from $\hat{s}_{t+1}$.

The procedure from (46) to (47) repeats until it loops all possible pairs of $\{s_t, x_t\}$ (s_t *workable*). Thus far, we complete the *discretization and state aggregation*, and then the forward induction moves one stage ahead from t to $t+1$. The pseudo code of DP based methodology is presented in Appendix B.

4.5. Algorithm complexity

From the perspective of algorithm complexity, the total number of states in our DP based methodology is equivalent to the total number of outcomes, given by (34):

$$\prod_{t=t_0}^{T} \sum_{m_{\text{c}}=0}^{M^{\text{C}}_t} \sum_{m_{\text{a}}=0}^{M^{\text{A}}_t} \frac{(N!)^2}{m_{\text{a}}!\ m_{\text{c}}!\ (N-m_{\text{a}})!\ (N-m_{\text{c}})!} \tag{50}$$

where N is the total number of aircraft, and M^{C}_t and M^{A}_t are the maintenance capacity of C-check and A-check respectively.

Given a state s_t at time stage t, following an action x_t, the algorithm has to call the state transition function (6) at most $T - t_0 - t + 1$ times to check whether (42) holds (from t to T). In each stage t, the number of states depends on the discretization resolution Δu:

$$n_{\text{state}} = \left(1 + \frac{1}{\Delta u}\right)^2 \tag{51}$$

Since each state can have at most n_{act} actions, this implies the following relation between the stage t and the number of state transition:

Day t_0:	$n_{\text{act}}(1+T-t_0)$
Day t_0+1:	$n_{\text{state}}n_{\text{act}}(1+T-1-t_0)$
...	...
Day $T-1$	$n_{\text{state}}n_{\text{act}}(1+T-T)$
Day T	$n_{\text{state}}n_{\text{act}}(1+T-T-1)$

After summing up of all state transitions from t_0 to T we obtain

$$\begin{aligned} & n_{\text{act}}(1+T-t_0) + n_{\text{state}}n_{\text{act}}\sum_{\tau=0}^{T-t_0}\tau \\ &= n_{\text{act}}(1+T-t_0)\left[1+\frac{n_{\text{state}}(T-t_0)}{2}\right] \\ &= n_{\text{act}}(1+T-t_0)\left[1+\frac{1}{2}\left(1+\frac{1}{\Delta u}\right)^2(T-t_0)\right] \end{aligned} \tag{52}$$

Computing (52) gives the maximum number of state transitions in forward induction. Since each of the state transitions generates a new state, this means that the total number of states to be visited is equal to the total number of state transitions in forward induction.

Moreover, (52) also indicates that the total number of states visited during forward induction depends on the A- and C-check capacity (n_{act} is determined by the A- and C-check capacity M_t^{A} and M_t^{C}), and the increment of discretization Δu and planning horizon $T-t_0$. Since (52) increases quadratically with T, this means that (52) can be much smaller than (50) for a large T:

$$\begin{aligned} & n_{\text{act}}(1+T-t_0)\left[1+\frac{1}{2}\left(1+\frac{1}{\Delta u}\right)^2(T-t_0)\right] \\ & << \prod_{t=t_0}^{T}\sum_{m_c=0}^{M_t^{\text{C}}}\sum_{m_a=0}^{M_t^{\text{A}}}\frac{(N!)^2}{m_a!\ m_c!\ (N-m_a)!\ (N-m_c)!} \end{aligned} \tag{53}$$

5. Case study

In this section, the proposed DP based methodology is evaluated using the aircraft maintenance data from a European airline. Two case studies are presented in this evaluation. The first case uses data from the historical period 2013–2016 to validate the proposed DP based methodology. We compare the results obtained by the DP based methodology with the A- and C-check schedule executed by the airline. A detailed scheduled from the airline is used and the comparison is easy to make. However, this comparison is somewhat unfair since the airline in the executed schedule had to take aircraft routing into account and potentially deal with unscheduled maintenance events. Therefore, the second case focuses on the period of 2018–2021 and compares the results from the DP based methodology with the maintenance schedule planned by the airline. This case is also used to support a sensitivity analysis on some of the model parameters. The data set supporting case study is available on https://doi.org/10.4121/uuid:1630e6fd-9574-46e8-899e-83037c17bcef.

5.1. Test cases

The test fleet is the Airbus A320 family (A319, A320 and A321) operated by the airline, consisting of 45 aircraft. These three aircraft types happen to share the same A- and C-check intervals and tolerances, in terms of same flight hours, flight cycles and calendar days (Table 2). The planning horizon is 4 years in both cases. For 2013–2016, this starts from January 1st of 2013 to December 31st of 2016, while 2018–2021 the planning horizon goes from September 25th of 2017 to December 31st of 2021. For both cases the initial data contains the information for aircraft average monthly utilization; initial fleet status, in terms of DY, FH and FC, and utilization of tolerance in previous inspections; maintenance slots available per day; and average elapsed time of the multiple A- and C-checks labels. The average daily utilization of the aircraft is computed per month and per aircraft type, according to the historic flight data from the airline. On average, it is estimated that the A320 family fleet has a daily utilization of 10.5 FH and 4.7 FC per day.

Table 2
A- and C-check intervals and tolerance for the Airbus A319, A320 and A321 (AIRBUS, 2017).

	Check type	Calendar days	Flight hours	Flight cycles
Inspection intervals	A-Check	120	750	750
	C-Check	730	7500	5000
Tolerance	A-Check	12	75	75
	C-Check	60	500	250

5.2. Maintenance constraints and key performance indicators

The maintenance schedule needs to follow a set of operational and capacity constraints, namely, for the A-check:

- there is 1 A-check slot per day from Monday to Thursday during IATA winter (from the last Sunday of October to the last Sunday of March);
- during IATA Summer (from the last Sunday of March to last Sunday of October), there is an extra A-check slot on Tuesdays (2 slots on Tuesday);
- from 2018 onwards, there are 2 A-check slots on Tuesdays (all year) and 2 A-check slot on Wednesdays during IATA Summer;
- there are no A-checks on Fridays, weekends, or public holidays;
- an A-check lasts 1 day and can be merged into a C-check without increasing the C-check elapsed time or affecting the existing available A-check slots.

For the C-check:

- there can be a maximum of 3 C-checks ongoing in parallel;
- there are a minimum of 3 days between the start dates of two C-checks, for resource availability reasons (i.e., $d_{\text{C}} = 3$);
- C-check works are interrupted during weekends and public holidays;
- no C-check can be scheduled during the commercial peak seasons (except some extraordinary occasions in which the airline is forced to have additional slots to avoid aircraft waiting on the ground for a C-check).

Since there are at least 3 days between two start dates of two successive C-checks, there could be at most 1 C-check starting on a day. The maximum of 2 A-checks on a Tuesday and considering the possibility of merging A-checks into C-check, leads to the combination of daily A- and C-check capacities with 7 possible actions on a day:

(i) 0 A-check and C-check
(ii) 0 A-check and 1 C-check
(iii) 1 A-check and 0 C-check
(iv) 1 A-check and 1 C-check
(v) 2 A-checks and 0 C-check
(vi) 2 A-checks and 1 C-check
(vii) 3 A-checks and 1 C-check

The commercial peak seasons of the airline are defined to be between June 1st and September 30th, two weeks before the New Year's and one week after, and the weeks before and after Easter. The days of the year are converted into calendar days where, e.g., the New Year's Day is set as day 1 and Christmas is day 359 of the year (or day 360 if it is a leap year).

Table 3
Descriptive statistics of KPIs for 2013–2016 ($\Delta u = 0.1$ in the DP based method). For the term "Tolerance Events", if an aircraft uses tolerance (DY/FH/FC) in planning, it is counted as 1 tolerance event.

Type	KPI 2013–2016	Airline	DP-based Method	Difference
C-Check	Average FH	6795.9	6798.7	0.04%
	Standard Deviation	1013.7	572.2	−43.6%
	Total FH Tolerance	2230	349.2	−84.3%
	Tolerance Events	17	1	−94%
	Extra C-Check Slot	73	0	−100%
	Total C-Check	89	82	−7.9%
A-Check	Average FH	690.8	701.2	1.5%
	Standard Deviation	65.6	31.6	−51.8%
	Total FH Tolerance	1277	457.4	−64.2%
	Tolerance Events	72	34	−52.8%
	Extra A-Check Slots	101	0	−100%
	Total A-Check	818	758	−7.3%

To discuss the results, we use a set of key performance indicators (KPIs) for each type of maintenance check. These are the average FH of the fleet, the total number of checks, the total amount DY/FH/FC used as tolerance and computation time etc. during the planning horizon. For deterministic problems, we make the transition probability $p\left(s_{t+1}\middle|s_t\right) = 1$ in (31). $\bar{R}_A$ and $\bar{R}_C$ are set to 21 and 365, meaning that an aircraft can only be scheduled an A-/C-check if the corresponding remaining operation days is lower than 21/365 days. Since no information is given for discount factor γ, we set $\gamma = 1$ and the penalty of using tolerance P_a and P_d in (28) are given to be 10^8. This avoids using tolerance in forward induction and grounding the aircraft in the situation of no A- or C-check slot.

The airline schedules the aircraft A- and C-checks separately. The C-checks are scheduled first with a time horizon of 4-years, followed by scheduling of the A-checks for the next year. In both cases, the airline follows a greedy approach with the goal to schedule the checks as close as possible to the end of their intervals. The common conflicts resulting from this approach are them manually solved by the maintenance planner, which anticipates the dates of the checks until a feasible plan is obtained. This manual process is a puzzle, hard to solve for the A-checks and for the C-checks close to the peak seasons during which no checks can be scheduled. This results in a sub-optimal schedule that takes a couple of days of work to be fully developed from scratch.

In addition, if an aircraft uses tolerance before undergoing an A-/C-check, the more of extra DY/FH/FC used in tolerance has to be subtracted from the next A-/C-check interval, namely, the interval to its next A-/C-check becomes shorter. For instance, the A320 family has an A-check interval of 750 FH (see Table 2), if an aircraft has to fly 770 FH before undergoing an A-check, then the amount of tolerance used is $770 - 750 = 20$ FH, and the next A-check interval will be $750 - 20 = 730$ FH (this rule has already been considered in the problem formulation (22)–(23)).

5.3. Optimization results for 2013–2016

The proposed algorithm is first evaluated for the planning horizon of 2013–2016. Table 3 shows a comparison of KPIs between the airline schedule and the DP schedule. We observe that the average FH increases with respect to both A- and C-checks. For the A-check, there is a growth of 10.4 FH on average per aircraft, which equates to approximate an extra day in operation per aircraft per A-check cycle. This increase has an impact on the number of checks needed for the 4-year period. There is a reduction of more than 7% for both A- and C-checks, which is equivalent to 60 fewer A-checks and 7 fewer C-checks. From the perspective of maintenance cost, assuming that airlines spend on average \$70K–\$350K (Ackert, 2010) on a C-check and \$10K—\$15K on a A-check, the results from the proposed DP-based methodology can result in a maintenance costs saving of approximate \$1.1M—\$3.4M for the fleet of A320 family.

Since it takes 10–30 days to complete a C-check and 1 day for a A-check, 7 reduced C-checks and 60 reduced A-checks are equivalent to approximately 130–270 more days of aircraft availability for revenue generation. One day of operation generates \$75K–\$120K of revenue and 130–270 more days available for commercial use means an additional \$9.8M–\$32.4M of revenue for an airline.

The optimized schedule uses tolerances of 349.2 FH and 457.4 for the A- and C-check scheduling, respectively. These are 84% and 64% less than the FH tolerances used by airline. More importantly, the optimized schedule reduces the frequency of using tolerance (if an aircraft uses tolerance, it is counted as 1 tolerance events), from 17 to 1 with respect to C-check, and from 72 to 34 for A-check. Recall that using tolerance needs to be approved by the national aviation authority and it is a troublesome process that should not be used recurrently.

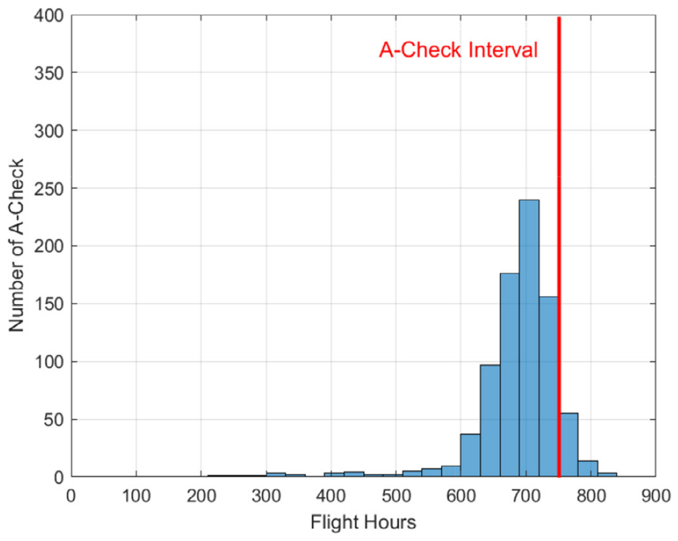
In Figs. 4 and 5, we observe that the optimized schedule generated by the DP-based methodology concentrates the aircraft FH of A- and C-check close to its corresponding inspection interval. For the A-check, 17% of the checks are scheduled with 95% of the interval used, while for the airline this value was double, up to 34%. A similar result is obtained for the C-check, where these values are 23% and 43%, for the optimized schedule and the airline schedule, respectively. As a result of the greedy approach followed by the airline, the airline has a large number of A- and C-checks scheduled very closed to their interval limit. However, this is achieved by using tolerance in 9% of the A-checks and 19% of the C-checks; by scheduling other checks with a quite low interval utilization; and by creating A- and C-check slots, not considered in the optimized schedule, to solve occasional critical situations with several aircraft with high utilization. It is important to notice that the optimized results only used tolerance in the checks at the beginning of the time horizon. It was not possible to schedule these checks without using tolerance, given the initial state of the fleet and the maintenance slots available.

5.4. Optimization results for 2018–2021

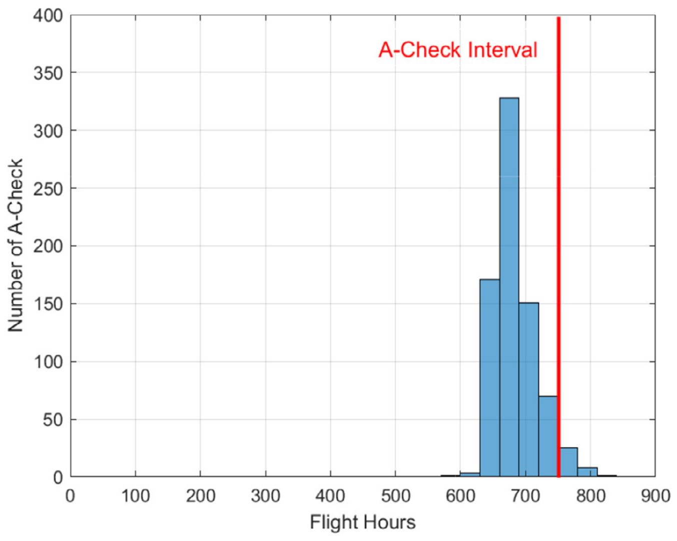
Although the proposed DP based methodology appears to outperform the planning approach of the airline, based on the KPIs showed in Table 3, this comparison is somewhat unfair since the airline has to take aircraft routing into account and deal with all kinds of unscheduled maintenance events. In order to verify and validate the proposed DP based methodology together with the maintenance planners from the airline, we use it to subsequently generate an optimized A- and C-check schedule for future 2018–2021, and then compare this schedule with the one made by the maintenance planners of the airline.

In this test case, both the maintenance planners of airline and us plan the 4-year maintenance check schedule using exactly the same input, average aircraft daily utilization, operational constraints and excluding unscheduled maintenance events and aircraft routing. We compare the KPIs with respect to C-check from both schedules, as well as the optimization results of different discretization resolutions ($\Delta u = 1,\ 0.1$ and 0.01). Given that the airline only plans the A-check for the coming year, no A-check metrics were compared. The optimized schedules of different discretization resolutions (both A- and C-checks use the same level Δu in discretization) are obtained using parallel computing function on a quad-core workstation.

Again we see that the proposed DP based methodology outperforms the planning approach of airline, in terms of KPIs and computation time. The optimized schedules reduce the number of C-

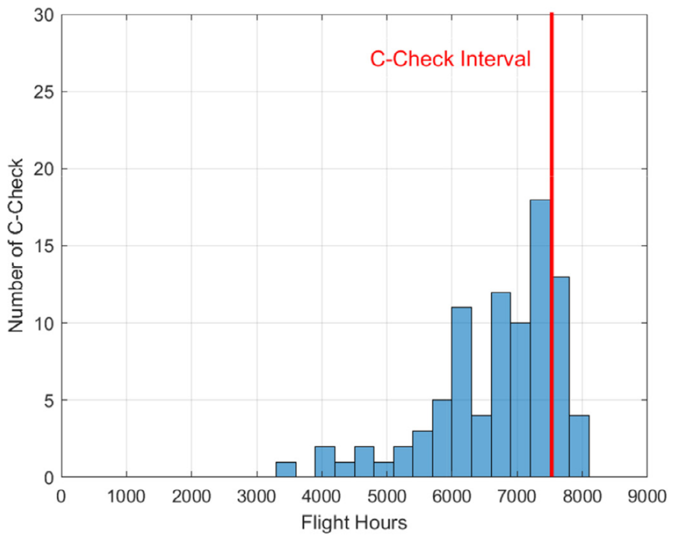


(a) Airline Schedule

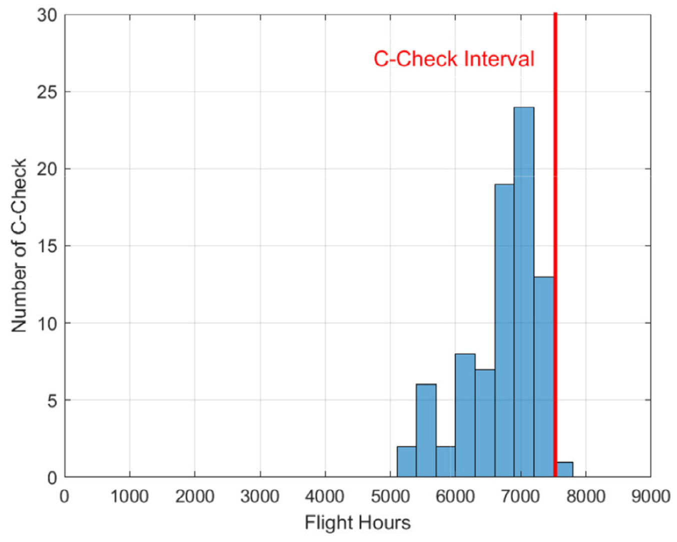


(b) Optimized Schedule

Fig. 4. Comparison of aircraft FH with respect to A-check between schedule of airline and the optimized schedule.



(a) Airline Schedule



(b) Optimized Schedule

Fig. 5. Comparison of aircraft FH with respect to C-check between schedule of airline and the optimized schedule.

checks, varying from 8.3% (for $\Delta u = 0.1$) to 11.4% (for $\Delta u = 0.01$), while the same amount of tolerance is used in all three optimized schedules. The tolerance and the number of tolerance events from our results is significantly less than what the airline scheduled. The use of this tolerance is inevitable for aircraft that at the starting date of the optimization are already closed to their C-check interval. The number of A-checks vary from 877 (for $\Delta u = 0.1$) and 881 (for $\Delta u = 1$ and $\Delta u = 0.01$), when the airline estimates around 895 to 920 A-checks for these four years. The number of A-checks merged in the C-checks has little variance among three discretization resolutions.

Besides, an overall trend is found where the level of discretization impacts the solution quality and algorithm computation time, as illustrated in Fig. 6(a) and (b). In terms of optimality, as expected, the smaller Δu is, the better the results are. However, the KPI's presented in Table 4 are no significantly different between the three Δu values tested, indicating that good results can be obtained even with low discretization resolution. This happens be-

Table 4
Results of A- and C-check scheduling optimization from different discretization resolution, compared with the C-check schedule from Airline.

KPI (2018–2021)	Airline	$\Delta u = 1$	$\Delta u = 0.1$	$\Delta u = 0.01$
Objective Value [FH]	–	1.2140×10^5	1.1524×10^5	1.1371×10^5
C-Check Avg. FH	< 6600	6558.1	6615.2	6634.7
Total C-Checks	96	86	88	85
C-Check Tol. DY	> 48	18	18	18
C-Check Tol. FH	> 490	135.3	135.3	135.3
C-Check Tol. FC	0	0	0	0
Tolerance Events	6	4	4	4
A-Check Avg. FH	–	714.3	717.6	714.5
Total A-Checks	895–920*	881	877	881
Tolerance Events	–	0	0	0
Merged A- and C-Check	–	19	18	19
Computation Time [s]	≥ 3 Days	504.9	510.3	20243.5

* Airline estimation.

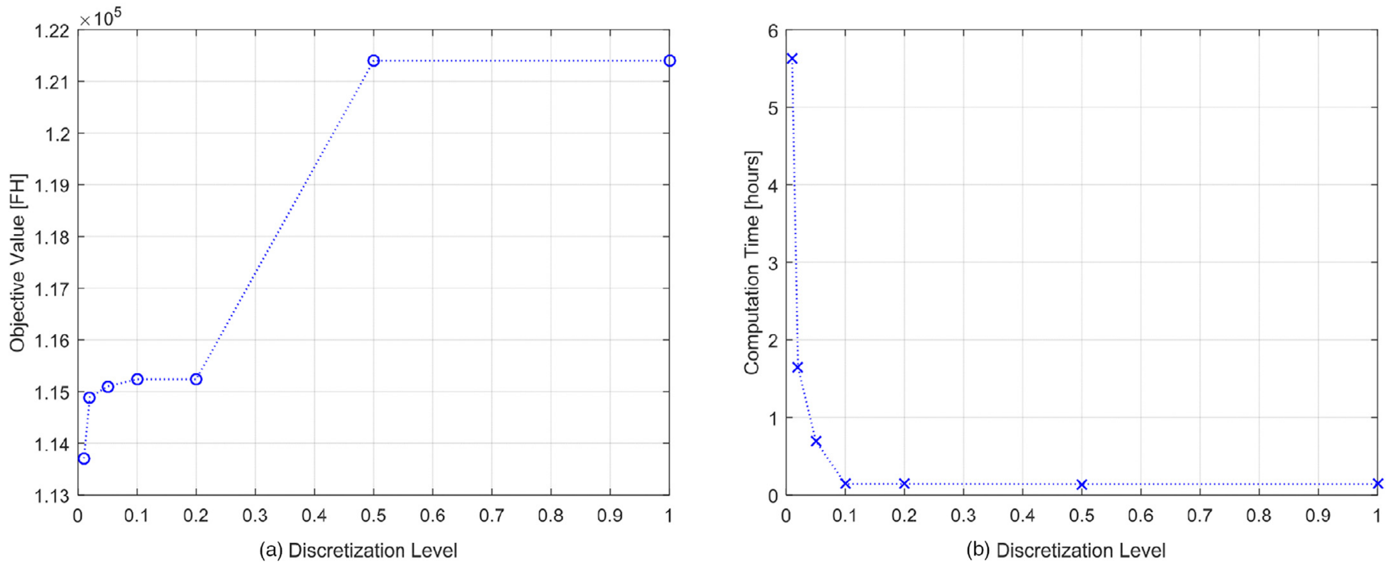


Fig. 6. (a) Correlation between discretization level (resolution) and objective value; (b) correlation between discretization level (resolution) and algorithm computation time.

cause most *workable* states are in a limited range of the outcome space. Just a few of the space regions from our discretization do really have a *workable* state after aggregation. Nevertheless, there is trade-off between optimality and computation efficiency. The objective function consistently reduces when we increase the discretization resolution. However, the computational times sharply increase when we increase the discretization resolution beyond $\Delta u = 0.1$.

For the algorithm computation time, decreasing Δu results in longer computation times. This happens because the number of outcome space region increases with decreasing Δu, e.g., there are 121 outcome space regions when $\Delta u = 0.1$, and 10,201 regions for $\Delta u = 0.01$. As a result, the number of representative states of outcome space region in each stage also increases. However, for this particular case study, the computational time only starts to sharply increase at Δu values of lower than 0.1. After breaking down the computation time, we observe that each state transition requires about 0.023 seconds. Looking at the case of $\Delta u = 0.1$, it takes 510.3 seconds to obtain an optimized schedule using parallel computing on a quad-core workstation, meaning that the actual computation time should be about 2141.2 seconds. In particular, during 2141.2 seconds, there are about $2141.2/0.023 = 9.31 \times 10^4$ state transitions for 1461 stages (there being 1461 days from Jan 1st 2018 to Dec 31st 2021). The actual computation is still much less than the worst case computed from (52):

$$\begin{aligned}&\text{computation time}\\&= 0.023 \times n_{\text{act}}(1+T-t_0)\left[1+\frac{1}{2}\left(1+\frac{1}{\Delta u}\right)^2(T-t_0)\right]\\&= 0.023 \times 7 \times 1461 \times \left[1+\frac{1}{2}\times 11^2 \times 1460\right] \approx 2.08 \times 10^7 \ (s)\end{aligned} \tag{54}$$

The shorter actual computation time than the worst case is due to the checking of *workability* and *state aggregation*, where we only keep some *workable* states in each stage, which in the end are sufficient to generate an optimized schedule.

5.5. Sensitivity analysis for 2018–2021

This subsection investigates the impact of some airline capacity constraints on the results of the AMCS problem, relative to the following 4 scenarios:

- *Scenario 0*: the baseline scenario, as pre-computed in the previous subsection;
- *Scenario 1*: conditions from Scenario 0 and one additional A-check slot on Friday, weekends and bank holidays (i.e., one A-check slot every day of the week, plus an extra slot on Tuesdays and Wednesdays during IATA Summer);
- *Scenario 2*: conditions from Scenario 0 and three additional C-checks on weekends and bank holidays (i.e., three C-check slots every day of the week during off peak seasons, reducing the elapse time of the C-checks);
- *Scenario 3*: conditions from Scenario 0, Scenario 1, and Scenario 2 combined.

All four scenarios are computed assuming $\Delta u = 0.1$, relative to the sensitivity shown in Table 4 regarding precision and computation time. The results, shown in Table 5, indicate that a natural improvement of the aircraft average utilization is obtained when either the A-check or the C-check slots are increased. By increasing the number of checks, we are increasing the maintenance opportunities, given more flexibility for a schedule in which the checks are planned closer to their due date. For example, compared with *Scenario 0*, the objective value in *Scenario 3* is reduced by 33%, and the average FH of aircraft is increased by 2.7% and 3.1% for C-check and A-check, respectively. As consequence of this, there are 2 fewer C-checks and 14 fewer A-checks scheduled for *Scenario 3* when compared with *Scenario 0*. For the scenarios involving more C-check slots, it is observed that the results are improved by the fact that more maintenance opportunities exist to merge A-checks and C-checks. In fact, *Scenarios 2* and *3* have around 500 to 600 fewer days on the ground than *Scenarios 1* and *2*, respectively. These are days when the aircraft can be used in operation to generate revenue. Furthermore, it is interesting to notice the interdependence between A- and C-checks when analyzing the results from *Scenario 1*. Although only additional A-check slots are added, the results for the C-checks also improve, due to the fact that more A-check slots create more A-check maintenance opportunities. This gives more flexibility to schedule some of the C-checks that in *Scenario 0* were anticipated to enable the merge with an A-check.

The consideration of extra aircraft maintenance capacity has to be analyzed by the airline by comparing the additional costs of these extra slots and the benefits of having less maintenance checks and higher aircraft availability. This analysis is outside the

Table 5
Sensitivity analysis for having different A- and C-check slots in 2018–2021, the discretization resolution is set to $\Delta u = 0.1$. No A-check tolerance was used in the scenarios tested.

KPI (2018–2021) Objective Value [FH]	Scenario 0 1.1524×10^5	Scenario 1 0.8934×10^5	Scenario 2 1.0623×10^5	Scenario 3 0.7719×10^5
C-Check Avg. FH	6615.2	6635.0	6699.3	6790.5
Total C-Checks	88	85	85	86
C-Check Tol. DY	18	18	0	0
C-Check Tol. FH	135.3	135.3	23.6	23.6
C-Check Tol. FC	0	0	0	0
A-Check Avg. FH	717.6	738.5	716.5	739.8
Total A-Checks	877	856	890	863
Merge A- and C-check	18	7	22	10
Computation Time [seconds]	510.3	1780.2	743.0	2625.5

scope of this paper but these results are crucial to the airline in assessing such capacity increases (or reductions) scenarios.

6. Conclusion

A practical dynamic programming based methodology for the long-term aircraft maintenance check scheduling (AMCS) problem is presented. This integrates both A- and C-checks, including requirements that are associated with the previous denominated B- and D-checks, operational constraints and maintenance capacity for specific days. The goal was to minimize the total wasted FH interval between checks, hereby increasing aircraft availability in the long run.

The proposed methodology followed a *forward induction* approach, incorporating a maintenance *priority solution* to deal with the multi-dimensional action vector, as well as a *discretization and state aggregation* strategy to reduce outcome space at each time stage. In addition, a *Thrifty Algorithm* was used to estimate the consequence of an action at the current stage on the remaining planning horizon. All these adaptations in the DP framework are novel compared with the classic dynamic programming. The proposed methodology is capable of optimizing both A- and C-check schedules in a matter of minutes for multiple years horizon and heterogeneous aircraft fleets. It is suitable for practical implementation and it can be used not only for scheduling but also, for example, to predict if an airline has sufficient maintenance capacity in the future; or to assess if it is beneficial to expand maintenance capacity with additional hangar slots.

The proposed DP based methodology is evaluated using the case-study of an A320 family fleet from a European airline. Comparing the optimized A- and C-check schedules with the schedule of the airline, we can infer that the proposed methodology reduces the total number of A- and C-check, potentially resulting in the long run in maintenance cost savings of about \$1.1M-\$3.4M for a fleet of about 40 aircraft. Besides, the reduction of A- and C-checks implies extra days of aircraft availability for revenue generation, an estimation of \$9.8M–\$32.4M can be generated when the proposed methodology is applied on historical data.

This study is the first to address the AMCS optimization problem despite its relevance for practice, despite its relevance for practice. It opens the door for future research on the topic. For instance, future research can consider the uncertainty associated with both the maintenance check elapsed time and the aircraft utilization. These uncertainties will not only affect the schedule robustness, but also the computational time needed to find such optimal schedules. One such improvement can be achieved by using approximate dynamic programming, extending the dynamic programming principle adopted in this paper. Another research opportunity, is the consideration of the task allocation problem (i.e., the problem of defining the tasks to be performed on each aircraft check) as part of the AMCS problem. Although this would significantly increase the complexity of the problem, it would extend the AMCS problem to good benefit, producing an optimal integrated check and tasks schedule.

Acknowledgments

This research work is part of AIRMES project, which received funding from the Clean Sky 2 Joint Undertaking under the European Union's Horizon 2020 research and innovation programme under grant agreement No. 681858. We would like to show gratitude to the associated AIRMES partners for providing their aircraft maintenance data and validation of the proposed methodology and its practical relevance. For more information please visit www.airmes-project.eu. We would also like to thank the editor and reviewers for their careful reading and constructive suggestions.

Appendix A. Nomenclature

Parameters:

d_k	minimum interval between the start dates of two type k checks.
$e^i_{k\text{-DY}}$	maximum DY tolerance of type k check interval of aircraft i
$e^i_{k\text{-FH}}$	maximum FH tolerance of type k check interval of aircraft i
$e^i_{k\text{-FC}}$	maximum FC tolerance of type k check interval of aircraft i
$\text{fc}_{i,t}$	average daily FC usage for aircraft i at day t
$\text{fh}_{i,t}$	average daily FH usage for aircraft i at day t
h	hangar indicator
i	aircraft indicator
$I^i_{k\text{-DY}}$	interval of type k check of aircraft i in terms of DY
$I^i_{k\text{FH}}$	interval of type k check of aircraft i in terms of FH
$I^i_{k\text{-FC}}$	interval of type k check of aircraft i in terms of FC
k	maintenance check type, $k \in \{\text{A, C}\}$
N	total number of aircraft
n_k	the number of hangars for type k check
n_{act}	the number of actions on day t
P_d	daily penalty for having an aircraft on the ground waiting for a maintenance slot
P_a	penalty for an aircraft using the tolerance
$\overline{R}_k$	remaining day threshold of type k check
t	indicator of calendar day
T	final day in planning horizon
t_0	first day in planning horizon
Δu	increment of fleet utilization for discretization
π	scheduling policy
γ	discount factor

Main decision variables:

$\chi^k_{i,t}$	binary variable to indicate if aircraft i starts type k check on t
x_t	available action on day t, $x_t = \left\{\left\{\chi^{\text{C}}_{i,t}\right\}, \left\{\chi^{\text{A}}_{i,t}\right\}\right\}$
x^*_t	the optimal action among $\{x_t\}$
$\mathcal{X}^\pi(s_t)$	scheduling policy function, $x_t = \mathcal{X}^\pi(s_t)$

State related decision variables:

$a_{i,t}$	the attributes of aircraft i in the beginning of day t
A_t	$A_t = \{a_{i,t} \mid i = 1, 2, \ldots, N\}$
$C_t(s_t, x_t)$	contribution of choosing action x_t on s_t
$\mathrm{DY}^k_{i,t}$	total DY of aircraft i in the beginning of day t for type k check
$\mathrm{FC}^k_{i,t}$	cumulative FC of aircraft i at t since last type k check
$\mathrm{FH}^k_{i,t}$	cumulative FH of aircraft i at t for type k check
$J_{t,\bar{u}^A_t,\bar{u}^C_t}(s_t)$	cumulative contribution on day t when the fleet has mean utilization $\bar{u}^A_t$ and $\bar{u}^C_t$ for A-check and C-check respectively
$J^{\min}_{t,\bar{u}^A_t,\bar{u}^C_t}(s_t)$	$J^{\min}_{t,\bar{u}^A_t,\bar{u}^C_t}(s_t) = \min\left\{J_{t,\bar{u}^A_t,\bar{u}^C_t}(s_t)\right\}$
$L^k_i(y^k_{i,t})$	estimated elapsed time of next type k check with label $y^k_{i,t}$
$M^k_{h,t}$	binary variable to indicate if type k check can be performed in hangar h_k on day t
M^k_t	hangar capacity of type k check, $M^k_t = \sum_h M^k_{h,t}$
s_t	state variable
S_t	the set of workable states, $S_t = \left\{s_t \middle\| s_t \text{ workable}\right\}$
$R^k_{i,t}$	remaining fly days of aircraft i before the next type k check
$y^k_{i,t}$	next maintenance label for of type k check of aircraft i on day t
$z^k_{i,t}$	the end date of type k check of aircraft i
$\delta^k_{i,t}$	binary variable to indicate if aircraft i is undergoing type k check on day t
$\epsilon^{k\text{-DY}}_{i,t}$	extra DY before day t if previous type k check is deferred
$\epsilon^{k\text{-FH}}_{i,t}$	extra FH before day t if previous type k check is deferred
$\epsilon^{k\text{-FC}}_{i,t}$	extra FC before day t if previous type k check is deferred
$\eta^k_{i,t}$	binary variable to indicate if aircraft i is grounded and waiting for a type k check
$\theta^k_{i,t}$	tolerance usage indicator of type k check of aircraft i on day t
Ψ	$\Psi \in \{\mathrm{FH}, \mathrm{FC}\}$
$\Psi^k_{i,t}$	$\Psi^k_{i,t} \in \{\mathrm{FH}^k_{i,t}, \mathrm{FC}^k_{i,t}\}$
$\psi^k_{i,t}$	$\psi^k_{i,t} \in \{\mathrm{fh}^k_{i,t}, \mathrm{fc}^k_{i,t}\}$

Others:

$\mathcal{S}^X(s_t, x_t)$	transition function from s_t to s_{t+1}, $s_{t+1} = \mathcal{S}^X(s_t, x_t)$
$u^k_{i,t}$	utilization of aircraft i on day t with respect to type k check
$\bar{u}^k_t$	mean utilization of fleet on calendar day t for type k check
$V_t(s_t)$	the value of being in a state s_t

Appendix B. Algorithm

Algorithm 1 A dynamic programming based methodology for aircraft A- and C-check scheduling optimization.

```
Step 1: Initialize $\Delta u$ $(0 < \Delta u < 1)$, $t \leftarrow t_0$, $S_t \leftarrow \left\{s_{t_0}\right\}$, $S_{t+1} \leftarrow \emptyset$
Step 2: Discretize the interval [0,1] with $\Delta u$: $0, \Delta u, 2\Delta u, \ldots, 1$;
Step 3: For each workable $s_t \in S_t$:
    Step 3.1: Compute and sort the remaining utilization:
        $\tilde{R}^C_{1,t}, \ldots, \tilde{R}^C_{N,t} \quad \tilde{R}^C_{i,t} \le \tilde{R}^C_{i+1,t} \quad \tilde{R}^C_{i,t} \in \left\{R^C_{i,t}\right\}$
        $\tilde{R}^A_{1,t}, \ldots, \tilde{R}^A_{N,t} \quad \tilde{R}^A_{i,t} \le \tilde{R}^A_{i+1,t} \quad \tilde{R}^A_{i,t} \in \left\{R^A_{i,t}\right\}$
    Step 3.2: For each action $x_t$ of $s_t \in S_t$:
        Step 3.2.1: Compute $C_t(s_t, x_t)$;
        Step 3.2.2: Compute $s_{t+1}$ using $s_{t+1} = \mathcal{S}^X(s_t, x_t)$;
        Step 3.2.3: Check whether $s_{t+1}$ is a workable state;
        Step 3.2.4: Aggregate $s_{t+1}$ according to
            $J^{\min}_{t+1,\bar{u}^A_{t+1},\bar{u}^C_{t+1}}(s) = \min_{\{\bar{u}^A_{t+1},\bar{u}^C_{t+1}\}} \left\{J_{t+1,\bar{u}^A_{t+1},\bar{u}^C_{t+1}}(s_{t+1})\right\}$
            $s^*_{t+1,\bar{u}^A_{t+1},\bar{u}^C_{t+1}} = \operatorname{argmin}_{\{s,\bar{u}^A_{t+1},\bar{u}^C_{t+1}\}} \left\{J_{t+1,\bar{u}^A_{t+1},\bar{u}^C_{t+1}}(s)\right\}$
            $S_{t+1} = S_{t+1} \cup \left\{s^*_{t+1,\bar{u}^A_{t+1},\bar{u}^C_{t+1}}\right\}$
Step 4: $t \leftarrow t + 1$;
Step 5: If $t \le T$, go to Step 3;
    Else $s^*_{T+1,\bar{u}^A_{T+1},\bar{u}^C_{T+1}} = \operatorname{argmin}_s \left\{J_{T+1,\bar{u}^A_{T+1},\bar{u}^C_{T+1}}(s)\right\}$;
        $x^*_T(s_T) = \arg_{x_T} \left\{s^*_{T+1,\bar{u}^A_{T+1},\bar{u}^C_{T+1}} = \mathcal{S}^X(s_T, x_T)\right\}$;
Step 6: Recover $x^*_{T-1}, x^*_{T-2}, \ldots, x^*_{t_0+1}, x^*_{t_0}$
```

References

Ackert, S. P. (2010). Basics of aircraft maintenance programs for financiers. http://aircraftmonitor.com/uploads/1/5/9/9/15993320/basics_of_aircraft_maintenance_programs_for_financiers___v1.pdf. (Accessed on September 28, 2017).

AIRBUS (2017). Airbus A320 maintenance planning document [private document].

Asamov, T., Salas, D. F., & Powell, W. B. (2016). SDDP vs. ADP: The effect of dimensionality in multistage stochastic optimization for grid level energy storage. http://asamov.com/download/SDDP-ADP.pdf. (Accessed on October 1, 2017).

Başdere, M., & Bilge, U. (2014). Operational aircraft maintenance routing problem with remaining time consideration. *European Journal of Operational Research, 235*(1), 315–328.

Bauer-Stämpfli, H. (1971). Near optimal long-term scheduling of aircraft overhauls by dynamic programming. In *Proceedings of the Agifors symposium. Benalmadena, Spain*.

Belobaba, P., Odoni, A., & Barnhart, C. (2009). *Global airline industry*. The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom: John Wiley and Sons.

Boere, N. J. (1977). Air Canada saves with aircraft maintenance scheduling. *Interfaces, 7*(3), 1–13.

Bomberger, E. E. (1966). A dynamic programming approach to a lot size scheduling problem. *Management Science, 12*(11), 778–784.

Buskirk, C. V., Dawant, B., Karsai, G., Sprinkle, J., Szokoli, G., Suwanmongkol, K., & Currer, R. (2002). Computer-aided aircraft maintenance scheduling. technical report. http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=73A54FE02FFA1478C4BAEAE1A98C71A7?doi=10.1.1.201.9088&rep=rep1&type=pdf. (Accessed on December 2, 2017).

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

Duffuaa, S., & Al-Sultan, K. (1997). Mathematical programming approaches for the management of maintenance planning and scheduling. *Journal of Quality in Maintenance Engineering, 3*(3), 163–176.

Etschmaier, M., & Franke, P. (1969). Long-term scheduling of aircraft overhauls. In *Proceedings of the Agifors symposium. Broadway, Great Britain*.

Feo, T. A., & Bard, J. F. (1989). Flight scheduling and maintenance base planning. *Management Science, 35*(12), 1415–1432.

Gascon, A., & Leachman, R. C. (1988). A dynamic programming solution to the dynamic, multi-item single-machine scheduling problem. *Operations Research, 36*(1), 50–56.

Go, H., Kim, J.-S., & Lee, D.-H. (2013). Operation and preventive maintenance scheduling for containerships: Mathematical model and solution algorithm. *European Journal of Operational Research, 229*(3), 626–636.

Graves, G. H., & Lee, C.-Y. (1999). Scheduling maintenance and semiresumable jobs on a single machine. *Naval Research and Logistics, 46*(7), 845–863.

Hoppe, R. H. (2018). Optimization theory II – Lecture notes. https://www.math.uh.edu/~rohop/Spring_12/index.html. (Accessed on November 15, 2018).

Horder, P. (2003). Airline operating costs. http://www.dea.univr.it/documenti/Avviso/all/all520253.pdf. (Accessed on November 15, 2018).

IATA's Maintenance Cost Task Force (2015). Airline maintenance cost executive commentary. https://www.iata.org/whatwedo/workgroups/Documents/MCTF/MCTF-FY2017-Report-Public.pdf. (Accessed on November 15, 2018).

Kiefera, A., Schildeb, M., & Doerner, K. F. (2018). Scheduling of maintenance work of a large-scale tramway network. *European Journal of Operational Research, 270*(3), 1158–1170.

Kinnison, H. A., & Siddiqui, T. (2012). *Aviation maintenance management* (Second edition). Mcgraw-Hill Education.

Lawler, E. L. (1990). A dynamic programming algorithm for preemptive schedule of a single machine to minimize the number of late jobs. *Annals of Operations Research, 26*(1), 125–133.

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

Minister of Justice (2012). Canadian aviation regulations 2012-1, Part I – General provisions, subpart 1 – Interpretation. https://web.archive.org/web/20121227092905/http://www.tc.gc.ca:80/eng/civilaviation/regserv/cars/part1-subpart1-1104.htm. (Accessed on September 28, 2017).

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

Papakostas, N., Papachatzakis, P., Xanthakis, V., Mourtzis, D., & Chryssolouris, G. (2010). An approach to operational aircraft maintenance planning. *Decision Support Systems, 48*(4), 604–612.

Pereira, M., Campodónico, N., & Kelmam, R. (1998). Long-term hydro scheduling based on stochastic models. *EPSOM, 39*, 1170–1190.

Powell, W. B. (2011). *Approximate dynamic programming – Solving the curses of dimensionality*. New York: Wiley-Interscience.

Senturk, C., Kavsaoglu, M. S., & Nikbay, M. (2010). Optimization of aircraft utilization by reducing scheduled maintenance downtime. In *Proceedings of the tenth*

AIAA aviation technology, integration, and operations (ATIO) conference. Fort Worth, Texas.

Senturk, C., & Ozkol, I. (2018). 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. *International Journal of Mechanical Engineering and Robotics Research, 7*(2), 189–196.

Sriram, C., & Haghani, A. (2003). An optimization model for aircraft maintenance scheduling and re-assignment. *Transportation Research Part A, 37*(1), 29–48.

Steiner, A. (2006). A heuristic method for aircraft maintenance scheduling under various constraints. In *Proceedings of the sixth swiss transport research conference. Monte Verità, Ascona*.