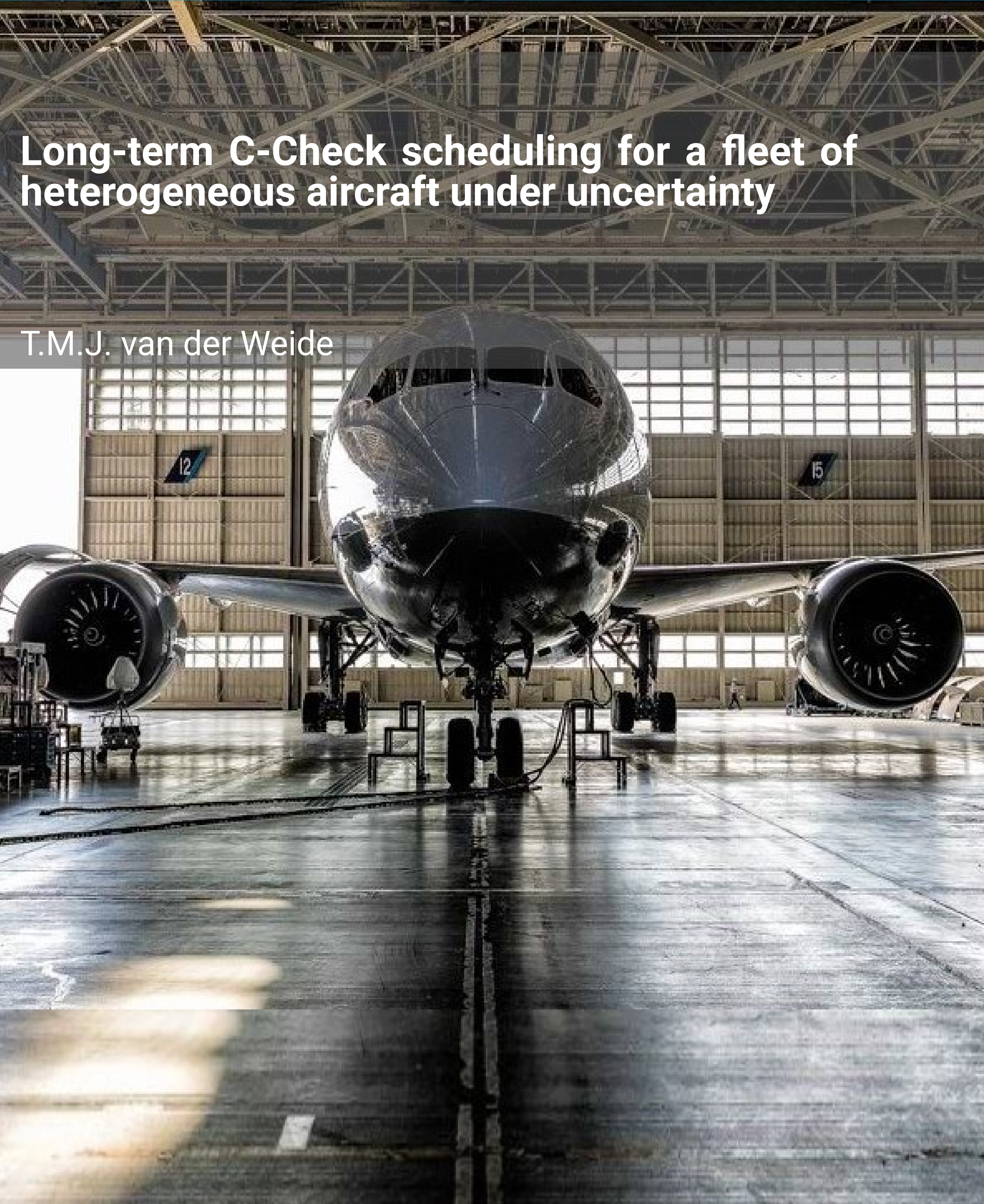
Long-term C-Check scheduling for a fleet of heterogeneous aircraft under uncertainty

T.M.J. van der Weide



TU Delft

[1] Onemoreweektogo (2018, Mar 1). Retrieved from https://www.onemoreweektogo.com/ana-maintenance-hangar-tour-haneda

Long-term C-Check scheduling for a fleet of heterogeneous aircraft under uncertainty

by

T.M.J. van der Weide

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on the 21st of July 2020

Student number:	4291824	
Project duration:	September, 2019 – June, 2020	
Thesis committee:	Dr. ir. B. F. Santos,	TU Delft, supervisor
	Q. Deng,	TU Delft, supervisor
	Ir. P. C. Roling,	TU Delft
	Dr. D. Ragni,	TU Delft

TU Delft

Preface

This report is the result of my thesis research and marks the end of my MSc in Aerospace Engineering at Delft University of Technology. This research will explore the scheduling of heavy-maintenance for aircraft under uncertainty with the use of meta-heuristics. The project combines my interest in operations research with my interest in aircraft maintenance. It was a great experience to apply the knowledge I gained during my study and further explore the field of scheduling optimization. Although challenging at times, I really enjoyed working on this complex problem for an extended period of time.

My overall studies have been very enjoyable and have allowed me to not only develop my academic skills but my interpersonal skills as well. I have met a great deal of bright people from different backgrounds and nationalities. Working with such a diverse group of people has been a pleasure. My time at the university is a period of my life I will look back on fondly.

I would like to express my gratitude to my supervisors, Bruno and Qichen, for getting me started and giving useful tips and feedback on how to proceed with the problem. A special thanks goes to Qichen for allowing me to cross-validate my model with his dynamic programming methodology. Thank you to my girlfriend Myrthe, not only for proofreading my thesis but also for supporting me throughout the project. Lastly, I want to thank my parents for making my studies possible and helping me pursue my goals in life. I would not be where I am today without your support.

Delft, The Netherlands
July 6, 2020

T.M.J. van der Weide

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

ACO	Ant Colony Optimization
AD	Airworthiness Directive
$AMOS$	Aircraft Maintenance Operations Simulation
DP	Dynamic Programming
GA	Genetic Algorithm
HMV	Heavy Maintenance Visit
LP	Linear Programming
MDP	Markov Decision Process
$MILP$	Mixed-Integer Linear Programming
MPD	Maintenance Planning Document
$MRCPSP$	Multi-mode Resource-Constrained Project Scheduling Problem
MRO	Maintenance, Repair, and Overhaul
MSG	Maintenance Steering Group
$MTBF$	Mean Time Between Failure
$MTTR$	Mean Time To Repair
NP	Non-Deterministic Polynomial-Time
$RCPSP$	Resource-Constrained Project Scheduling Problem
SA	Simulated Annealing
SB	Service Bulletin
TS	Tabu Search
TSN	Time-Space Network

Greek Symbols

α	Probability level for selection of duration and daily utilization matrix for scenarios
χ^2	Chi-squared distribution
ϵ	Probability that an aircraft is chosen as candidate for scheduling a C-check when usage parameters are not yet at their maximum interval

μ	Mean vector of multivariate normal distribution
Σ	Covariance matrix
σ	Standard deviation
Σ_x	Sample covariance matrix

Roman Symbols

C_E	Cost for having an extra maintenance slot
C_T	Cost for scheduling an aircraft a day after its deadline
$d_{i,j,S_n}(t)$	Duration of check j for aircraft i under scenario S_n if check starts at time t
$D_{i,t}$	Number of calendar days since last D-check for aircraft i at time t
$DY_{i,t}$	Number of calendar days since last C-check for aircraft i at time t
E_t	Extra slot at time t to increase maintenance capacity
I	Set of all aircraft in the fleet
I_i^{DY}	Maximum C-check interval expressed in calendar days for aircraft i
I_i^{D}	Maximum D-check interval expressed in calendar days for aircraft i
I_i^{FH}	Maximum C-check interval expressed in flight hours for aircraft i
J_i	Set of possible C-checks for aircraft i within planning horizon
L_T	Available maintenance slots at time t
M	Big number
$m_{i,t}^{C}$	Binary variable that is 1 if aircraft i is being maintained for a C-check during time t; 0 otherwise
$m_{i,t}^{D}$	Binary variable that is 1 if aircraft i is being maintained for a D-check during time t; 0 otherwise
$n\_generations$	Number of iterations without a change in the best solution after which the algorithm is terminated
$n_{i,j}$	Binary variable that is 1 if check j for aircraft i is a D-check; 0 otherwise
$P_{i,j}(t)$	Penalty for aircraft i if check j is planned after the deadline
Q	Unseen point in multivariate normal distribution
S	Number of duration matrices and utilization matrices that are selected for robust optimization
S_n	Specific scenario in set of scenarios
SC	Set of all scenarios input in robust optimization
T	Set of time steps in planning horizon

U	Daily utilization vector
u	Distance of point on ellipsoid of multivariate normal distribution
U_{i,t,S_n}	Daily flight hours for aircraft i at time t under scenario S_n
$x_{i,j,t}$	Binary variable that is 1 if check j is planned at time t for aircraft i; 0 otherwise
$y_{i,t}$	Flown flight hours since last C-check for aircraft i at time t
$Z_{i,j,t}$	Semi-continuous variable that equals $y_{i,t}$ if $x_{i,j,t} = 1$; 0 otherwise

I

Paper

*Still has to be graded

Long-term C-Check scheduling for a fleet of heterogeneous aircraft under uncertainty

T.M.J. van der Weide

Supervisors: dr. ir. B.F. Santos, Q.Deng
Section Air, Transport and Operations, Department Control and Operations, Faculty of Aerospace Engineering
Delft University of Technology, Delft, The Netherlands

The MRO market currently spans around 9.5% of the total operating cost of an airline. Of this, 70% is covered by heavy-maintenance. Reduction of these costs and efficiency could, therefore, be significant for an airline. A possible solution is the optimization of the long-term schedule of heavy-maintenance checks. Current approaches are found to be reliant on manual input and operator experience. Next to that, revisions to the initial schedule are made continuously due to the inherently stochastic nature of aircraft maintenance through non-routine maintenance. Taking this uncertainty into account could offer more robust schedules, saving cost and improve quality of service. A genetic algorithm methodology that can generate robust and efficient C-check schedules for a fleet of heterogeneous aircraft is proposed. Uncertainty in check duration and daily utilization are taken into account by assessing multiple scenarios through min-max optimization. This paper is the first to address the long-term scheduling of heavy-maintenance checks while taking uncertainty in check duration and daily utilization into account. The proposed genetic algorithm finds robust and efficient C-check schedules for a case study of a European airline for a fleet of over 40 aircraft in under 30 minutes. The algorithm reduces the total number of C-checks by 7% while increasing utilization by 4.4%. This could lead to a reduction of direct annual maintenance costs of \$122.5K - \$612.5K and an additional \$2.3M - \$4.9M in annual revenue due to the increased availability of aircraft. Monte Carlo simulations show that with a probability of 41% no adjustments to the schedule are necessary to maintain all aircraft on time.

Index Terms—**Aircraft Maintenance, Genetic Algorithm, Min-Max Optimization, Scheduling, Uncertainty**

I. INTRODUCTION

UNTIL recently, the size of the global commercial aircraft fleet was estimated to grow from 27,854 to 48,540 by 2037, [1], indicating that the global aircraft industry is expanding fast. However, with the impact of the COVID-19 pandemic, the aircraft industry is heavily affected. The expectation is that the global fleet size will have recovered to a size of 27,001 by the end of 2021 [2]. The MRO market has to react to the changes in demand accordingly. Since the global MRO spend spans on average 9% to 10% of the total operating costs of an airline [3], it is beneficial for airlines to keep introducing innovations concerning the scheduling of maintenance and to value efficiency. This can reduce maintenance cost and is important in the short-term due to the impact of COVID-19 as well as in the long-term to cope with an augmenting demand on maintenance.

Regular aircraft maintenance is necessary to assure airworthiness, to keep the aircraft reliable, and to provide assurance of flight safety. Currently, maintenance is divided into three noteworthy maintenance checks: A-, C-, and D-Checks or so called letter checks. Each check has a different duration, frequency, and set of tasks associated [4]. The C- and D-Checks are the largest check types, labelled as heavy-maintenance checks. During these checks the aircraft is not operational and kept on the ground for several weeks. Therefore, heavy-maintenance covers around 70% of the total maintenance costs and requires the most amount of resources [3]. The frequency of a letter check depends on the maximum interval between two checks and can differ between aircraft types. If an aircraft has flown a certain amount of flight hours, flight cycles, or a certain amount of days have elapsed since the last check, a new check has to be performed to keep the aircraft airworthy. The deadline of a maintenance check for an aircraft is thus dependent on the date of its last check, utilization, and maximum interval. Therefore, generating a maintenance schedule becomes a combinatorial optimization problem.

Current approaches to the long-term scheduling of the maintenance checks are often based on operator experience and simulation models where manual selection is a major component. Next to that, the initial schedule has to be continuously revised due to the stochastic nature of aircraft maintenance. These revisions can result in backlog and can have an impact on maintenance cost and quality of service. Taking this into account when scheduling could offer more robust schedules where fewer revisions are necessary. Developing an efficient long-term schedule of these heavy-maintenance checks, while taking into account uncertainty, could therefore not only result in significant cost savings, it could also reduce inefficiencies in the execution of the schedule.

This paper proposes a genetic algorithm that can generate efficient robust schedules based on a min-max scenario optimization approach and is the first to address the long-term scheduling of heavy-maintenance checks while taking uncertainty in check duration and daily utilization into account.

In Section II, a brief overview of the current approach to maintenance scheduling, the state-of-the-art, and the main uncertainties in scheduling maintenance checks is presented. Section III introduces the formulation for the problem of long-term maintenance check scheduling under uncertainty as well as the methodology for generating scenarios used for min-max optimization. The proposed genetic algorithm is discussed in Section IV. The algorithm is validated in Section V by comparison with a benchmark model and a case study. The results are discussed in Section VI. Lastly, conclusions are drawn in Section VII together with recommendations for future work.

II. Background

The complexity of large size scheduling problems has been discussed in Papadimitriou and Steiglitz [5]. The problem is classified as non-deterministic in polynomial-time (NP) hard, making it difficult to solve large scale problems with exact methods. The challenge in the scheduling of aircraft letter checks especially is the fact that they are interdependent combinatorial optimization problems. Scheduling a check at a certain date has an impact on aircraft utilization in the future and consequently influences the requirements on future maintenance checks.

A. Current Approach

Since it is difficult to solve the problem with exact methods, the long-term scheduling of aircraft heavy-maintenance checks has been dependent on a manual scheduling approach, which relies mainly on the experience of a scheduler. In the early '70s, using this manual approach, it took maintenance planning personnel several weeks to create a schedule [6]. To speed up this process Air Canada developed an aircraft maintenance operations simulation model (AMOS). This simulation model is described in Boere [6] and focuses on improving maintenance efficiency and reducing labour and material cost. In the paper, the scheduling problem is described as a discrete integer programming problem and it takes many of the constraints and conditions listed above into account. Despite the integer programming formulation, the solution approach is a priority-based simulation-heuristic. This approach is similar to the manual planning approach, in which an experienced scheduler has to decide the optimal schedule, shifting checks until a feasible solution has been found. The simulation aspect, however, together with the introduction of a lower utilization bound before a new check can be scheduled, reduced the time to develop a long-term maintenance schedule of 5 years from several weeks to several hours [6].

Many operators have since developed or adopted a similar approach to the scheduling of long-term maintenance and more overall integrated tools are being developed, like the fleet-planner IFS Maintenix tool [7]. All the tools, however, are heavily reliant on experience and manual input. In addition to the reliance on a manual approach, the developed schedules do not consider the uncertainty associated with the long-term scheduling and thus need to be revised frequently.

B. State-of-the-Art

In academic literature, there has been relatively little focus on the long-term planning of aircraft maintenance. In Etschmaier and Franke [8], an out-of-kilter algorithm [9] was introduced to minimize maintenance cost and in Bauer-Stämpfli [10] a dynamic programming approach was developed. Both methods were deemed not suitable by Boere [6] for the environment of Air Canada when developing the previously mentioned simulation model. Furthermore, the author deemed an optimization technique to be impractical because an optimal solution can become obsolete with a change in environment and aircraft utilization. More recently, a zero-one integer programming model was used with the commercial solver CPLEX in Yan et al.[11]. Deng et al. [12] propose a forward induction dynamic programming approach. To deal with the problem of a multi-dimensional action vector, the author proposed a priority-based approach to sort the list of aircraft. A trifty algorithm, combined with discretization and state aggregation is used to solve the resulting model.

Since relatively little literature exists on the long-term scheduling of aircraft maintenance, it is relevant to look into short-term aircraft maintenance scheduling problems and more general scheduling literature. The short-term scheduling of aircraft maintenance has received considerably more attention in literature through the aircraft maintenance-routing problem. Several varying objectives can be distinguished. The minimization of total maintenance cost is encountered in Sriram and Haghani [13], where the problem of scheduling maintenance is formulated as an integer multi-commodity network flow model with the objective to minimize total costs of type A and type B maintenance checks. In Kozanidis and Skipis [14], a mixed-integer bi-objective linear programming model is used to maximize aircraft availability and residual flight time for a fleet of fighter aircraft over the considered time-horizon. Lastly, the minimization of aircraft unavailability is encountered in Başdere and Bilge [15]. This objective is very similar to the maximization of the utilisation of maintenance intervals, since it indirectly decreases the number of maintenance checks in the long-term and number of days on the ground.

Since the scheduling of aircraft maintenance checks is NP-hard large scale problems are difficult to solve with exact methods. Therefore, the primary solution techniques found in literature are meta-heuristics. Of which an often encountered example in scheduling literature is the Genetic Algorithm (GA). The concept of a GA was first introduced in Holland [16], it is an adaptive method based on the genetic processes of biological organisms. More specifically, it is based on the theory of evolution, in which populations evolve over many generations based on survival of the fittest and natural selection. In Yang and Yang [17], a Genetic Algorithm is used to schedule maintenance opportunities based on an original flight plan. The GA is very basic, but a simulation experiment shows the possibility of the algorithm to come up with feasible maintenance schedules while minimizing the costs. Quan et al. [18] use a more complex genetic algorithm to address a multi-objective preventive maintenance scheduling problem. Kleeman and Lamont [19] also address a multi-objective problem, that of scheduling heavy-maintenance on aircraft engines. Experimental results show the possibility of a GA to efficiently solve the scheduling of maintenance while obtaining a "good" solution. One of the main downsides of the GA however, is the great computational effort that is required when the problem gets more complex. When considering the more general Resource-Constrained Project Scheduling Problem (RCPSP), Elshaer [20] compares several GAs introduced in literature for solving the RCPSP. The main research direction in the last years is the hybridization of metaheuristics which can be seen in Zamani [21], where cross-over operations from GAs are combined with local search methods. The main difference with a general GA is the fact that it concentrates on the continuous improvement of one good solution.

C. *Uncertainty*

When considering the scheduling of aircraft maintenance in literature, most cases consider the deterministic problem. However, Aircraft heavy-maintenance is a complex dynamic project and possesses varying degrees of uncertainty, which can influence the execution and creation of a maintenance schedule. For the long-term planning of aircraft heavy-maintenance, it is therefore important to identify these uncertainties and their effect on the maintenance schedule. Samaranayake [22] recognizes that uncertainty encountered in aircraft maintenance, coming particularly from non-routine and unscheduled maintenance, affects the planning and scheduling of aircraft maintenance. The author realises the importance of non-routine findings and states that about 50% to 60% of the maintenance workload result from these findings. A more recent case study in Dinis et al. [23] shows that this number can be even higher and increases with aircraft age. The uncertainty in the workload introduces an uncertainty in the duration of maintenance checks. This can hamper accurate planning and resource forecast and result in continuous adjustments to the initial maintenance schedule. Furthermore, as indicated in Section. I, the deadline of maintenance checks also depends on the utilization of an aircraft. Small changes in the amount of flight hours or flight cycles can already have a big impact on the deadlines of checks. Resulting in the need for revisions. This can result in backlog which can affect maintenance cost and the quality of service. Therefore, it would be beneficial to look into the stochastic problem to generate more robust schedules.

Currently, to the best of the author's knowledge, there is no literature available that addresses the long-term scheduling of aircraft maintenance checks while taking uncertainty into account. However, when considering the general problem of aircraft maintenance scheduling, some papers exist that do include different uncertainties. Mattila and Virtanen [24] take into account uncertainty in failure rates and maintenance duration when scheduling maintenance for a fleet of fighter jet. The uncertainty parameters are modelled 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 [25] use 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 more and more focused 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 literature are meta-heuristics. More specifically, an often encountered solution method is the Genetic Algorithm, which provides promising results for similar scheduling problems. From the literature, it is also clear that there has been little focus on the long-term scheduling of heavy-maintenance checks. Next to that, the inherently stochastic nature of maintenance is not taken into account when creating an initial schedule. A robust long-term schedule could reduce the number of adjustments to the schedule, reduce cost, and improve quality of service. Especially since, according to IATA [3], the MRO spend spans on average 9.5 to 11% 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 have a significant impact on 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 relevance of the research lies in filling this gap in literature by approaching the problem with a genetic algorithm and creating a general robust scheduling model framework that takes uncertainty into account.

III. Problem Formulation

This section describes the formal MILP formulation for the problem of scheduling C-checks for a fleet of heterogeneous aircraft under uncertainty. Section III-A introduces the C-check maintenance problem. Next, a list of assumptions for the problem is defined in Section III-B, followed by a description of how robust schedules are achieved by using multiple scenarios in Section III-C. Section III-D describes how these scenarios are selected, after which, Section III-E describes the mathematical notation used in this paper. Next, the objective function is described in Section III-F. Lastly, the constraints for the problem are formulated in Section III-G.

A. *C-check Maintenance Planning*

Currently, the planning of aircraft maintenance follows the MSG-3 approach [26]. This is a task-oriented approach that develops the maintenance tasks that need to be performed, together with their corresponding intervals. These intervals are defined in number of flight hours, amount of flight cycles, or calendar days. These tasks are noted in the Maintenance Planning Document. This document gives the threshold interval per required task and can be used to group tasks into task packages and letter checks. For the C-check this results in a cycle of C-checks with slightly varying tasks. For some checks task of the D-check can be incorporated. An example for an airline can be a cycle of 12 check types (C1,C2,. . .,C12), where every 3 cycles tasks of the D-check are incorporated. In aircraft maintenance, aircraft age by flight hours, flight cycles, and calendar days. The flight hours are increased by the elapsed time between wheel lift-off and touchdown, the flight cycles by every complete landing and take-off sequence, and calendar days by every full 24 hour period. If these usage parameters for an aircraft reach the threshold interval a C-check needs to be performed to keep the aircraft airworthy. After a check, the usage parameters are all reset to zero. During the duration of a C-check, an aircraft is not in operation and occupies a hangar. The amount of hangar space available is divided into hangar slots. Where 3 slots indicate that 3 C-checks can be performed in parallel. A long-term schedule for planning C-checks is generally created for a period of 4 to 5 years.

B. Assumptions

Deng et al. [12] adapt and describe several major conditions and assumptions, necessary for maintenance scheduling, based on Boere [6] and maintenance practice. Five assumptions (A.1, . . ., A.5) are adopted from the paper and two assumptions are added (A.6, A.7).

A.1 A C-Check ties up one hangar slot for the entire duration of the check;
A.2 An aircraft ages by 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 schedule is 1 calendar day;
A.4 Check duration can be estimated from historical data;
A.5 Location of a hangar does not influence check possibility and aircraft routing is flexible;
A.6 A-Check duration is 1 calendar day and is planned at its due date without considering resource constraints;
A.7 Additional hangar slots can be added to make a schedule feasible

Assumption A.2 is slightly adapted by assuming that only flight hours and calendar days are limiting. Flight cycles are assumed to never be limiting in practice for long-haul aircraft types. Assumption A.6 is added to take the A-check into account. If an A-check is performed on an aircraft it is taken out of operations for one day. This influences the usage parameters for the C-check, since no flight hours or flight cycles are flown on that day. It is assumed that the A-check is scheduled at its due date instead of a feasible opportunity. Therefore, the problem considers slightly less A-checks than would need to be scheduled in practice. However, this only causes the due date of the C-check to be slightly earlier than in practice and will thus still provide a feasible C-check schedule. Assumption A.7 ensures that if an aircraft reached its maximum interval for a C-check, while there are no available slots, an additional slot can be created by the MRO. This in order to avoid grounding the aircraft. It replaces the concept of using tolerance as used in Deng et al. [12].

C. Robust Optimization

This paper considers creating a framework that can deliver robust schedules for the C-check by taking uncertainty into account in optimization. To obtain robust solutions, the problem is formulated as a min-max optimization problem. Uncertainty is incorporated by considering multiple different scenarios and optimizing for the worst-case scenario. Each scenario contains different realizations of the main sources of uncertainty, being check duration and aircraft utilization. Equation 1 shows that for a predefined set of scenarios (SC) the objective is minimized against the worst-case scenario found through Equation 2. Section III-D describes how different scenarios are selected for the problem.

$$\min f^R_{obj}(x, SC) \tag{1}$$

$$f^R_{obj}(x, SC) = \max_{n \in SC} f_{obj}(x, S_n) \tag{2}$$

D. Scenario generation

To find robust schedules it is important to input varying scenarios that incorporate uncertainty in maintenance scheduling. These scenarios should be (near-)critical so that the solution will still be feasible for most unseen scenarios. From the state-of-the-art review in Section II-B, it is clear that the main sources of uncertainty come from non-routine and unscheduled maintenance manifested in the duration of the maintenance check. Next to that, daily utilization of aircraft is also a main source of uncertainty when scheduling C-checks. A scenario consists therefore out of realizations of future check duration and a daily utilization matrix, giving the flight hours per aircraft per month.

1) Check Duration

Non-routine maintenance can have a severe impact on the duration of a C-check. This paper makes the assumption that check duration can be estimated from historical data. For the scenarios (near-)critical cases are considered. In the case of check duration a critical case occurs when the check duration is longer than initially planned when creating a schedule. This is caused by possible capacity problems due to a longer check duration. However, longer check durations are not the only critical cases that can be encountered. If a check is shorter than initially expected, the aircraft comes into operation earlier. This has a direct effect on the deadline for the next C-check which will be at an earlier date. Shorter check durations can thus lead to future checks occurring too late in the initial schedule. For the scenarios, cases with a shorter check duration and cases with a longer duration than expected are therefore incorporated.

As indicated in Section III-A, a schedule is generally created for a period of 4 to 5 years, which is equal to a maximum of 4 C-checks during the planning horizon. Therefore, a scenario consists out of N x 4 check durations, where N is the amount of aircraft in the fleet. To identify which duration matrices should be selected, a Monte Carlo simulation is performed for a number of 10000 runs. Every run an instance from a list of historical durations is randomly chosen and assigned to the corresponding N x 4 duration matrix. This is done according to which type of C-check a future check in the matrix would be. For aircraft 1, the four future checks could be the C.2, C.3, C.4, and C.5 check for example. The first instance in the matrix would then be randomly chosen from the C.2 historical list, the second instance from the C.3 list, and so on. The Monte Carlo Simulation gives 10000 different duration matrices. Figure 1 shows an example of the distribution of the average check duration for all matrices.

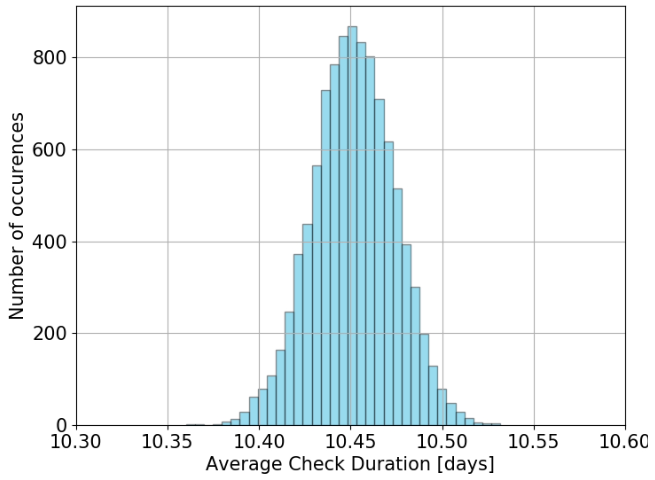


Fig. 1. Distribution of the average check duration from a Monte Carlo simulation with n = 10000 runs.

By looking at the average check duration, matrices that have check durations that are mostly shorter than the mean and matrices with durations mostly longer than the mean can be selected for the problem. This is done by generating a confidence interval $(d - c \cdot \sigma_D, d + c \cdot \sigma_D)$ for the histogram, where c is based on a predefined confidence level α. From this confidence interval, S duration matrices are selected with an average check duration corresponding to the upper or lower bound of the α confidence interval. This way, the minimum and maximum cases within the confidence interval can be taken into account. Since the durations are in the form of a N x 4 matrix, matrices with the same average check duration can also still vary significantly from each other.

2) *Utilization*

Aircraft daily utilization is defined by the average flight hours per day per month. A daily utilization matrix is therefore in the form of Table I. Being a N x 12 matrix, where N is the number of aircraft in the fleet. Critical cases occur when the number of flight hours is more than initially expected. This causes the aircraft to reach the maximum interval earlier and could result in checks being scheduled to late in the initial schedule.

TABLE I
UTILIZATION MATRIX, GIVING THE AVERAGE DAILY FLIGHT HOURS PER AIRCRAFT PER MONTH

	Jan	**Feb**	...	**Dec**
A/C 1	10.3	9.9	...	11.1
A/C 2	9.4	9.3	...	9.9
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
A/C n	11.3	11.1	...	11.2

For each aircraft and month, the mean and standard deviation are known based on historical data. Resulting in an utilization vector, U, of N x 12 random variables $(U_{1,Jan}, \dots, U_{1,Dec}, U_{2,Jan}, \dots, U_{n,Dec})$. It is assumed that these random variables are jointly normal and can therefore be modelled in the form of a multivariate normal distribution. The multivariate normal distribution, U $\sim N(\mu, \Sigma)$, has mean μ and covariance matrix Σ as seen in Equation 3 and 4.

$$\mu = \begin{bmatrix} \mu_{1.Jan} \\ \vdots \\ \mu_{1.Dec} \\ \mu_{2.Jan} \\ \vdots \\ \mu_{n.Dec} \end{bmatrix} \tag{3}$$

$$\Sigma = \begin{bmatrix} \sigma_{1.Jan,1.Jan} & \sigma_{1.Jan,1.Feb} & \cdots & \sigma_{1.Jan,n.Dec} \\ \sigma_{1.Feb,1.Jan} & \sigma_{1.Feb,1.Feb} & \cdots & \sigma_{1.Feb,n.Dec} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n.Dec,1.Jan} & \sigma_{n.Dec,1.Feb} & \cdots & \sigma_{n.Dec,n.Dec} \end{bmatrix} \tag{4}$$

The covariance matrix is an essential part in modeling the daily utilization matrix. In this paper the covariance matrix is estimated by the sample covariance matrix as in Equation 5, with a sample of $X = 10000$ independent points. It is recommended that a MRO does more research on the estimation of the covariance matrix and underlying relations. The reader is referred to Vershynin [27] for more details and issues with covariance estimation in high dimensional space.

$$\Sigma_x = \frac{1}{X} \sum_{i=1}^{X} U_i \otimes U_i \tag{5}$$

With given mean vector, μ, and sample covariance matrix, Σ_x, the aircraft utilization can be described. For robust optimization purposes it is necessary to select matrices which are consistent with the given mean vector and covariance matrix and enclose a given proportion of the sample space. Eck et al. [28] define 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 [29]. The fixed probability level α is the same α that is used for the confidence interval in Section III-D1.

At a chosen probability level there are, however, infinitely many utilization matrices with the corresponding Mahalanobis distance. In order to select daily utilization matrices, n=100 points are located corresponding to α. With regards to utilization, the critical cases are those with the highest average daily flight hours, since a higher daily utilization could mean that checks are scheduled too late. Therefore, for size S, the daily utilization matrices that are selected are those with the highest overall average daily flight hours from the n=100 points.

3) *Scenarios*

As described before, a scenario consists out of a duration matrix and a daily utilization matrix. Given size S and probability level α, S duration matrices and S daily utilization matrices are selected. By combining all possible duration matrices with all daily utilization matrices, $S \times S$ scenarios are generated. On top of the generated scenarios the algorithm also considers the deterministic scenario, as expected by the MRO. A size of $S = 2$ therefore results in 2x2 + 1 = 5 scenarios, and $S = 3$ already considers 10 scenarios.

E. *Mathematical Notation*

The mathematical notation used for this problem is described in Table II.

TABLE II
MATHEMATICAL NOTATION

Indices	
i	index for type of aircraft
j	index for type of C-check
t	index for time period
S_n	index for scenario
Notation	
I	set of aircraft in the Fleet
J_i	set of possible C-checks for aircraft i
T	set of time intervals
I_i^{FH}	maximum interval in flight hours
I_i^{DY}	maximum interval in calendar days
$U_{i,t,Sn}$	Daily flight hours for aircraft i at time t under scenario S_n
I_i^{D}	maximum interval D-check
$d_{i,j,Sn}$	Duration of check j for aircraft i under scenario S_n
C_E	Cost for having an extra slot assigned
d_c	Minimum time between two consecutive checks
Decision variables	
$x_{i,j,t}$	1 if check j at time t for aircraft i, 0 otherwise
$y_{i,t}$	$\geq$0, flown flight hours since last check for aircraft i at time t
E_t	1 if extra slot allocated at time t, 0 otherwise
$DY_{i,t}$	$\geq$0, calendar days since last check for aircraft i at time t
$D_{i,t}$	$\geq$0 calendar days since last D-check for aircraft i at time t
$m_{i,t}$	1 if aircraft i is being maintained at time t, 0 otherwise
$n_{i,j}$	1 if check j is a D-check for aircraft i, 0 otherwise

F. *Objective Function*

For the objective function several metrics can be used. It depends on the scenario S_n. This paper considers the minimization of unused flight hours since available cost data is often confidential or incomplete and hard to relate to check types. Next to that, the costs of an aircraft being out of operations outweigh the daily costs of a maintenance check. Minimizing unused flight hours indirectly reduces the number of maintenance checks and days out of operation. Therefore, the objective is considered the most suitable for the problem. The objective function can be found in Equation 6. The first part of the equation gives the total number of unused flight hours for all aircraft i and planned checks j at time t. As indicated in Section III-B, 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. Therefore a cost penalty (C_E) is assigned to needing extra slots at time t, in the second part of Equation 6.

$$f_{obj}(x, S_n) = \sum_{i \in I} \sum_{j \in J_i} \sum_{t \in T} \left(|I_i^{FH} - y_{i,t-1}| \right) \times x_{i,j,t} + \sum_{t \in T} E_t \times C_E \quad (6)$$

G. *Constraints*

When scheduling maintenance checks several constraints need to be taken into account. These constraints can be divided into utilization, operational, and check constraints.

1) *Utilization Constraints*

As indicated in Section III-A, the maximum intervals before an aircraft has to undergo a C-Check are defined in flight hours and calendar days. If an aircraft exceeds the interval it has to be grounded, resulting in heavy commercial revenue losses. Therefore, it is required that all aircraft are maintained within their respective intervals. So for the entire time period, T, the usage parameters of the aircraft should be lower or equal to the maximum interval. This is formulated in Equation 7 and Equation 8 for the flight hours ($y_{i,t}$) and calendar days ($DY_{i,t}$), respectively.

$$y_{i,t} \leq I_i^{FH} \qquad \forall i \in I, t \in T \quad (7)$$

$$DY_{i,t} \leq I_i^{DY} \qquad \forall i \in I, t \in T \quad (8)$$

The usage parameters of all aircraft need to be updated every time step. The flight hour parameter for an aircraft is updated with the average daily flight hours for the month of time step t. These average daily flight hours are estimated from historical data and are dependent on the scenario S_n and zero if an A-check is scheduled at time t. If an aircraft i is being maintained in the hangar, the flight hour parameter is reset to zero. This is defined in Equation 9. The calendar day parameter is updated by 1 day every time step t, for all aircraft i. The parameter is reset to zero for an aircraft if it is being maintained at that time. Updating the calendar day parameter is done through Equation 10.

$$y_{i,t+1} \geq \left(1 - m_{i,t}^C\right) \left(y_{i,t} + U_{i,t,S_n}\right) \qquad \forall i \in I, t \in [1, \ldots, T-1] \quad (9)$$

$$DY_{i,t+1} \geq \left(1 - m_{i,t}^C\right) \left(DY_{i,t} + 1\right) \qquad \forall i \in I, t \in [1, \ldots, T-1] \quad (10)$$

As indicated in Section III-A, a D-check is sometimes incorporated into a C-check. However, the D-check has a threshold interval expressed in calendar days of its own. Therefore, if the usage parameter of an aircraft for the D-check reaches its maximum interval, a C-check incorporating D tasks should be scheduled. This is a hard constraint and defined in Equation 11. The D-check usage parameter is updated by 1 every time step t, for all aircraft i (Equation 12). The parameter is reset to zero, only if an aircraft is being maintained for a C-check that incorporates tasks from the D-check. For all other planned C-checks, the parameter keeps being updated.

$$D_{i,t} \leq I_i^D \qquad \forall i \in I, t \in T \tag{11}$$

$$D_{i,t+1} \geq \left(1 - m_{i,t}^D\right)\left(D_{i,t} + 1\right) \qquad \forall i \in I, t \in [1, \ldots, T-1] \tag{12}$$

2) Operational Constraints

If an aircraft is being maintained it occupies 1 maintenance slot for the duration of its check. The duration of specific check j for aircraft i is dependent on the scenario, S_n, and also on the time t. The dependency on the time t is because work on a C-check is halted during weekends and public holidays. Therefore, the number of weekend days and public holidays in the specific check period are added to the total duration of the check ($d_{i,j,S_n}(t)$). The binary variable $m_{i,t}^C$ is switched to 1 through Equation 13 if aircraft i starts maintenance check j at time t for the duration of the check. The set of possible checks j, J_i, is different for each aircraft and is based on its last performed C-check and the four next C-checks in the cycle. To ensure that the binary variable remains $m_{i,t}^C$ 0 for all other instances, Equation 14 is introduced.

$$\sum_{t \in [t, t+d_{i,j,S_n}(t)]} m_{i,t}^C \geq d_{i,j,S_n}(t) \times x_{i,j,t} \qquad \forall i \in I, j \in J_i, t \in T \tag{13}$$

$$m_{i,t}^C \leq \sum_{j \in J_i} \sum_{t \in [t-d_{i,j,S_n}(t), t]} x_{i,j,t} \qquad \forall i \in I, t \in T \tag{14}$$

To set if an aircraft occupies a maintenance slot for a D-check at time t, Equation 15 is used. If an aircraft is not maintained for a D-check the variable $m_{i,t}^D$ is set to zero through Equation 16. Where the binary variable $n_{i,j}$ is defined by Equation 17 and defines whether the next C-check will have to incorporate tasks of the D-check or not.

$$\sum_{t \in [t, t+d_{i,j,S_n}(t)]} m_{i,t}^D \geq d_{i,j,S_n}(t) \times x_{i,j,t} \times n_{i,j} \qquad \forall i \in I, j \in J_i, t \in T \tag{15}$$

$$m_{i,t}^D \leq \sum_{j \in J_i} \sum_{t \in [t-d_{i,j,S_n}(t), t]} x_{i,j,t} \times n_{i,j} \qquad \forall i \in I, t \in T \tag{16}$$

$$n_{i,j} = \begin{cases} 1, & \text{if check j for aircraft i is a D-check} \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

In order for a schedule to be feasible, the used number of slots should not exceed the amount of slots that are available for every time step in the planning horizon. The available slots per time step t, L_t, have to be defined beforehand and considerations of an airline can be taken into account. It is for example often required 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 Section III-B, an airline can assign an extra slot, E_t, to a day if it is required to maintain all aircraft within their interval. Therefore, the capacity constraint is defined as in Equation 18.

$$\sum_{i \in I} m_{i,t}^C \leq L_t + E_t \qquad \forall t \in T \tag{18}$$

3) Check Constraints

The C-check is divided into a cycle of different check types. For example, (C1,. . ., C12). These C-checks are planned subsequently, meaning that after C1 a C2 check has to be scheduled, after C2 a C3, and so on. This is necessary due to the different durations of each check type. To formulate this in the problem, a constraint in the form of Equation 19 has been added. It ensures that check j has to be scheduled before a check of type j+1 is planned. Ensuring that the checks are scheduled according to the cycle.

$$\sum_{t \in T} x_{i,j,t} \geq \sum_{t \in T} x_{i,j+1,t} \qquad \forall i \in I, j \in [0, \ldots, J_i - 1] \tag{19}$$

Furthermore, a certain check type can only be scheduled once. Resulting in Equation 20.

$$\sum_{t \in T} x_{i,j,t} \leq 1 \qquad \forall i \in i, j \in J_i \tag{20}$$

Lastly, due to resource requirements, a minimum time between the start dates of two C-checks is required. This can be defined by the airline as d_c. The constraint is described in Equation 21

$$\sum_{t \in [t-d_c, t+d_c]} \sum_{i \in I} \sum_{j \in J_i} x_{i,j,t} \leq 1 \qquad \forall t \in T \tag{21}$$

IV. Genetic Algorithm

Since it is hard to solve the previously formulated problem for large scale cases, a customized GA is proposed. It is concerned with solving the problem of scheduling C-checks for a fleet of heterogeneous aircraft under uncertainty. The overall procedure of the proposed GA can be seen in Figure 2 and follows the basic GA algorithm structure as described in Kramer [30]. First, several initial schedules (Chromosomes) are generated using an ϵ-greedy algorithm. They form the initial population. After that, every schedule in the population is evaluated according to an evaluation function. Based on several parameters, selection of individual schedules for creating a new population (Parents) 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 is an iterative process until certain stopping criteria have been met. In order to reduce variability in the output of the algorithm, parallel machines are used to simulate different, independent runs of the proposed Genetic Algorithm. The best result out of the independent runs is the final outcome of the Algorithm. This approach, also known as probability amplification, has been proven effective for job shop scheduling of two machines and several other scheduling problems [31].

A. Chromosome Representation

In order 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 III. The rows represent the schedule for each individual aircraft (A/C 1, …, A/C n), and are referred to as genes. The columns represent which C-check is scheduled. C-1, indicates the first next C-check in the cycle of the aircraft. Meaning that if an aircraft previously had a C.8 check, C-1 represent the C.9 C-check, C-2 a C.10, etc. Each integer number in the gene indicates the starting date for that specific check. The C-1 check for aircraft 1 in Table III is, for example, scheduled to start at time step 396 in the planning horizon. An integer number of -1 indicates that that specific check is not scheduled in the planning horizon.

This paper considers the long-term C-check scheduling of a fleet aircraft over a period of 4 to 5 years. C-Checks are usually scheduled every 12-20 months subject to the MRO [32]. Therefore, a maximum number of 4 C-checks is assumed within the planning horizon. A schedule for a fleet of N aircraft therefore contains at maximum $N \times 4$ checks. If necessary, the maximum number of C-checks can easily be increased by adding an extra column to the chromosome.

TABLE III
Chromosome representation of the C-check maintenance schedule

	C-1	C-2	C-3	C-4
A/C 1	396	1089	-1	-1
A/C 2	88	506	1123	-1
A/C 3	135	-1	-1	-1
A/C 4	424	1120	-1	-1
⋮	⋮	⋮	⋮	⋮
A/C n	t_{n1}	t_{n2}	t_{n3}	t_{n4}

B. Initialization of Population

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

The algorithm simulates the usage parameters for all aircraft in the fleet over the planning period. The parameters are updated according to Equation 9, 10, and 12. If at time step t, an aircraft i has to be scheduled for an A-check the flight hour parameter, $y_{i,t}$, is not updated since the aircraft is being maintained and does not fly that day. All aircraft that have usage parameters within 90% of their interval at time step t, have a probability to be scheduled for a C-check. If the parameters are at their maximum, this probability is 1.

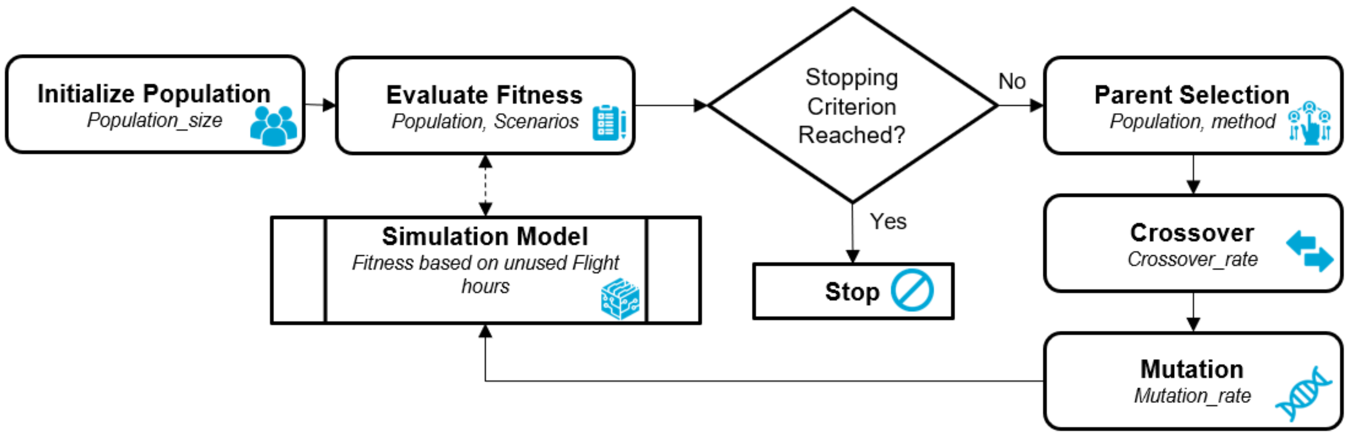


Fig. 2. Flowchart of the proposed Genetic Algorithm methodology, where the fitness is evaluated through a simulation model

Algorithm 1: ϵ-GREEDY ALGORITHM

```
Input: $\epsilon$, Aircraft, Initial Parameters, $\vec{L}$, $U_{S_n}$, , $D_{S_n}$
begin
    $\vec{y}^A_{t=0}, \vec{y}^C_{t=0}, \vec{DY}^C_{t=0}, \vec{D}_{t=0} \leftarrow$ Initial Parameters
    Chromosome, Chosen, Planning $\leftarrow$[ ]
    for $t$ in $T$ do
        $candidates \leftarrow$ [ ]
        for $i$ in Aircraft do
            $utilization \leftarrow U_{i,t,S_n}$
            if $y^A_{i,t} \geq I^{FH_A}_i$ then
                $y^A_{i,t} \leftarrow 0$
                $utilization \leftarrow 0$
            else if $t \in planning\{i\}$ then
                $y^C_{i,t}, DY^C_{i,t} \leftarrow 0$
                $utilization \leftarrow 0$
            $y^C_{i,t}, y^A_{i,t}$ += $utilization$
            $DY^C_{i,t}, D_{i,t}$ += 1
            if $y^C_{i,t}$ or $DY^C_{i,t}$ or $D_{i,t} \geq 0.9I$ & $rnd() \leq \epsilon$
              then
                candidates.insert($i$)
            else if $y^C_{i,t}$ or $DY^C_{i,t}$ or $D_{i,t} \geq I$ then
                candidates.insert($i$)
        end
        for $n$ in candidates.shuffle() do
            j $\leftarrow$ Aircraft$\{n\}$.Check
            find latest possible date, $p$, where:
            $L_p > 0 \quad \forall \quad p \in [p, p + d_{n,j,S_n}(p)]$
            with: $p \notin$ Chosen and $p \leq t$
            Chromosome$\{n\}$.insert($p$)
            Planning$\{n\}$.insert($[p, p + d_{n,j,S_n}(p)]$)
            Chosen.insert($[p - d_c, p + d_c]$)
            Aircraft$\{n\}$.Check += 1
            $y^C_{i,t}, DY^C_{i,t} \leftarrow 0$
            if Check is a D-Check then
                $D_{i,t} \leftarrow 0$
            for $d$ in $[p, p + d_{n,j,S_n}(p)]$ do
                $L_d$ -= 1
            end
            for $d$ in $[p + d_{n,j,S_n}(p), t]$ do
                $utilization \leftarrow U_{n,d,S_n}$
                $y^C_{i,t}, y^A_{i,t}$ += $utilization$
                $DY^C_{i,t}, D_{i,t}$ += 1
            end
        end
    end
    return Chromosome
end
```

Otherwise the probability is ϵ. This gives a list of aircraft for which a check has to be scheduled, on or before that specific time step. Each aircraft in the list is given a random priority. This priority is used to determine the order in which the aircraft are being scheduled. For each aircraft the latest possible date at which a C-check can start, is the date p, where for the entire duration of the check, $[p, p + d_{n,j,S_n}(p)]$, there is a maintenance slot available. This date has to be earlier or equal to the current time step t. When the latest possible date has been chosen for an aircraft, the available slots for the duration of that check are reduced by 1. The usage parameters, $y_{i,t}$ and $DY_{i,t}$ of the aircraft are also set to zero for the duration of the check. Depending on whether the check is a D-check or not, the $D_{i,t}$ parameter can also be reset. After this, the next aircraft in the priority order is scheduled, until all aircraft from the candidate list have been scheduled in. This process is repeated for every time step in the planning horizon. Resulting in a schedule for the entire fleet.

Due to the introduction of ϵ as well as a random scheduling priority, the algorithm can produce varying initial schedules that are feasible. Updating the usage parameters, and the duration of a check are dependent on the scenario that is used. When running the Genetic Algorithm with different scenarios, $S_n \quad \forall n \in SC$, the initial population is generated by running the ϵ-greedy algorithm, with an uniform randomly chosen scenario, or the average values from all scenarios. This is repeated until the population size is reached.

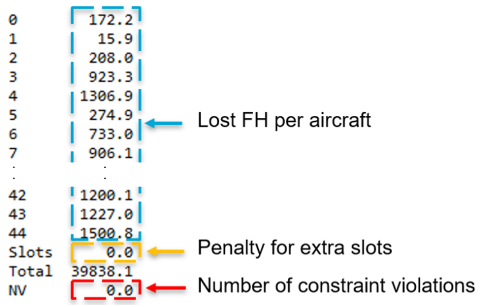
C. Fitness Evaluation/ Stopping Criterion

Each schedule in the population has to be evaluated in order to assess how good the schedule actually is. This evaluation is done according to Equation 1 and 2. The evaluation function for the genetic algorithm is based on Equation 6. However, the algorithm needs to incorporate some constraints, specifically the utilization constraints. To incorporate them in the GA, they are defined as soft constraints. Violations will result in a penalty added to the evaluation function. This penalty function $P_{i,j}(t)$, increases with days an aircraft is maintained after the interval and is defined according to Equation 22. The evaluation function for the fitness therefore changes to Equation 23. This means that there are two different additional penalty costs, C_E and C_T respectively. The ratio between those two determine whether the GA favours solutions with extra slots or solutions where aircraft are being maintained too late. Since grounding an aircraft means a high commercial revenue loss it is decided to have C_T an order of 10 higher than C_E. The algorithm will thus favour solutions that are made feasible through the creation of an extra slot.

$$P_{i,j}(t) = \begin{cases} 0, & \text{if } y_{i,t} \leq I^{FH}_i, DY_{i,t} \leq I^{DY}_i \\ \frac{y_{i,t} - I^{FH}_i}{U_{i,t,S_n}} C_T, & \text{if } y_{i,t} \geq I^{FH}_i \\ (DY_{i,t} - I^{DY}_i) C_T, & \text{if } DY_{i,t} \geq I^{DY}_i \end{cases} \tag{22}$$

$$f_{obj}(x, S_n) = \sum_{i \in I} \sum_{j \in J_i} \sum_{t \in T} \left(|I^{FH}_i - y_{i,t-1}| + P_{i,j}(t) \right) \times x_{i,j,t} + \sum_{t \in T} E_t \times C_E \tag{23}$$

By simulating the usage parameters for all aircraft in the fleet over the planning horizon, the evaluation function can be assessed for all scenarios and schedules in the population. Simulation is performed by using the SymPy package [33] in python and provides the amount of flight hours flown at each scheduled check together with any penalty that might have been assigned. An example of the simulator output can be seen in Figure 3. 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. It corresponds to the first part of Equation 23. The slots entry indicates whether extra slots are needed with the evaluated schedule and corresponds to the second part of Equation 23. The Total entry gives the overall fitness of the schedule and NV counts the total number of constraint violations for the evaluated schedule.



```
0        172.2
1         15.9
2        208.0
3        923.3
4       1306.9
5        274.9
6        733.0
7        906.1
:          :
42      1200.1
43      1227.0
44      1500.8
Slots      0.0
Total  39838.1
NV         0.0
```

Fig. 3. Example of fitness evaluation for a schedule with a fleet of 45 aircraft

Each schedule is assessed against all Scenarios, $S_n, \forall n \in SC$. The highest total fitness value over all scenarios indicates the worst-case scenario and is the final fitness of the evaluated schedule.

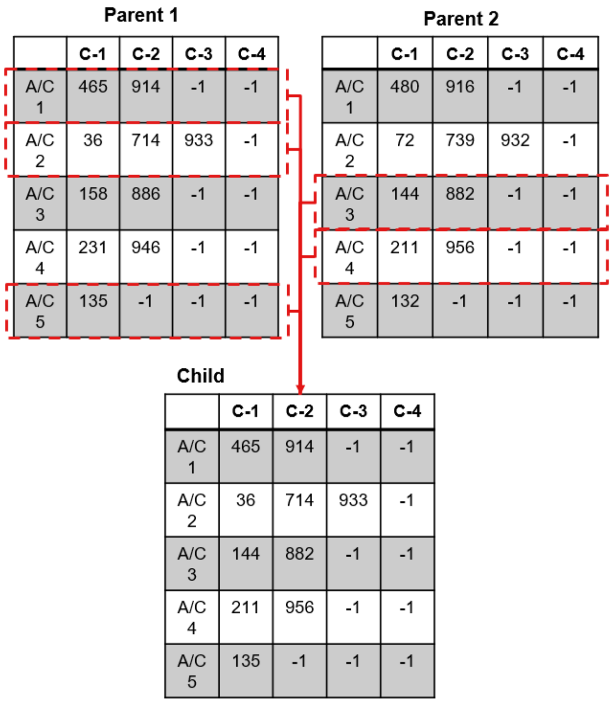
If the minimum fitness of the population has not changed for a predefined number of $n\_generations$ or if a maximum number of iterations has been achieved, the Genetic Algorithm is terminated. It returns the schedule which corresponds to the lowest total fitness value.

D. Parent Selection

Based on the fitness values of all 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 crossover. These new schedules form the next generation. Several selection techniques have been studied in Goldberg and Deb [34], where k-tournament selection is shown to have good convergence and computational time complexity properties. The principle of k-tournament selection is based on selecting a set of k individuals uniformly at random from the population. From the set of k 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 k is a trade-off between exploitation and exploration, and the most common size is k =2, the binary tournament selection [35]. A higher sample size will increase the probability that only the best individuals will be selected, losing some exploration properties. In scheduling literature, Hartmann [36], has researched several selection techniques, where tournament selection with size k =3 performs better than the binary tournament selection. Therefore, 3-tournament selection is used in this paper. In order to avoid not selecting the best individual from the population, the tournament selection is combined with an elitist method. The n best individuals from the population are selected to be included in the next generation. This way the best performing schedules are not lost during crossover or mutation.

E. Crossover

From the mating pool, individuals are randomly paired to form groups of parents. With random probability equal to the crossover rate, each pair of parents generate 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. Several, crossover operators exist in literature. The main techniques are, single-point, two-point, and uniform crossover [35]. For this problem, uniform crossover has been selected. Hu and Di Paolo [37] have successfully applied this approach for the aircraft arrival sequencing and scheduling problem with a similar integer matrix chromosome representation. In uniform crossover, each gene in the chromosome is randomly selected from one of the parents creating a new chromosome. This is done two times in order to generate two new schedules from each parent pair. An example of the uniform crossover operation can be seen in Figure 4 for a fleet of 5 aircraft.



Parent 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	158	886	-1	-1
A/C 4	231	946	-1	-1
A/C 5	135	-1	-1	-1

Parent 2

	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

Child

	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

The entire procedure of generating a new population based on the mating pool and uniform crossover can be seen in Algorithm 2. Firstly, a number of $n_{Elitist}$ 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
   Input: Mating Pool, crossover rate, Population
1  begin
2      New_Population ← [ ]
3      New_Population.insert(best $n_{Elitist}$ from Population)
4      for Parent_Pair ∈ Mating Pool do
5          if rnd() ≤ crossover rate then
6              child$_1$ ← Uniform_crossover(Parent_Pair)
7              child$_2$ ← Uniform_crossover(Parent_Pair)
8          else
9              child$_1$, child$_2$ ← Parent_Pair
10         New_Population.insert(child$_1$, child$_2$)
11     end
12     return New_Population
13 end
```

F. Mutation

After a new population has been generated by crossover, mutation in chromosomes can occur. 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, mutation is not random but follows the same ϵ-greedy algorithm presented in Algorithm 1. Probability $\epsilon = 0$ in order to schedule aircraft at the latest possible date. For a subset of aircraft of random size, mutation occurs. All aircraft outside of this subset keep their current schedule (gene). The aircraft in the subset are given a random priority and are scheduled at their latest possible date depending on the available slots, which in turn is dependent on the rest of the schedule. An example of the mutation procedure can be seen in Figure 5. For a schedule of a fleet of 5 aircraft, n=2 aircraft are chosen to be rescheduled. Here the size n, is a random number. Since the greedy algorithm is dependent on the scenario, a random scenario is selected if mutation occurs, or the mean value is used over all scenarios. The mutation procedure can be found in Algorithm 3.

```
Algorithm 3: MUTATION PROCEDURE
   Input: Population, mutation rate, Scenarios
1  begin
2      for Chromosome ∈ Population do
3          if rnd() ≤ mutation rate then
4              $n$ ← rnd(size=rnd()) //Rnd set of aircraft
5              $\vec{L}$ ← $\vec{L}$ based on $i \notin n$
6              case 1 do
7                  $U_{S_n}, D_{S_n} \leftarrow Scenarios.random()$
8              end
9              case 2 do
10                 $U_{S_n}, D_{S_n} \leftarrow Scenarios.mean()$
11             end
12             Procedure $\epsilon$-greedy($\epsilon$=0, n, $\vec{L}$, $U_{S_n}, D_{S_n}$)
13     end
14 end
```

V. VALIDATION

To validate the proposed methodology, several validation techniques are used. First, the problem is simplified in order to compare the output of the GA against the output of an exact solution method for several test cases. The exact solution method is based on a novel MILP formulation and solved by the commercial solver CPLEX. Secondly, the ability of the GA to find efficient near optimal solutions is assessed by comparing the outcome of the GA to results from a case study in Deng et al. [12]. Next, the robustness of schedules is assessed through Monte Carlo simulation. Lastly, a sensitivity analysis on the effect of probability level α and number of scenarios on schedule robustness is performed.

All runs with the genetic algorithm are performed with a mutation rate of 0.6, a crossover rate of 0.9, a population size of 20, and running the algorithm with 4 independent runs in parallel on a quad-core workstation. These parameters are found through sensitivity analysis and a trial-&-error approach.

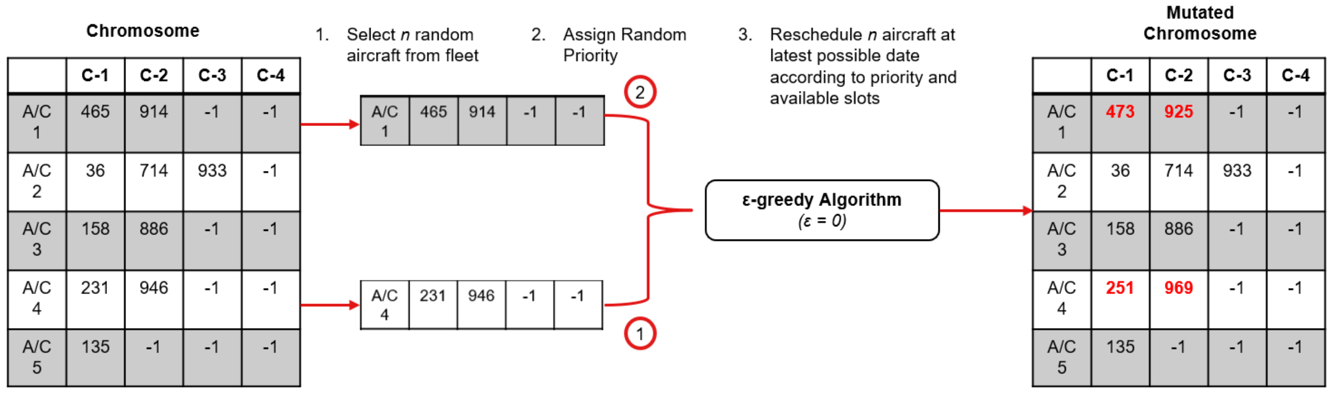


Fig. 5. Example of the mutation procedure for a chromosome, where 2 aircraft are randomly selected to be rescheduled using the greedy-algorithm

A. Benchmark Comparison

In order to check whether the proposed algorithm is a feasible solution to the problem, the outcome of the genetic algorithm is compared to an exact benchmark method for small test cases. The benchmark method is based on a novel MILP adaptation of the formulation presented in Section III. It is solved using the commercial solver CPLEX.

From the analysis of the state-of-the-art in Section II-B, it is clear that the complexity of scheduling aircraft maintenance checks makes it difficult to solve large scale problems with exact methods. Therefore, the GA and benchmark model are evaluated for small scale test problems. For the test problems the time step, t, is weekly and only the deterministic scenario is taken into account. Furthermore, the GA and MILP only consider the flight hour parameter constraints. The D-check, and calendar day parameters are dropped. Reducing the problem size by removing the constraints as in Equation 8, 10, 11, 12, 15, and 16. A total of 4 test cases are created with varying available slots and 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

The 4 test cases are run for a planning horizon of 2 and 3 years, with weekly time steps. The objective value and computation time are compared in order to assess the effectiveness of the GA for these small test cases. The outcome of the comparison can be seen in Table IV. Where the computation time for both the MILP and GA are displayed as well as the gap between their respective objective values. As can be seen, the computation time for the benchmark method increases exponentially when the problem size increases, while the GA solves the problem significantly faster when regarding problems with a longer time horizon. For the biggest test case of 40 aircraft and a planning horizon of 3 years the difference in computation time is already more than 35 minutes. It can also be seen, that the GA finds the optimal solution in most cases, having a maximum optimality gap of 0.39%.

TABLE IV
COMPARISON OF COMPUTATION TIME AND OBJECTIVE VALUE BETWEEN GA AND BENCHMARK METHOD FOR 4 SMALL TEST CASES

	Case 1		**Case 2**		**Case 3**		**Case 4**	
Planning Horizon [Yrs]	2	3	2	3	2	3	2	3
Comp. time MILP [s]	1.3	5.7	3.8	46.1	13.9	352	23.4	2271
Comp. time GA[s]	6.1	7	9.2	15.7	18.4	25.2	23.2	36.5
Obj. Gap[%]	0.0	0.0	0.0	0.31	0.0	0.39	0.02	0.37

B. Case Study

In order to assess the effectiveness of the GA for large scale problems, a case study is performed. The case study comes from Deng et al. [12] and considers the scheduling of C-checks for a European airline during the 2018-2021 period. Several operational constraints are defined for the test case:

- A maximum of 3 C-checks can be executed in parallel;
- During weekends and public holidays no work on a C-check is performed;
- During commercial peak periods (i.e. summer and holiday periods), no C-checks can be scheduled ($L_t = 0$);
- Due to resource availability reasons there has to be a minimum of 3 days between the start dates of two C-checks (d_c)

The starting date for the case study is the 25th of September 2017, and the planning horizon ends at the 31st of December 2021. The schedule created by the Genetic Algorithm is compared to a schedule created by maintenance planners of the airline and a schedule generated by the dynamic programming based methodology from Deng et al. [12]. The comparison is performed for the period of the 1st of January 2018 till the 31st of December 2021. The generation of a C-check schedule by the airline maintenance planners, the dynamic programming methodology, and the proposed genetic algorithm in this paper, use the same input data and operational constraints. Unscheduled maintenance events and

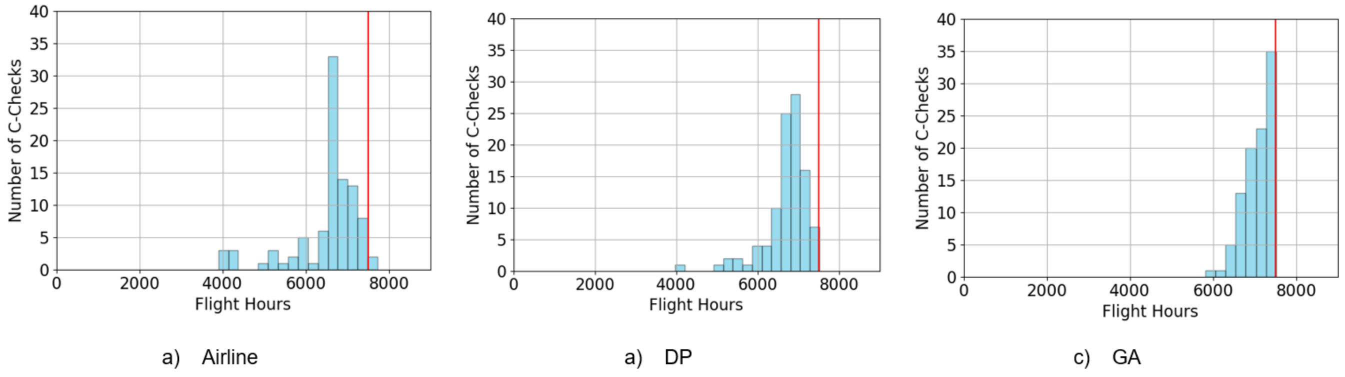


Fig. 6. Comparison of flight hours at C-check between schedules from the airline, the dynamic programming method, and the genetic algorithm

aircraft routing are not taken into account.The data set that is used for the case study is available at:

https://doi.org/10.4121/uuid:1630e6fd-9574-46e8-899e-83037c17bcef

The results of the comparison can be seen in Table V. From the table, it is readily clear that the Genetic Algorithm finds significantly better solutions than current practice for the airline. Optimization with the proposed 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 results of optimization with a DP based approach, the GA further reduces the total number of C-checks by 3, and increases the average flight hours by 5.9%. The computation time of the GA is over twice as long as the computation time of the dynamic programming approach. This is, however, still significantly better than the computation time of over 3 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 Figure 6. The distribution of flighth hours at C-checks for the GA is shifted to the right compared to the airline and DP schedule. The amount of checks that are scheduled near their deadline, represented as the red line, is also increased.

TABLE V
Results of optimization for the case study with the GA, compared to the results from an European airline, and dynamic programming methodology [12]

	Airline	**Dynamic Programming**	**GA Deterministic**
Average Flight Hours	6539	6691	7087
Number of C-checks	96	90	87
Computation time	$\geq$3 days	510 s	1059 s

Since there is a difference between the dynamic programming method [12] and the genetic algorithm, further cross-validation has been performed. In order to assess whether assumption A.6 from Section III-B has a significant impact on the flight hours, the schedule created by the DP methodology is run in the simulator used by the GA. The flight hours of the simulation are compared to the flight hours resulting from the DP algorithm. Figure 7 shows the difference in flight hours generated by the GA simulator and the DP methodology for the same schedule. As can be seen, the maximum difference is 12.9 flight hours, and the average difference is only 1.66 flight hours. This difference is caused by the assumption that the GA does not consider a feasible A-check schedule by using the A-check due dates (A.6).However, the difference in flight hours is small, indicating that the GA simulator provides accurate results for the usage parameters.

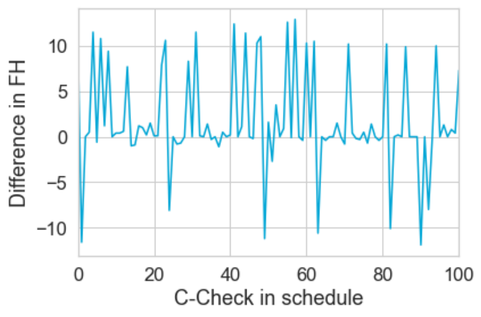


Fig. 7. Difference between flown flight hours resulting from the GA simulator and the DP methodology from Deng et al. [12], using the same schedule

C. *Robustness*

The previous section concerned the analysis of the effectiveness of the proposed genetic algorithm by means of a case study. However, the outcome of the case study is only based on the deterministic scenario. Figure 6 shows that a lot of C-checks are scheduled close to the maximum interval. A small increase in average daily flight hours or decrease in check duration could therefore already result in aircraft being maintained after their maximum interval. Therefore, the genetic algorithm has been run for the same test case, but taking into account multiple scenarios as described in Section III-D.

The outcome for running the GA with S=3 and probability level α = 0.8, can be seen in Table VI. As expected, the robust optimization results in lower utilization and two more C-checks compared to the deterministic optimization. Using 10 scenarios has also increased the computational time. The robust schedule is still efficient by reducing the total amount of C-checks by 7 and increasing the average utilization by 4.4% compared to the current approach of a European airline. When taking into account that an airline spends on average between \$70K and \$350K [32] on a C-check. This reduction could lead to an annual cost saving between \$122.5K - \$612.5K. On the other hand, the reduced number of C-checks also cause an additional of 100-140 days in operation over the planning horizon. The increase in average utilization adds a further 25 days of operation. Annually, this is an extra of 31-41 days. Considering that a day of operations of a short-haul aircraft can generate around \$75K-\$120K in revenue, this could mean \$2.3M-\$4.9M of additional annual revenue.

TABLE VI
Results of optimization for the case study with the GA, for 10 scenarios and probability level of 80%, compared to the results of the GA for the deterministic scenario

.	**GA Determinstic**	**GA S=3, α=0.8**
Average Flight Hours	7087.3	6825.8
Number of C-checks	87	89
Computation time	1059 s	1643 s

To evaluate the robustness of both schedules, a Monte Carlo simulation has been performed with n=10000 runs. Each run, the schedules are assessed with the simulator under a different duration and daily utilization matrix. A duration matrix is generated by choosing random durations from historical data. A daily utilization matrix is generated as a random vector from the multivariate normal distribution presented in Section III-D2. This way, the schedules are assessed against 10000 varying scenarios. Each run, the total number of days aircraft are maintained after their maximum interval and the amount of extra slots necessary under the scenario of that run are stored. No recovery procedures are in place, 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. The goal of this analysis is to find out how often the schedule would be infeasible and quantify the difference between the deterministic and robust approach.

Figure 8 shows the results of the Monte Carlo simulation regarding the amount of days aircraft are maintained after their maximum interval. By optimizing using only the deterministic scenario, small uncertainties can have a big influence. This can be seen in the fact that the average amount of days aircraft would have to be grounded in the deterministic schedule is 32.3 days, where this is reduced to an average of 2.7 days for the robust optimization schedule.

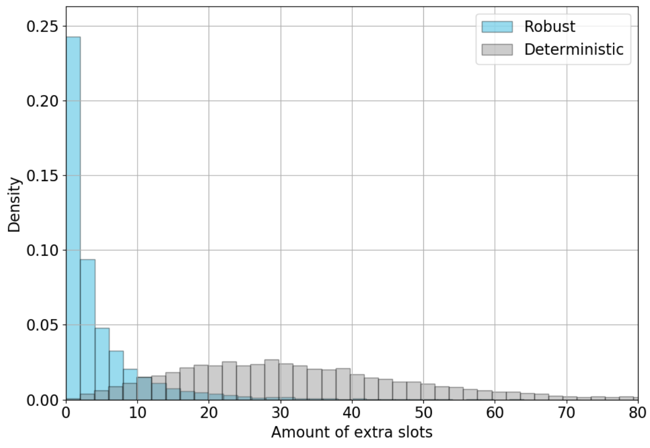


Fig. 8. Distribution of total amount of days aircraft in the fleet have to be grounded because they exceed their maximum interval during the planning horizon for the deterministic and robust GA optimization, result of MC simulation with n=10000 runs

Based on the distributions, the probability that the total days aircraft are maintained past their deadline lie within a certain range have been calculated as seen in Table VII. From the table it is readily clear that the min-max optimization is significantly more robust than optimization with only using the deterministic scenario. For the robust optimization schedule there is a 41% chance that no conflicts and disruptions occur, compared to a chance of 0.27% for the schedule generated by deterministic optimization.

TABLE VII
PROBABILITY FOR THE TOTAL DAYS AIRCRAFT ARE MAINTAINED PAST THEIR DEADLINE TO LIE WITHIN A CERTAIN RANGE

Days too late	**GA Deterministic probability**	**GA S=3, alpha=0.8 probability**
0-1	0.27%	41.12%
1-20	25.90%	55.34%
20-50	58.92%	3.36%
$\geq$50	14.91%	0.18%

Figure 9 shows the results of the Monte Carlo simulation regarding the amount of extra slots that are necessary due to uncertainty in check duration. As can be seen, the distribution is shifted to the left for the robust optimization. Having a mean of 2.1 extra slots compared to 17.2 in the case of deterministic optimization. Table VIII shows the probabilities for the amount of extra slots to lie within certain ranges. It can be seen that the schedule created by optimizing with 10 scenarios, is also more robust than the deterministic schedule when regarding available maintenance slots.

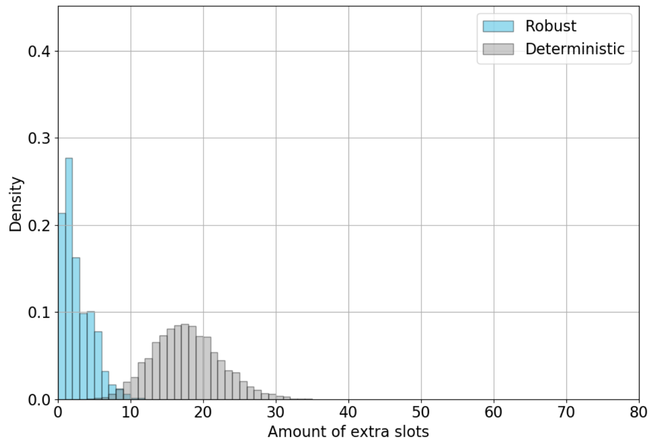


Fig. 9. Distribution of the amount of extra slots that are necessary due to uncertainty in check duration for the deterministic and robust GA optimization, result of MC simulation with n=10000 runs

TABLE VIII
PROBABILITY FOR THE AMOUNT OF EXTRA SLOTS IN THE PLANNING HORIZON TO LIE WITHIN A CERTAIN RANGE

Extra Slots	**GA Deterministic probability**	**GA S=3, alpha=0.8 probability**
0-1	0.02%	35.08%
1-10	5.40%	64.71%
10-20	68.30%	0.21%
$\geq$20	26.29%	0%

D. Sensitivity Analysis

In order to analyze the effect of the robust optimization parameters, S and α, a sensitivity analysis has been performed. For a test case with 20 aircraft and 2 hangar slots available, the GA has been run with varying robust optimization parameters. The probability factor α is either 0.5, 0.65, 0.8, 0.95, or a combination of all. A bigger probability factor would lead to more critical scenarios. When combining different probability factors, scenarios are generated according to the methodology in Section III-D1 and III-D2, but with sampling from the list of probability factors without replacement. The scenario size, S, lies in the range between 2 and 7, corresponding to a total number of scenarios between 5 to 50.

For each different combination of robust optimization parameters the average number of constraint violations have been recorded based on MC simulation with n=10000 runs. The number of constraint violations are the sum of total days aircraft have to be grounded and the total amount of extra slots needed during the planning horizon. A contour plot for this can be seen in Figure 10 (a). The average utilization for each optimization run has also been recorded. Where the utilization is defined as the average flight hours an aircraft has flown when it is scheduled for a C-check. A contour plot of the average utilization per robust optimization setting can be seen in Figure 10 (b).

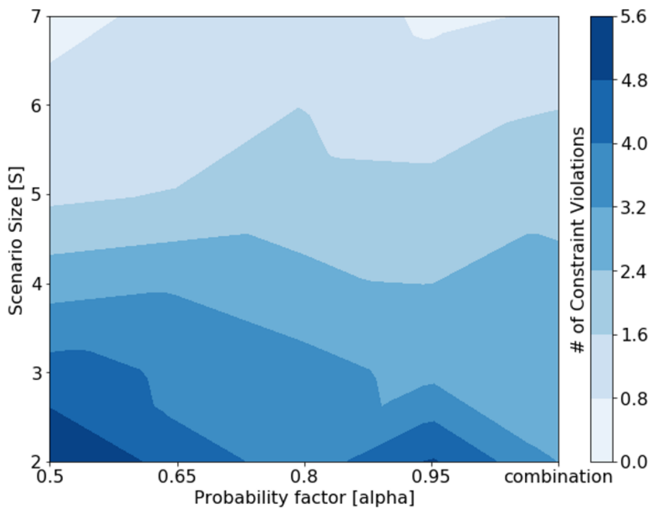
In the contour plots of Figure 10 it can be seen that a higher number of scenarios reduces the average number of constraint violations, thereby making the schedule more robust. This robustness, however, comes with a reduction in the average utilization. The contour plots also show, that for a smaller number of total scenarios, an increasing probability factor has a positive influence on the robustness of the schedules created by the GA. 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, due to the fact that the scenarios can already vary a lot at the same probability level. In most cases, using scenarios generated from a combination of α tend to be more robust than scenarios from a single probability factor for smaller scenarios. This results from the fact that a combination of probability factors increase the variability in the scenarios. A run for the same test case with only using the deterministic scenario resulted in an average number of constraint violations of 45.8 and an utilization of 7124 flight hours. From the contour plots it can be seen that even with only including 5 scenarios in the optimization, the average number of constraint violations is reduced by 90%. This comes at a cost of 2.5% reduction in average utilization.

Sensitivity to a change in scenario size can be seen in Table IX with respect to utilization and number of constraint violations. Where sensitivity is defined as an estimate of the partial derivative [38], based on the reference scenario of $S = 2$. The table confirms that an increase in scenario size reduces the number of constraint violations on average, but with a reduction in utilization.

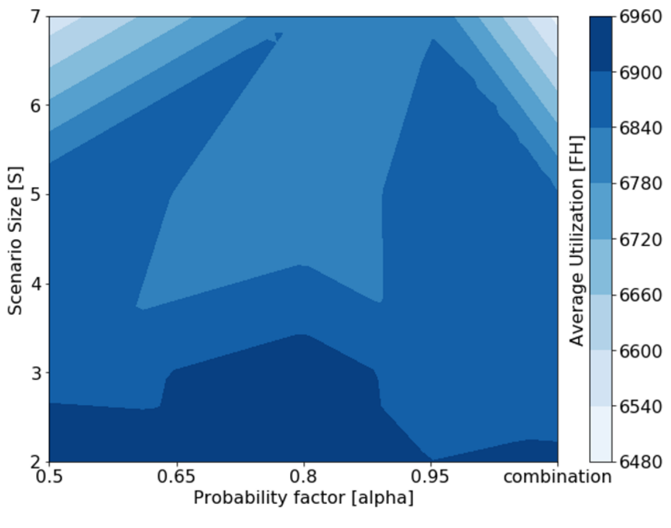
TABLE IX
SENSITIVITY TO A CHANGE IN SCENARIO SIZE S ON UTILIZATION AND NUMBER OF CONSTRAINT VIOLATIONS

S	Utilization [FH]	Sensitivity FH	Number of constraint violations	Sensitivity Number of constraint violations
2	6925.8 ± 52.5	114.75	4.27 ± 1.11	2.62
3	6887.5 ± 62.2	N/A	3.40 ± 0.67	N/A
5	6849.8 ± 102.6	-56.63	1.82 ± 0.35	-2.34
7	6687.5 ± 343.4	-150	0.87 ± 0.35	-1.90

The scenario size, S, does not only affect the robustness and utilization of schedules created by the GA it also has a significant influence on computation time. This can be seen in Figure 11. Choosing the robustness parameters is thus a trade off between robustness, average utilization, and computation time.



a) Number of Constraint Violations



b) Average Flight Hours

Fig. 10. Contourplots of the average number of constraint violations and flight hours for varying robust optimization parameter settings

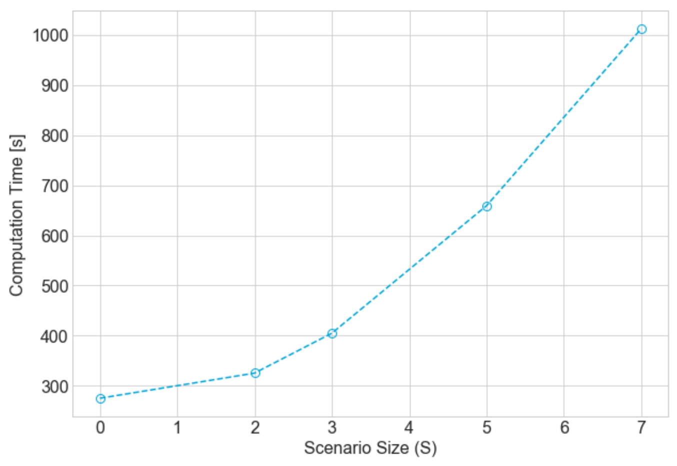


Fig. 11. Average computation time for the Genetic Algorithm with varying scenario size over 10 runs

VI. Discussion

There has been little focus on the long-term scheduling of heavy-maintenance checks in literature. Current approaches are heavily reliant on a manual approach and do not take the inherently stochastic nature of maintenance into account. This paper proposes a GA that can generate efficient robust C-check schedules for a fleet of heterogeneous aircraft. A robust long-term schedule could reduce the number of adjustments, reduce cost, and give a good forecast of required resources.

First, the proposed genetic algorithm has been validated for smaller test cases against an exact benchmark method. The results from Section V-A 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 has a significantly reduced computation time compared to the exact benchmark method. This is expected since the problem is NP hard.

The effectiveness of the GA to generate C-check schedules for larger scale problems is evaluated by a case study in Section V-B. The results of the GA are compared to the current approach of an European airline and a dynamic programming approach [12] from literature. For the case study, the GA finds a significantly better schedule when optimizing for the deterministic scenario. It reduces the total number of C-checks in the planning horizon by 9 and 3, compared to the airline and DP methodology respectively. The average utilization is also increased by 8% compared to the schedule created by the current approach of the airline and 6% compared to that of dynamic programming. The genetic algorithm finds an efficient schedule within 20 minutes. This is significantly more efficient than the 3 days of the current approach of the airline. Since there is a major difference between the proposed GA and DP methodology, cross validation has been performed. A comparison by running the same schedule in both algorithms did not provide major differences, indicating that the assumptions presented in this paper do not influence the evaluation of the fitness of a schedule significantly. The DP methodology has the possibility to combine A and C-checks. A possible explanation for the difference could be that, with the DP methodology, some C-checks are combined with an A-check, shifting the C-check deadline to the deadline of the A-check, whereas the approach presented in this paper does not take the feasibility of scheduling an A-check into account. By assuming that the A-checks are scheduled at their due date the GA considers around 100 A-checks less in the planning horizon than the DP based methodology does. Since an A-check duration is 1 day and the average daily utilization is 11 flight hours, this results in a total overestimation of 1100 flight hours for the GA schedule. This is divided over 87 C-checks in the planning horizon, so the average flight hours would only be reduced by 12.6 if A-check feasibility would be considered for the GA.

The ability of the GA to generate robust schedules is evaluated through Monte Carlo simulation in Section V-C. The schedule created by the GA without taking uncertainty into account is compared to a schedule generated by the GA when taking varying scenarios into account. These scenarios are generated based on predefined robustness parameters. The robust schedule is still efficient by reducing the total amount of C-checks by 7 and increasing the average utilization with 4.4% compared to the current approach of an European airline. When optimizing for only the deterministic scenario, many aircraft are scheduled close to their deadline. Therefore, a small change in utilization or check duration can already have a significant impact. When optimizing for varying scenarios, however, the impact of these changes are less significant. In only 0.27% of the cases the deterministic schedule would result in all aircraft being maintained on time. This is increased to 41% for the robust schedule. A sensitivity analysis in Section V-D shows that choosing the robustness parameters is a trade off between robustness, average utilization, and computation time. These findings are a result of the Monte Carlo simulation. However, no recovery procedures are in place. Inclusion of recovery procedures would give a more accurate depiction of the difference in robustness between the schedules. Furthermore, the schedule created by the airline is not entirely available. Therefore, no comparison between the robustness of the schedule created by the genetic algorithm and the current approach of the airline can be performed.

From this, it is clear that the proposed GA can tractably find efficient (near-)optimal solutions for smaller and larger scale problems. By including varying scenarios in the optimization, the algorithm finds a more robust schedule, compared to deterministic optimization, that is still efficient. Reducing the need for adjustments to the schedule if small uncertainties manifest itself in check duration or utilization while also reducing maintenance cost.

VII. Conclusions & Recommendations

With the impact of COVID-19 on the MRO market and the expected growth of the airline industry after 2022 [2], reducing costs and valuing efficiency is important in both the short- and long-term. The MRO market currently spans around 9.5% of the total operating cost of an airline [3]. Of this, 70% is covered by heavy-maintenance. Reduction of these costs and insight in future capacity needs could, therefore, be significant for an airline. A possible solution is the optimization of the long-term schedule of heavy-maintenance checks. Current approaches are found to be reliant on manual input and operator experience. In addition to that, revisions to the initial schedule are made continuously due to the inherent stochastic nature of aircraft maintenance through non-routine maintenance. Taking this uncertainty into account could offer more robust schedules, saving cost and providing reliable insight into future capacity needs.

This paper proposes a genetic algorithm methodology that can generate robust and efficient C-check schedules for a fleet of heterogeneous aircraft. Uncertainty in check duration and daily utilization are taken into account by generation of varying scenarios. These scenarios are assessed by the genetic algorithm through min-max optimization. When applied to a case study, optimization with the genetic algorithm for only the deterministic scenario, reduces the total amount of C-checks by 9 while increasing the average utilization by 8.4% compared to the current approach of an European airline. However, Monte Carlo simulations show that this schedule is sufficient in only 0.27% of cases, since small changes in check duration or daily utilization can have a significant impact. When including varying scenarios into the genetic algorithm a more robust schedule can be found which is sufficient in 41% of the cases. The robust schedule is still significantly more efficient than the current approach of the airline, reducing the total number of C-checks by 7 and increasing utilization by 4.4%. This could lead to a reduction of direct annual maintenance costs of \$122.5K - \$612.5K. Furthermore, the increase in aircraft availability could result in an additional \$2.3M - \$4.9M in annual revenue.

This paper is the first to address the long-term scheduling of heavy-maintenance checks while taking uncertainty in check duration and daily utilization into account. The proposed genetic algorithm finds robust and efficient C-check schedules for large scale problems in under 30 minutes.

Recommendations for future research include the addition of scheduling A-checks. Currently, the approach does not take into account feasibility of A-checks. By expanding the problem to take operational constraints on the A-check into account further cost reductions could be achieved. Furthermore, more research on the uncertainty in check duration and daily utilization needs to be performed. Currently, relatively little data is available. A more detailed probability description and scenario generation could increase robustness. Finally, a schedule disruption management tool can be implemented to better analyze the robustness of the schedules and even further reduce the need of adaptations to the maintenance schedule.

References

[1] Statista, "Size of aircraft fleets worldwide — statistic," 2019, accessed on 03.06.2019. [Online]. Available: https://www.statista.com/statistics/262971/aircraft-fleets-by-region-worldwide/

[2] O. Wyman, "Aviation aftermarket and covid-19," accessed on 13.05.2020. [Online]. Available: https://www.oliverwyman.com/our-expertise/insights/2020/mar/COVID-19-Impact-On-Commercial-Aviation-Maintenance.html

[3] IATA, "Iata's maintenance cost task force. airline maintenance cost executive commentary – an exclusive benchmark analysis," 2016.

[4] R. Gopalan and K. T. Talluri, "The aircraft maintenance routing problem," *Operations Research*, vol. 46, no. 2, pp. 260–271, 1998.

[5] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, 01 1982, vol. 32.

[6] N. J. Boere, "Air canada saves with aircraft maintenance scheduling," *Interfaces*, vol. 7, no. 3, pp. 1–13, 1977.

[7] IFS, "A new approach to maintenance planning," 2019, accessed on 13.06.2019. [Online]. Available: https://www.ifsworld.com/corp/solutions/ifs-maintenix/fleet-planner/

[8] M. Etschmaier and P. Franke, "Long-term scheduling of aircraft overhauls," *AGIFORS Symposium*, 1969.

[9] J.-M. Pla, "An "out-of-kilter" algorithm for solving minimum cost potential problems," *Mathematical Programming*, vol. 1, no. 1, pp. 275–290, Dec 1971.

[10] H. Bauer-Stämpfli, "Near optimal long-term scheduling of aircraft overhauls by dynamic programming," *AGIFORS Symposium*, 1971.

[11] S. Yan, C.-Y. Chen, and C.-Y. Yuan, "Long-term aircraft maintenance scheduling for an aircraft maintenance centre: A case study," *International Journal of Applied Management Science*, vol. 1, pp. 143–159, 02 2008.

[12] Q. Deng, B. F. Santos, and R. Curran, "A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization," *European Journal of Operational Research*, vol. 281, no. 2, pp. 256–273, 2020. [Online]. Available: https://dx.doi.org/10.1016/j.ejor.2019.08.025

[13] C. Sriram and A. Haghani, "An optimization model for aircraft maintenance scheduling and re-assignment," *Transportation Research Part A: Policy and Practice*, vol. 37, no. 1, pp. 29–48, 2003.

[14] G. Kozanidis and A. Skipis, "Flight and maintenance planning of military aircraft for maximum fleet availability," *Military Operations Research*, vol. 15, no. 1, 2010.

[15] M. Başdere and Bilge, "Operational aircraft maintenance routing problem with remaining time consideration," vol. 235, no. 1, pp. 315–328, 2014. [Online]. Available: https://dx.doi.org/10.1016/j.ejor.2013.10.066

[16] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[17] Z. Yang and G. Yang, “Optimization of aircraft maintenance plan based on genetic algorithm,” *Physics Procedia*, vol. 33, pp. 580 – 586, 2012.

[18] G. Quan, G. W. Greenwood, D. Liu, and S. Hu, “Searching for multiobjective preventive maintenance schedules: Combining preferences with evolutionary algorithms,” *European Journal of Operational Research*, vol. 177, no. 3, pp. 1969 – 1984, 2007.

[19] M. P. Kleeman and G. B. Lamont, *Solving the Aircraft Engine Maintenance Scheduling Problem Using a Multi-objective Evolutionary Algorithm*. Springer Berlin Heidelberg, 2005, pp. 782–796.

[20] R. Elshaer, “Solving resource-constrained project scheduling problem using genetic algorithm,” *Journal Of Al Azhar University Engineering Sector*, vol. 12, pp. 187–198, 02 2017.

[21] R. Zamani, “An accelerating two-layer anchor search with application to the resource-constrained project scheduling problem,” *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 6, pp. 975–984, Dec 2010.

[22] P. Samaranayake, “Current practices and problem areas in aircraft maintenance planning and scheduling–interfaced/integrated system perspective,” 01 2006.

[23] D. Dinis, A. Barbosa-Póvoa, and P. Teixeira, “A supporting framework for maintenance capacity planning and scheduling: Development and application in the aircraft mro industry,” *International Journal of Production Economics*, vol. 218, pp. 1–15, 2019.

[24] V. Mattila and K. Virtanen, “Scheduling fighter aircraft maintenance with reinforcement learning,” in *Proceedings of the 2011 Winter Simulation Conference (WSC)*, Dec 2011, pp. 2535–2546.

[25] S. Sohn and K. Yoon, “Dynamic preventive maintenance scheduling of the modules of fighter aircraft based on random effects regression model,” *The Journal of the Operational Research Society*, vol. 61, no. 6, pp. 974–979, 2010.

[26] H. A. Kinnison, *Aviation Maintenance Management ; Second Edition*. McGraw-Hill, 2013.

[27] R. Vershynin, “How close is the sample covariance matrix to the actual covariance matrix?” *Journal of Theoretical Probability*, vol. 25, no. 3, p. 655–686, 2012. [Online]. Available: https://dx.doi.org/10.1007/s10959-010-0338-z

[28] B. Eck, F. Fusco, and N. Taheri, *Scenario Generation for Network Optimization with Uncertain Demands*, pp. 844–852.

[29] “Reprint of: Mahalanobis, p.c. (1936) ”on the generalised distance in statistics.”,” *Sankhya A*, vol. 80, no. S1, p. 1–7, 2018.

[30] O. Kramer, *Genetic Algorithm Essentials*. Springer International Publishing, 01 2017.

[31] D. Sudholt, *Parallel Evolutionary Algorithms*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 929–959.

[32] A. Shannon, “Basics of aircraft maintenance programs for financiers,” accessed on 12.06.2019. [Online]. Available: http://aircraftmonitor.com/uploads/1/5/9/9/15993320/basics_of_aircraft_maintenance_programs_for_financiers___v1.pdf

[33] A. M. et al., “Sympy: symbolic computing in python,” *PeerJ Computer Science*, p. e103, Jan. 2017.

[34] D. E. Goldberg and K. Deb, “A comparative analysis of selection schemes used in genetic algorithms,” ser. Foundations of Genetic Algorithms, G. J. RAWLINS, Ed. Elsevier, 1991, vol. 1, pp. 69 – 93.

[35] J. E. Rowe, *Genetic Algorithms*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 825–844.

[36] S. Hartmann, “A competitive genetic algorithm for resource-constrained project scheduling,” *Naval Research Logistics (NRL)*, vol. 45, no. 7, pp. 733–750, 1998.

[37] X.-B. Hu and E. D. Paolo, “An efficient genetic algorithm with uniform crossover for air traffic control,” *Computers & Operations Research*, vol. 36, no. 1, pp. 245 – 259, 2009, part Special Issue: Operations Research Approaches for Disaster Recovery Planning.

[38] A. Sharpansky, “Lecture notes in agent based modelling and simulation in air transport,” February 2019.

II
Literature Research

*Previously graded under AE4020

Abstract

There is an increasing demand on the MRO market, which currently spans around 9.5% of the total operating cost. Of this, around 70% is covered by heavy-maintenance. Reduction of these costs and insight in future capacity needs could, therefore, be significant for an airline. A possible solution is the optimization of the long-term schedule of heavy-maintenance checks. In order to do so, the state of the art of aircraft maintenance scheduling and scheduling models are discussed. Current approaches are found to be reliant on manual input and operator experience. Next to that, revisions to the initial schedule are made continuously due to the inherent stochastic nature of aircraft maintenance through non-routine maintenance. Taking this uncertainty into account could offer more robust schedules, saving cost and providing reliable insight into future capacity needs. The incorporation of uncertainty is also one of the main research directions in scheduling literature, where hybrid meta-heuristics with Monte Carlo simulations are used to solve the NP-hard problem. The analysis and discussion of the state of the art has led to the following research aim: *"To develop a stochastic maintenance scheduling model capable of tractably delivering an effective long-term schedule for aircraft C-checks while taking into account the main sources of uncertainty in C-Check scheduling"*

1 Introduction

The size of the global commercial aircraft fleet is estimated to grow from 24,400 to 48,540 by 2037 [81], indicating that the global aircraft industry is expanding fast. With the increasing number of aircraft, the MRO market has to grow accordingly (expected annual growth rate of 4% [91]). Next to the augmenting demand on maintenance, the global MRO spend spans on average 9% to 10% of the total operating costs of an airline [42]. With the increasing maintenance demand and high associated costs, it is beneficial for airlines to keep introducing innovations with respect to the scheduling of maintenance and to value efficiency.

Regular aircraft maintenance is necessary in order to assure airworthiness, keep the aircraft reliable, as well as provide assurance of flight safety. Currently maintenance is scheduled according to flight hours, flight cycles, or calendar days according to the MSG-3 approach[49], resulting in three noteworthy maintenance checks: A-, C-, and D- Checks. Each check has different duration, frequency, and tasks [36], where the C-, and D-Checks are the largest check types, labeled as heavy-maintenance checks. During these checks, the aircraft is not operational and kept on the ground for several weeks. Therefore, heavy-maintenance covers around 70% of the total maintenance costs and requires the most amount of resources [42].

Current approaches to the long-term scheduling of the maintenance checks are often based on operator experience and simulation models where manual selection is a major component. Next to that, the initial schedule has to be continuously revised due to the stochastic nature of aircraft maintenance. These revisions can result in backlog and can have an impact on maintenance cost, quality of service, and the maintenance schedule. Taking this into account when scheduling could offer more robust schedules where fewer revisions are necessary. Developing an efficient long-term schedule of these heavy-maintenance checks, while taking into account uncertainty, could therefore not only result in significant cost savings but also provide a robust schedule with insight into future resource requirements.

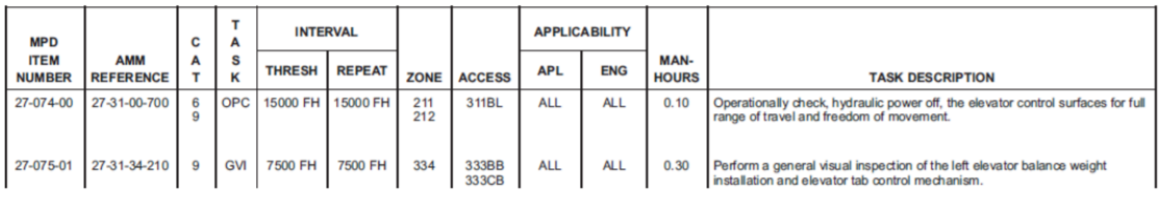
To study the research topic more in-depth, a literature review is carried out. The report starts with an overview of the state of the art in Chapter 2. This chapter focuses on getting a better insight in the areas concerning the scheduling of aircraft maintenance. Hereafter, a discussion of the found literature is stated together with the applicability to the research in Chapter 3. Next, in Chapter 4, the framework of the research is set up. Lastly, the literature review is concluded in Chapter 5.

2 State of the Art

2.1. Aircraft Maintenance and Planning

As indicated in Section 1, maintenance can be divided into three noteworthy maintenance checks: A-, C-. and D-Checks. Type A checks generally involve inspection of the aircraft interior/exterior, with focus on lubrication of the major systems of the aircraft. The check occurs biweekly to monthly [80]. C-Checks are usually scheduled every 12-20 months subject to the MRO [76]. This check falls under the heavy-maintenance category and requires the aircraft to be taken out off operation for up to a month [80]. A Heavy Maintenance Visit (HMV) or D-Check, is planned in the range of 6-12 years, determined by utilization and aircraft type[76]. Many airlines merge tasks of the D-check into a C-check, resulting in a so-called heavy C-check. Smaller operators even plan the lease of their aircraft in such a way that it is terminated before a D-check.

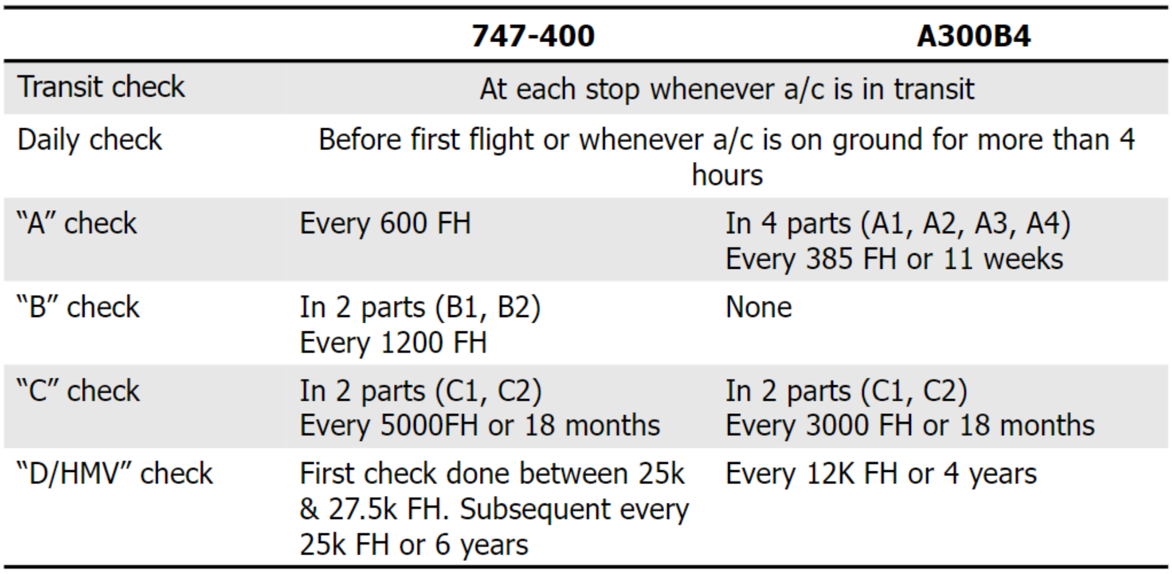
Currently, the planning of aircraft maintenance follows the MSG-3 approach [49]. This is a task-oriented approach that develops the maintenance tasks that need to be performed together with their corresponding intervals. These intervals are defined in number of flight hours, amount of flight cycles, or calendar days. They are determined according to hard time, on-condition or condition monitoring criteria. These tasks are noted in the Maintenance Planning Document (MPD), of which an example can be seen in Figure 2.1. The MPD gives the threshold interval per required task, and can be used to group tasks into task packages and letter checks. Every operator will use the maintenance schedule to suit its own operations and can therefore establish their own maintenance program, if it is accepted by the airworthiness authorities.



MPD ITEM NUMBER	AMM REFERENCE	CAT	TASK	INTERVAL		ZONE	ACCESS	APPLICABILITY		MAN-HOURS	TASK DESCRIPTION
				THRESH	REPEAT			APL	ENG		
27-074-00	27-31-00-700	6 9	OPC	15000 FH	15000 FH	211 212	311BL	ALL	ALL	0.10	Operationally check, hydraulic power off, the elevator control surfaces for full range of travel and freedom of movement.
27-075-01	27-31-34-210	9	GVI	7500 FH	7500 FH	334	333BB 333CB	ALL	ALL	0.30	Perform a general visual inspection of the left elevator balance weight installation and elevator tab control mechanism.

Figure 2.1: Example Maintenance Planning Document from Ackert [76]

Most larger operators follow the Block maintenance approach, which are often denoted by the previously mentioned letter checks. But smaller operators can also choose for the Equalised or Progressive system, where the letter checks are shorter and equal in size, but carried out more frequently. They are also known as de-phased letter checks. Where a larger C-Check would for example be divided into smaller more frequent checks C1, C2, C3, and C4. The normal block maintenance approach gives more fixed preparation and a reasonably low variability of planning and control. However, the aircraft may also be out of service for long periods and large gaps between checks occur. In the case of the equalised system many tasks are short enough to be carried out overnight, when no flights are scheduled. However, non-routine maintenance could provide problems and some tasks are repeated, bringing additional costs. For each operator the approach and MPD result in an initial maintenance schedule as illustrated in Figure 2.2. For each aircraft type in the fleet, the check intervals are given in flight hours or calendar days.



	747-400	**A300B4**
Transit check	At each stop whenever a/c is in transit	
Daily check	Before first flight or whenever a/c is on ground for more than 4 hours	
"A" check	Every 600 FH	In 4 parts (A1, A2, A3, A4) Every 385 FH or 11 weeks
"B" check	In 2 parts (B1, B2) Every 1200 FH	None
"C" check	In 2 parts (C1, C2) Every 5000FH or 18 months	In 2 parts (C1, C2) Every 3000 FH or 18 months
"D/HMV" check	First check done between 25k & 27.5k FH. Subsequent every 25k FH or 6 years	Every 12K FH or 4 years

Figure 2.2: Operator Initial Maintenance Schedule for Boeing 747-400 and Airbus A300B4 from Kinnison et al. [49]

2.2. Current Practice

From the initial maintenance schedule and MPD, an MRO can develop the long-term master schedule for the maintenance checks. When creating a simple maintenance schedule, certain operational constraints need to be taken into consideration. The planned maintenance tasks, for example, cannot exceed the workload for the available workforce and the planning of checks is limited by hangar capacity. It is also readily clear that together with capacity constraints the checks should adhere to deadlines set by the threshold interval of the initial maintenance schedule, otherwise an aircraft might need to be grounded. In practice each operator has more complex constraints, such as the following:

- During peak-periods no heavy-maintenance checks are allowed, in order to maximize utilization of the aircraft during these periods;
- Check interval deadlines can be extended, with a certain tolerance. However this tolerance in FH/FC needs to be subtracted from the next interval and approval from airworthiness authority is necessary;
- During weekends and bank holidays the work on heavy-maintenance checks is stopped;
- Two C-checks cannot start at the same day

The addition of these complex constraints make the scheduling of aircraft maintenance checks a difficult and challenging task. The scheduling of aircraft maintenance has been dependent on a manual scheduling approach, which relies mainly on the experience of a scheduler. In the early 70's using this manual approach, it took maintenance planning personnel several weeks to create a schedule [18]. In order to speed up this process Air Canada developed an aircraft maintenance operations simulation model (AMOS). This simulation model is described in Boere [18], and focuses on improving maintenance efficiency and the reduction of labor and material cost. In the paper the scheduling problem is described as a discrete integer programming problem and takes into account many of the constraints and conditions listed above. However, the solution approach is a priority based simulation-heuristic. This approach is similar to the manual planning approach, in which an experienced scheduler has to decide the optimal schedule, shifting checks until a feasible solution has been found. The simulation aspect however, together with the introduction of a lower utilization bound before a new check can be scheduled, reduced the time to develop a long-term maintenance schedule of 5 years from several weeks to several hours [18].

Many operators have since developed or adapted a similar approach to the scheduling of long-term maintenance and more overall integrated tools are being developed, like the fleet-planner IFS Maintenix tool [44]. All the tools however are heavily reliant on experience and manual input. In addition to the reliance on a manual approach, the developed schedules also need to be revised frequently due to the inherent stochastic nature of non-routine maintenance tasks.

2.3. Deterministic Scheduling Models

The complexity of large size scheduling problems has been discussed in Papadimitriou et al. [37]. The problem is non-deterministic in polynomial-time (NP) hard([65]), making it difficult to solve large scale problems with exact methods. The challenge in the scheduling of aircraft letter checks especially, is the fact that they are interdependent combinatorial optimization problems. Scheduling a check at a certain date has an impact on aircraft utilization in the future and consequently also influences the requirements on future maintenance checks.

In academic literature there has been relatively little focus on the long-term planning of aircraft maintenance. In Etschmaier and Franke [32] an out-of-kilter algorithm [67] was introduced to minimize maintenance cost and in Bauer-Stämpfli [11] a dynamic programming approach was developed. Both methods were deemed not suitable by Boere [18] for the environment of Air Canada when developing the previously mentioned simulation model. Furthermore, the author deemed an optimization technique to be impractical, due to the fact that an optimal solution can become obsolete with a change in environment and aircraft utilization. More recently, a zero-one integer programming model was used with the commercial solver CPLEX [43] in Yan et al.[92]. Deng et al.[25] propose a forward induction dynamic programming approach. To deal with the problem of a multi-dimensional action vector, a priority based solution is used and to further reduce the number of final states a thrifty algorithm is incorporated as well as discretization and state aggregation.

The short-term planning of aircraft maintenance however, has received a lot more attention in literature through the aircraft maintenance-routing problem. This is mainly due to the fact that optimizing the maintenance schedule has benefits only visible in the long term, where the optimization of short-term activities leads to direct cost savings and profits; which therefore attract the main focus of MROs and airlines. For the routing problem aircraft are assigned to maintenance checks after a certain numbers of flight hours according to their tail number. When assigned to a check, the aircraft is grounded at the maintenance station for at least the duration of one night[36]. The problem is often solved with integration of other airline scheduling sub-problems. An integrated problem of maintenance routing and flight scheduling is, for example, addressed in [34, 52, 53] as well as the integration with the fleet assignment problem in [10, 33, 39].

2.3.1. Objective Functions

In the literature of short-term scheduling of maintenance several objectives are encountered, that can be broadened to the overall scheduling of aircraft maintenance:

- **Minimize Costs**: Since maintenance cost is a significant part of the total operating cost of an airline, minimizing direct maintenance costs is an objective often found in literature. In Sriram [80] the problem of scheduling maintenance is formulated as an integer multi-commodity network flow model with the objective to minimize total costs of type A and type B maintenance checks. In Moudani and Mora-Camino [62] the maintenance scheduling and fleet assignment problem are combined with the objective to minimize the total operating costs of commercial and non-commercial flights. A greedy heuristic is used to solve the maintenance scheduling problem. Other examples where the objective is to minimize costs can be found in Tab. 2.1

- **Maximize Utilisation of Intervals**: Another objective encountered in the maintenance-scheduling problem, is to maximize the utilisation of the maintenance intervals. This indirectly reduces the number of maintenance checks over the long-term and days an aircraft is grounded. In Matilla and Virtanen [60] a simulation-based optimization technique is used in order to schedule the periodic maintenance for a fleet of fighter jets. The efficiency of the developed schedules are then assessed based on the average fleet utilisation. In Kozanidis and Skipis [52] a mixed integer bi-objective linear programming model is used to maximize aircraft availability and residual flight time for a fleet of fighter aircraft over the considered time-horizon. Other examples where the objective is formulated as a maximization of the residual flight time can be found in Tab. 2.1

- **Minimize Aircraft unavailability**: The minimization of aircraft unavailability is very similar to the maximization of the utilisation of maintenance intervals. Inasmuch that it indirectly decreases the number of maintenance checks in the long-term and number of days on the ground. This objective is encountered in Boere [18] and Deng et al.[25]. In Başdere and Bilge [12] the total unused legal flying time of critical aircraft is minimized with several solution techniques.

From Tab. 2.1 it is clear that the minimization of maintenance costs is the most encountered objective. However, when considering long-term scheduling of maintenance checks the available cost data is often confidential or incomplete and hard to relate to check types.

Table 2.1: Objective Functions encountered in maintenance scheduling literature

Objective Function	**Citations**
Minimize Costs	[5, 7, 26, 46, 62, 64, 74, 79, 80, 93]
Maximize Utilisation of intervals	[6, 34, 51–53, 59, 60]
Minimize Aircraft unavailability	[12, 18, 25]

2.3.2. Resource Constrained Project Scheduling

Since relatively little literature exists on the scheduling of aircraft maintenance, it is relevant to look into more general scheduling problems in literature. The RCPSP is applicable to any scheduling problem with prerequisite requirements and limited resources. Since this covers a large scale of problems in industry, applications can be found in many different sectors: in McKendall et al. [45] an adaption of the RCPSP is used to schedule maintenance in nuclear power plants; Zhou and Zhong [96] adapt the RCPSP to a train timetabling problem; In Chen and Weng [21] project scheduling in construction is adressed. A hybrid flow shop scheduling problem in steel production based on the RCPSP is described in Voß and Witt [87]; and in Roland et al. [69] the problem is adapted to the scheduling of operating rooms under human resource constraints.

The general formulation of the RCPSP considers a set V of n activities ($V = \{A_0, A_1, \dots, A_{n+1}\}$), where A_0 represents the starting dummy variable and A_{n+1} the ending. The set $A = \{A_1, \dots, A_n\}$ represents the jobs that have to be scheduled. The elapse time of each task is described by vector p in $\mathbb{N}^{n+2}$, where p_i represents the elapse time of activity A_i. The set E describes the scheduling relations between tasks, inasmuch that $(A_i, A_j) \in E$ indicates that activity A_i has to occur before activity A_j. These relations are captured in a graph $G(V, E)$, where the arcs correspond to precedence relations and activities to nodes. Each job (A_i) has a corresponding starting time, S_i in $\mathbb{R}^{n+2}$, and is run till completed. In order to run the activities a set of q renewable resources, $R = \{R_1, \dots, R_q\}$, is needed. The availability of the resources are described by a vector B, such that B_k is the availability of resource R_k. The total makespan of the schedule is defined by the start of the end task S_{n+1}, which is a dummy variable, and therefore the objective function can be mathematically described as in Eq. 2.1. Eq. 2.2 describes the precedence constraints and Eq. 2.3 describes the resource constraints.

Objective Function:

$$Minimize: \qquad S_{n+1} \tag{2.1}$$

Subject to:

$$S_j - S_i \geq p_i \qquad \forall (A_i, A, j) \in E \tag{2.2}$$

$$\sum_{A_i \in A_t} b_{ik} \leq B_k \qquad \forall R_k \in R, \quad \forall t \geq 0 \tag{2.3}$$

However, this basic formulation of the problem is very limited, and for real-life applications the model needs to be adapted. In literature, the vast majority of the RCPSPs focus on minimizing the makespan of a particular schedule, some variations exist that focus on minimizing cost, and some that consider multi-objectives [16]. Variations are more commonly accommodated by altering constraints and the set up of the model. This leads to several types of the RCPSP.

The most common variation in the RCPSP, is that of single- and multi-mode applications. In the Multi-mode version of the problem (MRCPSP) an activity can have a different elapse time, based on the amount of resources that are allocated to it, where the single-mode problem has a fixed duration. The MRCPSP is mostly used in the machine job-shop scheduling problem where several heuristic and exact solutions have been proposed [4, 66, 72]

Another distinction between RCPSP models is the possibility to add temporal constraints. These constraints limit the possible start and end times of activities to a predefined time window. Drezet and Billaut [29] deal with this problem and introduce a model in which jobs have a start and finish deadline. The model also includes a variation in the amount of resources that are required over time.

In some cases resources can be non-renewable or "used-up" during the schedule. In Slowinski [82] it is stated that often real-life projects make use of renewable as well as non-renewable resources.

The problem is formulated as a multi-objective linear programming model and the distinction of preemption is introduced, in which activities can be stopped when not yet completed, and continued at a later date.

2.3.3. Solution techniques

As discussed earlier, the scheduling of aircraft letter checks is NP hard. This is also the case for the RCPSP. This makes large scale problems difficult to solve with exact methods. In literature, several different techniques have therefore been encountered. These techniques range from exact methods to approximation methods and are discussed in this section.

Mathematical Programming

Since the problem is NP hard an exact solution method is only tractable for smaller size problems. However, the solution from the exact methods for a smaller size problem can be used as a benchmark for approximation techniques and are therefore still useful for the problem. Krause et al. [54] for example, use exact methods to benchmark evolutionary algorithms for the scheduling of a pipeline network.

A model often used in mathematical programming, is Mixed-Integer Linear Programming (MILP), which when solved with an exact method is capable of finding an exact solution for the problem. Commercial Linear Programming (LP) solvers such as CPLEX or Gurobi often have the best performance and make use of the Branch-and-Bound algorithm [41]. Papadakos [34] discusses the problem of maintenance-scheduling for a fleet of fighter jets with the purpose of maximizing average fleet availability. The problem is formulated as a MILP and is solved by an exact solution algorithm, which initially finds an upper bound on the optimum and then slowly converges by reducing the bound in a stepwise manner, until a solution is found that attains this bound. The algorithm shows significant computation time improvements in several test cases. However, when slight alterations are made to the model, the computation time increases significantly.

Beliën et al. [13] introduce a branch-and-bound enumeration algorithm for the integrated staffing and scheduling of line-maintenance in order to solve the problem in acceptable computation time. Each node in the branch-and-bound tree represents a distinct combination. However, in each node, the MILP is limited on computation time and thus not solved to optimality. In order to find a feasible solution, extra constraints are added and a limit on total computation time has to be set. Gabteni and Grönkvist[33] introduce column generation and constraint programming in order to reduce computational time and find close-to-optimal solutions for the tail assignment problem. The pricing algorithm from the column generation is used with constraint programming to model the maintenance constraints. Other techniques used with the MILP formulation found in aircraft maintenance literature are: Benders' decomposition [39], Branch-and-price [10], and Lagrangian relaxation [38]

Another mathematical programming approach is dynamic programming. Dynamic programming was first introduced by Bellman[28], and divides a complex large-scale problem into multiple stages and states. These reduced sub-problems are solved to find an optimal policy according to the Bellman optimality principle[15]. However, the direct implementation of DP in real-world applications is usually limited by the "curse of dimensionality" [14] and the "curse of modeling" [17]. Moudani and Mora-Camino [62] therefore introduce a hybrid dynamic programming approach, which solves the fleet assignment problem using dynamic programming, together with greedy-heuristics to address the sequential maintenance scheduling problem. The hybrid approach is meant for an on-line decision support system and focuses on improving total available flight hours. However, computational time does still increase significantly when considering larger scale problems. In order to combat the high computational effort,Deng et al.[25] propose a forward induction dynamic programming approach. To address the issue of a multi-dimensional action vector, a priority based solution is used and to further reduce the number of final states a thrifty algorithm is incorporated as well as discretization and state aggregation. Dynamic Programming requires a model of the environment in the form of a Markov Decision Process (MDP). Reinforcement or Q-learning approaches also solve to find an optimal policy according to the Bellman optimality principle, but do not require a model of the environment ("model-free"). Matilla and Virtanen [59] propose a λ-SMART algorithm that solves for Bellman optimality while simultaneously learning optimal Q-factors. The approach solves the optimal maintenance policy for maximum fleet availability of fighter aircraft and uses an ϵ-greedy strategy to explore other solutions.

Meta-heuristics
The primary solution techniques found in literature are meta-heuristics. The different meta-heuristics can be divided into local search heuristics, population-based heuristics, and learning heuristics.

The first type of local search heuristic that is often used is Tabu search. It was first introduced in Glover [35] and starts at an initial solution from which a set of neighbors is created, based on acceptable moves. The algorithm then chooses the best solution out of the set of neighbors and moves to it. Each iteration this process is repeated. El-Amin et al. [30] use Tabu search for the scheduling of preventive maintenance of electric generating units. However, the proposed algorithm requires a large number of iterations. Next to that, the optimality of the solution needs to be assessed by comparing it with exact benchmark results with no guarantee of an optimal solution. In order to improve the local search heuristic, a Simulated Annealing (SA) approach is often adapted. Simulated Annealing works relatively similar to Tabu search. From the neighborhood of an initial solution a new solution is sampled. If this solution is better (according to an evaluation function) than the present solution, it is established as the new solution. However, with the SA approach there is a chance of accepting a solution that performs worse than the present solution. Saraiva et al. [75] adapt a SA approach for the same generator maintenance scheduling problem as in El-Amin et al. [30]. The algorithm is tested against two case studies and can provide feasible maintenance schedules. However, the number of iterations necessary is still very large and the solution is far from optimal.

Population-based meta-heuristics are encountered often in scheduling literature. One of these type of heuristics is Ant Colony Optimization (ACO) and falls under swarm intelligence algorithms. Ant colony optimization is based on the natural phenomenon of how an ant colony locates food. Ants leave their nest in a random walk and leave a trail of a chemical substance called "pheromones", when an ant finds food it follows the trail back to its home, reinforcing that trail with more pheromones. An ant chooses based on the concentration of pheromones, resulting in the emergence of the shortest path. ACO heuristics use a multi-agent system in which a solution to the problem is found by each ant based on the collective experience of the colony. Khalouli et al [48] adapt the ACO technique for the scheduling of preventive railway maintenance. In the proposed algorithm each ant first creates an arraignment of maintenance opportunities based on a probability state transition rule. This process is iterated until all opportunities have been selected and stored into a Tabu list. From the list of opportunities a maintenance schedule is created based on constraints and a greedy heuristic. At the end of each iteration the local and global pheromone trails are updated based on a pheromone evaporating value and the best solution, respectively. Several test cases show promising results on solution effectiveness and computational time. It is however, a relatively new technique, and the time to converge can be uncertain [88].

The most widely used population-based meta-heuristic in scheduling literature is the Genetic Algorithm (GA). The concept of a GA was first introduced in Holland [40] and is an adaptive method based on the genetic processes of biological organisms. More specifically, it is based on the theory of evolution, in which populations evolve over many generations based on survival of the fittest and natural selection. A Genetic Algorithm starts with an initial population of possible solutions. Of each of the solutions the "fitness" to the problem is determined based on an evaluation function and constraints. New possible solutions are generated by selecting the best solutions from the current "generation", and by using cross over operations in order to produce a new set of solutions. Also, a possibility of mutations to the solution are incorporated for the new generation. The new generation will contain more solutions with characteristics from the best members of the previous iteration. Over many generations, the solution converges to an optimum, when designed properly. Mutation ensures exploration of other solutions and prevents getting stuck in local optima. Regarding aircraft maintenance, a GA is used in Yang and Yang [93] to schedule maintenance opportunities based on an original flight plan. The GA is very basic, but a simulation experiment shows the possibility of the algorithm to come up with feasible maintenance schedules while minimizing the costs. Quan et al. [68] use a more complex genetic algorithm to address a multi-objective preventive maintenance scheduling problem. Kleeman and Lamont [50] also address a multi-objective problem, that of scheduling heavy-maintenance on aircraft engines. Experimental results show the possibility of a GA to efficiently solve the scheduling of maintenance while obtaining a "good" solution. One of the main downsides of the GA however, is the great computational effort that is required when the problem gets more complex. When considering the more general RCPSP, Elshaer [31] compares several GAs introduced in literature for solving the RCPSP.

The main research direction in the last years is the hybridization of meta-heuristics. The usual way a hybrid meta-heuristic algorithm is developed combines local-search methods with population-based meta-heuristics, and fall under the group of integrative hybrids. When executing different meta-heuristics in parallel, the resulting algorithm falls under the collaborative hybrids.

Within the integrative hybrids, it is possible to make a distinction of two types: single-solution, and population-based hybrids. For single-solution hybrids an auxiliary hybrid is embedded into a main meta-heuristic in order to diversify the search. Zamani [95] combines cross-over operations from GAs with local search methods. The main difference with a general GA is the fact that it concentrates on the continious improvement of one good solution. For population-based solutions a local search method is embedded into a population-based meta-heuristic. Valls et al. [86] integrate the double justification local search method in various different meta-heuristics, such as Genetic Algorithms and Simulated Annealing. Several experiments show an improvement in the scheduling solution while not increasing computation time. In order to reduce computation time, Thammano and Phuang [84] combine Tabu search (TS), Simulated Annealing, and Genetic algorithm heuristics. Where a new population is generated in two parts: the first 80% is generated through standard GA cross-over methods, while the second 20% is generated by an integrated TS-SA search. Bench-mark data sets where used in order to evaluate the algorithm, showing promising results regarding the RCPSP.

When considering the class of collaborative heuristics the main distinction is that between sequential and parallel hybrids. Sequential hybrid meta-heuristics execute each method in a sequential manner, where the solution of the previous method is used. In parallel hybrids, the methods are executed in parallel and communicate with each other. The main focus of sequential hybrid methods has been on coupling GA with local search methods. Tseng and Cheng [85] combine the previously mentioned ACO with GA in a sequential manner. Here the initial population of the GA is found by ACO, where the execution of the GA updates the pheromone levels in the ACO. The proposed method is compared with other methods by testing with bench-mark data sets. Another promising sequential hybrid method is found in Agarwal et al. [1]. Here the learning based Artificial Neural Network (ANN) heuristic is combined with GA. This Neurogenetic approach feeds the best solution of each approach into each other. Zheng and Wang [57] propose a parallel hybrid method for the RCPSP. They introduce a multi-agent optimization algorithm, which is based on a multi-agent system and swarm intelligence. Wei et al.[89] combine GA with SA for the flow shop scheduling problem in a sequential manner.

Overview
An overview of the main independent techniques and formulations encountered in literature can be found in Tab. 2.2. Here the main drawbacks are briefly summarized and relevant papers in aircraft maintenance and scheduling are listed.

Table 2.2: Overview of solution techniques and formulations found in literature with their main drawbacks

Solution Technique/-Formulation	**Main Drawbacks**	**Scheduling papers**
Mixed-Integer Linear Programming	For large-scale NP hard problems intractable due to high computational effort; Non-Convexity	[7, 26, 34, 52]
Branch-and-Bound	For large complex problems, the amount of nodes can be very large, requiring very high computational effort	[13, 77, 90]
Column generation	More complex to formulate column generation problem in order for it to be benificial	[23, 33, 64]
Constraint Programming	High computational effort; Limited programming libraries available	[6, 33]
Benders' decomposition	Time-consuming iterations;Slow convergence at the end of the algorithm; Poor feasibility and optimality cuts	[39]
Dynamic Programming	Number of states grows exponentially in n-dimensional state space (Curse of dimensionality); Curse of modeling	[25, 62]
Tabu Search	Highly dependent on initial solution; Serial iterative search process so for large-scale problems high computational effort; Optimal solution is not guaranteed	[8, 24, 30]
Simulated Annealing	High computational time, especially if cost function is extensive to compute; Tailoring necessary for different constraints; Optimal solution is not guaranteed	[2, 75]
Ant Colony Optimization	Difficult to set initial design parameters; No way to deal with problems of scattering; Possibility of premature convergence in local optimum; Optimal Solution is not guaranteed	[48, 88]
Genetic Algorithm	Difficult to set initial design parameters; Strong dependence on initial solution; High computational effort for large complex problems; Optimal Solution is not guaranteed	[40, 50, 60, 68, 93]

2.4. Uncertainties

Uncertainty occurs when there is an absence of information or comprehension about a problem and its possible consequences. It is possible to understand, manage, and reduce uncertainty, but it cannot be completely removed [71]. Aircraft heavy-maintenance is a complex dynamic project and possesses varying degrees of uncertainty, which can influence the execution and creation of a maintenance schedule. For the long-term planning of aircraft heavy-maintenance, it is therefore important to identify these uncertainties and their effect on the maintenance schedule.

2.4.1. Heavy-maintenance

When performing a heavy maintenance check a set of routine tasks are defined in the scheduled maintenance program, the so called routine maintenance activities. However, inspection of the aircraft can find unexpected damage or failures for which unscheduled activities must be performed. These non-routine findings, or unscheduled maintenance, are common occurrences in maintenance checks and can increase the elapse time of a maintenance check, impacting the maintenance schedule, which in turn can affect maintenance cos and lead to disruptions. Samaranayake [73] recognizes that, uncertainty encountered in aircraft maintenance, coming particularly from non-routine/unscheduled maintenance, affects the planning and scheduling of aircraft maintenance. The author realises the importance of non-routine findings and states that about 50% to 60% of the maintenance workload result from these findings. A more recent case study in Dinis [27] shows that this number can even be higher and increases with aircraft age. This is confirmed by Fig. 2.3, where for the C-Check of an aircraft type of an European MRO over its lifetime, the ratio between scheduled and unscheduled maintenance can be seen.

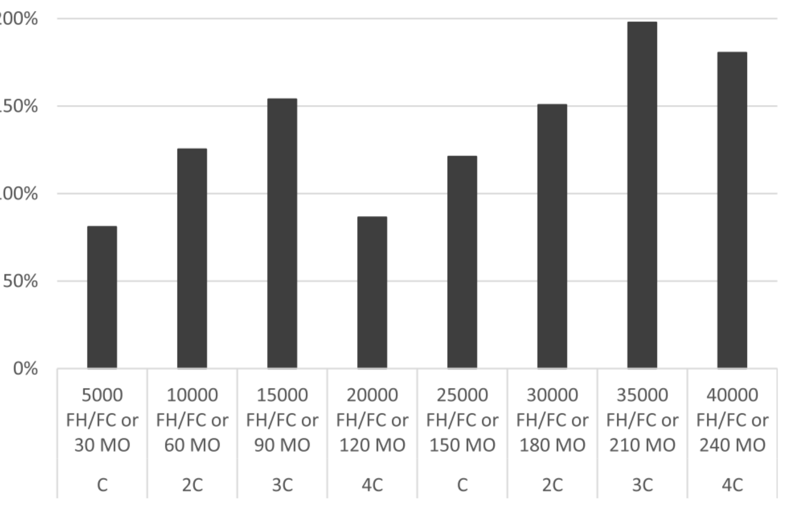


Figure 2.3: Ratio of routine and non-routine workload for aircraft C-check [27]

The stochastic nature of this non-routine maintenance can hamper accurate planning and resource forecast and result in continuous adjustments to the initial maintenance schedule. This can result in backlog can affect maintenance cost, quality of service, and the maintenance schedule. The problem often lies in the lack of historical data, such that describing the uncertainty with a probability distribution is difficult. However, when considering the long-term planning of aircraft maintenance checks these non-routine findings influence the duration of maintenance checks and based on historic data a distribution could be estimated.Next to non-routine findings, variable routine tasks can also influence the duration of a heavy-maintenance check. These variable routine tasks can consist of deferred maintenance from other checks, Airworthiness Directives (AD), or Service Bulletins (SB).

As discussed in Sec. 2.1 the need for scheduling an aircraft maintenance check is not only based on calendar days, but also on flight hours and flight cycles. The utilization of aircraft is therefore also very important when creating a long-term maintenance schedule. Since this schedule considers a time horizon of about 5 years [18] the utilization of aircraft can change over the years, based on for example the network structure and objectives of the airline. Point-to-point carriers, for example, tend to have a higher utilization than hub-and-spoke carriers. Next to that the utilization is also affected by the downtime necessary for aircraft maintenance.

Other factors that can influence the heavy maintenance are workforce and spare parts availability, but are more relevant when considering the short-term planning instead of long-term.

2.4.2. Modelling Uncertainty

As discussed uncertainty can occur in different degrees in a complex project. Based on the knowledge, different techniques of incorporating the uncertainty can be used.

Bounded Form

When little information is available about the uncertain parameter an accurate characterization in the form of a probability distribution is not possible. Sometimes only error bounds can be obtained, since, in that case, there is no need for information about the type of uncertainty. When considering a bounded form description, intervals can be used to incorporate uncertainty. The parameter can then be described by an interval with a lower and upper bound in interval $\theta \in [\theta_{min}, \theta_{max}]$. All possible realizations of the uncertain parameter are represented by these bounds, which can be determined from historical data.

Probability Description

The most common representation of an uncertain parameter is the use of probabilistic models. This is often only possible when more information about the behaviour of uncertainty is available. The probabilities of events describe the uncertain parameter. The probability distribution function of the uncertain parameter (random variable) X can then be defined by $F(a) = P(X \leq a) \quad$ for $-\infty < a < \infty$. This can be done for both a continuous or discrete random variable, although most uncertain parameters in aircraft maintenance, such as failure rates and non-routine findings, are most commonly described as discrete parameters. When incorporating this into scheduling models a continuous probabilistic function needs complicated integration schemes,where a discrete representation needs a large number of scenarios.

Fuzzy Description

When historical data is not readily available uncertainty can be modelled with Fuzzy sets and Fuzzy logic. Zadeh [94] introduced the use of fuzzy logic with possibility theory. In classical set theory the membership function μ_A can be 0 or 1 indicating whether it belongs to the set. In fuzzy sets however it can take a value between 0 and 1, indicating it partially belonging to the set. Where a high number indicates a high possibility and a low number a low possibility. Fuzzy sets have the benefit that there is no need lots of different scenarios as with a discrete probabilistic representation. An overview of fuzzy representation in scheduling can be found in Slowinski and Hapke [70] that gathers significant work in fuzzy scheduling.

2.5. Stochastic Scheduling Models

Currently, to the best of the authors knowledge, there is no literature available that addresses the long-term scheduling of aircraft maintenance checks while taking uncertainty into account. However, when considering the general problem of aircraft maintenance scheduling some papers exist that do. Mattila and Virtanen [59] take into account uncertainty in failure rates and maintenance duration when scheduling maintenance for a fleet of fighter jet. The uncertainty parameters are modelled 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 [78] use the random effects Weibull regression model in order 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 more and more focused on incorporating uncertainty in the models

2.5.1. Stochastic Scheduling Types

When considering scheduling, there is a distinction between reactive and preventive scheduling. Reactive scheduling focuses on modifying an already existing schedule based on the realization of uncertain events. On the other hand, preventive scheduling deals with uncertainty beforehand. As discussed in Sec. 2.4.2 several different techniques exist to incorporate uncertainty. For preventive scheduling a distinction can be made between: stochastic scheduling, robust optimization, and fuzzy programming.

When considering the stochastic programming approach the objective function is mostly based on an expectation. Take the RCPSP for example, the most common objective function is minimizing makespan. However, when considering the stochastic version the objective function can become minimizing the expected makespan. With stochastic scheduling a set of different scenarios is assessed and the solution to the problem consists of a number of different schedules. For each possible scenario a different schedule may arise. The solution to the problem therefore becomes a policy how to dynamically schedule based on different scenarios. For large scale problems the number of scenarios can be vast, where sample average approximation could be used to reduce the number of scenarios. Most problems are modelled as a two- or multi-stage stochastic programming model with recourse. In these type of models a baseline schedule is created and when there is a realization of uncertainty recourse actions are taken to adapt. Lamas and Deulemeester [55] adapt a stochastic model with chance constrained programming aproach for the RCPSP with uncertain task duration. This chance constrained version of the RCPSP finds a schedule of minimal makespan such that the constraints (Eq. 2.2 and Eq. 2.3) are feasible with a predefined confidence level. Sample average approximation (SAA) is used, which replaces draws task duration from an empirical distribution. The problem with this however, is that the solution to the SAA might be infeasible and non-optimal to the original problem, depending on sample size.

With robust scheduling the focus lies on building a schedule where the effect of disruptions is minimized. It tries to ensure that the predictive schedule is feasible for a vast number of scenarios, while maintaining a high performance. Robustness can be defined as solution robust and model robust. If a solution is solution robust, the variation in the solution is minimal with changing scenarios. If a solution is model robust, feasibility is maintained for a vast number of scenarios. The underlying idea behind robust optimization is to find a solution which is robust to changes in the environment introduced by uncertainty. Several robustness approaches exist in literature. Mogaadi and Fayech [61] for example, use a min-max approach for the stochastic RCPSP. Here the "fitness" of a solution is assessed by looking at multiple scenarios and finding the maximum makespan over all scenarios. Chaari et al. [20] use a robustness criterion based on a set N of disrupted scenarios and use it as an evaluation function seen in Eq. 2.4 where f_I is the performance of the initial scenario I, and f_{ζ_i} the performance of one of the disrupted scenarios $\zeta_i(I)$. This gives a bi-objective evaluation, where $\lambda \in [0,1]$ expresses the importance of the initial scenario and the deviation of the disrupted scenarios (degree of risk).

$$f_r(x) = \lambda f_I(x) + (1-\lambda)\sqrt{\frac{1}{N}\sum_{i\in N}\left(f_{\zeta_i(I)}(x) - f_I(x)\right)^2} \tag{2.4}$$

Fuzzy programming is not common in scheduling literature, but can be useful when information on probability distribution is not readily available. Balasubramanian and Grossmann [9] apply a fuzzy set approach to the flow shop scheduling problem with uncertain duration. The approach reduces computation time significantly and results in most likely, pessimistic and optimistic representations. The uncertainty is described with triangular fuzzy numbers. The benefit of this representation is that the computation of the objective function is easier, which makes the application of heuristic search algorithms a good option to solve within reasonable computing time.

2.5.2. Solution Techniques

As with the deterministic scheduling case, Sec. 2.3.3, several solution techniques can be found in literature for the scheduling under uncertainty.

Mathematical programming

Due to the different scenarios, the use of exact methods for larger scale problems can be even more intractable. Still, for some smaller cases exact methods can be found in literature. The most representative exact solution approach to the stochastic RCPSP is, as with the deterministic case, Branch-and-Bound. Leus and Herroelen [56] propose a Branch-and-Bound algorithm for resource allocation with variable activity durations. De Bruecker et al. [19] use a model enhancement heuristic for robust workforce scheduling of aircraft maintenance. A Branch-and-Bound algorithm is applied for the initial deterministic MILP, after which uncertainty is incorporated by running Monte Carlo simulations. From the simulations extra constraints are added to the model, after which the process reiterates. Lamas and Deulemeester [55] adapt a stochastic model with chance constrained programming aproach for the RCPSP with uncertain task duration. In order to solve the problem they adapt a Branch-and-Cut algorithm to reduce computation time. (Stochastic) Dynamic Programming is also an approach that is encountered in literature. The markov chains can be used to model uncertain parameters, but large scale problems result in an enormous state space, so often adaptations are necessary. Choi et al. [22] use DP, where the state space is confined through heuristics to solve the stochastic RCPSP with uncertain duration. Tereso et al. [83] suggested approximation schemes in order to reduce computational effort for the same problem.

Meta-Heuristics

The most commonly used technique for the stochastic scheduling problem is the previously discussed genetic algorithm. Chaari et al. [20] use a GA to solve the robust hybrid flow shop scheduling. The fitness of a schedule is evaluated according to Eq. 2.4, with the objective to minimize makespan for an initial scenario and the difference caused by disruptions. The GA was validated through simulation and showed that the evaluation method can incorporate a high degree of uncertainty, while trading-off robustness and optimality. Al-Dhaheri et al. [3] use a simulation-based GA approach to incorporate uncertainty in the scheduling of quay cranes. In the algorithm, the performance of each solution is evaluated based on an embedded simulation model using Monte Carlo sampling. An overview of the way the GA and the simulation are embedded can be seen in Fig. 2.4

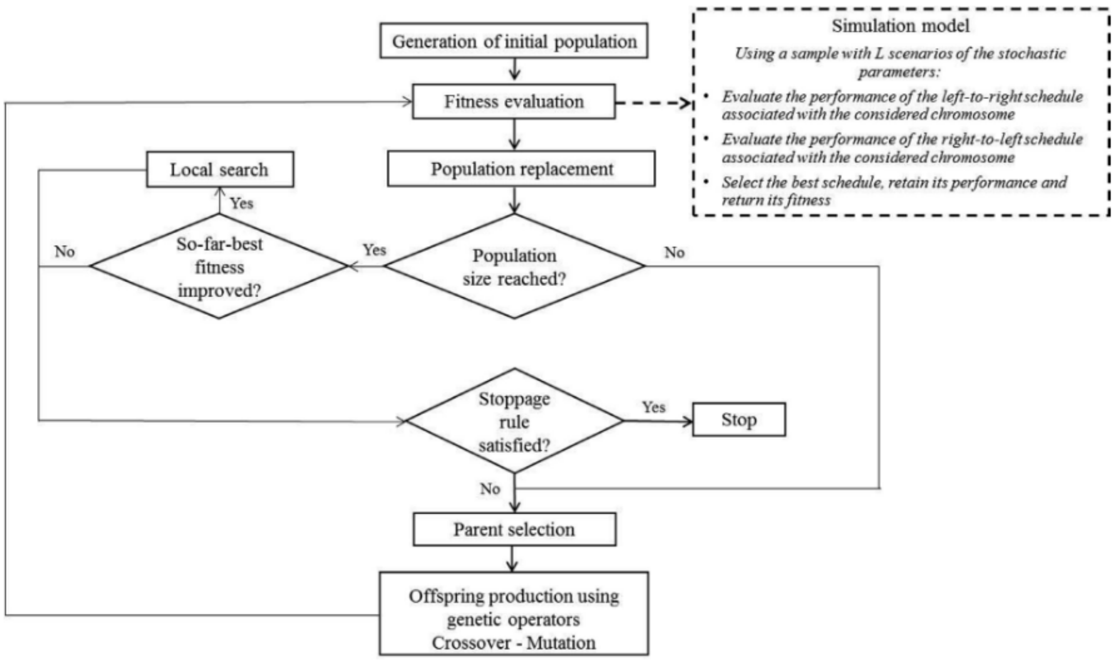

Generation of initial population
Fitness evaluation
Simulation model
Using a sample with L scenarios of the stochastic parameters:
• Evaluate the performance of the left-to-right schedule associated with the considered chromosome
• Evaluate the performance of the right-to-left schedule associated with the considered chromosome
• Select the best schedule, retain its performance and return its fitness
Local search
Population replacement
Yes
No
So-far-best fitness improved?
Yes
Population size reached?
No
Stoppage rule satisfied?
Yes
Stop
No
Parent selection
Offspring production using genetic operators Crossover - Mutation

Figure 2.4: Genetic Algorithm incorporating uncertainty through Monte Carlo Sampling [3]

Local search heuristics, although less, are also encountered in literature. In Pan and Yeh [63] for example, a Simulated Annealing algorithm is used together with Fuzzy operations in order to solve the RCPSP with uncertain task duration.

As is the case with the deterministic scheduling models, current research trends show a direction towards hybridization of meta-heuristics. Recently, Kerkhove and Vanhoucke [47] for example, have introduced a hybrid SA and dedicated algorithm together with discrete event simulation to schedule offshore construction projects under uncertain weather conditions. The discrete event simulation is used to generate weather simulations, where the SA and Dedicated algorithm combine as optimization heuristics. Masmoudi and Haït [58] combine a greedy algorithm with GA for project scheduling under uncertainty. The uncertain parameters are represented by Fuzzy modelling. The algorithm is applied to the task scheduling of civil helicopter maintenance, where task duration are based on expert opinion

For the algorithms that are used in the stochastic models, the same drawbacks as presented in Tab. 2.2 apply.

3 Discussion

Having explored the state of the art in Sec. 2, it is important to discuss the findings and how they are applicable to the thesis research. Additionally, it should become clear whether a gap in literature exists with respect to the long term scheduling of aircraft heavy maintenance checks.

Firstly, in Sec. 2.1 an overview and short summary of the scheduling of maintenance letter checks is discussed. This to give a theoretical basis to how and on which indicators the periodic maintenance of aircraft is scheduled. Sec 2.2 discusses the current approach of MROs to this scheduling. Where Boere [18] is the main literature work on the development of a simulation model (AMOS) that assists airlines in the development of long-term heavy-maintenance schedules. Many operators have since developed similar approaches, but most solutions are in-house and ad-hoc tools. Some overal fleet planner tools are currently being developed, like the IFS maintenix tool [44]. However, the main drawback in current approaches is that it is reliant on experience of the operator and manual input. Revisions are also a frequent necessity due to disruptions in the original schedule.

Secondly, Sec. 2.3 discusses the main literature on the long-term scheduling of heavy-maintenance checks, aircraft maintenance scheduling literature in general, and the resource constrained project scheduling problem. It is clear that there are not many papers that regard the long-term aircraft maintenance scheduling. Boere [18] is the main work and introduces a discrete integer programming formulation of the problem. This work also introduces a number of constraints and conditions based on aircraft maintenance scheduling experience. Deng et al. [25], recently expanded these constraints and provide a list with conditions that are commonly used in airlines/MROs. The list from Deng et al. [25] can be used to adapt the integer programming formulation from Boere. [18] for the thesis research. Although little research on long-term scheduling has been found, the short-term scheduling maintenance routing problem has received a lot more attention. From these papers regarding airline maintenance scheduling in general, several objectives found in literature are discussed in Sec. 2.3.1. The objective of minimizing maintenance cost is the most common objective. However, available maintenance cost data can be unreliable and hard to associate with specific maintenance checks. Next to that, is the fact that a day out of operations is more costly than maintenance costs, so operational costs would be a more relevant objective. The objective function that is most applicable to the research is therefore the minimization of aircraft unavailability as in Boere [18] and Deng et al. [25], and consequently the reduction of number of aircraft checks in the long-term. Next, in Sec. 2.3.2, the RCPSP is described. Since the RCSPS is a basic scheduling problem with common characteristics and a broad application, state of the art literature could be relevant for the scheduling of aircraft heavy-maintenance checks. Furthermore, Sec. 2.3.3 describes the most common solution techniques found in the before mentioned literature on long-term scheduling, aircraft maintenance scheduling, and the RCPSP. From literature it is clear that the problem is a NP hard one [37, 65], therefore exact solution techniques become intractable for larger scale problems. However, the formulation of the problem as a MILP, together with commercial solvers' branch-and-bound technique, can still be used as a benchmark for approximation techniques. In this sense, smaller test cases can be developed in order for the technique to remain tractable as is done in Krause et al. [54], where a MILP approach is used as a benchmark for several evolutionary algorithms. Due to the NP hardness of the problem the most encountered solution techniques are meta-heuristics. More specifically, genetic algorithms of which a comparison for the RCPSP of several GAs is found in Elshaer [31]. The main research direction found in deterministic scheduling literature is that of the hybridization of meta-heuristics. Where Wei et al. [89] combine the advantages of the genetic algorithm with that of the local search procedure simulated annealing. GA is used to generate an near-optimal solution, after which SA is employed in order to search for a better one based on that solution.

Next, the uncertainties encountered when scheduling or performing aircraft heavy-maintenance are discussed in Sec. 2.4. Due to the dynamic and complex nature of aircraft heavy-maintenance different degrees of uncertainty can be encountered. The main uncertainty is non-routine maintenance work. Samaranayake [73], discusses the issues of uncertainty of maintenance work and states that about 50 to 60% of total maintenance work comes from non-routine findings. A recent case study in Dinis [27] shows that this number can even be higher and is is correlated with the age of the aircraft. When considering the long-term scheduling of maintenance this uncertainty of non-routine and variable-routine work can indirectly be found in the elapse time of maintenance checks. The stochastic nature of the duration of maintenance checks can hamper accurate planning and resource forecast and result in continuous adjustments to the initial maintenance schedule. This can result in backlog and can have an impact on scheduled times, cost, and even service quality. Other factors that were found that can introduce uncertainty where workforce and spare parts availability. However, this can also be found back in the uncertainty of elapse times of maintenance checks when considering long-term scheduling. The uncertainty of aircraft utilization is relevant when considering the long-term scheduling, but data and estimations are scarce. Next to that, utilization is not expected to fluctuate by a huge amount. Therefore, considering uncertainty in check duration can be the main focus of the research.

Lastly, Sec. 2.5 discusses the literature on scheduling under uncertainty. It is clear, that when regarding the scheduling of aircraft maintenance little work has been performed that considers uncertainty. Mattila and Virtanen [59] take into account uncertainty in failure rates and maintenance duration when scheduling maintenance for a fleet of fighter aircraft. However, the capability of the model to be used in actual decision making requires the solution of several different problem instances and is based on a policy based approach. When considering the RCPSP, significantly more work has been done on incorporating uncertainty. As a matter of fact it is one of the main research directions in the field. Robust approaches are popular, and when considering the long-term scheduling of aircraft heavy-maintenance, could be benificial. Especially model robust methods, in which the schedule is feasible for most possible scenarios, would reduce the need for continuous schedule revisions. Chaari et al. [20] use a bi-objective robustness criteria, in which the performance of an initial scenario is evaluated together with the performance deviation of disrupted scenarios with respect to the initial one In order to solve the stochastic scheduling problems the main method that is used in literature GA, which complies with the deterministic case. Simulation, such as Monte Carlo sampling, or sample average approximation are often used with meta-heuristics to assess the objective function in an uncertain environment. Al-Dhaheri et al. [3] for example, use an embedded simulation model to evaluate the fitness when using a GA. This method, could be further developed for the long-term scheduling of heavy-maintenance by for example incorporating the local search technique SA, as in Wei et al. [89]. Following, the research trend of hybridization of meta-heuristics.

Inasmuch, it is clear that there has been little focus on the long-term scheduling of heavy-maintenance checks in literature. Next to that, the inherent stochastic nature of the maintenance in the scheduling is not taken into account when creating an initial schedule. A robust long-term schedule could reduce the number of adjustments to the schedule, reduce cost, and give a good forecast of required resources. Especially, since the MRO spend spans on average 9.5 to 11% of the total operating cost of an airline [42], reducing these costs can be very beneficial in the long-term for airlines. The goal of developing a scheduling model that takes into account the uncertainty of maintenance check duration should therefore be the main focus of the research. In this sense the relevance of the research lies in filling this gap in literature and creating a general stochastic scheduling model framework.

4
Research Framework

After the review of the state of the art, the research framework can be set up. This in order to give direction to the project and define the scope. The main goal of the research is stated as follows:

"To develop a stochastic maintenance scheduling model capable of tractably delivering an effective long-term schedule for aircraft C-checks while taking into account the main sources of uncertainty in C-Check scheduling"

In order to achieve this goal a set of research questions are developed that act as a guideline throughout the research:

1. How can uncertainty be incorporated into the model?
   (a) What are the main sources of uncertainty regarding the long-term scheduling of aircraft C-Checks?
   (b) How can the main sources of uncertainty be modelled?
2. In what way can the model be designed?
   (a) What are the main functions of the model?
   (b) What kind of constraints need to be incorporated in a long-term scheduling model for aircraft C-checks?
   (c) What technique(s) can be used to solve the problem?
3. Is the model capable of improving the schedules compared to current practice?
   (a) How do the results of the model compare to a benchmark simulating small test cases?
   (b) Do the results of the model simulating an airline case study deliver the required output?
   (c) Are the results of the model acceptable regarding computational time, assumptions, and optimization?

5 Conclusion

There is an increasing demand on the MRO market, which currently spans around 9.5% of the total operating cost. Of this, around 70% is covered by heavy-maintenance. Reduction of these costs and insight in future capacity needs could therefore be significant for an airline. A possible solution is the optimization of the long-term schedule of heavy-maintenance checks.

In order to get a better understanding on the topic a literature review on the state of the art was performed. Current approaches to the long-term scheduling of the maintenance checks are shown to be based on operator experience and simulation models where manual selection is a major component. Next to that, the initial schedule has to be continuously revised due to the stochastic nature of aircraft maintenance. These revisions can result in backlog and can have an impact on scheduled times, cost, and even service quality. The main reason of uncertainty comes from non-routine findings, which can be around 50% of the total maintenance workload. When considering the long-term scheduling, the uncertainty introduced by this non-routine maintenance can be found back in the uncertain duration of the aircraft maintenance checks. Taking this into account when scheduling could offer more robust schedules where fewer revisions are necessary. However, in the literature, relatively few papers have been found that address the long-term maintenance schedule, and all found papers focus on the deterministic scheduling problem. Analysis of the state-of-the art of deterministic aircraft maintenance scheduling problems together with the RCPSP, show that the main research direction in solution techniques is the hybridization of meta-heuristics. Where genetic algorithms are the most common solution technique encountered. The main research direction in scheduling literature, is addressing the stochastic scheduling problem. Where, similar to the deterministic case, the main solution techniques are meta-heuristics combined with Monte Carlo simulations to incorporate the uncertainty. The relevance of the research lies thus in filling the gap in literature by creating a long-term scheduling model for aircraft maintenance checks while taking into account uncertainty. Furthermore, a robust long-term scheduling model could reduce maintenance costs in the long-term and give a reliable estimation of future resource needs for airlines.

With the knowledge from the literature review the research aim has been described as follows: *"To develop a stochastic maintenance scheduling model capable of tractably delivering an effective long-term schedule for aircraft C-checks while taking into account the main sources of uncertainty in C-Check scheduling"*. Through several research questions the project research will try to fulfill this aim.

Bibliography

[1] Anurag Agarwal, Selcuk Colak, and Selcuk Erenguc. A neurogenetic approach for the resource-constrained project scheduling problem. *Computers & Operations Research*, 38(1):44 – 50, 2011. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2010.01.007. URL `http://www.sciencedirect.com/science/article/pii/S0305054810000183`. Project Management and Scheduling.

[2] Sanjay Ahire, Garrison Greenwood, Ajay Gupta, and Mark Terwilliger. Workforce-constrained preventive maintenance scheduling using evolution strategies. *Decision Sciences*, 31(4): 833–859, 2000. doi: 10.1111/j.1540-5915.2000.tb00945.x. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-5915.2000.tb00945.x`.

[3] Noura Al-Dhaheri, Aida Jebali, and Ali Diabat. A simulation-based genetic algorithm approach for the quay crane scheduling under uncertainty. *Simulation Modelling and Theory*, 66:122–138, 2016.

[4] J. Alcaraz, C. Maroto, and R. Ruiz. Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. *Journal of the Operational Research Society*, 54 (6):614–626, Jun 2003. ISSN 1476-9360. doi: 10.1057/palgrave.jors.2601563. URL `https://doi.org/10.1057/palgrave.jors.2601563`.

[5] Hesham K. Alfares. Aircraft maintenance workforce schedulinga case study. *Journal of Quality in Maintenance Engineering*, 5(2):78–89, 1999. doi: 10.1108/13552519910271784. URL `https://doi.org/10.1108/13552519910271784`.

[6] Maliheh Aramon and J Christopher Beck. Scheduling an aircraft repair shop. 01 2011.

[7] Suaibatul Aslamiah, Siti R. Simamora, Tan Kim Hek, Novin M. Sarina, Edi L. Harahap, and Malem Karina. Integer programming model for operational aircraft maintenance routing problem with side constraints. 2010.

[8] Tonius Baar, Peter Brucker, and Sigrid Knust. *Tabu Search Algorithms and Lower Bounds for the Resource-Constrained Project Scheduling Problem*, pages 1–18. Springer US, Boston, MA, 1999. ISBN 978-1-4615-5775-3. doi: 10.1007/978-1-4615-5775-3_1. URL `https://doi.org/10.1007/978-1-4615-5775-3_1`.

[9] J. Balasubramanian and I.E. Grossmann. Scheduling optimization under uncertainty—an alternative approach. *Computers & Chemical Engineering*, 27(4):469 – 490, 2003. ISSN 0098-1354. doi: https://doi.org/10.1016/S0098-1354(02)00221-1. URL `http://www.sciencedirect.com/science/article/pii/S0098135402002211`.

[10] Cynthia Barnhart, Natashia L. Boland, Lloyd W. Clarke, Ellis L. Johnson, George L. Nemhauser, and Rajesh G. Shenoi. Flight string models for aircraft fleeting and routing. *Transportation Science*, 32(3):208–220, 1998. doi: 10.1287/trsc.32.3.208. URL `https://doi.org/10.1287/trsc.32.3.208`.

[11] H. Bauer-Stämpfli. Near optimal long-term scheduling of aircraft overhauls by dynamic programming. *AGIFORS Symposium*, 1971.

[12] Mehmet Başdere and Ümit Bilge. Operational aircraft maintenance routing problem with remaining time consideration. 235(1):315–328, 2014. ISSN 0377-2217. doi: 10.1016/j.ejor.2013.10.066. URL `https://dx.doi.org/10.1016/j.ejor.2013.10.066`.

[13] Jeroen Beliën, Erik Demeulemeester, Philippe De Bruecker, Jorne Van Den Bergh, and Brecht Cardoen. Integrated staffing and scheduling for an aircraft line maintenance problem. 40(4): 1023–1033, 2013. ISSN 0305-0548. doi: 10.1016/j.cor.2012.11.011. URL `https://dx.doi.org/10.1016/j.cor.2012.11.011`.

[14] R. E. Bellman. Adaptive control processes—a guided tour. *Naval Research Logistics Quarterly*, 8(3):315–316, 1961. doi: 10.1002/nav.3800080314. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800080314`.

[15] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966. ISSN 0036-8075. doi: 10.1126/science.153.3731.34. URL `https://science.sciencemag.org/content/153/3731/34`.

[16] Fouad Ben Abdelaziz, Saoussen Krichen, and Olfa Dridi. A multiobjective resource-constrained project-scheduling problem. In Khaled Mellouli, editor, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 719–730, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-75256-1.

[17] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1st edition, 1996. ISBN 1886529108.

[18] N. J. Boere. Air canada saves with aircraft maintenance scheduling. *Interfaces*, 7(3):1–13, 1977.

[19] Philippe De Bruecker, Jorne Van den Bergh, Jeroen Beliën, and Erik Demeulemeester. A model enhancement heuristic for building robust aircraft maintenance personnel rosters with stochastic constraints. *European Journal of Operational Research*, 246(2):661 – 673, 2015. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2015.05.008. URL `http://www.sciencedirect.com/science/article/pii/S037722171500380X`.

[20] Tarek Chaari, Sondes Chaabane, Taicir Loukil, and Damien Trentesaux. A genetic algorithm for robust hybrid flow shop scheduling. *International Journal of Computer Integrated Manufacturing*, 24(9):821–833, 2011. ISSN 0951-192X. doi: 10.1080/0951192x.2011.575181. URL `https://dx.doi.org/10.1080/0951192X.2011.575181`.

[21] Po-Han Chen and Haijie Weng. A two-phase ga model for resource-constrained project scheduling. 18(4):485–498, 2009. ISSN 0926-5805. doi: 10.1016/j.autcon.2008.11.003. URL `https://dx.doi.org/10.1016/j.autcon.2008.11.003`.

[22] Jaein Choi, Matthew J. Realff, and Jay H. Lee. Dynamic programming in a heuristically confined state space: a stochastic resource-constrained project scheduling application. *Computers & Chemical Engineering*, 28(6):1039 – 1058, 2004. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2003.09.024. URL `http://www.sciencedirect.com/science/article/pii/S0098135403002412`. FOCAPO 2003 Special issue.

[23] Jean-François Cordeau, Goran Stojković, François Soumis, and Jacques Desrosiers. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, 35(4):375–388, 2001. doi: 10.1287/trsc.35.4.375.10432. URL `https://doi.org/10.1287/trsc.35.4.375.10432`.

[24] Jorne Van den Bergh, Philippe De Bruecker, Jeroen Beliën, Liesje De Boeck, and Erik Demeulemeester. A three-stage approach for aircraft line maintenance personnel rostering using mip, discrete event simulation and dea. *Expert Systems with Applications*, 40(7):2659 – 2668, 2013. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2012.11.009. URL `http://www.sciencedirect.com/science/article/pii/S0957417412012195`.

[25] Qichen Deng, Bruno F. Santos, and Richard Curran. A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization. 2019. [*Not Published Yet*].

[26] Ulrich Derigs and Stefan Friederichs. Air cargo scheduling: integrated models and solution procedures. *OR Spectrum*, 35(2):325–362, Mar 2013. ISSN 1436-6304. doi: 10.1007/s00291-012-0299-y. URL `https://doi.org/10.1007/s00291-012-0299-y`.

[27] Duarte Dinis, Ana Barbosa-Póvoa, and Ângelo Palos Teixeira. A supporting framework for maintenance capacity planning and scheduling: Development and application in the aircraft mro industry. *International Journal of Production Economics*, 218:1–15, 2019. ISSN 0925-5273. doi: 10.1016/j.ijpe.2019.04.029. URL `https://dx.doi.org/10.1016/j.ijpe.2019.04.029`.

[28] S.E. Dreyfus. Richard bellman on the birth of dynamic programming. *Operations Research*, 50: 48–51, 02 2002. doi: 10.1287/opre.50.1.48.17791.

[29] L.-E. Drezet and J.-C. Billaut. A project scheduling problem with labour constraints and time-dependent activities requirements. *International Journal of Production Economics*, 112 (1):217 – 225, 2008. ISSN 0925-5273. doi: https://doi.org/10.1016/j.ijpe.2006.08.021. URL `http://www.sciencedirect.com/science/article/pii/S0925527307001405`. Special Section on Recent Developments in the Design, Control, Planning and Scheduling of Productive Systems.

[30] Ibrahim El-Amin, Salih Duffuaa, and Mohammed Abbas. A tabu search algorithm for maintenance scheduling of generating units. *Electric Power Systems Research*, 54(2):91 – 99, 2000. ISSN 0378-7796. doi: https://doi.org/10.1016/S0378-7796(99)00079-6. URL `http://www.sciencedirect.com/science/article/pii/S0378779699000796`.

[31] Raafat Elshaer. Solving resource-constrained project scheduling problem using genetic algorithm. *Journal Of Al Azhar University Engineering Sector*, 12:187–198, 02 2017.

[32] M. Etschmaier and P. Franke. Long-term scheduling of aircraft overhauls. *AGIFORS Symposium*, 1969.

[33] Sami Gabteni and Mattias Grönkvist. Combining column generation and constraint programming to solve the tail assignment problem. *Annals of Operations Research*, 171:61–76, 10 2009. doi: 10.1007/s10479-008-0379-1.

[34] Andreas Gavranis and George Kozanidis. An exact solution algorithm for maximizing the fleet availability of a unit of aircraft subject to flight and maintenance requirements. *European Journal of Operational Research*, 242(2):631 – 643, 2015. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2014.10.016. URL `http://www.sciencedirect.com/science/article/pii/S0377221714008376`.

[35] Fred Glover. Artificial intelligence, heuristic frameworks and tabu search. *Managerial and Decision Economics*, 11(5):365–375, 1990. doi: 10.1002/mde.4090110512. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/mde.4090110512`.

[36] Ram Gopalan and Kalyan T. Talluri. The aircraft maintenance routing problem. *Operations Research*, 46(2):260–271, 1998.

[37] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*, volume 32. 01 1982. ISBN 0-13-152462-3. doi: 10.1109/TASSP.1984.1164450.

[38] Mohamed Haouari, Najla Aissaoui, and Farah Zeghal Mansour. Network flow-based approaches for integrated aircraft fleeting and routing. *European Journal of Operational Research*, 193(2):591 – 599, 2009. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2007.11.042. URL `http://www.sciencedirect.com/science/article/pii/S0377221707011411`.

[39] Mohamed Haouari, Hanif D. Sherali, Farah Zeghal Mansour, and Najla Aissaoui. Exact approaches for integrated aircraft fleeting and routing at tunisair. *Computational Optimization and Applications*, 49(2):213–239, Jun 2011. ISSN 1573-2894. doi: 10.1007/s10589-009-9292-z. URL `https://doi.org/10.1007/s10589-009-9292-z`.

[40] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* MIT Press, Cambridge, MA, USA, 1992. ISBN 0262082136.

[41] Chung-Yang (Ric) Huang, Chao-Yue Lai, and Kwang-Ting (Tim) Cheng. Chapter 4 - fundamentals of algorithms. In Laung-Terng Wang, Yao-Wen Chang, and Kwang-Ting (Tim) Cheng, editors, *Electronic Design Automation*, pages 173 – 234. Morgan Kaufmann, Boston, 2009. ISBN 978-0-12-374364-0. doi: https://doi.org/10.1016/B978-0-12-374364-0.50011-4. URL `http://www.sciencedirect.com/science/article/pii/B9780123743640500114`.

[42] IATA. Iata's maintenance cost task force. airline maintenance cost executive commentary – an exclusive benchmark analysis. 2016.

[43] IBM. Ilog cplex optimization studio 12.7.1: Cp optimizer online documentation. URL `http://ibm.biz/COS1271Documentation`. Accessed on 03.06.2019.

[44] IFS. A new approach to maintenance planning. URL `https://www.ifsworld.com/corp/solutions/ifs-maintenix/fleet-planner/`. Accessed on 13.06.2019.

[45] Alan McKendall Jr, James Noble, and Cerry Klein. Scheduling maintenance activities during planned outages at nuclear power plants. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 15(1):53–61, 2008. ISSN 1943-670X. URL `http://journals.sfu.ca.tudelft.idm.oclc.org/ijietap/index.php/ijie/article/view/62`.

[46] Radoje Karadžić, Darko Petkovic, and Muharem Šabić. A model for the maintenance of old aircraft. *Aviation*, 16:16–24, 03 2012. doi: 10.3846/16487788.2012.680756.

[47] L.-P. Kerkhove and M. Vanhoucke. Optimised scheduling for weather sensitive offshore construction projects. *Omega*, 66:58 – 78, 2017. ISSN 0305-0483. doi: https://doi.org/10.1016/j.omega.2016.01.011. URL `http://www.sciencedirect.com/science/article/pii/S0305048316000128`.

[48] S. Khalouli, R. Benmansour, and S. Hanafi. An ant colony algorithm based on opportunities for scheduling the preventive railway maintenance. In *2016 International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 594–599, April 2016.

[49] Harry A. Kinnison. *Aviation Maintenance Management ; Second Edition.* McGraw-Hill, 2013.

[50] Mark P. Kleeman and Gary B. Lamont. *Solving the Aircraft Engine Maintenance Scheduling Problem Using a Multi-objective Evolutionary Algorithm*, pages 782–796. Springer Berlin Heidelberg, 2005. ISBN 0302-9743. doi: 10.1007/978-3-540-31880-4_54. URL `https://dx.doi.org/10.1007/978-3-540-31880-4_54`.

[51] Antoon W.J. Kolen and Leo G. Kroon. License class design: complexity and algorithms. *European Journal of Operational Research*, 63(3):432 – 444, 1992. ISSN 0377-2217. doi: https://doi.org/10.1016/0377-2217(92)90160-B. URL `http://www.sciencedirect.com/science/article/pii/037722179290160B`.

[52] George Kozanidis and Athanasios Skipis. Flight and maintenance planning of military aircraft for maximum fleet availability. *Military Operations Research*, 15(1), 2010. doi: 10.5711/morj.15.1.53.

[53] George Kozanidis, Andreas Gavranis, and George Liberopoulos. Heuristics for flight and maintenance planning of mission aircraft. *Annals of Operations Research*, 221(1):211–238, Oct 2014. ISSN 1572-9338. doi: 10.1007/s10479-013-1376-6. URL `https://doi.org/10.1007/s10479-013-1376-6`.

[54] J. Krause, E. L. Sieczka, and H. S. Lopes. Differential evolution variants and milp for the pipeline network schedule optimization problem. In *2015 Latin America Congress on Computational Intelligence (LA-CCI)*, pages 1–6, Oct 2015. doi: 10.1109/LA-CCI.2015.7435949.

[55] Patricio Lamas and Erik Demeulemeester. A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations. *Journal of Scheduling*, 19(4):409–428, Aug 2016. ISSN 1099-1425. doi: 10.1007/s10951-015-0423-3. URL `https://doi.org/10.1007/s10951-015-0423-3`.

[56] Roel Leus and Willy Herroelen. Stability and resource allocation in project planning. *IIE Transactions*, 36(7):667–682, 2004. doi: 10.1080/07408170490447348. URL `https://doi.org/10.1080/07408170490447348`.

[57] Xiao long Zheng and Ling Wang. A multi-agent optimization algorithm for resource constrained project scheduling problem. *Expert Systems with Applications*, 42(15):6039 – 6049, 2015. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2015.04.009. URL `http://www.sciencedirect.com/science/article/pii/S0957417415002390`.

[58] Malek Masmoudi and Alain Haït. Project scheduling under uncertainty using fuzzy modelling and solving techniques. *Engineering Applications of Artificial Intelligence*, 26(1):135 – 149, 2013. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2012.07.012. URL `http://www.sciencedirect.com/science/article/pii/S095219761200200X`.

[59] V. Mattila and K. Virtanen. Scheduling fighter aircraft maintenance with reinforcement learning. In *Proceedings of the 2011 Winter Simulation Conference (WSC)*, pages 2535–2546, Dec 2011. doi: 10.1109/WSC.2011.6147962.

[60] Ville Mattila and Kai Virtanen. A simulation-based optimization model to schedule periodic maintenance of a fleet of aircraft. In *2005 European Simulation and Modelling Conference, Portugal, 24.-26.10.2005*, pages 479–483, 2005.

[61] Hayet Mogaadi and Besma Fayech. Robust optimization for rcpsp under uncertainty. *International Journal of Software Engineering & Applications*, 7:45–55, 03 2016. doi: 10.5121/ijsea.2016.7205.

[62] Walid El Moudani and Félix Mora-Camino. A dynamic approach for aircraft assignment and maintenance scheduling by airlines. *Journal of Air Transport Management*, 6(4):233–237, 2000. ISSN 0969-6997. doi: 10.1016/s0969-6997(00)00011-9. URL https://dx.doi.org/10.1016/S0969-6997(00)00011-9.

[63] Hongqi Pan and Chung-Hsing Yeh. A metaheuristic approach to fuzzy project scheduling. In Vasile Palade, Robert J. Howlett, and Lakhmi Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, pages 1081–1087, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-45224-9.

[64] Nikolaos Papadakos. Integrated airline scheduling. *Computers & Operations Research*, 36(1): 176 – 195, 2009. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2007.08.002. URL http://www.sciencedirect.com/science/article/pii/S0305054807001499. Part Special Issue: Operations Research Approaches for Disaster Recovery Planning.

[65] Christos H. Papadimitriou. Computational complexity. In *Encyclopedia of Computer Science*, pages 260–265. John Wiley and Sons Ltd., Chichester, UK. ISBN 0-470-86412-5. URL http://dl.acm.org.tudelft.idm.oclc.org/citation.cfm?id=1074100.1074233.

[66] Vincent Van Peteghem and Mario Vanhoucke. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 201(2):409 – 418, 2010. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2009.03.034. URL http://www.sciencedirect.com/science/article/pii/S037722170900191X.

[67] Jean-Marie Pla. An "out-of-kilter" algorithm for solving minimum cost potential problems. *Mathematical Programming*, 1(1):275–290, Dec 1971. ISSN 1436-4646. doi: 10.1007/BF01584092. URL https://doi.org/10.1007/BF01584092.

[68] Gang Quan, Garrison W. Greenwood, Donglin Liu, and Sharon Hu. Searching for multiobjective preventive maintenance schedules: Combining preferences with evolutionary algorithms. *European Journal of Operational Research*, 177(3):1969 – 1984, 2007. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2005.12.015. URL http://www.sciencedirect.com/science/article/pii/S0377221705008477.

[69] B. Roland, C. Di Martinelly, F. Riane, and Y. Pochet. Scheduling an operating theatre under human resource constraints. 58(2):212–220, 2010. ISSN 0360-8352. doi: 10.1016/j.cie.2009.01.005. URL https://dx.doi.org/10.1016/j.cie.2009.01.005.

[70] Slowinksi Roman and Maciej Hapke. *Scheduling under fuzziness.* Physica-Verlag, 2000.

[71] Leandro Julian Salazar Rosales. ANALYSING UNCERTAINTY AND DELAYS IN AIRCRAFT HEAVY MAINTENANCE, 2015.

[72] Majid Sabzehparvar and S. Mohammad Seyed-Hosseini. A mathematical model for the multi-mode resource-constrained project scheduling problem with mode dependent time lags. *The Journal of Supercomputing*, 44(3):257–273, Jun 2008. ISSN 1573-0484. doi: 10.1007/s11227-007-0158-9. URL https://doi.org/10.1007/s11227-007-0158-9.

[73] Premaratne Samaranayake. Current practices and problem areas in aircraft maintenance planning and scheduling–interfaced/integrated system perspective. 01 2006.

[74] Abdulkadir Sarac, Rajan Batta, and Christopher M. Rump. A branch-and-price approach for operational aircraft maintenance routing. *European Journal of Operational Research*, 175(3):

1850 – 1869, 2006. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2004.10.033. URL http://www.sciencedirect.com/science/article/pii/S0377221705004807.

[75] João Tomé Saraiva, Marcelo Leandro Pereira, Virgílio Torrado Mendes, and José Carlos Sousa. A simulated annealing based approach to solve the generator maintenance scheduling problem. *Electric Power Systems Research*, 81(7):1283 – 1291, 2011. ISSN 0378-7796. doi: https://doi.org/10.1016/j.epsr.2011.01.013. URL http://www.sciencedirect.com/science/article/pii/S0378779611000253.

[76] Ackert Shannon. Basics of aircraft maintenance programs for financiers. URL http://aircraftmonitor.com/uploads/1/5/9/9/15993320/basics_of_aircraft_maintenance_programs_for_financiers___v1.pdf. Accessed on 12.06.2019.

[77] Barry C. Smith and Ellis L. Johnson. Robust airline fleet assignment: Imposing station purity using station decomposition. *Transportation Science*, 40(4):497–516, 2006. doi: 10.1287/trsc.1060.0153. URL https://doi.org/10.1287/trsc.1060.0153.

[78] SY Sohn and KB Yoon. Dynamic preventive maintenance scheduling of the modules of fighter aircraft based on random effects regression model. *The Journal of the Operational Research Society*, 61(6):974–979, 2010. ISSN 01605682, 14769360. URL http://www.jstor.org/stable/40608269.

[79] Chellappan Sriram and Ali Haghani. An optimization model for aircraft maintenance scheduling and re-assignment. 2003.

[80] Chellappan Sriram and Ali Haghani. An optimization model for aircraft maintenance scheduling and re-assignment. *Transportation Research Part A: Policy and Practice*, 37(1): 29–48, 2003. ISSN 0965-8564. doi: 10.1016/s0965-8564(02)00004-6. URL https://dx.doi.org/10.1016/S0965-8564(02)00004-6.

[81] Statista. Size of aircraft fleets worldwide | statistic. URL https://www.statista.com/statistics/262971/aircraft-fleets-by-region-worldwide/. Accessed on 03.06.2019.

[82] Roman Słowinski. Multiobjective network scheduling with efficient use of renewable and nonrenewable resources. *European Journal of Operational Research*, 7(3):265 – 273, 1981. ISSN 0377-2217. doi: https://doi.org/10.1016/0377-2217(81)90348-9. URL http://www.sciencedirect.com/science/article/pii/0377221781903489.

[83] Anabela P. Tereso, M.Madalena T. Araújo, and Salah E. Elmaghraby. Adaptive resource allocation in multimodal activity networks. *International Journal of Production Economics*, 92 (1):1 – 10, 2004. ISSN 0925-5273. doi: https://doi.org/10.1016/j.ijpe.2003.09.005. URL http://www.sciencedirect.com/science/article/pii/S0925527303002925.

[84] Arit Thammano and Ajchara Phu-ang. A hybrid evolutionary algorithm for the resource-constrained project scheduling problem. *Artificial Life and Robotics*, 17(2):312–316, Dec 2012. ISSN 1614-7456. doi: 10.1007/s10015-012-0065-x. URL https://doi.org/10.1007/s10015-012-0065-x.

[85] Lin-Yu Tseng and Shih-Chieh Chen. A hybrid metaheuristic for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 175(2):707 – 721, 2006. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2005.06.014. URL http://www.sciencedirect.com/science/article/pii/S0377221705005023.

[86] Vincente Valls, Fransisco Ballestin, and Sacramento Quintanill. Justification and rcpsp: A technique that pays. *European Journal of Operational Research*, 165(2):375 – 386, 2005. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2004.04.008. URL `http://www.sciencedirect.com/science/article/pii/S0377221704002462`. Project Management and Scheduling.

[87] Stefan Voß and Andreas Witt. Hybrid flow shop scheduling as a multi-mode multi-project scheduling problem with batching requirements: A real-world application. 105(2):445–458, 2007. ISSN 0925-5273. doi: 10.1016/j.ijpe.2004.05.029. URL `https://dx.doi.org/10.1016/j.ijpe.2004.05.029`.

[88] V.Selvi and Dr.R.Umarani . Comparative analysis of ant colony and particle swarm optimization techniques. *International Journal of Computer Applications*, 5, 08 2010. doi: 10.5120/908-1286.

[89] Hongjing Wei, Shaobo Li, Houmin Jiang, Jie Hu, and Jianjun Hu. Hybrid genetic simulated annealing algorithm for improved flow shop scheduling with makespan criterion. *Applied Sciences*, 8(12), 2018. ISSN 2076-3417. doi: 10.3390/app8122621. URL `https://www.mdpi.com/2076-3417/8/12/2621`.

[90] Oliver Weide, David Ryan, and Matthias Ehrgott. An iterative approach to robust and integrated aircraft routing and crew scheduling. *Computers & Operations Research*, 37(5):833 – 844, 2010. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2009.03.024. URL `http://www.sciencedirect.com/science/article/pii/S0305054809000963`. Disruption Management.

[91] Oliver Wyman. 2018-2028 fleet and mro forecast commentary. URL `https://www.oliverwyman.com/our-expertise/insights/2018/jan/2018-2028-fleet-and-mro-forecast-commentary.html`. Accessed on 03.06.2019.

[92] Shangyao Yan, Chun-Ying Chen, and Chun-Ying Yuan. Long-term aircraft maintenance scheduling for an aircraft maintenance centre: A case study. *International Journal of Applied Management Science*, 1:143–159, 02 2008. doi: 10.1504/IJAMS.2008.021098.

[93] Zhixian Yang and Guobin Yang. Optimization of aircraft maintenance plan based on genetic algorithm. *Physics Procedia*, 33:580 – 586, 2012. ISSN 1875-3892. doi: https://doi.org/10.1016/j.phpro.2012.05.107. URL `http://www.sciencedirect.com/science/article/pii/S1875389212014204`. 2012 International Conference on Medical Physics and Biomedical Engineering (ICMPBE2012).

[94] Lotfi A. Zadeh. Fuzzy sets as a basis for a theory of possibility. 1978.

[95] R. Zamani. An accelerating two-layer anchor search with application to the resource-constrained project scheduling problem. *IEEE Transactions on Evolutionary Computation*, 14(6):975–984, Dec 2010. ISSN 1089-778X. doi: 10.1109/TEVC.2010.2047861.

[96] Xuesong Zhou and Ming Zhong. Single-track train timetabling with guaranteed optimality: Branch-and-bound algorithms with enhanced lower bounds. *Transportation Research Part B: Methodological*, 41(3):320 – 341, 2007. ISSN 0191-2615. doi: https://doi.org/10.1016/j.trb.2006.05.003. URL `http://www.sciencedirect.com/science/article/pii/S0191261506000737`.

III

Appendices

*Still has to be graded

A
Benchmark Model

To validate the Genetic Algorithm, a benchmark model has been developed based on a novel MILP formulation. The formulation is an adaptation of the formulation as presented in the paper in Part I. Adaptations are necessary since the formulation presented in the paper is nonlinear. The presented model is solved using CPLEX in order to assess the optimality of solutions found by the GA.

A.1. Objective Function

The objective of minimizing lost flight hours is slightly adapted by introducing a new semi-continuous variable, $Z_{i,j,t}$. This variable is introduced to linearize the objective function and represents the flown flight hours of aircraft i, if check j is scheduled at time t. The objective function can be found in Equation A.1 where the first part of the equation represents the difference between the maximum allowed flight hours and the flight hours an aircraft has flown when a C-check is scheduled. The second part of the equation represents a penalty if any extra slots are necessary.

$$\min f_{obj}(x, S_n) = \sum_{i\in I}\sum_{j\in J_i}\sum_{t\in T}\left(|I_i^{FH}\cdot x_{i,j,t} - Z_{i,j,t}\right) + \sum_{t\in T} E_t \times C_E \tag{A.1}$$

Equations A.2, A.3, and A.4 are introduced to ensure that the new semi-continuous decision variable $Z_{i,j,t}$ is either zero if check j is not scheduled at time t for aircraft i ($x_{i,j,t} = 0$) or $y_{i,t}$ when it is ($x_{i,j,t} = 1$).

$$Z_{i,j,t} \le x_{i,j,t}\cdot I_i^{FH} \qquad \forall i \in I, j \in J_i, t \in T \tag{A.2}$$

$$Z_{i,j,t} \le y_{i,t-1} \qquad \forall i \in I, j \in J_i, t \in T \tag{A.3}$$

$$Z_{i,j,t} \ge \left(x_{i,j,t} - 1\right)\cdot I_i^{FH} + y_{i,t-1} \qquad \forall i \in I, j \in J_i, t \in T \tag{A.4}$$

A.2. Constraints

Constraints for the problem can be divided into utilization, operational, and check constraints.

Utilization Constraints

The benchmark model is used for smaller test cases and therefore only considers the usage parameter of flown flight hours. Calendar days, flight cycles, A-check, and D-check are not taken into account. Therefore, only Equation A.5 is introduced. It ensures that the flown flight hours for aircraft i should not exceed the maximum interval.

$$y_{i,t} \le I_i^{FH} \qquad \forall i \in I, t \in T \tag{A.5}$$

Equation A.6, A.7, and A.8 update the flown flight hours for all aircraft based on the utilization at time step t. M is a big number, chosen such that the flown flight hours if an aircraft i is being maintained at time t ($m_{i,t} = 1$) are reset to zero.

$$y_{i,t+1} \ge y_{i,t} + U_{i,t,S_n} - M \times m_{i,t}^{C} \qquad \forall i \in I, t \in [1, \ldots, T-1] \tag{A.6}$$

$$y_{i,t+1} \le y_{i,t} + U_{i,t,S_n} \qquad \forall i \in I, t \in [1, \ldots, T-1] \tag{A.7}$$

$$y_{i,t+1} \le \left(1 - m_{i,t+1}^{C}\right) \cdot M \qquad \forall i \in I, t \in [1, \ldots, T-1] \tag{A.8}$$

Operational Constraints

If an aircraft is scheduled for a C-check it should occupy a hangar for the duration of the C-check. This is introduced through Equation A.9 where the binary variable $m_{i,t}^{C}$ is set to 1 for the duration of the C-check, $[t, t + d_{i,j,S_n}(t)]$, if check j for aircraft i is scheduled to start a time t ($x_{i,j,t} = 1$). Equation A.10 ensures that the variable $m_{i,t}^{C}$ is zero for all other instances.

$$\sum_{t \in [t, t+d_{i,j,S_n}(t)]} m_{i,t}^{C} \ge d_{i,j,S_n}(t) \times x_{i,j,t} \qquad \forall i \in I, j \in J_i, t \in T \tag{A.9}$$

$$m_{i,t}^{C} \le \sum_{j \in J_i} \sum_{t \in [t-d_{i,j,S_n}(t), t]} x_{i,j,t} \qquad \forall i \in I, t \in T \tag{A.10}$$

During the entire planning horizon, the number of occupied hangar slots should not exceed the maximum capacity. This is defined by Equation A.11.

$$\sum_{i \in I} m_{i,t}^{C} \le L_t + E_t \qquad \forall t \in T \tag{A.11}$$

Check Constraints

The C-check is divided into a cycle of 12 check types (C1,..., C12). These C-checks are planned subsequently, meaning that after C1 a C2 check has to be scheduled, after C2 a C3, and so on. In order to formulate this in the problem, several constraints have been added. Equation A.12 ensures that if a check of type j+1 is scheduled, it is at a later date than the previous check of type j. Due to Equation A.13, check j has to be scheduled before a check of type j+1 is planned. Ensuring that the checks are scheduled according to the cycle.

$$\sum_{t\in T}\big(x_{i,j+1,t}\times(t-M)\big)\geq\sum_{t\in T}\big(x_{i,j,t}\times t\big)-M \qquad \forall i\in I, j\in[1,\ldots,J_i-1] \tag{A.12}$$

$$\sum_{t\in T}x_{i,j,t}\geq\sum_{t\in T}x_{i,j+1,t} \qquad \forall i\in I, j\in[0,\ldots,J_i-1] \tag{A.13}$$

Furthermore, it is necessary that a certain check can only be scheduled once. Resulting in Equation A.14.

$$\sum_{t\in T}x_{i,j,t}\leq 1 \qquad \forall i\in i, j\in J_i \tag{A.14}$$

B
Sensitivity Analysis on GA Parameters

The Genetic Algorithm is a stochastic optimization approach. As such, the parameters of the genetic algorithm can significantly influence the solution quality of the method. The main parameters in this paper are the population size, crossover rate, and mutation rate. To analyze the sensitivity of the GA to these parameters, several runs of the GA have been performed with different parameter settings. This is done for the case study presented in the paper in Part I for a fleet of 40+ aircraft. The following operational constraints are taken into account:

- A maximum of 3 C-checks can be executed in parallel;
- During weekends and public holidays no work on a C-check is performed;
- During commercial peak periods (i.e. summer and holiday periods), no C-checks can be scheduled ($L_t = 0$);
- Due to resource availability reasons there has to be a minimum of 3 days between the start dates of two C-checks (d_c)

B.1. Mutation and Crossover Rate

Firstly, the GA has been run with different mutation rates, being either 0.2, 0.4, 0.6, 0.8, or 1.0. Next the crossover rates were chosen to be either 0.0, 0.3, 0.6, 0.9, or 1.0. The runs have been performed with a fixed population size of 10.

The sensitivity of the objective value and computation time to a change in mutation rate can be found in Table B.1, where sensitivity is expressed in the estimate of the partial derivative compared to the 0.6 mutation rate setting. The objective value and computation time are displayed with their mean value and 95% confidence interval. They are a result of running the GA with varying mutation rates while keeping the crossover rate constant at 0.9. As can be seen from the table, a change in mutation rate significantly influences the computation time of the algorithm. The increase in computation time with an increase in mutation rate is almost a linear relationship. This is expected, since an increase in mutation rate will increase the number of times the ϵ-greedy algorithm is run. From the table it can also be seen that a high mutation rate setting reduces the average objective value. This is because of the fact that mutation is not random but takes operational constraints and knowledge about the problem into account. However, a setting of 1.0 negates some of the benefits of crossover and is therefore not necessarily the best setting for finding the most optimum solution to the problem.

Table B.1: Sensitivity of objective value and computation time to change in mutation rate, where sensitivity is an estimation of the partial derivative

Mutation Rate	Objective[FH]	Sens. Objective	Computation Time [s]	Sens. Time
0.2	45401.3 ± 435.4	1107.6	627.6 ± 170.2	-1080.1
0.4	44767.5 ± 953.7	313.8	1003.3 ± 224.3	-1033.2
0.6	44662.9 ± 587.3	N/A	1347.7 ± 374.7	N/A
0.8	44418.1 ± 559.1	-734.4	1675.7 ± 195.1	984.0
1.0	44621.1 ± 317.3	-62.7	1961.1 ± 707.7	920.1

The sensitivity of the objective value and computation time to a change in crossover rate can be seen in Table B.2. The table shows the results of varying the crossover rate while keeping the mutation rate constant at 0.4. From the table it is clear that changing the crossover rate has relatively little impact on the computation time of the algorithm. However, increasing the crossover rate, to a certain extent, seems to be beneficial regarding the objective value.

Table B.2: Sensitivity of objective value and computation time to change in crossover rate, where sensitivity is an estimation of the partial derivative

Crossover Rate	Objective [FH]	Sens. Objective	Computation Time [s]	Sens. Time
0.0	44912.9 ± 721.9	222.8	1090.9 ± 143.1	27.7
0.3	44808.4 ± 981.9	236.6	1026.0 ± 182.9	-74.4
0.6	44690.1 ± 1146.4	N/A	1063.2 ± 298.8	N/A
0.9	44653.7 ± 1039.0	-72.8	1062.7 ± 439.8	-1.0
1.0	44805.8 ± 737.0	115.7	1105.0 ± 201.6	41.8

The influence of varying the mutation rate and crossover rate on the objective value can be visualized in the form of box plots through Figure B.1. It can indeed be seen that a high mutation rate is beneficial for finding good solutions to the problem. It can also be seen that having no crossover and only focusing on local search with respect to all initial schedules finds good solutions. However, the crossover procedure helps find better solutions to the problem. The interaction between the effects can be seen through a contour plot in Figure B.2. When the mutation rate is low, a higher crossover rate is beneficial when considering the objective value. Exploration of new solutions then mostly occurs through the crossover procedure. A high mutation rate, finds solutions to the problem mostly through locally improving existing schedules, where sometimes the crossover procedure is beneficial by introducing new combinations.

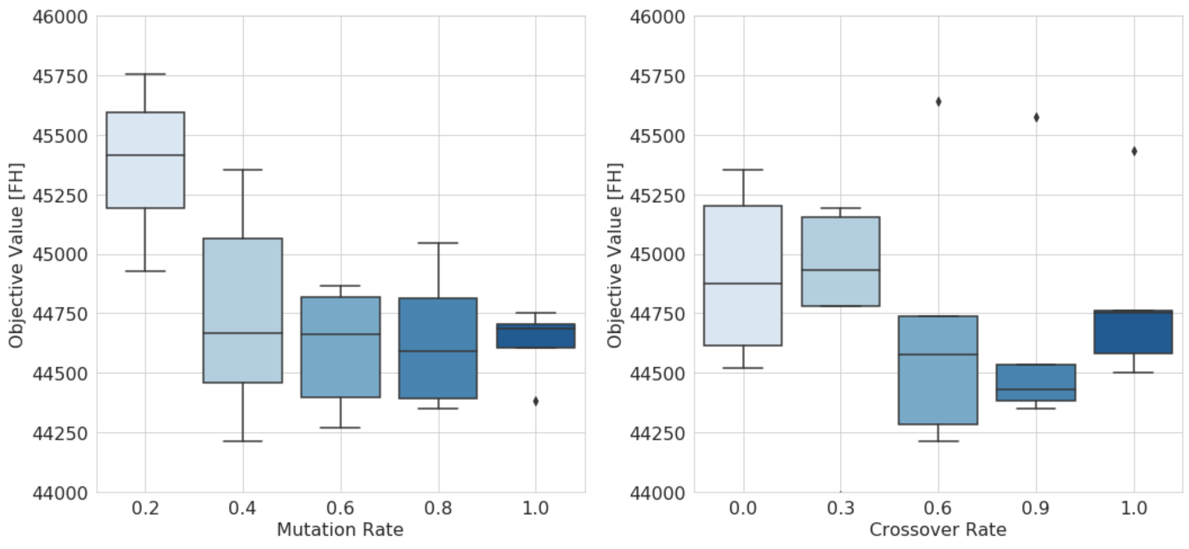


Figure B.1: Box plots of objective value under varying mutation rate and crossover rate

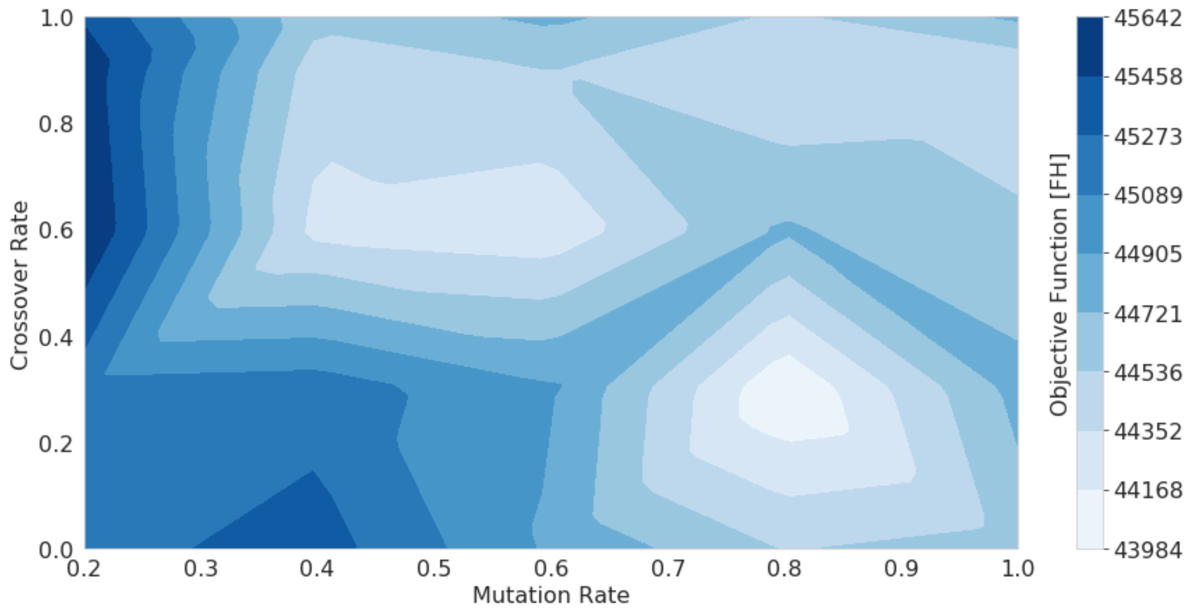


Figure B.2: Contour plot of objective value under varying mutation rate and crossover rate

The effects of varying the mutation rate and crossover rate on the computation time of the GA are visualized in Figure B.3. As is readily clear, increasing the mutation rate indeed increases the computation time almost linearly. Where the effects of crossover rate on the computation time of the genetic algorithm are almost non existent. Figure B.4 shows a contour plot of the computation time under varying mutation and crossover rate. From this figure it is also readily clear, that increasing the mutation rate has the most effect on computation time.

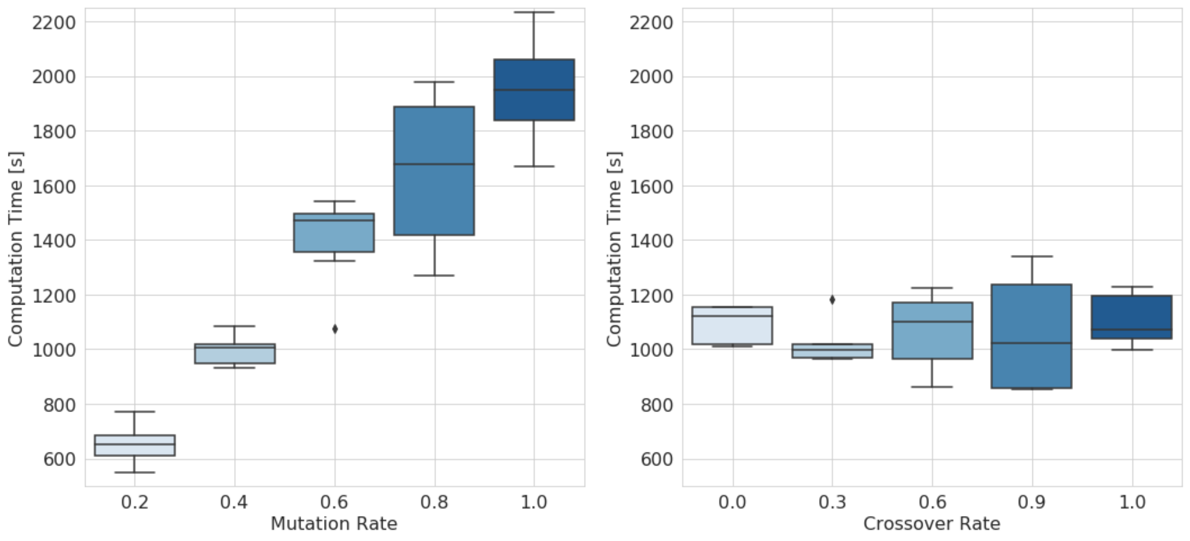


Figure B.3: Box plots of computation time under varying mutation rate and crossover rate

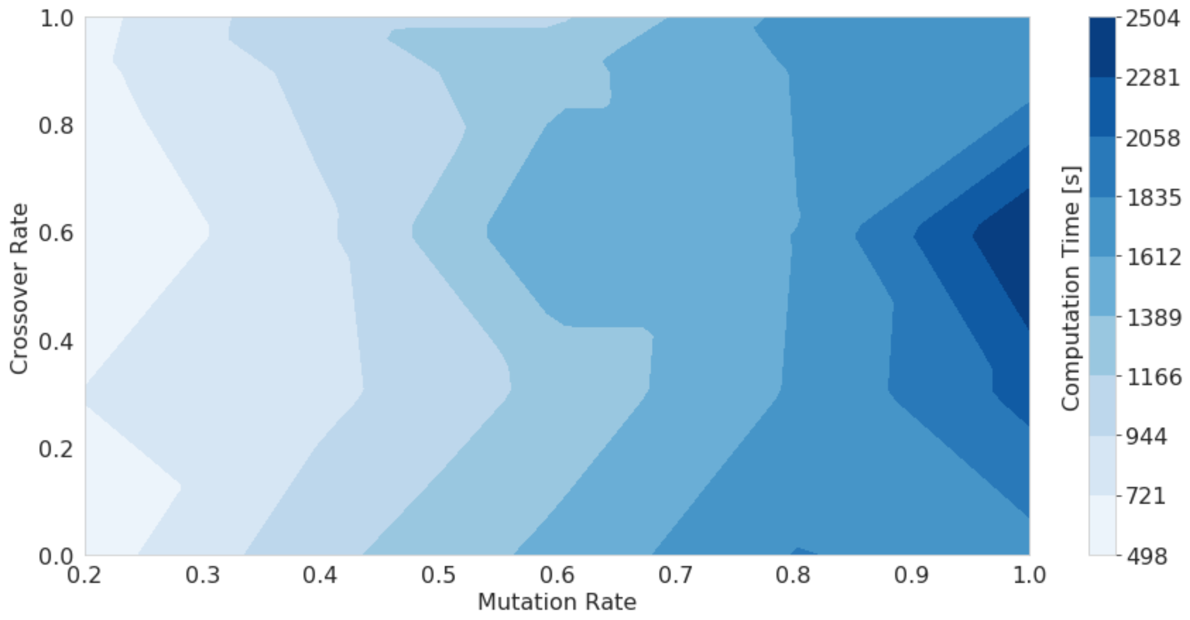


Figure B.4: Contour plot of computation time under varying mutation rate and crossover rate

B.2. Population Size

The population size indicates how many schedules are contained within the population. An increase of population size could allow for more exploration while running the algorithm, but with the added cost of an increase in computation time. This is evaluated by running the GA with a mutation rate of 0.2 and crossover rate of 0.9. The population size is chosen to be 10, 20, or 40. The results of this analysis can be seen in Table B.3. Where it can be seen that an increase in population size indeed results in a small improvement of the objective value. This, however, comes with a significant increase in computation time.

Table B.3: Sensitivity of objective value and computation time to a change in population size, where sensitivity is an estimation of the partial derivative

Population Size	Objective [FH]	Sens. Objective	Computation Time [s]	Sens. Time
10	45380.6 ± 628.5	1000.7	652.7 ± 153.3	-1278.9
20	44880.3 ± 166.5	N/A	1292.2 ± 92.6	N/A
40	44108.1 ± 873.6	-772.2	3246.5 ± 182.5	1954.3

B.3. Stopping Criterion

The iterative process of the genetic algorithm stops based on a predefined set of criteria. The algorithm is terminated if the maximum number of iterations has been reached or when the best solution has not changed for a predefined number of iterations ($n\_generations$). In practice, $n\_generations$ is the limiting factor. To evaluate the effect of this criterion, several runs have been performed with a mutation rate of 0.2 and crossover rate of 0.9. The stopping criterion of $n\_generations$ is either 5, 15, or 25 iterations. The result of the analysis can be seen in Table. B.4. It can be seen that for a small number of iterations the objective value lies within a wide 95% confidence interval. Increasing the maximum number of iterations, with no change in the best found solution, results in better solutions within a narrower confidence interval. This effect is most apparent in the change from 5 to 15 iterations. The difference between $n\_generations$ of 15 and 25 is much less. This is also directly related to the computation time of the algorithm. The increase from 5 to 15 is bigger compared to the step from 15 to 25, since the algorithm finds better new solutions that reset the counter of iterations the best solution has not changed. With the step from 15 to 25 the computation time is mostly larger due to the extra iterations and less so due to finding better solutions.

Table B.4: Sensitivity of objective value and computation time to changing stopping criteria, where sensitivity is an estimation of the partial derivative

n_generations	Objective [FH]	Sens. Objective	Computation Time [s]	Sens. Time
5	47817.0 ± 2680.7	3654.7	275.2 ± 38.8	-566.4
15	45380.6 ± 628.5	N/A	652.7 ± 153.3	N/A
25	45148.0 ± 350.0	-348.9	777.5 ± 74.6	187.1

C
GA Simulator

In order to evaluate the fitness of each schedule in the population, the GA uses a simulator. The general layout of the simulator can be found in Algorithm 1. It returns the fitness of the worst case scenario for all schedules in the population.

Algorithm 1: GA SIMULATOR

```
Input: Population, Scenarios, Available Slots, Memory
begin
    Population_Fitness ← [ ]
    for Chromosome ∈ Population do
        Fitness ← []
        for Scenario ∈ Scenarios do
            $U_{S_n}, D_{S_n} \leftarrow Scenario$    //Utilization and Duration matrix based on Scenario
            Hangar ← []
            for i ∈ I do
                text ← 'Aircraft i ' + Chromosome[i].astype(str)    // Schedule of aircraft i
                if text ∈ Memory then
                    ac ← Memory[Scenario][text]    //Get simulation from memory
                else
                    env ← initialize environment
                    ac ← class Aircraft($U_{S_n}, D_{S_n}$, Chromosome, env)    // Initialize class
                    env.run(until=T+1)    // Run simulation for entire planning horizon
                    Memory[Scenario][text].insert(ac)    //add simulation to memory
                Fitness[Scenario].insert(ac.Fitness)
                Hangar.insert(ac.Hangar)
            end
            if Available Slots - Hangar < 0 then
                Fitness[Scenario].insert(Extra slots penalty)    // Penalty for extra slots
        end
        Population_Fitness.insert(max(Fitness))    //Add fitness of worst case Scenario
    end
    return Population_Fitness, Memory
end
```

The fitness, in the form of lost flight hours, is determined for each aircraft per scenario per schedule. If the exact same simulation has already been performed the results are drawn from memory. After simulation for all aircraft and all scenarios, the fitness of the analyzed schedule can be determined as the maximum fitness over all scenarios. This is done for all schedules in the population. The core of the simulation makes use of the simpy package in python [1]. The simulation with simpy uses the Aircraft class as defined in the following code:

```
class Aircraft(object):
    def __init__(self, env, name, max_FH, max_FH_A, Initial_FH, \
                 Initial_FH_A, current_check, i, Chromosome, T,\
                 Utilization, Duration, Out_of_service, Initial_DY, max_DY):

        self.env = env #simpy environment #
        self.action = env.process(self.run(env, i, T)) #Run for time period #
        self.name = name # A/C Tail #
        self.Fitness = 0 # Initialize Fitness of aircraft #
        self.constr = 0 # Initialize Number of Constraint violations #
        self.max_FH = max_FH[i] # Maximum FH interval C-check #
        self.max_FH_A = max_FH_A[i] # Maximum FH interval A-check #
        self.FH = Initial_FH[i] # Intialize FH C-check #
        self.FH_A = Initial_FH_A[i] # Initialize FH A-check #
        self.D = Initial_D[i] # Intialize DY D-check #
        self.cycle = D_cycle[i] # Initialize D-check cycle #
        self.DMAX = D_max[i] # Maximum cycles before D-check #
        self.max_D = max_D[i] # Maximum DY interval D-check #
        self.DY = Initial_DY[i] # Intialize DY C-check #
        self.max_DY = max_DY[i] # Maximum DY C-check #
        self.current_check = current_check[i] #Current C-check for aircraft #
        self.planned_checks = \
        Chromosome[i][0:(len(np.where(np.diff(Chromosome[i])>0)[0])+1)]\
                                #C-check schedule for aircraft #
        self.check = 0 # Initialize amount of C-checks planned #
        self.Hangar = [] # Initialize hangar occupation #
        self.Utilization = Utilization # Daily Utilization matrix #
        self.Duration = Duration[i] # Duration matrix #
        self.Out_of_service = Out_of_service # After which C-check \ #
                                        #aircraft i goes out of service #
        self.FH_Tol = FH_Tol[i] # Previously used tolerance on FH C-check #
        self.DY_Tol = DY_Tol[i] # Previously used tolerance on DY C-check #

    def run(self, env, i, T): #Run for entire planning horizon #
        while True:
            yield self.env.process(self.update_utilization(env, i, T))

    def update_utilization(self, env , i , T): #Update aircraft parameters #

        if self.check == self.Out_of_service[i]: \
        # If aircraft out of service end simulation #
            yield self.env.timeout(T+1 - env.now)

        self.FH_A += (self.Utilization[i][months[env.now]]) \
                                                # Update FH A-check #
```

[1] simpy module for python: https://simpy.readthedocs.io/en/latest/contents.html

```
        self.D += 1 # Update DY D-check #

        if self.FH_A > self.max_FH_A: # If FH A-check \#
                                        #exceeds maximum interval #
            self.FH_A = 0 # reset FH A-check #
            a_check = True # Schedule an A-check #
        else:
            a_check = False # No A-check scheduled #

        if env.now not in self.planned_checks: # If time- \#
                                                #step not in schedule #
            self.DY += 1 # Update DY C-check #
            if a_check != True:
                self.FH += (self.Utilization[i][months[env.now]])\
                # If no A-check update FH C-check #

            if self.FH > self.max_FH - self.FH_Tol and\
            self.check > (len(self.planned_checks)- 1): \
            # If no check scheduled anymore but FH exceeds limit #
                self.constr += T-env.now # Update Number of violations #
                self.Fitness += M_Cost*(T-env.now) # Update Fitness #

                yield self.env.timeout(T+1 - env.now) # End simulation #
            else:
                yield self.env.timeout(1) # Continue to next time step #

        else: # If C-check scheduled at time step #

            fitness = self.max_FH - self.FH - self.FH_Tol # Update Fitness #

            if fitness >= 0:
                self.Fitness += fitness # If scheduled on time no penalty #
            else:

                self.constr += np.ceil(abs(fitness)/(self.Utilization[i]\
                                       [months[env.now]])) \
                                       #Update No of violations #
                self.Fitness += M_Cost*\
                np.ceil(abs(fitness)/(self.Utilization[i][months[env.now]]))\
                                        #Update Fitness#

            if self.max_DY - self.DY - self.DY_Tol < 0: \
            # If DY C-check exceeds limit update fitness and violations #
                self.Fitness += M_Cost*(self.DY-(self.max_DY-self.DY_Tol))
                self.constr +=  self.DY - (self.max_DY-self.DY_Tol)

            self.cycle += 1 # Update cycle #

            if self.cycle == self.DMAX: # If cycles at maximum #
                dcheck = True # D-check is scheduled #
            elif self.D >= self.max_D: # If DY D-check equals maximum #
                dcheck = True # D-check is scheduled #
            else:
                dcheck = False # No D-check is scheduled #

            if dcheck:
                if self.max_D - self.D < 0: # If D-check is too late #
                    self.Fitness += 999999 # Penalty for D-check too late #
                self.cycle = 0 # reset cycle #
                self.D = 0 # Reset DY D-check parameter #

            self.FH = 0 # Reset FH C-check #
            self.FH_A = 0  # Reset FH A-check #
            self.DY = 0 # Reset DY C-check #
            self.FH_Tol = 0 # No tolerance used for FH C-check #
            self.DY_Tol = 0 # No tolerance used for DY C-check #

            check_duration = self.Duration[self.check] # Duration of C-check #
            if dcheck:
                check_duration += d_extra # If D-check increase duration #

            diff = 1
            wknds = 0
            while diff != 0: # Add weekends/public holidays to duration #
                diff = len(np.where(weekends\[env.now:env.now+check_duration]\
                                    == True)[0]) - wknds
                wknds = len(np.where(weekends[env.now:env.now+check_duration]\
                                     == True)[0])
                check_duration = check_duration + diff

            self.current_check += 1 # Update current C-check #
            self.check += 1 # Update amount of checks scheduled #
            self.Hangar.extend(np.arange(env.now, env.now + check_duration)) \
                # Occupy hangar for duration of check #
            if dcheck == False: # If no D-check is scheduled #
                self.D += check_duration # Update DY D-check parameter #

            yield self.env.timeout(check_duration) \
            # Continue simulation after check has been performed #
```