Un modèle de programmation par contraintes pour la maintenance opérationnelle aéronautique

J.-B. Sciau[1,2*] **A. Goyon**[1] **A. Sarazin**[1] **J. Bascans**[1] **C. Prud'homme**[3] **X. Lorca**[2]

[1] Airbus Protect, Blagnac, [2] CGI, IMT Albi-Carmaux, Albi, [3] TASC, IMT-Atlantique, Nantes
[1]{prenom.nom}@airbus.com [2]{prenom.nom}@mines-albi.fr [3]{prenom.nom}@imt-atlantique.fr

Résumé de l'article paru dans le Journal of Air Transport Management, Vol.115 : 0969-6997 (2024), Elsevier.

Résumé

Les compagnies aériennes cherchent à optimiser leur maintenance pour maximiser le potentiel opérationnel de leur flotte. Leurs objectifs est réduire les coûts de maintenance et d'optimiser la gestion des ressources. Le problème de planification de la maintenance opérationnelle aéronautique est décrit comme un Resource Constrained Project Scheduling Problem (RCPSP). Un modèle de Programmation Par Contraintes (PPC) est utilisé pour résoudre le problème. Une comparaison avec un solveur industriel est proposée sur des instances réelles.

1 Introduction

La maintenance joue un rôle majeur dans le transport aérien. Elle est obligatoire pour assurer la sécurité des passagers mais représente une part importante dans le budget d'une compagnie, entre 10 et 20 % des coûts opérationnels [4]. Aujourd'hui, la majorité de la maintenance aérienne est préventive [3]. Cet article s'intéresse plus particulièrement à la maintenance à court et moyen terme (de quelques jours à quelques mois), dites de "ligne". L'idée est de pouvoir profiter des courts créneaux de maintenance, lorsque l'avion est au sol entre deux vols, afin d'exécuter de "petites" tâches de maintenance (inspections, etc...) pour réduire le temps de maintenance sur les longs créneaux (le week-end ou la nuit). Par conséquent, cette approche permet d'augmenter la disponibilité opérationnelle d'un avion.

La section 2 détaille le problème de maintenance opérationnelle. La section 3 présente la formalisation du RCPSP et sa résolution en utilisant la PPC. Enfin, une comparaison avec un solveur heuristique industriel est proposée en section 4, utilisant des instances réelles.

*Papier doctorant : J.-B. Sciau[1,2] est auteur principal.

2 Description du problème

L'optimisation de la planification fait partie d'un processus global de la maintenance opérationnelle. Tout d'abord, les tâches à effectuer sont identifiées en respectant un échéancier suivant des programmes de maintenance. Ces documents procurent tous les renseignements nécessaires à la planification d'une tâche (la description et l'ordre des différentes actions à réaliser, leurs durées et les ressources requises). Dans la maintenance préventive, les tâches se répètent en respectant une durée maximale entre deux occurrences appelée période. Ceci forme une contrainte de périodicité et fixe une date de butée dans l'exécution des tâches.

D'autre part, la planification de la maintenance doit s'inscrire dans un cadre opérationnel. Les *créneaux de maintenance* sont déduits des plannings de vols. Les ressources de maintenance sont organisées en trois types : les *ressources humaines* (techniciens) maîtrisant parfois plusieurs compétences et ayant leurs propres horaires de travail ; les *équipements sérialisés* (grues, vérins hydrauliques) devant être réservés en avance ; enfin les *ingrédients*, suivis de manière logistique, pouvant être consommables ou réutilisables.

Afin de fournir un planning directement exécutable, le planning doit être précis, à la minute près. Pour cela, trois niveaux de granularité sont distingués.

- La granularité de la *tâche* regroupant les informations générales (périodicité, date butoire).
- Plusieurs *opérations* sont décrites à l'intérieur d'une tâche pouvant être ordonnées.
- Des *activités* élémentaires composent la réalisation d'une opération. Une activité est spécifiée par une ressource requise, une durée et des zones d'intervention sur l'avion.

3 Modélisation

Le problème de maintenance aéronautique peut se modéliser sous la forme d'un RCPSP [1]. Il nécessite toutefois certaines extensions par rapport à la forme standard. Tout d'abord, le problème étudié est un Multi-Skill RCPSP (MSRCPSP) du fait de la multi-compétences des ressources humaines. D'autre part, certains ingrédients sont consommables et leurs stocks diminuent après l'exécution de l'activité. Ceci rajoute la particularité de prendre en compte des ressources non renouvelables dans le RCPSP étudié. Enfin, le problème d'optimisation est multi-objectifs. Les planificateurs veulent d'abord planifier le plus de tâches possible puis utiliser le moins possible de ressources additionnelles et enfin planifier le plus proche de la date butoire pour maximiser le potentiel de l'avion. Un ordre lexicographique permet de gérer les priorités imposées par le métier.

La Programmation Par Contraintes (PPC) permet de modéliser et de résoudre efficacement ce problème. En effet, elle permet de répondre aux exigences du métier. Il est relativement facile d'activer ou désactiver des contraintes pour s'adapter aux besoins de chaque compagnie [5]. De plus, les planificateurs demandent un temps de résolution assez rapide (moins de 5 minutes) et de pouvoir reproduire les solutions. Ceci leur permet de comparer différents scénarios de maintenance et de choisir le meilleur. La PPC, étant une méthode déterministe, garantie la reproductibilité des solutions. Enfin une stratégie de recherche constructive a été implémentée permettant de trouver rapidement une première solution satisfaisante d'un point du vue métier. Le processus de décision, suivant une expertise métier, est explicable et compréhensible par les experts.

Le modèle de PPC est décomposé en trois types de contraintes. Des contraintes sont spécifiques à la *maintenance*, par exemple la planification d'une tâche nécessite la planification de toutes ses activités. Les contraintes de *précédence* modélisent l'ordonnancement des opérations, les disjonctions entre les tâches et la contrainte de périodicité. Enfin, les contraintes *cumulatives* vérifient les capacité des ressources. La multi-compétences des ressources humaines et leurs emploi du temps individuel sont gérés grâce à une contrainte de type DiffN définie par [2].

4 Expérimentation

Un cas pratique sur des instances réelles de maintenances est proposé afin de comparer les résultats obtenus avec *Choco-Solver* et un solveur heuristique actuellement utilisé par *Airbus Protect*. L'heuristique utilisée est similaire à celle utilisée dans [6]. La solution est construite en deux étapes. D'abord les tâches sont planifiées par ordre de priorité et ensuite les ressources sont affectées à chaque activité. La taille des instances utilisées varie d'un horizon d'une semaine à deux mois pour une flotte de quatre avions maximum. Ceci représente des RCPSP entre 319 et 8795 activités.

En comparant les objectifs suivant l'ordre lexicographique (Table 1), le solveur PPC est capable, sur trois instances, de planifier plus de tâches que le solveur heuristique. Dans la majorité des cas, il utilise moins de ressources additionnelles et présente une plus petite déviation par rapport à la date de butée. Même si le solveur PPC prend plus temps à retourner une première solution, il respecte quand même la limite du temps imposée de cinq minutes.

Inst.	Tâches planifiées	Res. add. *($)*	Déviation *(min.)*	Résolution *(sec.)*
1	0	-1330	0	+0.123
2	+1	-2660	-7	-0.463
3	0	-1862	+1	+9.442
4	+1	-5852	-113	+12.842
5	+13	14231	-48	+133.155

Table 1 – Comparaison des solveurs[1]

1. Comparaison du solveur PPC par rapport au solveur heuristique en valeurs absolues.

Références

[1] Christian Artigues, Sophie Demassey et Emmanuel Neron : *Resource-Constrained Project Scheduling : Models, Algorithms, Extensions and Applications*. ISTE/Wiley, 2008.

[2] N Beldiceanu et E Contejean : Introducing global constraints in CHIP. *Mathematical and Computer Modelling*, 20(12):97–123, 1994.

[3] FAA : Advisor circular 43-12a CHG 1-preventive maintenance, 2007.

[4] U PeriyarSelvam, T Tamilselvan, S Thilakan et M Shanmugaraja : Analysis on costs for aircraft maintenance. *Advances in Aerospace Science and Applications*, pages 177–182, 2013.

[5] Francesca Rossi, Peter van Beek et Toby Walsh : *Handbook of Constraint Programming*. Elsevier, 2006. Google-Books-ID : Kjap9ZWcKOoC.

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

Using constraint programming to address the operational aircraft line maintenance scheduling problem

Jean-Baptiste Sciau [a,c,*], Agathe Goyon [a], Alexandre Sarazin [a], Jérémy Bascans [a], Charles Prud'homme [b], Xavier Lorca [c]

[a] *Airbus Protect, 36 rue Grimaud, Blagnac, 31700, France*
[b] *IMT Atlantique, 4 Rue Alfred Kastler, Nantes, 44300, France*
[c] *IMT Mines Albi, All. des sciences, Albi, 81000, France*

A B S T R A C T

Maintenance plays a major role in air transport management. Airlines are looking to reduce aircraft unavailability. Optimizing maintenance is a perspective to increase the operational potential of aircraft and offers novel managerial implications. Maintenance tasks are traditionally organized in periodic blocks of activities. In the modern aviation, more and more maintenance jobs can be performed between two flights, which is called Line Maintenance Scheduling Problem (LMSP). The expected goal of our paper is to propose an operational schedule that can be directly executed by maintainers, i.e. assigning a start time to each elementary activity and a resource to perform it. This new problem is named Operational Aircraft Line Maintenance Scheduling Problem (OALMSP). A scheduling assistant could help airlines to reduce maintenance costs and resource management. Planners have to respect task deadlines imposed by regulations, precedence constraints between certain operations and also ensure the availability of resources in order to perform specific actions. This problem is an extension of a Resource Constrained Project Scheduling Problem (RCPSP). In this article, we propose an industrial application of an automatic aircraft line maintenance scheduler based on a Constraint Programming (CP) model. The flexibility of our approach means we can easily adapt to airline use cases without changing the properties of the model. The optimization problem is generally made up of several objectives ordered according to their importance. The objectives are respectively to plan as many tasks as possible according to their priority, then to minimize both the use of resources and the deviation time between scheduled dates and target dates of tasks. The lexicographical order enables the use of human reasoning and the management of business priorities. A constructive search strategy is designed to compute a satisfying schedule within an acceptable execution time for industry application. A practical use case based on real airline data is presented and the results are compared with those found by an industrial solver taken as a reference.

1. Introduction

Airlines continually strive to improve their Air Transport Management (ATM), developing processes to achieve their goals with optimal resources utilization. To be attractive, airlines must provide a personalized service to their customers and ensure passenger safety. Enhancing aircraft availability is one of the crucial concern airlines need to address when managing ATM. One solution to maximize operating aircraft time is to focus on the fleet maintenance in order to optimize resource allocation, reduce maintenance costs, and increase aircraft availability. Although maintenance is mandatory to ensure passenger safety, it temporarily disables the aircraft from operational activities. Optimization of maintenance management can be a competitive advantage for an airline. Therefore, aircraft maintenance planning is both a safety and a strategic issue for airlines.

Aircraft Scheduling Problem has been studied in many different ways and can be classified in several sub-problems which have different goals and can be solved sequentially (Barnhart et al., 2003). Firstly, the *schedule design or flight schedule* focuses on strategic plans over several years based on marketing studies to determine city connections. Based on this schedule, the *fleet assignment* determines the type and the configuration of the aircraft to perform a flight. This planning can be done approximately one year before in advance. After that, the *crew assignment* and the *aircraft routing* involve tactical decisions

made a few months ahead. The crew assignment ensures that every flight is assigned to a qualified crew, taking into account the employee availability and work rules. The aircraft routing manages the sequence of flights of one aircraft ensuring the overall maintenance plan and that all flights are covered once. Lastly, the *tail assignment* can be seen an operational extension of the aircraft routing which adjusts, a few days before operations, the planned routes and assigns a unique aircraft (identified by its tail number) respecting its own maintenance constraints (Gabteni and Grönkvist, 2009).

Maintenance planning ensures the safety and reliability of the service but it is expensive considering man hours, the cost of spares, and delays that may occur. According to a study by the International Air Transport Association (IATA) on 37 airlines in 2021 (Cros, 2022), direct maintenance costs equate to $3.12M per aircraft. Maintenance represents 10–20% of the total operating costs of an airline (PeriyarSelvam et al., 2013). Some maintenance tasks are mandatory and can cause flight delays or even cancellations if not performed. These are critical Aircraft On Ground (AOG) situations. According to a study by Badkook (2016) on 28 aircraft of the same model and using AOG data collected from maintenance and finance departments for one year, average flight delay costs amounted to $22753.53. This is why it is very important for planners to find the best time to schedule a task both respecting regulations and controlling maintenance costs. In other words, the aim is to maximize the potential of tasks while ensuring the safety of the aircraft. Maintenance has two economical impacts on airlines. Firstly, it incurs direct maintenance costs, which involve expenses related to task execution, such as man hours and equipment costs. Secondly, the indirect costs are generated as consequences of the maintenance actions. They include, as examples, the aircraft unavailability, the processing time and man power required to draw up the maintenance planning or the cost of operational interruptions such as delays or flight cancellations due to unexpected events.

At present, to respect regulations and recommendations, most airlines use a Preventive Maintenance process detailed in the Advisory Circular n° 43-12 A of the Federal Aviation Administration (FAA, 2007). Depending on their country and routing plans, airlines have to respect airworthiness regulations. Tasks are defined in technical documents written by aircraft manufacturers and airlines. These documents detail the successive actions which have to be performed by a specific type of resource and the specific equipment required.

Maintenance tasks are typically organized into packages called ‘‘maintenance checks’’. Maintenance can be classified into two main types. *Heavy maintenance* is a grouping of large tasks that need to be performed in a hangar. On the other hand, *line maintenance* refers to simpler tasks which encompass routine inspections and minor procedures. These tasks can be performed between two flights when the aircraft is on the ground. The objective of Line Maintenance Scheduling Problem (LMSP) as defined in Shaukat et al. (2020) is to determine the starting time and the location of each maintenance job within a specified planning horizon. The input data includes the flight plan, the fleet of aircraft, and the list of maintenance jobs to be carried out for each individual aircraft of this fleet. This scheduling problem focuses on the short term and operational aspect, aiming to minimize the overall deviation from deadlines. Carrying out some ‘‘short’’ maintenance jobs during line operations, i.e. between flights, may provide airlines with a strategic advantage. This approach helps to reduce the time spent on routine maintenance checks, subsequently increasing the availability of the aircraft (Hughes, 2006).

One of the main difficulties of the LMSP is the resource allocation. It requires to check the availability of the required resources during the scheduled maintenance slot to perform a task. Thus, LMSP can be seen as Resource Constraint Project Scheduling Problem (RCPSP) considering both task priority relationships and resource capacity constraints. However, the existing literature on LMSP, as found in Callewaert et al. (2018), Lagos et al. (2020), Shaukat et al. (2020), does not explicitly consider scheduling details within a task and does not schedule at the minute level of granularity. Indeed, from a Maintenance Control Center (MCC) point of view, maintenance tasks are structured into sub-tasks of individual activity. Each activity is subject to various constraints such as access capacity constraints, precedence relations or synchronization constraints. To create a schedule that can be directly executed by maintenance technicians, we delve into the scheduling process at a finer level of granularity, looking at the smallest activity within a task and scheduling up to the minute. We called this new problem the Operational Aircraft Line Maintenance Scheduling Problem (OALMSP).

Today, most operational planning is carried out by humans. They are able to decide when and where to plan a task thanks to their expertise. However, due to the complexity of the problem, it is sometimes difficult for a human to identify the best way to ensure that all of the constraints are met while optimizing maintenance costs. In addition, planning is time-consuming and it is complex for a human to analyze all the combinatorial solutions which can exceed thousands of possibilities.

An industrial solution developed by AIRBUS PROTECT (Airbus, 2022) proposes a simulation of maintenance planning for a fleet of aircraft. This solver enables the state of maintenance of the fleet to be visualized alongside a detailed outline of required tasks in a given maintenance slot. By setting up certain parameters, the user can very quickly simulate a situation and choose the right configuration or test a scenario. The heuristic used mimics the practice of experts and it follows the principles of Shaukat et al. (2020), which consists in two consecutive stages, *task assignment* then *timetabling*. The Key Performance Indicators (KPIs) highlight the value assessment of the planning. Based on heuristics, this smart virtual assistant can generate a planning proposal in a few minutes. Manual adjustments in the planning are then allowed and a dynamic rescheduling can be triggered.

In this context, a maintenance optimization digital assistant could be profitable for airlines. This tool could enable new maintenance opportunities to be explored that reduce maintenance costs by optimizing resource utilization, reducing additional rental costs, and minimizing task execution time. Moreover, this digital assistant could quickly provide valuable decision support by evaluating multiple maintenance scenarios simultaneously to help planners select the most accurate option. To be truly effective, the solution must be quickly understandable by a human and also reproducible. Planners must validate the schedule and assess the benefit of the solution. Furthermore, maintenance strategies can vary from one airline to another depending of their fleet size, route plan, maintenance teams or service contract. Thus, the digital solution needs to be flexible to address airlines use cases, allowing the addition or modification of a constraint or an objective without changing the model itself.

The following contributions of this papers can be summed up into two points:

1. We model the OALMSP as an RCPSP to plan an operational schedule that can be executed by technicians. The smallest task granularity is considered in the model with a one-minute precision. Synchronization and precedence relations between activities inside a task are respected. Resources availability are considered and they are allocated to each elementary activity. Schedule optimization is based on the resources utilization and the potential of parts to reduce maintenance costs.
2. We propose a Constraint Programming (CP) model, adaptable to airlines use cases and understandable by planners. CP has been chosen for its flexible aspect, to enable or disable constraints or to modify objectives order without compromising model itself. Additionally, CP approaches are suited to the RCPSP (Schutt et al., 2013). A constructive search following technical expertise is declared in order to improve the problem-solving and an initial satisfying solution is found in a reasonable time. The solutions are analyzed with reference to particular KPIs, measuring the number of tasks scheduled, the utilization of resources and the potential loss between the scheduled date and the task’s

deadline. The expected goals are to find a first solution in less than five minutes, one that closely aligns with the results of the industrial solver and improving certain KPIs. Experiments are carried out on real maintenance instances.

This paper is organized as follows. Firstly, Section 2 describes in greater detail the industrial context of line maintenance and the operational scheduling issues. Then, Section 3 presents a brief literature review of Aircraft Scheduling Problems and places our work in the context of recent papers around the LMSP. Section 4 recalls the standard RCPSP formalism and proposes some adaptations to model the LMSP. After presenting briefly a background of CP and the motivations of using it, Section 5 proposes a model for the OALMSP and describes all constraints divided in three groups: maintenance constraints in 5.3.4, precedence constraints in 5.3.5 and cumulative constraints in 5.3.6. Some extensions will be introduced in 5.4 with regard to the industrial context. The objectives are explained in 5.5 as well as an outline of the search strategy in 5.6. Experiments are divided into two parts. Section 6.1 proposes an evaluation of the CP model on a small representative instance and a comparison between Constraint Programming and Mixed Integer Programming solvers. Then Section 6.2 highlights a practical use case of aeronautic maintenance. The industrial solver taken as reference is described in Section 6.2.1 and compared to the heuristic used in Shaukat et al. (2020). The performances of the CP model and the industrial solution are detailed in Section 6.2.2 on several instances of a real aeronautic maintenance use case. Finally, Section 7 provides our conclusions and exposes future perspectives for an industrial scheduling process.

2. Industrial problem description

2.1. Maintenance context

The aircraft maintenance problem consists in finding a time for each task to perform at fleet level. The assignation of tasks must respect operational constraints such as the availability of a location and the required human and material resources. Nowadays, preventive scheduling is organized following a maintenance program produced by the aircraft manufacturer and customized by airlines. These documents provide the description of the task execution and the periodicity expressed in various units such as flight hours (FH), flight cycles (FC) or calendar days (Cal). According to this information, a deadline to execute each task can be calculated and mandatory constraints are formalized.

The majority of airlines use an A-B-C-D check system as suggested in Kinnison and Siddiqui (2013). Maintenance tasks are regrouped in check packages depending on their periodicity. The organization of these checks depends on the aircraft type and the airline. The maximal interval to execute the check corresponds to the shortest due date given in FH, FC or Cal, depending on the utilization of the aircraft. Currently the maintenance program is organized as follows:

- A-checks are maintenance checks performed approximately every month (or every 300 flight hours for example). They consist of visual inspections of the airframe, circuit power, avionics and accessories. They check all major systems of the aircraft such as landing gear, engines and control surfaces. An A-check takes about eight hours of downtime.
- B-checks are similar to A-checks with additional tasks, such as inspection of panels, cowlings, fluid servicing and lubrication. It is performed on average every three months and involves the checking of stabilizers and ailerons. This type of check takes about a day to perform in full.
- C-checks are carried out usually about once a year. They include the detailed inspection of the airframe and engines. A full lubrication is performed and corrosion is analyzed on the structure. Major mechanisms and systems are tested and flight controls are calibrated. It requires around one week to complete a C-check.
- D-checks are performed about every 4–6 years. The whole structure is inspected thoroughly and the cabin interiors are removed. These checks stop the aircraft on average for one month.

Task maintenance can be divided into two main levels. Fig. 1 proposed by Van den Bergh (2013) illustrates this classification.

- *Heavy maintenance* regroups LONG-term checks as C-checks and D-checks. These large tasks must be performed in a hangar.
- *Line maintenance* focuses on MID-term checks such as A-checks and B-checks, some of which may require a hangar. Line maintenance includes also several SHORT-term smaller tasks such as inspections, Transit checks, Daily checks and Weekly checks. These recurring tasks are set up by the airlines to consolidate their maintenance. Most of these maintenance actions can be performed at the gate during the ground time of the aircraft.

Most of the time, maintenance is carried out while the airplane is at a standstill. No flight is possible during maintenance actions. In the Line Maintenance Scheduling Problem (LMSP), the flight planning is provided by the airline and maintenance slots are deduced from it. It is also important to check if the slots are compatible to receive maintenance tasks (e.g., maintenance localization, hangars, equipment, etc.). Two levels of slots can be identified:

- SHORT-term slots where the aircraft stays at the gate between two flights, able to receive only small line maintenance tasks. These slots are also called Turn Around Time (TAT).
- MID-term slots where the aircraft goes to a hangar, designed to receive heavy LONG-term tasks (C or D-checks) and some MID-term tasks (such as some A or B-checks) which cannot be performed at the gate and requiring a hangar. These slots can also accept SHORT-term line tasks.

Heavy maintenance is usually scheduled several years ahead, in other problems such as the Aircraft Maintenance Check Scheduling (AMCS) (Deng et al., 2020). This article covers the LMSP taking routine preventive SHORT-term and MID-term tasks into consideration. The primary economic goal is to schedule these routine tasks close to their due dates to maximize their potential. Scheduling line maintenance tasks, even for the smallest ones, is complex due to their high frequency and limited opportunities, making slot and resource allocation challenging. In addition, corrective and predictive maintenance are not included in this study because they serve different goals, such as scheduling corrective tasks as early as possible.

2.2. Operational line maintenance planning

Scheduling can be divided into three stages (Abreu and Piedade, 2018): *strategic*, *tactical* and *operational* scheduling. *Strategic planning* provides a high-level view of the organization a long time in advance (about five years). It helps to define the global vision, the financial aspect and the human resources. *Tactical planning* works on a medium-term horizon (typically around one year) to achieve the strategy defined by the previous planning. It includes logistical and technical aspects, synergies between operations, and skills development. In this phase, the process flow is analyzed through reports and tactical decisions are made to fine-tune the process. Finally, the *operational planning* schedules for day-to-day operations to make the planned actions feasible and manage functional failures.

LMSP is part of the operational maintenance process. Usually, airlines mange their maintenance planning in the Maintenance Control Center (MCC). Fig. 2 sums up the maintenance process into three steps. First, the identification of maintenance tasks to perform according the health condition of the aircraft and its maintenance program. Then, planners look for a maintenance slot and resources to schedule the identified tasks by optimizing as much as possible the planning. At this stage, flights schedule of the aircraft are known as well as the

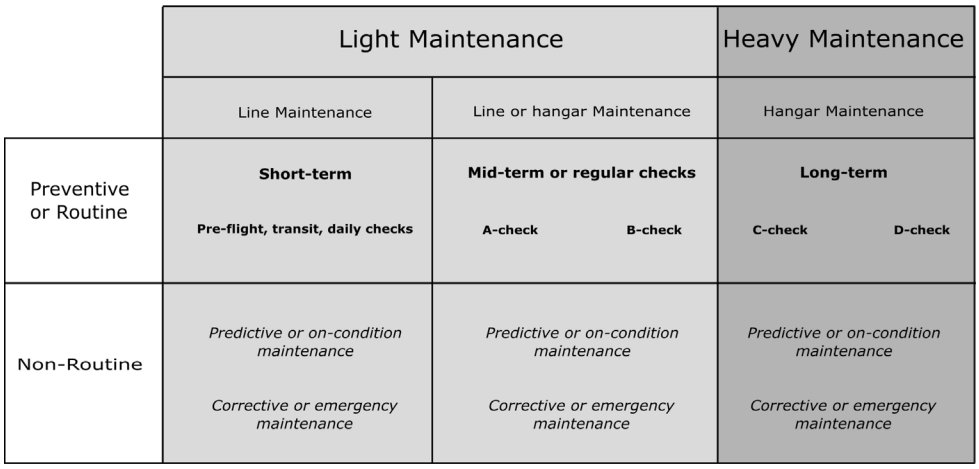

Fig. 1. Taxonomy for aircraft maintenance.
Source: Van den Bergh (2013).

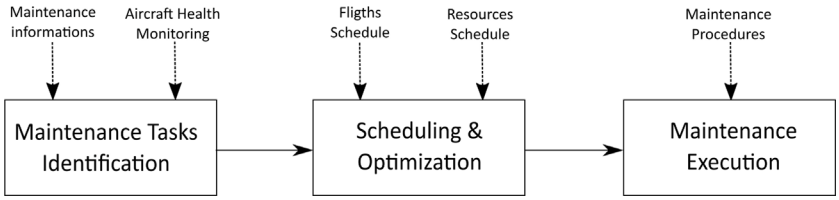

Fig. 2. Process of operational maintenance planning.

timetabling of resources. Finally the maintenance planning is sent to maintenance teams to perform the tasks. Planning must take into account as many details as possible. For instance, conflicts or priorities between activities must be considered to be as operational as possible. The horizon time for an operational planning is a few months, after that the planning is submitted to over incertitude (for instance resource timetabling) and the operational aspect is not longer relevant.

The Operational Aircraft Line Maintenance Scheduling Problem (OALMSP) considered in this study distinguishes three levels of granularity:

1. **Task granularity**: This level looks at *tasks* that are ended before their due dates and the matching of allocated slots (duration and type). Tasks may belong to periodic series and must be executed before a maximal period which acts as a deadline. Tasks in a series must respect their position and the interval between them has to be less than the series's period. Indeed, when a task is scheduled with anticipation, the deadlines of subsequent tasks in the series must be adapted. This is a precedence constraint called *periodicity*. Finally, some tasks are incompatible between each other because of disjunction criteria. For instance, some tasks require electrical power off and others power on.
2. **Operation granularity**: Inside each task, maintenance sub tasks called *operations* are detailed with their requirements and estimated duration. The order between operations within a task might be specified and must be respected. For instance, oxygenation operations cannot be performed at the same time as refueling for security reasons.
3. **Activity granularity**: An operation is composed of several requirements, also called *activities*. Each activity requires one specific resource (a human resource or an item of equipment respectively with the right skill or reference) during a given duration, accurate to the minute. All activities in an operation must be executed at the same time.

In this scheduling problem, an activity must be assigned to an available resource with the required skill. Three main categories of maintenance resources can be distinguished:

- *Human resources or staff*: These are the maintenance teams (technicians) characterized by one or several mastered skills (including accreditation). Working time of each human resource is given as input.
- *Serialized equipment*: This equipment consists of metric devices, tools, cradles, cranes, etc. necessary for technicians to perform the task and must be booked prior to the scheduled maintenance. They are all identified by a serial number and a unique reference (equivalent to skill levels of staff). Their availability is known in advance.
- *Ingredients*: These spares (screws, glue, bolts, etc.) are organized in batches. They are followed from a logistical aspect (stock available and next deliveries, specifying quantity ordered and delivery date). Ingredients can be reusable or consumable (stock decreases of the units consumed by the activity).

All resources have a localization (city, hangar, store, etc.). There are restrictions on the number of human resources allowed in aircraft zones and access panels. Staff and serialized equipment can be considered as *temporal resource* which can be characterized by two states: *unavailable* (off working hours or occupied on a task) or *free*. Moreover, all activities are *non-preemptive*. A resource must fully complete an activity before moving to another one, and an activity cannot be interrupted once it is started. To give an example, a Weekly check can be composed of 47 activities between 30 min and five hours including 25 activities requiring human resource in three skills, one tool and 21 ingredients.

2.3. Business requirements

The main goal of developing a digital assistant for maintenance optimization is to respond to business needs. Each airline has its own organizational and tactical challenges. They do not have the same types

of aircraft, the same maintenance bases, the same resource management, the same subcontractor contracts or the same logistics deliveries. In order to adapt to each specific airline use case, the proposed approach needs to be flexible. Constraints can be adapted or new ones can be added. For example, an airline might have an important logistical inventory and might be able to relax some of the resource constraints on ingredients. Then, depending on the airport, some restrictions may or may not apply as the registration is different between countries. On the other hand, the priorities in the objectives may differ from one airline to another. Some minor optimization objectives could be added. Some ideas are discussed at the end of Section 5.5. In this study, we set a standard lexicographical order that most airlines use, but which can be modified. The first goal is to schedule the highest number of tasks while respecting task priority (MID-term then SHORT-term). Then, the solution may use the least additional resources. Finally, the deviation between the starting time of tasks and their respective deadline must be reduced as much as possible to minimize potential loss. However, airlines with subcontractor agreements may have a preference for scheduling close to the task deadlines to conserve the potential of the parts even if additional resources are used, while an airline without a subcontractor contract will prioritize scheduling with its own staff team.

In addition, the scheduling assistant must be robust to changes in maintenance programs or new situations where there is no historical maintenance data. It is common that new airworthiness regulations are in place and new tasks (such as service bulletins) must be integrated. Another case can be the integration of new aircraft types into the fleet. The algorithm must be able to react to these special cases.

To address the OALMSP, the digital scheduler model must return a solution in a reasonable time frame. The accepted solving time for an expert is less than five minutes for a medium fleet of about 10 aircraft and a scheduling horizon of about a month. The solver should answer quickly to allow planners to compare multiple configurations and scenarios and to select the best option. Then, the solution must respect constraints at task level (periodicity, disjunction, ...) as well as the constraints at operation level. Scheduling accuracy must be up to the minute to respect precedence relations between operations and activities synchronization inside an operation.

Another requirement of planners is that the scheduling solution returned by the digital assistant must be deterministic and reproducible. Industrialists seek to use the scheduler as a simulator of scenarios and to be able to reproduce their simulations and compare them while adjusting certain parameters. Otherwise, experts need to quickly understand the generated solution and optimization choices in order to make some adjustments. The algorithm must follow the current practice of human reasoning in order to be accepted.

Finally, to quickly evaluate a scheduling solution, experts look at some key performance indicators (KPIs), such as the number of scheduled tasks, the cost of additional resources used, and the average deviation time of tasks between their start time and their due date. The solving time is also considered as an acceptance criterion.

3. Literature review

Aircraft Scheduling Problem has been studied from multiple aspects. Levin (1971) was the first to solve a fleet routing planning with an Integer Linear Model. Bird (1976) was one of the first to be interested in aircraft maintenance schedules and to formalize a maintenance model. The literature is divided into four main categories that can be addressed sequentially (Klabjan, 2005). The Schedule (or Flight) Design Problem (SDP), (Lohatepanont and Barnhart, 2004), is a marketing-led planning process used by airlines to determine flight offerings based on revenue analysis. Subsequently, the Fleet Assignment Problem (FAP), (Hane et al., 1995; Sherali et al., 2006), allocates the aircraft types in the airline's fleet, taking into account the capacity of each aircraft type, demands, operational costs and potential revenues. This leads us to the Crew Assignment Problem (CAP), (Hoffman and Padberg, 1993), which establishes the crew composition to cover all aircraft routes, considering the skills, availability and labor work rules of crew members. Afterwards, the Aircraft Maintenance Routing Problem (AMRP), (Gopalan and Talluri, 1998), focuses on constructing a feasible sequence of flights for an aircraft while adhering to maintenance requirements without assigning a specific aircraft to the newly created flight plan. The Line Maintenance Scheduling Problem (LMSP), (Van den Bergh, 2013), can be seen as another problem, focusing on the short-term maintenance scheduling. Some recent studies work on an Airline Integrated Robust Planning considering all the previous scheduling problems as in Xu et al. (2021). The authors use a Column Generation (CG) decomposition and a local neighbors search to accelerate the process. Without local search, CG processing time can be up to 24 h.

These problems are related to different types of planning and different time horizons. The SDP designs strategic choices of the airline a few years ahead. The FAP is typically planned several months to several years in advance, while the CAP and the AMRP involve tactical decisions made several weeks to a few months before operations. Fig. 3 summarizes the different scheduling problems faced by the airlines. It classifies the related works present in the literature and it locates the contribution proposed in this paper.

Typically, the AMRP is approached from a tactical planning perspective to design routes for an aircraft and to regularly visit maintenance bases to fulfill its maintenance requirements. Feo and Bard (1989) present a large-scale Mixed Integer Programming (MIP) model to address aircraft routing problems, considering maintenance constraints and crew-related factors. Clarke et al. (1996) solve the Aircraft Rotation Problem with feasible maintenance routes using a Lagrangian Relaxation. The majority of research on AMRP focuses on mid-term planning particularly considering A and B-checks. Moving to a more operational context, in Sriram and Haghani (2003), authors suggest a heuristic-based model to schedule the maintenance of a mid-sized fleet of eight aircraft, minimizing maintenance costs and re-assignment costs of aircraft to flights in a dynamic routing context. The heuristic is based on a hybridization of random search and depth-first search. They demonstrate that heuristic models can quickly generate solutions for complex problems where other models might require exponential time.

In a more operational way, other papers suggest the repetition of a daily aircraft route plan to create a short-term operational schedule. This problem consists in determining a valid daily aircraft routing considering the individual characteristics of one aircraft. Sarac et al. (2006) named this problem the Operational Aircraft Maintenance Routing Problem (OAMRP). Haouari et al. (2013) propose a reformulation-linearization technique to propose a daily repetitive schedule with optimized rotations considering maintenance costs and penalties for short connections. In this study, hangar capacities are considered and A-check have a fixed duration. This daily schedule requires a return to a base station at the end of the day. It can be repeated over several weeks for a tactical vision at mid-term. However, it does not take into account the heavy checks. Al-Thani et al. (2016) address the OAMRP using a MIP model with a polynomial number of variables and constraints. Their very large-scale neighborhood search algorithm was able to return a solution close to the optimal (4%) in 140 s for a flight network involving 16 aircraft, 226 flights and 26 maintenance stations.

Similarly, the Tail Assignment Problem (TAP), (Grönkvist and Kjerrström, 2005), can be seen as an operational extension of the Aircraft Maintenance Routing Problem. TAP adjusts planned routes to various operational constraints just a few days before operations. It assigns these routes to individual aircraft, identified by their tail number, taking into account the appropriate maintenance for the selected aircraft. Some hybrid approaches use Constraint Programming (CP) in decomposition methods to accelerate their generation process. Gabteni and Grönkvist (2009) use column generation algorithm

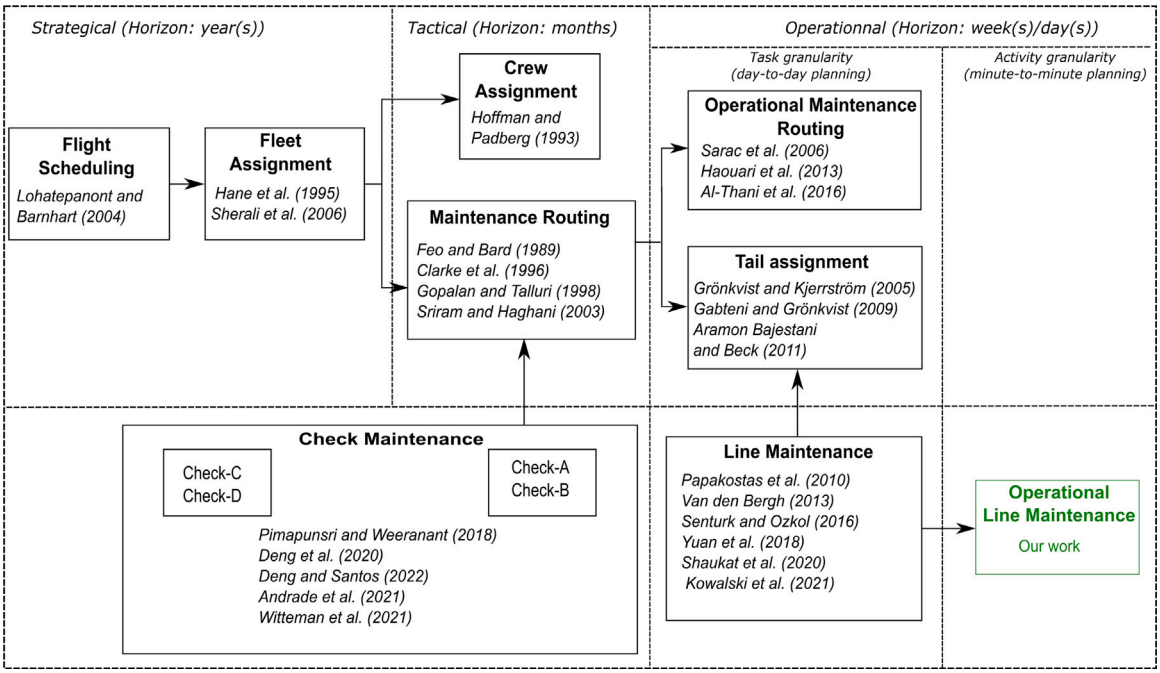

Fig. 3. Airline scheduling problems.
Source: Adapted from Lagos et al. (2020).

to address the TAP, modeling maintenance constraints with CP in the pricing problem. CP enables to quickly produce initial solutions and to check feasibility during a column generation based heuristic branching algorithm. MIP and CP hybridization is also used in Aramon Bajestani and Beck (2011) with a Bender's decomposition to schedule aircraft assignments to missions in a military context.

Furthermore, maintenance scheduling can also be studied in a long term horizon to provide a strategical overview of heavy check scheduling. In this case, flight schedules are simulated or an average aircraft utilization is considered. Deng et al. (2020) propose a dynamic approach based on a Markov Decision Process to address the Aircraft Maintenance Check Scheduling (AMCS) problem. They consider all maintenance checks from A-check to D-check for a heterogeneous fleet of 40 aircraft. Since B-checks are rarely used in practice, they are included in successive A-checks as well as for D-checks which can be incorporated into some heavy C-checks. In addition, the A-check and C-check schedules are dependent. Some A-check tasks can be combined within some C-check, following certain rules, without significantly increasing the duration of the C-check. One goal of the AMCS is to optimize the merging of these checks. Their results on real airline cases show a potential reduction in the number of maintenance checks of about 7% over a period of four years and generate aircraft availability. In their following study (Deng and Santos, 2022), authors consider the uncertainty of aircraft daily utilization and maintenance check elapsed time to address the Stochastic Aircraft Maintenance Check Scheduling (S-AMCS) problem. Their method first plans heavy maintenance using deterministic forecasts and then incorporates line maintenance using stochastic forecasts. Their experiment shows a reduction in the number of A and C-checks and in the number of additional slots. Learning approaches can be used to address the AMCS. Andrade et al. (2021) present a Reinforcement Learning (RL) model with Q-Learning algorithm to optimize long-term maintenance for an aircraft fleet. They consider maintenance capacity such as the available number of hangar slots. They schedule close to due dates to reduce the number of checks and to improve fleet availability. Their RL model is adaptable to some small disturbances in the initial conditions. After 20 h of training, the solution can be obtained in a few seconds.

In Witteman et al. (2021), authors propose a model for task allocation after solving the AMCS. The approach is inspired of a bin backing problem to model maintenance opportunities with limited capacity. A bin (maintenance slot) has a maintenance availability, a workforce of resources divided by skills and respect of work time rules. A priority rule following a practical behavior is given to schedule the most urgent task first. The approach enables to schedule in 15 min, a five-year horizon planning with around 50-500 maintenance checks scheduled for a fleet of 45 aircraft. However, this study does not consider small line tasks and precedence relations inside a bin.

After task allocation, resource assignation is another complex problem involved in operational aircraft maintenance scheduling problem. This problem is closely related to the RCPSP (discussed in Section 4). In Pimapunsri and Weeranant (2018), authors represent the scheduling of heavy checks as an RCPSP incorporating skills requirements and precedence relations between activities. They used a Design Structure Matrix and heuristic method to solve the problem. They came up with a solution enabling a reduction in waiting time of resources between precedence-constrained activities and minimizing global makespan in scheduling heavy checks. Others methods as Genetic Algorithms (GA) can be used to solve an RCPSP. Works by Yuan et al. (2018) and Kowalski et al. (2021) show satisfying results in a dynamic maintenance context for short time windows (several hours or days maximum) with a computational time of around 100 s. In Yuan et al. (2018) GA is used to test different heuristics for resource allocation in either serial or parallel mode. Despite of that, the computational time sharply increases with larger time windows. GA are effective in getting close to the optimal solution of a problem because they search the optimal point not from a single location but from several locations in search tree. However, it is recommended to run several time the problem because GA use stochastic algorithms, and they can generate different result with the same inputs.

Line Scheduling Maintenance Problem is often overlooked in the literature. This is unfortunate because the availability of the aircraft can be improved if individual maintenance tasks are carried out every time an aircraft is on the ground (Senturk and Ozkol, 2016). Papakostas et al. (2010) propose a simple heuristic to schedule short term maintenance

tasks of one aircraft at a time, considering the tasks of the other aircraft as fixed. More recently, Shaukat et al. (2020) summarize the objective of the LMSP as the problem of assigning a start time and a location for each maintenance job to be carried out within a planning horizon in order to minimize total deviation from due dates, given a flight plan or a list maintenance slot, a fleet of aircraft and the respective maintenance tasks to be performed on each aircraft. The LMSP is related as a variant of the RCPSP. They consider machines as maintenance opportunities and that jobs have no precedence constraints. They define a mathematical formalization of LMSP with a MIP model. A sequential heuristic approach, which mimics the practice of schedulers is proposed to respect priority rules of the maintenance jobs. This heuristic algorithm is divided into two steps: first, the job assignment to maintenance opportunities and then timetabling decisions of resource allocation. Their experiments provide an exact solving in an average of 10 min and a 3.5% gap to optimality in about 10 s using the heuristic approach.

Most of the papers in the literature address the maintenance scheduling at task granularity level. Maintenance checks are scheduled as a unit package and the activities inside a task are not precise. In this paper, we add constraints related to the granularity of operation and activity scheduling to address a maintenance plan that is operationally feasible and can be executed by maintenance personnel. This implies a (i) resource allocation to elementary maintenance activity, (ii) precedence and synchronization relations between operations and (iii) capacity constraints to perform simultaneously activity in an aircraft zone.

Another branch of the literature studies recovery problems when disruptions occur compromising the feasibility of the scheduled plans (Eggenberg et al., 2010). These are called Airline Recovery Problems which are not directly in the scope of this paper. Disruptions may be caused by many unexpected phenomenon, such as bad meteorological conditions, unpredicted maintenance repair, propagated delays in maintenance or unavailable resources. These disruptions may have multiple impacts on the original scheduled plans. A flight cancellation or a delay due to maintenance may compromise an aircraft route and flight connection for passengers. Initial maintenance and crew schedules become infeasible and they must be rescheduled on short and mid term with the propagated impacts. Recovery decisions are taken at the Operations Control Center (OCC). Their role is to reroute aircraft, adjust crew members and passenger connections in a defined recovery period. In Xu et al. (2023), authors propose a mathematical model to handle the integrated recovery by minimizing the passenger travel cost and minimize the weighted recovery efforts. SWAP scenarios are considered, when an aircraft cannot ensure a flight and must be replace by another aircraft. The flight schedules of two aircraft are swapped to ensure operability. Recovery problems are seen as reactive approach to re-accommodate sequentially aircraft routing, crew and passengers allocation. One cause of disruptions is when an aircraft require a maintenance at the end of a day, which is called a maintenance misalignment. Maher et al. (2014) present a MIP model for a single day aircraft routing problem to ensure sufficient aircraft routes ending in a maintenance station to perform maintenance requirements during the night. They propose a recoverable robust approach to penalize maintenance misalignment. Other approaches can be more proactive to anticipate disruptions in the schedule. Xu et al. (2019) propose a stochastic model to address the Tail Assignment Problem and increase the robustness of the schedule. The model considers the possible recovery reactions in case of operational perturbations. Another important factor affecting the feasibility of aircraft routing plans is the availability of resources (Eltoukhy et al., 2018). Resources workforce may be underestimated and conflicts can emerged between resources. Wen et al. (2022) proposed an approach to minimize the impact of recovery actions on the original plan. They studied a novel strategy named SMR which consists to allocate maintenance resource to other airports to ensure the maintenance when misalignment occur. These scenarios are expensive because the resources must be sent to the airport or flight in the aircraft but it can avoid AOG and cancellation costs. The majority of these studies use a column and row generation to solve these recovery problems.

As said before, the Airline Recovery or Aircraft Routing problems are not studied in this paper. However, some links can be noticed with OALMSP. The model proposed in this paper can be used to ensure or test the operational maintenance feasibility of the schedule returned for the Aircraft Routing Problem. Planners may identify some potential conflicts and make some adjustments or anticipate requirements. Otherwise, the operational maintenance scheduler could be useful to recover maintenance schedules after a SWAP scenario. Finally, the OALMSP reduces both resource conflicts and underestimated resource workforce which are a major factors in disruptions.

4. Resource Constrained Project Scheduling Problem (RCPSP)

4.1. Basic RCPSP model

The OALMSP described in Section 2 forms part of what is referred to as the Resource Constrained Project Scheduling Problem. The first project scheduling problem was introduced by Kelley and Walker (1959) to minimize the critical path of job scheduling in a resource-constrained environment. In its standard form (Artigues et al., 2008), an RCPSP enables problems of scheduling job-activities to be modeled, taking into account their duration and the requirement of resources with limited availability. The schedule must respect the precedence relations between activities. The objective consists of finding the schedule by assigning a start date to each activity, respecting all precedence constraints between activities and the availability of resources, and minimizing the *makespan*, i.e., the global duration to perform all activities.

Concretely, an RCPSP is composed of a set of n *activities*, $\mathcal{A} = \{a_1, \dots, a_n\}$, of respective *duration* $p_j > 0, \forall\, j \in \{1, \dots, n\}$. By convention, two activities a_0 and a_{n+1} are added to represent respectively the start and end activity of zero duration.

Precedence relations are given by a set E of pairs (a_i, a_j) meaning that activity a_i must be scheduled before activity a_j. This can be represented by an acyclic graph G where each node represents an activity and the edges the precedence relations. A set of m *resources* is given, $\mathcal{R} = \{r_1, \dots, r_m\}$, with their respective *availability* $B_k > 0, \forall\, k \in \{1, \dots, m\}$. If $B_k = 1$, the resource is *unitary* and if $B_k > 1$ the resource is called *cumulative*. Each activity a_j requires an amount b_{jk} of resources k, called *demand*. Starts of activities are represented in a set S where S_i is the start of the activity a_i. S_0 is set to 0 and S_{n+1} is the final point of the schedule. Minimizing the makespan means to minimize S_{n+1}. The set $\mathcal{A}_t = \{a_j \in \mathcal{A} \mid S_i \leq t < S_i + p_i\}$, represents activities processing at the instant t. A *feasible* solution assigns values to S that satisfy all precedence constraints (1) and all resource constraints (2):

$$\forall\, (a_i, a_j) \in E,\ S_j - S_i \geq p_i \tag{1}$$

$$\forall\, t > 0, \forall\, r_k \in \mathcal{R},\ \sum_{a_i \in \mathcal{A}_t} b_{jk} \leq B_k \tag{2}$$

In the literature, resources are qualified as *renewable* and *non-renewable* (Artigues et al., 2008; Neumann and Schwindt, 2003). A renewable resource is used by an activity and then returns to its initial state when the activity ends. Renewable means that the number of units of a resource remains the same at every time of the planning horizon. Non-renewable means that the number of units of a resource decreases after utilization by an activity. In its primary form, RCPSP considers renewable resources, so B_k is constant through the project. Furthermore, all activities are *non-preemptive* that is to say they cannot be interrupted. All the inputs are deterministic and assumed to be non-negative integer.

4.2. Solving RCPSP

RCPSP is one of the hardest combinatorial optimization problem to solve. It belongs to the strongly NP-hard problems demonstrated by Blazewicz et al. (1983). It is very useful for modeling industrial problems but solving such issues can be very complex in practice. For example, the optimal makespan of randomly generated problems extracted from the Project Scheduling Problems library (PSPLIB, 2005) with 60 activities and four resources is still unsolved. Several benchmarks (Artigues et al., 2008) for competition solvers have been created to compare the performances of different solving methods. To cite some of them, 33 of 48 instances with 103 activities named ALV103 were optimally solved in Beldiceanu et al. (1996) using a CP solver CHIP. The optimal proof required two weeks of CPU time for one-third of solved instances. Another famous benchmark is KSD30 which gathers 48 groups of instances of 30 activities. Demeulemeester and Herroelen (1997) proved the optimal solution with a branch and bound algorithm (denoted *DH*) in an average of 14.76 s for 479 of 480 instances and three hours for the last remaining instance. They also measured the average deviation from the optimal solution, using heuristics and a truncated branch and bound procedure, in relation to the CPU time limit. The solution returned in 0.1 s with this method has an average deviation from the optimal less than 2% and it decreased to 0.83% for a time limit of one second, and less than 0.06% for a maximum time of one minute.

On the other hand, some methods exist to find a lower bound of the optimal solution. To cite some of them, *PWW* designs the lower bound obtained by solving to optimality the Linear Programming (LP) relaxation proposed by Pritsker et al. (1969). They proposed one of the first mathematical formulations of RCPSP with a 0-1 Integer Linear Programming (ILP) model. In addition to precedence and resource constraints, a constraint ensures that each activity is started exactly once over the planning horizon. In Christofides et al. (1987), an ILP model suggests a variant of precedence constraints that allows for stronger relaxation to the detriment of a pseudo-polynomial number of precedence constraints. The lower bound obtained by solving optimally the LP relaxation is denoted *CAT*. Another exact approach based on CP, *MCS* (Laborie, 2005), enables the optimal makespan or a lower bound of it to be identified via an iterative search process. Finally, heuristic algorithms such as Tabu Search (Pinson et al., 1994) or Large Neighborhood Search (LNS) (Palpant et al., 2004) return solutions very close to the optimal solution.

The largest RCPSP benchmark to be solved in these standard data sets are instances with 120 activities. Nowadays, the optimal solutions for instances with 60, 90 and 120 activities are still unknown. Indeed, as seen in Section 6.2.2, the largest instance solved in common RCPSP competition benchmark is smaller than the smallest use case for the maintenance of two aircraft in one week composed of 345 activities. Because of the size of industrial instances, the research of an optimal solution or a lower bound estimation is not considered in this article.

4.3. RCPSP extensions and adaptations to our industrial model

Several extensions of the standard RCPSP have been proposed which can be implemented in a real scheduling problem. A survey sums up these different variants (Hartmann and Briskorn, 2010). As per its description detailed in Section 2, the industrial problem studied in this paper is similar to an RCPSP. Indeed, maintenance tasks are composed of a set of activities, requiring certain resources during a given duration. There are different levels of precedence relations: first between particular activities within a task and then between tasks from a same periodic series. The aim of the problem is to find a schedule by assigning a start date to each activity.

However, the basic RCPSP structure needs some adjustments to fully model our industrial problem. One particularity of the aircraft maintenance scheduling problem is that activities can be scheduled only if the plane is on the ground between two flights. This cuts out temporal possibilities. Temporal availability is not only led by resource availability but also by slot opportunities. Then, in this industrial problem, activities are grouped by task. Therefore, all activities of a task must be scheduled on the same slot. Furthermore, a maximal duration for a task can be given to complete all its activities. After that, tasks can be also divided in operations. An operation gathers several activities which must have the same starting date to execute them in parallel. This task structure broken down in groups adds some constraints and imposes to have a block reasoning.

As seen previously, there are a number of resources deployed in aeronautic maintenance. Staff and serialized equipment are considered as renewable and unitary resources. They must respect *temporal constraints* (Bartusch et al., 1988). Each physical person or serialized item of equipment has its own agenda with a quantity available of 1. Then, ingredient resources are cumulative resources. Their stock is known in advance. However, some consumable ingredients are non-renewable. For all ingredients, the initial stock increases after each delivery of the quantity ordered. Non-renewable resources are not considered in the basic form of RCPSP. An extension is proposed in Carlier et al. (2009) to model RCPSP to include resource consumption. Another important aspect are the multiple skills of staff. This increases the complexity of the RCPSP because a resource can be deployed for several types of activities but it can be only assigned to one activity at a time. This particular type of RCPSP is called the Multi-Skill Project Scheduling Problem (MSPSP), introduced in Bellenguez and Néron (2004). This model considers renewable and disjunctive resources, mastering several skills, such as staff members who can be assigned to at most one requirement at a time. It enables personnel planning involving different timetables to be taken into account.

The last extension needed for this aircraft maintenance scheduling problem is the multi-objective perspective. In its basic form, the objective of an RCPSP is to minimize the makespan, but in our problem, several strictly ordered objectives are considered. In Tian et al. (2022), authors proposed a multi-objective optimization framework based on the Evolution Strategy which enables a comparison of the different combinations and schedule builders. Ghamginzadeh et al. (2021) also modeled an MSPSP by minimizing both the makespan and the total cost of labor allocation with uncertainty in estimation of activities' duration. In our problem, because of their business priority, the lexicographical order is a way to manage the multi-objective aspect taken on by the company.

5. Constraint programming model

In this section, we explain our choice of using CP to address the OALMSP and we present our model to schedule the specific RCPSP applied to aircraft line maintenance. Constraints used will be divided into three main categories: maintenance constraints, precedence constraints and cumulative constraints. First of all, a brief reminder of Constraint Programming is provided.

5.1. Background

Constraint Programming is a general purpose technique for solving complex and high-dimensional problems, such as planning or scheduling. This technique uses a high level language that enables a problem to be modeled easily.

The problem is described as a set of discrete variables, each defined on a finite domain, a set of constraints involving these variables, whose goal is to find a solution satisfying all the constraints (Brailsford et al., 1999). More precisely, the problem can be modeled by the triplet (X, D, C), where X is the set of problem's variables, D the set of the respective domains of each variable and C is the set of constraints. This constitutes a *Constraint Satisfaction Problem* (CSP). The CSP can handle several types of variables such as booleans, integers, sets or subsets.

A *constraint* is a relation between several variables that limits the set of values that these variables can take simultaneously, i.e., formally, each constraint is a subset of the Cartesian product of the domains of the variables it deals with.

A *solution* is an instantiation of all the variables (in X) to a value in their respective domain (among D) that satisfy all the constraints (in C). On the other hand, the problem is said to be *unsatisfiable* if no values in their respective domains satisfy all constraints (Rossi et al., 2008).

A CSP can be solved to find a solution, to list all the solution, or to prove the absence of a solution. A CSP can be converted into a *Constrained Optimization Problem* (COP) by adding an expression (*objective*) that is expected to be maximized or minimized (Barták, 1999). This enables the return of a better or even the best solution.

The quality of the model and an accurate search strategy (Kanet et al., 2004) are the keys to good performances both in terms of the quality of the solutions found and the time needed to obtain them. These methods can be adapted according to the complexity of the problem and are useful in practical cases.

5.2. Motivations of using constraint programming

Constraint Programming can be used to answer business requirements (Section 2.3) and it has been selected to tackle the OALMSP for the following reasons:

(i) CP offers great modeling flexibility but requires a good formulation in the declaration of the logic model and in the elaboration of the search strategy (Kanet et al., 2004). It is easy to integrate new constraints or disable some constraints (Rossi et al., 2006). Similarly, the lexicographical objective function offers the possibility to manage sub-objectives and their priorities.
(ii) CP paradigm can guarantee the determinism of the method and the replicability of the solution with the same inputs (imposed by business).
(iii) The decision process can be expressed thanks to a constructive search strategy. A heuristic close to a human behavior makes the solution understandable by a human.
(iv) Filtering algorithms of CP are adapted for precedence and resource constraints in scheduling problems (Artigues et al., 2008; Liess and Michelon, 2008).
(v) The paradigm of CP ensures the enumerations of all solutions of a CSP (Rossi et al., 2006). An adapted search strategy enables to build quickly an acceptable first solution, even for large problems.

Based on the considerations mentioned earlier, we exclude stochastic and learning methods such as Genetic Algorithms (GA) or Reinforcement Learning (RL). Indeed, these methods are not deterministic and can yield different results for the same initial problem which is not compatible with our use case. In addition, multiple solutions are recommended with GA to obtain an average solution. Kowalski et al. (2021). Then, in case of modifications in the model (adding constraints or changing objective function), the steps of selection, crossover and mutations steps in the GA process must be adapted. On the other hand, RL can be used to model RCPSP (Sung et al., 2020; Zhao et al., 2022). Their experiments are carried out on instances of maximum 90 and 120 activities and some improvements may be done to extract underlying decision-making rules or managerial insights from the policy. Even if RL can be robust to small disturbances in the initial conditions (Andrade et al., 2021), one difficulty identified with RL is to adapt to new situations. For instance, new tasks are coming frequently from airworthiness or the airline can add a new aircraft type in its fleet. Historical data are not present in the training set and it is difficult to learn specific rules. Therefore, the model needs to be retrained and it can take a long time, for instance 24 h in Andrade et al. (2021).

Furthermore, compared to heuristic solvers, CP solvers can explore several solutions and not only one constructive solution. Thus, the optimal solution is theoretically guaranteed in a COP. However, the solving time can be inconsiderable for large instances. A heuristic search strategy enables to build quickly a first solution for complex problems and then to explore others solutions with local search algorithms (Rossi et al., 2006).

A lot of papers in the literature use MIP to address Aircraft Scheduling Problems. However, RCPSP uses cumulative and disjunctive constraints. The MIP model size increases monotonically with the cardinality of the considered tasks whilst in CP one constraint is sufficient to model a set of n tasks in disjunction. Note that CP offers the possibility to model non linear constraints as AllDifferent (Lauriere, 1978). In addition, the size of the instances explode in the OALMSP with the consideration of activity granularity. As a comparison, the instance for two-week scheduling for four aircraft represents 77 maintenance jobs in Shaukat et al. (2020) while in our study, for the same horizon, we consider 131 maintenance tasks equivalent to an RCPSP of 1250 activities. Then, for practical reason, the experiment must be carried out on a simple laptop with memory limitations. As will be shown in experiment 6.1, the number of variables and constraints between CP and MIP models can be multiplied by a factor of 1000. A practical memory problem may arise as the model size increases by several gigabytes. Given the scale and the complexity of our problem, CP seems to be an appropriate method to control the number of variables and constraints.

5.3. CP model of OALMSP

In this section, the CP model of the OALMSP introduced in Section 2 is presented. First, the input data and the variables will be described. After that, the constraints description will be presented in three parts: first, the constraints related to maintenance context, then the precedence constraints linked to the execution order of tasks and finally the resource constraints checking resource capacity to execute tasks. Some extensions due to the specificity of this problem will be explained at the end. Let us explain first the constraint notations.

5.3.1. Constraint notations

As seen in the previous section, RCPSP is managed by two types of constraints: Cumulative and Disjunctive constraints. The Constraint Global Catalogue (GCC) (Beldiceanu et al., 2005) is taken as reference.

A Cumulative constraint (Aggoun and Beldiceanu, 1993) is defined as Cumulative(TASKS, LIMIT) where TASKS is a set of n task variables and LIMIT is a non-negative integer representing the maximal capacity. A task is defined by its *origin* integer variable o, its *duration* integer variable d such as $d > 0$, its *end* integer variable such as $e = o + d$ and its *height* variable such as $h > 0$, denotes resource consumption. The Cumulative constraint ensures that at each point in time, the cumulated height of tasks that overlap that point, does not exceed a given limit. More precisely, we will note further Cumulative constraints as this:

$$\forall\, t \in \text{TASKS},$$

$$\text{Cumulative}\left(\left\langle \begin{matrix} o_1 & d_1 & e_1 & h_1 \\ & \vdots & & \\ o_n & d_n & e_n & h_n \end{matrix} \right\rangle, \text{LIMIT}\right)$$

A Disjunctive constraint (Carlier, 1982) defined for a set of n task variables as Disjunctive(TASKS) with o_i and $d_i \geq 0$, $\forall\, i \in [1, \ldots, n]$, the respective *origin* and *duration* variables of tasks. The Disjunctive constraint ensures that no tasks in TASKS overlap. The Disjunctive constraint is a special case of the Cumulative constraint.

5.3.2. Input data and constants

1. Let s and e represent respectively the *start* and the *end* of the planning horizon, such as $0 \leq s < e$.
2. Let $\mathcal{AC}$ represent the set of aircraft to maintain.
3. A *zone* z indicates the geographical location in an aircraft. Each aircraft has a set of zones $\mathcal{Z}_{ac}$ which have a maximal capacity Cap_z (number of people could work at the same time).
4. Let $Slots$ represent the set of available slots for maintenance (deduced from flights planning). Each slot has a localization L_{slot}, a maintenance level $Level_{slot}$, a start date s_{slot}, a duration p_{slot} and an end date such as $e_{slot} = s_{slot} + p_{slot}$. Let $Slots_{ac} \subset Slots$ represent the set of aircraft slots $ac \in \mathcal{AC}$. Slots can have transfer margins which are unavailable time for maintenance in slot. For example gate transfer ($marginGate$) or time to go to the hangar ($marginHangar$). For MID-term slots, $margin_{slot} = marginGate + marginHangar$, and other slots $margin_{slot} = marginGate$.
5. Let $Skill$ represent the set of all skills (including references for equipment and spares).
6. Let $\mathcal{R}$ represent the set of temporal resources. $\forall\ r \in \mathcal{R}$, a resource is defined by a set of skills $Skill_r$, a localization L_r and a set of unavailable times, $\mathcal{U}_r = \left\{ \left[s_{u_{1_r}}, e_{u_{1_r}} \right], \ldots, \left[s_{u_{z_r}}, e_{u_{z_r}} \right] \right\}$. Let us note $\mathcal{R}^{skill} = \left\{ r \in \mathcal{R} \mid skill \in Skill_r \right\}$.
7. Let $\mathcal{I}$ represent the set of ingredients. $\forall\ ing \in \mathcal{I}$, an ingredient (consumable or not) is defined by an initial stock $stock_{ing} \geq 0$, a skill $skill_{ing}$, a localization L_{ing} and a set of tuples of delivery date and quantity ordered $\mathcal{D}_{ing} = \left\{ \left(d_1, b_{d_1} \right), \ldots, \left(d_y, b_{d_y} \right) \right\}$.
8. Let $\mathcal{T} = \left\{ t_1, \ldots, t_n \right\}$ represent the set of n tasks to schedule. Each task corresponds to one unique aircraft. An interval of scheduling is given for each task between an authorized release date $releaseDate_i$ and deadline $deadline_i$ ($s \leq releaseDate_i \leq deadline_i \leq e$). Then, a task is determined by a maintenance level $Level_i$: SHORT-term for tasks that can be carried out at the gate and MID-term for A-check and B-check requiring a hangar. Moreover, a task can be labeled as *URGENT*. In this case the task must be scheduled as soon as possible. A minimal duration p_{min_i} to perform a task can be given. For instance when all activities from a task are executed in parallel, p_{min_i} corresponds to the largest duration of activities. In the same way, a maximal duration p_{max_i} can be provided. For example the sum of the duration of all activities if they are executed sequentially.
9. We call *Series* a set of tasks with the same characteristics on the same aircraft which must be scheduled at a maximal interval (called *period*) respecting their position. Let $\mathcal{S} = \left\{ S_1, \ldots, S_c \right\}$ represent the set of series. $\bigcup_{k \leq c} S_k$ is a partition of integer from 1 to n. $S_k\langle r \rangle = i$ defines the index i from task at position $r \in \left[1, \ldots, |S_k| \right]$ in series S_k.
10. Let D_k stand for the maximal duration (called *Period*) between two consecutive tasks within the same series.
11. Let $G = \langle V, E \rangle$ indicate the graph of disjunction between tasks. A node $V_i \in V$ represents a task t_i and an edge $(t_i, t_j) \in E$ means that two tasks cannot be executed at the same time.
12. Let $\mathcal{A} = \left\{ a_1, \ldots, a_m \right\}$ stand for the set all m activities. For each activity a_j, a task is linked $Task(j)$, a skill is required $skill_{a_j}$ and a duration $p_j \geq 0$, a demand $b_j > 0$, and a set of zones $\mathcal{Z}_j \subseteq \mathcal{Z}$ are defined. Each task $t_i \in \mathcal{T}$ have a set of activities $\mathcal{A}^i \subseteq \mathcal{A}$ such as $\forall\ i, j \in [1, n],\ i \neq j,\ \mathcal{A}^i \bigcap \mathcal{A}^j = \emptyset$ and $\bigcup_{i \in [1,n]} \mathcal{A}^i = \mathcal{A}$. Activities $\mathcal{A}^i$ must be scheduled during the execution of t_i. For activities requiring temporal resources $\mathcal{A}_r \subseteq \mathcal{A}$ (staff or serialized equipment) $b_j = 1$ and for activities requiring ingredients $\mathcal{A}_{ing} \subseteq \mathcal{A}$, $b_j \geq 1$. Note that $\mathcal{A}_r \bigcup \mathcal{A}_{ing} = \mathcal{A}$ and $\mathcal{A}_r \bigcap \mathcal{A}_{ing} = \emptyset$. Let us note $\mathcal{A}^{skill} = \left\{ a_j \in \mathcal{A} \mid skill_{a_j} = skill \right\}$.
13. Let Op stand for the set of all operations. An operation op is a set of activities. Op^i refers to the operation of task t_i. Note that $\bigcup_{i \in [1,n]} Op^i = \mathcal{A}$. All activities in the same operation start at the same time. Operations from a task can be ordered and activities must be executed respecting this order. Let us note $Op^{skill} = \left\{ op \in Op \mid a_j \in op,\ a_j \in \mathcal{A}^{skill} \right\}$, the subset in each operation of activities requiring skill $skill$.

5.3.3. Variables

As a reminder, a variable is an unknown taking its value in a set of authorized values called domain. The variables used in this CP model will be defined in this section:

1. A task t_i is defined by a start date s_i, an end date, e_i and a duration p_i such that $s_i + p_i = e_i$. Note that $releaseDate_i \leq s_i$ and $e_i \leq deadline_i$:

$$\forall\ t_i \in \mathcal{T},$$

$$D(s_i) = \left[releaseDate_i,\ deadline_i - p_{min_i} \right] \tag{3}$$

$$D(e_i) = \left[releaseDate_i + p_{min_i},\ deadline_i \right] \tag{4}$$

$$D(p_i) = \left[p_{min_i},\ p_{max_i} \right] \tag{5}$$

2. An activity a_j is defined by a start date s_j, an end date e_j and a duration p_j, such as $s_j + p_j = e_j$.

$$\forall\ t_i \in \mathcal{T}, \forall\ a_j \in \mathcal{A}^i$$

$$D(s_j) = \left[releaseDate_i,\ deadline_i - p_{min_i} \right] \tag{6}$$

$$D(e_j) = \left[releaseDate_i + p_{min_i},\ deadline_i \right] \tag{7}$$

3. Let us note $slot_i$ the assigned slot to task t_i and its domain $D(slot_i)$:

$$\begin{aligned} \forall\ t_i \in \mathcal{T},\ D(slot_i) = \Big\{ & slot_k \mid slot_k \in Slots_{ac}, \\ & releaseDate_i \leq s_{slot_k} + margin_{slot_k}, \\ & e_{slot_k} + margin_{slot_k} \leq deadline_i - p_{min_i}, \\ & p_{min_i} \leq p_{slot} - 2 * margin_{slot_k}, \\ & Level_i \in Level_{slot_k} \Big\} \end{aligned} \tag{8}$$

Note that we can define certain inclusions for maintenance levels between slot and task. For instance, a task with SHORT-term level can be executed on a MID-term slot, so $Level_{SHORT} \in Level_{MID}$ (but $Level_{MID} \notin Level_{SHORT}$).

4. Let r_{a_j} the resource assigned to activity a_j:

$$\forall\ t_i \in \mathcal{T},\ \forall\ a_j \in \mathcal{A}^i$$

$$r_{a_j} = \begin{cases} \left\{ r_h \mid r_h \in \mathcal{R}^{a_j} \right\} \text{ if } a_j \in \mathcal{A}_r \\ \left\{ r_h \mid r_h \in \mathcal{I}^{a_j} \right\} \text{ otherwise} \end{cases} \tag{9}$$

with

$$\begin{aligned} & \mathcal{R}^{a_j} = \left\{ r \mid r \in \mathcal{R}^{skill_{a_j}}, L_r = L_{slot_i} \right\} \\ \text{and } & \mathcal{I}^{a_j} = \left\{ ing \mid ing \in \mathcal{I}, skill_{ing} = skill_{a_j}, L_{ing} = L_{slot_i} \right\} \end{aligned} \tag{10}$$

5.3.4. Maintenance constraints

First of all, certain constraints are due to the integration of maintenance in an operational flight planning schedule. Airlines may not require the scheduling of MID-term maintenance actions during the week or summer pick activities. This is presented in Gopalan and Talluri (1998) as *maintenance constraints*. Here, the maintenance constraints used in our problem are presented.

A task must be assigned to a possible aircraft maintenance slot:

$$\forall\ t_i \in \mathcal{T}, \forall\ slot_i \in Slots_{ac},$$

$$\left[s_i, e_i \right] \subseteq \left[s_{slot_i} + margin_{slot_i},\ e_{slot_i} - margin_{slot_i} \right] \tag{11}$$

All activities of a task must be performed during the task execution :

$$\forall\, t_i \in \mathcal{T}, \quad s_i = \min_{a_j \in \mathcal{A}^i}(s_j) \tag{12}$$

$$\max_{a_j \in \mathcal{A}^i}(e_j) \leq e_i \tag{13}$$

Let us note that activities can finish before the end of the task. For example, some tasks need drying or cooling time after the end of activities. This represents additional time to accomplish the task that does not require resources.

Other constraints are induced by the division of the task into operations. Typically, all activities, belonging to the same operation, start at the same time:

$$\forall\, t_i \in \mathcal{T},\ \forall\, op\ \in Op^i,\ \forall\, a_j,\ a_{j'} \in op,\ s_j = s_{j'} \tag{14}$$

From a resources perspective, the resource assigned to an activity must be located in the same place as the slot it has been allocated:

$$\forall\, r \in \mathcal{R}^{a_j},\ L_{slot_i} = L_r \tag{15}$$

$$\forall\, ing \in \mathcal{I}^{a_j},\ L_{slot_i} = L_{ing} \tag{16}$$

As induced by (14), resources assigned to activities from a same operation must all be different:

$$\forall\, t_i \in \mathcal{T}, \forall\, op \in Op^i,$$
$$\textsc{AllDifferent}\left(\{r_{a_j} \mid a_j \in op\}\right) \tag{17}$$

5.3.5. Precedence constraints

RCPSP models are designed to take into account precedence constraints. In our problem, there are three levels of precedence constraints.

The first is due to the periodicity interval imposed by regulations between two consecutive tasks in a same series. The time interval must be less than the specified period D_k, then scheduling order must respect the position of tasks in the series and their assigned slots must be different. Let us consider a series $S_k \in S$ and $|S_k| = \mathcal{K}$.

The separation time d_r between two consecutive tasks of a same series is defined as:

$$\forall\, r \in [1, \ldots, \mathcal{K}-1], \quad d_r = s_{slot_{S_k\langle r+1\rangle}} - e_{slot_{S_k\langle r\rangle}} \tag{18}$$

The constraint (19) ensures that the maximal separation time cannot exceed the period D_k of the series S_k. The constraint (20) expresses that a task in a series must be schedule on a further slot than its previous task in the series.

$$\max_{\forall\, r \in \mathcal{K}-1}(d_r) \leq D_k \tag{19}$$

$$\forall\, r \in [1, \ldots, \mathcal{K}-1],\ d_r > 0 \tag{20}$$

Note that, only calendar periodicity is considered in constraint (19). It exists some coefficient to convert flight hours and flight cycles intervals into calendar unit. However, knowing the flight schedule, it would be easy to add the same type of constraint to verify accurately the periodicity in FH and FC.

Another type of precedence constraint is the disjunction between tasks when it is possible to identify in advance conflicts between tasks from multiples criteria.

All couple of tasks which cannot be executed at the same time can be stored in $G = \langle V, E\rangle$, the graph of tasks in disjunction.

$$\forall\, t_i \in V, t_j \in V \mid t_i \neq t_j \text{ and } (t_i,\ t_j) \in E,$$
$$\textsc{Disjunctive}\left(\{t_i,\ t_j\}\right) \tag{21}$$

Finally, inside a task, when the execution of an operation requires a precise order, activities scheduling must respect the order given. We note $op_A \rightarrow op_B$ the relation "operation A precedes operation B".

$$\forall\, t_i \in \mathcal{T} \mid (op_A, op_B) \in Op^i,\ \forall\, a_j \in op_A,\ \forall\, a_{j'} \in op_B,$$
$$op_A \rightarrow op_B \Longrightarrow e_j \leq s_{j'} \tag{22}$$

5.3.6. Cumulative constraints

The last essential category of constraints in an RCPSP is Cumulative constraints (Aggoun and Beldiceanu, 1993). The following constraints enable the control of workflow and checking of resource capacity.

Firstly, human capacity in an aircraft zone and consumption of resource stocks need to be checked by Cumulative constraints.

There are physical constraints in the problem due to space in the aircraft. Indeed, each part of the plane (called a *zone*) has a human capacity. Therefore, the number of activities requiring resources working on a same zone is limited to the zone capacity:

$$\forall\, z \in \mathcal{Z},\ \forall\, a_j \in \mathcal{A} \mid z \in \mathcal{Z}_j,$$
$$\textsc{Cumulative}\left(\left\langle \begin{matrix} s_1 & p_1 & e_1 & 1 \\ & \vdots & & \\ s_q & p_q & e_q & 1 \end{matrix} \right\rangle,\ Cap_z\right) \tag{23}$$

with $q = \left|\{a_j \in \mathcal{A} \mid z \in \mathcal{Z}_j\}\right|$.

A Cumulative constraint thus ensures that activities using a reusable ingredient may not exceed the stock available in all localization:

$$\forall\, ing \in \mathcal{I},\ \forall\, a_j \in \mathcal{A}^{skill_{ing}},$$
$$\textsc{Cumulative}\left(\left\langle \begin{matrix} s_1 & p_1 & e_1 & b_1 \\ & \vdots & & \\ s_q & p_q & e_q & b_q \end{matrix} \right\rangle, stock_{ing}\right) \tag{24}$$

where $q = \left|\mathcal{A}^{skill_{ing}}\right|$.

Note that the cumulative profile ($stock_{ing}$) is increased by the quantity ordered after each delivery. For this, we calculate the maximal quantity of the ingredient during the simulation and we use fictive tasks to model the unavailability before a delivery. The consumption of a non-renewable ingredient is represented by a fictive task until the end of the simulation.

Moreover, for temporal resources, a constraint by skill checks whether the cumulative charge of operations is compatible with the cumulative capacity of available resources:

$$\forall\, skill \in Skill,\ \forall\, r \in \mathcal{R}^{skill},\ \forall\, op \in Op^{skill},$$
$$\textsc{Cumulative}\left(\left\langle \begin{matrix} s_{u_{1_{r_1}}} & \left(e_{u_{1_{r_1}}} - s_{u_{1_{r_1}}}\right) & e_{u_{1_{r_1}}} & 1, \\ & \vdots & & \\ s_{u_{z_{r_k}}} & \left(e_{u_{z_{r_k}}} - s_{u_{z_{r_k}}}\right) & e_{u_{z_{r_k}}} & 1, \\ s_{min_{op_1}} & p_{min_{op_1}} & e_{min_{op_1}} & b_{skill_{op_1}}, \\ & \vdots & & \\ s_{min_{op_q}} & p_{min_{op_q}} & e_{min_{op_q}} & b_{skillop_q} \end{matrix} \right\rangle,\ k\right) \tag{25}$$

where $k = \left|\mathcal{R}^{skill}\right|$, $q = \left|Op^{skill}\right|$ and $s_{min_{op_1}}$, $p_{min_{op_1}}$, $e_{min_{op_1}}$ represent respectively the start, duration and end of the smallest activity in operation op_1, $b_{skill_{op_1}}$ is the number of activities requiring $skill$ in operation op_1. The cumulative profile of available resources of skill $skill$ is represented by integrating the unavailability of each resource.

5.3.7. DiffN constraint

This section applies only for temporal resources. It is evident that these resources may be assigned to, at most, one activity at the same time. As the problem is an MSPSP for staff resources, a Cumulative constraint by skill is not sufficient, so DiffN constraint (Beldiceanu and Contejean, 1994) has been chosen to model resource allocation activities. In this problem, tasks are represented as rectangles with a width representing a duration (an absence or an activity) and a height of 1 representing an individual (physical resource). Tuples are created to represent the binding between the skill of an activity and the skills of a

resource and the localization of resources with localization of slot using a table constraint. Formally, the DiffN constraint is modeled as:

$$\forall\, r_h \in \mathcal{R},\ \forall\, t_i \in \mathcal{T},\ \forall\, a_j \in \mathcal{A}^i,$$

$$\text{DiffN}\left(\left\langle \begin{array}{llllll} < s_{u_{1_{r_1}}} & \left(e_{u_{1_{r_1}}} - s_{u_{1_{r_1}}}\right) & e_{u_{1_{r_1}}}, & 1 & 1 & 2 >, \\ & \vdots & & & & \\ < s_{u_{z_{r_k}}} & \left(e_{u_{z_{r_k}}} - s_{u_{z_{r_k}}}\right) & e_{u_{z_{r_k}}}, & k & 1 & (k+1) >, \\ < s_1 & p_1 & e_1, & h & 1 & (h+1) >, \\ & \vdots & & & & \\ < s_m & p_m & e_m, & h & 1 & (h+1) > \end{array} \right\rangle\right) \tag{26}$$

with $k = |\mathcal{R}|$, $h \in [1, \ldots, k]$ the index of resource r_h in $\mathcal{R}$ and $m = |\mathcal{A}|$.

5.4. Extensions

In order to adapt effectively to real industrial applications, some extensions to this model can be proposed. The first is to insert the possibility to return a "partial" solution even if all tasks are not scheduled. Indeed, because of the nature of data, configuration of simulations or state of resources, the problem can be over-constrained. In this case, it is requested to return the best partial solution with the more tasks scheduled. To ensure the existence of a solution in all cases, a "universal" solution with infinite resource capacity which absorbs all constraints is added. Tasks not scheduled will be assigned to an *absorbent* slot $Slots_{abs}$.

On the other hand, another extension which could be helpful for planners is the capacity to add certain additional *(fictive)* resources. This enables a relaxation of resource constraints. The user manages the number, the skills and the availability of additional resources. This is helpful for planners to simulate how many exceeded resources they need to schedule all tasks. For resources of type ingredient, it is simply an additional stock. Let us call $\mathcal{R}^{add}$ the set of additional temporal resources and $\mathcal{I}^{add}$ the set of additional ingredients. Resource capacity is still controlled by DiffN constraint (26).

Naturally, the objective will be to schedule tasks with the most resources initially given and use additional resources only as a last resort. The man hour cost of reference (in average for all skills) is about \$75 per hour (Cho, 2021) and \$100 per hour for additional resources (depending on the rates of the individual subcontractor). To penalize additional resources utilization, a coefficient of 1.33 is applied for the use on an additional resource.

Furthermore, the model allows for the extension of a partial instantiation, and thus to complete an existing partially filled planning. All variables of start dates, duration and assigned slot of initial scheduled tasks have been already instanced. The same has been done for all activities related to scheduled tasks. The resources used to carry them out are already assigned. Note that absorbent slots are not allowed for tasks initially scheduled. Consequently, scheduled tasks keep exactly the same schedule in the returned solution.

Finally, some unexpected *findings* may be discovered during the execution of a task and involve some additional urgent works. A task can take longer than expected and in the worst case the aircraft can be frozen if the safety conditions are not adhered to. This situation can generate disruptions and lead to economic impacts (delays, cancellations, emergency logistic orders, etc.). To manage these findings, a notion of risk may be considered in the model. For example, a finding rate with its severity can be associated with the task based on empirical data. The risk could be interpreted as a varying duration to perform additional works if the unexpected event occurs. Depending on the severity and the occurrence rate of the disturbance, the risk should be anticipated as having a proactive schedule. For this, a duration margin depending of risky tasks in a slot could be defined. This margin would provide time to react if the disturbances occur in the slot. The margin time could be treated in the model as a constraint or as an objective observing the cumulative risk of tasks in a slot. The aim would be to avoid the scheduling tasks in critical areas where global risk is high.

5.5. Objectives

The common objectives in aircraft maintenance scheduling are minimization of maintenance costs (Moudani and Mora-Camino, 2000; Sriram and Haghani, 2003) and the maximization of remaining useful life (Basdere and Bilge, 2014; Boere, 1977). Clearly, objectives are driven by business priorities that express a multi-objective problem to schedule the most tasks, maximizing their potential and minimizing resource utilization to achieve it. To follow this functional reasoning, a lexicography objective has been implemented to order different sub-objectives. Let us recall that the lexicographical order enables to custom the objective function to each airline preference.

The first objective (o_1) in lexicographical order is to schedule the most of MID-term maintenance level tasks:

$$\begin{aligned} &\min \sum_{t_i \in \mathcal{T}_C} cost_{slot_i} \\ &\text{with} \quad \mathcal{T}_C = \left\{t_i \mid t_i \in \mathcal{T},\ Level_i = Level_{MID}\right\} \\ &\text{and} \quad \forall\, t_i \in \mathcal{T},\ cost_{slot_i} = \begin{cases} 1 \text{ if } slot_i \in Slots_{abs} \\ 0 \text{ otherwise} \end{cases} \end{aligned} \tag{27}$$

Then, the second objective (o_2) is to schedule the most of SHORT-term maintenance level tasks:

$$\begin{aligned} &\min \sum_{t_i \in \mathcal{T}_L} cost_{slot_i} \\ &\text{with} \quad \mathcal{T}_L = \left\{t_i \mid t_i \in \mathcal{T},\ Level_i = Level_{SHORT}\right\} \end{aligned} \tag{28}$$

After that, the third objective (o_3) is to prioritize the use of given resources in the initial stock, thus minimizing the use of additional resources:

$$\begin{aligned} &\min \sum_{t_i \in \mathcal{T}} \sum_{a_j \in \mathcal{A}^i} cost_{r_{a_j}} \\ &\text{with} \quad cost_{r_{a_j}} = \begin{cases} \lfloor 1.33 * 100 \rfloor \text{ if } r_{a_j} \in \mathcal{R}^{add} \bigcup \mathcal{I}^{add} \\ 0 \text{ otherwise} \end{cases} \end{aligned} \tag{29}$$

where r_{a_j} is the resource assigned to activity a_j(9).

Because the exact unitary resource cost is not known, we considered an arbitrary price of \$1 for the use of an additional resource and \$0 if the resource is in the input data set. The value is multiplied by the penalty coefficient of additional resource of 1.33. Then we multiply by 100 and we keep the integer part in order to obtain an integer value.

Let us remark that this lexicography order is likely to favor a solution with a real resource on a farther slot than a solution with an additional resource on a closer slot from the task deadline.

Another objective (o_4) enables deviation time between task start date and its target date (birth date for urgent task or deadline for others) to be minimized. In fact, it makes it possible to identify the closest slot to the target date:

$$\begin{aligned} &\min \sum_{t_i \in \mathcal{T}} \left| s_i - targetDate_i \right| \\ &\text{with} \quad targetDate_i = \begin{cases} releaseDate_i \text{ if } t_i \in URGENT \\ deadline_i \text{ otherwise} \end{cases} \end{aligned} \tag{30}$$

Finally, if not all task durations are bounded, a last optional objective (o_5) is declared. It aims to minimize the deviation between scheduled task duration and minimal task duration, p_{min_i}.

$$\min \sum_{t_i \in \mathcal{T}} \left| p_i - p_{min_i} \right| \tag{31}$$

Note that it would be possible to add additional objectives in lexicographical order. For instance, if a risk is defined, an objective to minimize the global risk in the schedule could be considered. On the other hand, an objective to minimize the number of slots used could be requested for an airline. For this, task distribution to other slots should be observed (through a NValue constraint (Pachet and Roy, 1999)) and the number of tasks in a slot would be maximized.

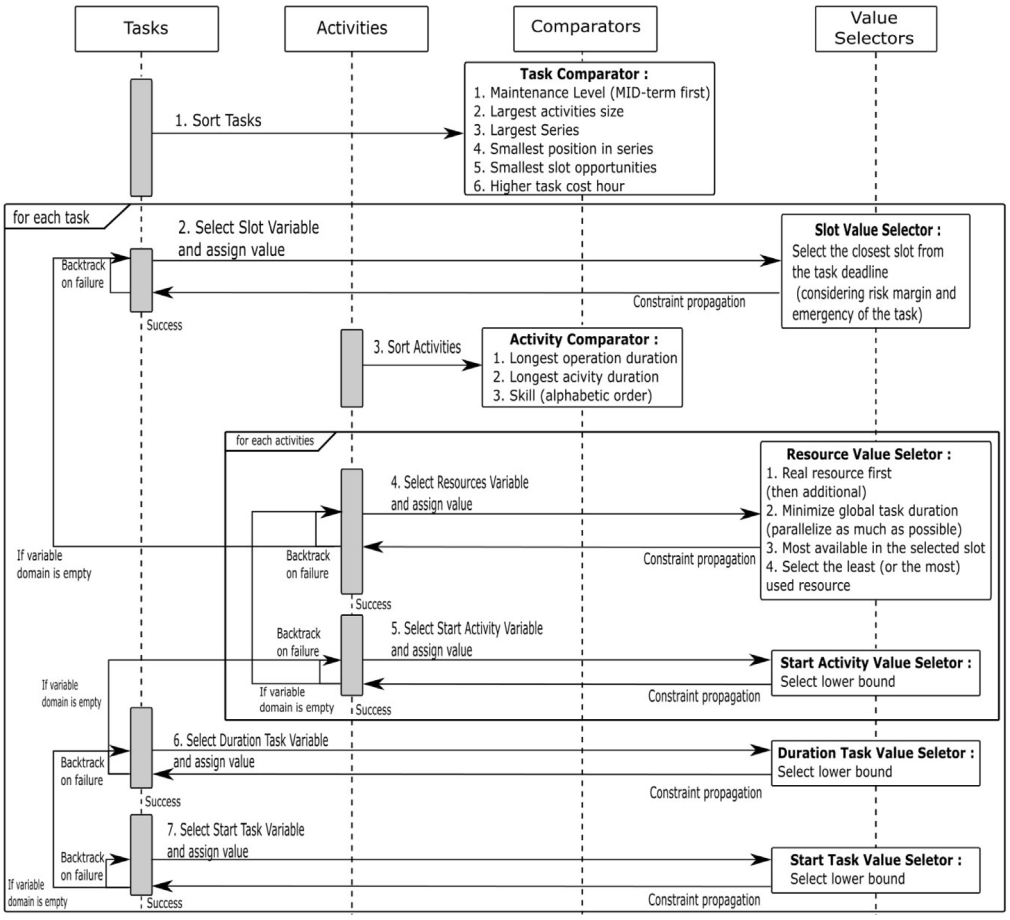

Fig. 4. Sequential diagram of decision process.

5.6. Search strategy

One major difficulty in large scheduling problems is the size of the search space and the important number of solutions. It is practically impossible to enumerate them all.

Hence, it is essential to build a search strategy to answer to business requirements. This custom search is designed to generate a first "good" solution in a "reasonable execution time". By "good solution", we mean a solution satisfying the criteria of an expert. The schedule respects tasks priority with a scheduling rate greater than 60%, resource utilization is optimized and tasks are scheduled close to their respective deadline. By "reasonable execution time" we mean that the solver should be able to find an initial solution in less than five minutes.

It is common in scheduling problems to mimic human behavior to search a first solution. This practice leads the solver using human expertise and it makes the solution understandable by the planners. Shaukat et al. (2020) proposes a sequential heuristic decomposed into two stages: first the job assignment then the timetabling. The search strategy presented in this study follows an expert reasoning. The decision process is described by a sequential diagram in Fig. 4. The strategy is described into two parts: the selection of variables and then the selection of values for each type of variables.

5.6.1. Variable selector

One strategy used in practice is to sort tasks by priorities to try to reduce problem complexity. It is called the rule of "the most urgent first". The strategies usually follow the scheduling methods used in practice. In Hölzel et al. (2012) authors obtained a better optimization by sorting tasks by decreasing labor cost hours to allocate the most expensive task closer to its deadline. Witteman et al. (2021) define three classes of priorities in function of their periodicity interval. The search strategy presented here is also driven by tasks. All tasks are sorted to the following multi-criteria:

1. decreasing maintenance levels (first MID-term then SHORT-term)
2. decreasing activity size
3. decreasing series size
4. increasing task positions in its series
5. increasing slot opportunities
6. decreasing task costs per hour

The first variable to be selected for a task is the assigned slot. Afterwards, all activities from a task are sorted to this multi-criteria:

1. decreasing operation's duration
2. decreasing activities' duration (inside an operation)
3. skill (by alphabetic order)

For each activity (inside a task), an assigned resource variable is selected then its start variable is selected. This makes it possible to immediately check the consistency between the scheduling of the current activity with the other activities of the task already assigned, especially the constraint of maximal task duration.

After that, a task duration variable is chosen, if it is not already assigned during the scheduling of activities. Task duration must be assigned after scheduling activity variables, as it depends whether activities are executed in parallel or sequentially. Starting task variables are assigned at the end.

This selection is repeated for each task according to the initial task sort.

5.6.2. Value selector

A value selector is customized for each type of variable. For slot variables, the value selected corresponds to the closest slot from the task deadline or the earliest slot for urgent tasks.

For resource variables, real resources are selected prior to additional resources. Then, the search selects a resource which minimizes the global task duration by using different resources to parallelize activities as much as possible. After that, the most available resource in the selected slot is chosen. Finally, to minimize (respectively to standardize) resource utilization, the selector chooses the least (respectively the most) used resource.

A lower bound value selector is also used for start of activity variables to fill the beginning of the slot first.

The value selection of task duration variable chooses the lower bound in the domain to minimize the makespan of the task.

The same applies for the starting task variable; the lower bound is chosen first.

6. Experimentation

Two experiments are proposed in this section. In the first one, the performances of CP model are analyzed and compared to MIP solvers to valid the choice of using CP. The second experiment presents a practical use case on real line maintenance instances and compared the results of the CP model with an industrial heuristic solver. Both experiments are performed using an Intel(R) Core(TM) i7-6820HQ CPU 2.70 GHz with 32 Go of RAM.

6.1. Performances analysis

This experimental section analyzes the performance of the CP in comparison with MIP. For this, the model described in the previous sections is implemented in *MiniZinc 2.7.6* (Nethercote et al., 2007) to test easily different solvers. The source code of the model and the instance is available online.[1] Two CP solvers are chosen, *Choco-solver 4.10.14* and *OR-Tools 9.7.2996* and two MIP solvers, *CPLEX 12.10.0 and Gurobi 10.0.3*. The experiment consists in solving, up to the optimal solution, the OALMSP on a small representative instance. The optimal solution is the one that schedules all the tasks that are closest to their target time without using any additional resources and with the least task duration. Solvers are used in this experiment as a "black-box" without specific configurations. Because there is no lexicographical objective in MiniZinc, we use a weighted objective respecting the lexicographical order of Section 5.5. This works on a small instance but could be problematic for instances with larger domain sizes. In order to carry out the experiment in a simple laptop, four threads are used for each solver. A time limit of 12 h is fixed. Note that the solving process is not deterministic due to the use of multiple threads. This can lead to some variability in the solving statistics. However it does not compromise the purpose of the experiment to compare CP and MIP solvers.

The instance used in this experiment simulates a simple use case of line maintenance scheduling for three aircraft on a two-week horizon time. The instance is composed of 28 tasks (one *URGENT* MID-term and 27 SHORT-term tasks), decomposed in 122 activities in total, with 23 available slots on two airports. Maintenance slots range from 40 to 260 min. Activity requirements are divided into 78 activities for human resources, 17 activities for serialized equipment, and 27 for ingredients. The duration for each activity ranges from five minutes to one hour. This simulation presents six periodic series composed of three to seven tasks with a periodicity from two to five days. Then, 12 tasks are in disjunction and 14 precedence relations between operations are represented. In addition, the resource workforce consists of seven simultaneous teams covering nine skills at both airports. They work eight hours a day. Each technician is proficient in two skills. Moreover, six serialized equipment are available on each airport and ingredients are decomposed into five categories (three consumable and two reusable) with sufficient quantities in stock. Additional resources are allowed in limited quantity. Finally, maintenance activities are located in six aircraft zones. Each zone has a limit of one to seven people at a time.

Solver performances are analyzed on three aspects: the size of the models with the number of variables and constraints, the time to find the optimal solution and finally the time to return a first solution and its distance to the optimal one. As shown in Table 1, the MIP solvers use more variables and constraints than CP solvers (maximum 10898 variables and 14406 constraints for CP solvers compared to 769635 variables and 1778571 constraints for MIP solvers). Therefore, the size of models increases from less than 100MB for CP models to four GB for MIP models. The compilation time to convert the model into a flatzinc file is given for information only. On this small representative instance, CP solvers find faster the optimal solution than MIP solvers (16 min maximum compared to 1h15 with *Gurobi*). Note that *CPLEX* is unable to find the optimal solution within 12 h. In addition, CP solvers are able to find a first solution quickly, 35 s for *Choco-Solver* while *CPLEX* needs 2h20. Even if the first solutions returned by CP solvers are far from the optimal solution, they look like the first solution returned by MIP solvers (within 2%).

Therefore, this experiment confirms that CP is suitable to solve the OALMSP, in particular thanks to the small size of the model, its fast solving time and its ability to quickly provide a first solution whose quality can be controlled using an adapted search strategy.

6.2. Practical use case

This section presents a practical line maintenance scheduling use case from an airline's daily life. Aircraft maintenance experts currently create maintenance plans. Suitable slots to schedule tasks in the maintenance plan are quickly found by planners thanks to their knowledge and experience. The tasks are scheduled by priority:

1. the highest priority tasks are scheduled first to ensure the best opportunity and available resource.
2. planning is filled by other lower priority tasks to the best available position as long as it is possible, (for example, increased duration to execute a task, certain activities subcontracted etc.).

In order to choose the best compromise, these decisions require constant discussions between maintenance planners, team leaders, flight operators and authorities. This process can therefore take time and is complex for humans to consider. This logical reasoning by priorities is efficient in practice, but optimizing several objectives at once is challenging for a human being.

In fact, airlines are looking for virtual planning support to simulate multiple maintenance schedules, taking into account operating parameters. Planners need to quickly identify how many tasks can be scheduled depending on the simulation mode and the optimization obtained on several indicators. AIRBUS PROTECT (Airbus, 2022) develops an industrial solution based on a heuristic solver. The schedule generated is considered as acceptable by planners. It will be our reference in this part of the experiment. The heuristic used to solve the LMSP is close to the one proposed by Shaukat et al. (2020). It is inspired by experts practice and it is decomposed in two stages: first the task assignment, then the timetabling of resource in a maintenance opportunity. After the description of this heuristic solver, the CP model proposed in this article is compared with the industrial reference solution on different instances of a real maintenance use case. As seen in Section 4, the aircraft line maintenance scheduling problem is an extension of RCPSP. To recall, in the state of the art RCPSP optimally solve up to 60 activities. The smallest instance considered in our practical use case

[1] https://anonymous.4open.science/r/OALMSPmz-237F/

Table 1
Solvers comparison.

Solvers	Choco-Solver *(4.10.14)*	OR-Tools *(9.7.2996)*	CPLEX *(12.10.0)*	Gurobi *(10.0.3)*
Nb Boolean constraints	832	5164	0	0
Nb Boolean variables	1901	6824	352	352
Nb Integer constraints	5407	9242	1 776 542	1778571
Nb Integer variables	4193	4074	686 831	769 283
Nb Set constraints	156	0	0	0
Nb Set variables	78	0	0	0
Model size	22 MB	65 MB	4316 MB	400 0MB
Compilation time	0.57 s	1.28 s	1484.88 s	1555.19 s
Nb solutions to optimal	76	144	–	8
Solving time	963.24 s	464.37 s	–	8734.79 s
Solving time for the 1st solution	35 s	3 min 1 s	2 h 20 min 40 s	43 min 8 s
Gap to optimality	99.71%	97.81%	99.71%	97.73%
Processing time (global)	16 min 9 s	7 min 46 s	>12 h	1 h 14 min 26 s

is composed of 345 activities. The goal of these experiments is to gradually measure the effect of the size of the instances in order to test both models on a real use case and to show the limits of both solvers. The expected time horizon to reach a real application is about one or two months.

Since the heuristic solver can only generate one solution, only the first solution of the CP solver will be considered. Actually, the constructive search described in Fig. 4 has been tested on the instance from 6.1 with the lexicographical objective of 5.5 using *Choco-Solver* on *Java*. It enables to provide a first solution in 14 s with a gap of seven minutes to the minimum task duration on the last objective, which is very close to optimality.

6.2.1. Reference industrial solution: Heuristic Solver

The industrial solution, that we call *Heuristic Solver*, replicates the current practices of planners. The algorithm can be perceived as a greedy algorithm, but in practice it is important to reproduce human behavior to ensure that the solution is understandable and acceptable to those who intend to use it. This constructive heuristic is designed to return within seconds or minutes at most a partial solution which tries to minimize the potential loss objective (deviation time between scheduled date and target date). The result of the solver helps planners to save valuable time in drawing up maintenance planning schedules. With a solid understanding of the process the schedule can be further refined quickly by planners to ensure that all of the aircraft's maintenance requirements are performed. In addition, to obtain a quick glimpse of the near future, it is useful for planners to simulate a number of scenarios with different configurations. For example, testing the feasibility of maintenance on a horizon time according to available staffing, inventory or flight schedules. Thus, staff or flight schedules can be updated in advance, additional staff may be needed, the supply chain can be adjusted, and certain problems can be anticipated.

Shaukat et al. (2020) give a highest priority to mandatory jobs, i.e. tasks with deadlines and jobs are sorted by deadlines. Consequently, jobs with the same priority value but earlier deadlines will be scheduled first in the aim to minimize their respective deviation between their starting time and their due time. Jobs without deadline are assigned at the end because they present more maintenance opportunities. They are sorted by processing time, from the shortest to the largest. To ensure scheduling of large jobs, a large job is periodically inserted into the sequence. Lastly, some preferences can be included in the sorting process such as the preferred maintenance airports for certain types of jobs, the availability of person-hours for specific types of jobs at different airports, or the availability of spare parts and specialized tools for maintenance jobs. At this task assignation stage, capacity constraints are roughly estimated in order to produce a more or less feasible schedule.

The *Heuristic Solver* follows also a strategy of the "highest priority first". In this case, all tasks are considered as mandatory. The tasks priority is defined by its level of maintenance, MID-term tasks first then SHORT-term tasks. For the same maintenance level, tasks are sorted by their maintenance cost, from the most expensive to the cheapest. Maintenance costs include parts costs and man-hour costs. Finally, tasks are sorted by their deadlines, to schedule first the earliest task. This heuristic determines, in pre-processing stage, possible combinations for assigning a slot and resources to a maintenance task. The model identifies the time slots that match the duration of the task and it ensures that all the required resources are present. A score is calculated for each possible combination to prioritize first the maintenance opportunities close to the deadlines. In the same way, the score of each combination can take in account resource management preferences. For example, a better score can be given to combinations using intern resources than those using additional resources or the most available resources in the inventory are given priority over those that are less available. Like Shaukat et al. (2020), at this stage, resource capacities are checked globally, by aggregating (by skill) the sum of resources used in a maintenance slot (number of resource and required duration) compared to the global capacity (by skill) in each maintenance slot.

The second stage of timetabling is almost identical for both heuristics, except that *Heuristic Solver* allows multi-skilled resources by considering a combination for each skill and a constraint to ensure that a resource is used on only one task at any time. Building upon the first stage result, the idea is to find a feasible schedule with a focus on managing resource consumption at each moment and within every maintenance slot. This stage ensures that simultaneously scheduled jobs can occur within the capacity constraints of the opportunity. All jobs are initially scheduled to begin at the start of the maintenance slot. When a resource violation is detected, which occurs when the resource consumption of scheduled tasks exceeds the capacity of available resources, the conflict must be addressed by exploring alternative starting times. When no solution is possible, a task must be removed. Shaukat et al. (2020) select the job for which the less resources are required and reschedule it on the closet maintenance opportunity. In the *Heuristic Solver* , the second stage still uses the task priority of the first stage and the task with the lowest priority is removed and send back to the task allocation stage.

The *Heuristic Solver* consists of repeating the process of task assignment and timetabling stages several times while scheduling a certain amount of tasks or a maximum time limit. This approach first assigns as many tasks as possible to their best combination. When this becomes impossible, it iteratively chooses a combination with a lower score until all tasks are scheduled. It is not possible to reschedule a task once it has already been scheduled. This incremental process allows the solver to build up the schedule step by step, optimizing the assignment of each task in a partially fixed schedule.

However, this model requires certain assumptions. Only maintenance tasks have a start date, not individual activities. In other words, a task is considered as a block of activities in which activities can be executed in parallel. This model does not consider the sequencing of activities in a task. In addition, as in the RCPSP of Shaukat et al.

Table 2
Instances description.

N° Instance	Time horizon	Fleet size	Nb tasks	Total activities	Nb slots available	Slots duration (*in* h)
Instance 1	1 week	2 AC	28	345	6	8.58–37.50
Instance 2	1 week	4 AC	66	743	18	8.33–37.50
Instance 3	2 weeks	1 AC	30	319	7	1.83–37.50
Instance 4	2 weeks	4 AC	131	1250	30	1.83–57.08
Instance 5	1 month	4 AC	325	3460	70	1.83–57.08
Instance 6	2 months	4 AC	731	8795	156	1.83–77.36

(2020), precedence relations such as the disjunction between the tasks are not considered in this model. However, *Heuristic Solver* considers the periodicity constraint for tasks that belong to the same series. To approximate a human decision-making process, this solver needs to make a large number of assumptions and perform pre-processing to determine the score of each combination. Because of its constructive approach, the solver does not perform a global optimization. However, the solver is still able to assist planners in their decisions and quickly provide a solution which conforms with current industrial standards.

6.2.2. Instances description

The following real-world instances are taken from a use case for the maintenance department of an airline with a fleet of four A330 aircraft of the same type. The input data is extracted from the Maintenance Information System of the airline. The operational flight and staff schedules are drawn from real situations. Most of the maintenance tasks are routine checks such as Daily checks and Weekly checks. There are a few A-checks with a periodicity of less than two months. Certain assumptions were made to compare the two solvers:

1. Logistic inventory (equipment and ingredients) is partially incomplete, so the use of additional resources is authorized so as not to constrain the solver regarding this aspect.
2. Staff team is composed of 72 people covering seven skills. Skills distribution among the staff team is proportional to the activities skills required. Technicians are proficient in one or two skills and they work between six hours and nine hours with two shifts per day.
3. Additional staff is not authorized, only additional serialized equipment and ingredients are allowed.
4. Each zone has an access limited to one or three people (depending on the size of the zone).
5. To compare with the *Heuristic Solver*, precedence constraints (order) between operations in tasks are not considered. So, all operations can be parallelized. Precedence relations will be studied in a specific use case in 6.2.4.
6. Tasks in disjunction are not precised (not covered by *Heuristic Solver*). Only periodicity constraint is considered.
7. Scheduling starts from scratch (no tasks scheduled in initial planning).
8. Task duration is given (value of duration variable is already assigned).
9. Risk management is not studied.

Six instances of increasing size are covered. Table 2 describes the dimensions of the instances. The horizon time of scheduling ranges from one week to two months and the size of the fleet increases from one to four aircraft. Hence, the RCPSP grows from 28 to 731 maintenance tasks i.e., from 319 to 8795 activities. The instances are presented by time horizon and size of the fleet. The first four instances have a maximal task duration of seven hours and 12 min and tasks are composed of a maximum 47 activities. Activities last between seven minutes and five hours. Tasks of the two last instances have a maximal duration of 13 h and 21 min with 307 activities maximum, of duration between seven minutes and nine hours and 11 min. Aircraft have different flight plans. Short-haul aircraft have a lot of short slots and medium/long-haul have less slots but longer.

Results are analyzed by examining KPIs used in a practical context to evaluate the quality of a schedule. The criteria studied are the number of scheduled tasks, the number of distinct resources used, the cost generated by the use of additional resources, the mean deviation time between the scheduled date of a task and its target date and finally the solving time to reach a first solution. Only the first solution returned by the solvers is analyzed, due to the industrially accepted solving time of a few minutes maximum. All simulations are performed using *Choco-solver 4.10.12*. Code runs on *Java 1.8*.

6.2.3. Results analysis

The six instances have been scheduled using the two solvers. Results are summarized in Table 3. The solutions provided by each solver are examined respecting the lexicography order by followings KPIs:

1. the *number of scheduled tasks* which indicates the performance of the solver scheduling first priority tasks.
2. the *additional resources cost* which represents the additional cost of using additional resources to perform activities. This KPI shows the quality of resource allocation of the schedule.
3. the *mean deviation time* which means the average time between the starting time of task and their respective deadline. This indicator highlights the quality of the schedule to minimize potential loss.
4. the *solving time* which expresses the computational performances of the solver. This KPI is rather an acceptance criterion. To be accepted, a first solution must be returned in less than five minutes. Values are given for information only to compare the two solvers.

Firstly, the number of tasks scheduled by the two solvers are closed, even if *CP Solver* is able to schedule one more task on *Instance 2* and *4*. On *Instance 5*, *CP Solver* presents a better scheduling rate of 4%. This is reassuring when seeking to obtain a solution close to an industrial reference in terms of planning rate.

In terms of additional resource costs, *CP Solver* uses less additional resources on the first four instances where the number of scheduled tasks is similar for both solvers. To recall, only additional ingredients and tools are authorized. On *Instance 5*, *CP Solver* presents an additional resource cost higher than *Heuristic Solver* because *CP Solver* schedules 13 more tasks than *Heuristic Solver*. Overall, *CP Solver* uses fewer available time resources (people and serialized equipment) in all instances (5.75 fewer people on average). This frees up more time and allows for better resource allocation.

The deviation time is averaged to study the quality of the scheduled start time of the tasks, regardless of the number of scheduled tasks. To recall, the aim is to save the potential loss by scheduling the tasks close to their target date. On *Instance 1*, the mean deviation time is the same for the two solvers. Indeed, the two solvers scheduled exactly the same tasks at the same time. Globally *CP Solver* presents a better plan with a smaller deviation on average, except in *Instance 3* where *Heuristic Solver* obtained a lower mean of one minute. The difference seems more noteworthy in large instances (1% on *Instance 2*, 8% on *Instance 4* and 4% on *Instance 5*).

Finally, as expected *Heuristic Solver* is faster than *CP Solver* in the majority of instances, except in *Instance 2* where *CP Solver* returns a solution 0.463 s faster. Over a one-week horizon, for one or four aircraft

Table 3
Solver results comparison.

N° Instance	Nb scheduled task		Add. resource cost (*in $*)		Mean deviation time (*in* min)		Solving time (*in* s)	
	Heuristic Solver	*CP Solver*	*Heuristic Solver*	*CP Solver*	*Heuristic Solver*	*CP Solver*	*Heuristic Solver*	*CP Solver*
Instance 1	17/28	17/28	10 640	9310	619	619	1.375	1.498
Instance 2	43/66	44/66	26 201	23 541	865	858	5.026	4.563
Instance 3	23/30	23/30	15 029	13 167	1558	1559	1.580	11.022
Instance 4	90/131	91/131	50 806	44 954	1401	1288	6.992	19.834
Instance 5	219/325	232/325	111 986	126 217	1433	1385	15.596	148.751
Instance 6	527/731	–	898 149	–	2404	–	11.757	–

Table 4
Results of *CP Solver* on *Instance 6*, relaxing of four times maximal task duration.

KPI	*CP Solver*
Nb scheduled tasks	554/731
Additional resource cost (*in* $)	454 461
Mean deviation time (*in* min)	2063
Solving time (*in* s)	325.825
Maximal duration exceed[a] (*in* min)	1296
Mean duration exceed[a] (*in* min)	62.90
Standard deviation duration exceed[a] (*in* min)	209.85

[a] From target duration.

(*Instance 1 and 2*), the two solvers have similar solving time, but over a two-week horizon time (*Instance 3*), *CP Solver* is seven times slower than *Heuristic Solver*. It seems that *CP Solver* needs to reconsider its branching decision several times in order to find a solution in this instance. In *Instance 4*, *CP Solver* is three times slower than *Heuristic Solver* and 10 times slower in *Instance 5*. However, even if solving time is an important aspect for planners, *CP Solver* is always able to find a solution within the five-minute time limit.

Instance 6 is chosen to illustrate a limiting case of two-month, four-aircraft scheduling. *CP Solver* is unable to provide an initial solution within 30 min while *Heuristic Solver* finds a solution scheduling 72% of tasks in about 12 s. However, we have tried to relax the constraint on the maximum duration of the tasks, that is, the activities of the tasks can be scheduled in any number of slots and the solver is able to find a solution. Therefore, we proposed to manage the relaxation of this constraint by allowing the upper bound of the task duration to be proportional to the target duration proposed by the airline. The best coefficient finding a solution within 15 min is four times the target task duration. Table 4 describes the results of this solution.

By relaxing the constraint on the maximum duration for a task to complete all of its activities, *CP Solver* can find a first solution in about five minutes. It schedules more tasks than *Heuristic Solver* (75% vs 72%) with a smaller deviation time from deadlines on average. However, the total task duration exceeds the target duration of the tasks by 10 h with an average excess of one hour per task. The high standard deviation indicates some tasks were overdone. It is recommended to execute all the activities of a task within a close temporal window. This situation deserves to be discussed with experts. Should they extend the duration of all tasks by an average of one hour or should they extend locally the duration of only specific tasks? Another solution could be to add locally additional resources on specific slots. Furthermore, the objective to reduce the exceeded time from target duration could be improved to optimize subsequent solutions but it would take some time.

6.2.4. CP model improvements

In this section, two situations are tested where the *CP Solver* shows scheduling improvements. The first case illustrates the scheduling of priority tasks as Weekly check over a long horizon time. Then, the second case shows how *CP Solver* integrates precedence constraints in a situation where operations inside a task must be ordered to execute it.

Table 5
Results comparison for a schedule of 26 weekly maintenance tasks on four aircraft on seven weeks.

KPI	*Heuristic Solver*	*CP Solver*
Nb scheduled tasks	17/26	23/26
Additional resource cost (*in* $)	45 885	58 121
Mean deviation time (*in* min)	2024	1746
Solving time (*in* s)	7.831	18.571

Table 6
CP Solver results for a schedule of 19 maintenance tasks respecting operations order.

KPI	*CP Solver*
Nb scheduled tasks	19/19
Additional resource cost (*in* $)	0
Mean deviation time (*in* min)	1753
Solving time (*in* s)	11.885

Weekly tasks

This instance is composed of 26 Weekly checks of four aircraft over a seven-week horizon time. These tasks gather 46 activities from 22 min to five hours. 122 slots are available from four hours and five minutes to 52 h. This problem represents the difficulties faced in a real scheduling situation. Indeed, planners usually allocate first large-scale tasks, as Weekly checks, and thereafter they complete the schedule with the remaining tasks.

In this situation presented in Table 5, *CP Solver* returns a better solution in terms of scheduling rate and mean deviation time. The additional resource cost of *CP Solver* is higher because of the 16 more scheduled tasks. Both solvers use 49 available time resources (technicians and serialized equipment). The CP proposition helps planners to better distribute resources in order to schedule a greater number of tasks.

Ordering of activities in tasks

This instance presents a future requirement for planners. Today, precedence between operations through a task is not explicitly given as inputs. Knowledge belongs to planners and human experts. However, in the near future, this information will be indicated and solvers must integrate it. That is also the reason why CP has been chosen to easily model an RCPSP. The following example simulates a problem of 19 tasks on three aircraft between three and 25 activities distributed from one to four operations. Nine tasks have precedence constraints between their operations and 12 tasks have disjunctive constraints. Tasks periodicity range from two days to five days. The maintenance planning is composed of 23 slots between 40 min to four hours and 40 min.

This instance is given as example to illustrate the purpose. All tasks' disjunctive constraints and precedence constraints between operations inside a task are respected. The solver provides a solution scheduling all tasks in 11 s (see Table 6). These results motivate to schedule more precisely tasks sequencing in a slot to be more in adequacy with the real execution of tasks.

6.3. General results

The solvers are compared according general KPIs used by planners to evaluate a schedule. Globally the two solvers are quite close but *CP Solver* presents some significant improvements on its first solution on main objectives. *CP Solver* has a better scheduling rate, especially on complex instances. Then, it is able to optimize better resources allocation and it saves additional resources. Furthermore, *CP Solver* saves more potential by scheduling on average closer to the tasks deadlines. Without surprises, *Heuristic Solver* is faster than *CP Solver* but *CP Solver* returns a first solution in acceptable time of less than five minutes. However both of the two solvers present some limits on some instances. *CP Solver* is suitable for new types of problems thanks to its model flexibility. Even if its solving performances are reasonable for a complex RCPSP, *CP Solver* fails to find a solution on the largest instance within an acceptable time for planners. This will require further works to propose an industrial process for a real production application.

7. Conclusion & future works

This paper presents a Constraint Programming (CP) scheduling model for improving air transport management by optimizing maintenance schedules. One preliminary step is to assist planners in the scheduling of large-scale maintenance tasks so that resources are available for all requirements. The first contribution of this paper is to provide an operational plan for line maintenance that can be directly executed by technicians. This study proposes a model for the Operational Aircraft Line Maintenance Scheduling Problem (OALMSP) by allocating a resource to all the elementary activities that make up a maintenance tasks. Therefore, all precedence relations between operations inside a maintenance task can be scheduled. This multi-level scheduling offers a better granularity in the operational line maintenance scheduling and it aims to reduce maintenance costs and to optimize the use of resources. Additionally, the solver's rapid response enables a reduction in the time required to create a maintenance plan. This problem of aircraft maintenance scheduling is described as a Multi-Skill RCPSP with some extensions applied to maintenance. Addressing OALMSP using CP is the second contribution of this work. CP is adapted to manage the complexity of the OALMSP and to answer to all business requirements. It provides a deterministic method and explicable scheduling decisions: the solution is understandable and directly usable by schedulers. In addition, no learning stage is required and the model can be applied to situations with no historical background. Actually, CP offers a flexibility to adapt the model to other airlines use cases by integrating easily new constraints or objectives. The multi-objective optimization is based on a lexicographical order to respect hierarchical business priorities between objectives. Because of planners' need to respond quickly, we only focused on an initial solution. Our approach achieved the established goals, to address the OALMSP in a single scheduling model and to find a first solution in a reasonable time thanks to our constructive search strategy. The CP model has been compared with an existing industrialized heuristics model applied to real maintenance use cases. Results showed that our method is more accurate in terms of the main objectives. It schedules more tasks in representative instances, it minimizes additional resource utilization and it schedules on average closer to the task deadlines to minimize potential loss. However, the RCPSP presents certain limits to solving large instances and the solving time increases significantly with the horizon time scheduling, which is not completely satisfactory for a daily application in an industrial context.

A future work could be to use the first solution returned by the constructive search and try a local search strategy to improve the quality of the solution. Then, future works will need to focus in particular on a process to use this model proposed in an industrialized application. Indeed, we have encountered certain limits with a global CP model to solve particular industrialized RCPSP in a fast time, as expected by users. Even by improving the search strategy, it would be difficult to ensure a result in a shorter execution time, and we are concerned about proposing a search method that is too specific to one airline and not generic. The main idea will be to split the general model into two sub-problems of lower complexity. A column generation approach might be an alternative to mathematically represent the two levels of scheduling. The master problem could consider the task allocation and global maintenance constraints at task level and then, the pricing problems could manage local constraints of resource allocation at activity level. An hybridization with CP could be used to model the pricing problem and to benefit of CP properties (Gualandi and Malucelli, 2009; Gabteni and Grönkvist, 2009).

Another perspective would be to make planning more dynamic and to use it to manage unexpected events. In order to closely mirror real-world airline scenarios, the model could incorporate risk management to consider unforeseen issues that may arise during task execution. This proactive approach could predict potential failures and anticipate the time to address them. Based on the empirical data shared by the airline and manufacturer, the risk could take into account for each task the probability of finding an issue and its severity. Thanks to the flexibility of Constraint Programming, it should be easy to insert new constraints or new objectives in the model. The risk would be to translate these in the model as a margin duration to anticipate the delay if disruptions occurred. Otherwise, thanks to its short solving time, this line maintenance solver could be used for airline recovery problems. Hence, it would enable to reschedule quickly a maintenance planning in case of disruptions such as unscheduled resource unavailability or a change in flights plan due to a SWAP decision.

Once maintenance scheduling is optimized in an industrial context, a last perspective could be to analyze how maintenance optimization affects the reduction of maintenance duration. This could lead to the reduction or even elimination of some maintenance slots. This would be an improvement to the operational potential of the aircraft.

CRediT authorship contribution statement

Jean-Baptiste Sciau: Writing – original draft, Conceptualization, Data curation, Formal analysis. **Agathe Goyon:** Writing – original draft, Conceptualization, Data curation. **Alexandre Sarazin:** Conceptualization, Writing – review & editing, Data curation, Supervision. **Jérémy Bascans:** Conceptualization, Data curation, Project administration, Writing – review & editing. **Charles Prud'homme:** Formal analysis, Supervision, Writing – review & editing, Methodology. **Xavier Lorca:** Formal analysis, Methodology, Supervision, Writing – review & editing.

Data availability

The data that has been used is confidential.

References

Abreu, A., Piedade, R., 2018. Decision support model in the strategic management of the portuguese air force alpha jet fleet. Asian J. Soc. Sci. Manag. Stud. 5, http://dx.doi.org/10.20448/journal.500.2018.53.72.81.

Aggoun, A., Beldiceanu, N., 1993. Extending chip in order to solve complex scheduling and placement problems. Math. Comput. Modelling 17 (7), 57–73. http://dx.doi.org/10.1016/0895-7177(93)90068-A, URL: https://linkinghub.elsevier.com/retrieve/pii/089571779390068A.

Airbus, P., 2022. Airbus protect - cybersecurity - safe mobility - sustainability. URL: https://www.protect.airbus.com/safety/smartplanif/.

Al-Thani, N.A., Ben Ahmed, M., Haouari, M., 2016. A model and optimization-based heuristic for the operational aircraft maintenance routing problem. Transp. Res. C 72, 29–44. http://dx.doi.org/10.1016/j.trc.2016.09.004, URL: https://linkinghub.elsevier.com/retrieve/pii/S0968090X16301619.

Andrade, P., Silva, C., Ribeiro, B., Santos, B.F., 2021. Aircraft maintenance check scheduling using reinforcement learning. Aerospace 8 (4), 113. http://dx.doi.org/10.3390/aerospace8040113, URL: https://www.mdpi.com/2226-4310/8/4/113.

Aramon Bajestani, M., Beck, J.C., 2011. Scheduling an aircraft repair shop. In: Proceedings of the International Conference on Automated Planning and Scheduling, Vol. 21. pp. 10–17. http://dx.doi.org/10.1609/icaps.v21i1.13450, URL: https://ojs.aaai.org/index.php/ICAPS/article/view/13450.

Artigues, C., Demassey, S., Neron, E., 2008. Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications. ISTE/Wiley, URL: https://hal.archives-ouvertes.fr/hal-00482946.

Badkook, B., 2016. Study to determine the aircraft on ground (AOG) cost of the boeing 777 fleet at X airline. Am. Sci. Res. J. Eng. Technol. Sci. (ASRJETS) 25 (1).

Barnhart, C., Belobaba, P., Odoni, A.R., 2003. Applications of operations research in the air transport industry. Transp. Sci. 37 (4), 368–391. http://dx.doi.org/10.1287/trsc.37.4.368.23276, URL: https://pubsonline.informs.org/doi/10.1287/trsc.37.4.368.23276.

Barták, R., 1999. Constraint programming - What is behind? In: Proceedings of the Workshop on Constraint Programming in Decision and Control.

Bartusch, M., Möhring, R.H., Radermacher, F.J., 1988. Scheduling project networks with resource constraints and time windows. Ann. Oper. Res. 16 (1), 199–240. http://dx.doi.org/10.1007/BF02283745, URL: http://link.springer.com/10.1007/BF02283745.

Basdere, M., Bilge, U., 2014. Operational aircraft maintenance routing problem with remaining time consideration. European J. Oper. Res. 235, 315–328. http://dx.doi.org/10.1016/j.ejor.2013.10.066.

Beldiceanu, N., Bourreau, E., Rivreau, D., Simonis, H., 1996. Solving resource-constrained project scheduling problems with CHIP. In: Fifth International Workshop on Project Management and Scheduling. Poznan, Poland.

Beldiceanu, N., Carlsson, M., Rampon, J.-X., 2005. Global Constraint Catalog. URL: https://sofdem.github.io/gccat/gccat/titlepage.html.

Beldiceanu, N., Contejean, E., 1994. Introducing global constraints in CHIP. Math. Comput. Modelling 20 (12), 97–123. http://dx.doi.org/10.1016/0895-7177(94)90127-9, URL: https://linkinghub.elsevier.com/retrieve/pii/0895717794901279.

Bellenguez, O., Néron, E., 2004. Lower Bounds for the Multi-Skill Project Scheduling Problem with Hierarchical Levels of Skills. p. 243. http://dx.doi.org/10.1007/11593577_14.

Van den Bergh, J., 2013. Aircraft maintenance operations: state of the art. URL: https://lirias.kuleuven.be/1677212?limo=0.

Bird, C.J., 1976. A Branch and Bound Approach to an Aircraft and Maintenance Scheduling Problem (Ph.D. thesis). UNSW Sydney, http://dx.doi.org/10.26190/UNSWORKS/14456, URL: http://hdl.handle.net/1959.4/69905.

Blazewicz, J., Lenstra, J., Kan, A., 1983. Scheduling subject to resource constraints: classification and complexity. Discrete Appl. Math. 5 (1), 11–24. http://dx.doi.org/10.1016/0166-218X(83)90012-4, URL: https://linkinghub.elsevier.com/retrieve/pii/0166218X83900124.

Boere, N.J., 1977. Air Canada saves with aircraft maintenance scheduling. Interfaces 7 (3), 1–13, URL: https://www.jstor.org/stable/25059463. Publisher: INFORMS.

Brailsford, S.C., Potts, C.N., Smith, B.M., 1999. Constraint satisfaction problems: Algorithms and applications. European J. Oper. Res. 119 (3), 557–581. http://dx.doi.org/10.1016/S0377-2217(98)00364-6, URL: https://linkinghub.elsevier.com/retrieve/pii/S0377221798003646.

Callewaert, P., Verhagen, W., Curran, R., 2018. Integrating maintenance work progress monitoring into aircraft maintenance planning decision support. Transp. Res. Procedia 29, 58–69. http://dx.doi.org/10.1016/j.trpro.2018.02.006.

Carlier, J., 1982. The one-machine sequencing problem. European J. Oper. Res. 11 (1), 42–47, URL: https://econpapers.repec.org/article/eeeejores/v_3a11_3ay_3a1982_3ai_3a1_3ap_3a42-47.htm. Publisher: Elsevier.

Carlier, J., Moukrim, A., Xu, H., 2009. The project scheduling problem with production and consumption of resources: A list-scheduling based algorithm. Discrete Appl. Math. 157 (17), 3631–3642. http://dx.doi.org/10.1016/j.dam.2009.02.012, URL: https://linkinghub.elsevier.com/retrieve/pii/S0166218X09000584.

Cho, K.R., 2021. A study on the outsourcing of aircraft maintenance contracts. J. Aerosp. Syst. Eng. 15 (6), 66–71. http://dx.doi.org/10.20910/JASE.2021.15.6.66.

Christofides, N., Alvarez-Valdes, R., Tamarit, J., 1987. Project scheduling with resource constraints: A branch and bound approach. European J. Oper. Res. 29 (3), 262–273. http://dx.doi.org/10.1016/0377-2217(87)90240-2, URL: https://linkinghub.elsevier.com/retrieve/pii/0377221787902402.

Clarke, L.W., Hane, C.A., Johnson, E.L., Nemhauser, G.L., 1996. Maintenance and crew considerations in fleet assignment. Transp. Sci. 30 (3), 249–260. http://dx.doi.org/10.1287/trsc.30.3.249, URL: https://pubsonline.informs.org/doi/10.1287/trsc.30.3.249.

Cros, G., 2022. Airline maintenance cost executive commentary.

Demeulemeester, E.L., Herroelen, W.S., 1997. New benchmark results for the resource-constrained project scheduling problem. Manage. Sci. 43 (11), 1485–1492. http://dx.doi.org/10.1287/mnsc.43.11.1485, URL: https://pubsonline.informs.org/doi/10.1287/mnsc.43.11.1485.

Deng, Q., Santos, B.F., 2022. Lookahead approximate dynamic programming for stochastic aircraft maintenance check scheduling optimization. European J. Oper. Res. 299 (3), 814–833. http://dx.doi.org/10.1016/j.ejor.2021.09.019, URL: https://linkinghub.elsevier.com/retrieve/pii/S0377221721007943.

Deng, Q., Santos, B.F., Curran, R., 2020. A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization. European J. Oper. Res. 281 (2), 256–273. http://dx.doi.org/10.1016/j.ejor.2019.08.025, URL: https://linkinghub.elsevier.com/retrieve/pii/S0377221719306782.

Eggenberg, N., Salani, M., Bierlaire, M., 2010. Constraint-specific recovery network for solving airline recovery problems. Comput. Oper. Res. 37 (6), 1014–1026. http://dx.doi.org/10.1016/j.cor.2009.08.006, URL: https://linkinghub.elsevier.com/retrieve/pii/S030505480900210X.

Eltoukhy, A.E.E., Chan, F.T.S., Chung, S.H., Niu, B., 2018. A model with a solution algorithm for the operational aircraft maintenance routing problem. Comput. Ind. Eng. 120, 346–359. http://dx.doi.org/10.1016/j.cie.2018.05.002, URL: https://www.sciencedirect.com/science/article/pii/S0360835218302031.

FAA, 2007. Advisor circular 43-12A CHG 1-preventive maintenance. URL: https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_43-12A_CHG_1.pdf.

Feo, T.A., Bard, J.F., 1989. Flight scheduling and maintenance base planning. Manage. Sci. 35 (12), 1415–1432. http://dx.doi.org/10.1287/mnsc.35.12.1415, URL: https://pubsonline.informs.org/doi/10.1287/mnsc.35.12.1415.

Gabteni, S., Grönkvist, M., 2009. Combining column generation and constraint programming to solve the tail assignment problem. Ann. Oper. Res. 171 (1), 61–76. http://dx.doi.org/10.1007/s10479-008-0379-1, URL: http://link.springer.com/10.1007/s10479-008-0379-1.

Ghamginzadeh, A., Najafi, A., Khalilzadeh, M., 2021. Multi-objective multi-skill resource-constrained project scheduling problem under time uncertainty. Int. J. Fuzzy Syst. 23, http://dx.doi.org/10.1007/s40815-020-00984-w.

Gopalan, R., Talluri, K.T., 1998. The aircraft maintenance routing problem. Oper. Res. 46 (2), 260–271. http://dx.doi.org/10.1287/opre.46.2.260, URL: https://pubsonline.informs.org/doi/10.1287/opre.46.2.260.

Grönkvist, M., Kjerrström, J., 2005. Tail assignment in practice. In: Fleuren, H., den Hertog, D., Kort, P. (Eds.), Operations Research Proceedings 2004. Springer, pp. 166–173. http://dx.doi.org/10.1007/3-540-27679-3_21.

Gualandi, S., Malucelli, F., 2009. Constraint programming-based column generation. 4OR 7 (2), 113–137. http://dx.doi.org/10.1007/s10288-009-0101-4, URL: http://link.springer.com/10.1007/s10288-009-0101-4.

Hane, C.A., Barnhart, C., Johnson, E.L., Marsten, R.E., Nemhauser, G.L., Sigismondi, G., 1995. The fleet assignment problem: Solving a large-scale integer program. Math. Program. 70 (1), 211–232. http://dx.doi.org/10.1007/BF01585938, URL: http://link.springer.com/10.1007/BF01585938.

Haouari, M., Shao, S., Sherali, H., 2013. A lifted compact formulation for the daily aircraft maintenance routing problem. Transp. Sci. 47, 508–525. http://dx.doi.org/10.1287/trsc.1120.0433.

Hartmann, S., Briskorn, D., 2010. A survey of variants and extensions of the resource-constrained project scheduling problem. European J. Oper. Res. 207 (1), 1–14. http://dx.doi.org/10.1016/j.ejor.2009.11.005, URL: https://linkinghub.elsevier.com/retrieve/pii/S0377221709008558.

Hoffman, K.L., Padberg, M., 1993. Solving airline crew scheduling problems by branch-and-cut. Manage. Sci. 39 (6), 657–682, URL: https://www.jstor.org/stable/2632545. Publisher: INFORMS.

Hölzel, N.B., Schröder, C., Schilling, T., Gollnick, V., 2012. A maintenance packaging and scheduling optimization method for future aircraft. In: Air Transp. Oper.. IOS Press, pp. 343–353. http://dx.doi.org/10.3233/978-1-61499-119-9-343, URL: https://ebooks.iospress.nl/doi/10.3233/978-1-61499-119-9-343.

Hughes, G., 2006. Right place, right time: the art of short- long- term planning. Aircfat Commer. June/July (46), 56–57, URL: https://pdfcoffee.com/maintenance-planning-4-pdf-free.html.

Kanet, J., Ahire, S., Gorman, M., 2004. Constraint programming for scheduling. In: Handbook of Scheduling: Algorithms, Models, and Performance Analysis. URL: https://ecommons.udayton.edu/mis_fac_pub/1.

Kelley, J.E., Walker, M.R., 1959. Critical-path planning and scheduling. In: Papers Presented at the December 1-3, 1959, Eastern Joint IRE-AIEE-ACM Computer Conference on - IRE-AIEE-ACM '59 (Eastern). ACM Press, pp. 160–173. http://dx.doi.org/10.1145/1460299.1460318, URL: http://portal.acm.org/citation.cfm?doid=1460299.1460318.

Kinnison, H.A., Siddiqui, T., 2013. Aviation Maintenance Management, second ed. McGraw-Hill.

Klabjan, D., 2005. Large-scale models in the airline industry. In: Column Generation. pp. 163–196. http://dx.doi.org/10.1007/0-387-25486-2_6, Journal Abbreviation: Column Generation.

Kowalski, M., Izdebski, M., Żak, J., Gołda, P., Manerowski, J., 2021. Planning and management of aircraft maintenance using a genetic algorithm. Eksploatacja i Niezawodność (Maint. Reliab.) 23 (1), 143–153. http://dx.doi.org/10.17531/ein.2021.1.15, URL: https://ein.org.pl/Planning-and-management-of-aircraft-maintenance-using-a-genetic-algorithm,158377,0,2.html.

Laborie, P., 2005. Complete MCS-based search: Application to resource constrained project scheduling. In: Nineteenth International Joint Conference on Artificial Intelligence. Edinburgh, Scotland, UK.

Lagos, C., Delgado, F., Klapp, M.A., 2020. Dynamic optimization for airline maintenance operations. Transp. Sci. 54 (4), 998–1015. http://dx.doi.org/10.1287/trsc.2020.0984, URL: https://pubsonline.informs.org/doi/10.1287/trsc.2020.0984.

Lauriere, J.-L., 1978. A language and a program for stating and solving combinatorial problems. Artificial Intelligence 10 (1), 29–127. http://dx.doi.org/10.1016/0004-3702(78)90029-2, URL: https://www.sciencedirect.com/science/article/pii/0004370278900292.

Levin, A., 1971. Scheduling and fleet routing models for transportation systems. Transp. Sci. 5 (3), 232–255. http://dx.doi.org/10.1287/trsc.5.3.232, URL: https://pubsonline.informs.org/doi/abs/10.1287/trsc.5.3.232. Publisher: INFORMS.

Liess, O., Michelon, P., 2008. A constraint programming approach for the resource-constrained project scheduling problem. Ann. Oper. Res. 157 (1), 25–36. http://dx.doi.org/10.1007/s10479-007-0188-y.

Lohatepanont, M., Barnhart, C., 2004. Airline schedule planning: Integrated models and algorithms for schedule design and fleet assignment. Transp. Sci. 38 (1), 19, URL: https://www.academia.edu/18245646/Airline_Schedule_Planning_Integrated_Models_and_Algorithms_for_Schedule_Design_and_Fleet_Assignment.

Maher, S.J., Desaulniers, G., Soumis, F., 2014. Recoverable robust single day aircraft maintenance routing problem. Comput. Oper. Res. 51, 130–145. http://dx.doi.org/10.1016/j.cor.2014.03.007, URL: https://linkinghub.elsevier.com/retrieve/pii/S0305054814000628.

Moudani, W.E., Mora-Camino, F., 2000. A dynamic approach for aircraft assignment and maintenance scheduling by airlines. J. Air Transp. Manag. 6 (4), 233–237. http://dx.doi.org/10.1016/S0969-6997(00)00011-9, URL: https://www.sciencedirect.com/science/article/pii/S0969699700000119.

Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G., 2007. MiniZinc: Towards a standard CP modelling language. In: Bessière, C. (Ed.), Principles and Practice of Constraint Programming – CP 2007, Vol. 4741. Springer Berlin Heidelberg, pp. 529–543. http://dx.doi.org/10.1007/978-3-540-74970-7_38, URL: http://link.springer.com/10.1007/978-3-540-74970-7_38. Series Title: Lecture Notes in Computer Science.

Neumann, K., Schwindt, C., 2003. Project scheduling with inventory constraints. Math. Methods Oper. Res. 56 (3), 513. http://dx.doi.org/10.1007/s001860200251, URL: https://search.ebscohost.com/login.aspx?direct=true&AuthType=sso&db=plh&AN=8887522&site=ehost-live&custid=ns065117. Publisher: Springer Nature.

Pachet, F., Roy, P., 1999. Automatic Generation of Music Programs. pp. 331–345. http://dx.doi.org/10.1007/978-3-540-48085-3_24, Pages: 345.

Palpant, M., Artigues, C., Michelon, P., 2004. LSSPER: Solving the resource-constrained project scheduling problem with large neighbourhood search. Ann. OR 131, 237–257. http://dx.doi.org/10.1023/B:ANOR.0000039521.26237.62.

Papakostas, N., Papachatzakis, P., Xanthakis, E., Mourtzis, D., Chryssolouris, G., 2010. An approach to operational aircraft maintenance planning. Decis. Support Syst. 604–612. http://dx.doi.org/10.1016/j.dss.2009.11.010.

PeriyarSelvam, U., Tamilselvan, T., Thilakan, S., Shanmugaraja, M., 2013. Analysis on costs for aircraft maintenance. Adv. Aerosp. Sci. Appl. 177–182.

Pimapunsri, K., Weeranant, D., 2018. Solving complexity and resource-constrained project scheduling problem in aircraft heavy maintenance. Int. J. Appl. Eng. Res. 8998–9004.

Pinson, E., Prins, C., Rullier, F., 1994. Using tabu search for solving the resource-constrained project scheduling problem. In: International Workshop on Project Management and Scheduling, PMS'94. Louvain, Belgium, URL: https://hal-utt.archives-ouvertes.fr/hal-02895382.

Pritsker, A.A.B., Watters, L.J., Wolfe, P.M., 1969. Multiproject scheduling with limited resources: A zero-one programming approach. Manage. Sci. 16 (1), 93–108, URL: http://www.jstor.org/stable/2628369.

PSPLIB, 2005. Project scheduling problem library - PSPLIB. URL: https://www.om-db.wi.tum.de/psplib/main.html.

Rossi, F., Beek, P.v., Walsh, T., 2006. Handbook of Constraint Programming. Elsevier.

Rossi, F., van Beek, P., Walsh, T., 2008. Chapter 4 constraint programming. In: van Harmelen, F., Lifschitz, V., Porter, B. (Eds.), Foundations of Artificial Intelligence. In: Handbook of Knowledge Representation, vol. 3, Elsevier, pp. 181–211. http://dx.doi.org/10.1016/S1574-6526(07)03004-0, URL: https://www.sciencedirect.com/science/article/pii/S1574652607030040.

Sarac, A., Batta, R., Rump, C.M., 2006. A branch-and-price approach for operational aircraft maintenance routing. European J. Oper. Res. 175 (3), 1850–1869. http://dx.doi.org/10.1016/j.ejor.2004.10.033, URL: https://www.sciencedirect.com/science/article/pii/S0377221705004807.

Schutt, A., Feydy, T., Stuckey, P.J., Wallace, M.G., 2013. Solving RCPSP/max by lazy clause generation. J. Sched. 16 (3), 273–289. http://dx.doi.org/10.1007/s10951-012-0285-x.

Senturk, C., Ozkol, I., 2016. The effects of the use of single task-oriented maintenance concept and more accurate letter check alternatives on the reduction of scheduled maintenance downtime of aircraft. Int. J. Mech. Eng. Robotics Res. 7 (2), 189–196. http://dx.doi.org/10.18178/ijmerr.7.2.189-196, URL: http://www.ijmerr.com/index.php?m=content&c=index&a=show&catid=158&id=935.

Shaukat, S., Katscher, M., Wu, C.L., Delgado, F., Larrain, H., 2020. Aircraft line maintenance scheduling and optimisation. J. Air Transp. Manag. 89, 101914. http://dx.doi.org/10.1016/j.jairtraman.2020.101914, URL: https://linkinghub.elsevier.com/retrieve/pii/S096969972030497X.

Sherali, H., Bish, E., Zhu, X., 2006. Airline fleet assignment concepts, models, and algorithms. European J. Oper. Res. 172, 1–30. http://dx.doi.org/10.1016/j.ejor.2005.01.056.

Sriram, C., Haghani, A., 2003. An optimization model for aircraft maintenance scheduling and re-assignment. Transp. Res. A 37 (1), 29–48. http://dx.doi.org/10.1016/S0965-8564(02)00004-6, URL: https://linkinghub.elsevier.com/retrieve/pii/S0965856402000046.

Sung, I., Choi, B., Nielsen, P., 2020. Reinforcement learning for resource constrained project scheduling problem with activity iterations and crashing. IFAC-PapersOnLine 53 (2), 10493–10497. http://dx.doi.org/10.1016/j.ifacol.2020.12.2794, URL: https://www.sciencedirect.com/science/article/pii/S2405896320335588.

Tian, Y., Xiong, T., Liu, Z., Mei, Y., Wan, L., 2022. Multi-objective multi-skill resource-constrained project scheduling problem with skill switches: Model and evolutionary approaches. Comput. Ind. Eng. 167, 107897. http://dx.doi.org/10.1016/j.cie.2021.107897, URL: https://www.sciencedirect.com/science/article/pii/S0360835221008019.

Wen, X., Sun, X., Ma, H.L., Sun, Y., 2022. A column generation approach for operational flight scheduling and aircraft maintenance routing. J. Air Transp. Manag. 105, 102270. http://dx.doi.org/10.1016/j.jairtraman.2022.102270, URL: https://linkinghub.elsevier.com/retrieve/pii/S0969699722000904.

Witteman, M., Deng, Q., Santos, B.F., 2021. A bin packing approach to solve the aircraft maintenance task allocation problem. European J. Oper. Res. 294 (1), 365–376. http://dx.doi.org/10.1016/j.ejor.2021.01.027, URL: https://linkinghub.elsevier.com/retrieve/pii/S0377221721000576.

Xu, Y., Wandelt, S., Sun, X., 2019. Stochastic Tail Assignment under Recovery.

Xu, Y., Wandelt, S., Sun, X., 2021. Airline integrated robust scheduling with a variable neighborhood search based heuristic. Transp. Res. B 149, 181–203. http://dx.doi.org/10.1016/j.trb.2021.05.005, URL: https://linkinghub.elsevier.com/retrieve/pii/S0191261521000850.

Xu, Y., Wandelt, S., Sun, X., 2023. IMMUNER: Integrated multimodal mobility under network disruptions. IEEE Trans. Intell. Transp. Syst. 1–19. http://dx.doi.org/10.1109/TITS.2022.3224413, URL: https://ieeexplore.ieee.org/document/10027853/.

Yuan, P., Han, W., Su, X.c., Liu, J., Song, J., 2018. A dynamic scheduling method for carrier aircraft support operation under uncertain conditions based on rolling horizon strategy. Appl. Sci. 8, 1546. http://dx.doi.org/10.3390/app8091546.

Zhao, X., Song, W., Li, Q., Shi, H., Kang, Z., Zhang, C., 2022. A deep reinforcement learning approach for resource-constrained project scheduling. In: 2022 IEEE Symposium Series on Computational Intelligence. SSCI, IEEE, pp. 1226–1234. http://dx.doi.org/10.1109/SSCI51031.2022.10022122, URL: https://ieeexplore.ieee.org/document/10022122/.