Optimisation of Preventive A-check Maintenance Tasks: Integrated and Distributed Approaches

João Borrego, Marta Ribeiro, Alireza Amiri-Simkooei
Aerospace Faculty, Control & Operations
Delft University of Technology
Delft, Netherlands

***Abstract*—A-check maintenance scheduling is a complex and critical undertaking for airlines requiring an efficient allocation of resources. Current state-of-the-art focuses primarily on long-term A-check planning, typically targeting a longer scheduling horizon while foregoing individual task planning. This paper introduces a novel integrated approach for A-check scheduling at a seasonal level for an airline fleet, which accounts for both repetitive and one-off maintenance tasks. The A-check maintenance scheduling problem is formulated as a mixed-integer linear program (MILP), which optimises for minimal interval waste and timely initiation of one-off activities. Furthermore, we explore the scalability and flexibility of this problem by proposing three distinct distributed architectures. Subsets of maintenance tasks are scheduled by individual components, guided by a genetic algorithm (GA) acting as a global optimiser, with each architecture managing shared resources differently. We demonstrate our method with a case study from a major European airline using recent data of a fleet of wide-bodied passenger aircraft. While our MILP baseline produces comparable results to real-world schedules within minutes, the distributed architectures, despite their potential for scalability, generally underperform compared to the central planner. We analyse the degradation of solution quality across these distributed architectures, providing insights into their design limitations and the inherent indivisibility of the problem. We propose that our central MILP-based scheduler can be used by airlines as a decision-support tool for A-check task planning at the seasonal level.**

***Keywords*—Airline Maintenance, Distributed Framework, Mixed-Integer Linear Programming, Genetic Algorithm, A-check**

I. Introduction

Maintenance, Repair and Overhaul (MRO) costs account for up to 11% of global airline operational expenses, according to a recent study by the International Air Transport Association (IATA) [8]. For this reason, airline maintenance optimisation remains an attractive research field, as even seemingly marginal improvements can result in substantial operational savings.

Additionally, aircraft maintenance must comply with stringent safety and regulatory requirements, often mandating that certain tasks be performed at specified intervals. These intervals are typically defined in terms of flight hours, cycles, or calendar time. Missing corresponding deadlines can lead to costly groundings or regulatory penalties. In practice, airlines group maintenance tasks into so-called letter checks, with A-checks being the lightest in terms of workload and D-checks the heaviest. B-checks and D-checks are seldom performed in modern practice and are instead replaced by more frequent A-checks and heavier C-checks, respectively [1], [15].

The maintenance planning process is inherently complex and fragmented, with different teams often optimising local objectives for specific aircraft or task subsets. This siloed approach frequently leads to suboptimal fleet-level scheduling. Maintenance scheduling is not an isolated problem; it's a critical, interconnected component within the broader airline operational landscape, directly impacting network planning, flight scheduling, and crew rostering. The fundamental challenge for airlines is two-fold: (1) accurately modelling these individual operational problems as optimisation tasks, and (2) effectively defining and managing the interfaces and interactions between these distinct planning stages.

Prior work has simplified the complexity of the maintenance check scheduling problem by considering A- and C-checks as homogeneous within a particular type and with fixed repetition intervals [1], [3], [15]. However, in practice, letter checks differ greatly in task content, and their interval is dictated by which specific tasks are grouped in each maintenance opportunity. This work covers this gap by considering individual tasks and their interval, which are particularly relevant from a seasonal planning perspective.

Additionally, previous work has employed fully integrated approaches, which result in prohibitively large solution spaces, rendering central planners computationally expensive and inflexible for real-world scenarios.

To overcome this challenge, we break the seasonal A-check task planning down into smaller sub-problems, emulating current practice at major airlines. Our goal is thus to create smaller systems that can interoperate towards a global optimisation process.

The contribution of this paper is two-fold:

1) Introduction of a novel method for scheduling A-check tasks at a seasonal level for an airline fleet, balancing compliance, efficiency, and resource constraints;
2) Decomposition into several components, each handling a subset of tasks, and integrate them into a comprehensive system for maintenance planning.

The remainder of the paper is structured as follows. Section II defines the problem and establishes the scope of the study. Section III describes prior research and highlights the

research gap. Section IV provides the methodology for this paper. Section V describes the hypotheses proposed in this study. Section VI describes a case study with real data from an airline, and Section VII presents relevant results. A discussion of findings and implications is provided in Section VIII, and finally, the conclusions are drawn in Section IX.

II. Problem Definition

This work focuses on A-checks, which typically require the aircraft to be grounded (removing it from service) for 24 hours. Figure 1 summarises the types of tasks performed in A-checks. Broadly, these tasks fall into two groups: Corrective and Preventive.

Corrective work addresses faults discovered during inspections. Because they cannot be predicted in advance, allocating resources efficiently for corrective work is inherently difficult. Recent studies have investigated methods for forecasting such requirements [11]. Due to this uncertainty, this study focuses specifically on the scheduling of non-corrective (preventive and planned) maintenance tasks.

Preventive tasks are regular tasks with known due dates. For planning and logistical reasons, airlines typically employ a *blocking* strategy, grouping preventive maintenance tasks with close due dates into a repeating sequence of blocks. For A-checks, this translates into a sequence of blocks A1, A2, ... AN, and then returning to A1. This greatly simplifies the logistics of letter checks across a specific aircraft fleet. Repetitive tasks with intervals that do not align with the block sequence are considered *out-of-phase* and must be individually planned.

Finally, airlines seek to maximise the value of this downtime by planning additional maintenance tasks within the same slot, subject to resource constraints. These additional tasks include modifications and other one-time activities, which, while not directly impacting airworthiness, are essential for fleet reliability and performance.

Based on this categorisation, the current study focuses on creating an A-check maintenance schedule by assigning non-corrective tasks (i.e. block, out-of-phase, and one-time) to specific maintenance opportunities (slots). This also entails deciding which aircraft is assigned to each slot. The schedule will cover a whole operational season and include all aircraft for a given sub-fleet.

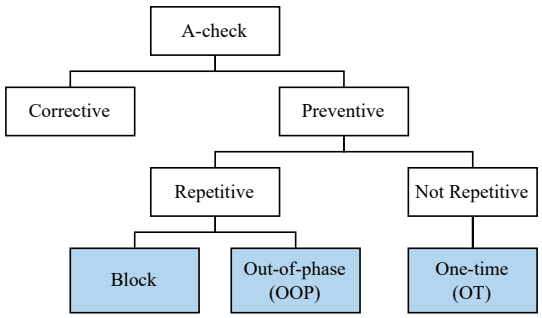

Figure 1. Structure of types of tasks carried out in A-checks.

III. Related Work

Prior research has tackled optimising multiple stages of the airline maintenance planning process. These include long-term planning of C-checks [15], [16], integrated A- and C-check planning [1], [3], as well as line maintenance carried out during layovers between scheduled flights [12], [13]. Such problems are often modelled as a Resource-Constrained Project-Scheduling Problem (RCPSP) [4], which are known to be NP-hard [10]. As a result, and as shown by a recent survey paper [17], maintenance is generally considered to constrain the aircraft routing problem rather than as a primary optimisation objective.

Existing work employs integrated approaches modelled mostly as mixed-integer linear programs (MILPs). For example, van Kessel et al. [14] model airline maintenance task rescheduling in a disruptive environment as a MILP. They validate their framework on a case study with data from a major airline, concluding that the approach can increase schedule stability and reduce ground time. While their model addresses both corrective and preventive tasks, it focuses solely on hangar maintenance, excluding letter checks (e.g., A-checks).

In the line maintenance context, Shaukat et al. [13] propose a MILP model alongside novel heuristic methods that enable faster convergence while retaining schedule quality. Sciau et al. [12] frame the optimisation of a line maintenance schedule as a RCPSP and employ constraint programming to solve it.

Few works have proposed heuristic methods to tackle the computational burden of these maintenance optimisation models. Genetic algorithms (GAs), for example, have been used for C-check scheduling under uncertainty. Van der Weide et al. [15] formulate long-term heavy aircraft maintenance scheduling as a min-max integer linear programming (ILP) problem, which is solved using an efficient GA. The heuristic GA demonstrates much better scalability compared to the exact baseline.

Some studies attempt to decompose the large-scale optimisation problem into multiple iterative processes. Deng et al. [3] propose a dynamic programming method to optimise the long-term maintenance schedule, namely A- and C-checks. Their model minimises the wasted interval between checks, thereby reducing the total number of checks required. A similar study tackles this problem by training a reinforcement learning agent employing deep Q-learning [1].

To tackle the joint challenges of aircraft routing and crew pairing, prior studies have explicitly iteratively tackled two sub-problems. Cordeau et al. [2] propose a process based on Benders decomposition, which iteratively solves a master problem for aircraft routing and a sub-problem for crew pairing. A recent study tackles a similar problem employing multi-agent reinforcement learning [5].

A. Research Gap

Despite extensive research in airline maintenance scheduling, a significant gap remains in addressing the integrated

optimisation of detailed A-check scheduling that simultaneously considers both repetitive and non-repetitive tasks with distinct scheduling objectives. Most prior work either simplifies A-checks as homogeneous entities [1], [3] or focuses on higher-level C-checks [15] or line maintenance [12], [13]. A-check task scheduling occupies a middle ground between the opportunistic, short-term nature of line maintenance and the long-term, resource-intensive planning required for C-checks.

Furthermore, while distributed optimisation has been a promising avenue for managing complexity, there is no existing framework that enables a direct, comparative analysis between a central, integrated optimiser and various distributed optimisation processes for this specific problem. In summary, our work uniquely contributes by:

1) Developing an individual A-check task scheduling model that differentiates between repetitive and non-repetitive tasks, considering specific task labour requirements and slot capacity;
2) Proposing a distributed framework, employing generic optimisation strategies (specifically GAs) with minimal assumptions about local planners, for reuse in other optimisation problems;
3) Providing a direct comparison between integrated and distributed optimisation architectures.

IV. Methodology

This section details the methodology employed to address the A-check maintenance scheduling problem. Section IV-A describes the MILP formulation for our integrated scheduler, while Section IV-B introduces the aforementioned distributed architectures.

Our primary objective is to develop a model that, given a task backlog with due dates and resource requirements and a set of maintenance opportunities with respective labour capacity, determines an optimal task-slot allocation. We first present an integrated, centralised optimisation model that simultaneously considers all tasks and resources. This model provides a globally optimal solution against which the distributed approaches can be evaluated. The distributed implementations are a response to the computational challenges of large-scale MILPs and the fragmented nature of real-world airline planning. Airlines typically design schedules manually in a staged, 'trickle-down' fashion, allocating blocks first, then out-of-phase tasks, and finally one-time tasks. We aim to replicate this staged planning process by considering each task type in a dedicated module.

A. Integrated Baseline

The integrated baseline formulates the seasonal A-check task scheduling problem as a MILP. Sets, parameters, and decision variables for the A-check seasonal scheduling problem are defined in tables I to III, respectively.

We identify three main desired objectives for our A-check task scheduler, in order of priority: (1) schedule as many tasks as possible, (2) schedule repetitive tasks as close to their due date as possible, and (3) schedule one-time tasks as soon as possible. These objectives can be, respectively, modelled as the minimisation of:

- The amount of task deferrals, weighed by the respective cost depending on task type (eq. (1));
- The total wasted interval for repetitive tasks. The WI matrix is pre-computed for all considered task-slot combinations (eq. (2));
- The total number of days until each one-off task is performed (eq. (3)).

TABLE I. Indices and Sets

Set	Description
$a \in A$	set of aircraft
$t \in T$	set of tasks to be scheduled
$t \in T_B \subseteq T$	subset of block tasks
$t \in T_{OOP} \subseteq T$	subset of out-of-phase tasks
$t \in T_{OT} \subseteq T$	subset of one-time tasks
$t \in T_{REP} \subseteq T$	subset of repetitive tasks ($T_B \cup T_{OOP}$)
$s \in S$	set of ground slots
$s \in S_D$	superset of ground slots with an additional virtual slot d for task deferral ($S_D = S \cup \{d\}$)
$st \in ST$	set of skill types considered for task requirements and slot capacities

TABLE II. Parameters and Weights

Parameter	Description
$R_{(st,t)}$	requirement of task t for skill type st, in hours
$C_{(st,s)}$	capacity of slot s for skill type st, in hours
$TA_{(t)}$	aircraft a to which task t is tied to
$TD_{(t)}$	due date of task t, in days from a reference date
$SS_{(s)}$	start date of slot s, in days from a reference date
$WI_{(t,s)}$	wasted interval when repetitive task $t \in T_{REP}$ is assigned to slot s, in number of days. $WI_{(t,s)} = TD_{(t)} - SS_{(s)}$
$C^D_{(t)}$	cost of task deferral for task $t \in T$
$C^{WI}_{(t)}$	cost of each day of wasted interval for repetitive task $t \in T_{REP}$
$C^{ASAP}_{(t)}$	cost of each additional day without being done for one-off task $t \in T_{OT}$
W^D	weight of the task deferral objective
W^{WI}	weight of the wasted interval objective
W^{ASAP}	weight of the one-time task delay objective

$$O_D(ts) = \sum_{t \in T} C^D_{(t)} \cdot ts_{(t,s)}, \quad s = d \tag{1}$$

$$O_{WI}(ts) = \sum_{t \in T_{REP}} \sum_{s \in S} C^{WI}_{(t)} \cdot WI_{(t,s)} \cdot ts_{(t,s)} \tag{2}$$

$$O_{ASAP}(ts) = \sum_{t \in T_{OT}} \sum_{s \in S} C^{ASAP}_{(t)} \cdot SS_{(s)} \cdot ts_{(t,s)} \tag{3}$$

The objectives are inherently conflicting, as, for example, the model could easily achieve a zero wasted interval by deferring all the tasks. To balance trade-offs, we combine the sub-objectives using a simple weighted linear combination, as shown in eq. (4).

$$\begin{aligned} \min O = {} & W^D \cdot O_D(ts) + W^{WI} \cdot O_{WI}(ts) \\ & + W^{ASAP} \cdot O_{ASAP}(ts) \end{aligned} \tag{4}$$

Alternatively, a hierarchic or lexicographic ordering approach can be applied to prevent the model from improving

TABLE III. DECISION VARIABLES

Decision Variable	Description
$as_{(a,s)} \in \{0,1\}$	whether aircraft a is assigned to slot s
$ts_{(t,s)} \in \{0,1\}$	whether task t is assigned to slot s

the average wasted interval at the expense of increasing the number of task deferrals. In this case, each sub-objective is assigned a priority rather than a weight. Ultimately, by establishing an order of the weights such that $W^D > W^{WI} > W^{ASAP}$, it is possible to obtain similar solutions to the hierarchical multi-objective optimisation with the blended approach.

Additionally, the following constraints are defined:

- At most one aircraft can be assigned to each slot (eq. (5)).
- Tasks can only be assigned to slots if the corresponding aircraft is also assigned to that slot (eq. (6)).
- Tasks must be scheduled no later than their deadline/due date (eq. (7)). In practice, this constraint is implemented by excluding trivial decision variables.
- Each task must be either assigned to a slot or deferred (eq. (8)).
- The sum of assigned task labour requirements cannot exceed the respective slot labour capacity (eq. (9)).

$$\sum_{a \in A} as_{(a,s)} \leq 1, \quad \forall s \in S \tag{5}$$

$$ts_{(t,s)} \leq as_{(TA_t,s)}, \quad \forall t \in T, \forall s \in S \tag{6}$$

$$ts_{(t,s)} \cdot TD_{(t)} \leq SS_{(s)}, \quad \forall t \in T, \forall s \in S \tag{7}$$

$$\sum_{s \in S_D} ts_{(t,s)} = 1, \quad \forall t \in T \tag{8}$$

$$\sum_{t \in T} ts_{(t,s)} \cdot R_{(st,t)} \leq C_{(st,s)}, \quad \forall s \in S, \forall st \in ST \tag{9}$$

Finally, the following assumptions are made. First, it is assumed that the number of A-check blocks equals the number of available maintenance slots. However, it is possible to extend the model to include more slots than A-checks, e.g. by adding a constraint stating aircraft can only be assigned to a slot if an A-check block is also assigned. Second, since the main use of the model is to plan a season of A-checks, material availability is assumed not to be a limiting factor. This assumption is reasonable given how much in advance the planning takes place, and is also consistent with the current practice among planners.

B. Distributed Modes

As previously mentioned, centralised integrated planning may lead to a very large number of decision variables, which can result in long solver runtime and large memory requirements. To address this, we present a distributed approach that can break down the problem into smaller components. In this framework, local optimisers allocate specific tasks, while driven by a global optimisation process that steers the local search by imposing shared resource constraints. At a seasonal level of A-check maintenance planning, we identify two main shared resources: (1) the aircraft itself, and (2) the labour capacity of each maintenance slot.

This distributed framework is designed to make minimal assumptions about the techniques used by local schedulers, enabling reuse across different planning tasks. Each local planner is therefore treated as a black box. We start by dividing the tasks by type, each of which is handled by a dedicated local scheduler: (1) Block task scheduler, (2) Out-of-phase (OOP) task scheduler, and (3) One-time task (OT) scheduler.

In practice, the local schedulers share the same model as the baseline and differ only in the specific set of tasks they schedule. These local schedulers are still bound to the constraints on shared resources. We propose three distinct architectures where these are either allocated in a cascading fashion or pre-allocated before any scheduling takes place. These architectures are described as follows:

a) Aircraft-Slot: Figure 2 presents the aircraft-slot optimiser (Mode 1). This architecture focuses on iteratively assigning aircraft to slots. The aircraft-slot representation consists of a categorical value representing an aircraft in the fleet for each of the slots in the schedule. The slot capacity is consumed in a cascading fashion, as the local schedulers run sequentially, first Block, then OOP, and finally OT. At each iteration, the global optimiser proposes an aircraft-slot assignment. The Block scheduler can only assign block tasks under the global aircraft-slot constraint. The labour requirements of the assigned blocks are then subtracted from the total slot capacity and passed as a new slot capacity Cap' to the OOP scheduler.

b) Slot Capacity: Figure 3 presents the capacity optimiser (Mode 2). This second system assigns a capacity allocation to each maintenance slot. A slot capacity provides the total number of hours available per labour skill, and how these hours are distributed across task types. As a result, each local scheduler operates against a fixed and independent capacity estimate. However, the initial aircraft-slot allocation still

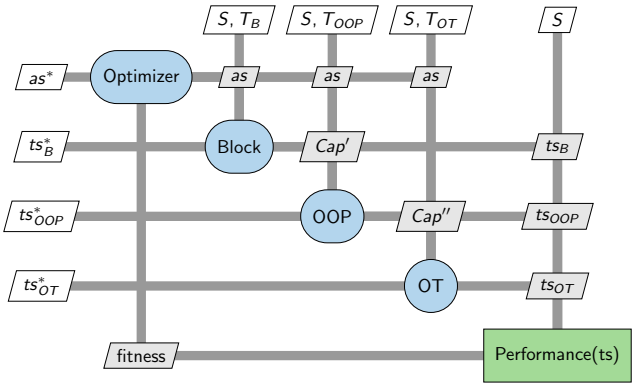

Figure 2. Mode 1. The optimiser searches for an optimal aircraft-slot assignment. as and ts correspond to the decision variables of the baseline. ts_B, ts_{OOP} and ts_{OT} correspond to the task-slot decision variables for each of the considered task types. Cap' and Cap'' refer to the slot capacity after deducting the requirements of tasks assigned by the Block and OOP schedulers, respectively. Figure generated using open-source PyXDSM [9].

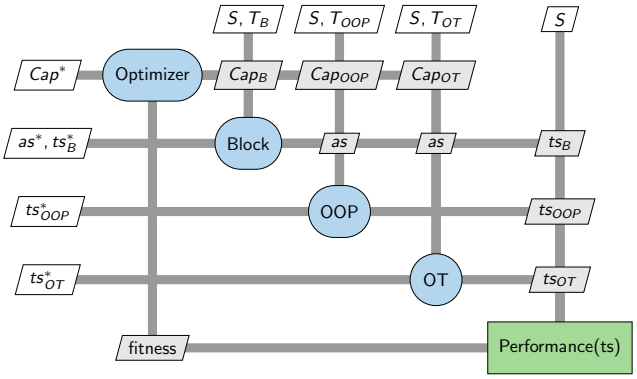

Figure 3. Mode 2. The optimiser searches for an optimal slot labour capacity partition.

heavily influences the quality of the schedule. In this mode, the Block scheduler determines the aircraft-slot allocation, which is then cascaded to the other local planners.

Concerning the capacity distribution, a naive approach could assume N continuous decision variables, one per task type, labour skill type, and slot, with $N = |\{B, OOP, OT\}| \times |ST| \times |S|$. This quickly becomes impractical: the search space is extremely large, valid ranges for each task-skill type must be defined, and the total allocation must be ensured not to exceed the maximum capacity. For now, we also consider a discrete set of pre-determined combinations of capacity allocations per slot. In the end, a solution is represented by $|S|$ categorical variables.

c) Aircraft-Slot and Slot Capacity: Figure 4 presents the simultaneous aircraft-slot and capacity optimiser (Mode 3). Finally, we merge the two previous architectures so that the global optimiser simultaneously decides the aircraft-slot and capacity assignments.

Aside from the three local planners, we need two other main components: (1) the Performance component, which evaluates the quality of a global solution by combining the results of each local optimiser, and (2) the Global optimiser, which outputs the shared resource allocation, receives the corresponding performance and iterates on the solution.

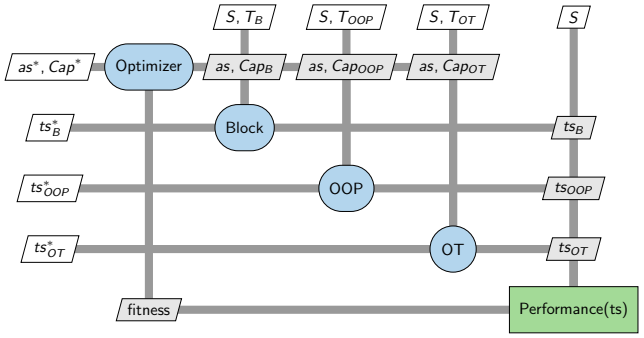

Figure 4. Mode 3. The optimiser simultaneously searches for an optimal aircraft-slot assignment and slot labour capacity partition.

The global optimiser is implemented using a GA. This choice is motivated by the following three factors: (1) it does not require gradient information about the local planners and makes few, if any, assumptions about their internals, (2) it can handle discrete (categorical) variables, and (3) it supports a heterogeneous solution space, which is particularly convenient for the simultaneous aircraft-slot and capacity optimiser (Mode 3). Whereas each MILP-based local optimiser leverages gradient information to steer the search of the solution space, the global optimiser can only assess the overall quality of each solution, and can be less sample efficient. Nevertheless, prior work has shown that such heuristic methods can perform well in airline maintenance scheduling [15].

We tested several combinations of hyperparameters, including the choice of parent selection algorithm, population size, number of parents per mating, crossover operators and mutation rates. Ultimately, we settled on a fixed set of parameters that demonstrated promising performance in initial trials. Generally speaking, we use tournament selection for parent choice, single and two-point crossover operators, as well as conservative mutation (affecting only 1 or 2 genes). For the aircraft-slot chromosome, a swap mutation is employed as it corresponds to a tail-swap operation. Otherwise, we employ random mutations.

The fitness of each solution is given at the output of a performance module. In our implementation, the fitness is obtained by simply adding up the objective values of all local planners. Since GAs conventionally maximise a fitness score rather than minimise an objective, we define the fitness to be symmetric to the baseline's objective ($f = -O$).

V. Hypotheses

The design decisions described in the section IV section are motivated by the following hypotheses:

1) An integrated central planner is likely to achieve the best solution, but may be too computationally expensive or inflexible for real-world scenarios.
2) In a distributed approach, each local planner solves a substantially simpler problem with fewer decision variables, potentially improving computational tractability at the expense of solution optimality.
3) While the assignment of block tasks is a primary factor, the overall quality of the A-check task schedule is also significantly influenced by the effective scheduling of OOP and OT tasks.
4) Providing an initial solution can accelerate convergence of the GA, but may cause the global optimiser to be stuck in a local optimum.
5) Distinct distributed architectures with varying degrees of reliance on global constraints and handling of shared resources will perform noticeably differently. Shifting greater control from domain-specific local planners to the generic global optimiser worsens performance.

TABLE IV. INTEGRATED SCHEDULER RESULTS

Dataset	Schedule	Wasted Interval (days)	Time to start (days)
Dataset A	Airline	34.464	76.916
	Integrated	**33.350**	**46.172**
Dataset B	Airline	44.533	84.657
	Integrated	**40.079**	**42.599**

VI. CASE STUDY

We validate our integrated scheduler and distributed architectures using real-world data from a major European airline. Specifically, we gathered task and slot information for a fleet of modern wide-bodied passenger aircraft during two separate operational seasons, henceforth referred to as Dataset A and B. Dataset A contains slightly fewer slots but a substantially higher count of out-of-phase and one-time tasks, resulting in a generally harder problem instance compared to Dataset B.

The cost and weight parameters defined in table II are set according to airline preference. Additionally, the costs are simply dependent on task type.

In practice, repetitive task due dates are specified in terms of aircraft utilisation. For this study, we assume that the flight schedule is available to estimate task due dates, which is a common assumption when planning at the seasonal level.

VII. RESULTS

The integrated and local schedulers are implemented using the commercial solver Gurobi [7]. We leverage the open-source genetic algorithms framework PyGAD [6] in the distributed architectures. All models are executed on a cluster with an Intel Xeon Gold 6130 CPU.

Performance is evaluated using three metrics, each corresponding to one of the sub-objectives (eqs. (1) to (3)):

- Number of task deferrals, per task type;
- Average wasted interval across repetitive tasks, reflecting how effectively the schedule exploits the task interval and thereby reduces the frequency of task execution within a fixed period.
- Average start time across one-time tasks, indicating how fast the scheduler works through a backlog of one-time tasks.

Section VII-A and Section VII-B present the results for the integrated and distributed schedulers, respectively.

A. Integrated

We ran the integrated scheduler for both datasets, with results summarised in table IV. The quality of the resulting schedules is comparable to the actual historical schedules provided by the airline, with slight improvements in the proposed performance metrics. It should be noted that the historic schedule is merely a record of the executed tasks and does not explicitly identify deferred tasks (except for one-time tasks assigned to other maintenance opportunities outside of A-checks). Additionally, it is not strictly constrained by the capacity estimates available to our optimisers. This makes a direct comparison of task deferrals non-trivial.

The resulting schedule is highly sensitive to the slot capacities $C_{(st,s)}$, and it is not simple to specify a single total capacity value that applies uniformly across all slots. Underestimating the capacity totals leads to a high number of deferred tasks, whereas overestimating them yields an overly optimistic maintenance plan. We observed, in some cases, that the model chooses to defer OOP tasks in favour of scheduling one-time tasks.

While the hierarchical multi-objective optimisation ensures that important tasks are not deferred in favour of reducing wasted interval, we demonstrate that similar results can be obtained by calibrating the weights in the blended objective. Furthermore, a blended objective allows for a direct numerical comparison of the objective of the integrated approach and the distributed architectures. Table V reports results for both blended and hierarchic multi-objective formulations on Dataset A, with solver time limits of 5 and 30 minutes. We observe that the blended approach shows little improvement beyond the 5 minutes. Although the hierarchical approach defers fewer tasks, it does lead to a higher wasted interval and time-to-start metrics.

B. Distributed

Table VI summarises the best results obtained for each of the developed models. Each distributed GA-based optimiser was run for a fixed number of generations (100), so that each run had a similar runtime to the baseline.

As expected, the integrated baseline achieves the best overall performance. The distributed architectures consistently underperform, with Mode 1 achieving the closest to the baseline and Mode 3 performing the worst. In Mode 1, the increase in OOP task deferrals relative to the baseline is mainly due to incompatibilities between due dates and the proposed aircraft-slot assignment.

The sequential, cascading nature of resource allocation in Modes 1 and 2 naturally restricts the solution quality. While this emulates some real-world planning practices, it also means that early, potentially suboptimal decisions by the first local scheduler can propagate and severely limit the solution space for subsequent stages. This greedy stage-by-stage approach, without a strong feedback loop or look-ahead mechanism, inherently sacrifices global optimality. Since Mode 3 simultaneously optimises both aircraft-slot and slot capacity, the complexity of the search space significantly increases, leading to the poorest performance.

The problem at hand is also heavily influenced by the specific labour requirement and slot capacity estimates. Modes 2 and 3 require a breakdown of slot capacity per task type, which can result in an unfair comparison with Mode 1 and the baseline if overestimated. Conversely, underestimating capacity in specific skill categories can force local planners to defer large sets of tasks. This is particularly relevant in the presence of outlier tasks with abnormally high specific labour requirements.

To accelerate convergence, we warm-start the optimisation with a custom initial solution rather than random values. For

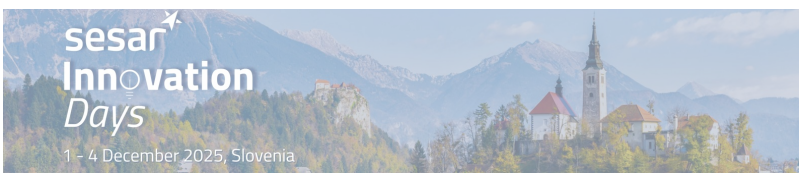

TABLE V. BLENDED VS HIERARCHICAL OBJECTIVE RESULTS FOR DATASET A

Multi-objective	Runtime (min)	Deferred block	Deferred OOP	Deferred OT	Wasted interval (days)	Time to start (days)
Blended	5	0	23	130	33.350	46.181
	30	0	23	130	**33.350**	**46.172**
Hierarchical	5	0	23	130	46.258	63.593
	30	0	**22**	**125**	53.135	57.077

TABLE VI. BEST RESULTS FOR EACH MODEL

Dataset	Model	Deferred Block	Deferred OOP	Deferred OT	Wasted Interval (days)	Time to start (days)	Objective	Runtime (min)
Dataset A	Integrated	0	23	130	33.350	46.172	4.59×10^6	30
	Mode 1	0	76	152	35.754	50.750	1.01×10^7	22
	Mode 2	0	102	298	46.205	56.370	3.18×10^7	16
	Mode 3	0	403	390	53.777	54.590	4.50×10^7	24
Dataset B	Integrated	0	7	55	40.079	42.599	2.33×10^6	8
	Mode 1	0	72	68	49.391	44.779	8.90×10^6	5
	Mode 2	0	209	342	51.861	50.152	6.09×10^7	7
	Mode 3	0	612	356	46.949	64.897	6.55×10^7	8

aircraft-slot assignment, we initialise the optimisation with the solution of the Block optimiser. Similarly, for capacity distribution, the initial assignment allocates all labour to block tasks, which dominate the objective function. This partly explains how the models can consistently schedule all the block tasks, often at the expense of other task types.

It is crucial to acknowledge that heuristic methods, such as GAs, are inherently stochastic, and their performance can vary significantly across independent runs due to the random operations like crossover and mutation. While a systematic and in-depth study of GA parameter tuning could further improve results, it was beyond the scope of this initial investigation.

VIII. DISCUSSION

Our discussion is primarily split into two main themes. Section VIII-A puts forward a comparison between the integrated and distributed approaches, while section VIII-B focuses on how airlines can deploy such models in an operational context.

A. Integrated vs Distributed Approaches

A key finding of this study is that the integrated scheduler is capable of producing maintenance schedules within minutes that are comparable in quality to those meticulously crafted by experienced airline planners over several weeks. This highlights the practical utility and efficiency of our centralised approach as a decision-support tool. Secondly, we observe a consistent underperformance of the proposed distributed architectures compared to the central planner. This degradation in solution quality, despite the potential benefits of scalability and flexibility, warrants a deeper analysis of the inherent nature of the A-check scheduling problem and the design choices of our distributed models.

We attribute the underperformance of the distributed architectures to a combination of factors: (1) the inherent rigidity and strong interdependencies within the seasonal A-check task scheduling problem, (2) the limitations in the current design of the distributed architectures, (3) the generic nature of the GA, and (4) the difficulties in estimating accurate task resource requirements and slot capacity from real airline data.

The A-check task scheduling problem, particularly at a seasonal level, exhibits a degree of inherent rigidity. Each aircraft typically undergoes roughly two A-checks per season, and the overall quality of the schedule is largely dictated by the assignment of the block tasks. This inherent structure suggests that the problem might not be easily decomposable without incurring significant performance penalties, as block assignment heavily constrains subsequent tasks. Therefore, while distributed approaches may offer scalability, achieving near-optimal solutions requires careful alignment with problem structure and the design of more sophisticated coordination mechanisms.

While the distributed modes did not outperform the integrated scheduler for the problem size studied, the individual sub-problems they solve are indeed much smaller and faster. The main challenge lies in effectively integrating these local solutions into a coherent global optimal plan. This suggests that while the current distributed architectures are not yet suitable for end-to-end seasonal A-check planning, their underlying principles could be refined for scalability in larger or different problems where a central MILP might become computationally intractable.

As mentioned in the section VII-B, the results are also highly sensitive to the estimated task labour requirements and slot capacity. Using each slot's historical capacity introduces strong biases in the resulting schedule, forcing schedulers to follow their past scheduling decisions. Averaging the slot capacity renders the problem vulnerable to outlier tasks with abnormally high requirements, some of which are attributable to data quality issues.

B. Operational Usage

Typically, airlines design the maintenance schedules manually, which requires the expertise of seasoned planners and is quite time-consuming. Our framework can serve as a decision-support tool by providing initial schedules or by illustrating how the composition of a task backlog impacts the quality of the resulting maintenance schedule.

While distributed architectures offer theoretical advantages in breaking down complex problems, our current findings indicate that, for the specific A-check scheduling problem at a seasonal level, they do not provide superior schedule quality or substantial runtime benefits compared to the integrated MILP. However, the distributed framework still offers valuable insights into specific aspects of resource allocation. For instance, planners can leverage these models to explore various capacity distribution strategies (e.g., how labour capacity is split between particular skills and task types) and fine-tune slot capacities.

C. Future Work

This work employs a GA, which, although robust for discrete-variable and gradient-free optimisation, may not be sufficiently tailored for the structure of this specific problem. The generic crossover and mutation operators of the GA may be insufficient to explore the constrained solution space efficiently or to resolve conflicts arising from shared resources. Incorporating more domain-specific knowledge will likely improve the GA's ability to steer the local planners to a global optimum. Incorporating domain knowledge in the GA, e.g. by implementing custom crossover and mutation operators that leverage the problem structure while respecting constraints, might significantly improve its search efficiency and solution quality.

Furthermore, to overcome the limitations of the current sequential and hierarchical distributed models, future work should explore more flexible multi-agent formulations. Instead of cascading decisions, such approaches could allow local optimisers (agents) to collaboratively propose solutions and negotiate for shared resources through sophisticated coordination mechanisms. Explicit conflict resolution techniques and robust feedback loops would be crucial to reconcile local optima into a globally coherent plan.

IX. Conclusion

This work introduces a novel A-check optimisation model, which schedules both repetitive and one-time maintenance tasks at a seasonal level for a fleet, while subject to labour capacity constraints. We demonstrate the effectiveness of our integrated scheduler with a case study from a major European airline with data from a fleet of wide-bodied passenger aircraft, achieving a slight improvement over the historical schedules. It is important to emphasise that even marginal improvements in metrics like wasted interval can translate into substantial operational cost reductions for airlines.

Furthermore, we propose a set of distributed architectures which attempt to divide the scheduling problem into different stages in a global optimisation process. Each local scheduler is tasked with planning tasks of a specific type and, therefore, solves a much smaller problem than the integrated scheduler. The global scheduler is tasked with allocating shared resources and limiting the solution space of the local planners. Their performance of the distributed schedulers falls short of that of the integrated scheduler. This highlights the advantages of a central planner leveraging the problem's gradient information.

From a practical perspective, this research offers valuable tools for airlines. The integrated MILP scheduler can serve as a powerful decision-support system, providing optimised seasonal A-check plans that account for diverse task types and resource constraints. Furthermore, the insights gained from the distributed architectures can inform the design of more flexible and scalable planning systems, allowing airlines to manage the inherent complexity of maintenance scheduling more effectively. Future work will focus on exploring alternative decomposition strategies (multi-agent and/or conflict resolution mechanisms), and investigating the scalability of these approaches for even larger and more complex maintenance planning scenarios.

References

[1] Pedro Andrade, Catarina Silva, Bernardete Ribeiro, and Bruno F. Santos. Aircraft Maintenance Check Scheduling Using Reinforcement Learning. *Aerospace*, 8(4):113, 2021.

[2] 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.

[3] Qichen Deng, Bruno F. Santos, and Richard Curran. A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization. *European Journal of Operational Research*, 281(2):256–273, 2020.

[4] Sheldon H. Dike. Project scheduling with resource constraints. *IEEE Transactions on Engineering Management*, EM-11(4):155–157, December 1964.

[5] Chengjin Ding, Yuzhen Guo, Jianlin Jiang, Wenbin Wei, and Weiwei Wu. Aircraft routing and crew pairing solutions: Robust integrated model based on multi-agent reinforcement learning. *Aerospace*, 12(5), 2025.

[6] Ahmed Fawzy Gad. Pygad: An intuitive genetic algorithm python library. *Multimedia Tools and Applications*, pages 1–14, 2023.

[7] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2025.

[8] International Air Transport Association (IATA). Airline maintenance cost executive commentary, 2023. Accessed: 2024-06-08.

[9] Andrew B. Lambe and Joaquim R. R. A. Martins. Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. *Structural and Multidisciplinary Optimization*, 46:273–284, 2012.

[10] J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of Scheduling under Precedence Constraints. *Operations Research*, 26(1):22–35, 1978.

[11] Haonan Li, Marta Ribeiro, Bruno Santos, and Iordanis Tseremoglou. Prediction of Non-Routine Tasks Workload for Aircraft Maintenance with Supervised Learning. In *AIAA SCITECH 2024 Forum*. American Institute of Aeronautics and Astronautics, 2024.

[12] Jean-Baptiste Sciau, Agathe Goyon, Alexandre Sarazin, Jérémy Bascans, Charles Prud'homme, and Xavier Lorca. Using constraint programming to address the operational aircraft line maintenance scheduling problem. *Journal of Air Transport Management*, 115:102537, 2024.

[13] Syed Shaukat, Mathias Katscher, Cheng-Lung Wu, Felipe Delgado, and Homero Larrain. Aircraft line maintenance scheduling and optimisation. *Journal of Air Transport Management*, 89:101914, 2020.

[14] Paul J. Van Kessel, Floris C. Freeman, and Bruno F. Santos. Airline maintenance task rescheduling in a disruptive environment. *European Journal of Operational Research*, 308(2):605–621, 2023.

[15] Tim Van Der Weide, Qichen Deng, and Bruno F. Santos. Robust long-term aircraft heavy maintenance check scheduling optimization under uncertainty. *Computers & Operations Research*, 141:105667, 2022.

[16] Max Witteman, Qichen Deng, and Bruno F. Santos. A bin packing approach to solve the aircraft maintenance task allocation problem. *European Journal of Operational Research*, 294(1):365–376, 2021.

[17] Yifan Xu, Sebastian Wandelt, and Xiaoqian Sun. Airline scheduling optimization: Literature review and a discussion of modelling methodologies. *Intelligent Transportation Infrastructure*, 3:liad026, 2024.