Robust Long-Term Aircraft Heavy Maintenance Check Scheduling Optimization under Uncertainty

van der Weide, Tim; Deng, Q.; Santos, Bruno F.

DOI
10.1016/j.cor.2021.105667

Publication date
2021

Document Version
Final published version

Published in
Computers & Operations Research

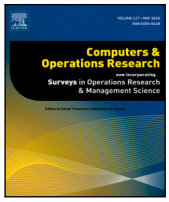
Citation (APA)
van der Weide, T., Deng, Q., & Santos, B. F. (2021). Robust Long-Term Aircraft Heavy Maintenance Check Scheduling Optimization under Uncertainty. *Computers & Operations Research*, *141*, Article 105667. https://doi.org/10.1016/j.cor.2021.105667

Computers & Operations Research 141 (2022) 105667

Contents lists available at ScienceDirect

Computers and Operations Research

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



Robust long-term aircraft heavy maintenance check scheduling optimization under uncertainty

Tim van der Weide, Qichen Deng *, Bruno F. Santos

Air Transport and Operations, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands

ARTICLE INFO

Keywords:
Scheduling
Min–max optimization
Genetic algorithm
Robustness optimization
Aircraft maintenance

ABSTRACT

Long-term heavy maintenance check schedules are crucial in the aviation industry since airlines need them to prepare the required maintenance tools, workforce, and aircraft spare parts. However, most airlines adopt a manual approach to plan the heavy maintenance check schedules in current practice. This manual process relies on the experience of their maintenance planners, and the resulting heavy maintenance schedules need frequent adjustment because of uncertainty. This paper applies a genetic algorithm (GA) to generate robust aircraft heavy maintenance check schedules. It aims to reduce the workload and the frequency of revising heavy maintenance schedules considering uncertainties associated with heavy maintenance check duration and aircraft daily utilization. A major European airline case study shows that the GA finds robust and efficient multi-year aircraft heavy maintenance schedules for a fleet of 45 aircraft in 30 min. Compared with the current approach followed by the airline, the algorithm reduces the total number of heavy maintenance checks by 7% while increasing utilization by 4.4%, which could potentially lead to a reduction of direct annual maintenance costs between $122.5K and $612.5K. Furthermore, when testing the robustness of the 4-years maintenance check schedules produced, a Monte Carlo analysis has shown that all aircraft could be maintained before their check due date for 41% of the episodes simulated, compared to 0.27% of the episodes for the single deterministic scenario approach.

1. Introduction

The spending of global maintenance, repair, and overhaul (MRO) represented 9%–10% of total operational costs of commercial airlines (IATA's Maintenance Cost Task Force, 2019), and heavy aircraft maintenance accounts for more than 70% of these costs and requires the most amount of resources involved in aircraft maintenance. Therefore, it is beneficial for airlines to keep introducing innovations concerning the scheduling of aircraft maintenance and value efficiency, reducing maintenance operation costs in the long term and coping with augmenting demand for aircraft maintenance.

Regular aircraft maintenance is necessary to assure airworthiness, keep the aircraft reliable, and provide assurance of flight safety. Airlines usually group aircraft maintenance tasks into letter checks: A-, B-, C-, and D-checks. In particular, A- and B-checks belong to light maintenance, while C- and D-check are considered heavy maintenance. Each letter check has a different duration, frequency, and set of tasks associated with (Gopalan and Talluri, 1998). During the letter checks, the aircraft stays in the hangar for maintenance. Light maintenance may need one or two days to finish, while heavy maintenance can last for a few weeks.

The frequency of a letter check is determined by the maximum interval between two checks, in the units of calendar days (DY), flight hours (FH), and flight cycles (FC), stated in the maintenance planning document (MPD). The maintenance planners allocate aircraft to maintenance slots at specific days to perform maintenance before its usage parameters (i.e., DY, FH, and FC) reach the limits, in which one maintenance slot is one day of availability of a hangar for performing aircraft maintenance. The deadline for a maintenance check for an aircraft is thus dependent on the start date of its previous maintenance check execution, aircraft utilization, and maximum interval. Overall, aircraft maintenance check scheduling (AMCS) is to determine when and what type of maintenance check should be performed on an aircraft given a planning horizon. It is a typical combinatorial optimization problem (Boere, 1977).

A long-term (usually 3–5 years) heavy aircraft maintenance check schedule is crucial for an airline in the aviation industry. Firstly, it gives stability to the short-term maintenance plans, since the scheduling of the heavy maintenance checks (C-/D-checks) influences the light maintenance checks such as A-/B-checks (Witteman et al., 2021). Therefore, a robust heavy maintenance check schedule can allow stable planning

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

https://doi.org/10.1016/j.cor.2021.105667
Received 19 February 2021; Received in revised form 13 August 2021; Accepted 2 December 2021
Available online 4 January 2022

of light maintenance and the associated resources (e.g., maintenance tools, workforce, and aircraft spare parts). Secondly, it reduces the time and effort of constantly reviewing the maintenance plans, including heavy and, consequently, light maintenance checks. Thirdly, it decreases the risk of having heavy maintenance checks scheduled during the commercial peak periods of the year (e.g., summer, school holidays, Easter, and Christmas periods), given disruptions and the need to keep the fleet always airworthy. However, current methods to the long-term AMCS are mostly computer-aid manual approaches, heavily relying on the experience of maintenance planners (Deng et al., 2021). The initial schedule has to be continuously updated due to the uncertainties from aircraft utilization, unscheduled maintenance tasks, and maintenance activities. Each revision can last for a few days or a week and result in a backlog and impact maintenance costs and quality of service. Considering the influences of uncertainties in AMCS could offer more robust schedules where fewer modifications are necessary. It is essential to plan a robust long-term schedule for aircraft heavy maintenance checks considering uncertainty, resulting in significant cost savings and reducing inefficiencies in the execution of the maintenance schedule.

This paper proposes a min–max scenario optimization approach, combined with a genetic algorithm (GA) methodology, to generate robust aircraft heavy maintenance schedules. The work extends the AMCS optimization presented in Deng et al. (2020) by considering the uncertainties in check duration and daily utilization while computing the long-term schedule of heavy AMCS (H-AMCS). The contributions include:

- It is the first research to formulate the stochastic H-AMCS problem as an integer linear programming (ILP) model.
- It is the first time to address the robust long-term H-AMCS problem considering uncertainty, proposing a min–max approach to solve the problem.
- It uses a GA methodology to generate a robust 4-year heavy maintenance schedule in less than 30 min for a fleet of 40 aircraft. The results of 10 000 test scenarios indicate that more than 40% of the heavy maintenance checks in the robust schedule planned by the GA require no adjustment under different scenarios, i.e., 40% of the heavy maintenance checks can always start before the corresponding aircraft reach their utilization limit regardless of uncertainty from maintenance check duration and aircraft utilization.

This paper's outline is as follows: Section 2 gives an overview of the relevant literature on the current practice of aircraft maintenance scheduling and describes the primary source of uncertainties in H-AMCS. The robust optimization approach and the integer linear programming min–max formulation of the H-AMCS problem is introduced in Section 3. Section 4 presents a GA methodology adapted for the H-AMCS. Next, Section 5 shows the min–max approach results when applied to a case study from a European airline. This section discusses the algorithm performance analysis and the schedule robustness assessment, followed by a sensitivity analysis on some of the parameters used to capture uncertainty in our model. The last section concludes this research and gives an outlook on future work.

2. Literature review

As a part of the AMCS, H-AMCS is also intricate since scheduling an aircraft heavy maintenance check on a specific date impacts aircraft utilization in the future and, consequently, the requirements on future heavy maintenance checks. The long-term H-AMCS has been relying on a manual scheduling approach according to the experience of maintenance planners. In the early 1970s, it took maintenance planning personnel several weeks to create a maintenance schedule (Boere, 1977). Air Canada developed an aircraft maintenance operations simulation model (AMOS) to speed up the planning process, focusing on improving maintenance efficiency and reducing labor and material costs. This work formulated the H-AMCS as a discrete integer programming problem and incorporated many detailed operational constraints. Despite the contribution to H-AMCS formulation, the solution approach was a priority-based simulation-heuristic, similar to the manual planning approach. An experienced maintenance planner had to decide the optimal schedule, shifting checks until a feasible solution was found. The simulation of aircraft utilization, and together with the introduction of a lower utilization bound before a new maintenance check can be scheduled, reduced the time to develop a long-term maintenance schedule of 5 years from several weeks to several hours (Boere, 1977).

Many maintenance planners have since adopted a similar approach to the scheduling of long-term maintenance. Simultaneously, some airlines and researchers have developed more comprehensive integrated tools for AMCS, such as the fleet-planner IFS Maintenix tool (IFS, 2019). However, all the available tools are still relying on the experience of maintenance planners and manual input. Moreover, none of the tools considers the uncertainty in planning long-term aircraft maintenance scheduling to our best knowledge. Thus the maintenance schedules need to be revised frequently.

On the other hand, there is little literature focusing on long-term AMCS. Etschmaier and Franke (1969) introduced an out-of-kilter algorithm to minimize maintenance cost, and Bauer-Stämpfli (1971) developed a dynamic programming (DP) approach for H-AMCS. Both methods were deemed not suitable by Boere (1977) for the environment of Air Canada when developing the previously mentioned simulation model. Furthermore, the author stated that optimization techniques were impractical because an optimal solution can become obsolete with a change in environment and aircraft utilization. More recently, Yan et al. (2008) formulated a zero–one integer programming model for the long-term AMCS and demonstrated a case study with the commercial solver CPLEX. Deng et al. (2020) proposed a DP-based methodology for the long-term AMCS, adopting the forward induction. To deal with the challenge of a multi-dimensional action vector, the authors proposed a priority-based solution to sort the list of aircraft. Besides, a thrifty algorithm, combined with discretization and state aggregation, is used to obtain an optimal schedule for the long-term AMCS.

Since relatively little literature exists on the long-term AMCS, it is relevant to look into short-term aircraft maintenance scheduling problems. The short-term scheduling of aircraft maintenance has received considerably more attention in literature through the aircraft maintenance-routing problem. Airlines can see tangible benefits, such as cost-saving or extra revenue, by optimizing the short-term aircraft maintenance activities or the corresponding flight schedules in a few days or weeks. In the short-term aircraft maintenance scheduling, there are two types of objective functions. First of all, the minimization of the total maintenance cost. This type of objective function can be found in Sriram and Haghani (2003), Papadakos (2009) and Maher et al. (2014), where the problem is formulated as an integer multi-commodity network flow model to minimize total costs of light maintenance. The second type of objective function is to maximize aircraft availability (or to minimize aircraft unavailability). For example, Kozanidis and Skipis (2010) used a mixed-integer bi-objective linear programming model to maximize aircraft availability and flight time for a fleet of fighter aircraft over the considered time-horizon. Alternatively, Başdere and Bilge (2014) aimed at minimizing the aircraft unavailability. The objective in Başdere and Bilge (2014) is very similar to maximizing the utilization of maintenance intervals since it indirectly decreases the number of maintenance checks in the long-term and the number of days on the ground.

There are numerous studies on aircraft maintenance routing problems, such as optimizing aircraft routing and flight departure times to minimize passenger disruptions (Lan et al., 2006), minimizing the total expected propagated delay of the aircraft routes (Liang et al.,

2015), minimizing the total delay and the number of delayed passengers (Ahmed et al., 2017), and minimizing the total expected cost of the propagated delay (Eltoukhy et al., 2018, 2020). Although the aforementioned research works use simulations or scenarios to capture the impact of uncertainty or disruption, the main focus is aircraft routing, and the maintenance requirements are usually incorporated as constraints. None of them touches the heavy maintenance check scheduling and thus provides very little insight on planning robust heavy maintenance check schedules.

From the solution approach perspective, the formulations of maintenance scheduling and aircraft maintenance routing problems discussed before are typically NP-hard, difficult to solve when they are on a large scale. Therefore, the primary solution techniques found in the literature are meta-heuristics, and of which an often encountered example in scheduling literature is the genetic algorithm (GA). Holland (1992) was the first introduced GA as an adaptive method based on the genetic processes of biological organisms. More specifically, the essence of GA is the theory of evolution, in which populations evolve over many generations based on survival of the fittest and natural selection. In Yang and Yang (2012), GA is used to schedule maintenance opportunities based on an original flight plan. Although the idea of GA is fundamental, many simulation experiments show the possibility of the algorithm to come up with feasible maintenance schedules while minimizing the costs. Quan et al. (2007) used a more complex genetic algorithm to address a multi-objective preventive maintenance scheduling problem. Kleeman and Lamont (2005) applied GA on multi-objective scheduling of heavy-maintenance on aircraft engines. Experimental results show the possibility of a GA to efficiently solve the maintenance scheduling while obtaining an acceptable solution. However, one of the main downsides of the GA is the considerable computational effort required for a more complex, large-scale problem. When considering the more general Resource-Constrained Project Scheduling Problem (RCPSP), Elshaer (2017) compares several adaptations of GA introduced in the literature for solving the RCPSP. In the last years, the main research direction is the hybridization of metaheuristics (Zamani, 2010), which combines cross-over operations from GA with a local search method. The main difference with a general GA is that it usually concentrates on the continuous improvement of one primary solution.

When considering the scheduling of aircraft maintenance in literature, most cases focus on the deterministic problem. However, H-AMCS is a complex dynamic project and possesses varying degrees of uncertainty, influencing the execution and creation of a maintenance schedule. For the long-term H-AMCS, it is therefore essential to identify these uncertainties and their effect on the maintenance schedule. Samaranayake (2006) recognizes that uncertainty encountered in aircraft maintenance, particularly from non-routine and unscheduled maintenance, affects the planning and scheduling of aircraft maintenance. The author realized the importance of non-routine findings and stated that about 50% to 60% of the maintenance workload results from the maintenance inspection. A more recent case study in Dinis et al. (2019) shows that this number can be higher and increases with aircraft age.

The uncertainty in workload due to unscheduled maintenance tasks or activities prolongs maintenance checks, affecting the start dates and due dates of subsequent maintenance checks, thus resulting in frequent adjustments to the initial maintenance schedule. Besides, the deadline for a maintenance check also depends on the utilization of an aircraft. Small deviations from expected daily FH and FC can accumulate and, in the long term, significantly impact aircraft utilization, which also impacts the due dates of the upcoming maintenance checks. Because of the uncertainty from maintenance check elapsed time and aircraft daily utilization, a maintenance check schedule often rapidly becomes obsolete (Boere, 1977), leading to the need for revisions. Eventually, it results in a backlog, increasing maintenance costs and the corresponding quality of service. Hence, it is beneficial to look into the stochastic problem to generate more robust schedules.

By far, to our best knowledge, there is no literature available that addresses the AMCS problem while taking uncertainty into account. However, when considering the general aircraft maintenance scheduling problems, some papers exist that include various uncertainties. For example, Mattila and Virtanen (2011) considered the uncertainty in failure rates and maintenance duration when scheduling maintenance for a fleet of fighter aircraft. The uncertainty parameters are modeled with a probabilistic approach and are Gamma distributed. A reinforcement learning approach is applied to find an optimal maintenance policy. However, the capability of the model to be used in actual decision making requires the solution of several different problem instances. Sohn and Yoon (2010) used the random effects Weibull regression model to take non-constant mean time between failure (MTBF) and mean time to repair (MTTR) into account for the dynamic scheduling of preventive maintenance. Overall, the general trend in scheduling literature is laying increasing emphasis on incorporating uncertainty in the models.

It can be concluded that, due to the complexity of scheduling aircraft letter checks, the problem is hard to solve with exact methods. Therefore the main solution techniques found in the literature are meta-heuristics. An often encountered solution method is GA, which provides promising results for similar scheduling problems. From the literature review, it is also clear that there has been little focus on long-term H-AMCS. Next to that, the inherently stochastic nature of maintenance is not considered when creating an initial schedule. A robust long-term schedule for heavy aircraft maintenance would require fewer adjustments, reduce the operation costs, and improve the quality of maintenance service. According to IATA's Maintenance Cost Task Force (2019), the MRO spend spans, on average, 9%–10% of the total operating cost of an airline. Reducing these costs can be very beneficial in the long-term for airlines. The main uncertainties that can significantly impact the maintenance schedule and cost are check duration and aircraft utilization. The goal of developing a scheduling model that takes these uncertainties into account is, therefore, the main focus of this research. In this sense, the research's relevance lies in filling this gap in the literature by approaching the problem with a genetic algorithm and creating a general robust scheduling model framework that takes uncertainty into account.

3. Modeling approach

This section describes the modeling approach proposed to address the H-AMCS for a heterogeneous fleet under uncertainty. We start by introducing the C-check maintenance problem in Section 3.1, followed by a list of assumptions for the problem defined in Section 3.2. Section 3.3 introduces the min–max optimization approach proposed to solve the H-AMCS problem under uncertainty. Section 3.4 describes how scenarios are generated and selected. Lastly, in Section 3.5 we present the ILP model formulation, including the nomenclature, constraints, and the objective function that compose the H-AMCS problem.

3.1. Aircraft heavy maintenance check scheduling

The heavy maintenance of aircraft is commonly divided into two-letter checks, C- and D-checks. In practice, an aircraft undergoes a C-check every 18–36 months and a D-check every 5–6 years. Besides, C-checks are usually planned in sequence and have a sequential number (C1, C2, C3, …). Many airlines already merge D- in C-checks and label them as "heavy C-checks" to avoid grounding the aircraft twice for maintenance. That is, having one D-check in every three/four C-checks, for example, for the aircraft type that has a C-check interval of 24 months:

$$\text{C1},\quad \text{C2},\quad \underbrace{\text{C3}}_{\text{D-check}},\quad \text{C4},\quad \text{C5},\quad \underbrace{\text{C6}}_{\text{D-check}},\quad \text{C7},\quad \ldots \tag{1}$$

During these checks, the aircraft stays in a hangar until the technicians complete all the maintenance tasks. The number of hangars indicates the maximum possible maintenance checks in parallel. After a maintenance check, the associated usage parameters (DY/FH/FC) are all reset to zero, and a new interval restarts, defining the limit for the following check on the same aircraft.

3.2. Assumptions

In this study, we made several assumptions maintenance scheduling, as showed in (**A.1**–**A.6**), based on Boere (1977) and Deng et al. (2020) and real-life maintenance practice:

- **A.1** A heavy maintenance check (C-/D-check) ties up one hangar for the entire duration of the check;
- **A.2** An aircraft ages only based on daily flight hours, for which a probability distribution can be estimated monthly per aircraft from historical data and calendar days;
- **A.3** The minimum time step of the maintenance schedule is one calendar day;
- **A.4** Location of a hangar does not influence check possibility and aircraft routing is flexible;
- **A.5** Other types of letter check can be planned at its due date without considering resource constraints;
- **A.6** Additional hangar slots can be added to make a schedule feasible. In practice, this can either be done with extra work, for instance, at days of the week in each no heavy maintenance work was planned, or with hiring third party services. In both cases, creating additional hangar slots is an expensive option.

A.1 indicates that if an aircraft is undergoing a heavy maintenance check in a hangar, it will stay there until all maintenance tasks are performed. This is exactly the case in practice since towing the aircraft that is undergoing heavy maintenance from one hanger to another not only wastes time and effort but also affects the maintenance works of other aircraft. **A.2** is made because FH and DY are the main usage parameters to determine the heavy maintenance, and **A.3** reflects common practice in the aviation industry. The reason for having **A.4** is that the planning horizon of H-AMCS is 3–5 years (to include at least one C-check for each aircraft), but airlines usually plan the flight routes a few weeks beforehand. Thus, there is no available long-term flight schedule or route. We add **A.5** to take the A-/B-checks into account. If an aircraft is undergoing an A-/B-check, it will be out of operations for 1–2 days, impacting the C- and D-check usage parameters since this aircraft will not be in commercial operation on those days. Like what airline does in practice, we assume that the A- and B-checks are scheduled on their estimated due date. This eventually generates fewer A- and B-checks, providing a conservative approach to estimate the due dates for the C- and D-checks and, hence, still generate a feasible C- and D-check schedule. Assumption **A.6** ensures that if an aircraft reached its maximum interval for a C-check, while there is no available slot, an additional maintenance slot would be created by the MRO to avoid grounding the aircraft. It replaces the concept of using interval tolerance, as discussed in Deng et al. (2020).

3.3. Robust optimization

This paper formulates the H-AMCS as a min–max optimization. According to this approach, we can obtain a robust solution that is optimal in the worst scenario, and it is feasible to all other scenarios considered (Ben-Tal and Nemirovski, 1999):

$$\min F(x, S) = \min_{x} \max_{s_n \in S} F(x, s_n) \tag{2}$$

where x are the decision variables, S is the set of scenarios considered, and F is a convex-concave objective function. Each scenario $s_n \in S$ contains various realizations of the main sources of uncertainty. Min–max optimization problems are often NP-hard, even when considering tractable problems, as shown for several combinatorial problems (Kouvelis and Yu, 1997). Despite this, it is a common approach to solve optimization problems in many application domains, including energy (Ma et al., 2012; Huang et al., 2015), transportation (Venkata Narasimha et al., 2013; Soylu, 2015), and communication (Chiu et al., 2012; Lin et al., 2018) systems.

3.4. Scenario generation

To find robust schedules, we use various scenario inputs that incorporate uncertainty in maintenance scheduling. In our problem, we considered uncertainty to be present when estimating:

- The duration of the heavy maintenance checks: the elapsed time of these checks depends on the number of non-routine tasks (from aircraft manufacturers) added to these checks before the maintenance checks start and the unscheduled tasks found during maintenance inspections. Non-routine tasks include, for instance, tasks associated with non-critical aircraft systems that are run until they fail and tasks created after anomalies are detected in the days before the check.
- the daily utilization of the aircraft in the fleet: the aircraft ages according to the way it is used while performing flights. This depends on the airline network and allocation of specific aircraft to the routes in this network.

The following subsections provide details from the approaches followed to model these sources of uncertainty and how we generated scenarios, described using a maintenance checks duration matrix and a daily utilization matrix.

3.4.1. Heavy maintenance check duration

Airlines usually plan the heavy maintenance checks for their fleet for a future time window of 3–5 years. Since aircraft C-check happens every 18–36 months, a planning horizon of 3–5 years is equivalent to a maximum of four C-checks that have to be scheduled for each aircraft (Ackert, 2010). Therefore, a maintenance checks duration matrix consists of $N \times 4$ checks duration, where N is the number of aircraft in the fleet. To generate the scenarios, we assumed that C- and D-check elapsed times can be estimated from historical data. Data was collected per different classes of C-checks. That is, the C-checks are typically performed in cycles, labeled in a sequential way (i.e., for a cycle of 12 checks, it will be C1, C2, C3, …, C12, C1, C2, …) and having different maintenance tasks associated to them. Therefore, each maintenance label has a different duration, and historical data has to be collected per label. When generating the scenarios, we considered the previous C-check label of each aircraft to determine which checks duration to consider in the duration matrix for each aircraft. For example, if the previous C-check label of Aircraft 1 is C1, the four future checks could be C2, C3, C4, and C5. The elapsed time of C2 would then be randomly chosen from the historical data of C2, and the same for C3, C4, and C5, respectively, to fill in the respective aircraft checks duration matrix for a given scenario.

We randomly generated 10 000 scenarios and randomly selected n_s of the most critical scenarios generated to be the set of scenarios S considered for our min–max optimization approach. We defined the most critical scenarios as those having check durations that are, on average, much shorter or much longer than the mean average checks duration observed in all the scenarios generated. If the check duration is longer than initially planned when creating a C- and D-check schedule, it may result in maintenance capacity problems. On the other hand, if a check is shorter than initially expected, the aircraft comes into operation earlier, directly affecting the deadline for the next C-check (coming at an earlier date). Thus, a shorter check duration can lead

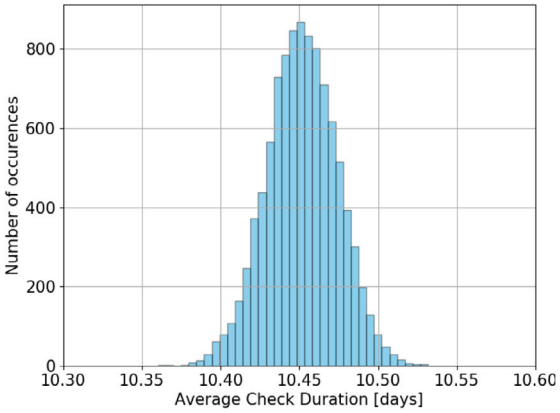


Fig. 1. Distribution of the average check duration from Monte Carlo simulations with 10 000 runs.

Table 1
Utilization matrix, giving the average daily flight hours per aircraft in each month.

Fleet	Jan	Feb	Mar	…	Nov	Dec
Aircraft 1	10.3	9.9	10.5	…	10.5	11.1
Aircraft 2	9.9	10.1	10.1	…	10.3	11.1
⋮	⋮	⋮	⋮	⋱	⋮	⋮
Aircraft N	9.7	9.3	9.5	…	9.6	9.0

to future checks occurring too late in the initial schedule. To select the most critical scenarios we considered the scenarios outside the confidence interval $(d - c \cdot \sigma_d, d + c \cdot \sigma_d)$ for the histogram of the average durations per interval (Fig. 1), where c is based on a predefined confidence level α.

3.4.2. Utilization

We define daily aircraft utilization by the average FH per day in a month. It was assumed that the airline network is equal per each month of the year over the scheduling horizon. Namely, for each aircraft, the utilization of a month is constant, but different months have different constant utilization. A daily utilization matrix is, therefore, an $N\times 12$ matrices, where N is the number of aircraft in the fleet (Table 1). Critical cases occur when the number of FH observed is more than what was expected. The extra FH causes the aircraft to reach the maximum interval earlier and could result in planning checks too late in the initial schedule.

For each aircraft and month, the mean and standard deviation are known based on historical data. Resulting in a utilization vector, $\mathbf{U}$, of $N \times 12$ random variables:

$$\mathbf{U} = \{U_1,\ U_2,\ \ldots,\ U_N\} \tag{3}$$

$$U_i = \{U_i^{\text{Jan}},\ U_i^{\text{Feb}},\ \ldots, U_i^{\text{Dec}}\} \tag{4}$$

We assume that these random variables are jointly normal, and they can be modeled in a multivariate normal distribution. The multivariate normal distribution, $\mathrm{U} \sim N(\mu, \Sigma)$, has mean μ and covariance matrix Σ as shown in (5) and (6).

$$\mu = \begin{bmatrix} \mu_1^{\text{Jan}} \\ \vdots \\ \mu_1^{\text{Dec}} \\ \mu_N^{\text{Jan}} \\ \vdots \\ \mu_N^{\text{Dec}} \end{bmatrix} \tag{5}$$

$$\Sigma = \begin{bmatrix} \sigma_{1,1}^{\text{Jan,Jan}} & \sigma_{1,1}^{\text{Jan,Feb}} & \cdots & \sigma_{1,1}^{\text{Dec,Dec}} \\ \sigma_{1,2}^{\text{Jan,Jan}} & \sigma_{1,2}^{\text{Jan,Feb}} & \cdots & \sigma_{1,2}^{\text{Dec,Dec}} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{N,N}^{\text{Jan,Jan}} & \sigma_{N,N}^{\text{Jan,Feb}} & \cdots & \sigma_{N,N}^{\text{Dec,Dec}} \end{bmatrix} \tag{6}$$

The covariance matrix captures the fact that the daily utilization of an aircraft depends on the utilization of the other aircraft in the fleet. If one aircraft is used more than the average, there should be other aircraft in the fleet that were underused. We follow the idea of Vershynin (2012) to estimate the covariance matrix with a sample size X:

$$\Sigma_x = \frac{1}{X}\sum_{i=1}^{X} U_i \otimes U_i \tag{7}$$

Given a mean vector μ, and sample covariance matrix Σ, we can reproduce the aircraft utilization for Monte Carlo simulation. To ensure a good representation of the possible utilization outcomes, it is necessary to select matrices consistent with the given mean vector and covariance matrix and enclose a given proportion of the sample space. Eck et al. (2015) defines a method of selecting scenarios at a fixed probability level, α, for robust network optimization under uncertain demands. In the multivariate distribution, the points with the same probability can be found based on the Mahalanobis distance (Mahalanobis, 1936). The fixed probability level α is the same α that is used for the confidence interval in Section 3.4.1. However, there is an infinite number of possible utilization matrices with the corresponding Mahalanobis distance at a chosen probability level. To select daily utilization matrices, $n = 100$ points are located corresponding to α. For the daily aircraft utilization, the critical cases are those with the highest average daily flight hours, since a higher daily utilization could mean that the aircraft would reach the limit of the interval before the start date of the maintenance check. Therefore, the n_s daily utilization matrices with the highest overall average daily flight hours are selected.

3.4.3. Scenarios generation

As described earlier, a scenario consists of a duration matrix (D_{s_n}) and a daily utilization matrix (U_{s_n}), i.e., $s_n = \left\{D_{s_n}, U_{s_n}\right\}$. Given size n_s and probability level α, n_s duration matrices and n_s daily utilization matrices are selected. By combining all possible duration matrices with all daily utilization matrices, $n_s \times n_s$ scenarios are generated. In addition to the generated scenarios, the most expected scenario is also considered, being composed by the matrices with the average C-checks duration and average daily utilization per aircraft, as observed in the historical data. A size of $n_s = 2$ therefore results in 2×2+1 = 5 scenarios, and for $n_s = 3$ we would consider ten scenarios.

It is worth mentioning that the historical data have maximum and minimum values for maintenance check elapsed time and aircraft daily utilization. Therefore, all data points are generated from a truncated probability distribution with values between the associated maximum and minimum for the H-AMCS study.

3.5. Problem formulation

This subsection presents the novel integer linear programming (ILP) formulation of the H-AMCS problem. We start by introducing the nomenclature, followed by the objective function and constraints.

3.5.1. Nomenclature

Sets

I	Set of aircraft in the fleet
J_k^i	Set of sequential type k checks to consider for aircraft i
S	Set of scenarios, of size $(n_s)^2 + 1$
t_0	The first day in planning horizon
T	The final day in planning horizon

Parameters

i	Aircraft indicator
j_k	Check indicator (of type k check)
k	Check type indicator
C_{E}	Cost for having an extra slot assigned
C_{G}	Daily penalty for grounding an aircraft and waiting for an available maintenance slot
d_k	Minimum time between the start dates of two consecutive checks
$d^t_{i,j_k}(s_n)$	Duration of check j_k for aircraft i in scenario s_n
D_{s_n}	Aircraft maintenance check duration matrix under scenario s_n
$I^i_{k\text{-FH}}$	Maximum interval in FH for type k check, $k \in \{\mathrm{A},\mathrm{B},\mathrm{C}\}$ for aircraft i
$I^i_{k\text{-DY}}$	Maximum interval in DY for type k check, $k \in \{\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}\}$ for aircraft i
L^k_t	Number of available slots for type k check at time t
N	Fleet size
s_n	A realizations of specific C- and D-check elapsed time and aircraft daily utilization
U_{s_n}	Fleet utilization matrix under scenario s_n
$U^t_i(s_n)$	Daily flight hours for aircraft i at time t under scenario s_n

Decision variables

x^t_{i,j_k}	$x^t_{i,j_k} = 1$ if a check j_k (of type k) starts at t for aircraft i, 0 otherwise
$y^k_{i,t}(s_n)$	FH since previous type k check for aircraft i at time t, $k \in \{\mathrm{A},\mathrm{B},\mathrm{C}\}$, for scenario s_n
$D^k_{i,t}(s_n)$	DY since previous type k check for aircraft i at time t, $k \in \{\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}\}$, for scenario s_n
$E_t(s_n)$	$E_t(s_n) = 1$ if extra slot allocated at time t, for scenario s_n, 0 otherwise
$m^k_{i,t}(s_n)$	$m^k_{i,t} = 1$ if aircraft i is having a type k check at t, for scenario s_n, 0 otherwise, $k \in \{\mathrm{C},\mathrm{D}\}$
$M_{i,t}(s_n)$	$M_{i,t} = 1$ if $\exists k,\ m^k_{i,t} = 1$, 0 otherwise

3.6. Constraints formulation

Several constraints are involved in the formulation of the H-AMCS problem. These constraints can be divided into utilization, operational, and check constraints.

3.6.1. Utilization constraints

An aircraft must undergo a type k check before the corresponding usage parameters (DY and FH) reach the maximum as defined by the type k check interval. If any of the usage parameters aircraft exceeds the interval, the aircraft has to be grounded, resulting in heavy commercial revenue losses. Therefore, it is required that all aircraft are maintained within their respective intervals. Thus, given a time window of T, all aircraft usage parameters should be lower or equal to the interval. This is formulated in (8) and (9) for the flight hours ($y^k_{i,t}$) and calendar days ($D^k_{i,t}$), respectively (D-check has a threshold interval expressed only in calendar days):

$$y^k_{i,t}(s_n) \le I^i_{k\text{-FH}} \qquad \forall i \in I,\ t \in T,\ k \in \{\mathrm{A},\mathrm{B},\mathrm{C}\}\ \ s_n \in S \tag{8}$$

$$D^k_{i,t}(s_n) \le I^i_{k\text{-DY}} \qquad \forall i \in I,\ t \in T,\ k \in \{\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}\}\ \ s_n \in S \tag{9}$$

The usage parameters of all aircraft need to be updated every time step, defined in Eqs. (10) and (11). Note that the FH for an aircraft is only updated with the average daily average FH for the month of time step t if the aircraft is in commercial operation, i.e., $M_{i,t} = 0$ ($m^k_{i,t} = 0\ \forall k$). These average daily FH are pre-computed and dependent on the scenario s_n.

$$y^k_{i,t+1}(s_n) = \left[1 - m^k_{i,t}(s_n)\right] y^k_{i,t}(s_n) + \left[1 - M_{i,t}(s_n)\right] U^t_i(s_n)$$
$$\forall i \in I,\ t = 1,\dots,T-1,\ k \in \{\mathrm{A},\mathrm{B},\mathrm{C}\} \tag{10}$$

$$D^k_{i,t+1}(s_n) = \left[1 - m^k_{i,t}(s_n)\right]\left[D^k_{i,t}(s_n) + 1\right] \quad \forall i \in I,\ t = 1,\dots,T-1,$$
$$k \in \{\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}\} \tag{11}$$

3.6.2. Operational constraints

If an aircraft is being maintained, it occupies one maintenance slot for the duration of its check. The duration of a specific check j_k for aircraft i, $d^t_{i,j_k}(s_n)$, depends on the start time t and the scenario s_n. Since maintenance work is halted during weekends and public holidays, the number of weekend days and public holidays for a specific check period also need to be added to the total duration of the check:

$$d^t_{i,j_k}(s_n) = \text{number of working days for a check } j + \text{weekends} + \text{public holidays} \tag{12}$$

Since the binary variables are $m^k_{i,t} = 1$ if aircraft i starts maintenance check j_k at time t for the duration of the check, we have the following constraint:

$$\sum_{t \in \left[t, t + d^t_{i,j_k}(s_n)\right]} m^k_{i,t}(s_n) \ge d^t_{i,j_k}(s_n) \times x^t_{i,j_k} \qquad \forall i \in I,\ j \in J_i,\ t \in T,\ s_n \in S \tag{13}$$

Besides, the number of slots used for type k check at t should not exceed the capacity L^k_t (L^k_t has to be defined beforehand by airlines). For example, airlines often require that no C-checks are scheduled during peak periods (i.e., summer and holiday periods). This is because performing a C-check during these periods will cause a high commercial revenue loss. During these periods, the available slots can be set to zero. As indicated in **A.6**, an airline can create an extra slot, $E_t(s_n)$, to a day if required to maintain all aircraft within their interval. Therefore, the capacity constraint is defined as.

$$\sum_{i \in I} m^k_{i,t}(s_n) \le L^k_t + E_t(s_n) \qquad \forall t \in T,\ s_n \in S \tag{14}$$

3.6.3. Maintenance check constraints

A maintenance cycle consists of several sequential maintenance checks. These checks are planned subsequently. For example, in H-AMCS (C-/D-check), after the first check (C1), C2 has to be scheduled and then C3, and so on. To formulate the maintenance cycle, we introduced the following to ensures that check j_k has to be scheduled before a check $j_k + 1$:

$$x^t_{i,j_k} \ge x^t_{i,j_k+1} \qquad \forall t = 0,1,\dots,T,\ \forall i \in I,\ j_k = 0,1,\dots \tag{15}$$

Furthermore, a specific check j_k can only be scheduled once.

$$\sum_{t \in T} x^t_{i,j_k} \le 1 \qquad \forall i \in i, j \in J^k_i \tag{16}$$

Lastly, we considered that a minimum time between the start dates of two consecutive type k checks may be required due to resource preparation. Some airlines need it to prepare the maintenance tools and aircraft spare parts. This can be defined by the airline as d_k. The respective constraint is described in (17):

$$\sum_{i \in I} \sum_{j_k \in J^k_i} x^\tau_{i,j_k} \le 1 \qquad \forall \tau \in \left[t - d_k, t + d_k\right] \tag{17}$$

3.7. Objective function (Fitness Function)

This paper uses the following objective function, which is also the proposed fitness function for the genetic algorithm (GA):

$$\min_x \max_{s_n \in S} F(x, s_n) \tag{18}$$

where

$$F(x, s_n) = \sum_{t=t_0}^{T} \sum_{k} \sum_{i \in I} \sum_{j \in J_i} \left[\left| I^i_{k\text{-FH}} - y^k_{i,t}(s_n) \right| + P^k_{i,t}(s_n) \right] \times x^t_{i,j_k} + C_{\mathrm{E}} \sum_{t=t_0}^{T} E_t(s_n), \quad x = \bigcup_{i,j_k,t} x^t_{i,j_k} \tag{19}$$

In (19), $\left| I^i_{k\text{-FH}} - y^k_{i,t}(s_n) \right|$ represents the unused flight hours of aircraft i, checks j_k at time t. Any violation of utilization constraints is penalized by $P^k_{i,t}(s_n)$ proportional to the level of the infeasibility of the schedule:

$$P^k_{i,t}(s_n) = \begin{cases} 0, & y^k_{i,t}(s_n) \le I^i_{k\text{-FH}} \text{ and } D^k_{i,t}(s_n) \le I^i_{k\text{-DY}} \\ \frac{y^k_{i,t}(s_n) - I^i_{k\text{-FH}}}{U^t_i(s_n)} C_{\mathrm{G}}, & y^k_{i,t}(s_n) \ge I^i_{k\text{-FH}} \\ \left[D^k_{i,t}(s_n) - I^i_{k\text{-DY}} \right] C_{\mathrm{G}}, & D^k_{i,t}(s_n) \ge I^i_{k\text{-DY}} \end{cases} \tag{20}$$

C_{G} is the cost per day when an aircraft has to stay on the ground waiting for the check. The second part $\sum_{t \in T} E_t(s_n) \times C_{\mathrm{E}}$ represents the cost for creating additional slots at time t. As indicated in **A.6**, an airline can assign an extra hangar slot to make a schedule feasible. However, this requires the need for mechanics to work extra time and can be costly. Besides, the ratio between C_{G} and C_{E} determines the trade-off between having extra slots or having aircraft being maintained too late. Since grounding an aircraft means a high commercial revenue loss, in principle, C_{G} should be much higher than C_{E}.

The general idea of (18) is to minimize the total unused flight hours while having the least violations of utilization constraints and creating the least extra maintenance slot. The reason is that aircraft maintenance cost data is often confidential or incomplete and hard to relate to specific check types. Moreover, the costs of an aircraft being out of operations are higher than the daily costs of a maintenance check. Minimizing the total unused flight hours of a fleet exploits the maintenance intervals between checks, indirectly reducing the number of maintenance checks in the long-term and, consequently, the days out of operation in the scheduling horizon. It makes all aircraft the fleet in commercial operation as long as possible and implicitly reduces the total costs for maintenance operations and aircraft spare parts. Hence, it is considered the most suitable objective function for the problem.

Each schedule in the population has to be evaluated to assess how good the schedule actually is. This evaluation is done according to the min–max objective function (18) subject to a penalty that captures violations to the utilization constraints (8) and (9).

In general, a maintenance check schedule with a lower cost is considered a better candidate. Simulation of multiple schedules (chromosomes) for all scenarios (i,e, $\forall s_n \in S$) was performed using the SymPy package (Meurer et al., 2017) in python and provided the number of flight hours flown at each scheduled check and any penalty that might have been assigned. An example of the simulator output can be seen in Fig. 2. Here the fitness is displayed for a fleet of 45 aircraft. The first entries in the fitness list (0–44) indicate the lost flight hours per aircraft and any penalty that might have been given, corresponding to the first part of (19). The slots entry indicates whether extra slots are needed with the evaluated schedule, corresponding to the second part of (19). The total entry gives the overall fitness of a maintenance check schedule.

Each heavy maintenance schedule is assessed against all scenarios $s_n \in S$. The one that leads to the highest total cost is considered the worst-case scenario, and the associated total cost is the fitness of the schedule. If the maximum number of iterations has been achieved, the GA stops and returns the schedule corresponding to the lowest total fitness value.

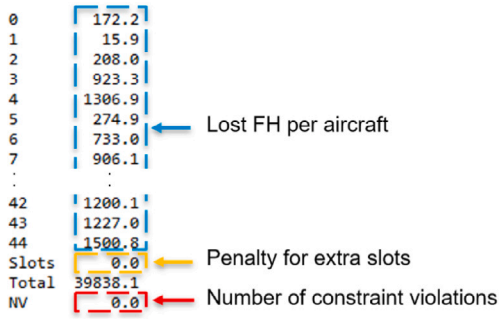


Fig. 2. Example of fitness evaluation for a schedule with a fleet of 45 aircraft.

4. Genetic algorithm

The problem formulated in the previous section is hard to solve for large instances. Therefore, we adapt a GA for solving H-AMCS for a fleet of heterogeneous aircraft under uncertainty. The overall procedure of the GA is illustrated in Fig. 3, following the traditional GA algorithm structure as described in Kramer (2017). First of all, we generate several initial aircraft maintenance check schedules (chromosomes), not necessarily all feasible, using an ϵ-greedy algorithm. This set of schedules is the initial population. After that, every initial schedule in the population is evaluated according to an evaluation function. A set of individual schedules are then selected (parents) and progress to create a new set of schedules (a new generation).

Based on several parameters, selecting individual schedules (Parents) for creating a new population occurs. From the combination of parents, new schedules are created through crossover and mutation, which form a new population. The fitness of the new population is then assessed, and the cycle starts again. This iterative process continues until certain stopping criteria are met. To reduce variability in GA output, we use parallel computing to simulate different, independent runs on a quad-core workstation. After that, we choose the best outcome from the independent runs as the final result. This approach, also known as probability amplification, has been proven effective for job shop scheduling of two machines and several other scheduling (Sudholt, 2015). The following sub-sections explain in more detail the different modules of the GA.

4.1. Chromosome representation

To represent a C-check maintenance schedule in the GA, it has to be encoded in the form of a chromosome. A list of integer numbers is chosen to represent the schedule. The layout can be seen in Table 2. The rows represent the C-check schedule of each aircraft (Aircraft 1, …, Aircraft N) and are referred to as genes. The columns represent which C-check is scheduled. C-I indicates the first coming C-check in the cycle of the aircraft. Meaning that if an aircraft previously had a C8 check, **C-I** represents the C9 C-check, **C-II** for C10, etc. Each integer number in the gene indicates the starting date for that specific check, measured in days since the start of the scheduling horizon. The **C-I** check for Aircraft 1 in Table 2 is, for example, scheduled to start at time step 396 in the planning horizon, meaning 396 days after the start of the horizon. An integer number of "-1" indicates that specific check is not scheduled in the planning horizon.

4.2. Initialization of population

To start searching for near-optimal solutions for H-AMCS with GA, an initial population has to be generated. This initial population consists of a predetermined number (population size) of varying schedules. These initial schedules are created with an ϵ-greedy algorithm, as presented in Algorithm 1.

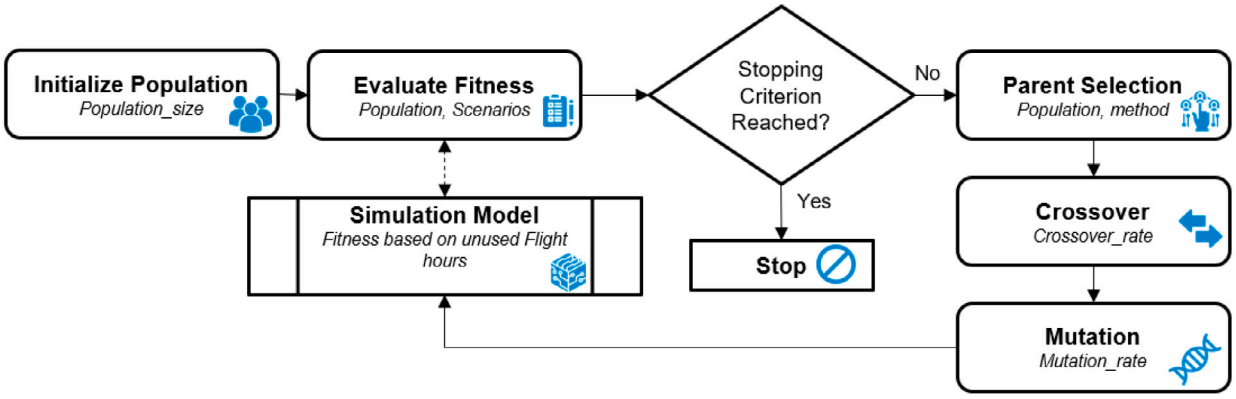


Fig. 3. Flowchart of a genetic algorithm methodology for heavy aircraft maintenance check scheduling.

Table 2
Chromosome representation of the C-check maintenance schedule.

Tail number	C-I	C-II	C-III	C-IV
Aircraft 1	396	1089	−1	−1
Aircraft 2	88	506	1123	−1
Aircraft 3	135	885	1295	−1
Aircraft 4	424	1120	−1	−1
⋮	⋮	⋮	⋮	⋮
Aircraft N	t_{n1}	t_{n2}	t_{n3}	t_{n4}

The algorithm simulates the usage parameters for all aircraft in the fleet over the planning period. The parameters are updated according to (10) and (11). If at time step t, an aircraft i has to be scheduled for an A- or B-check, the flight hour parameter, $y^{\mathrm{C}}_{i,t}$, is not updated since the aircraft is being maintained and does not fly on that day. All aircraft that have usage parameters within ϵ $(0 < \epsilon \leq 1)$ of their interval at time step t have to be scheduled for a C-check, where ϵ is a randomly generated normally distributed number. This gives a list of aircraft for which a check has to be scheduled at t.

Note that the ϵ-greedy algorithm does not have to generate a feasible schedule. There can be a situation that many aircraft have C- and D-checks on the same day, and each of the aircraft is given a random priority. This priority is used to determine the order of maintenance check execution. If an aircraft i has a type k check scheduled at t, the usage parameters, $y^k_{i,t}$ $(\forall k \in \{\mathrm{A},\mathrm{B},\mathrm{C}\})$, and $D^k_{i,t}$ $(\forall k \in \{\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}\})$ of the aircraft are set to zero for the duration of the check (D-check has only one usage parameter $D^{\mathrm{D}}_{i,t}$). After this, the next aircraft in the priority order is scheduled until all aircraft from the candidate list have been planned (creating extra slots if necessary). This process is repeated for every time step in the planning horizon. Resulting in one initial maintenance schedule for the entire fleet.

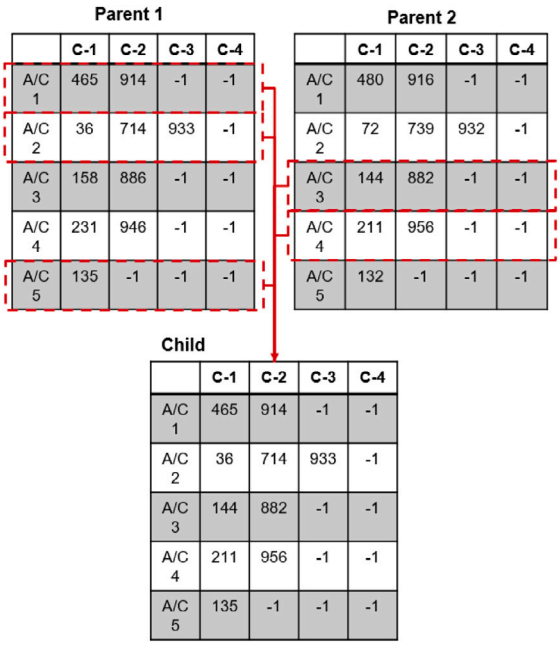
Due to the introduction of ϵ and a random scheduling priority, the algorithm can produce various initial schedules that may or may not be feasible. Updating the usage parameters and the duration of a type k check depends on the scenario used. When running GA with different scenarios, $s_n \in S$, the initial population is generated by running the ϵ-greedy algorithm, with a uniform randomly chosen scenario. This process repeats until the population size is reached.

4.3. Parent selection

Based on the fitness values of all heavy maintenance schedules in the population, several schedules are selected for the mating pool. From this mating pool, schedules are paired up in parent combinations, from which new schedules are generated through the crossover. These new schedules form the next generation. Several selection techniques have been studied in Goldberg and Deb (1991), where n-tournament selection is shown to have good convergence and computational time complexity properties. The principle of n-tournament selection is based

Algorithm 1 ϵ-greedy algorithm for creating initial heavy maintenance check (C-/D-check) schedule

Initialize $D^{\mathrm{C}}_{i,0}$, $D^{\mathrm{D}}_{i,0}$, $y^k_{i,0}$ for all aircraft $i = 1, 2, \ldots, N$ and for $k \in \{\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}\}$ ▷ Initial fleet status
for ← 1 to N **do**
  if Aircraft i is undergoing a C- and D-check **then**
    Aircraft{i}.check ← End date of the C- and D-check
  else
    Aircraft{i}.check ← −1
  end if
end for
chromosome ← [ ]
for $t \leftarrow 1$ to T **do**
  for $i \leftarrow 1$ to N **do**
    if Aircraft{i}.check < t **then**
      Aircraft{i}.check ← −1
    end if
    $\Delta u \leftarrow U^t_i(s_n)$
    if $y^k_{i,t} \geq I^k_{k-\mathrm{FH}}$ $\forall k \in \{\mathrm{A},\mathrm{B}\}$ **then**
      $y^k_{i,t} \leftarrow 0$ $\forall k \in \{\mathrm{A},\mathrm{B}\}$
      $\Delta u \leftarrow 0$
    else
      if Aircraft{i}.check > 0 **then**
        $y^{\mathrm{C}}_{i,t} \leftarrow 0$ and $D^{\mathrm{C}}_{i,t} \leftarrow 0$ ▷ Aircraft i is undergoing a C-check
        $\Delta u \leftarrow 0$
      end if
    end if
    $y^k_{i,t} \leftarrow y^k_{i,t} + \Delta u$
    $D^k_{i,t} \leftarrow D^k_{i,t} + 1$ $\forall k \in \{\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}\}$
    $\epsilon \leftarrow$ randu(1) ▷ randu(1) function generates a uniformly distributed random number between 0 and 1
    if $y^{\mathrm{C}}_{i,t} \geq \epsilon \cdot I^i_{\mathrm{C\text{-}FH}}$ or $D^{\mathrm{C}}_{i,t} \geq \epsilon \cdot I^i_{\mathrm{C\text{-}DY}}$ or $D^{\mathrm{D}}_{i,t} \geq \epsilon \cdot I^i_{\mathrm{D\text{-}DY}}$ and $\sum_{i \in I} \sum_{j_{\mathrm{C}} \in J^{\mathrm{C}}_i} x^{\tau}_{i,j_{\mathrm{C}}} \leq 1$ $\forall \tau \in [t - d_{\mathrm{c}}, t]$ **then**
      chromosome ← chromosome $\cup \left\{ \left(i, t, t + d^t_{i,j_{\mathrm{C}}}(s_n) \right) \right\}$ ▷ Schedule a C-check for Aircraft i
      Aircraft{i}.check ← $\left\{ t + d^t_{i,j_{\mathrm{C}}}(s_n) \right\}$
      $y^{\mathrm{C}}_{i,t} \leftarrow 0$ and $D^{\mathrm{C}}_{i,t} \leftarrow 0$ ▷ Reset the usage parameters of C-check to 0 for Aircraft i
      if $D^{\mathrm{D}}_{i,t} \geq \epsilon \cdot I^i_{\mathrm{D\text{-}DY}}$ **then**
        $D^{\mathrm{D}}_{i,t} \leftarrow 0$ ▷ Reset the usage parameters of D-check to 0 for Aircraft i
      end if
    end if
  end for
end for
return chromosome



	C-1	C-2	C-3	C-4
A/C 1	465	914	-1	-1
A/C 2	36	714	933	-1
A/C 3	158	886	-1	-1
A/C 4	231	946	-1	-1
A/C 5	135	-1	-1	-1

	C-1	C-2	C-3	C-4
A/C 1	480	916	-1	-1
A/C 2	72	739	932	-1
A/C 3	144	882	-1	-1
A/C 4	211	956	-1	-1
A/C 5	132	-1	-1	-1

	C-1	C-2	C-3	C-4
A/C 1	465	914	-1	-1
A/C 2	36	714	933	-1
A/C 3	144	882	-1	-1
A/C 4	211	956	-1	-1
A/C 5	135	-1	-1	-1

Fig. 4. Uniform crossover, where offspring is generated by randomly selecting a gene from one of the parents.

on selecting a set of n individuals uniformly at random from the population. From the set of n individuals, the individual with the best fitness value is selected for the mating pool. This whole process is repeated n times, with replacement. The size of set n is a trade-off between exploitation and exploration, and the most common size is $n = 2$, the binary tournament selection (Rowe, 2015). Larger sample sizes will increase the probability that only the best individuals will be selected, losing some exploration properties. In scheduling literature, Hartmann (1998) has researched several selection techniques, where tournament selection with size $n = 3$ performs better than the binary tournament selection. Therefore, the 3-tournament selection is used in this paper. To select the best individual from the population, the tournament selection is combined with an elitist method. The n best individuals from the population are included in the next generation. This way, the best performing schedules are not lost during crossover or mutation.

4.4. Crossover

From the mating pool, individuals are randomly paired to form groups of parents. With random probability equal to the crossover rate, each pair generates two new schedules for the next generation based on crossover techniques. If no crossover occurs, both individuals in the parent pair will go through to the next generation. The main techniques are single-point, two-point, and uniform crossover Rowe (2015). For this problem, the uniform crossover has been selected. Hu and Paolo (2009) successfully applied this approach for the aircraft arrival sequencing and scheduling problem with a similar integer matrix chromosome representation. In the uniform crossover, each gene in the chromosome is randomly selected from the parents creating a new chromosome. This is done two times to generate two new schedules from each parent pair. An example of the uniform crossover operation can be seen in Fig. 4 for a fleet of five aircraft.

The procedure of generating a new population can be seen in Algorithm 2. Firstly, several n_{best} best schedules from the population are added to the new population. Secondly, each parent pair generates two new schedules to add to the new population. If a random number is lower than or equal to the crossover rate, these two schedules are generated by uniform crossover. Otherwise, both individuals from the parent pair are added to the new population.

Algorithm 2 Crossover Procedure

```
Initialize crossover_rate, mating_pool population
New_Population ← [ ]
New_Population←New_Population∪ {best n_best from population}
for parent_pair∈ mating_pool do
    if randu(1) ≤ crossover_rate then      ▷ randu(1) generates a uniformly distributed random number between 0 and 1
        child_1 ← uniform_crossover(parent_pair)
        child_2 ← uniform_crossover(parent_pair)
    else
        {child_1, child_2} ← parent_pair
    end if
    New_Population←New_Population∪ {child_1, child_2}
end for
return New_Population
```

Algorithm 3 Mutation Procedure

```
Initialize population, mutation_rate, scenarios
n_check ← Total number of heavy maintenance checks
for chromosome ∈ population do
    if randu(n_check) ≤ mutation_rate then
        n ← randu(n_check) ▷ randu(n_check) generates a uniformly distributed random number between 0 to n_check
        Randomly switch the start dates of n maintenance checks in the chromosome.
    end if
end for
```

4.5. Mutation

After a new population has been generated by crossover, mutations (change of start dates of maintenance check) in chromosomes can occur for some aircraft. The probability that mutation occurs is defined as the mutation rate. It is most beneficial to schedule the C-checks at the latest possible date. Therefore, the mutation is not random but follows the same ϵ-greedy algorithm presented in Algorithm 1. All aircraft outside of this subset keep their current schedule (gene). The mutation procedure is explained in Algorithm 3. An example of the mutation procedure can be seen in Fig. 5.

4.6. Simulation model and fitness evaluation

Given a heavy maintenance check schedule x, the "Simulation Model" shown in Fig. 3 is a model framework using a SymPy package of Python (Meurer et al., 2017) to calculate the unused FH, the number of constraints violated, and the number of extra slots created day by day, from t_0 to T according to selected utilization matrix (U_{s_n}) and maintenance check duration matrix (D_{s_n}). After that, we can compute the fitness value according to Eqs. (19) and (20).

4.7. Algorithm detail

The GA methodology for H-AMCS is shown in Algorithm 4. When it terminates, i.e., the algorithm reaches its pre-defined maximum iterations, it returns the schedule with the minimum fitness value.

5. Case study

We validate the proposed modeling approach and the GA methodology using some randomly generated aircraft maintenance data. Firstly, the problem is simplified to compare the output of the GA to the output of an exact method for several test cases (Section 5.1). The exact method is based on the ILP formulation and solved using the commercial LP solver CPLEX. Secondly, the ability of the GA to find

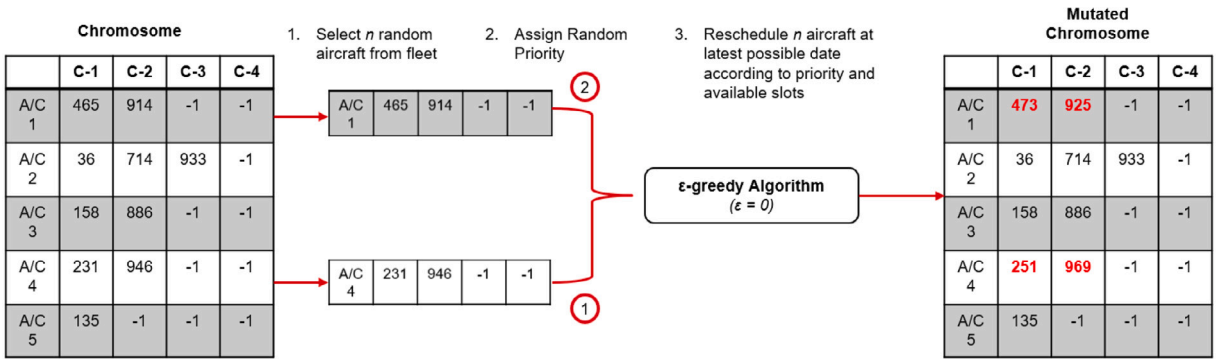


Fig. 5. An example of the mutation procedure, where 2 aircraft are randomly selected to be rescheduled.

Algorithm 4 GA Methodology

```
Initiate population, scenarios, available slots, fleet
Population_Fitness ← [ ],   Output ← [ ]
for iteration ∈ [0, Max_Iteration] do
    fitness_val ← 0
    for Chromosome ∈ population do
        checks ← Chromosome
        fitness_temp ← 0
        output_temp ← [ ]
        for Scenario ∈ scenarios do
            U_{s_n}, D_{s_n} ← Scenario                ▷ Utilization and duration matrix based on scenario
            fitness_val ← Evaluation_Fitness(checks, U_{s_n}, D_{s_n})   ▷ Compute fitness value according to Eq. (19)
            if fitness_val > fitness_temp then
                fitness_temp ← fitness_val
                output_temp ← checks
            end if
        end for
    end for
    Population_Fitness.insert(fitness_temp)
    Output.insert(output_temp)
    population ← Crossover(population, crossover_rate)   ▷ Crossover according to Algorithm 2
    population ← Mutate(population, mutation_rate)       ▷ Mutation according to Algorithm 3
end for
index ← argmin(Population_Fitness)
x_final ← Output[index]
```

efficient near-optimal solutions is assessed by comparing the outcome of the GA to results from a dynamic programming (DP) based methodology described in Deng et al. (2020) (Section 5.2). C_G is 10^5 according to Deng (2019) and C_E is set to 500 according to current practice of our airline partner. In Section 5.3, we discuss the value of the min–max optimization approach by assessing the robustness of the schedules produced with such approach. Lastly, a sensitivity analysis on the effect of probability level α and the number of scenarios on schedule robustness is performed (Section 5.4). All test runs with GA are performed with a mutation rate of 0.6, a crossover rate of 0.9, a population size of 20, using parallel computing on a quad-core workstation. These parameters were found using grid search (Bergstra and Bengio, 2012). The details for determining the mutation rate, crossover rate, and population size can be seen in van der Weide (2020).

5.1. Model validation

From the literature review in Section 2, it is clear that the complexity of AMCS makes it difficult to apply exact methods to large-scale problems. Therefore, the GA and benchmark model are evaluated for small-scale test problems. For the test problems, the time step, t, was set to be weekly, and only the deterministic scenario is taken into account. Furthermore, the GA and ILP only consider the flight hour parameter constraints. The D-check and calendar day parameters were excluded from the problem formulation. A total of four test cases are created with varying available slots per week and the number of aircraft in the fleet:

- **Case 1:** 10 aircraft, 1 hangar available
- **Case 2:** 20 aircraft, 2 hangars available
- **Case 3:** 30 aircraft, 3 hangars available
- **Case 4:** 40 aircraft, 3 hangars available

In the ILP model, the number of decision variables depends on the fleet size N and the planning horizon $T - t_0 + 1$ since one day of an aircraft corresponds to one decision variable. This gives $N\left(T - t_0 + 1\right)$ decision variables. Besides, since only C- and D-checks are considered in this study and D-checks are merged in C-checks, the fleet size N, the planning horizon $T - t_0 + 1$, the hangar capacity L_t^{C} and (8)–(17) together determine the following number of constraints:

$$\underbrace{N(T-t_0+1)}_{\text{Eq. (8)}}+\underbrace{2N(T-t_0+1)}_{\text{Eq. (9)}}+\underbrace{N(T-t_0+1)}_{\text{Eq. (10)}}+\underbrace{2N(T-t_0+1)}_{\text{Eq. (11)}}$$
$$+\underbrace{N(T-t_0+1)}_{\text{Eq. (13)}}+\underbrace{N\sum_{t_0}^{T}L_t^{\mathrm{C}}}_{\text{Eq. (14)}}$$
$$+\underbrace{N(T-t_0+1)}_{\text{Eq. (15)}}+\underbrace{N(T-t_0+1)}_{\text{Eq. (16)}}+\underbrace{N(T-t_0+1)}_{\text{Eq. (17)}}$$
$$=10N(T-t_0+1)+N\sum_{t_0}^{T}L_t^{\mathrm{C}} \tag{21}$$

Table 3
Comparison of computation time and objective value between the GA and the exact method for 4 small test cases.

Planning horizon [years]	Case 1		Case 2		Case 3		Case 4	
	2	3	2	3	2	3	2	3
Computation time exact method [s]	1.3	5.7	3.8	46.1	13.9	352.0	23.4	2271.0
Computation time GA [s]	6.1	7.0	9.2	15.7	18.4	25.2	23.2	36.5
Optimality gap [%]	0.00	0.00	0.00	0.31	0.00	0.39	0.02	0.37

For instance, given a fleet of 40 aircraft and a planning horizon of 4 years, and three hangars for C-checks, the number of decision variables is more than 5.8×10^4, and the number of constraints is about 7.6×10^5.

The four test cases are run for a planning horizon of two and three years. The objective value and computation time are compared to assess the effectiveness of the GA for small test cases. None of the test cases requires creating extra slots or schedules heavy maintenance checks later than due dates for both approaches. The outcome comparison can be seen in Table 3. The computation time for the exact method increases exponentially when the problem size increases. The GA solves the problem significantly faster for problems with a longer time horizon without compromising the quality of the solution obtained. For the biggest test case of 40 aircraft and a planning horizon of three years, the computation time difference is already more than 35 min. The optimality gap, i.e., the difference between the objective function value of the exact method and the GA, is only 0.37%. For the 2-year planning horizon cases, the gap is not higher than 0.02%, while for the 3-year case, the gaps are always below 0.4%, suggesting the efficiency of the GA.

5.2. 2018–2021 H-AMCS optimization results

To assess the effectiveness of the GA methodology, we perform a case study using a fleet of maintenance data from a major European airline (Deng, 2019). The data set contains real maintenance check information from a heterogeneous fleet of 45 aircraft from the Airbus A320 family (A319, A320, and A321). The fleet status was obtained on September 25th 2017, and the planning horizon is maximal five years. We used the GA to optimize the heavy maintenance check schedule for 2018–2021. Since heavy maintenance checks (C-/D-checks) have intervals larger than 18 months, there are at most four C-/D-checks from September 25th 2017 to December 31st 2021 for each aircraft, meaning that there are 180 decision variables for the GA.

Several operational constraints were defined to be able to replicate the test case as presented by the previous authors:

- A maximum of three C- and D-checks can be executed in parallel;
- C-check works pause on weekends and public holidays;
- Due to resource availability reasons, there has to be a minimum of three days between the start dates of two C-checks ($d_c = 3$);
- During the following commercial peak periods, no C-checks can be scheduled:
  (1) The weeks before and after Easter;
  (2) From June 1st to September 30th;
  (3) December 18th to January 7th.

Note that for the ILP model, there are at least 6.5×10^4 decision variables and 8.5×10^5 constraints according to (21), and we had memory issues running the ILP model on CPLEX. Thus, we compared the heavy maintenance check schedule created by GA to a schedule created by the maintenance planners of the airline and a schedule generated by the dynamic programming based methodology described in Deng et al. (2020). The comparison was performed only for the period between the Jan 1st 2018 and the Dec 31st 2021.

The results of the test case 2018–2021 show that the GA clearly outperforms the airline's current aircraft maintenance planning approach (Table 4). The GA reduces the total number of C-checks by 9 and increases the average flight hours by 8.4% over the planning horizon. Compared to the optimization results from a DP-based approach, GA further has three fewer C-checks and increases the average FH by 5.9%. From a saving and revenue management perspective, since airlines spend on average $150 K–$350 K on a C-check (Ackert, 2010), 9 fewer C-check can result in a potential saving of $1.35M–$3.15M for the considered time horizon. Furthermore, since a C-check lasts about 1–4 weeks in this case study, 9 reduced C-checks are equivalent to about 63–252 days of aircraft availability for commercial operations. This may generate a considerable amount of revenue for the airline.

Table 4
C-check optimization results for the test case 2018–2021.

	Airline	DP-based	GA
Average FH	6539	6691	7087
Number of C-checks	96	90	87
Computation time	≥3 days	510 s	1059 s

The computation time of the GA is over twice as the DP-based approach. However, this is still significantly better than the computation time of over three days from the airline. The improvement in aircraft utilization can be seen when looking at the distribution of flown flight hours at a C-check in Fig. 6. The distribution of FH at C-checks for the GA is shifted to the right compared to the airline and DP schedules. The number of checks scheduled near their deadline, represented as the red line, is also increased.

The assumption **A.6**, which considers that A- and B-checks can be planned at its due date, is the most significant difference between the problem addressed by Deng et al. (2020) and our approach in the paper. Therefore, we have assessed the impact of this assumption on the quality of the results obtained. To do that, the schedule created by the DP-based methodology was run in the simulator used by the GA (i.e., Algorithm 4 with a single scenario). The result was used as a benchmark against the results obtained by the DP methodology. Fig. 7 shows the difference in flight hours between the result obtained with the DP methodology and the GA simulator. The maximum difference is 12.9 flight hours. The average difference is only 1.66 flight hours, suggesting that assumption **A.6** does not have a major impact on the quality of the solutions obtained with the GA.

5.3. Robustness

The previous section concerned the analysis of the effectiveness of the GA. However, the outcome of the case study was only based on a single deterministic scenario. Fig. 6 shows that many C-checks are scheduled close to the maximum interval. A small increase in average daily flight hours or variation in check duration could disrupt the schedule and result in aircraft being maintained after their maximum interval. Therefore, we run our min–max optimization model for the same test case but considering multiple scenarios generated according to the methodology described in Section 3.4.

We considered ten scenarios (i.e., $n_s = 3$) and a probability level (α) equal to 0.8. The summary of the results is presented in Table 5. As expected, the robust optimization results in lower utilization and two more C-checks than the deterministic optimization. Using ten scenarios instead of a single scenario also increases the computation time but just by 55%. When we compared the results with the schedule obtained with the airline's current planning approach, the robust schedule is

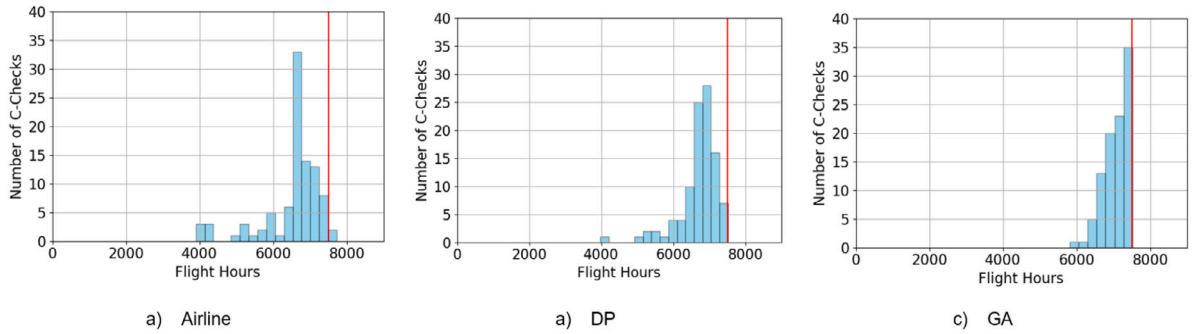


Fig. 6. Comparison of FH at C-check between schedules from the airline, the DP-based methodology, and GA.

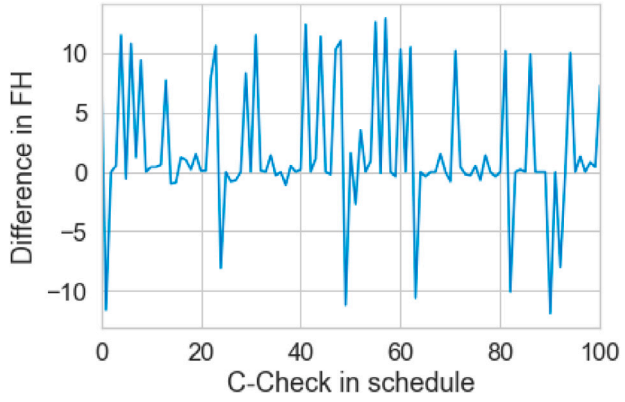


Fig. 7. Difference between the flown flight hours resulting from the GA simulator and the DP-based methodology from Deng et al. (2020), using the same schedule.

Table 5
Results of optimization for the case study with the GA, for 10 scenarios and a probability level of 80%, compared to the results of the GA for the deterministic scenario.

	Deterministic	Robust ($n_s = 3,\ \alpha = 0.8$)
Average flight hours	7087.3	6825.8
Number of C-checks	87	89
Computation time	1059 s	1643 s

even more efficient. There are seven fewer C-checks scheduled, and the average utilization increases by 4.4%. When taking into account that an airline spends on average between $70 and $350 thousand on a C-check (Ackert, 2010), this reduction could lead to an annual cost saving between $122.5 and $612.5 thousand. Furthermore, the reduced number of C-checks also results in fewer days for aircraft to be on the ground for maintenance. In this case, there could potentially result in 100 to 140 days extra in operation over the planning horizon or about 25 to 35 extra days per year for this fleet of 45 aircraft. Considering that a day of operations of a short-haul aircraft can generate between $75 to $120 thousand in revenue, this could represent $1.8 to $4.2 million of additional annual revenue.

To evaluate the robustness of the schedule obtained with the min–max optimization approach, we compare the performance of the schedule with the performance of a schedule based on a single deterministic scenario. We performed a Monte Carlo simulation analysis with 10 000 runs. In each run, the schedules were assessed with the simulator under a different maintenance duration and aircraft daily utilization matrices randomly generated using historical data and the approach described in Section 3.4.2. This way, the schedules were checked against 10 000 different scenarios to assess the number of times the schedule would become unfeasible, either because the aircraft reach the end of the interval before starting the maintenance check or because additional

Table 6
Probability for the total days aircraft are maintained past their deadline to lie within a certain range. "Days too late" refers to the number of days that aircraft already reach their utilization limit (inspection interval) and have to be grounded until they get available maintenance slots. The category "<1" includes the maintenance checks performed on time and before the corresponding aircraft reach their utilization limit.

Days too late	Deterministic	Robust ($n_s = 3,\ \alpha = 0.8$)
<1	0.27%	41.12%
1–19	25.90%	55.34%
20–49	58.92%	3.36%
≥50	14.91%	0.18%

slots are necessary to complete the maintenance events scheduled. No recovery procedures were considered in the simulation, so the schedule is not adjusted if an aircraft has to be grounded or if extra slots are needed. In practice, a schedule disruption management system can be used to avoid grounding an aircraft. Thereafter, we will call the min–max optimization schedule the robust schedule and the schedule obtained with one single scenario as the deterministic schedule.

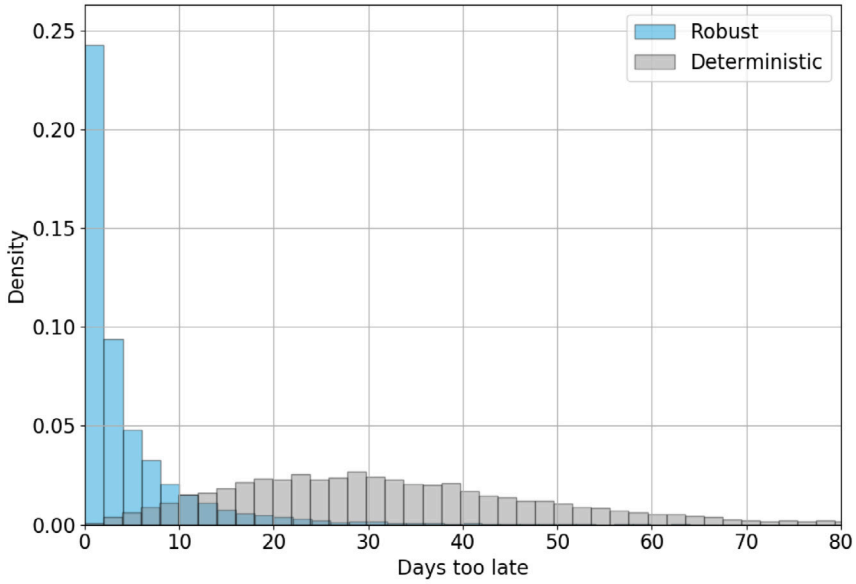
Fig. 8a shows the Monte Carlo simulation results regarding the number of days aircraft are maintained after their maximum interval due to higher daily utilization of the aircraft. This would force the aircraft to be on the ground for these days, waiting for the maintenance to be performed. The results indicate that uncertainty can significantly impact the feasibility of the schedule for the deterministic case. On average, for all the runs computed, an aircraft would have to be grounded for 32.3 days during the 4 years simulated. On the other hand, only 2.7 days would be needed for the case of the robust schedule.

Based on the distributions, the probability that the total days in which aircraft are maintained past their deadline lie within a certain range is computed. In Table 6, it is clear that the min–max optimization is significantly more robust than optimization using the deterministic scenario. There is a 41% chance that no conflicts and disruptions occur for the robust schedule, compared to a chance of 0.27% for the schedule generated by deterministic optimization.

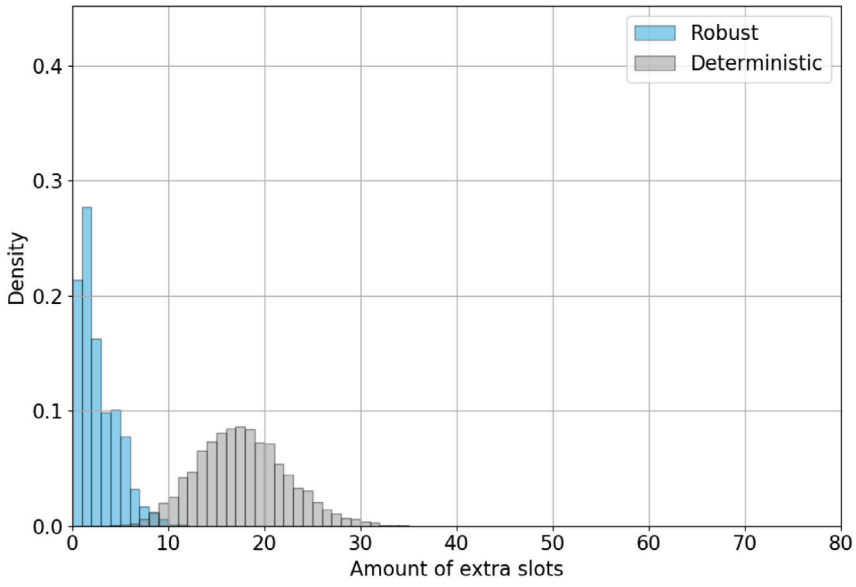
Fig. 8b shows the Monte Carlo simulation results regarding the number of extra slots required due to uncertainty in the check duration. Similar to what we observed in the previous analysis, the distribution is shifted to the left for the robust schedule. The simulations resulted in a mean of 2.1 extra slots for the robust schedule compared to 17.2 for the deterministic schedule. Table 7 shows the probabilities for the number of extra slots to lie within certain ranges, indicating that the schedule created by the min–max optimization approach is more robust than the deterministic schedule when regarding available maintenance slots.

5.4. Sensitivity analysis

We performed a sensitivity analysis to investigate the impact of the model parameters, n_s and α. To do this, we run a Monte Carlo simulation analysis solving the min–max optimization model for a four



(a)



(b)

Fig. 8. (a) Distribution of total number of days the aircraft in the fleet have to be grounded because of exceeding their maximum interval during the planning horizon for the deterministic and robust schedules from 10 000 Monte Carlo simulation runs; (b) Distribution of the number of extra slots that are necessary due to uncertainty in check duration for the deterministic and robust GA optimization, from 10 000 Monte Carlo simulation runs.

Table 7
Probability for the amount of extra slots in the planning horizon to lie within a certain range.

Extra slots	Deterministic	Robust ($n_s = 3, \alpha = 0.8$)
<1	0.02%	35.08%
1–9	5.40%	64.71%
10–19	68.30%	0.21%
≥20	26.29%	0%

years test case with 20 aircraft and 2 hangar slots available and varying these scenario generation parameters. We considered $\alpha = 0.5$, 0.65, 0.8, 0.95 and followed by a fifth case in which we combined different probability factors by randomly sampling one of these values for each simulation run. A larger probability factor would lead to more critical scenarios. A smaller factor results in selecting scenarios with aircraft utilization and maintenance check duration closer to their respective mean. For the scenario size, n_s varies between 2 and 7, corresponding to a total number of scenarios S between 5 to 50. The computation times per run varies between 5 min for $S = 5$ and about 16 min for $S = 50$, on a standard laptop computer. Since n_s means that we select the n_s matrices with the shortest/longest average check duration among the 10 000 generated elapsed time matrices and the top n_s among the 10 000 generated utilization matrices, even if n_s is large, we only minimize the worst of all $n_s \times n_s$ scenarios in the end. Therefore, it is not necessary to make n_s a large number. Besides, n_s would be determined

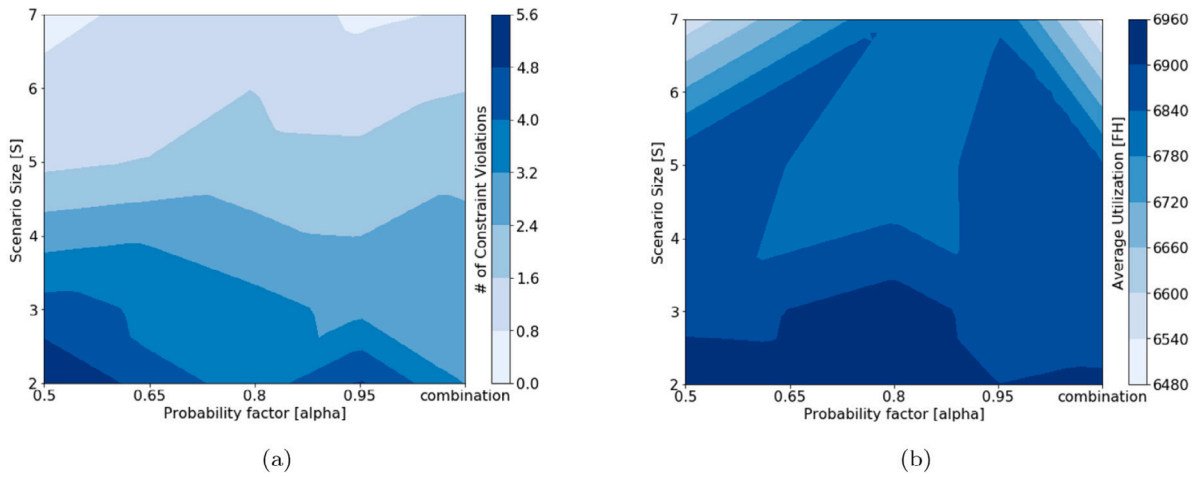


Fig. 9. Contour plots of the average number of constraint violations and flight hours for varying robust optimization parameter settings: (a) Number of constraint violations; (b) Average FH.

by the decision-makers in the airline in practice to make a trade-off between computation time and reliability of the results.

We performed 10 000 Monte Carlo simulation runs to simulate different aircraft utilization and maintenance checks duration for each combination of robust optimization parameters and stored the average number of constraint violations. A violation is considered to be one day that an aircraft has to be grounder or an extra slot that needs to be added during the planning horizon. A contour plot for this can be seen in Fig. 9a. The average utilization for each optimization run has also been recorded. The utilization is defined as the average FH an aircraft has flown at the start of scheduled C-check. A contour plot of the average utilization per robust optimization setting can be seen in Fig. 9b.

Fig. 9 shows that a larger number of scenarios reduces the average number of constraint violations, thereby making the maintenance check schedule more robust. This robustness, however, is coupled with a reduction in the average aircraft utilization. A simulation run using a single deterministic scenario with the mean aircraft utilization and check durations would result in an average of 45.8 constraint violations and average aircraft utilization of 7124 FH. From the plots, it can be seen that even with only 5 scenarios in the optimization, the average constraint violations are reduced by 90%, at the cost of a 2.5% reduction in aircraft utilization. The contour plots also show that an increasing probability factor makes the schedule more robust for a smaller number of scenarios. This is most likely, due to the higher chance of including more critical scenarios in the optimization. However, this effect is lost when the total number of scenarios increases since the scenarios can already vary a lot at the same probability level. In most cases, using scenarios generated from a combination of α and n_s tends to be more robust than from a single probability factor for a smaller number of scenarios since a combination of probability factors increases the variability in the scenarios. These conclusions are confirmed by the analysis presented in Table 8, indicating that an increase in scenario size reduces the number of constraint violations on average while decreasing the average aircraft utilization.

Note that from an application perspective, airlines have the flexibility to use interval tolerance for their fleet or create extra maintenance capacity when necessary. It is not desirable to have a robust heavy maintenance check schedule as in the case of $n_s = 7$, i.e., a schedule that results in mean utilization 6344.1 FH and constraint violation 0.87. The low aircraft utilization and low mean number of constraint violation indicate that the entire fleet has to be grounded very often for C-/D-check (a C-/D-check lasts 1–4 weeks), resulting in very low aircraft availability for commercial operations, and this also makes difficult for the operations control center (OCC) to plan the flight routes and design flight schedules. Therefore, an ideal robust heavy maintenance check schedule should lead to higher mean aircraft utilization but have as few constraint violations as possible.

Table 8
Sensitivity to a change in scenario size n_s on utilization and number of constraint violations.

Scenario size (n_s)	Mean utilization [FH]	Mean number of constraint violations
2	6873.3	4.27
3	6825.8	3.40
5	6747.2	1.82
7	6344.1	0.87

6. Conclusion

This paper is the first to address the long-term heavy aircraft maintenance check scheduling (H-AMCS) optimization considering uncertainty in aircraft maintenance check duration and aircraft daily utilization. A min–max integer linear programming (ILP) optimization approach is proposed that can consider multiple generated scenarios to compute robust and efficient heavy maintenance check schedules for a heterogeneous aircraft fleet. In fact, this is the first research that formulates the stochastic aircraft maintenance check scheduling as an ILP optimization model.

The optimization model was solved using an efficient genetic algorithm (GA). The GA was first validated for smaller test cases against an exact method. The results show that the GA can generate (near-)optimal solutions for these test cases with a maximum optimality gap of 0.39%. When the problem size increases, the GA also significantly reduces computation time compared to the exact benchmark method. In a case study for a European airline, the GA reduced the total number of C-checks by 9.4% while increasing the average aircraft utilization by 8.4%, compared with the schedule produced by the airline. However, Monte Carlo simulations showed that such a deterministic schedule was very susceptible to changes — only 0.27% of the simulation runs need no changes to keep the schedule feasible for a 4-year planning horizon. When including various scenarios in the min–max optimization approach, a more robust schedule could be obtained, in which about 40% of the cases need no changes to the schedule. The robust schedule is still significantly more efficient than the airline's current approach, reducing the total number of C-checks by 7 and increasing the average aircraft utilization by 4.4%. This could lead to a potential reduction of direct annual maintenance costs between $122.5 K and $612.5 K.

Recommendations for future work could be including other maintenance check types, namely, A-checks, in the optimization model. In that case, it would give a better estimation of the potential maintenance cost reduction. Furthermore, a schedule disruption management tool can be implemented so that the robustness of aircraft maintenance check schedules are analyzed in more detail, further reducing the need for revisions and promptly recovering the maintenance activities. Finally, one could also consider planning the maintenance tasks within each check. This would provide a more robust fleet maintenance plan, from maintenance check scheduling to task execution.

CRediT authorship contribution statement

Tim van der Weide: Methodology, Investigation, Formal analysis, Validation, Visualization, Writing – original draft. **Qichen Deng:** Conceptualization, Data curation, Investigation, Formal analysis, Supervision, Writing – review & editing. **Bruno F. Santos:** Conceptualization, Writing – review & editing, Supervision, Project administration.

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).

Ahmed, M.B., Ghroubi, W., Haouari, M., Sherali, H.D., 2017. A hybrid optimization-simulation approach for robust weekly aircraft routing and retiming. Transp. Res. C 84, 1–20.

Başdere, M., Bilge, U., 2014. Operational aircraft maintenance routing problem with remaining time consideration. European J. Oper. Res. 235 (1), 315–328.

Bauer-Stämpfli, H., 1971. Near optimal long-term scheduling of aircraft overhauls by dynamic programming. In: AGIFORS Symposium. Benalmadena, Spain.

Ben-Tal, A., Nemirovski, A., 1999. Robust solutions of uncertain linear programs. Oper. Res. Lett. 25 (1), 1–13.

Bergstra, J., Bengio, Y., 2012. Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13, 281–305.

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

Chiu, W., Chen, B., Yang, C., 2012. Robust relative location estimation in wireless sensor networks with inexact position problems. IEEE Trans. Mob. Comput. 11 (6), 935–946.

Deng, Q., 2019. Aircraft Maintenance Check Scheduling Data Set. http://dx.doi.org/10.4121/uuid:1630e6fd-9574-46e8-899e-83037c17bcef, Dataset.

Deng, Q., Santos, B.F., Curran, R., 2020. A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization. European J. Oper. Res. 281, 256–273.

Deng, Q., Santos, B.F., Verhagen, W.J.C., 2021. A novel decision support system for optimizing aircraft maintenance check schedule and task allocation. Decis. Support Syst. 146, 113545.

Dinis, D., Barbosa-Póvoa, A., Ângelo Palos Teixeira, 2019. A supporting framework for maintenance capacity planning and scheduling: Development and application in the aircraft MRO industry. Int. J. Prod. Econ. 218, 1–15.

Eck, B.J., Fusco, F., Taheri, N., 2015. Scenario generation for network optimization with uncertain demands. In: World Environmental and Water Resources Congress 2015.

Elshaer, R., 2017. Solving resource-constrained project scheduling problem using genetic algorithm. J. Al-Azhar Univ. Eng. Sect. 12 (42), 187–198.

Eltoukhy, A.E.E., Wang, Z.X., Chan, F.T.S., 2018. Joint optimization using a leader-follower Stackelberg game for coordinated configuration of stochastic operational aircraft maintenance routing and maintenance staffing. Comput. Ind. Eng. 125, 46–68.

Eltoukhy, A.E.E., Wang, Z.X., Chan, F.T.S., Chung, S.H., Ma, H.-L., Wang, X.P., 2020. Robust aircraft maintenance routing problem using a turn-around time reduction approach. IEEE Trans. Syst. Man Cybern.: Syst. 50, 4919–4932.

Etschmaier, M., Franke, P., 1969. Long-term scheduling of aircraft overhauls. In: AGIFORS Symposium. Broadway, Great Britain.

Goldberg, D.E., Deb, K., 1991. A comparative analysis of selection schemes used in genetic algorithms. Found. Genet. Algorithms 1, 69–93.

Gopalan, R., Talluri, K.T., 1998. The aircraft maintenance routing problem. Oper. Res. 46 (2), 260–271.

Hartmann, S., 1998. A competitive genetic algorithm for resource-constrained project scheduling. Nav. Res. Logist. 45, 733–750.

Holland, J.H., 1992. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. The MIT Press, Cambridge, Massachusetts, USA.

Hu, X.-B., Paolo, E.D., 2009. An efficient genetic algorithm with uniform crossover for air traffic control. Comput. Oper. Res. 36, 245–259.

Huang, C., Li, F., Jin, Z., 2015. Maximum power point tracking strategy for large-scale wind generation systems considering wind turbine dynamics. IEEE Trans. Ind. Electron. 62 (4), 2530–2539.

IATA's Maintenance Cost Task Force, 2019. Airline Maintenance Cost Executive Commentary Edition 2019. https://www.iata.org/contentassets/bf8ca67c8bcd4358b3d004b0d6d0916f/mctg-fy2018-report-public.pdf. (Accessed on September 11, 2020).

IFS, 2019. IFS MAINTENIX: Aircraft Fleet Management Software – A New Approach to Maintenance Planning. https://www.ifs.com/corp/solutions/ifs-maintenix/fleet-planner/. (Accessed on October 4, 2020).

Kleeman, M.P., Lamont, G.B., 2005. Solving the aircraft engine maintenance scheduling problem using a multi-objective evolutionary algorithm. In: Evolutionary Multi-Criterion Optimization. Springer, pp. 782–796.

Kouvelis, P., Yu, G., 1997. Robust Discrete Optimization and its Applications. Springer, Boston, MA.

Kozanidis, G., Skipis, A., 2010. Flight and maintenance planning of military aircraft for maximum fleet availability. Mil. Oper. Res. 15 (1).

Kramer, O., 2017. Genetic Algorithm Essentials. Springer.

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

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. Transp. Res. B 78, 238–259.

Lin, Z., Lin, M., Wang, J., Huang, Y., Zhu, W., 2018. Robust secure beamforming for 5G cellular networks coexisting with satellite networks. IEEE J. Sel. Areas Commun. 36 (4), 932–945.

Ma, J., Qin, J., Salsbury, T., Xu, P., 2012. Demand reduction in building energy systems based on economic model predictive control. Chem. Eng. Sci. 67 (1), 92–100, Dynamics, Control and Optimization of Energy Systems.

Mahalanobis, P.C., 1936. On the generalized distance in statistics. http://bayes.acs.unt.edu:8083/BayesContent/class/Jon/MiscDocs/1936_Mahalanobis.pdf. (Accessed on October 4, 2020).

Maher, S.J., Desaulniers, G., Soumis, F., 2014. Recoverable robust single day aircraft maintenance routing problem. Comput. Oper. Res. 51, 130–145.

Mattila, V., Virtanen, K., 2011. Scheduling fighter aircraft maintenance with reinforcement learning. In: 2011 Winter Simulation Conference (WSC). pp. 2540–2551.

Meurer, A., Smith, C.P., Paprocki, M., Certik, O., Kirpichev, S.B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J.K., Singh, S., Rathnayake, T., Vig, S., Granger, B.E., Muller, R.P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M.J., Terrel, A.R., Roucka, S., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., Scopatz, A., 2017. SymPy: symbolic computing in Python. PeerJ Comput. Sci. 3, e103.

Papadakos, N., 2009. Integrated airline scheduling. Comput. Oper. Res. 36, 176–195.

Quan, G., Greenwood, G.W., Liu, D., Hu, S., 2007. Searching for multiobjective preventive maintenance schedules: Combining preferences with evolutionary algorithms. European J. Oper. Res. 177, 1969–1984.

Rowe, J.E., 2015. Genetic algorithms. In: Springer Handbook of Computational Intelligence. Springer Berlin Heidelberg, pp. 825–844.

Samaranayake, P., 2006. Current practices and problem areas in aircraft maintenance planning and scheduling–Interfaced/integrated system perspective. In: Proceedings of the 7Th Asia Pacific Industrial Engineering and Management Systems Conference.

Sohn, S.Y., Yoon, K.B., 2010. Dynamic preventive maintenance scheduling of the modules of fighter aircraft based on random effects regression model. J. Oper. Res. Soc. 61, 974–979.

Soylu, B., 2015. A general variable neighborhood search heuristic for multiple traveling salesmen problem. Comput. Ind. Eng. 90, 390–401.

Sriram, C., Haghani, A., 2003. An optimization model for aircraft maintenance scheduling and re-assignment. Transp. Res. A 37 (1), 29–48.

Sudholt, D., 2015. Parallel evolutionary algorithms. In: Springer Handbook of Computational Intelligence. Springer, pp. 929–959.

Venkata Narasimha, K., Kivelevitch, E., Sharma, B., Kumar, M., 2013. An ant colony optimization technique for solving min-max Multi-Depot Vehicle Routing Problem. Swarm Evol. Comput. 13, 63–73.

Vershynin, R., 2012. How close is the sample covariance matrix to the actual covariance matrix? J. Theoret. Probab. 25 (3), 655–686.

van der Weide, T.M.J., 2020. Long-Term C-Check Scheduling for a Fleet of Heterogeneous Aircraft under Uncertainty (Master Thesis). http://resolver.tudelft.nl/uuid:db953324-f70d-4c2f-a1a7-3c9d318df4f8.

Witteman, M., Deng, Q., Santos, B.F., 2021. A bin packing approach to solve the aircraft maintenance task allocation problem. European J. Oper. Res. 294, 365–376.

Yan, S., Chen, C.-Y., Yuan, J.-H., 2008. Long-term aircraft maintenance scheduling for an aircraft maintenance centre: A case study. Int. J. Appl. Manage. Sci. 1 (2), 143–159.

Yang, Z., Yang, G., 2012. Optimization of aircraft maintenance plan based on genetic algorithm. Physics Procedia 33, 580–586.

Zamani, R., 2010. An accelerating two-layer anchor search with application to the resource-constrained project scheduling problem. IEEE Trans. Evol. Comput. 14, 975–984.