Université Fédérale

Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut Supérieur de l'Aéronautique et de l'Espace

Présentée et soutenue par :
Franco PESCHIERA

le jeudi 19 novembre 2020

Titre :

Méthodes exactes et heuristiques pour l'optimisation de la planification conjointe des maintenances et des missions des aéronefs militaires

Exact and heuristic methods to optimize maintenances and flight schedules of military aircraft

---

École doctorale et discipline ou spécialité :

ED AA : Génie industriel

Unité de recherche :

Équipe d'accueil ISAE-ONERA MOIS

Directeur(s) de Thèse :

M. Alain HAïT (directeur de thèse)
Mme Olga BATTAïA (co-directrice de thèse)

Jury :

M. Farouk YALAOUI Professeur Université de Technologie de Troyes - Président, Rapporteur
M. Alain HAïT Professeur ISAE-SUPAERO - Directeur de thèse
Mme Olga BATTAïA Professeure KEDGE Business School - Co-directrice de thèse
M. Nicolas DUPIN Maître de conférences Université Paris Saclay - Co-directeur de thèse
M. Cédric LEBOUCHER Dr. Centre d'analyse technico-opérationnelle de défense
M. Ronald MC GARVEY Associate Professor University of Missouri
M. Robert PELLERIN Professeur École Polytechnique Montréal - Rapporteur

Acknowledgment

I thank the DGA for having partly financed this thesis. I also thank Dassault Aviation for having financed this thesis and for their time and feedback.

I thank my PhD advisors: Alain Haït, Nicolas Dupin and Olga Battaïa for believing in me and for pushing me to do my best during these 3 years.

I thank all my colleagues at SUPAERO for serious, non-serious and always interesting lunch discussions.

I thank Robert Dell, his family and everyone at NPS for the warm welcome I had in Monterey.

I thank my parents Alfonso and Patricia, my brothers Alfonso and Sergio and my sister Natalia, because they're responsible for most of who I am.

I thank Laura for motivating me in last stretch of the process and making me a better person.

Finally, I thank Alvaro Garcia and Miguel Ortega for being such good professional and life mentors and for, consciously or unconsciously, inspiring me to do a PhD in Operations Research.

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

AIA	*Atelier industriel de l'aéronautique*
AMS	Aircraft Maintenance Scheduling
CG	Column Generation
CO	Combinatorial Optimization
CP	Constraint Programming
CVaR	Conditional Value at Risk
DGA	*Direction générale de l'armement*
DMAé	*Direction de la maintenance aéronautique*
DoD	Department of Defense (USA)
DP	Dynamic Programming
FMP	Flight and Maintenance Planning problem
GRASP	Greedy Randomized Adaptive Search Procedures
HC	Hill Climbing
HDV	Flight Hours (*Heures de vol*)
HH	Man hours (*Heures-homme*)
IP	Integer Programming
ISAE	*Institut supérieur de l'aéronautique et de l'espace*
LP	Linear Programming
MFMP	Military Flight and Maintenance Planning problem
MH	Metaheuristic
MIP	Mixed Integer Programming
ML	Machine Learning
MP	Mathematical Programming
MSPPNP	Maintenance Scheduling and Production Planning of Nuclear Plants
NPS	Naval Postgraduate School

NRP	Nurse Rostering Problem
OR	Operations Research
PLNE	*Programmation Linéaire en Nombres Entiers*
PPC	*Programmation Par Contraintes*
RCT	Remaining Calendar Time (*Butée calendaire*)
RFPP	Reduced Flight Planning Problem
RFT	Remaining Flight Time (*Butée horaire*)
RSAMP	Rolling Stock Assignment and Maintenance Plan
SL	Supervised Learning
SMPTSP	Shift Satisfaction Personnel Task Scheduling Problem
SPPRC	Shortest Path Problem with Resource Constraints
SPP	Shortest Path Problem
TA	Tail Assignment
TOL	Tolerance
TSP	Traveling Salesman Problem
VG	Oil check (*Visite de graissage*)
VI	Intermediate check (*Visite intermédiaire*)
VND	Variable Neighborhood Descent
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem
VS	Security check (*Visite de sécurité*)
VX	Overhaul check (*Grande visite*), also referred to as “M”

Glossary

Aircraft Any flying vehicle such as: airplane, helicopter or drone that accomplishes a mission.

Availability Indicates the number of aircraft in a fleet that are not in maintenance in each time period.

Candidate Aircraft that has the correct type and capabilities and so can be assigned to a particular mission.

Capabilities Set of optional aircraft characteristics that may be required by a mission in order for aircraft to be considered candidates. An aircraft can have none or more capabilities, a mission can have at most one.

Check A preventive maintenance operation done to an aircraft to detect and repair any problem that may cause a malfunction on the unit. Several types of checks exist, varying in the duration, thoroughness, frequency and parts to be maintained.

Cluster A set of missions such that each mission has exactly the same type, capabilities and, as a result, aircraft candidates.

Fleet A set of aircraft that shares the same maintenance resources.

Flight potential Synonym to "Remaining flight time".

Maintenance capacity The maximum number of aircraft that can be in maintenance at each time period.

Minimum default usage Flight hours each aircraft is required to fly when not assigned to a mission or in maintenance..

Mission A set of flights with a military objective such as training, reconnaissance, humanitarian or defense. They are planned usually months in advance and require a number of aircraft with specific characteristics. A mission can last a few weeks to several months or years.

Mission assignment The deployment of an aircraft to a mission during a set of consecutive periods. This deployment can have a length smaller or equal to the length of the mission. Each mission has limits on how short or long the deployment can be.

Remaining calendar time Expresses the maximum number of periods, starting at the end of each time period, before an aircraft must undergo a check.

Remaining flight time Measures the maximum number of hours an aircraft can fly before requiring a check, at the end of each time period.

Serviceability Indicates if an aircraft is capable at the beginning of each time period to perform a mission (i.e., is not undergoing a check).

Special mission Mission that has a capability (around 10% of missions).

Sustainability Measures the capacity of an aircraft to continue doing missions in the future (i.e., remaining flight time).

Type of mission and aircraft Each mission and each aircraft have one and only one type. Only aircraft with the same type as the mission can be considered candidates.

Résumé de la thèse

Introduction

Contexte industriel

Qu'il s'agisse d'une maison, d'une voiture ou d'une centrale nucléaire, la maintenance est un élément clé lors de la construction ou de l'achat d'un actif. Par exemple, les coûts annuels d'entretien d'une maison peuvent être égaux à 1 à 4 % du prix d'achat [6]. Ce coût représente le prix à payer pour conserver l'actif acquis dans des conditions d'opération normales et prolonger sa durée de vie aussi longtemps que cela est économiquement viable.

En ce qui concerne des caractéristiques des maintenances : coût élevé, opacité et importance; l'entretien des aéronefs militaires n'échappe pas à la règle. Par exemple, en prenant le coût : l'investissement en maintenance du département de la Défense des États-Unis en 2019 est d'environ 78 milliards de dollars [8]. De plus, c'est souvent le même fournisseur qui approvisionne les actifs, l'infrastructure et leur entretien. Cela ne fait qu'augmenter l'opacité intrinsèque des projets militaires. Enfin, l'importance de garantir le bon fonctionnement d'un appareil chargé de la sécurité d'un pays ne peut être sous-estimée.

Sur le plan technique, la maintenance des aéronefs militaires est similaire à celle de l'industrie civile : Il existe plusieurs types de contrôles effectués sur les aéronefs, bien que la nomenclature, les fréquences et les parties prenantes changent. En France, c'est la "Direction de la Maintenance Aéronautique (DMAé)" récemment créée (2018) qui est chargée de planifier et d'organiser les contrôles des avions militaires. Les types des opérations de maintenance pour la flotte de Mirage 2000 sont : (1) la visite de graissage (VG), réalisée tous les 6 à 8 mois calendaires; (2) la visite intermédiaire (VI), réalisée tous les 14 à 16 mois calendaires; (3) la visite de sécurité (VS), réalisée toutes les 300 à 600 heures de vol; et (4) la grande visite (VX), réalisée toutes les 1000 à 1200 heures de vol ou tous les 60 mois.

Les trois premiers types de maintenance sont effectués dans la base aérienne où se situe l'avion. Les grandes visites (VX) se font à l'Atelier Industriel de l'Aéronautique (AIA), situé dans la région de Clermont-Ferrand (voir Figure 1).

Étant donné le coût élevé des ressources nécessaires pour effectuer les tâches de maintenances et la spécialisation de l'expertise requise, l'offre est généralement limitée et peu flexible. Par conséquent, un plan de maintenance qui ne cherche pas à profiter de cette capacité de manière optimale peut entraîner un grand nombre d'aéronefs immobilisés au sol en attente d'une maintenance à certaines périodes. Ce même plan aura, à d'autres périodes, une partie de la capacité de maintenance inutilisée. Cette indisponibilité réduit le potentiel réel de la flotte et demande donc d'avoir un nombre d'avions plus élevé pour satisfaire les contraintes opérationnelles. Il est donc primordial de disposer des outils de planification performants afin d'améliorer la disponibilité de la flotte et de réduire son coût opératif total.

Figure 1: Une photo de l'atelier AIA de maintenance à Clermont-Ferrand où la maintenances de type VX est effectuée pour le Mirage 2000[1].

Le coût réel d'un plan de maintenance dépend du nombre de maintenances nécessaires pour respecter toutes les exigences de la mission. Bien que les coûts de ces visites ne soient pas publics dans le cas des avions militaires, les coûts réels des visites de type D (équivalents aux visites VX) des avions Boeing varient entre 1 million de dollars et 6 millions de dollars selon le modèle d'avion [3]. En France, le coût total annuel de la maintenance des avions militaires est passé de 3,2 milliards d'euros en 2012 à 4 milliards d'euros en 2017 [113]. Un plan de maintenance efficace devrait minimiser le nombre d'heures de vol restantes qui sont "perdues" lorsque la maintenance est effectuée et que le potentiel de vol est rétabli. Ainsi, ce plan utilisera chaque avion autant que possible avant de programmer une maintenance.

Actuellement, la planification des maintenances militaires se fait généralement manuellement (dans une feuille de calcul Excel) et prend du temps. Cela limite le nombre de scénarios qui peuvent être évalués avant de prendre une décision et augmente le temps de réaction lors du traitement de nouvelles informations qui demandent de modifier le plan d'origine. Par conséquent, ce processus de planification manque de robustesse et de flexibilité.

Ainsi, le besoin d'une planification efficace et fiable des vols et de la maintenance des avions est clairement une priorité tant dans les contextes commerciaux que militaires. Néanmoins, produire un plan aussi détaillé pour une flotte de taille importante, tout en planifiant sur un long horizon et en tenant compte des multiples objectifs inhérents à la planification à long terme n'est pas une tâche facile.

Les techniques de recherche opérationnelle sont employées pour résoudre des problèmes combinatoires tels que le "Military Flight and Maintenance Planning" (MFMP) afin de produire des solutions de très bonne qualité (voire optimales). Le succès de l'application de telles techniques à un nouveau problème nécessite la combinaison d'une expertise du domaine et d'une connaissance théorique avancée. Cette dernière ne peut être obtenue qu'en réalisant une

étude approfondie de la structure du problème mathématique sous-jacent et en recherchant des approches de solutions innovantes compatibles avec le problème. Cette thèse fournit une telle analyse.

Présentation du problème

Le problème MFMP décide de l'affectation d'une flotte hétérogène d'avions militaires $i \in \mathcal{I}$ à un ensemble de missions $j \in \mathcal{J}$ déjà planifiées tout en décidant des opérations de maintenances nécessaires pour chaque aéronef. Les contraintes peuvent être classifiées en trois groupes: (1) les besoins des missions, (2) les maintenances à effectuer pour maintenir la flotte en bon état et (3) la capacité de la flotte à chaque période.

Un ensemble des missions existe tout au long d'un horizon divisé en périodes $t \in \mathcal{T}$. Chaque mission j nécessite un nombre minimum d'aéronefs R_j parmi les aéronefs "compatibles" $i \in \mathcal{I}_j$ (i.e., aéronefs possédant les caractéristiques requises pour cette mission). Chaque aéronef affecté à une mission j vole un nombre d'heures H_j à chaque période pendant laquelle il est affecté et doit rester affecté à cette mission pendant au moins MT_j^{min} et au plus MT_j^{max} périodes consécutives.

Chaque maintenance a une durée fixe de M périodes et ne peut pas être interrompu: pendant ce temps, l'avion ne peut être affecté à aucune mission. La butée calendaire (rct) est définie comme le nombre maximal de périodes après une maintenance pendant lequel un avion peut voler si les autres contraintes sont respectées. Même si un avion n'a pas volé depuis sa dernière maintenance, il devra subir une maintenance à la fin de sa butée calendaire. La butée horaire (rft) est définie comme le nombre maximal d'heures de vol qu'un aéronef peut effectuer avant avoir besoin d'une maintenance. La rct (rft) avant la première période de l'horizon de planification pour l'avion i est Rct_i^{Init} (Rct_i^{Init}). Après une visite, un avion récupère sa rct (rft) à sa valeur maximal de E^{max} (H^{max}). De plus, après chaque visite, l'avion ne peut pas rentrer en maintenance pendant au moins E^{min} périodes. Le nombre total de maintenances simultanées à chaque période ne peut pas dépasser la capacité de l'atelier C^{max}.

Nous appelons un avion "disponible" s'il peut être utilisé en mission au début d'une période donnée, i.e. l'avion n'est pas en maintenance. La "durabilité" d'un avion est défini comme le nombre total d'heures de vol lui restantes à la fin de chaque période. Pour garantir à la fois la durabilité et la disponibilité de la flotte à chaque période, les missions sont regroupées en clusters. Formellement, un cluster est un ensemble de missions telles que toutes les missions partagent le même type, les mêmes capacités et, par conséquence, les mêmes avions candidats. Pour chaque cluster, un nombre minimal d'aéronefs disponibles (A_{kt}^{Clust}) et une durabilité minimale (H_{kt}^{Clust}) sont définis comme contraintes pour chaque période. Tous les aéronefs disponibles ont par défaut une consommation minimale à chaque période égale à U^{min} heures de vol, qu'ils sont obligés d'effectuer dès qu'ils ne sont pas affectés à une mission ou à une maintenance.

Comme cela a été mentionné, un des objectifs du plan de maintenance est de minimiser les heures de vol perdues en programmant les maintenances vers la fin des butées calendaires.

Un objectif complémentaire est d'équilibrer la charge de vol entre les avions de la flotte afin que les fréquences des maintenances des avions soient similaires.

État de l'art

Dans la littérature, le problème de planification de vol et de maintenance pour des avions militaires est appelé "Military Flight and Maintenance Planning problem" ou MFMP, une variante du problème "Flight and Maintenance Planning" ou FMP. Le MFMP fut présenté dans un premier temps par [153], puis de nombreuses contributions dans ce domaine ont été réalisées [98, 48, 174, 108, 154, 116, 55, 152].

Ces travaux peuvent être classifiés selon la taille de l'horizon de planification en trois groupes: court, moyen et long terme. Les problèmes de court terme planifient une année maximum et divisent l'horizon en périodes d'un jour ou une demi journée [116, 48, 146]. Ceux de moyen terme construisent des plannings d'entre 6 mois et 2 années dont chaque période a un taille hebdomadaire ou mensuelle [152, 174, 98, 85, 136]. Les maintenances sont ordonnées toutes les 200 à 400 heures de vol, ce qui correspond à des types B et C. Finalement, dans le long terme les tailles d'horizon varient entre 5 et 10 ans et les périodes sont mensuelles [55]. Les maintenances concernées sont celles de type D, qui souvent prennent plusieurs mois et dont la fréquence oscille entre 1000 et 1200 heures de vol ou 60 mois calendaires.

Le MFMP ressemble au problème de FMP civil, où l'on doit planifier les opérations de maintenance des avions civils en même temps que l'on choisit les affectations de plan de vols. Ce deuxième problème est plus présent dans la littérature [26, 161, 124, 150, 95, 148]. Cependant, les deux problèmes rencontrent des différences importantes. Par exemple, les avions militaires retournent à leur base après chaque mission contrairement aux avions civils qui effectuent des rotations. Les objectifs son aussi différents: le problème militaire maximise la disponibilité de la flotte et le problème civil cherche la réduction des coûts.

Bien qu'étant un problème lié à l'aviation, le MFMP a des similitudes avec d'autres problèmes d'optimisation de maintenances traités par des méthodes de recherche opérationnelle comme pour les chemins de fer : "rolling stock assignment and maintenance plan" [61, 102, 109]. Nous pouvons aussi remarquer une certaine proximité entre le problème que nous étudions et le problème d'emploi du temps des infirmières (Nurse Rostering Problem, NRP) où l'on doit choisir les créneaux de travail pour chaque employé et chaque jour. D'autres travaux sur le sujet de l'ordonnancement du personnel ont été réalisés par [38, 39, 45, 156].

La méthodologie la plus utilisée dans la littérature pour résoudre le MFMP est la Programmation Linéaire aux Nombres Entiers (PLNE) en combinaison avec des heuristiques permettant de trouver des solutions optimales ou presque optimales. Des exemples de ces heuristiques sont la fixation de variables ([55]) et la décomposition du problème en plusieurs sous-problèmes par rapport à la flotte [116]. Une alternative à la PLNE, fréquemment utilisée actuellement, est la Programmation Par Contraintes (PPC) [138], elle a été notamment employée dans le contexte du concours Optiplan pour optimiser les maintenances de la flotte

de Mirage 2000 de l'Armée de l'Air en France en 2018 [9].

Pour le problème FMP, plusieurs formulation et approches de résolution existent dans la littérature: Génération de Colonnes (CG), formulations de flux, et formulations de réseau temps-espace. Les techniques de résolution utilisées sont des modèles PLNE [150, 95, 148], PPC [83, 74] et des heuristiques [124].

Dans le cas de CG, le temps de résolution du sous-problème et particulièrement important. En conséquence, des algorithmes de Programmation Dynamique (DP) sont appliqués. Quelques exemples sont le problème du Plus Court Chemin (SPP) [24, 27] et le Plus Court Chemin avec contraintes de ressources (SPPRC) [150, 148]. Ces techniques sont aussi utiles pour résoudre des voisinages dans de Recherche à Voisinage très Large (VLNS) [17].

Une des faiblesses des modèles exacts comme la PLNE est l'impossibilité de passer à l'échelle pour résoudre des très grandes instances du problème combinatoire traité. Ces dernières années, des contributions de l'Apprentissage Automatique (ML) à l'optimisation combinatoire permettent d'obtenir des informations sur les solutions optimales ou quasi-optimales Bello et al. [28], Bengio et al. [29]. Cette information peut être utilisée pour réduire la taille du problème de base, en rendant possible une résolution exacte [104, 179, 111].

L'objectif de cette thèse est de réaliser une étude approfondie de la structure du problème mathématique lié à la planification de maintenances d'avions militaires dans le contexte français et de développer des approches innovantes pour le résoudre et aider les décideurs. Cette étude commence par l'analyse de la complexité du problème étudié et l'élaboration de méthodes exactes de résolution.

Analyse de la complexité et méthodes exactes

Ce chapitre analyse la complexité du problème MFMP, propose une formulation mathématique pour le résoudre de façon exacte et présente une heuristique pour trouver des solutions initiales. Finalement, le modèle résultant et l'heuristique sont évalués avec des instances générées avec un simulateur d'instances compatible avec les besoins de l'Armée de l'Air française.

Analyse de la complexité

Nous partons du fait que le FMP existe dans NP parce que nous pouvons vérifier en temps polynomial si une solution est réalisable ou pas. Il reste à déterminer si le problème FMP se trouve dans P ou dans $NP - Difficile$. Pour arriver à ce résultat, nous avons fait la réduction suivante.

Tout d'abord nous, prenons un problème déjà démontré $NP - Difficile$, que nous appelons le "Shift Satisfaction Personnel Task Scheduling Problem" ou SSPTSP, présenté par

Arkin and Silverberg [20]. Deuxièmement, nous simplifions notre problème MFMP à un cas particulier où nous n'avons pas de décisions de maintenance ni de fonction objectif. Nous appelons ce problème simplifié le "Reduced Flight Planning Problem" ou RFPP. Troisièmement, nous démontrons que n'importe quelle instance du problème SSPTSP peut être transformée de façon polynomial à une instance du problème RFPP.

Cette transformation consiste à faire une équivalence directe entre les tâches du problème SSPTSP et les missions du problème RFPP et la compatibilité tâche-ressource devient la compatibilité mission-avion. Ainsi nous démontrons que si le problème SSPTSP est $NP - Difficile$, le problème RFPP l'est aussi et si le problème RFPP l'est, notre problème de base MFMP l'est aussi. Après cette conclusion, nous développons un modèle PLNE pour ce problème afin d'étudier sa résolution exacte en pratique et les difficultés à passer à l'échelle lors de la résolution des instances de taille réelle.

Méthodes exactes

Dans notre formulation du modèle PLNE, nous utilisons les variables de décision suivantes : a_{ijt} qui vaut 1 si l'avion i est affecté à la mission j à la période t et m_{it} qui vaut 1 si l'avion i commence une opération de maintenance au début de la période t.

Ce premier modèle est développé pour deux objectifs : d'une part, pour maximiser la disponibilité de la flotte en minimisant le nombre total de maintenances, d'autre part, pour maximiser le potentiel en nombre d'heures de vol (durabilité) de la flotte à la fin de l'horizon de planification.

Nous intégrons dans ce modèle des nouvelles contraintes qui n'avaient pas été utilisées dans la littérature du problème MFMP : les contraintes des butées calendaires pour les maintenances et les durées minimales d'affectation pour les missions.

Heuristique pour construire des solutions initiales

Une métaheuristique constructive de type recuit simulé est implémenté pour générer rapidement des solutions réalisables ou presque réalisables. L'objectif est de fournir ces solutions au modèle PLNE pour ainsi faire un démarrage à chaud et améliorer la performance de ce dernier. A chaque itération, l'heuristique décide, dans un premier temps, les maintenances des avions en fonction du besoin et, ensuite, elle réalise toutes les affectations possibles aux missions en considérant les limites imposées par les maintenances déjà décidées.

Toutes les affectations aux maintenances et missions se font de façon aléatoire. A la fin de chaque itération, un morceau de la solution et libéré. Ce morceau peut consister en (1) toute l'information pour un avion, (2) une fenêtre de temps pour un ensemble d'avions, ou (3) une maintenance qui peut changer de position. La condition d'arrêt est un temps maximal d'exécution ou le fait de trouver une solution réalisable.

Simulation des instances de taille réelle

Une instance comprend un ensemble de données d'entrée suffisant pour décrire complètement un cas particulier d'un problème MFMP. Il s'agit, entre autres, de toute l'information des missions à réaliser, du nombre et toute l'information des avions, des caractéristiques des opérations de maintenances à réaliser et de l'horizon de planification.

Pour pouvoir comparer et valider la performance des différentes techniques de résolution de façon objective, un simulateur d'instances est créé. Ce simulateur prend comme entrée des paramètres qui conditionnent la taille et la difficulté d'une instance à résoudre et retourne des instances aléatoirement générées en fonction des paramètres spécifiés.

Les sources de diversification utilisées pour générer les instances différentes sont, pour les avions : l'état initial et capacité ; et pour les missions : la durée, les heures de vol par période, les avions requis, les capacités requises et la durée minimale et maximale d'affectation.

Expérimentation et résultats

Plusieurs expériences sont effectuées pour évaluer les performances du modèle PLNE en fonction de la taille des instances et de leurs configurations, ainsi que l'impact des solutions initiales obtenues par l'heuristique sur la performance du modèle.

Concernant la sensibilité du modèle par rapport à la taille des instances, la performance diminue avec la longueur de l'horizon $|\mathcal{T}|$ et avec la taille de la flotte $|\mathcal{I}|$. D'un autre côté, les changements en la consommation minimale d'heures de vol (U^{min}), la quantité minimale d'heures de vol pour chaque cluster (HP^K) et la fréquence des maintenances en heures de vol (H^{max}) sont les paramètres dont les changements ont le plus d'impact sur la performance du modèle.

Finalement, les solutions fournies par l'heuristique améliorent, en moyenne, la performance du modèle, même si cet impact reste modeste.

Nouveau modèle, bornes valides et bornes apprises pour le MFMP

Dans ce chapitre, nous proposons un approche basé sur un nouveau modèle PLNE auquel nous appliquons plusieurs types de coupes. D'une part, des coupes valides à partir des conditions initiales et, de l'autre, des coupes apprises à partir des prédictions des caractéristiques de la solution optimale ou quasi optimales. Ces prédictions sont obtenues en entraînant un modèle d'apprentissage automatique sur les données d'entrée et les résultats de 5000 instances.

Cette approche permet de réduire le temps de résolution avec peu de pertes d'optimalité et de faisabilité par rapport aux méthodes matheuristiques alternatives. Les résultats expéri-

mentaux obtenus montrent l'intérêt d'une nouvelle façon d'ajouter des coupes apprises aux problèmes en fonction de la prédiction des caractéristiques spécifiques des solutions.

Nouvelle formulation PLNE

Une nouvelle formulation PLNE est conçue, où les affectations des missions sont énumérées explicitement ainsi que toutes les possibilités d'affectations des maintenances. Tandis que cette formulation génère un grand nombre de variables, elle fournit une meilleure relaxation linéaire que la formulation initiale. Ces propriétés vont nous permettre d'implémenter des coupes agressives et efficaces.

Les variables principales de décision sont : $a_{ijtt'}$ qui vaut 1 si l'avion i est affecté à la mission j entre les périodes t et t'; et $m_{itt'}$ qui vaut 1 si l'avion i commence sa première maintenance au début de la période t et commence sa deuxième maintenance au début de la période t'.

Études de bornes valides

Plusieurs bornes ont été formulées et calculées pour contraindre le nombre total des maintenances et des missions dans une flotte d'avions. Ces bornes s'appliquent au niveau d'un avion individuel, d'un ensemble d'avions similaires et de la flotte entière. Ces bornes sont construites sur les variables de décision d'affectation des missions ($a_{ijtt'}$) et des maintenances (m_itt').

Par exemple, dans le cas d'un avion, nous pouvons calculer les missions qui peuvent être affectées à un avion au début de l'horizon en fonction de son état initial : plus la butée horaire de l'avion est élevée au début de l'horizon, plus il pourra voler, et par conséquent, plus il pourra être affecté à des missions. Autrement dit, avant que l'avion i puisse réaliser sa première maintenance, les heures de vol des affectations aux missions ne peuvent pas être supérieures à sa butée horaire initiale.

En même temps, des affectations proches du début de l'horizon peuvent être directement éliminées si les heures de vol demandées par ces affectations dépassent la butée horaire initiale de l'avion.

Apprentissage des solutions optimales

Notre approche consiste à générer et résoudre un grand nombre de petites instances (5000) du problème MFMP. Des algorithmes d'apprentissage supervisé (régressions linéaires en quantiles) sont appliqués sur l'ensemble de solutions optimales (ou proche de l'optimum) obtenues en tenant compte de leur caractéristiques, pour construire une prédiction de certaines propriétés des solutions en fonction des données d'entrée.

Les données d'entrée validées comme statistiquement pertinentes dans cette prédiction sont : la somme totale d'heures de vol des missions, la somme des états initiaux des avions, la variance de la demande d'heures de vol entre les périodes, entre autres. En sortie, la durée moyenne entre les deux maintenances et le nombre total de maintenances dans une planification sont les deux caractéristiques les plus utiles. Pour chacune des deux, nous estimons une borne inférieure et une borne supérieure.

Pour chaque nouvelle instance, nous appliquons le modèle de prédiction entraîné pour obtenir quatre bornes (ou coupes) pour la solution optimale (bornes maximales et minimales pour chacune de deux caractéristiques de la solution). Ces bornes ne sont pas des bornes valides : leur application peut théoriquement invalider une solution optimale qui ne fera plus partie de ce nouvel espace de solutions réduit. Cependant, la probabilité de garder une solution optimale ou proche de l'optimale est élevée et dépend de la qualité de la prédiction.

Expérimentation et résultats

Nous avons conduit des expériences numériques afin d'évaluer la performance de notre approche, les pertes d'optimalité et les pertes de faisabilité. Les méthodes comparés sont (1) les deux modèles PLNE du Chapitre 3 et Chapitre 4; (2) plusieurs variantes des coupes apprises appliquées à chacun de ces deux modèles; (3) des matheurisitiques plus traditionnelles implémentées à partir du modèle PLNE du Chapitre 4. Les critères de comparaison sont : la faisabilité, l'optimalité et la performance. Les modèles du groupe (1) sont utilisés comme référence.

Concernant la perte de faisabilité, les modèles du groupe (2) se trouvent beaucoup plus proches des modèles exactes (1) que ceux du groupe (3), où la quantité des solutions non réalisables et des violations des contraintes souples augment significativement. Les modèles avec les coupes apprises n'ont presque pas de solution irréalisable additionnelle et un nombre très bas de contraintes souples violées.

Concernant la perte d'optimalité, les modèles du groupe (2) ont des pertes médianes d'optimalité de 3.5% et généralement de moins de 5%.

Finalement, les approches des groupes (2) et (3) offrent des gains importants en temps de calcul : le temps de résolution est réduit considérablement pour l'ensemble d'instances ainsi que l'écart d'optimalité.

Ces résultats montrent donc les avantages de l'utilisation de ce type de coupes apprises pour obtenir des meilleures performances sans pour autant perdre des solutions optimales.

Matheuristique de descente à voisinage variable basée sur des graphes

Tant les travaux précédents que l'analyse de la littérature du problème de FMP indiquent que les instances de ce problème sont souvent suffisamment grandes pour que les méthodes standards de résolution ont des difficultés pour passer à l'échelle. Dans ces cas, un compromis entre la qualité de la solution obtenue et les ressources nécessaires (dont le temps de résolution) pour l'atteindre doit être fait. Un exemple de ce compromis est l'utilisation de techniques d'apprentissage détaillées dans le Chapitre 4. Une alternative plus traditionnellement employée dans le domaine de la recherche opérationnelle est l'utilisation des métaheuristiques où, pour un problème donné, on conçoit un algorithme qui parcourt l'espace des solutions en suivant un paradigme de résolution existant dans la littérature. Les paradigmes métaheuristiques les plus connus sont le recuit simulé, l'algorithme génétique et l'algorithme de colonies de fourmis.

Nous proposons une métaheuristique appelée Descente à Voisinage Variable (VND) basée sur deux voisinages différents et complémentaires : un horizon roulant et un algorithme dynamique. Ce dernier s'appuie sur la représentation de l'espace des solutions comme un ensemble de graphes.

Représentation en graphes

Chaque graphe représente toutes les possibilités d'états et d'affectations de missions et de maintenances à un avion pendant tout l'horizon de planification. Cette représentation en tant que graphe permet l'utilisation des algorithmes efficaces pour l'échantillonnage de chemins et pour l'obtention du plus court chemin entre deux nœuds. Nous nommons *pattern* un choix de chemin pour chaque avion.

Malheureusement, la taille de ces graphes augmente de façon exponentielle en fonction de la taille de l'horizon et ils doivent être réduits avec des méthodes d'approximation pour obtenir des graphes qui puissent être stockés en mémoire.

Voisinages

Deux types de voisinage sont envisagés. Le premier exploite le graphe existant d'un avion pour changer toutes ses affectations en résolvant un SPP entre le nœud du début et le nœud de la fin de l'horizon de planification. Les poids donnés au graphe à chaque itération dépendent de l'état actuel de la solution de façon à ce que la solution optimale du SPP corresponde au voisinage optimal de la solution. Nous l'appelons "SPA".

Le deuxième, appelé "RH", consiste à appliquer un voisinage RH construit sur le modèle PLNE du Chapitre 4. Le problème initial est résolu en fixant à tour de rôle toutes les affectations qui n'appartiennent pas à la fenêtre définie par un ensemble d'avions et périodes.

L'utilisation de ce voisinage permet d'améliorer significativement la qualité de la solution pour certaines instances, mais au prix d'une résolution plus longue.

Solution initiale

Une solution initiale est construite avec trois méthodes : (1) le voisinage "SPA" exécuté une fois pour chaque avion ; (2) l'algorithme "maintFirst" ; (3) le modèle PLNE du Chapitre 4, qui corresponde au voisinage "RH" avec une fenêtre de taille $|\mathcal{I}| \times |\mathcal{T}|$.

Expérimentation et résultats

Nous comparons nos approches selon les critères suivants : la qualité de voisinages, la qualité de la solution initiale, la performance en temps de résolution et l'écart d'optimalité.

La combinaison des voisinages "SPA" et "RH" se montre très efficace quand elle est comparée avec la performance de chaque voisinage indépendant. Cette synergie est liée à la nature différente et complémentaire de chaque voisinage et valide le choix du schéma de la métaheuristique.

Finalement, la performance de la métaheuristique est comparée avec celle du modèle PLNE pour des instances de grande taille. Les résultats permettent de voir des gains importants en temps de résolution et qualité des solutions démontrant ainsi le grand potentiel des techniques de ce type.

Conclusions et perspectives

Cette thèse étudie le problème de la planification de vols et de maintenance des avions militaires (MFMP) et propose plusieurs nouvelles méthodes pour le résoudre efficacement. Nous avons élaboré la première formulation du problème dans le contexte français avec des contraintes et objectifs inédits, jamais considérés dans la littérature. Nous avons étudié la complexité de ce nouveau problème d'optimisation en le reliant au problème de la planification des emplois du temps du personnel (NRP).

La première méthode de résolution que nous avons développée pour ce nouveau problème est basée sur un modèle PLNE. Ainsi les instances de petite et moyenne taille peuvent être résolues à l'optimalité et des solutions de bonne qualité sont fournies pour les instances de grande taille. Nous avons conduit l'analyse de sensibilité pour ce modèle afin de déterminer quels étaient les paramètres qui impactaient le plus d'efficacité de la résolution. Pour améliorer la performance du solveur PLNE, nous avons élaboré et implémenté une métaheuristique de recuit simulé qui fournit des solutions initiales. Les tests numériques ont démontré l'utilité de l'application des ces méthodes approchées pour diminuer le temps de résolution.

La deuxième méthode de résolution est basée sur un modèle PLNE différent pour lequel nous avons développée des coupes valides efficaces et une heuristique basée sur l'apprentissage automatique (ML) afin de générer des coupes "apprises". L'apprentissage est réalisé sur des milliers d'instances de petite taille pour trouver des relations entre les caractéristiques des données d'entrée et les caractéristiques spécifiques des solutions optimales. L'heuristique utilise les informations apprises pour appliquer des coupes non valides à l'espace de solutions, en réduisant considérablement la taille du problème sans pour autant exclure des solutions de bonne qualité. Les tests numériques ont montré des améliorations considérables des performances de résolution. Ce travail est l'un des premiers dans le domaine à combiner avec succès des techniques d'apprentissage supervisé pour prédire les caractéristiques de solutions optimales lors de la résolution d'un modèle PLNE.

Enfin, une méthode de résolution alternative que nous avons développée pour le traitement efficace des instances de très grande taille est une matheuristique basée sur la combinaison d'une technique de résolution de type horizon roulant (RH) et d'un algorithme de programmation dynamique (DP) avec une descente à voisinage variable (VND). Cette approche permet de générer de bons patterns pour chaque avion en résolvant un problème du plus court chemin. La combinaison de partitions basées sur le temps (RH) et de partitions basées sur les avions (DP) s'avère particulièrement efficace pour éviter les minimum locaux et atteindre des solutions quasi optimales en temps de résolution très court.

Les contributions de cette thèse ont donné lieu aux publications suivantes:

- F. Peschiera, O. Battaïa, A. Haït, and N. Dupin. Bi-objective mip formulation for the optimization of maintenance planning on french military aircraft operations. 2018. URL `http://oatao.univ-toulouse.fr/20766/`
- F. Peschiera, R. Dell, J. Royset, A. Haït, N. Dupin, and O. Battaïa. A novel solution approach with ML-based pseudo-cuts for the Flight and Maintenance Planning problem. *OR Spectrum*, pages 1–30, jun 2020. ISSN 0171-6468. doi: 10.1007/s00291-020-00591-z. URL `http://link.springer.com/10.1007/s00291-020-00591-z`
- F. Peschiera, A. Haït, N. Dupin, and O. Battaïa. Long term planning of military aircraft flight and maintenance operations. Technical report, ISAE-SUPAERO, UniversitÃĬ de Toulouse, France, 2020. URL `https://arxiv.org/abs/2001.09856` (soumis pour publication)
- F. Peschiera, N. Dupin, A. Haït, and O. Battaïa. Novel graph-based matheuristic to solve the flight and maintenance planning problem. Forthcoming (soumis pour publication)

Des résultats ont été présentés dans les communications suivantes :

- F. Peschiera, A. Haït, N. Dupin, and O. Battaïa. Maintenance planning on french military aircraft operations. In *Congrès annuel de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)*, pages 1–2, Lorient, FR, 2018. URL `http://oatao.univ-toulouse.fr/20036/`

- F. Peschiera, A. Haït, N. Dupin, and O. Battaïa. A novel mip formulation for the optimization problem of maintenance planning of military aircraft. In *XIX Latin-Iberoamerican Conference on Operations Research*, pages 1–2, Lima, PE, 2018
- F. Peschiera, N. Dupin, O. Battaïa, and A. Haït. An alternative mip formulation for the military flight and maintenance planning problem. In *Congrès annuel de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)*, pages 1–2, Montpellier, FR, 2020. URL `https://oatao.univ-toulouse.fr/26033/`

Finalement, une version adaptée de l'heuristique de recuit simulé a été testée, validée et exploitée par Dassault Aviation sur des instances réelles des flottes des avions Mirage 2000.

Le travail réalisé ouvre plusieurs perspectives de recherche.

En ce qui concerne la modélisation du problème, des facteurs de terrain tel que la possibilité de stocker un avion non utilisé pour prolonger sa butée calendaire ou la modélisation de la capacité de maintenance non pas comme une quantité fixe mais une variable peuvent enrichir la formulation du problème en ajoutant des nouvelles variables liées à ces décisions.

L'analyse de complexité peut également être étendue. Dans cette thèse, seul le problème de planification à long terme a été étudié. Il sera intéressant de considérer les problèmes de planification à court et moyen terme, car dans ce contexte les missions sont généralement représentées par des affectations continues d'heures de vol par période. Cela pourrait également conduire à identifier des cas particuliers où le problème devient polynomial. C'est probablement le cas lorsque les exigences de la mission sont des heures de vol continues et que les opérations de maintenance sont déjà planifiées (un problème de planification de vol sous contraintes de maintenance).

Lorsqu'il s'agit d'horizons de planification de plusieurs années, la prise en compte des incertitudes est cruciale. En garantissant certains niveaux de disponibilité et de pérennité à chaque période pour chaque catégorie d'avion, les solutions que nous trouvons avec les méthodes de résolution proposées dans cette thèse sont déjà caractérisées par un certain niveau de robustesse. Néanmoins, en ayant plus d'informations disponibles sur les sources des incertitudes et leur impact, la robustesse d'une solution peut être considérablement améliorée en utilisant une approche adaptée à la nature des informations qui peuvent être obtenues concernant les incertitudes.

Le potentiel de la combinaison des techniques d'apprentissage et de l'optimisation n'a pas été pleinement exploré. Plusieurs pistes pour de futures recherches peuvent être envisagées, comme par exemple, l'utilisation de la régression logistique pour estimer la probabilité qu'un pattern (i.e., une variable) participe à la solution optimale. Cette probabilité peut ensuite être utilisée pour échantillonner des solutions. Un autre exemple est l'utilisation de la sélection automatique des caractéristiques pour obtenir, plus rapidement et plus facilement, le modèle prédictif.

En ce qui concerne l'application des informations ML, une séparation claire entre la prédiction et l'optimisation a été maintenue : de bons patterns sont d'abord prédits pour une

instance donnée, puis utilisés pour résoudre le problème avec ces informations. Une alternative consiste à appliquer la prédiction à l'intérieur du processus de résolution en échantillonnant des patterns dans le cadre d'une technique de décomposition plus large où l'échantillonnage est utilisé comme première étape pour résoudre un sous-problème. Dans ce cas, l'échantillonnage peut être adapté par de nouvelles informations provenant de la solution actuelle, telles que des violations de contraintes ou des prix ombre.

Les techniques de décomposition qui peuvent bénéficier de ce type d'échantillonnage sont celles où le nombre de variables de décision croît de manière exponentielle par rapport à la taille du problème. Un candidat évident est les représentations utilisées dans les décompositions CG. Un autre bon candidat potentiel est celui où un très grand graphe explicite est construit pour chaque état possible de chaque avion.

Les graphes acycliques dirigés peuvent être utilisés pour échantillonner des modèles potentiellement intéressants. En attribuant des poids aux arêtes et en fixant une distance maximale K pour les motifs extraits, on peut garantir la qualité des chemins extraits. Puis, en affectant soigneusement des probabilités pour choisir les voisins de chaque nœud, on obtient un échantillonnage non biaisé de patterns qui ont une qualité meilleure que K.

Enfin, cet échantillonnage (quelle que soit sa mise en œuvre) peut ensuite être utilisé de plusieurs manières. La première consiste à transmettre les patterns échantillonnés à un modèle maître du type couverture par ensembles qui est compatible avec les contraintes de routage trouvées dans les problèmes FMP et VRP. Une deuxième option consiste à intégrer les patterns échantillonnés dans la phase constructive d'une heuristique GRASP afin de produire de nombreuses solutions rapides et raisonnablement bonnes qui peuvent être améliorées ultérieurement avec la recherche locale.

Exact and Heuristic Methods to Optimize the Flight and Maintenance planning of Military Aircraft

Chapter 1

Introduction

Contents

1.1 Industrial context

1.1.1 The maintenance industry

Be it a house, a car or a nuclear power plant, maintenance is a key consideration when building or acquiring a new asset. For example, the yearly maintenance costs for a house can equal 1-4% of the purchase price [6]. This cost represents the price to pay to keep the acquired asset in normal operation conditions and prolong its life as long as it is economically possible.

Maintenance is important. It is an intrinsic part of property rights. Being the owner of a product that cannot be repaired easily and cheaply reduces the value of said product. There exist recent movements that demand a "right to repair" for acquired goods. For example, thanks to a new law ratified by the European Commission, from 2021, manufacturers will be forced to make appliances long-lasting and to provide spare parts easily for up to ten years [7].

Maintenance can also be opaque. New large infrastructure projects are always in people's minds and, as a result, in politicians mouths. Once the project is finished, though, little public and therefore media attention is given to the maintenance of those projects. This can, and often does, make the maintenance operations more opaque than the actual construction because they are done without as much society attention. As an example, after several fatal accidents, an investigation in Indonesia in 2008 into the Adam Air airline "revealed serious deficiencies in maintenance and safety procedures" [167], including conflicts of interest in the inspection of aircraft and bribes in certifications.

Maintenance is quite expensive. Usually, it involves several specialized labor-intensive tasks that are hard to automatize. On top of that, in a quest for lower production costs, companies have made products harder and harder to repair or maintain. Canada estimates its companies spent 3.3% of GDP on repairs in 2016, more than twice as much as the country spends on research and development [5]. Although the value of a good maintenance policy is usually hidden behind the uneventful and correct functioning of a system, the cost of a lack of maintenance of public infrastructures can be measured from time to time. This became all too evident recently with the collapse of the Genoa bridge in Italy. The lack of investment in maintenance was one of the main reasons for this tragedy that caused the death of 43 people in August 2018.

Finally, maintenance plays an important role in sustainable development. The most cost-effective way to reduce overall consumption, and thus the pollution associated with that consumption, is to extend the life of goods. It is not a coincidence that the same movements calling for easier repairs are also those that condemn "planned obsolescence". This is the name of a practice attributed to goods manufacturers where the end of life of a product is purposely decided since its conception in order for customers to buy new ones, and thus increase the demand for the product. This practice is a crime in France since 2015 [2]. The length of life for a product is specially important for the environment when discussing infrastructure that has a large upfront cost (be it financial, social or environmental) but a very low operation cost. This is precisely the case for renewable energy sources such as hydroelectric, wind and solar: their return of investment, and thus their competitiveness, is tightly tied to the duration of their infrastructure.

With regards to previously presented characteristics of maintenances: high cost, opaqueness and importance; military maintenance is like the regular maintenance, only more so. Let's take cost. If maintenance is expensive, and military maintenance is more so: the 2019 US Department of Defense investment in maintenance is around $ 78 billion [8]. In addition to this, it is usually the case that, by design, there is a unique supplier for the goods, infrastructure and their maintenance. This only increases the inherit opaqueness that military projects tend to already have in the first place. Finally, the importance of guaranteeing the correctly functioning of the apparatus that is in charge of the security of a country cannot be underestimated.

1.1.2 Maintenances in the aviation industry

In the aircraft industry, maintenance is done via various types of maintenance operations also known as checks. These checks vary in frequency, duration and thoroughness. The frequency is usually measured in a combination of flight hours, takeoff-landing cycles, and calendar months; the duration, in months or man-hours of labor. What follows is an extract from the "UK Aerospace Maintenance, Repair, Overhaul (MRO) & Logistics" report in 2018 [73].

Base or Heavy maintenance for airlines and other air transport operators has a range of 'lettered' checks from a simple A-check through to a comprehensive D-Check. The type of

check required depends on the number of hours the aircraft has flown since its last check, the age of the aircraft, and the number of cycles (take-offs and landings) carried out. These checks are labour intensive and require taking the aircraft out of service resulting in lost revenues and aircraft availability.

These 'letter checks' typically include:

A-Check This check is carried out approximately every 80 to 100 flight hours, which is every 7 to 9 days. It needs between 10 and 20 man-hours and is usually performed overnight while the aircraft is at the gate or in a hangar.

B-Check This check is a more thorough maintenance check and is normally carried out every two months (approximately 500 to 600 flight hours). This maintenance is carried out in a hangar and requires approximately 100 to 300 man-hours depending on the size and complexity of the aircraft.

C-Check This check is very thorough and comprehensive. Virtually the entire aircraft goes through an exhaustive series of checks, inspections and overhaul work. The C-Checks typically occur every two years and require 10,000 to 30,000 man-hours, depending on the aircraft type and take two to four weeks to complete.

D-Check This check is the most comprehensive and occurs approximately every 6 years. It is a check that, more or less, takes the entire airplane apart for inspection and overhaul. Such a check can usually demand up to 50,000 man-hours and it can generally take up to 2 months to complete, depending on the aircraft and the number of technicians involved. It must be performed at a suitable maintenance base.

There is a recent trend to include some D-Check work in each C-Check and try and eliminate the D-Check, to improve the availability of the aircraft for commercial service. D-checks would normally be carried out at a heavy maintenance and engineering facility such as at British Airway's Engineering in Cardiff and Monarch Engineering at Luton.

Table 1.1 shows an example of check frequencies per type of check for some commercial aircraft.

In many ways, the military maintenance of aircraft is similar to the civil industry. There are several types of checks that are done to aircraft, although the names, frequencies and actors change. And this information closely depends on the model and manufacturer.

In France, it is the recently created (2018) "Direction de la Maintenance Aéronautique (DMAé)" the one in charge to plan and organize the checks for military aircraft. These checks have the following types for the Mirage 2000 series:

Aircraft	'A' Check	'B' Check	'C' Check	'D' Check
B737-300	275 FH	825 FH	18 M	48 M
B737-400	275 FH	825 FH	18 M	48 M
B737-500	275 FH	825 FH	18 M	48 M
B737-800	500 FH	n/a	4000-6000 FH	96-144 M
B757-200	500-600 FH	n/a	18 M / 6000 FH / 3000 FC	72 M
B767-300ER	600 FH	n/a	18 M / 6000 FH	72 M
B747-400	600 FH	n/a	18 M / 7500 FH	72 M
A319	600 FH	n/a	18-20 M / 6000 FH / 3000 FC	72 M
A320	600 FH	n/a	18-20 M / 6000 FH / 3000 FC	72 M
A321	600 FH	n/a	18-20 M / 6000 FH / 3000 FC	72 M
ATR42-300	300-500 FH	n/a	3000-4000 FH	96 M
ATR72-200	300-500 FH	n/a	3000-4000 FH	96 M

Table 1.1: Frequency of checks per aircraft and type (FH= flight hours, FC= flight cycles, M=months) from Cook, A.J. and Tanner, G. [52]

VG Called "Visite de graissage" or Oil Check, it is carried out every 6 to 8 calendar months. It takes about 3 days.

VI Called "Visite intermédiaire" or Intermediate Check, it is carried out every 14 to 16 calendar months. It takes about 6 days.

VS Called "Visite de securité" or Security Check, it is carried out every 300 to 600 flight hours. It takes about 4 days.

VX Called "Grande visite" or Overhaul Check, it is carried out every 1000 to 1200 flight hours or 60 months, whichever arrives first. It takes between 4 and 6 months. These checks are equivalent to the D-checks.

The first three checks are done in the aircraft's airbase. The overhaul checks (VX) are done in the "Atelier Industriel de l'Aéronautique (AIA)", located in the Clermont-Ferrand region. The latter can be seen in Figure 1.1.

1.1.3 Planning maintenances

The Flight and Maintenance Planning (FMP) problem, first presented by Barnhart et al. [26], studies how these maintenance operations are scheduled and how flight activities are assigned to a fleet of aircraft along a planning horizon.

In the civil variant of the FMP [26, 161], by far the one that has received the most attention, aircraft need to be routed along different destinations by assigning them flight legs

Figure 1.1: A photo of the AIA maintenance workshop in the Clermont-Ferrand where the VX checks are done for the Mirage 2000[1].

in order to build daily trips. Checks are done during the night and are usually limited to A and B type checks. The planning horizon covers several days (e.g. Sriram and Haghani [161] use 7 days). The objective is usually to reduce the overall cost or to maximize total profit for the plan.

In the military variant, which we call the Military Flight and Maintenance Planning (MFMP) problem, flights (which are called missions) always return to the airbase and so no routing is performed. Checks consist of B, C and D types. The planning horizon covers a few months or several years. The objective is usually to maximize the operational status of the fleet while minimizing the total number of checks.

In most air forces, a derivative of the Sliding Scale Scheduling Method (SSSM) [89] is used to plan the mission assignments for aircraft. This method consists of a simple heuristic that attempts to distribute remaining flight hours among aircraft in a ladder-like distribution, i.e., there is a constant probability of finding an aircraft for each possible amount of remaining flight hours. When an aircraft passes certain threshold of flight hours, it is sent for a check. Figure 1.2 shows an example of using this technique. In the example, Aircraft "944" has the most surplus remaining flight hours (bank time) with respect to the curve and thus should fly the next mission.

This technique attempts to distribute the flight load uniformly between each aircraft and is well suited when the future mission requirements are known and do not change over time, the fleet is homogeneous and the initial status is more or less well distributed. As can be expected, these hypothesis rarely apply in real life and so more sophisticated techniques are needed if good quality maintenance plans are to be obtained for realistic situations.

Recently, improving the quality of these maintenance plans has become a priority for many

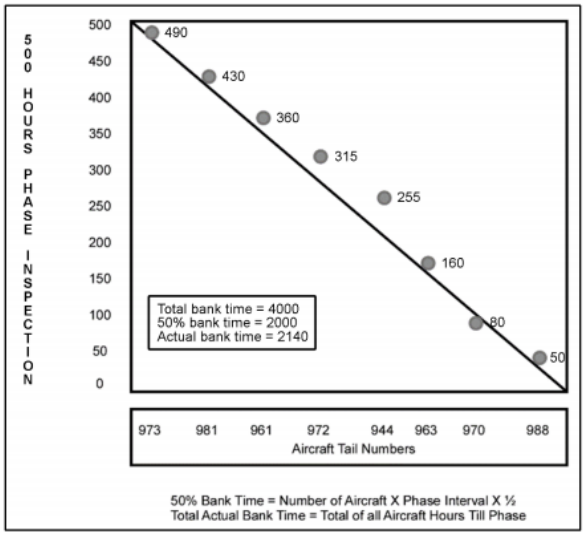

Figure 1.2: An example showing the SSSM for 8 aircraft in a 500-flight-hour cycle.

governments [116]. In particular, the French Air Force became interested in mathematical solutions to schedule maintenance for the Mirage 2000 fleet [4] after it was revealed that increases in maintenance costs had not been followed by improved availability of aircraft [113].

There are two main questions that need to be answered when improving maintenance decisions: when is the maintenance needed (prediction) and when is the maintenance actually done (scheduling).

Calculating maintenance needs has usually consisted of what is called preventive maintenance. In preventive maintenance, a simple rule is established that ties usage and maintenance need, e.g., an aircraft cannot fly more than 500 hours without a maintenance. In contrast, predictive maintenance involves the analysis of historical data to estimate windows of time when maintenance has to be done so as to guarantee a risk of failure under a certain threshold. According to the US Air-Force tests on command-and-control planes, the use of predictive maintenance could reduce unscheduled work by a third [8].

Regardless of how we calculate those maintenance needs, a suitable maintenance schedule is essential in order to produce an efficient overall plan. To achieve this, those time windows are taken together with resource capacities and future usage planning, among other information, in order to produce an actual maintenance schedule that is feasible and satisfies the needs of its planners.

The difference between any maintenance plan and a good one is important. As the resources needed to actually perform the checks are expensive and the expertise required is highly specialized, the capacity to provide this service is usually quite limited and inflexible. As a result, a plan that does not allocate the capacity optimally may end with more grounded aircraft waiting for a check in some periods while in other periods the capacity usage is not

at maximum. This unavailability reduces the actual potential of the fleet and thus requires having even more aircraft to satisfy a certain fleet potential.

The context of long term planning is inevitably tied to uncertainties in mission requirements, fleet availability and maintenance capacity restrictions, .e.g., a new mission, additional aircraft for an existing mission, an unexpected grounding of an active aircraft, a partial reduction in the capacity of the maintenance workshop, etc. These uncertainties require robust plans where each period is guaranteed to have enough aircraft in good status to not only comply with planed missions but also with changes in a set of potentially heterogeneous missions. A plan that does not take this into account explicitly, risks a scenario where there are not enough aircraft ready for an unforeseen emergency.

The actual cost of a maintenance plan is measured by the number of checks needed to comply with all the mission requirements: the lower that number is, the better. Although costs for these checks are not public in the case of military aircraft, the actual costs for type D checks in Boeing aircraft range between 1 million dollars and 6 million dollars [3] depending on the aircraft model. In France, the total yearly cost of military aircraft maintenances rose from 3.2 billion euro in 2012 to 4 billion euro in 2017 [113]. A good maintenance plan will try to use each aircraft as much as possible before scheduling a check in order to minimize the number of remaining flight hours that is "lost" when the check is done and flight hours are reset to their maximum.

Finally, the planning of maintenances is usually done manually (in an Excel spreadsheet) and is time consuming. This limits the number of scenarios that can be evaluated before taking a decision and increases the reaction time when dealing with new information that requires to change the original plan. As a result, robustness and flexibility are lost as part of the planning process.

Thus, the need for efficient and reliable flight and maintenance planning is clearly a priority both in commercial as in military contexts. Nevertheless, producing such a detailed plan for a sizable fleet while planning for a long-term horizon and taking into account the multiple objectives inherent to long-term planning is not an easy task.

Operations Research techniques are specialized in solving combinatorial problems such as the MFMP in order to produce very good quality solutions (often optimal) given a set of constraints and an objective function. The success in applying these techniques to a new, difficult, problem requires a combination of domain expertise and profound theoretic background. The latter consists of a deep study on the structure of the underlying mathematical problem and the research of innovative solution approaches that are compatible with that problem. This thesis provides such analysis.

1.2 Problem description

In this section we describe the main characteristics of the MFMP problem studied in this thesis and provide a small example. A formal definition of the problem is presented in Chapter 3.

Aircraft. Aircraft are identified by index i. Each aircraft starts the planning horizon with an initial status with respect to its maintenance needs. This status consists of the number of hours that it can fly before needing a check Rft_i^{Init} and the number of calendar periods before needing a check Rct_i^{Init}. In addition, each aircraft has one type and an optional set of capabilities, that allow it to perform missions.

Missions. Missions are identified by index j. Each mission has a start $Start_j$ and end End_j date and requires a certain amount of aircraft R_j to be assigned when it is active. When an aircraft is assigned to a mission, it is required to stay assigned for at least a certain amount of time periods MT_j^{min}. As long as these rules are taken into account, an aircraft can be assigned for less than the total length of the mission. Each mission has a type, which needs to be the same as the aircraft that are assigned to it, and may require certain additional capabilities for these aircraft. Each aircraft that is assigned to a mission needs to fly a certain amount of hours H_j each period. When an aircraft is not flying a mission and is not in maintenance, it still flies a certain amount of default hours per period U^{min}.

Maintenances. When an aircraft has reached its limit of flight hours H^{max} or its limit in calendar periods E^{max}, it requires a check. Each check takes a certain amount M of periods to be performed. There is a limit to the total number of checks that can be performed simultaneously, which corresponds to the workshop capacity C^{max}.

Fleet status. The missions are classified into clusters where each mission has the same type and capabilities. Each mission belongs to one cluster but an aircraft can belong to many clusters. Each of these clusters needs to have a minimum number of total remaining flight hours H_{kt}^{Clust} and available aircraft (not in maintenance) A_{kt}^{Clust} in order to guarantee the good status of the fleet and its resilience.

Objectives. A good flight and maintenance plan complies with all previous rules by scheduling the checks as late as possible in the planning horizon and avoiding unnecessary checks. In real cases, some of the previously defined rules will need to be violated and so these violations need to be minimized too. Finally, the frequency of checks should not vary too much between aircraft of the same type.

To better illustrate the problem, a small example follows. Take a fleet of 5 aircraft and 6 consecutive missions over a planning horizon of 20 periods. $M = 2$, $H^{max} = 300$, $E^{max} = 15$,

j	MT_j^{min}	$Start_j$	End_j	H_j	R_j
0	2	1	4	24	1
1	2	5	7	34	3
2	3	8	11	18	3
3	3	12	15	30	3
4	2	16	18	35	3
5	2	19	20	25	1

Table 1.2: Input data for each mission $j \in \mathcal{J}$ in the small example.

i	Rct_i^{Init}	Rft_i^{Init}
0	7	120
1	13	220
2	7	140
3	8	140
4	6	160

Table 1.3: Remaining flight time and remaining calendar time for each aircraft $i \in \mathcal{I}$ in the small example.

$C^{max} = 1$, $U^{min} = 0$. The first mission ($j = 0$) can only use aircraft 3 or 4. All the other missions can use any of the 5 aircraft. Two clusters are thus formed: the first contains mission 0 and has as candidates aircraft 3 and 4, the second contains missions 1-5 and has as candidates all five aircraft. $H_{0t}^{Clust} = 300$, $H_{1t}^{Clust} = 750$, $A_{0t}^{Clust} = 1$, $A_{1t}^{Clust} = 4$.

Table 1.2 shows the input data for the 6 missions and Table 1.3 shows the input data for the 5 aircraft.

Figure 1.3 shows the optimal solution for the problem. Aircrafts are shown in rows, periods in columns. Checks are in gray. Mission assignments share the color of the mission and show the total number of hours flown.

1.3 Thesis structure

Chapter 2 presents a review on the FMP, the MFMP and related problems from the literature in order to compare and position the problem described above with respect to previous work. Furthermore, a comprehensive study on the techniques employed to solve these problems is presented and a categorization of problems and solution methods for the MFMP is provided.

Chapter 3 formalizes the MFMP problem, studies its structure and solves it with exact methods. First, a complete instance of the MFMP problem is described and a random

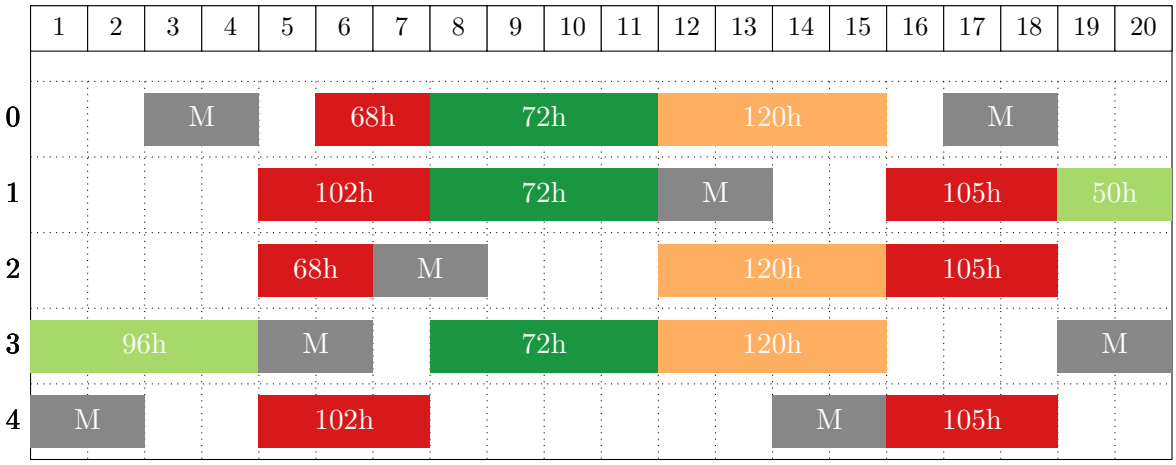

Figure 1.3: Gantt showing the optimal solution for the example.

instance generator is produced. This generator will be used to produce instances for the rest of the thesis. The structure of the long term MFMP problem is analyzed and, as a result, its complexity is studied. In order to solve the problem, a tight Mixed Integer Programming (MIP) formulation is proposed and complemented by a simulated annealing heuristic that generates initial feasible solutions. The sensibility of the model is inspected with regards to changes in size and various characteristics of instances.

Chapter 4 formulates a new MIP model and applies Machine Learning (ML) methods to increase the solution performance. The alternative MIP model consists of a long formulation based on explicitly enumerating all check patterns for each aircraft. This model is then expanded by the addition of valid cuts, based on the instance input data, and invalid cuts, based on the prediction of optimal or near-optimal solutions. These predictions are done by a supervised learning model that matches the input data for an instance with certain characteristics of its solution and is trained with a dataset of thousands of small instances solved to optimality or near-optimality. The performance of this approach is compared with previous methods as well as with other more conventional matheuristics.

Chapter 5 deals with methods to solve very large scale problems of the MFMP using a Variable Neighborhood Descent with two complementary neighborhoods. The first one is an Rolling Horizon solved using the MIP model from Chapter 4. For the second neighborhood, a complete graph of all possible states for each aircraft is produced. This graph is an extension of the pattern formulation in Chapter 4 were each node represents the state of an aircraft with respect to the need of a check. The neighborhood is then solved by solving a Shortest Path Problem.

Finally, Chapter 6 summarizes the main conclusions and sheds light on directions for future research.

Chapter 2

State of the Art

Contents

In this chapter, we present a review of the work that has been done in the MFMP and its related problems up to now. The Flight and Maintenance Problem (FMP) and its variants are presented in Section 2.1; Section 2.2 describes several problems with the same structure such as the Nurse Rostering problem, the Rolling Stock Assignment and Maintenance Planning problem, and the Maintenance Scheduling and Production Planning of Nuclear Plants. Section 2.3 presents an exhaustive review of the techniques that have been successfully utilized to solve these problems including exact ones such as MIP and CP; and heuristic ones such as matheuristic, metaheuristic and hybrid methods. Finally, Section 2.4 presents recent approaches using machine learning in combination with mathematical programming.

2.1 The Flight and Maintenance Planing problem

2.1.1 Civil Aviation

The Flight and Maintenance Planning (FMP) problem, first introduced in Barnhart et al. [26], consists of the combination of two problems: the Tail Assignment (TA) problem and

the maintenance scheduling. The former, first studied in [68], consists of assigning a series of already scheduled flights (often named legs) to a fleet of aircraft (also referred as tails) and has been widely studied [50, 81, 67, 83, 95]. The latter consists of the scheduling of maintenance operations (which we call checks) to aircraft in order to comply with calendar-based and usage-based limits mandated by safety regulation. This problem is also known as the Airline fleet Maintenance Scheduling (AMS) [60, 148]. The resulting FMP problem is also known as AMS with Tail Assignment [161, 148] or Aircraft Maintenance Planning (AMP) [154].

With regards to the checks in the airline industry, there exist four major types [161]: A (every 65 flight hours or 1 week), B (every 300-600 flight hours), C and D (every 1-4 years). As a result of the short planning horizons that are used in FMP problems, only the two first types of checks are taken into account. The other two types are sometimes taken as constraints: these checks are already decided for some aircraft and this unavailability needs to be taken into account during the planning process.

From the numerous contributions to solve the FMP problem, only a few take into account the legal required limit on the number of flight hours between checks [124, 150, 95, 148].

Murat Afsar et al. [124] present a 10-week FMP problem where a fleet of 200 identical aircraft is assigned flights and maintenances. Flights are organized in a mono hub and spoke structure and only the hub city has the capacity to do any check. Checks are assumed to be done during the night so the aircraft are ready in the morning of the following day. The maintenance capacity is modeled by the total number of aircraft in maintenance at any given night and depends on the instance but is always between two and three aircraft per night. Flights per week range between 2500 and 3100 2-way flights for 91 destinations. The initial status of the fleet is randomly generated and the solution method applies a weekly heuristic inside a rolling-horizon decomposition. The heuristic is based on modeling each aircraft as a graph where each node is a possible flight assignment, the source is the initial status and the sink is the assignment of a check. They then apply a longest path algorithm for each aircraft in a specific order, then slightly modify the assignments and do swaps between aircraft to improve the solution.

Sarac et al. [150] deal with a short-term FMP problem where a set of flights are assigned to a fleet of aircraft while scheduling checks, respecting the flight hour regulation and considering the maintenance constraints. Checks are assumed to be done during the night. Several types of checks can be done in one of several overnight stations if that station is also a maintenance station. Maintenance capacity is controlled by resources and slots: each check consumes a certain amount of man-hours and a slot from the maintenance station where it is done. Each maintenance station can only do certain types of checks. The objective is to minimize the total number of unused remaining flight hours. In order to solve the problem, a network is generated were each node i represents a flight leg and each arc (i, j) represents a feasible sequence of flights $i \rightarrow j$ (i.e. the destination of i is the same as the origin of j and the arrival time for i plus the turn time is less than or equal to the departure time of flight leg j). A string-formulation model is proposed, where routes are assigned to aircraft and a Column Generation technique is used to generate feasible routes. The computations experiments include a fleet of 32 aircraft, 175 flights, 5 maintenance stations and a planning horizon of 1

day.

Khaled et al. [95] present the Tail Assignment problem and solves it with a multi-commodity flow formulation. Then, the problem is adapted to schedule checks. Only type A checks are taken into account and these checks are assumed to be done during the night after the last flight of the day. Each airport is equipped with a maintenance workshop that has a capacity that varies according to each day. All checks are assumed to have the same calendar and flight hour limit. The original objective function based on minimizing cost is modified by adding a cost per each check depending on the airport, the aircraft and the day. Computational tests are done on instances of 7-30 days, 10-40 aircraft and with a check frequency of 4-5 days and 64-72 flight hours.

Sanchez et al. [148] show two problems related to maintenance scheduling: the first schedules checks with a given flight planning taken as input. The objective is to minimize the number of violations of maintenance regulations while maximizing fleet sustainability at the last period. The scheduling includes multiple check types in multiple maintenance workshops. The maintenance capacity is measured as a consumption of resources at each period. Checks have a fix duration for a given type and aircraft. The second problem is a variant of the first that in addition to scheduling checks also reassigns part of the already scheduled flights in order to reduce violations of maintenance regulation. Here 4 new objectives are added: minimize the maximum resource consumption, the number of flight reassignments, the total number of checks, and the total resource usage. Computational tests were done over a 30-day planning horizon with 16.000 flights, 529 aircraft and 8 maintenance workshops.

Repair and recovery problems involve the re-optimization of a previously solved solution (incumbent) under the light of a relatively small change in the original input data. This change is usually the result of an unforeseen event: a delay, an accident that affects the provision of the service or an external event that causes a change in demand. More specifically, the change can be (1) an additional constraint, (2) the change of some coefficients on the constraint matrix or (3) the change of some coefficients on the objective function. Usually, it is the combination of all three. As a consequence, the incumbent solution may no longer be valid (feasible) or may no longer be optimal (or, even, good). The main hypothesis of a repair / recovery solution method is that a good candidate solution should not be too far from the incumbent, given the small proportion of change in the input data. Repair and recovery methods for the FMP have been proposed in [70, 97, 178, 94, 91, 148].

2.1.2 Military Aviation

The Military Flight and Maintenance Planning (MFMP) problem was first presented in Sgaslik [153] and the main differences with the FMP are the use of missions instead of flight legs. Aircraft are assigned to missions for an interval of time periods. Since all aircraft return to base after each assignment, the location of the aircraft (airport) is not taken into account. These problems also require additional availability and sustainability constraints to guarantee a good state for the fleet at each period. In contrast to FMP problems plan-

ning A and B checks, most contributions in the MFMP are centered in planning B checks [152, 174, 98, 85, 136] and D checks [55].

Sgaslik [153] presents a MFMP where a fleet of 45 helicopters are planed along a planning horizon of 12 periods of one month each. Missions are assigned to each aircraft in order to comply with planned flight hours. Checks provide additional flight potential and must respect the maintenance capacity. The fleet needs to meet a monthly desired number of total remaining flight hours, guaranteeing the good status of the fleet. The objectives penalize penalties from soft constraints. The problem is solved by using a two-model setup. First, a medium term problem determines D-type checks and assigns continuous flight hours. Later, this solution is fed into a short-term model where planned missions are assigned to each aircraft in a heterogeneous fleet.

Kozanidis [98] presents a problem where a wing of 24 aircraft is organized into 3 squadrons of 8. The planning horizon consisted on 6 periods of 1 month each. Each squadron requires certain flight hours per period and each aircraft has a range of hours they can fly per period. There is a demand of flight hours over the whole planning horizon. Finally, the capacity of the maintenance workshop is measured in available working hours per period. The decision is the continuous number of hours to fly each aircraft in each period and when to put them into maintenance. The link between these decisions and constraints is modeled via a resource flow balance: the first decision "consumes" the aircraft remaining flight hours and the latter "consumes" the workshop maintenance hours. Additional accounting needs to take place to guarantee that the hours are consumed and updated after each period. All four objectives were targeted at maximizing the minimum of certain KPI over all planning periods, thus improving the worst performing period in the planning horizon. Those KPI were: (1) the total number of available aircraft, (2) the number of available aircraft for the worst squadron-period, (3) the total remaining flight time, and (4) the remaining flight time for the worst squadron-period. Weights are finally given to each objective in order to combine them in one objective function. It also presents two simple heuristics that could be used for larger instances: the first is an implementation of the "Sliding Scale Scheduling Method" [89], the second is a rolling horizon approach. We will call this formulation the Check Flow Balance (CFB) formulation because of the way the maintenance workshop resource is "consumed" by each aircraft in maintenance. This CFB formulation has been the base for many future contributions, such as [99, 100, 77, 78, 174, 116, 152]. An efficient solution method for a particular case (maximizing overall sustainability) of this problem was presented in [77] and expanded in [78] to deal with the multi-objective version of the problem where overall sustainability is maximized at the same time as its variability is minimized.

Marlow and Dell [116] solve a short term problem where a squadron of up to 24 aircraft is planned in a planning horizon of up to 30 periods of one day each. Each aircraft is assigned a number of flight hours each day in order to meet an overall demand per period and for the whole planning horizon. Two types of checks are included, one is calendar based, the other is flight hour based. The capacity of the maintenance workshop is modeled with a CFB formulation. Several objectives are modeled as soft constraints: compliance with the total and daily demand of flight hours, the correct distribution of remaining flight hours for the

fleet and the compliance of flight hours between checks. Piecewise linear penalties are used as weights for these objectives.

Seif and Yu [152] propose a generalized version of the problem in [77]. The changes consist of including several check types, each with a particular flight hour frequency; several maintenance workshops, each with a capacity to do checks and specific check types it can do; the demand for flight hours is given for each aircraft type. The objective stays the same: maximize the effective remaining flight time and the procedure to solve the problem is an adaptation of the one in [77].

Cho [48] presents a problem where missions and checks are scheduled for a fleet of 15 aircraft over a planning horizon of 520 half-day periods (two years without weekends). In contrast with CFB, missions (also called sorties) and checks are modeled in discrete assignments. Two types of missions were used, each requiring a different amount of flight hours. Checks are modeled via assignments of a fixed duration of periods (20, representing two weeks). Remaining flight hours are still controlled by a flow balance. Another contribution is the fact of assigning a discrete state for each aircraft at the end of the planning horizon, in order to guarantee the correct cycle-like status of the fleet. The objective is to minimize the maximum number of aircraft under maintenance over all periods. We call this formulation the Discrete Check and Mission (DCM) formulation. The DCM formulation has been the base for other more recent contributions, such as [108, 154].

Shah et al. [154] add two variants to the DCM formulation: (1) missions are relaxed into continuous flight hours, just like in [98] and (2) the objective is the minimization of the total number of checks in the whole planning period. The maintenance capacity is modeled as a constraint. The plan is a fleet of 7 aircraft, during 52 periods. Li et al. [108] incorporated the use of random simulated initial states for aircraft and random simulated durations for checks. The values followed uniform distributions and were known in advance of the solution process.

Hahn and Newman [85] presents a problem where checks and missions (deployments) are scheduled for a fleet of 10 helicopters over a planning horizon of 12 weekly periods. The model uses mostly a DCM formulation where aircraft are deployed into missions where they fly a fixed amount of hours per period. In addition to these considerations, aircraft that are not deployed are considered “in base”, where they fly a variable but bounded amount of hours. The base has a range for the total flight hours per period and each aircraft has a range of flight hours for the planning horizon. Checks, which are done in base, can only be done when the amount of remaining flight hours reaches some value close to zero. The number of concurrent checks is limited by the base capacity. Initial conditions regarding current deployment, maintenance status and remaining flight hours are taken into account. End conditions are also taken into account to guarantee continuity. The objective is to penalize not reaching flight hours goals in the base and changing location too often without the need for a check. Winata [177] uses this same benchmark problem with slight variations (no “in base” flying) and a larger fleet (10 - 40 aircraft).

As can be interpreted from the use of less frequent checks, the MFMP deals with longer terms than the FMP. Still, we can differentiate between be short-, medium- and long-term

planning horizons. The short term has a time horizon of at most 1 year and is usually divided into periods of one day [116, 48, 146].

Medium term planning is concerned with a weekly or monthly schedule over 6 months to 2 years [152, 174, 98, 85, 136]. Here, checks are assigned every 200 to 400 flight hours, which correspond to type B and C checks. The capacity for these check types is seen as the number of available man-hours at each period of time.

Long term planning covers time horizons between five and ten years and mostly addresses scheduling of D checks [55]. These operations are particular in that they last several months. Checks are scheduled every 1000 - 1200 flight hours or at most 5 years after the last overhaul maintenance.

A common hypothesis considers an homogeneous fleet, i.e. each aircraft being capable of performing any of the existing missions. One exception is where an heterogeneous fleet is used is Seif and Yu [152].

One can easily imagine that long term military operations are subject to uncertainty regarding missions, destinations and flight hours. However, contributions incorporating uncertain parameters are quite rare. One of the first attempts to take into account the stochastic nature of maintenance requirements and durations was presented by Mattila et al. [119], where a simulation model was built in order to find good maintenance policies. Kessler [93] developed a model based on a multi-armed bandit superprocess to choose between two different heuristics or policies in order to maximize the availability of the fleet.

The planning of missions and maintenance for military aircraft shares some similarities with the planning of the procurement and retirement of said aircraft [125, 75]. For example, the decision on how to distribute the remaining flight hours among the fleet by choosing a good policy of mission assignment under a certain budget.

We present a classification of existing MFMP formulations by grouping features in three categories: maintenance, mission and aircraft-related. These features consist represent differences in objective functions and constraints.

Maintenance related features

FP Flight potential: frequency of checks constrained by flight hours.

CP Calendar potential: frequency of checks constrained by calendar periods.

FD Fixed duration: each check has constant duration that depends on the type of check.

CA Capacity: maximum number of aircraft undergoing a check in any given period.

MS Multiple stations: each maintenance station has its own capacity and serves a subset of aircraft or a subset of check types.

RC Relaxed capacity: the maintenance capacity is measured in man-hours and each check requires a certain amount. If a check is not finished at the end of a period, the remaining man-hours can be covered in the next period.

MT Multiple types: aircraft may have more than one type of check. Each type has a frequency, duration and capacity usage.

Mission related features

DA Discrete assignment: missions require a certain number of aircraft to be assigned each period. These aircraft need to fly a fixed amount of flight hours.

HD Hour demand: a demand of flight hours is required in each period per group of aircraft.

HF Heterogeneous fleet: only compatible aircraft can be assigned to each mission.

MD Min duration: if an aircraft is assigned to a mission, there is a minimum amount of calendar periods it has to remain assigned to this mission.

HT Hours in total: the total number of flight hours in the horizon needs to fall within a given range. Sometimes, each aircraft or group of aircraft has its own range.

Aircraft related features

IS Initial state: mission assignments and checks that start before the beginning of the planning horizon are fixed.

AV Availability: the total amount of checks per group of aircraft is limited:

(a) at all periods, minimize the sum.
(b) at each period during the planning horizon, upper bound.
(c) at all periods, minimize the maximum.

SU Sustainability: limit the amount of remaining flight hours per group of aircraft:

(a) at the last period of the planning horizon, lower bound.
(b) at some periods, lower bound.
(c) at each period during the planning horizon, lower bound.
(d) at all periods, maximize the minimum.
(e) at all periods, maximize the sum.
(f) at all periods, minimize the variance.

Table 2.1 shows the problem formulations used in the previous work and available information about solved instances of the problem.

Table 2.2 shows the constraints implemented in existing problem formulations for the MFMP.

Reference	$\|\mathcal{I}\|$	$\|\mathcal{T}\|$	Unit	H^{max}	Method
Kozanidis [98]	24	6	m	300	MIP
Hahn and Newman [85]	10	12	w	200	MIP
Winata [177]	40	12	w	200	MIP, VNS, TS, SA
Cho [48]	15	520	0.5 d	300	MIP
Verhoeff et al. [174]	20	52	w	400	MIP
Li et al. [108]	20	50	0.5 d	200	MIP
Gavranis and Kozanidis [78]	50-100	6	m	300	MIP
Marlow and Dell [116]	12-24	30	d	200	MIP
Shah et al. [154]	7	52	m	25	MIP
Seif and Yu [152]	100	6	m	125-500	MIP
Chapters 3 and 4	15-60	90	m	800-1200	MIP
Chapter 5	45-120	90	m	800-1200	MIP, DP

Table 2.1: Previous work: solved instances. Units: m=month, d=day, w=week. $|\mathcal{I}|$=fleet's size, $|\mathcal{T}|$=horizon's length, H^{max}=check flight-hour frequency.

Reference	Maintenance							Missions					Fleet		
	FP	**CP**	**FD**	**CA**	**MS**	**RC**	**MT**	**DA**	**HD**	**HF**	**MD**	**HT**	**IS**	**AV**	**SU**
Kozanidis [98]	C			C		C			C				C	O_c	O_d
Hahn and Newman [85]	C		C	C				C	C		O	C	C		C_a
Winata [177]	C		C	C				C	C		O		C		
Cho [48]	C		C	O				C					C		C_a
Verhoeff et al. [174]	C			C		C			C			C	C	C_b	O_d
Li et al. [108]	C		C	O				C					C		
Shah et al. [154]	C		C	C					C				C	O_a	C_a
Gavranis and Kozanidis [78]	C			C		C			C				C		O_e,O_f
Marlow and Dell [116]	O			C		C			C			O	C		O_b
Seif and Yu [152]	C			C	C	C	C		C	C			C		O_e
This thesis	C	C	C	O				C		C	C		C	O_a,O_b	O_c

Table 2.2: Constraints taken into account in the existing formulations: “O” means an objective or a soft constraint, “C” means a hard constraint.

As can be seen from this comparison, the majority of existing formulations were developed for an homogeneous fleet that needs to comply with general flight-hour demands under flight-hour constraints to control checks and maintenance capacity constraints (flexible or not).

2.2 Related problems

We understand related problems as problems that have a similar structure and thus may be solved with similar techniques, regardless of the end application. In the case of the MFMP, this structure involves the presence of a set of resources, a set of planned tasks, and the existence of maintenance operations (or resting periods) linked to the usage of said resources.

This section presents three related problems that share many similarities with the MFMP. Section 2.2.1 presents the Nurse Rostering Problem (NRP). Section 2.2.2 describes the Rolling Stock Assignment and Maintenance Planning (RSAMP) problem. Finally, Section 2.2.3 discusses the Maintenance Scheduling and Production Planning of Nuclear Plants (MSPPNP).

2.2.1 The Nurse Rostering problem

The Nurse Rostering Problem or Personnel Scheduling Problem decides shifts for a set of workers over a sequence of consecutive periods along a planning horizon. The result of this planning process is called a roster. This roster consists of the assignment of one shift (including a rest shift) to each worker (or nurse) on each day. Three types are enumerated by Baker [23]: shift scheduling (time-of-day scheduling), days off scheduling (day-of-week scheduling) and tour scheduling (a combination of both). The MFMP problem fits well with the second category: daily shifts (missions) are scheduled for an heterogeneous workforce (aircraft fleet) whose working periodicity (check frequency) does not match the operating frequency (planning horizon or missions).

Several surveys of the NRP have been done. For classification of the different problems according to the types of constraints imposed, see [173, 54]. De Causmaecker and Vanden Berghe [54] presented a naming convention following the $\alpha|\beta|\gamma$ convention in the scheduling domain. For the state of the art on solving techniques, see [47, 41]. More recently, [170] provides an up-to-date summary of the contributions to the problem according to the technique used.

As the literature shows, the NRP is more a large family of problems that share certain characteristics than a single problem. This has not impeded the study of the structure and the complexity of each of its variants. Some examples of these studies can be found in [105, 45, 38, 39, 156, 59, 172].

Caprara et al. [45] presents a staff scheduling problem where a set of workers must comply with a set of duties, each duty requires a certain amount of workers on each day. Workers need to have assigned a rest block. There are several rest types, each with a different duration in

number periods. Each worker needs a certain amount of rests of each type during the planning horizon. The complexity is proven to be NP-Hard via the Three Partitioning Problem.

In Brunner et al. [39], the flexible days-on and days-off scheduling problem is shown. This problem deals with the assignment of a minimum and maximum number of consecutive working (days on) and non-working days (days off) to each worker over a planning horizon while guaranteeing a total number of workers per day. The problem is proven NP-complete by reducing it to the the circulant problem.

Smet [156] presents many cases of the NRP, some of which can be solved in polynomial time. In particular, a cost-minimization problem $\mathcal{P}_2$ is presented where each nurse is required a constant number of days to work, a range of nurses is required at each day per shift type, and a set of unavailable days for each nurse is taken into account. The objective is to minimize the sum of costs for assigning each shift to each nurse on each day. The problem is proven to be solvable by a minimum cost network flow problem. Smet [156] also presents the shift minimization personnel task scheduling problem (SMPTSP), where an heterogeneous workforce is assigned a set of scheduled tasks with a given start time and end time. The objective is to minimize the required number of workers needed to perform all the tasks.

2.2.2 The Rolling Stock Assignment and Maintenance Planning problem

A problem that is closely related to the FMP in nature and structure but is less studied is the Rolling Stock Assignment and Maintenance Plan (RSAMP) problem, applied to the railroad industry to plan the maintenances of trains units, also called rolling stock. Here, a timetable of predefined tasks is taken as input data, each task being a trip between two stations. The problem consists of assigning tasks to rolling stock (the train units that will actually do the trip) while also scheduling the checks operations with their respective frequencies. Sometimes, the solutions need to take into account complex shunting operations inside the stations to guarantee feasibility.

According to Lai et al. [102], trains in Taiwan Railways Administration regular inspections can generally be divided into four levels: daily inspection (every 1800 km or 3 days), monthly inspection (every 90,000 km or 3 months), bogie inspection (every 500,000-1,000,000 km or 1.5-3 years), and general inspection (2,000,000-4,000,000 km or 6-9 years). The former two types are done in the train depot while the two latter are done in a dedicated workshop.

In Doganay and Bohlin [61], checks are scheduled over a 2 year horizon in weekly periods by taking into account the availability of spare parts by minimizing the total number of checks, storage of spare parts and the time the trains are in maintenance. Each check has a different frequency and use of spare parts. In Lai et al. [102], daily and monthly checks are scheduled in an heterogeneous fleet of rolling stock. In Lin et al. [109] preventive checks are planned every 1.5 years or 600,000 km for a high-speed train network are decided over a one year horizon with a simulated annealing technique. The number of checks at each period is limited to capacity and to availability objectives at each period (because of seasonality).

Maróti and Kroon [117] show the re-routing of an existing rolling stock schedule in order for "urgent" train units, that require a maintenance inside the planning horizon, be routed in time to a maintenance facility. The original rolling stock schedule includes a sequence of tasks for each train unit. Each urgent train unit has a set of maintenances that it needs to accomplish during the new plan and each maintenance requires an additional "buffer time" in order to take place. Maintenance capacity is not explicitly incorporated but there are variable costs associated with the actual changes to the plan, e.g. the time and station of these changes, the distance between arrival and departure tracks, the time between tasks, etc. The planning horizon is around three days.

Tréfond et al. [168] proposes robust plans for the rolling stock routing problem by assigning sequences of tasks to train units while scheduling maintenance operations after some of the tasks with calendar-based constraints. A maintenance task can be fitted in between two consecutive assignments, if enough time exists from the end of the first task and the start of the second one to cover its duration. The number of maintenances that each train has to do is known and the maintenances do not depend on the usage of the train unit. Mira et al. [121] expands on this work by including a constraint on maintenance capacity measured in man-hours that is consumed by each maintenance task's work load.

2.2.3 The Maintenance Scheduling and Production Planning of Nuclear Plants

The 2010 ROADEF/EURO Challenge [137] consisted on the production planning and maintenance scheduling of nuclear plants. It was sponsored by EDF, France's electricity company.

The MSPPNP problem consists of deciding when to stop each nuclear plant for refueling and maintenance operations, as well as its production plan in order to satisfy a series of constraints. In relation to the MFMP problem, each nuclear plant can be seen as an aircraft, the electricity demand can be viewed as the missions' needs and the production plan like the mission assignments.

Checks behave similarly as in the MFMP. They replenish the production potential, and thus define production cycles. During the check, the unit becomes unavailable. Each check lasts a fixed amount of time and there is a limit on the amount of simultaneous checks that can be carried. One difference is that in the MSPPNP there are additional decisions to take during the check, such as the quantity of refueling for the plant.

The production plan needs to satisfy technical limitations on capacity and changes on the quantity of production. Just as with the heterogeneous fleet of aircraft, there are two types of nuclear plants and each type has specific constraints.

There are many sources of uncertainty, mainly related to the demand of electricity: the price, the demand and the quantities that can be supplied. This is why a multi-scenario stochastic problem is formulated.

The instances used in this challenge have a planning horizon of 260 weeks (5 years), with 7 to 21 time-steps per week. The number of power plants rises up to 100 or 70, depending on the type. In order to model the uncertainty of the input data, up to 120 scenarios are presented. Several (less than six) checks are conceived for each power plant in this planning horizon.

2.3 Solution approaches for the FMP and related problems

This section is organized as follows. Section 2.3.1 presents Mathematical Programming (MP) approaches, including MIP models, graph algorithms, and (heuristic) decompositions based on MIP or LP models: rolling horizon, column generation among others. Section 2.3.2 lists Constraint Programming (CP) implementations applied to relevant problems. Section 2.3.3 shows relevant metaheuristics (MH) methods. Finally, Section 2.3.4 presents the hybrid implementations where two or more of the previous three groups are combined.

2.3.1 Mathematical programming

MP-based approaches have been by far the most common techniques used to solve the MFMP and similar problems. Most of MP approaches are based on modeling the problem as a linear programming model consisting on a set of continuous and binary variables related by linear equations (named constraints) which constitute the valid solution space of any solution. Finally, a linear objective function on the set of variables is used to distinguish the quality of each solution and be able to determine the best one. These models are called Linear Programs (LP) if all variables are real, Integer Programs (IP) if all variables are binary or Mixed Integer Programs (MIP) in case a combination of real and binary variables exist. In this thesis we will use MIP to refer to any of the latter two.

For each MIP model, there exist a counterpart LP model, where the only difference is each binary variable is assumed real. We call this LP model the LP relaxation of the MIP model. There exist efficient algorithms to find the optimal solution to an LP problem, the most commonly used is the simplex method. Many techniques build on top of this method to develop sophisticated solution approaches. The most common approach is the Branch and Bound (B&B) method, where the binary variables are fixed one at a time in a tree-shape graph that results in an exponential exploration of the solution space (branching), solving an LP at each node until an integer solution is found. The efficiency of this exploration is greatly improved by being able to prune the tree with information from the best valid solution (integer solution) yet, which imposes an upper bound for the optimal solution (bounding).

Several additional techniques are used to help prune branches of the tree in order to improve performance. Exact methods, such as probing [151], preprocessing [12] and valid cuts [84] can achieve reductions of the solution space without taking out any feasible integer solution (i.e., without loss of optimality). Also, most solvers use a number of primal heuristics

to efficiently find new integer solutions through the use of the Linear Programming (LP) relaxation of the problem and incumbents solutions (e.g., RINS, Diving Heuristics, Local Branching, Feasibility Pump [11, 151, 51, 72]). In Diving Heuristics, a subset of variables is fixed (usually inspired by the LP relaxation), see [64, 15] for examples of this.

Finally, "a priori" heuristic decisions can be incorporated to guide the whole solution process. These heuristics are guided by external information about the problem and correspond to the fixing of variables and the incorporation of heuristic-cuts (also known as pseudo-cuts, see [106]) more generally.

The advancements of MIP solvers, both open-sourced and commercial, over the last two decades [34, 35] make them a very powerful tool for solving a wide range of combinatorial optimization problems, when the appropriate model is built using the best practices [176, 180].

In the case of the MFMP, techniques employed have usually been a mix of MIP models and heuristics built on top of those MIP models [153, 85, 98, 99, 100, 77, 78, 174, 116, 152, 48, 146, 55]. More details on hybrid formulations based on MIP are found in Section 2.3.4.

MIP formulations used to solve the FMP problem and the TA problem are grouped in three categories: String-based models solved by Column Generation (CG) techniques [150], multi-commodity flow formulations (MCNF) [95] and time-space network (TSN) models [86]. In CG and MCNF, a graph is constructed where each node represents a flight and each arc represents a feasible sequence of two consecutive flights. A path in this graph constitutes a feasible schedule for one aircraft for the whole planning horizon. String-based models assign one path to each aircraft. MCNF assign a set of arcs to each aircraft.

In time-space network models, each node represents the event at which an aircraft arrives or departs from a flight. Each arc corresponds to a flight leg (where an aircraft changes airport and the length corresponds to the duration of the flight) or a waiting arc (where the aircraft remains in the same airport and the length corresponds to the lapsed time before the next available flight).

In CG techniques, each variable represents a feasible sequence of assignments to an aircraft. This creates a non-polynomial number of variables with respect with the problem size and becomes impossible to store in memory for medium to large instances. As a result, a CG algorithm is used to solve the Linear Programming (LP) relaxation of such MIP. This LP relaxation is the basis for CG heuristics or exact Branch and Price (B&P) implementations[24, 27].

A key point in the efficiency of a CG scheme is the ability to solve quickly integer subproblems, using Dynamic Programming (DP) algorithms such as the Shortest Path Problem (SPP) [24, 27] and the Shortest Path Problem with Resource Constraints (SPPRC) [150, 148]. These algorithms are also used to solve neighborhoods in hybrid techniques [17] and the corridor method [158]. The particular appeal of using DP algorithms resides in that they explore the exponentially large neighborhoods in polynomial time and obtain better local optimum compared to using traditional local search heuristics with small neighborhoods.

Other DP algorithms, such as A^*, can also be used directly to solve the problem in question. Deng et al. [60] uses a Dynamic Programming-inspired heuristic to solve the AMS problem.

Matheuristics that apply heuristics on top of MIP models are used to solve MFMP problems by variable fixing [55] and splitting the problem by fleet and recombining [116]. Cho [48] proposes a two-stage MIP matheuristic for the complete problem where the first phase decides maintenances and relaxes the missions requirements. The second phase decides the final, discrete, mission assignments.

In large planning problems, where the horizon is divided in discrete periods, rolling horizon (RH) heuristics are commonly used as a constructive decomposition heuristic, as in [30]. The FMP problem is no exception, as presented in [124]. A RH heuristic slices a problem into two or more smaller subproblems with, typically, the same number of periods (time windows size). The subproblems are ordered according to their time windows. Each iteration solves one of the newly created subproblems while using the solution of the previous subproblem as additional constraints in order to guarantee a feasible solution when the last subproblem is solved. This technique shares several similarities with repair / recovery formulations, which are common in the planning of flights [70, 97, 91, 178, 94].

The most common traditional methods to solve the NRP have been exact methods, specially MIP models. MIP-based approaches, including decompositions such as CG and B&P, are used in [22, 31, 40, 80, 90, 115, 149]. Several heuristic approaches based on CG are also proposed [25, 163].

With respect to the RSAMP problem, MIP models based on network-flow formulations are quite common [61, 102, 121, 117, 168]. Lai et al. [102] uses a four-phase matheuristic based on a network-flow MIP model to solve the RSAMP problem.

In the MSPPNP problem, Lusby et al. [114] constructs a two-phase MP approach where the first phase consists of a large MIP model that relaxes some constraints; and the second phase repairs the solution with respect to those relaxed constraints. Jost and Savourey [92] proposed a three-step matheuristic where the first and second phases decided the maintenances and the third assigned the production planning.

2.3.2 Constraint Programming

CP is a technique born from the artificial intelligence domain, initially conceived to solve decision problems where a feasible solution is needed. Nowadays, it also permits to model and solve combinatorial optimization problems. Its main modeling concepts are not too different from the ones in MP. A set of variables represents decisions and a set of constraints relate those variables by limiting certain combinations of values. Nevertheless, the similarities stop here. Each variable is defined by a set of discrete finite possible values, called a domain. Constraints are not limited to linear equations: they apply any function to a subset of variables that limits the product of the domains of those variables.

CP uses deduction (called Constraint Propagation) and induction (called search and backtracking) to reduce the non-valid solution space as much as possible by adding constraints and restricting the domains of each variable. During Constraint Propagation, infeasible values of a variable domain are detected by combining information from other variables' domains and constraints. During search and backtracking, a tree-search is done by temporarily fixing the value of one or more variables in order to find a solution or prove an infeasible solution. This search is similar to the branching present in MIP models. When an infeasible value for a variable is proven, its domain can be reduced, which in turn leads to better propagation. When a domain of a variable reaches a size of 1, the variable can be fixed to that value. If the domain of a variable is empty, an infeasible problem is proven. In order to guide the search towards better solutions, each time a feasible solution is found it is stored and used as an upper bound to constraint the problem even further.

CP approaches usually perform at its best in highly constrained formulations where the valid solution space is relatively small. More detail on CP can be found in Marriott et al. [118], Apt [19], Dechter and Others [56]. In recent years, attempts have done to integrate CP and MIP under the same discipline. In particular Achterberg [10] presents Constraint Integer Programming (CIP) as a new paradigm that includes MIP, CP and SAT modeling and solving techniques combined in a low-level search tree. Combined usage of CP and MP to solve specific relevant problems is detailed in Section 2.3.4.

In the MFMP problem, there is limited evidence of CP models being used. In the case of the French Air Force, there is an ongoing collaboration with Airbus Defense and Cosling, the latter developers of the open source CP solver Choco [138]. The result of this collaboration is the tool called "OptaForce" [9].

Grönkvist [83] proposes several CP models to solve the TA problem. Grönkvist [82] shows how CP can used to pre-process and eliminate impossible flight patterns in a CG technique to solve the TA problem. Gabteni and Grönkvist [74] combines CG and CP to produce near-optimal solutions for long and mid term planning horizons.

CP models have been particularly prominent in the NRP or personnel scheduling applications [36, 46, 103, 120, 126, 160, 169].

2.3.3 Metaheuristics

The world of MH is vast, with as many algorithms as species exist in the animal kingdom [159]. Talbi [164] does a comprehensive study on their nomenclature. In particular, local search and simulated annealing are cataloged as single-solution based MH (S-metaheuristics) in contrast with population based MH (P-metaheuristics) such as evolutionary algorithms and ant colony optimization.

The S-metaheuristics are the most common implementations used to solve the MFMP, FMP and similar problems and so we will focus on this type. Most search-based MH have a similar structure: (1) a choice of one or more types of neighborhood to modify a solution, (2)

a descent step where small changes are explored greedily, (3) an exploration step where local minima is avoided by expanding the range of the search, and lastly (4) a memory of previous visited solutions or changes.

Simulated Annealing (SA) first presented in [96], is a technique based on local search where, at each iteration, a neighbor solution candidate is selected by applying a small change (or move) to the current solution. The candidate is then compared with the current solution and accepted to replace the current solution with a certain probability. The probability depends on several factors such as: the difference in quality between the two solutions; the temperature, which is a function of the iteration number; and configuration parameters.

Variable Neighborhood Search (VNS), first introduced in [123], combines three steps in each iteration: first a shaking phase introduces random changes to a solution to produce a distant neighbor incumbent solution; then a local search step is applied to this new incumbent solution; finally, the new incumbent solution becomes the current solution if a certain condition is reached, e.g., it improves the objective function.

Greedy Randomized Adaptive Search Procedure (GRASP) first presented in Feo and Resende [69] is a technique based on local search were, at each iteration, a new solution is build, then locally improved and finally kept or not. The solution creation process involves applying, at each iteration, a greedy function that varies according to the state of the solution to select the best N candidates moves and then choses one randomly to apply to the function. This is repeated until a solution is obtained. The local improvement is done with a greedy local neighborhood search and stops when the solution can no longer be improved.

In the MFMP problem, Winata [177] uses SA, Tabu Search (TS) and VNS to generate fast solutions and compares them with an exact MIP formulation. In the FMP problem, Murat Afsar et al. [124] use a rolling horizon heuristic solution method where each problem was solved heuristically by exploring a network representation of the possible flights and maintenances for each aircraft. In [63], a heuristic VNS approach was used to solve the TA problem in order to minimize the size of the fleet. For the RSAMP problem, Lin et al. [109] use Simulated Annealing to plan the long term maintenances of high-speed trains. For the NRP, S-metaheuristics are the most common techniques used in the last years, namely VNS [162, 166, 175, 33] and SA [110]. Other techniques used in recent contributions are a hybrid artificial bee colony [21] and a scatter search approach [42]. Heuristic VNS approaches for the NRP are found in [127]. For the MSPPNP problem, Gardi et al. [76] provided one of the best results in the challenge with a pure local search approach.

2.3.4 Hybrid methods

In recent years a growing interest on hybrid MH has motivated the mix of techniques from different domains in order to achieve performance. Usually, these combinations take advantage of the strengths of each of the individual techniques and produce a more robust solution method. Talbi [165] classifies hybrid MH hierarchically according to the level (low-level, high-level) and the mode (relay, teamwork), resulting in 4 types: LRH (low-level relay hybrid),

HRH, LTH, HTH (high-level teamwork hybrid).

In the LRH, one possibility described in Talbi [165] is to have an S-metaheuristic call mathematical programming techniques as very large neighborhood searches (VLNS). These techniques can be branch and bound, dynamic programming or network flow algorithms. In Raidl [142], large neighborhood searches (first presented in Shaw [155] with Constraint Programming) are described as a frequent choice when combining MH and MIP models. In these cases, a compact model together with intelligent variable fixing permits an efficient exploration of the neighborhood: MIP models usually perform well with medium-size instances of problems but often perform poorly on very large ones. Ahuja et al. [16] illustrate different ways to use VLNS with exact methods, including mathematical programming, network flow (i.e., shortest path) and assignment problems specially applied for the TSP. For other applications of MIP on VLNS, see Lopes et al. [112].

Hybridizing several types of neighborhoods in a VNS approach [87] makes the result more robust to local optima: a local optimum for the resulting VNS will be the intersection of the local optimum of all its neighborhoods. The Hill Climbing (HC) version of VNS, known as Variable Neighborhood Descent (VND), is naturally used to solve MIP problems with a B&B exploration of neighborhoods defined by MIP variable fixing.

Many hybrid matheuristics are applied to the NRP by combining MP and search-based MH. Dowsland and Thompson [62] propose a matheuristic based on solving a knapsack model for a feasible problem and then a combination of Tabu Search (TS) and several graph-based neighborhoods modeled in DP are used. Smet and Berghe [157] use a MIP-based VNS matheuristic approach by random fix-and-repair. Other examples of VNS using MIP computations are [140, 43, 175, 57, 170].

For the MSPPNP problem, several hybrid matheuristics are used. Rozenknop et al. [147] builds a CG-based two-phase heuristic hybrid algorithm. Dupin and Talbi [65] present a VND that uses several neighborhoods: time windows, subsets of plants, LP-based variable fixing, among others. Anghinolfi et al. [18] use a three-phase approach where the first phase finds a good initial solution for the maintenance decisions with a MIP model, the second improves the solution with SA and the third phase decides the production planning.

Çakırgil et al. [44] presents a two-phase hybrid heuristic for the NRP. The first phase is a matheuristic build by four phases: clustering, assignment models, routing models and re-assignment models. The second phase is a multi-objective reduced VNS matheuristic to explore non-dominated solutions of the solution of phase one.

In the case of VNS with a CP subproblem applied, examples for the NRP are found in [141, 139]. In Cipriano et al. [49] a CP model was complemented by local search routines and Li and Womer [107] use CP in combination with TS.

For the MSPPNP problem, Gavranović and Buljubašsić [79] iteratively schedule checks using a CP subproblem, and plans production using a greedy heuristic; then improve the plan by local search. Likewise, Brandt et al. [37] device a two-phase heuristic where the first phase

uses a CP subproblem for scheduling checks and the second one a greedy heuristic to produce a production planning.

Hybrids methods including CP and MIP have been specially present in solving NRP. In some cases, CP is used in combination with a CG or B&P approach [181, 58, 88], usually as a subproblem to generate columns. Bertels and Fahle [32] combine CP, MIP and MH and Côté et al. [53] use CP to inspire alternative MIP formulations.

2.4 Machine Learning applied to Mathematical Programming

Given the success of Mathematical Programming (MP)-based approaches to solve the MFMP [48, 108, 116, 55, 154], a natural step forward is to explore new ways of improving the performance of said MP approaches. New advances in Machine Learning (ML) promise to provide such improvements.

The application of ML is a recent complement to existing techniques for solving large-scale CO problems, such as matheuristics and MH [13, 14, 165]. ML models are an heterogeneous group of techniques that were previously known for predicting results based on past information. More recently, though, the surge in popularity of Reinforcement Learning (RL) has made more explicit the link between the CO and ML worlds [28]. Bengio et al. [29] provide several definitions and classifications for the implementations of ML that can be applied to the CO domain. In terms of ML techniques applied to CO the two most common frameworks are supervised learning and reinforcement learning. With respect to the goal on the application of ML, Bengio et al. [29] cite three scenarios: end to end learning, learning meaningful properties of optimization problems, and machine learning alongside optimization algorithms. Talbi [165] offers an overview of hybrid algorithms by combining MH, MP and ML. More related to our case, there exists some previous work on applications of ML on MIP formulations. Following [29], these techniques fall under the category "Learning meaningful properties of optimization problems" by using "demonstration" (or "imitation learning"). In other words, supervised learning models are trained with the help of a set of several instances solved (offline) up to or near to optimality by some exact method. This method is sometimes called "oracle" and usually consists of the original mathematical model solved over a long time and/or over small instances. The objective is to gain insights on the possible solution of a new unseen problem. This information can be used directly to guide decision making (as in [71]) or can be used to increase the performance of the existing model (as in [104, 179, 111]).

Xavier et al. [179] show it is possible to learn from the resolution of an offline set of problems similar to a yet unknown one in order to improve performance by learning features of the solution. For their CO case, solving large-scale security-constrained unit commitment problems, three oracles were constructed: one chooses a subset of computationally heavy constraints that will probably not be needed; the second selects good candidate initial solutions; the third one discards part of the solution space without (much) loss of optimality. Here, the experimentation is done in a training set that corresponds to 300 random variations for each one of eight existing instances from the literature. The ML techniques used are the polling

of the training optimal solutions and Support Vector Machines depending on the oracle.

Fischetti and Fraccaro [71] use ML to predict the value of the objective function on several hard-to-solve instances in order to take strategic decisions without solving the complete tactical CO problem. The application studied is the choice of the optimal location (strategic) and design (tactical) distribution of offshore wind parks. The techniques used are a Support Vector Regression model applied on four features of the input data as well as the optimal objective function for a relaxed problem. The training set consists of 3000 instances of near-optimal solutions solved by heuristic means.

In a similar fashion, Larsen et al. [104] use ML to predict the optimal tactical solution for an operational CO problem. The application studied consists of deciding the configuration and number of railcars (tactical) and container-slot-railcar assignments (operational) in a double-stack intermodal railcar load planning problem. In particular, the tactical decision in question needs to be taken with incomplete information on the instance (namely, the weights of containers). The method used is a regression feedforward neural network, applied on datasets of 100k, 200k and 20M instances solved to under 5% of optimality using a commercial solver.

Lodi et al. [111] use ML to predict the similarity of two given instances. The application studied is a Facility Location Problem. Learning constraints are added in order to force a new instance to have a minimum number of opened facilities. This number depends on the number of facilities that were opened in a reference instance (solved to optimality) and how similar the two instances are.

Dupin and Talbi [66] apply ML to predict if two scenarios will have similar solutions and applies it to the MSPPNP problem. In the stochastic optimization formulation of this problem, scenarios are first clustered so that each group has as set of scenarios as diverse as possible. Then, each cluster is solved independently in order to provide good lower bounds. The similarity of two scenarios is measured by the sum of absolute differences in their demand profiles.

In order to predict characteristics of solutions, certain care needs to be taken when dealing with the possible error in prediction. Most supervised learning methods use a least-squares-minimization technique (or similar) to calculate the expected value of a function. These techniques give no information about the distribution of the variance and they can be specially susceptible to outliers. A more robust technique to predict bounds of dependent variables is to use “superquantiles” or quantile regressions, which are based in the Conditional Value at Risk (CVaR). Rockafellar and Uryasev [145] first introduced the term conditional value at risk in optimization and work by [143, 171, 144] further developed the idea, coining the name “superquantiles”, and applying it to engineering and reliability decision making.

2.5 Conclusions

This chapter presented a review of previous work done to solve the FMP, MFMP, related problems as well as the solution approaches used.

With respect to the civil FMP problem, most work focuses either on the TA problem, where flights are routed by taking into account maintenance constraints, or the AMS problem, where maintenances are scheduled with some or all flights remaining fixed. Only a few actually combine the maintenance scheduling with the flight routing by taking into account the maintenance capacity and the usage-based maintenance needs. Also, short term instances are planned with A-type checks and planning horizons of several days.

With respect to the MFMP problem, the common thread between contributions is the usage-based maintenance scheduling. Missions are sometimes assigned discretely to aircraft and other times are flight hours that need to be distributed among the fleet. An homogeneous fleet is usually assumed. Medium term instances are planned with B or C-type checks and planning horizons of several months.

MP is by far the most common technique to solve the FMP, the MFMP as well as similar problems. In order to solve large instances, the problem is often split into smaller parts (by fleet or by calendar) or decomposed into two or more phases. CG and CG-based heuristics offer an advantage over other types of decompositions by not relaxing the low-level relationship between flights and checks and thus are usually well-suited to solve many of these types of problems.

Recently, hybrid methods that combine optimization approaches from different domains are gaining ground. In particular, MP is currently merged successfully with MH and with ML. These matheuristics take advantage of the good performance and optimality guarantees of MP in small and medium-sized problems and the efficiency with which MH and ML represent and explore the whole solution space.

In the following chapter, we will present a new, long term variant of the MFMP that includes new features not previously seen in the literature, such as calendar-based checks, the minimum durations for mission assignments and the extended size of the planning horizon. Several methods based on MP are conceived that include traditional decompositions, ML-based learned cuts and hybrid matheuristics.

Chapter 3

Complexity analysis and exact methods

Contents

In this chapter, the complexity for the long term MFMP is studied. The instance of a MFMP problem is formalized and then applied in the design of a configurable dataset generator inspired by the real needs of the French Air Force. An exact MIP model for the MFMP problem is formulated and used to test scenarios that vary in fleet size, planning horizon length and mission size in order to obtain insights on the sensitivity of the model to changes in the problem size. Also, additional scenarios are solved in order to test the changes in performance of the model because of changes in individual parameters of the MFMP. Finally, a heuristic is built to generate fast feasible solutions, that in some cases are shown to help warm-start the model.

The chapter is structured as follows. Section 3.1 formally presents the MFMP problem and Section 3.2 studies its complexity. Section 3.3 explains all the input data used in this chapter and the following chapters. Section 3.4 formulates an exact MIP model and Section 3.5 describes a heuristic to generate initial solutions. A description of the instance generation is done in Section 3.6 and the experimentation and results are presented in Section 3.7. Finally, Section 3.8 offers conclusions.

The contributions of this chapter were presented in the following communications: F. Peschiera, A. Haït, N. Dupin, and O. Battaïa. A novel mip formulation for the optimization problem of maintenance planning of military aircraft. In *XIX Latin-Iberoamerican Conference on Operations Research*, pages 1–2, Lima, PE, 2018, F. Peschiera, O. Battaïa, A. Haït, and N. Dupin. Bi-objective mip formulation for the optimization of maintenance planning on french military aircraft operations. 2018. URL `http://oatao.univ-toulouse.fr/20766/`. An article has been submitted for publication: F. Peschiera, A. Haït, N. Dupin, and O. Battaïa. Long term planning of military aircraft flight and maintenance operations. Technical report, ISAE-SUPAERO, UniversitÃĺ de Toulouse, France, 2020. URL `https://arxiv.org/abs/2001.09856`. Finally, a modified version of the heuristic presented Section 3.5 has been tested, validated and successfully exploited by the company Dassault Aviation on real-life instances of Mirage 2000 fleets.

3.1 The long term military flight and maintenance planning problem

Section 3.1.1 formally presents the MFMP problem, including the nomenclature that will be used for the rest of this thesis. These requirements are the result of several meetings with specialists in charge of the planning of maintenances in the Mirage 2000 fleet in the French Air Force as well as maintenance specialists from Dassault Aviation, the company that manufactures the Mirage 2000. During these meetings, the problem was first described in detail, then formalized by us, and finally the requirements were validated together. Section 3.1.2 details the realistic assumptions we take based on these exchanges.

3.1.1 Problem statement

The MFMP problem in question consists of assigning an heterogeneous fleet of military aircraft $i \in \mathcal{I}$ to a given set of scheduled missions $j \in \mathcal{J}$ over a fixed time horizon while also planning when each aircraft will be conducting checks. Constraints can be classified into three groups: (1) the missions requirements, (2) the checks that are needed to keep the fleet in good status and (3) the fleet and its status at any given period.

A series of missions exist along a horizon divided into $t \in \mathcal{T}$ periods. Each mission $j \in \mathcal{J}$ requires a minimum number of aircraft (R_j) among the aircraft that can be assigned to it ($i \in \mathcal{I}_j$). Each assigned aircraft flies H_j hours for each period it is assigned to the mission. An aircraft assigned to a mission j must be assigned for at least MT_j^{min} and at most MT_j^{max} consecutive periods.

Each check has a fixed duration of M periods and cannot be interrupted: during this time the aircraft cannot be assigned to any mission. Let Remaining calendar time (rct) express the maximum number of periods, starting at the beginning of a given time period, before an aircraft must undergo a check; and Remaining flight time (rft), the maximum number of flight hours an aircraft can be flown before requiring a check, at the end of a given time period. The rct (rft) of aircraft i before the first period is Rct_i^{Init} (Rct_i^{Init}). After a check, an aircraft restores its remaining calendar and flight time to their maximum values of E^{max} and H^{max} respectively. Also after a check, the aircraft cannot undergo another check for at least E^{min} periods. The total number of simultaneous checks during each period cannot surpass the workshop capacity C^{max}.

Let serviceability indicate if an aircraft is capable, at the beginning of a given time period, to perform a mission (i.e., is not undergoing a check) and let sustainability be the number of total remaining flight hours for each aircraft at the end of each period. To guarantee both serviceability and sustainability at each time period, missions are grouped into clusters. For each cluster k, a minimal number of serviceable aircraft (A_{kt}^{Clust}) and a minimal sustainability (H_{kt}^{Clust}) is set as a constraint for each period t. All serviceable aircraft have a minimum default usage for each period equal to U^{min} flight hours, which they are required to fly when not assigned to a mission or in a maintenance.

Finally, the main objective is to schedule the checks for all aircraft as late as possible and to minimize the deviations from all elastic constraints. A secondary objective is to balance the flying load among aircraft in the fleet so that the variance of the frequency of checks of each aircraft in the fleet is minimized.

3.1.2 Assumptions

There are constraints that can be violated at a cost per unit of violation. For such elastic constraints, the violation is bounded within intervals where the cost per unit of violation within the interval is constant. Multiple bounded intervals permit increasing the cost per

unit of violation.

Each mission is considered active only during a determined contiguous set of periods. Missions are assumed to require a constant amount of flight hours per period for each aircraft and a constant amount of aircraft per period, when active. Each mission j and each aircraft i have one and only one type Y_j and Y_i, respectively. We assume a capability to be a set of optional aircraft characteristics that may be required by a mission. An aircraft can have none or more capabilities (Q_i), a mission can have at most one (Q_j) and if it does it is called a Special mission. An aircraft i is considered suitable for a mission j (i.e., a candidate $i \in \mathcal{I}_j$) if it shares the same type (i.e., $Y_j = Y_i$) and has the capability required by the mission (i.e., $Q_j \in Q_i$). A cluster is a set of missions such that each mission has exactly the same type, capabilities and, as a result, aircraft candidates.

We assume (realistically for our data) a maximum number of two checks for each aircraft and a minimum of one. Maintenance capacity is constant over the planning horizon. All checks have the same duration and frequency conditions.

We assume some aircraft (N_t^{Init}) are already in maintenance in period t at the beginning of the planning horizon. Other aircraft are conducting missions that started before the start of the planning horizon and their continued assignment extends into the planning horizon by $\mathcal{A}_{ij}^{Init}$ (the fixed set of periods aircraft i extends assignment for mission j at the start of the planning horizon).

3.2 Complexity analysis

In the current Section, a proof of complexity for the MFMP problem is provided. The proof is organized as follows: Section 3.2.1 presents the Shift Satisfaction Personnel Task Scheduling Problem (SSPTSP), Section 3.2.2 presents the Reduced Flight Planning Problem (RFPP) as a special case of the MFMP. Finally, Section 3.2.3 uses the RFPP to solve the decision problem in the SSPTSP, thus concluding the proof.

3.2.1 Shift Satisfaction Personnel Task Scheduling Problem

The SSPTSP is a NP-Complete problem presented by Arkin and Silverberg [20]. It is a special case of the Shift Minimization Personnel Task Scheduling Problem (SMPTSP) [101, 156] where there is no objective function and thus, only the satisfaction of constraints is required.

A description of the problem, input data and model follow, using the notation in Smet [156]:

Let $\mathcal{P} = 1, ..., n$ be the set of tasks to be assigned and $\mathcal{E} = 1, ..., m$ the set of employees. Each task $p \in \mathcal{P}$ has a duration u_p, a start time s_p and an end time $f_p = s_p + u_p$. Each employee e has a set of tasks $\mathcal{P}_e \subseteq \mathcal{P}$ that he/she can perform. Similarly, for each task p, a

set $\mathcal{E}_p \subseteq \mathcal{E}$ exists, which contains all employees that can perform task p. Both $\mathcal{P}_e$ and $\mathcal{E}_p$ are defined based on qualifications, time windows of tasks and the availability of employees. In a feasible solution, all tasks in $\mathcal{P}$ are assigned to qualified employees from $\mathcal{E}$ in a non-preemptive manner.

The SSPTSP consists in answering the question: is a planning covering all tasks feasible?

Let $\mathcal{N}_{SSPTSP}$ be the set of all instances for the SSPTSP. Any instance $n \in \mathcal{N}_{SSPTSP}$ can then be expressed by the following notation:

$\mathcal{E}$	employees.
$\mathcal{P}$	tasks.
u_p	duration of task p.
s_p	start time of task p.
f_p	end time of task p.
$\mathcal{P}_e$	set of tasks employee e can perform.
$\mathcal{E}_p$	set of employees that can perform task p.

Decision variables and model Let binary variable x_{pe} take the value 1 if task p is assigned to employee e and 0 otherwise.

Two tasks p and p' overlap if their time intervals $[s_p, f_p]$ and $[s_{p'}, f_{p'}]$ overlap. Let a clique K be defined as a set of tasks that overlap in time and thus cannot be scheduled to the same employee. K is said to be a maximal clique if there is no other clique K' that includes K as a smaller subset, i.e., if there is no other K' such that $K \subset K'$. Let $K \in C_e$ be the set of maximal cliques among the tasks $\mathcal{P}_e$ that the employee e can be assigned to.

The model to solve the SSPTSP is then:

$$feasibility \tag{3.1}$$

subject to:

$$\sum_{e \in \mathcal{E}_p} x_{pe} = 1 \qquad p \in \mathcal{P} \tag{3.2}$$

$$\sum_{p \in \mathcal{K}} x_{pe} \leq 1 \qquad e \in \mathcal{E}, K \in \mathcal{C}_e \tag{3.3}$$

$$x_{pe} \in \mathbb{B} \qquad p \in \mathcal{P}, e \in \mathcal{E}_p \tag{3.4}$$

Constraints (3.3) set the number of employees assigned to each task to 1. Constraints (3.3) guarantee that no overlapping tasks are assigned to the same employee. Constraints

(3.4) set the bounds for the decision variables.

3.2.2 Reduced Flight Planning Problem

A simplified description of our original problem, that we call Reduced Flight Planning Problem (RFPP) is drafted so that it complies with the formulation of the SSPTSP.

The RFPP is a MFMP problem with the following characteristics: (1) each mission $j \in \mathcal{J}$ requires one aircraft, i.e. $R_j = 1$; (2) missions do not require any flight hours when assigned and there are no minimum flight hours per period, i.e., $H_j = 0, U^{min} = 0$; (3) each aircraft has just exited a maintenance before the beginning of the first period and thus have sufficient initial rct and rft to not need any check during the planning horizon, i.e., $Rct_i^{Init} = |\mathcal{T}| + 1, Rft_i^{Init} = H^{max}$; (4) because of points 2 and 3, there is no need of scheduling checks and no need of constraining the sustainability and serviceability of each cluster, i.e., $A_{kt}^{Clust} = |\mathcal{I}|, H_{kt}^{Clust} = 0$; (5) aircrafts have no previous fixed assignments, i.e., $N_{kt}^{Clust} = 0, N_t^{Init} = 0, \mathcal{A}_{ij}^{Init} = \emptyset$. The objective function of the RFPP is a constant. Any RFPP instance is thus a particular case of the MFMP problem and so, if the RFPP is NP-complete, the MFMP problem is also NP-complete.

The RFPP consists then in the following decision problem. Let $j \in \mathcal{J}$ be the set of missions planned along a horizon of $t \in \mathcal{T}$ planning periods. Let $i \in \mathcal{I}$ the set of aircraft. Each mission j is active during periods $\mathcal{T}_j \subseteq \mathcal{T}$ and requires one aircraft to be assigned at each period $t \in \mathcal{T}_j$. Each assignment of an aircraft i to a mission j has a minimum (maximum) duration of MT_j^{min} (MT_j^{max}) periods. Each aircraft i has a set of missions $\mathcal{J}_i \subseteq \mathcal{J}$ that it can be assigned to. Similarly, for each mission j, a set $\mathcal{I}_j \subseteq \mathcal{I}$ exists, which contains all aircraft that can be assigned to mission j.

The RFPP answers the question: is a planning covering all missions feasible?

Decision variables and model Let binary variable a_{jti} take the value 1 if mission j is assigned during period t to aircraft i and 0 otherwise. Let binary variable a_{jti}^s take the value 1 if aircraft i starts a new assignment to mission j in period j, i.e., if $a_{jti} = 1$ and $a_{j(t-1)i} = 0$ and 0 otherwise.

The model to solve the RFPP is then:

$$feasibility \tag{3.5}$$

subject to:

$$\sum_{i \in \mathcal{I}_j} a_{jti} = 1 \qquad j \in \mathcal{J}, t \in \mathcal{T}_j \tag{3.6}$$

$$a^s_{jti} \geq a_{jti} - a_{j(t-1)i} \qquad t = 2, ..., \mathcal{T}, j \in \mathcal{J}_t, i \in \mathcal{I}_j \tag{3.7}$$

$$a^s_{jti} \geq a_{jti} \qquad t = 1, j \in \mathcal{T}_t, i \in \mathcal{I}_j \tag{3.8}$$

$$\sum_{t' \in \mathcal{T}^{MT}_{jt}} a^s_{jt'i} \leq a_{jti} \qquad j \in \mathcal{J}, t \in \mathcal{T}_j, i \in \mathcal{I}_j \tag{3.9}$$

$$a^s_{jti} \in \mathbb{B} \qquad j \in \mathcal{J}, t \in \mathcal{T}_j, i \in \mathcal{I}_j \tag{3.10}$$

$$a_{jti} \in \mathbb{B} \qquad j \in \mathcal{J}, t \in \mathcal{T}_j, i \in \mathcal{I}_j \tag{3.11}$$

Constraints (3.6) set the number of aircraft assigned to each task j at each period t to 1. Constraints (3.7-3.8) guarantee that the variable starts are correctly modeled. Constraints (3.18-3.19) force an assignment to start at period t if aircraft i is firstly assigned to mission j i.e. aircraft i is not assigned to mission j in period $(t-1)$. Constraints (3.20) control the minimum duration of a consecutive mission assignment. If aircraft i is firstly assigned to mission j in period t, it has to be assigned to it during the following $t' \in \mathcal{T}^{MT}_{jt}$ periods. Constraints (3.11-3.10) set the bounds for the decision variables.

3.2.3 Reduction

Theorem 1
Finding a feasible solution to the RFPP is equivalent to solving the SSPTSP.

Proof. For each employee $e \in \mathcal{E}$ in SSPTSP, we create an analogous aircraft $i \in \mathcal{I}$ in RFPP, we will use e and i indistinctly. For each task $p \in \mathcal{P}$ in SSPTSP, we create a $j \in \mathcal{J}$ mission in RFPP, we will use p and j indistinctly. The compatibility between missions and aircraft is equivalent to that of tasks and employees: $\mathcal{J}_i = \mathcal{P}_i$ and $\mathcal{I}_j = \mathcal{E}_j$. The minimal and maximal assignment duration time of each mission are equal to the duration of the task $MT^{min}_j = MT^{max}_j = u_j$. Start times and end times define the moment when the mission is active: $\mathcal{T}_j = t \in \{s_j ... f_j\}$.

Let $n \in \mathcal{N}_{RFPP}$ be the set of all instances of problem RFPP. For each instance $n \in \mathcal{N}_{SSPTSP}$, an instance $n' = f(n)$ is created. The details of transformation f are shown in Table 3.1.

Let:
Q_{SSPTSP}: for an instance $n \in \mathcal{N}_{SSPTSP}$: $\exists$ a feasible solution?
Q_{RFPP}: for an instance $n' = f(n)$: $\exists$ a feasible solution?

Given that an answer to Q_{RFPP} is also an answer to Q_{SSPTSP} for each $n \in \mathcal{N}_{SSPTSP}$ and that SSPTSP is NP-complete, this proves that the RFPP is NP-complete. This, in turn, proves that the MFMP problem is NP-complete.

RFPP	Meaning	f(n)
$t \in \mathcal{T}$	planning horizon.	$\{\min_p(s_p)... \max_p(f_p)\}$
$i \in \mathcal{I}$	aircraft.	$\mathcal{E}$
$j \in \mathcal{J}$	missions.	$\mathcal{P}$
$t \in \mathcal{T}_j$	time periods in which task $j = p$ is active.	$t \in \{s_j...f_j\}$
$j \in \mathcal{J}_t$	missions $j \in \mathcal{J}$ to be realized in period t.	$j\|t \in \{s_j...f_j\}$
$i \in \mathcal{I}_j$	aircraft that can be assigned to mission $j = p$.	$\mathcal{E}_j$
$j \in \mathcal{J}_i$	missions for which aircraft $i = e$ can be used.	$\mathcal{P}_i$
MT_j^{min}	minimum number of consecutive periods for task $j = p$.	u_j
MT_j^{max}	maximum number of consecutive periods for task $j = p$.	u_j

Table 3.1: Interval scheduling set translation

□

3.3 Input data

In this Section we explicitly present all sets, parameters as well as many derived sets and parameters for the MFMP problem. These same definitions and nomenclature will be shared among all chapters.

3.3.1 Sets and parameters

Basic sets

$i \in \mathcal{I}$	aircraft.
$t \in \mathcal{T}$	time periods included in the planning horizon. We use $t = 0$ for starting conditions and $t = T$ for the last period.
$j \in \mathcal{J}$	missions.

Auxiliary sets

$y \in \mathcal{Y}$	type of aircraft.
$k \in \mathcal{K}$	cluster of missions that require the same functionality.
$c \in \mathcal{C}$	capabilities for missions.

Mission parameters [units]

H_j	flight hours required per period and aircraft for mission j.	[hours]
R_j	number of aircraft required per period for mission j.	[aircraft]
MT_j^{min}	minimum number of consecutive periods required for an aircraft to be assigned to mission j.	[periods]
MT_j^{max}	maximum number of consecutive periods an aircraft can be assigned to mission j.	[periods]
U^{min}	flight hours required per period and aircraft when not assigned to any mission nor in maintenance.	[hours]
Y_j	type of mission j.	[type]
Q_j	optional capability required for mission j.	[capability]

Maintenance parameters [units]

M	number of periods for a check.	[periods]
C^{max}	maximum number of simultaneous aircraft checks.	[aircraft]
E^{min}	minimum number of periods between two consecutive checks for each aircraft.	[periods]
E^{max}	maximum number of periods between two consecutive checks for each aircraft.	[periods]
H^{max}	maximum number of flight hours between two consecutive checks for each aircraft.	[hours]

Fleet parameters [units]

N_t^{Init}	number of aircraft pre-assigned to a maintenance check at the start of period t.	[aircraft]
N_{kt}^{Clust}	number of aircraft in cluster k pre-assigned to a maintenance check at the start of period t.	[aircraft]
A_{kt}^{Clust}	maximum number of cluster k aircraft that can be simultaneously in maintenance at start of period t.	[aircraft]
H_{kt}^{Clust}	required remaining flight hours for cluster k at end of period t.	[hours]
Rft_i^{Init}	remaining flight time for aircraft i from the start of the planning horizon.	[hours]
Rct_i^{Init}	remaining calendar time until aircraft i reaches E^{max} from the start of the planning horizon.	[periods]
Y_i	type of aircraft i.	[type]

Index sets

$i \in \mathcal{I}_y$	aircraft $i \in \mathcal{I}$ belonging to type y. One aircraft can belong to only one type.
$j \in \mathcal{J}_y$	missions $j \in \mathcal{J}$ belonging to type y. One mission can belong to only one type.
$c \in \mathcal{Q}_i$	capabilities $c \in \mathcal{C}$ belonging to aircraft i.
$j \in \mathcal{J}_i$	missions $j \in \mathcal{J}$ where aircraft i is suitable.
$i \in \mathcal{I}_j$	aircraft $i \in \mathcal{I}$ suitable for mission j.
$t \in \mathcal{T}_j$	time periods $t \in \mathcal{T}$ when mission j is active.
$j \in \mathcal{J}_t$	missions $j \in \mathcal{J}$ that are active in period t.
$i \in \mathcal{I}_k$	aircraft $i \in \mathcal{I}$ belonging to cluster k. One aircraft can belong to more than one cluster.
$i \in \mathcal{A}_{ij}^{Init}$	periods $t \in \mathcal{T}$ where aircraft i is pre-assigned to mission j.

Note $\mathcal{J}_i$ and $\mathcal{I}_j$ are calculated based on $\mathcal{I}_y$, $\mathcal{J}_y$, $\mathcal{Q}_i$ and Q_j. For an aircraft i to be able to be assigned to a mission j it needs to share the same type y as the mission and have the required capability ($Q_j \in \mathcal{Q}_i$).

3.3.2 Time-related index sets

We define several sets based on the input data to simplify the constraint formulation of all models in this thesis. The equations related to these sets, together with an example, are given in Appendix A.

$t \in \mathcal{T}_i^{M_{Init}}$	time period options $t \in \mathcal{T}$ for aircraft i to start its first check.
$t \in \mathcal{T}_i^{m_{Init}}$	time periods $t \in \mathcal{T}$ when aircraft $i \in \mathcal{I}$ cannot start its first check.
$t \in \mathcal{T}_{t'}^{s}$	time periods $t \in \mathcal{T}$ required for a check that ends in t'.
$t \in \mathcal{T}_{t'}^{m}$	time periods $t \in \mathcal{T}$ when a second check cannot start if the first check starts in period t'.
$t \in \mathcal{T}_{t'}^{MM}$	time period options $t \in \mathcal{T}$ when a second check must start if the first check starts in period t'.
$t \in \mathcal{T}_{t'}^{M}$	time periods $t < T$ permitted for a second check to start, given the first check started in t' and excluding the need for a third check.
$t \in \mathcal{T}_{t'}^{M+}$	time periods permitted for a second check to start, including the possibility $t = T$ for not doing a second maintenance.
$(t_1, t_2) \in \mathcal{TTT}_t$	pairs of time periods $t_1 \in \mathcal{T}, t_2 \in \mathcal{T}_{t_1}^{M}$ when a first and second check can start and the aircraft is in maintenance in period t.
$t \in \mathcal{T}_{jt'}^{MT}$	time periods $t \in \mathcal{T}$ when, if a mission assignment starts, the assignment continues in t'.
$(t, t') \in \mathcal{TT}_j$	set of all possible start t and finish t' combinations for assignment of mission.
$(t_1, t_2) \in \mathcal{TTJ}_{jt}$	allowed assignments for mission j that start (end) at period t_1 (t_2) and contain period t.
$(j, t, t') \in \mathcal{JTT}_{it_1t_2}$	allowed mission assignments that start (end) at period t (t') for each aircraft i and for each mission $j \in \mathcal{J}_i$ between checks starting at t_1 and t_2.

3.3.3 Auxiliary parameters

To condense notation, we define parameters that aggregate flight hour usage and initial status. $U'_{tt'}$ is the flight hour usage for each aircraft between t and t' without taking into consideration any mission or check assignment. $H'_{jtt'}$ is the additional flight hour usage for each aircraft when assigned to mission j between periods t and t'. AT_{ijt}^{Init} is a binary representation of $t \in \mathcal{A}_{ij}^{Init}$.

$$
\begin{aligned}
U'_{tt'} &= U^{min}(t' - t + 1) \\
H'_{jtt'} &= (H_j - U^{min})(t' - t + 1) \\
AT_{ijt}^{Init} &= \begin{cases} 0 & t \notin \mathcal{A}_{ij}^{Init} \\ 1 & t \in \mathcal{A}_{ij}^{Init} \end{cases}
\end{aligned}
\tag{3.12}
$$

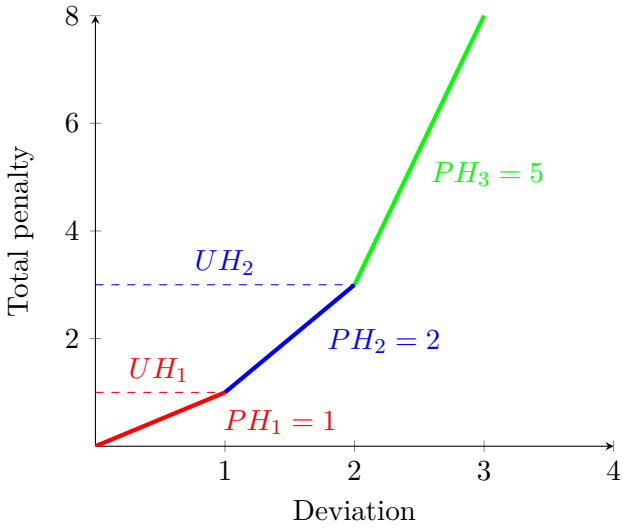

Figure 3.1: Example interval deviation for sustainability constraint with $|\mathcal{S}| = 3$.

3.3.4 Interval deviation and objective function parameters

The following parameters are used to penalize the objective function in the model in Chapter 4. The weights are piece-wise linear and non-decreasing. An example of the usage of these weights is shown in Figure 3.1 where the relationship between the penalty cost PH_s and the maximum deviation UH_s is shown. The last interval, i.e., $s = 3$, has no upper bound.

$s \in \mathcal{S}$	interval for constraint violation.	
UA_s	maximum deviation for violating the serviceability limit in interval s.	[aircraft]
PA_s	penalty cost for violating serviceability constraint in interval s.	$[\frac{penalty}{aircraft-period}]$
UH_s	maximum deviation for violating the sustainability limit in interval s.	[hours]
PH_s	penalty cost for violating sustainability constraint in interval s.	$[\frac{penalty}{hour-period}]$
UC_s	maximum deviation for violating the maintenance capacity limit in interval s.	[aircraft]
PC_s	penalty cost for violating capacity constraint in interval s.	$[\frac{penalty}{aircraft-period}]$
$P2M$	reward per period for the start of the second check.	$[\frac{penalty}{aircraft-period}]$

3.4 Mathematical formulation

The following model provides a tight MIP formulation that solves the Military Flight and Maintenance Problem described in Chapter 3.1. Decision variables for assigning missions and maintenances are similar to the DCM formulation presented in [48].

3.4.1 Variables

The following decision variables control the assignment of missions and checks to aircraft.

a_{jti}	=1 if mission $j \in J$ is assigned to aircraft $i \in \mathcal{I}_j$ in period $t \in \mathcal{T}_j$, 0 otherwise.
a^s_{jti}	=1 if aircraft i starts a new assignment to mission j in period t. If $a_{jti} = 1$ and $a_{j(t-1)i} = 0$.
m_{it}	=1 if aircraft $i \in I$ starts a check in period $t \in \mathcal{T}$, 0 otherwise.

The following decision variables control the used and remaining flight time in aircraft.

u_{it}	flight hours (continuous) of aircraft $i \in I$ during period $t \in \mathcal{T}$.
rft_{it}	remaining flight time (continuous) for aircraft $i \in I$ at the end of period $t \in \mathcal{T}$.

Fixed values Note that a_{jti} and m_{it} are initially set up to 0 for all aircraft already in maintenance at the beginning of the planning horizon for the remaining time periods of the check. N_t is calculated based on this information. Similarly, for aircraft that have not yet complied with their minimum mission assignment duration at the beginning of the planning horizon, a_{jti} is fixed to comply with the constraints.

3.4.2 Objective function and constraints

Two objectives have been studied. Objective (3.13) minimizes the number of checks. (3.14) combines the first one with the goal of maximizing the final total flight hours potential of the fleet. These objectives do not take into account the balancing of the flight load for the fleet.

$$\text{Min} \sum_{t \in \mathcal{T}, i \in \mathcal{I}} m_{it} \tag{3.13}$$

$$\text{Min} \sum_{t \in \mathcal{T}, i \in \mathcal{I}} m_{it} \times H^{max} - \sum_{i \in \mathcal{I}} rft_{iT} \tag{3.14}$$

The first term counts all the flight hours given to aircraft following checks and the second term quantifies the amount of remaining flight hours for all aircraft at the end of the planning horizon. These two objectives have the same units, can be easily compared and ensure the aircraft are used in the most efficient way.

The following constraints are used in the model:

$$\sum_{t' \in \mathcal{T}_t^s} \sum_{i \in \mathcal{I}} m_{it'} + N_t \leq C^{max} \qquad t \in \mathcal{T} \tag{3.15}$$

$$\sum_{i \in \mathcal{I}_j} a_{jti} \geq R_j \qquad j \in \mathcal{J}, t \in \mathcal{T}_j \tag{3.16}$$

$$\sum_{t' \in \mathcal{T}_t^s} m_{it'} + \sum_{j \in \mathcal{J}_t \cap \mathcal{J}_i} a_{jti} \leq 1 \qquad t \in \mathcal{T}, i \in \mathcal{I} \tag{3.17}$$

Maintenance capacity is controlled by (3.15). The aircraft requirements of missions are defined by (3.16). Constraints (3.17) ensure that an aircraft can only be used for one mission or undergo check in the same period.

$$a^s_{jti} \geq a_{jti} - a_{j(t-1)i} \qquad t = 2, ..., \mathcal{T}, j \in \mathcal{J}_t, i \in \mathcal{I}_j \tag{3.18}$$

$$a^s_{jti} \geq a_{jti} - AT^{Init}_{ijt} \qquad t = 1, j \in \mathcal{T}_t, i \in \mathcal{I}_j \tag{3.19}$$

$$\sum_{t' \in \mathcal{T}_{jt}^{MT}} a^s_{jt'i} \leq a_{jti} \qquad j \in \mathcal{J}, t \in \mathcal{T}_j, i \in \mathcal{I}_j \tag{3.20}$$

Constraints (3.18) captures period t where aircraft i is firstly assigned to mission j i.e. it is not assigned to it in period $(t-1)$. Constraints (3.19) are introduced for the first period in the planning horizon.

Constraints (3.20) control the minimum duration of a consecutive mission assignment. If aircraft i is firstly assigned to mission j in period t, it has to be assigned to it during the following $t' \in \mathcal{T}_{jt}^{MT}$ periods. This is a stronger version of the constraint $a^s_{jt'i} \leq a_{jti}$.

To our knowledge, Constraints (3.18-3.20) have not been taken into account in previous military MFMP problems.

$$\sum_{t' \in \mathcal{T}_t^s} \sum_{i \in \mathcal{I}_k} m_{it'} + N^K_{kt} \leq A^{Clust}_{kt} \qquad k \in \mathcal{K}, t \in \mathcal{T} \tag{3.21}$$

$$\sum_{i \in \mathcal{I}_k} rft_{it} \geq H^{Clust}_{kt} \qquad k \in \mathcal{K}, t \in \mathcal{T} \tag{3.22}$$

Constraints (3.21) guarantee a minimum serviceability of aircraft for each cluster k. A cluster is defined by the largest group of aircraft that is required exclusively for at least one mission. Constraints (3.22) ensure there is a minimum amount of remaining flight time for each cluster k.

$$u_{it} \geq \sum_{j \in \mathcal{J}_t \cap \mathcal{J}_i} a_{jti} H_j \qquad t = 1, ..., \mathcal{T}, i \in \mathcal{I} \qquad (3.23)$$

$$u_{it} \geq U^{min}(1 - \sum_{t' \in \mathcal{T}_t^s} m_{it'}) \qquad t = 1, ..., \mathcal{T}, i \in \mathcal{I} \qquad (3.24)$$

$$u_{it} \in [0, \max_j \{H_j\}] \qquad t = 1, ..., \mathcal{T}, i \in \mathcal{I} \qquad (3.25)$$

$$rft_{it} \leq rft_{i(t-1)} + H^{max} m_{it} - u_{it} \qquad t = 1, ..., \mathcal{T}, i \in \mathcal{I} \qquad (3.26)$$

$$rft_{i0} = Rft_i^{Init} \qquad i \in \mathcal{I} \qquad (3.27)$$

$$rft_{it} \geq H^{max} m_{it'} \qquad t \in \mathcal{T}, t' \in \mathcal{T}_t^s, i \in \mathcal{I} \qquad (3.28)$$

$$rft_{it} \in [0, H^{max}] \qquad t \in \mathcal{T}, i \in \mathcal{I} \qquad (3.29)$$

The flight time per aircraft and period is calculated in (3.23)-(3.25). The remaining flight time is defined by (3.26)-(3.27) and its limits by (3.28)-(3.29).

$$m_{it'} + m_{it} \leq 1 \qquad t \in \mathcal{T}, t' \in \mathcal{T}_t^m, i \in \mathcal{I} \qquad (3.30)$$

$$\sum_{t' \in \mathcal{T}_t^{MM}} m_{it'} \geq m_{it} \qquad t \in \mathcal{T}, i \in \mathcal{I} \qquad (3.31)$$

$$m_{it} = 0 \qquad t \in \mathcal{T}_i^{m_{Init}}, i \in \mathcal{I} \qquad (3.32)$$

$$\sum_{t \in \mathcal{T}_i^{M_{Init}}} m_{it} \geq 1 \qquad i \in \mathcal{I} \qquad (3.33)$$

The minimum and maximum calendar times are defined by (3.30) and (3.31) respectively. Constraints (3.30) guarantee that if a check is done in some period t, we know that another one cannot be done in the immediately consecutive $t' \in \mathcal{T}_t^m$ periods. Constraints (3.31) ensure that if a check is planned in period t, we need to start at least one check in periods $t' \in \mathcal{T}_t^M$. Constraints (3.32) and (3.33) control the minimum and maximum remaining calendar times respectively at the beginning of the planning period. They follow the same logic as constraints (3.30) and (3.31), respectively.

To our knowledge, calendar based constraints such as Constraints (3.30-3.33) have not been taken into account in previous MFMP problems.

This model will be used in the experimentation done in Section 3.7.

3.5 Heuristic initial solution

In order to improve the performance of the model presented in Section 3.4, a heuristic that produces fast feasible or near-feasible solutions is devised. These solutions will be used to

warm-start the MIP solver.

The heuristic consists in a Simulated Annealing heuristic detailed in Algorithm 1. The stop criteria are two: getting a solution with 0 errors or an iteration limit. Given the structure of the constraints and the design of the algorithm, the heuristic does not guarantee a feasible solution to the problem.

The first iteration $c = 1$ generates an initial solution by first scheduling random mandatory checks (see Line 6) and, then, randomly assigning missions as needed (see Line 7). Both functions are explained in detail in Algorithms 2 and 3, respectively. At each subsequent iteration $c \geq 2$, the solution is first perturbed by removing or moving maintenances and mission assignments (see Lines 15 and 16), in order to randomly re-schedule checks and then re-assign missions.

Each candidate selected for releasing in Line 15 consists of a couple (aircraft, period). These candidates are selected based on the location of errors in the incumbent solution. The perturbations in Line 16 release a slice of the $\mathcal{I} \times \mathcal{T}$ matrix of mission assignments and checks schedules. This slice can be a whole row, i.e., free all assignments and checks for aircraft i; a group of columns, i.e., free all assignments and checks for all aircraft between periods t and t'; or a combination of the two, i.e., free assignments and checks for subset of aircraft between periods t and t'. The type of release is randomly generated.

After each cycle, the solution is compared with the previous one (see line 9) and is accepted depending on the difference in quality, a decreasing temperature and a random factor. The greater the temperature, the greater the probability to accept a new solution that has more errors than the incumbent.

3.6 Dataset generation

The data set for the numerical experiments is generated on the basis of possible data structures used by Air Forces. The methodology used in this section is employed to generate all instances studied in this thesis, although the scenario configuration will vary depending on each study.

In order to generate the data, a formal specification of an instance is done. This specification is more general than the standard defined in 3.3 and can be viewed in detail in Appendix B.

The notation used is the following. A discrete choice of values is indicated by values separated by commas. Intervals indicate that a value is chosen in a "uniform random way" from the intervals for continuous values or through random sampling with replacement for integer values. Values with a * are deterministic control parameters.

The rest of the Section is structured as follows: Section 3.6.1 describes the main parameters used to generate instances: the length of the planning horizon, the number of active missions per period and the size of the fleet. Section 3.6.2 covers the maintenance parameter generation,

Algorithm 1: maintFirst

Data:
x: incumbent solution, err its errors.
x': new candidate solution.
x^*: best solution found.
T: temperature.
C: maximum iterations.
R: cooldown rate.

```
1  begin
2      x ← InitializeEmptySolution()
3      err ← err* ← GetErrors(x)
4      x' ← x
5      for c ← 1 to C do
6          x' ← AssignChecks(x')
7          x' ← AssignMissions(x')
8          err' ← GetErrors(x')
9          if AcceptanceFunc(err, err', T) then
10             x, err ⟵ x', err'
11             if ∑err' < ∑err* then
12                 x*, err* ⟵ x', err'
13         T ← R × T
14         if |err*| = 0 then break
15         C ← GetCandidatesReassign(err)
16         x' ← PartialRelease(x, C)
```

Algorithm 2: AssignChecks()

Data:
x: the current solution

```
1  begin
2      for i ∈ ℐ do
3          needs ← GetMaintenanceNeeds(x, i)
4          𝒯^c ← GetMaintenanceCandidates(x, i, needs)
5          if |𝒯^c| > 0 then
6              t ← choice(𝒯^c)
7              SetMaintenance(x, i, t);
```

Algorithm 3: AssignMissions()

```
Data:
x: the current solution
begin
    for j ∈ shuffle(J) do
        needs ← CheckMissionNeeds(x, j)
        I^c ← GetMissionCandidates(j)
        while |I^c| > 0 ∧ |needs| > 0 do
            i ← choice(I^c)
            T^c ← GetCandidatePeriods(needs, i)
            if |T^c| = 0 then
                I^c.pop(i)
            for t ∈ shuffle(T^c) do
                success ← SetMissionAssignment(x, i, t, j)
                if success then
                    needs[t] ← needs[t] − 1
```

Section 3.6.3 cover mission parameters and Section 3.6.4 describe the aircraft parameters. Sections 3.6.5 and 3.6.6 explain the aircraft-mission compatibility, cluster creation and service levels for availability and sustainability.

3.6.1 Sets

Code	Parameter	Value
$\|\mathcal{J}^P\|$	Total number of parallel missions*	1, 2, 3, 4
$\|\mathcal{I}\|$	Number of aircraft*	15, 30, 45, 60
$\|\mathcal{T}\|$	Number of periods*	90, 120, 140

3.6.2 Maintenances

Code	Parameter	Value
C^{perc}	Maintenance capacity (percentage)*	0.10, 0.15, 0.2
C^{max}	Maintenance capacity	$\lceil C^{perc} \times \|\mathcal{I}\| \rceil$
E^{max}	Time limit in periods*	40, 60, 80
E^{size}	Time limit window*	20, 30, 40
H^{max}	Flight hours limit*	800, 1000, 1200
E^{min}	Time limit in periods	$(E^{max} - E^{size})$
M	Check duration*	4, 6, 8

3.6.3 Missions and flight hours

The total number of types of missions corresponds to the number of parallel missions one can allow in order to guarantee that, at any moment in time there is only one active mission for each type. The flight hours are generated using a triangular distribution between 30 and 80 with a mode of 50 and rounded down to the closest integer value. Regarding types and standards, see Section 3.6.5.

The following logic has been used to generate the missions, assuming $N = |J^P|$ active missions in each period. We create N missions with a random duration that start at period $t = 1$. Every time a mission ends, we create a new mission with new random parameters. If the newly created mission ends after period T, we make that mission end at period T and we do not create a new mission. Algorithm 4 shows the logic for the mission generation that guarantees there are *always* N active missions at any given time.

Algorithm 4: Mission generation logic

Data:
start: start date for mission.
m: created mission.

```
1 begin
2     for 1 to N do
3         start ← 1
4         repeat
5             m ← create_random_mission_at(start)
6             if end(m) > T then
7                 end(m) ← T
8             start ← end(m) + 1
9         until end(m) = T
```

Code	Parameter	Value
$\|\mathcal{T}_j\|$	Duration (periods)	6 – 12
MT_j^{min}	Minimum assignment (periods)	2, 3, 6
MT_j^{max}	Maximum assignment (periods)	$\|\mathcal{T}_j\|$
R_j	Number of required aircraft	2 – 5
H_j	Number of required hours	triangular(30, 50, 80)
U^{min}	Default assignment flight hours*	0, 5, 15, 20
Y_j	Type	choice 1
Q_j	Standard	10% chance

3.6.4 Aircraft

Each aircraft has specific characteristics that allow it to accomplish missions. These characteristics are represented by a type and a standard. More detail on types and standards is

discussed in 3.6.5.

Code	Parameter	Value
Y_i	Type	choice
Q_i	Standards	choice

Each aircraft has an initial state based on Rct_i^{Init}, Rft_i^{Init} and the fixed assignments At_{ij} and checks NM_i from the periods previous to the start of the planning horizon. Algorithm 5 summarizes the logic behind the creation of these parameters.

Code	Parameter	Value
NP	Percentage of aircraft starting in maintenance.	7%
At_{ij}	Number of periods previously done under mission j	$0 - 2MT_j^{min}$
Rct_i^{Init}	Remaining calendar time	$0 - E^{max}$
Rft_i^{Init}	Remaining flight time	$0 - H^{max}$
NM_i	Remaining maintenance periods	$0 - M$

The initial states are simulated according to the following rules. (i) NV aircraft are sampled from the set of aircraft and become the aircraft under maintenance $\mathcal{I}^M \subset \mathcal{I}$; (ii) for each sampled aircraft $i \in \mathcal{I}^M$, NM_i is generated randomly and $Rct_i^{Init} = H^{max}$, $Rft_i^{Init} = E^{max}$ (see Line 4).

For the remaining $I - \mathcal{I}^M$ aircraft that are not in maintenance: (i) Rct_i^{Init} is first generated randomly and then Rft_i^{Init} is generated randomly from the value of the former (see Line 8); (ii) for each mission j belonging to the set of missions active at the beginning of the planning period: R_j aircraft are sampled and assigned to each such a mission with At_{ij} previous assignments (see Line 16).

3.6.5 Mission-aircraft compatibility

Mission compatibility parameters are generated in the following way. For each mission, a type $Y_j \in Y$ and a standard $Q_j \in Q$ are assigned. Q_j can be null, which implies the mission has no standard. A minimum number of aircraft of type y is calculated based on $\sum_{\{j \in J | Y_j = y\}} R_j$.

In order to guarantee a feasible number of aircraft to comply with missions, the requirements for each type of aircraft are calculated for the whole planning horizon. Then, this serves as a lower bound on the number of aircraft of each type to create. For the remaining aircraft, their type is chosen randomly taking the weight of the requirements for each type. In order to guarantee a feasible number of aircraft per standard, we chose to generate twice the number of required standards among the aircraft.

Algorithm 5: Initial status generation logic

Data:
$J^{Init} \subset \mathcal{J}$: missions that start at $t = 1$.
$\mathcal{I}_M^{Init} \subset \mathcal{I}$: aircraft in maintenance before $t = 1$.
$\mathcal{I}_j^{Init} \subset \mathcal{I}$: aircraft assigned to mission j before $t = 1$.
$\mathcal{U}\ (a, b)$: discrete uniform distribution between a and b.

begin
  $NV \leftarrow |\mathcal{I}| \times NP$
  $\mathcal{I}^M \leftarrow sample(\mathcal{I}, NV)$
  for $i \in \mathcal{I}_M^{Init}$ **do**
    $NM_i \leftarrow \mathcal{U}(1, M)$
    $Rct_i^{Init} \leftarrow H^{max}$
    $Rft_i^{Init} \leftarrow E^{max}$
  for $i \in \mathcal{I} \setminus \mathcal{I}_M^{Init}$ **do**
    $Rct_i^{Init} \leftarrow \mathcal{U}(0, E^{max})$
    $Rct_i^{I\prime} \leftarrow Rct_i^{Init} + \mathcal{U}(-3, 3)$
    if $Rct_i^{I\prime} < 0$ **then**
      $Rct_i^{I\prime} \leftarrow 0$
    if $Rct_i^{I\prime} > T$ **then**
      $Rct_i^{I\prime} \leftarrow E^{max}$
    $Rft_i^{Init} \leftarrow \lceil Rct_i^{I\prime} \times \frac{H^{max}}{E^{max}} \rceil$
  for $j \in J^{Init}$ **do**
    $\mathcal{I}_j^{Init} \leftarrow sample(\mathcal{I}_j \setminus \mathcal{I}_M^{Init}, R_j)$
    for $i \in \mathcal{I}_j^{Init}$ **do**
      $At_{ij} \leftarrow \mathcal{U}(0, 2 \times MT_j^{min})$

3.6.6 Cluster and service levels

A cluster is a group of missions where every mission has exactly the same requirements (i.e. same type and standard). To explain the model input parameters, the following notations are needed. Let $\mathcal{I}_k \subset \mathcal{I}$ represent the aircraft candidates for cluster k and $QH_k = |\mathcal{I}_k| \times H^{max}$ is the maximum flight hours for the whole set of aircraft in a given cluster k.

Code	Parameter	Value
AN^K	Minimal number of serviceable aircraft per cluster.*	1, 2, 3
AP^K	Percentage of serviceable aircraft per cluster.*	0.05, 0.1, 0.2
HP^K	Percentage of sustainability per cluster.*	0.3, 0.5, 0.7
H_{kt}^{Clust}	Minimal remaining flight hours for cluster k.	$HP^K \times QH_k$
A_{kt}^{Clust}	Minimal serviceable aircraft for cluster k.	$\max\{AP^K Q_k, AN^K\}$

3.7 Experimentation and results

Following the techniques explained in Section 3.6, a base scenario is conceived. Then, several scenarios are generated by changing one control parameter at a time during the generation of instances. Table 3.2 shows the values used to generate the base scenario as well as the derived scenarios. 'Base scenario' corresponds to the default values. 'Studied scenarios' corresponds to the values that are modified to create each scenario.

For each scenario, 50 instances are randomly generated. Among scenarios, the same position of instance always has the same random seed. This is done so that random differences between instances in the same position among different scenarios are as small as possible and comparisons can be more broadly generalized.

The following statistics regarding size, performance and Linear Programming (LP) relaxation are obtained for each scenario. With respect to the size of the problem: the average number of variables (vars), constraints (cons) and non-zero values (non-zero) in the matrix before the solver starts the branching phase. With respect to the performance of the solution method: the number of instances with no integer solution after the time limit (no-int), the minimum (t^{min}), maximum (t^{max}) and average (t^{avg}) solving times and the average gap (g^{avg}) in %.

With respect to the quality of the linear relaxation and the applied cuts, several differences are obtained (in %): the LP relaxation against the best solution found (rinit); the LP relaxation after cuts in the root node against the best solution found (rcuts); and the best solution found after cuts in the root node against the best solution found (icuts). Finally, the nodes in the branch and bound it took to prove optimality in the instances where it is proved (nodes).

All instances are solved using the MIP model described in 3.4. The model is built in Python with the PuLP library and solved with CPLEX 12.8. All tests are run on a 12-core,

Parameter	Name	Base scenario	Studied scenarios
E^{size}	maintenance calendar time size	30	20, 40
E^{max}	maintenance calendar time	60	40, 80
H^{max}	maintenance flight hours	1000	800, 1200
C^{perc}	capacity in percentage of fleet	0.15	0.1, 0.2
M	maintenance duration	6	4, 8
$\|J^P\|$	number of parallel tasks	1	2, 3, 4
$\|\mathcal{T}\|$	number of periods in horizon	60	120, 140
U^{min}	minimum flight hours consumption	0	5, 15, 20
HP^K	minimum rft per cluster	0.5	0.3, 0.7
$\max\{rft\}$	maximize rft at the end	0	1

Table 3.2: Experiments and studied scenarios.

64 GB RAM machine running Linux Fedora 20 with a CPU speed (in MHz) of 2927.000.

The rest of the current Section presents three experiments, each one uses a group of scenarios. Section 3.7.1 presents an analysis on the sensitivity of the model to each parameter of the problem. Section 3.7.2 compares the performance by changing the size of the problem. Finally, Section 3.7.3 evaluates the contribution of using a generated feasible solution as input to the MIP model.

3.7.1 Parameter sensitivity analysis

Experiment 1 consisted in analyzing the sensitivity of the model to changes in its input parameters. Table 3.3 summarizes the performance after solving the model with each scenario. It can be seen that most instances are solved to optimality, although the resolution times are close to the imposed 1-hour limit. The variations in the size of the problem are due to the differences in the solver's pre-solving capabilities given the fact that these scenarios did not change the size of the original problem.

The results obtained show that parameters with influence on execution times and in remaining relative gaps included the ones that regulate the frequency of checks, e.g. the amount of flight hours between checks (H^{max}): increasing available hours, without changing the flight load, will dramatically reduce solution times. This is seen in Figure 3.2, where scenarios are shown in the X-axis while the times are shown in the Y-axis. This modification also has an impact on whether a solution is feasible or not (see Table 3.3). Another parameter that had a very sensible impact is the minimum amount of sustainability per cluster HP^K. The impact of both of these parameters can also be confirmed via the difference in the average needed nodes to reach optimality, shown in Table 3.4.

Figure 3.3 shows the gaps obtained (in the Y-axis) for each scenario (in the X-axis).

The minimum consumption of flight hours per period U^{min} makes the problem significantly harder to solve. This can be confirmed both via the remaining gaps, solving times and with

case	t^{min}	t^{avg}	non-zero	vars	cons	no-int	inf	g^{avg}
HP^K=0.3	1.8	5.2	50976.5	4275.5	6273.0	0	0	0.0
H^{max}=1200	2.1	76.7	51030.3	4295.0	6298.2	0	0	0.1
E^{size}=20	1.6	172.8	29772.8	3826.1	5120.3	0	3	0.2
E^{size}=40	4.0	266.8	64152.6	4496.1	6994.6	0	1	0.4
base	2.2	310.6	51167.1	4310.7	6315.9	0	1	0.3
E^{max}=40	8.1	530.9	68612.7	4525.5	7632.9	0	0	0.2
E^{max}=80	1.5	1250.6	28257.9	3877.8	5010.4	0	3	1.9
HP^K=0.7	80.7	1746.9	50805.8	4393.9	6320.6	0	42	2.9
H^{max}=800	4.4	2168.5	51219.7	4327.2	6327.2	0	5	2.7
U^{min}=5	24.6	2650.3	60950.1	5525.3	8583.6	0	3	4.3
U^{min}=20	3600.0	3600.0	53562.3	5379.8	8149.6	25	8	5.2
U^{min}=15	3600.0	3600.0	60716.4	5529.0	8573.6	10	6	6.3

Table 3.3: Experiment 1: summary per scenario sorted by average solving time.

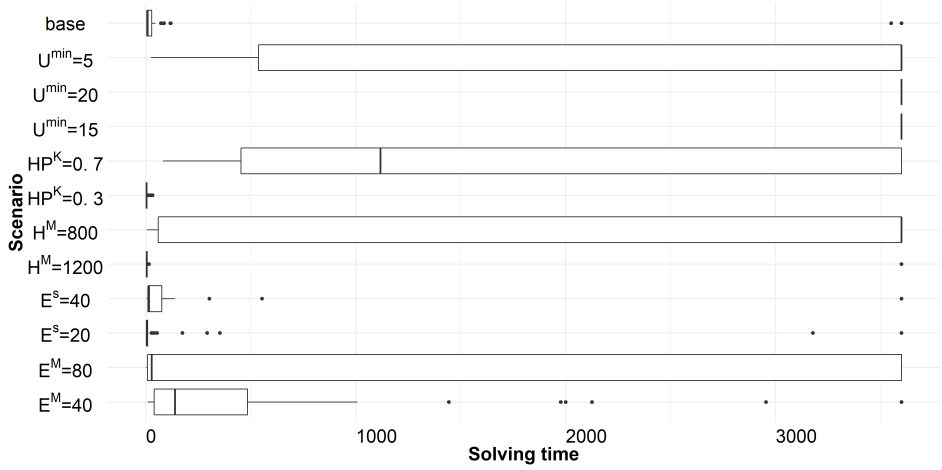

Figure 3.2: Box-plot showing the distribution of solution times for each of the instances of Experiment 1.

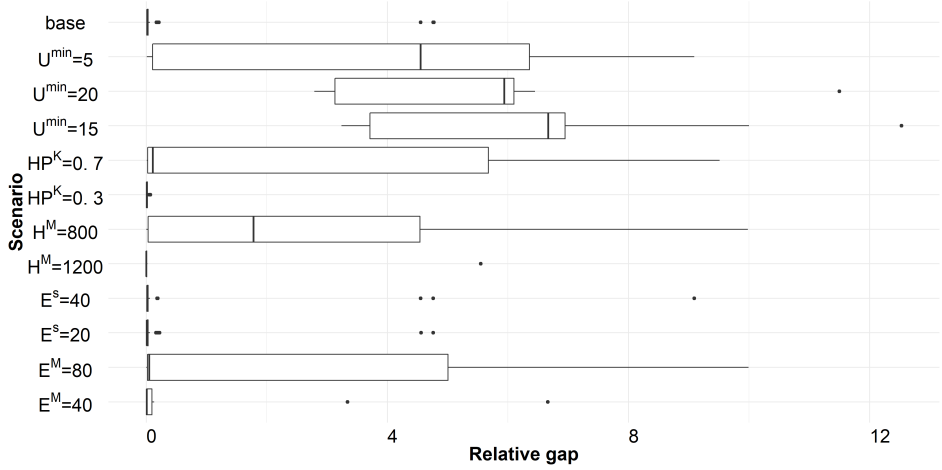

Figure 3.3: Box-plot showing the distribution of relative gaps for each of the instances of Experiment 1.

several instances where a feasible solution is not found after 1 hour (table 3.3). The effect is evident even when adding a relatively small quantity of consumption hours (U^{min}=5), although a greater impact is correlated with a higher minimum consumption. In addition, Table 3.3 shows that the solver's pre-processor is less able to reduce the problem size in these scenarios than in most of the other ones. Table 3.4 shows how, although the initial relaxation is particularly bad for these scenarios, the cuts phase (helped by a manual configuration of the solver) significantly improves the relaxation.

3.7.2 Problem size sensitivity analysis

Experiment 2 studied changes in the problem size and the objective function. First, the horizon is increased in size by changing the amount of planning periods. Second, the number of parallel tasks is increased with an equivalent increase in the size of the fleet. Lastly, an objective function that maximizes the final state in addition to minimizing the number of checks is tested.

By activating maximization of the end state for the whole fleet ($\max\{rft\}$=1), the efficiency of the solving process, measured in solving times, declines significantly. Another condition with a similar effect is increasing the size of the planning horizon ($|T|$=140). Both scenarios seem to share the same difficulty.

A similar effect is detected when increasing the number of parallel missions $|J^P|$ and the size of the fleet proportionally. This effect can be explained by the fact that the model size grows in proportion to the number of parallel missions (see 'non-zero' column in Table 3.5).

To summarize, although the model performance seems to deteriorate with larger instances, the effect in resulting gaps seems to keep a lineal relationship with regards to $|J^P|$ and $|T|$, for the studied scenarios and the resulting gaps are still acceptable. See Figure 3.4, where

case	rinit	rcuts	icuts	nodes
base	15.9	1.2	8.0	4335.9
E^{size}=20	2.1	0.7	2.9	2999.0
E^{size}=40	17.5	2.5	23.3	4384.3
E^{max}=40	41.0	5.0	7.3	15172.6
E^{max}=80	4.5	2.3	7.4	46174.8
H^{max}=1200	17.1	0.1	3.5	121.2
H^{max}=800	10.3	5.0	17.8	89602.2
HP^{K}=0.3	16.7	0.1	3.1	372.8
HP^{K}=0.7	16.7	9.1	13.5	26534.8
U^{min}=15	23.8	7.4	8.0	
U^{min}=20	20.1	6.3	3.5	
U^{min}=5	18.2	6.8	22.3	13642.6
C^{perc}=0.2	15.1	1.0	8.8	1386.2

Table 3.4: Experiment 1: mean performance of relaxations per scenario (in % difference).

case	t^{min}	t^{avg}	non-zero	vars	cons	no-int	inf	g^{avg}
base	2.2	310.6	51167.1	4310.7	6315.9	0	1	0.3
$\|T\|$=120	19.9	376.0	88668.3	5815.8	9161.6	0	6	0.8
$\|J^P\|$=2	20.1	1313.9	101572.8	8317.9	12266.2	0	5	0.8
$\|T\|$=140	44.5	1651.0	115910.9	6738.7	11081.5	0	4	3.3
max$\{rft\}$=1	7.5	2198.8	51167.1	4310.7	6315.9	0	1	1.6
$\|J^P\|$=3	62.7	2723.7	157213.7	12907.6	18915.8	1	8	2.5
$\|J^P\|$=4	114.7	3228.6	209747.2	17045.4	24947.4	3	9	2.6

Table 3.5: Experiment 2: summary per scenario sorted by average solving time.

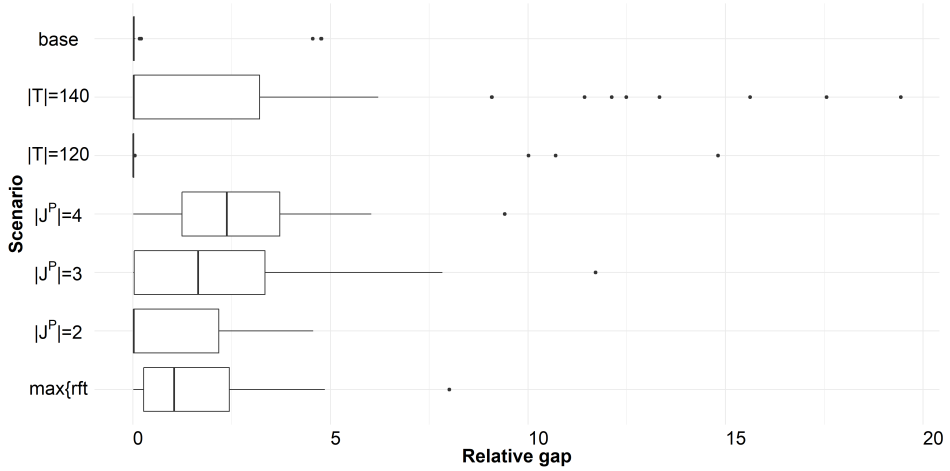

Figure 3.4: Box-plot showing the distribution of relative gaps for each of the instances of Experiment 2.

case	rinit	rcuts	icuts	nodes
base	15.9	1.2	8.0	4335.9
$\|J^P\|$=2	15.1	2.1	9.9	10518.1
$\|J^P\|$=3	16.1	4.0	17.9	11876.2
$\|J^P\|$=4	15.2	3.8	11.4	9875.4
$\|T\|$=120	24.8	10.4	21.7	5239.8
$\|T\|$=140	22.4	15.6	23.7	23023.9
max$\{rft\}$=1	7.2	3.3	59.3	142925.7

Table 3.6: Experiment 2: mean performance of relaxations per scenario (in % difference).

each scenario (in the X-axis) shows its gap in the Y-axis.

Table 3.6 how the quality of the cuts phased decreases with the size of the planning horizon, in relaxation quality as in integer solution quality. Also, the number of nodes needed to find an optimal solution considerably increases in the ($|T|$=140) scenario. This is possible due to the fact that aircraft need a third maintenance in these circumstances and the possible maintenance combinations grow in a combinatorial sense. Lastly, guaranteeing an optimal solution appears to prove difficult when considering the final state in the objective function, as seen in the average number of nodes needed.

3.7.3 Heuristic comparison

Experiment 3 studied the impact of using the heuristic presented in Section 3.5 to generate fast feasible solutions for instances. These solutions are, firstly, compared to the best available solutions obtained using the mathematical model (usually optimal) and, later, used as input in order to warm-start the solution process by the solver.

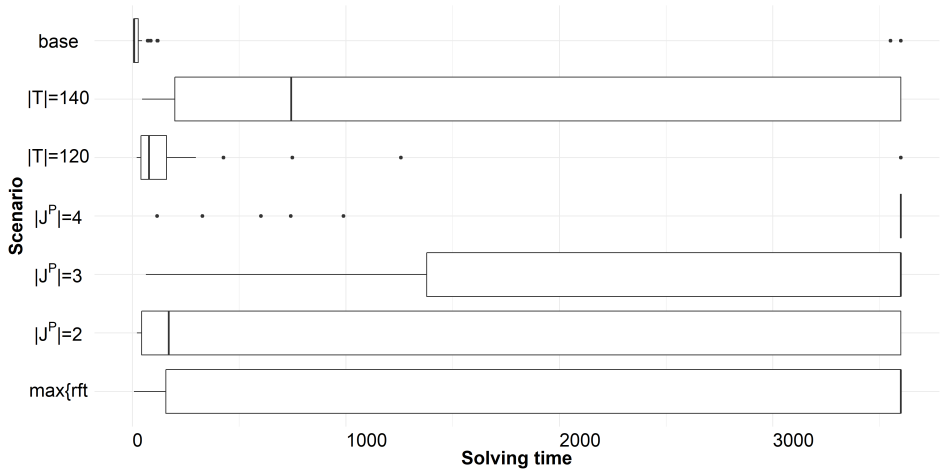

Figure 3.5: Box-plot showing the distribution of solution times for each of the instances of Experiment 2.

case	t_M^{avg}	t_H^{avg}	t_{M+H}^{avg}	g_M^{avg}	g_{M+H}^{avg}	$\%Dif_H$	$\%Init_H$
base	247.1	23.7	183.0	0.2	0.1	22.0	95.9
U^{min}=5	2417.5	92.8	2364.2	3.8	3.4	19.0	74.5
$\|J^P\|$=2	706.9	86.2	976.9	0.4	0.7	22.0	64.4
$\|J^P\|$=3	1979.5	217.1	1712.3	1.6	1.0	23.0	34.1
$\|J^P\|$=4	3102.1	401.1	2963.7	1.1	1.1	19.5	18.4
$\|T\|$=120	247.6	47.8	305.0	0.5	0.8	19.7	75.0
$\|T\|$=140	1493.3	65.8	1211.7	2.9	1.9	23.4	87.0
max$\{rft\}$=1	2214.1	23.7	2194.2	1.6	1.7	89.9	95.9

Table 3.7: Comparison of all instances where a feasible solution is found by the heuristic in a selected set of difficult scenarios.

Table 3.7 shows several ways to properly measure the heuristic's performance: (1) the average time it takes to find an initial solution (t_H^{avg}); (2) the relative distance between the solution found by the heuristic and the best solution found with the MIP model ($\%Dif_H$); (3) the probability of finding an initial solution in a short time (10 minutes) ($\%Init_H$); (4) the difference between the original average solving times t_M^{avg} (average gaps g_M^{avg}) and the ones after providing an initial solution t_{H+M}^{avg} (g_{H+M}^{avg}).

The relative quality seems to depend particularly on the type of objective function being used (max$\{rft\}$=1 scenario) but not so on the size of the problem. On the other hand, the probability of finding a solution appears to depend on the number of parallel tasks at any given time. Finally, the average times to find a solution do increase with problem size but not in a uncontrollable way, especially for increases in planning horizon size.

Secondly, feeding an initially generated solution to the solver slightly increases the solving process, both in resolution times and in gap, although not in a meaningful quantity. Taking into account the heuristic performance and impact on resolution, it can be concluded that it

is particularly useful for longer planning horizons, where the performance remains high and the impact is also greatest.

Finally, since the solver permitted it, giving a nearly-feasible solution to the solver is tested. Usually this solution is then repaired by the solver during the cutting phase. No gains in solution times and gap are observed for these cases.

3.8 Conclusions

This chapter presented a new MIP formulation for the long-term Flight and Maintenance Planning problem for military aircraft. Its performance is measured by solving an array of scenarios inspired by real French Air Force needs.

Compared to the existing literature, the problem studied includes several new constraints while still managing to solve fairly large instances. Also, a complexity proof is presented.

The study shows that the mathematical model's performance is quite robust with respect to increases in fleet size, number of missions and the size of the planning horizon. On the other hand, adding fixed additional consumptions outside of missions proves challenging.

In terms of performance, gains in resolution time are obtained by developing a construction heuristic that provides starting solutions for the cases where an integer solution is not easily obtained by the model. It is shown to be potentially useful in scenarios with long planning horizons.

With respect to extending the model, additional constraints from real world application, such as long-term storage of grounded aircraft, can be incorporated.

In order to better integrate long term schedules with the existing medium- and short-term maintenance planning, a matheuristic that alternates between the two problems could potentially satisfy the needs of the different scopes with a good quality solution that takes several types of aircraft maintenance into account simultaneously.

Regarding uncertainty treatment, explicit ways to measure the stochastic nature of the input parameters can be implemented. For example, by using robust optimization or stochastic programming in order to guarantee the feasibility of the solution even in extreme scenarios.

Next chapter will show an alternative solution approach based on Machine Learning aimed at obtaining a better performance while producing a more balanced maintenance planning, and thus, more robust to small changes.

Chapter 4

Alternative MIP model, valid bounds and learned constraints

Contents

In this chapter, we propose a new solution approach based on a new Mixed Integer Program and the use of both valid cuts generated on the basis of initial conditions and learned cuts based on the prediction of certain characteristics of optimal or near optimal solutions. These learned cuts are generated by training a Machine Learning model on the input data and results of 5000 instances. This approach helps to reduce the solution time with little losses in

optimality and feasibility in comparison to alternative matheuristic methods. The obtained experimental results show the benefit of a new way of adding learned cuts to problems based on predicting specific characteristics of solutions.

The chapter is structured as follows. Section 4.1 formulates an alternative MIP model for the MFMP problem. Section 4.2 presents valid cuts and Section 4.3 presents learned bounds and constraints for this MIP model. Section 4.4 explains the experimentation methodology and Section 4.5 shows the results of those experiments. Finally, Section 4.6 summarizes the conclusions.

The contributions of this chapter were presented in the following publications: F. Peschiera, R. Dell, J. Royset, A. Haït, N. Dupin, and O. Battaïa. A novel solution approach with ML-based pseudo-cuts for the Flight and Maintenance Planning problem. *OR Spectrum*, pages 1–30, jun 2020. ISSN 0171-6468. doi: 10.1007/s00291-020-00591-z. URL `http://link.springer.com/10.1007/s00291-020-00591-z`; F. Peschiera, N. Dupin, O. Battaïa, and A. Haït. An alternative mip formulation for the military flight and maintenance planning problem. In *Congrès annuel de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)*, pages 1–2, Montpellier, FR, 2020. URL `https://oatao.univ-toulouse.fr/26033/`.

4.1 Alternative mathematical formulation

This section presents the base model: the decision variables, constraints and objective function. With respect to the model on chapter 3.4, it models both mission and maintenance assignments as start-stop assignments.

4.1.1 Variables

The following binary decision variables prescribe the assignment of missions and checks to aircraft.

- $a_{ijtt'}$ has value one if aircraft i starts an assignment to mission j at the beginning of period t and finishes at the end of period t', zero otherwise.
- $m_{itt'}$ has value one if aircraft i starts a check at the beginning of period t and then starts the next check at the beginning of period t' or does not have a second check ($t' = T$), zero otherwise.

The following continuous auxiliary variables prescribe the status of each aircraft or group of aircraft.

- rft_{it} remaining flight time for aircraft $i \in I$ at the end of period $t \in \mathcal{T}$.
- e^A_{kts} Deviation in serviceability for cluster k at end of t in interval s.
- e^H_{kts} Deviation in sustainability for cluster k at end of t in interval s.
- e^C_{ts} Deviation in maintenance capacity at end of t in interval s.

4.1.2 Objective function and constraints

The main objective function (4.1) expresses the difference between the total deviation from all goals on serviceability, sustainability and maintenance capacity and the mean starting time of the second maintenance.

$$\begin{aligned}\text{Min} \sum_{\substack{k\in\mathcal{K},\\ t\in\mathcal{T},\\ s\in\mathcal{S}}} PA_s \times e^A_{kts} + \sum_{\substack{k\in\mathcal{K},\\ t\in\mathcal{T},\\ s\in\mathcal{S}}} PH_s \times e^H_{kts} + \sum_{t\in\mathcal{T},s\in\mathcal{S}} PC_s \times e^C_{ts} \\ - P2M \sum_{\substack{i\in\mathcal{I},\\ t\in\mathcal{T}_i^{MInit},\\ t'\in\mathcal{T}_t^{M+}}} m_{itt'} \times t' \end{aligned} \tag{4.1}$$

$$\sum_{\substack{i\in\mathcal{I},\\ (t_1,t_2)\in\mathcal{TTT}_t}} m_{it_1t_2} + N_t^{Init} \leq C^{max} + \sum_{s\in\mathcal{S}} e^C_{ts} \qquad t\in\mathcal{T} \tag{4.2}$$

$$\sum_{\substack{i\in\mathcal{I}_j,\\ (t_1,t_2)\in\mathcal{TTJ}_{jt}}} a_{ijt_1t_2} \geq R_j \qquad j\in\mathcal{J}, t\in\mathcal{T}_j \tag{4.3}$$

$$\sum_{\substack{(t_1,t_2)\in\\ \mathcal{TTT}_t}} m_{it_1t_2} + \sum_{\substack{j\in\\ \mathcal{J}_t\cap\mathcal{J}_i}} \sum_{\substack{(t_1,t_2)\in\\ \mathcal{TTJ}_{jt}}} a_{ijt_1t_2} \leq 1 \qquad t\in\mathcal{T}, i\in\mathcal{I} \tag{4.4}$$

Constraints (4.2) limit the number of unpenalized simultaneous checks. Constraints (4.3) enforce aircraft mission requirements. Constraints (4.4) restrict each aircraft to at most one assignment each period.

$$\sum_{(j,t,t')\in\mathcal{JTT}_{i1t_1}} a_{ijtt'}H'_{jtt'} + U'_{1t_1} \leq Rft_i^{Init} + H^{max}(1-m_{it_1t_2}) \qquad i\in\mathcal{I}, t_1\in\mathcal{T}_i^{MInit}, t_2\in\mathcal{T}_{t_1}^{M+} \tag{4.5}$$

$$\sum_{(j,t,t')\in\mathcal{JTT}_{it_1t_2}} a_{ijtt'}H'_{jtt'} + U'_{t_1t_2} \leq H^{max} + H^{max}(1-m_{it_1t_2}) \qquad i\in\mathcal{I}, t_1\in\mathcal{T}_i^{MInit}, t_2\in\mathcal{T}_{t_1}^{M+} \tag{4.6}$$

$$\sum_{(j,t,t')\in\mathcal{JTT}_{it_2T}} a_{ijtt'}H'_{jtt'} + U'_{t_2T} \leq H^{max} + H^{max}(1-m_{it_1t_2}) \qquad i\in\mathcal{I}, t_1\in\mathcal{T}_i^{MInit}, t_2\in\mathcal{T}_{t_1}^{M+} \tag{4.7}$$

Constraints (4.5) - (4.7) limit the total flight hours of each aircraft before the first check,

between checks and after the second check.

$$\sum_{\substack{i\in\mathcal{I}_k,\\(t_1,t_2)\in\mathcal{TT}_t}} m_{it_1t_2} + N^{Clust}_{kt} \leq A^{Clust}_{kt} + \sum_{s\in\mathcal{S}} e^{A}_{kts} \qquad k\in\mathcal{K}, t\in\mathcal{T} \tag{4.8}$$

$$\sum_{i\in\mathcal{I}_k} rft_{it} \geq H^{Clust}_{kt} + \sum_{s\in\mathcal{S}} e^{H}_{kts} \qquad k\in\mathcal{K}, t\in\mathcal{T} \tag{4.9}$$

Constraints (4.8) limit the number of unpenalized aircraft from cluster k simultaneously undergoing a check in period t. This measures serviceability. Constraints (4.9) record any deviation from the remaining flight hour requirement for each cluster k and each period t.

$$\begin{aligned} rft_{it} \leq\ & rft_{i(t-1)} + H^{max} \sum_{\substack{(t_1,t_2)\in\\ \mathcal{TT}_t}} m_{it_1t_2} \\ & - U^{min} - \sum_{\substack{j\in\mathcal{J}_t\cap\mathcal{J}_i,\\(t_1,t_2)\in\mathcal{TTJ}_{jt}}} a_{ijt_1t_2}(H_j - U^{min}) \qquad t\in\{1,...,T\}, i\in\mathcal{I} \end{aligned} \tag{4.10}$$

$$rft_{it} = Rft^{Init}_i \qquad t=0, i\in\mathcal{I} \tag{4.11}$$

$$rft_{it} \geq H^{max} \sum_{\substack{(t_1,t_2)\in\\ \mathcal{TT}_t}} m_{it_1t_2} \qquad t\in\mathcal{T}, i\in\mathcal{I} \tag{4.12}$$

Constraints (4.10) - (4.12) define the remaining flight time for each aircraft i and each period t resulting from planned mission and maintenance assignments.

$$\sum_{\substack{t\in\mathcal{T}_i^{MInit},\\ t'\in\mathcal{T}_t^{M+}}} m_{itt'} = 1 \qquad i\in\mathcal{I} \tag{4.13}$$

Constraints (4.13) require maintenance assignments for each aircraft. Each aircraft is assigned one (if $t'=T$) or two checks over the whole planning horizon.

$$\sum_{(t_1,t_2)\in\mathcal{TTJ}_{jt}} a_{ijt_1t_2} \geq 1 \qquad i\in\mathcal{I}, j\in\mathcal{J}_i, t\in\mathcal{A}^{Init}_{ij} \tag{4.14}$$

Constraints (4.14) require aircraft to comply with pre-assigned tasks during periods in $\mathcal{A}^{Init}_{ij}$.

$$a_{ijtt'} \in \mathbb{B} \qquad i \in \mathcal{I}, j \in \mathcal{J}_i, (t,t') \in \mathcal{TT}_j \tag{4.15}$$
$$m_{itt'} \in \mathbb{B} \qquad i \in \mathcal{I}, t \in \mathcal{T}_i^{M_{Init}}, t' \in \mathcal{T}_t^{M+} \tag{4.16}$$
$$rft_{it} \in [0, H^{max}] \qquad t \in \{0, ..., T\}, i \in \mathcal{I} \tag{4.17}$$
$$e^A_{kts} \in [0, UA_s] \qquad k \in \mathcal{K}, t \in \mathcal{T}, s \in \mathcal{S} \tag{4.18}$$
$$e^H_{kts} \in [0, UH_s] \qquad k \in \mathcal{K}, t \in \mathcal{T}, s \in \mathcal{S} \tag{4.19}$$
$$e^C_{ts} \in [0, UC_s] \qquad t \in \mathcal{T}, s \in \mathcal{S} \tag{4.20}$$

Constraints (4.15) - (4.20) declare all the decision variables' bounds and domains.

4.2 Deterministic bounds and valid cuts

This section presents various levels of cuts and bounds obtained from the initial status of the fleet and applied to the MIP model in Section 4.1. First, each individual aircraft is analyzed: at each period (Section 4.2.1), at the start of the planning horizon (Section 4.2.2), and at the end of the planning horizon (Section 4.2.3). Then, Section 4.2.4 analyzes each group of aircraft by type and period. Finally, Section 4.2.5 deals with the whole fleet at each period.

4.2.1 Accumulated checks per aircraft and period

For each period t, and using the aircraft initial states, we calculate the minimum and maximum number of checks that an aircraft could have already started and ended at the start of the period.

We define $TM1_i^{min} \in \mathcal{T}$ and $TM1_i^{max} \in \mathcal{T}$ as, respectively, the minimum and maximum periods for starting the first check for aircraft i. Analogously, $TM2_i^{min} \in \mathcal{T}$ and $TM2_i^{max} \in \mathcal{T}$ are the minimum and maximum periods for starting the second check for aircraft i.

Using previous notation,

$$TM1_i^{min} = \min_t \{t \in \mathcal{T}_i^{M_{Init}}\}$$
$$TM1_i^{max} = \min\{\max_t \{t \in \mathcal{T}_i^{M_{Init}}\}, \lfloor Rft_i^{Init}/U^{min} \rfloor\}$$
$$TM2_i^{min} = \min_t \{t \in \mathcal{T}^M_{TM1_i^{min}}\}$$
$$TM2_i^{max} = \min\{\max_t \{t \in \mathcal{T}^M_{TM1_i^{max}}\}, TM1_i^{max} + M + \lfloor H^{max}/U^{min} \rfloor\}$$

To guarantee a feasible solution, the ranges of periods to start the first and second checks

need not be empty:

$$TM1_i = [TM1_i^{min}, TM1_i^{max}] \neq \emptyset$$
$$TM2_i = [TM2_i^{min}, TM2_i^{max}] \neq \emptyset$$

Lower and upper bounds for the accumulate number of checks started ($M^{Acc}S_{it}^{min}$, $M^{Acc}S_{it}^{max}$) and finished ($M^{Acc}F_{it}^{min}$, $M^{Acc}F_{it}^{max}$) can be pre-calculated for each period using these units. We name the range $M^{Acc}S_{it}^{b}$ to represent both the lower level and upper level of the range of accumulated starts by replacing the terms *min* and *max* by *b*, as in *bound*.

$$M^{Acc}S_{it}^{min} = \begin{cases} 0 & t \leq TM1_i^{max} \\ 1 & TM1_i^{max} < t \leq TM2_i^{max} \\ 2 & t > TM2_i^{max} \end{cases}$$
$$M^{Acc}S_{it}^{max} = \begin{cases} 0 & t < TM1_i^{min} \\ 1 & TM1_i^{min} \leq t < TM2_i^{min} \\ 2 & t \geq TM2_i^{min} \end{cases}$$

$$M^{Acc}F_{it}^{b} = M^{Acc}S_{i(t-M)}^{b} \qquad i \in \mathcal{I}, t \in \mathcal{T}, b \in \{min, max\}$$

4.2.2 Mission assignments at the start of the horizon for each aircraft

Before aircraft i can undergo its first check (i.e. at $t_i^s = TM1_i^{min} - 1$), flight hours for assigned missions must not exceed its initial remaining flight hours, as represented by cuts (4.21) (a subset of constraints (4.5) in section 4.1).

$$\sum_{\substack{(j,t,t') \in \\ \mathcal{JTT}_{i1t_i^s}}} a_{ijtt'} H'_{jtt'} + U'_{1t_i^s} \leq Rft_i^{Init} \qquad i \in \mathcal{I}, t_i^s = TM1_i^{min} - 1 \qquad (4.21)$$

Clearly, some mission assignments can be discarded because the flight hour usage for those assignments alone, together with the minimum default usage, is more than the initial remaining time:

$$a_{ijtt'} = 0 \qquad i \in \mathcal{I}, (j,t,t') \in \mathcal{JTT}_{i1t_i^s} \mid Rft_i^{Init} < H'_{jtt'} + U'_{1t_i^s} \tag{4.22}$$

4.2.3 Accumulated checks at the end of the horizon per aircraft

Each aircraft's lower bound and upper bound of performed checks in the last period (T), is: [1, 1], [1, 2] or [2, 2], e.g., [1, 2] implies a lower bound of 1 and an upper bound of 2. Let I^{1M} be the set of aircraft with range [1, 1] and I^{2M} the set of aircraft with range [2, 2].

With certainty on the number of checks, it is possible to enforce this on the decision variables using the following cuts.

$$I^{1M} = \{i \in \mathcal{I} \mid M^{Acc}S_{iT}^{max} = 1\}$$
$$I^{2M} = \{i \in \mathcal{I} \mid M^{Acc}S_{iT}^{min} = 2\}$$

$$\sum_{t \in \mathcal{T}_i^{M_{Init}}} m_{itT} = 1 \qquad i \in I^{1M} \tag{4.23}$$

$$m_{itT} = 0 \qquad i \in I^{2M}, t \in \mathcal{T}_i^{M_{Init}} \tag{4.24}$$

4.2.4 Accumulated checks per aircraft type and period

Both $M^{Acc}F_{it}^b$ and $M^{Acc}S_{it}^b$ from Section 4.2.1 can be aggregated by aircraft type y.

$$YM_1^{Acc}S_{yt}^b = \sum_{i \in \mathcal{I}_y} M^{Acc}S_{it}^b \qquad t \in \mathcal{T}, y \in \mathcal{Y}, b \in \{min, max\}$$
$$YM_1^{Acc}F_{yt}^b = \sum_{i \in \mathcal{I}_y} M^{Acc}F_{it}^b \qquad t \in \mathcal{T}, y \in \mathcal{Y}, b \in \{min, max\} \tag{4.25}$$

By using the required mission assignments, we calculate the number of checks that fit until period t (aircraft that are in a mission cannot be in maintenance). Because each mission demands a specific type of aircraft, this bound is made at the aircraft type level. Let JR_{jt}^{Acc} represent the accumulated required number of assignments (in aircraft-periods) for mission j until the end of period t. Let IR_{yt}^{Acc} be the number of aircraft-periods available for maintenance for type y up to period t. Then, $YM_2^{Acc}F_{yt}^{max}$ is an upper bound on the number of checks that can be finished until the end of period t for all aircraft of type y.

$$
\begin{aligned}
JR_{jt}^{Acc} &= \sum_{t' \in \mathcal{T}_j | t' \leq t} R_j && t \in \mathcal{T}, j \in \mathcal{J} \\
IR_{yt}^{Acc} &= |IY_y| \times t - \sum_{j \in \mathcal{J}_y} JR_{jt}^{Acc} && t \in \mathcal{T}, y \in \mathcal{Y} \\
YM_2^{Acc} F_{yt}^{max} &= \lfloor \frac{IR_{yt}^{Acc}}{M} \rfloor && t \in \mathcal{T}, y \in \mathcal{Y}
\end{aligned} \tag{4.26}
$$

By using the mission required flight hours over time, we can calculate how many checks we need until period t. Let YH_{yt}^{Acc} be the sum of all flight-hour needs of missions of type y until the end of period t. This demand of flight hours can be subtracted from the initial remaining flight time of the group of aircraft and then divided over the H^{max} flight hours each check provides. Thus, we get a lower bound $YM_2^{Acc} F_{yt}^{min}$ on the number of checks we need to do for aircraft of type y until period t.

$$
\begin{aligned}
YH_{yt}^{Acc} &= \sum_{j \in \mathcal{J}_y} (H_j - U^{min}) \times JR_{jt}^{Acc} + |\mathcal{I}| U'_{1t} && t \in \mathcal{T}, y \in \mathcal{Y} \\
YM_2^{Acc} F_{yt}^{min} &= \lceil \frac{YH_{yt}^{Acc} - \sum_{i \in IY_y} Rft_i^{Init}}{H^{max}} \rceil && t \in \mathcal{T}, y \in \mathcal{Y}
\end{aligned} \tag{4.27}
$$

So, in this way, we arrive to two lower bounds (4.25 and 4.27) and two upper bounds (4.25 and 4.26) per period and aircraft type. We then get the maximum and the minimum respectively to get bounds on the number of checks until period t. $YM^{Acc} F_{yt}^{b}$ represents the bound b in the number of total checks since the beginning of the planning horizon for all aircraft of type y until period t.

$$
\begin{aligned}
YM^{Acc} F_{yt}^{min} &= \max\{YM_1^{Acc} F_{yt}^{min}, YM_2^{Acc} F_{yt}^{min}\} \\
YM^{Acc} F_{yt}^{max} &= \min\{YM_1^{Acc} F_{yt}^{max}, YM_2^{Acc} F_{yt}^{max}\}
\end{aligned}
$$

Let $QM_{tt_1t_2}^{num}$ represent the number of finished checks before the end of period t given t_1 and t_2 are the start of the first and second checks, respectively.

$$
QM_{tt_1t_2}^{num} = \begin{cases} 0 & t < t_1 + M \\ 1 & t_1 + M \leq t < t_2 + M \\ 2 & t \geq t_2 + M \end{cases}
$$

This provides the following cut:

$$YM^{Acc}F^{min}_{yt} \leq \sum_{\substack{i\in\mathcal{I}_y,\\ t_1\in\mathcal{T}_i^{MInit},\\ t_2\in\mathcal{T}_{t_1}^{M+}}} m_{it_1t_2} \times QM^{num}_{tt_1t_2} \leq YM^{Acc}F^{max}_{yt} \qquad t\in\mathcal{T}, y\in\mathcal{Y} \tag{4.28}$$

Cuts (4.28) limit the starts of checks of aircraft of type y in order to have the number of finished checks to fall between the $YM^{Acc}F^{b}_{yt}$ bounds.

4.2.5 Accumulated checks per period

The bounds on accumulated checks by type and period defined in section 4.2.4 can be aggregated into the whole fleet.

$$TM^{Acc}_1 S^b_t = \sum_{y\in\mathcal{Y}} YM^{Acc}S^b_{yt} \qquad t\in\mathcal{T}, b\in\{min,max\}$$
$$TM^{Acc}_1 F^b_t = \sum_{y\in\mathcal{Y}} YM^{Acc}F^b_{yt} \qquad t\in\mathcal{T}, b\in\{min,max\}$$

Additionally, the maintenance capacity together with the maintenance duration offer an upper bound on the maximum number of checks that can be finished until a given period t.

$$TM^{Acc}_2 F^{max}_t = \lfloor \frac{t}{M} \rfloor \times C^{max} \qquad t\in\mathcal{T}$$

Following the same path as in the previous section, we calculate the net upper bound for maintenance per period.

$$TM^{Acc}F^{max}_t = \min\{TM^{Acc}_1 F^{max}_t, TM^{Acc}_2 F^{max}_t\}$$

Which provides the following cut:

$$TM^{Acc}F^{min}_t \leq \sum_{\substack{i\in\mathcal{I},\\ t_1\in\mathcal{T}_i^{MInit},\\ t_2\in\mathcal{T}_{t_1}^{M+}}} m_{it_1t_2} \times QM^{num}_{tt_1t_2} \leq TM^{Acc}F^{max}_t \qquad t\in\mathcal{T} \tag{4.29}$$

Cuts (4.29) limit the starts of checks of all aircraft in order to have the total number of finished checks to fall between the $TM^{Acc}F_t^b$ bounds.

4.3 Learned bounds and constraints

Learned bounds, as we present them, are similar to deterministic bounds in terms of implementation: they can be both represented by an additional set of constraints or a reduction in the set of decision variables. The main difference is that the latter (as presented in Section 4.2) are guaranteed not to remove valid solutions from the solution space while the former can, and often do, remove valid solutions. The reason for this is that learned bounds do not focus in the feasible solution space itself but in the statistical distribution of the optimal or near optimal solution in that space. This, in turn, permits learned bounds to drastically reduce the solution space even if there is a chance of removing an optimal solution.

Whenever relevant, we use notation similar to that used in Larsen et al. [104]. Let a particular instance of our problem be represented by the input vector x and the optimal solution to our problem by $y^*(x) :\equiv arg\ \min_{y\in\mathcal{Y}(x)} C(x,y)$. Where $C(x,y)$ and $\mathcal{Y}(x)$ are the cost function and the solution space respectively. Finally, let $g_n(y)\ \forall n \in \{1,..,N\}$ represent N features from the solution y. Our goal is, then, to predict $g_n(y^*)$ for each $n \in \{1,..,N\}$ by means of the input vector x and a function $\hat{g}_n(x)$ learned from matching features on both input and output data.

Each predicted feature n of an instance's optimal solution generates one or more pseudo-cuts that reduce the solution space $\mathcal{Y}(x)$. This reduction removes valid solutions from the solution space and can potentially remove optimal solutions. Following notation in [111], we refer to these pseudo-cuts as "learned constraints". The result of applying all learned constraints thus creates a new solution space $\mathcal{Y}'(x)$. Let $\hat{y}^*(x)$ be the optimal solution for this new problem, i.e., $\hat{y}^*(x) = arg\ \min_{y\in\mathcal{Y}'(x)} C(x,y)$.

It would be desirable that the following holds:

$$C(x,\hat{y}^*(x)) \approx C(x,y^*(x))$$

In other words we allow an invalid reduction of the original solution space as long as the optimal objective function value of the reduced solution space $\mathcal{Y}'(x)$ is not too far from the optimal objective function value of the original solution space.

In what is left of this section, we first explain the general case of constraining maintenance cycles in 4.3.1 and we then present $g_n(y)$ and the method to calculate $\hat{g}_n(x)$ using a supervised learning algorithm in 4.3.2.

4.3.1 Constraining maintenance cycles

We seek to limit the combinations of possible maintenance cycles (check patterns) for each aircraft in the fleet. This decision is motivated by: (1) having a check frequency as homogeneous as possible among similar aircraft, presented as a second objective in Section 3.1; (2) improving solution time, by reducing the number of decision variables; and (3) permitting the creation of a forecasting method that provides the planner with information about the optimal solution of a given instance without having to solve it.

Let $\mathcal{H}$ be the set of constraints to add as learned constraints and $\mathcal{D}$ the set of variables $m_{itt'}$. For each $h \in \mathcal{H}$, let $A^h \in \mathbb{R}^{|\mathcal{D}|}$ and $b^h \in \mathbb{R}$. Finally, let $\mathcal{D}^h \subset \mathcal{D}$ be a selected subset of variables used in constraint $h \in \mathcal{H}$.

Equation 4.30 shows the generic formulation of every possible learning constraint $h \in \mathcal{H}$.

$$\sum_{m \in \mathcal{D}^h} A^h_m \times m \geq b^h \qquad h \in \mathcal{H} \tag{4.30}$$

This formulation includes stronger constraints $\mathcal{H}' \subset \mathcal{H}$ of the type seen in Equation 4.31.

$$m = 0 \qquad h \in \mathcal{H}', m \in \mathcal{D}^h \tag{4.31}$$

Other special case where $A^h \in \mathbb{B}^{|\mathcal{D}|}$ creates cover-cut-like constraints $\mathcal{H}'' \subset \mathcal{H}$ of the type seen in Equation 4.32.

$$\sum_{m \in \mathcal{D}^h | A^h_m = 1} m \geq b^h \qquad h \in \mathcal{H}'' \tag{4.32}$$

4.3.2 Predicting maintenance cycle constraints

One key difference among the check patterns for a given aircraft is the distance between the two checks. We define the distance between two checks as the number of periods that take place between the end of the first check and the beginning of the second check. The minimum (maximum) distance between two checks is E^{min} (E^{max}) periods (see Section 3.3.1).

Because the objective function encourages the model to plan the second check as late as possible (see the Equation (4.1)), the model rewards making the two checks far apart from each other, avoiding the second check altogether in certain cases.

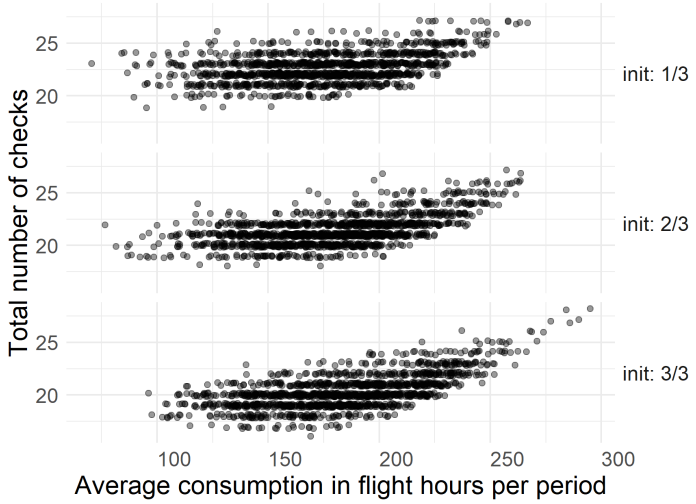

Figure 4.1: Response NM (y axis) vs input features μ_C (x axis) and $Init$ (rows).

In fact, instances where the demand in flight hours is low (e.g., the sum of all flight hours along the horizon is low), typically have only one check for each aircraft. Contrary to this, instances with a high demand for flight hours have more checks and the second check is typically done sooner. A similar relationship can be expected from the initial status of the fleet. If a given fleet is in good status (e.g., aircraft at the beginning of the planning horizon haven't flown that many hours since their last check), one would expect less checks overall and farther apart. Both the total demand of flight hours and the initial status of the fleet are known parameters.

This intuition can be formalized via a supervised ML model where a response n is a function $g_n(y^*)$ on the optimal maintenance cycle distribution (e.g., in Figure 4.1 the response is the total number of checks) and the input features are a function on the mission flight hours demand and the fleet initial status distributions (e.g., in Figure 4.1 the input feature is the average flight hour demand). Note that the instances used in the figure are solved to optimality or close to optimality and that "Init: 1/3" includes instances under the percentile 33th for feature $Init$ (as defined in table 4.2) and group "Init: 2/3" includes instances between percentiles 33th and 66th for that feature.

The method consists in the following. First, we choose a set of candidate responses to predict. Then, we calculate several input features that we suspect can predict those responses. Finally, after validating the ML model on said responses and input features, we obtain, for each response, the subset of input features that best predict the chosen responses and the function that minimizes the loss function: $\hat{g}_n(x)$.

For our problem, we choose the responses in Table 4.1.

With the following equations:

NM	Total number of checks.
$\mu_{T-t'}$	Average distance between the second check and the end of the horizon over fleet.
$\mu_{t'-t}$	Average distance between two checks over the fleet.

Table 4.1: Responses extracted from the solution of any MFMP problem that are predicted using features from input data for the problem.

μ_C	Average consumption per period.
$Init$	Sum of fleet remaining flight hours before first period.
$Spec$	Sum of all special mission flight hours.
μ_{WC}	Period that splits total consumption in two equal parts. Can be fractional.
σ_C^2	Variance of consumption per period.
max_C	Max consumption per period.

Table 4.2: Input features extracted from the input parameters of any MFMP problem that are used to predict responses from the solution for that problem.

$$NM = \sum_{(i,t,t') \in \mathcal{D} | t' < T} m_{itt'} + |\mathcal{I}|$$

$$\mu_{T-t'} = \frac{1}{|\mathcal{I}|} \sum_{(i,t,t') \in \mathcal{D}} m_{itt'} \times (T - t')$$

$$\mu_{t'-t} = \frac{1}{|\mathcal{I}|} \sum_{(i,t,t') \in \mathcal{D}} m_{itt'} \times (t' - t - M)$$

After validating the ML model, we obtain the input features in Table 4.2.

Let the consumption in period t be represent by the following:

$$C_t = \sum_{j \in \mathcal{J}\mathcal{T}_t} H_j R_j \qquad\qquad t \in \mathcal{T}$$

And let $\mathcal{J}\mathcal{Q}$ represent the set of special missions, i.e., that have a capability or where $Q_j \neq \varnothing$.

Then the equations for those input features are:

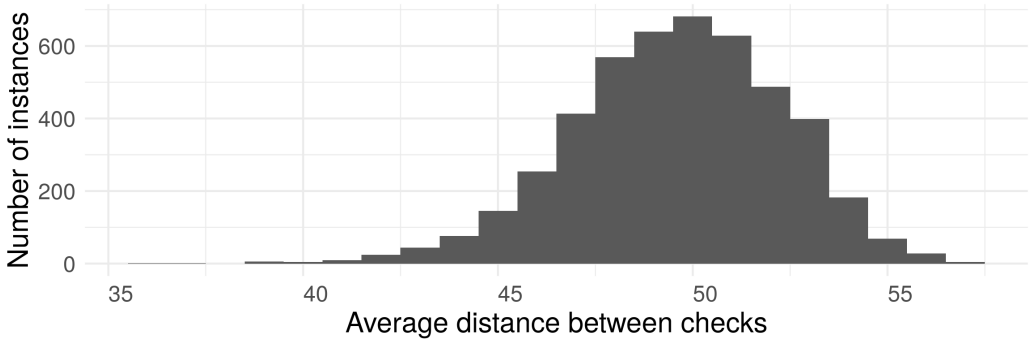

Figure 4.2: Distribution of the average distance between checks in the 4667 successfully solved instances.

$$\mu_C = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} C_t$$
$$Init = \sum_{i \in \mathcal{I}} Rft_i^{Init}$$
$$Spec = \sum_{j \in \mathcal{J}\mathcal{Q}} H_j R_j |\mathcal{T}\mathcal{J}_j|$$
$$\mu_{WC} = \frac{\sum_{t \in \mathcal{T}} C_t \times t}{\sum_{t \in \mathcal{T}} C_t}$$
$$\sigma_C^2 = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} (C_t - \mu_C)^2$$
$$max_C = \max_{i \in \mathcal{T}} \{C_t\}$$

4.4 Experimentation

Five thousand small (15 aircraft) instances are randomly generated following the method found in Section 3.6. The sources of randomness are the missions, i.e. the quantity, hour needs, resource quantities, minimum durations, special requirements; and the initial fleet status, i.e. remaining calendar time, remaining flight time, special capabilities for each aircraft at the start of the planning horizon. These instances are used as an input to obtain learned constraints.

Figure 4.2 shows the distribution on the average distance between checks for all solved instances. The minimum (maximum) distance allowed for each aircraft in each instance is 30 (60) periods.

3 additional sets of 1000 instances each are randomly generated to test the implementation of these learned constraints. Each set corresponds to a particular size of fleet: 30, 45 and 60 aircraft.

In what is left of this section, we first explain the mathematical model implementation and execution in 4.4.1. We then present the statistical model implementation in 4.4.2. Finally, all tested mathematical models are explained in 4.4.3.

4.4.1 Mathematical model implementation

Mathematical models are generated using python 3.7 and the PuLP library.

All instances are solved until optimality with a time limit of 1 hour and a tolerance (absolute gap) of 10. We use CPLEX 12.8 running on single thread Windows 7 with 72 2.3GHz processors and 128 GB RAM workstation. Up to 70 experiments are run in parallel. CPLEX parameters are optimized for the problem using the CPLEX Tuner tool.

4.4.2 Implementation of learned constraints

Of the 5000 instances, around 1000 instances are discarded in order to build the prediction model because of having violated soft constraints or having an absolute gap too large (bigger than 100). The remaining 4084 instances are split into two groups: training (70%) and testing (30%). The training set is used to train a statistical model. The testing set is used for the feature selection process.

For forecasting, we test and compare several methods: Linear Regression (LR), Decision Tree Regression (DTR), Multi-layer Perceptron regression (MLPR), Support Vector Regression (SVR), Quantile Regression (QR) and Gradient Boosted Regression Trees (GBRT).

Robust predictions involve predicting bounds, or quantiles. Only two implementations offered the possibility of predicting quantiles: QR and GBRT. Both techniques are found to have similar effectiveness in predicting the 10% and 90% quantiles. At the end, the former is chosen because it returned scalar coefficients for every regressor and so is more intuitive to validate. To build the QR models, python 3.7 is used together with the statsmodels library.

The learning constraint associated with the number of checks is:

$$\sum_{m \in \mathcal{S}_1} m \leq N\hat{M}^{ub} - |\mathcal{I}| \tag{4.33}$$

$$\sum_{m \in \mathcal{S}_1} m \geq N\hat{M}^{lb} - |\mathcal{I}| \tag{4.34}$$

Where $\mathcal{S}_1 = \{m_{itt'} \in \mathcal{D} | t' < T\}$. The learning constraint associated to the average lateness of the second checks would be then:

$$\sum_{m_{itt'} \in \mathcal{D}} (T - t') \times m_{itt'} \leq \hat{\mu}^{ub}_{T-t'} \times |I| \tag{4.35}$$

$$\sum_{m_{itt'} \in \mathcal{D}} (T - t') \times m_{itt'} \geq \hat{\mu}^{lb}_{T-t'} \times |I| \tag{4.36}$$

Finally, concerning the mean distance between checks, two types of constraints are devised. The first type is equivalent to the previous two, limiting the average distance between checks.

$$\sum_{m_{itt'} \in \mathcal{D}} (t' - t - M) \times m_{itt'} \leq \hat{\mu}^{ub}_{t'-t} \times |I| \tag{4.37}$$

$$\sum_{m_{itt'} \in \mathcal{D}} (t' - t - M) \times m_{itt'} \geq \hat{\mu}^{lb}_{t'-t} \times |I| \tag{4.38}$$

The second type, assumes each aircraft has a maximum deviation with respect to the mean distance between checks in order to limit the combinations of possible checks:

$$m_{itt'} = 0 \qquad m_{itt'} \in \mathcal{D} | t' - t - M < \hat{\mu}^{lb}_{t'-t} - tol \tag{4.39}$$

$$m_{itt'} = 0 \qquad m_{itt'} \in \mathcal{D} | t' - t - M > \hat{\mu}^{ub}_{t'-t} + tol \tag{4.40}$$

Figure 4.3 shows the prediction of the upper bound for the mean distance between checks ($\hat{\mu}^{ub}_{t'-t}$) plotted against three input features from Table 4.2. Each instance has one black (blue) point with the actual value (predicted upper bound). For the vertical and horizontal facets, "1/3" corresponds to instances under the 33th percentile for that feature and "2/3" to instances between the 33th and the 66th percentile for that feature.

4.4.3 Model experimentation

We call "base" the model described in Section 4.1, "old" the one formulated in Section 3.4 and "base_*" ("old_*") the various derivatives from each model. The model "base_determ" refers to the "base" model with deterministic cuts added as described in Section 4.2.

Each learned cuts model involves the combination of two configuration parameters, corresponding to two steps during the pattern production. In the first step, we control the maximum deviation (*tol*) we allow each individual maintenance pattern to be from the mean distance between maintenances prediction bounds (see equations (4.39) and (4.40)). In the second step, we control how many of the previously rejected maintenance patterns should we incorporate nonetheless to the model, randomly, as a percentage (*recyc*) of the already

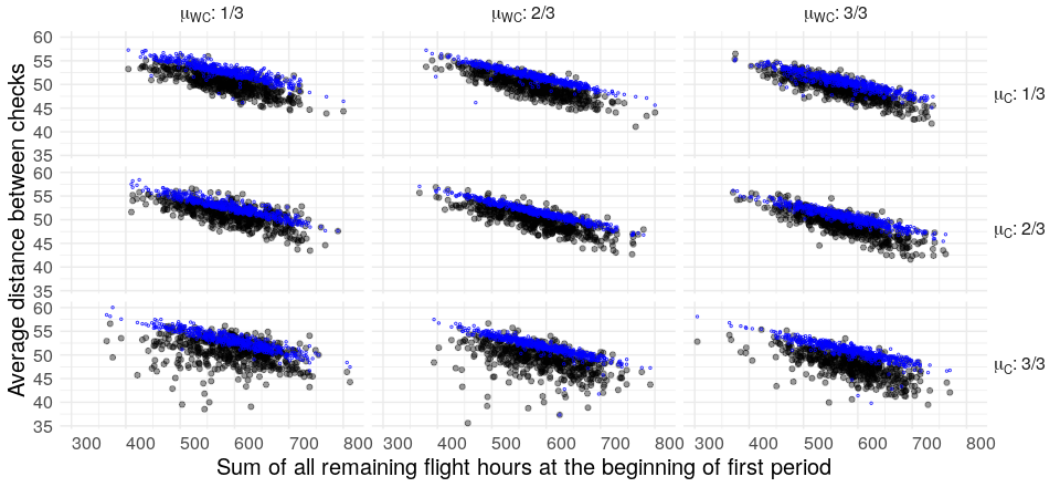

Figure 4.3: Response $\mu_{t'-t}$ (y axis) vs input features $Init$ (x axis), μ_{WC} (columns), μ_C (rows) for the over 4000 instances solved.

Scenario	*tol*	*recyc*
base	∞	0
base_a1	2	0
base_a2	0	0
base_a3	$-\infty$	0
base_a2r	0	0.2
base_a3r	$-\infty$	0.2

Table 4.3: All learned cuts models that are based on the "base" model.

reduced number of patterns. The "base" model ($tol = \infty$) has no added cuts. The most aggressive model ($tol = -\infty$), assumes all aircraft should have the same distance between checks, equal to the predicted average.

The notation for the learned cuts models is shown in Table 4.3 and they are consistent between the "base" model and the "old" model. Each consists of a particular combination of the tolerance for creating patterns (tol) and the percentage of random extra patterns ($recyc$). Reducing tol reduces the number of patterns created and increasing $recyc$ increases the number of patterns recycled (included).

Three additional matheuristics are tested to compare the performance gains offered by the previously presented learned cuts. The matheuristics are described below:

base_flp The linear relaxation of the "base" model is solved. Only maintenance patterns with a non-zero value in the relaxed optimal solution are kept for a second run of the "base" model.

base_flp2 The linear relaxation of the "base" model is solved. Only maintenance patterns that are similar to a pattern with a non-zero assignment (i.e., both patterns share the same aircraft and at least one date of the two checks) are kept for a second run of the "base" model.

base_flp3 The linear relaxation of the "base" model is solved. Let t_i^f (t_i^l) be the soonest (latest) check for aircraft i with a non-zero value in the optimal relaxed solution. Only maintenance patterns that have the first maintenance after a t_i^f and the second maintenance before t_i^l are used in the second run.

4.5 Results

All comparisons presented in this section, with the exception of Tables 4.9 and 4.10, are done using the medium size dataset ($|\mathcal{I}| = 30$). We define the following possibilities for the status of the solution for any problem. Infeasible: problem proven infeasible. IntegerFeasible: an integer solution is found but not proven optimal before time limit. IntegerInfeasible: no integer solution is found before time limit. Optimal: difference between relaxation and best integer solution is less than the absolute gap. Each status is exclusive one from the other (i.e. they sum the totality of correctly generated instances).

This section is structured as follows. First, Section 4.5.1 briefly analyses the "base" and "old" models in terms of their performance; Section 4.5.2 presents the results of learned cuts applied on both models; Section 4.5.3 compares the learned cuts with other more traditional matheuristic techniques; finally, Section 4.5.4 shows a complete comparison including larger dataset sizes and alternative variants on the learned cuts.

4.5.1 Comparison between models and deterministic cuts

We compare the "base", "old" and "base_determ" models with respect to solution time. This performance is expressed as: (1) the number of nodes that are visited in the branch and bound phase before proving optimality, (2) the quality of the LP relaxation (before and after the cuts phase), (3) the capacity to obtain feasible solutions and (4) the time it takes to prove an optimal solution.

Table 4.4 shows statistics on the status of the solutions returned by each model. The "old" model is considerably better at obtaining feasible solutions in less than one hour. Table 4.5 shows the quality of the relaxation and the number of nodes needed to obtain an optimal solution. "nodes" ("time") is the average number of nodes (solution time) to prove optimality, "LP_first" ("LP_cuts") is the average relative distance between the initial LP (LP after root node cuts) and the optimal. Only optimal instances are counted. The "base" model is

Indicator	base	base_determ	old
Infeasible	41	44	40
IntegerFeasible	277	289	640
IntegerInfeasible	384	368	29
Optimal	279	280	272
Total	981	981	981

Table 4.4: Comparison of the number of instances per status returned in each model: "base", "old" and "base_determ".

Indicator	base	base_determ	old
LP_cuts	0.39	0.54	0.48
LP_first	4.93	4.91	22.04
nodes	705.32	692.60	3234.61
time	1031.48	1054.36	996.36

Table 4.5: Quality of the relaxation and number of nodes needed to obtain an optimal solution in each model: "base", "old" and "base_determ".

considerably better at obtaining a good initial LP relaxation while also needing considerably less nodes to prove optimality. With respect to solution times to obtain an optimal solution, they present a similar performance. The deterministic "base_determ" model offers slight improvements on the "base" model.

4.5.2 Comparison of learned cuts

In order to assess the merits of the learned constraints, we use several indicators that measure three main concepts: quality degradation, performance gains and feasibility sensibility.

Figures 4.4 and 4.5 show a summary of the proportion and changes of the status of the solution for both models, "base" and "old", respectively, when applied learned cuts. The number of solutions with an "Optimal" status increases in both cases. With respect to finding feasible solutions, the "base" model sees a decrease on the number of instances without a solution ("IntegerInfeasible" status) when adding learned cuts while the "old" model sees a regression in this respect. Note that the number of infeasible solutions remains almost the same with the given configuration regardless of the model used.

When adding learned constraints, we eliminate valid solutions from the pool (i.e. these are invalid cuts or pseudo cuts). This implies there is a risk of taking out the actual optimal solution. In order to measure the effect of these cuts on the value of the optimal solution, we measure quality degradation as the distance (in % with respect to the "base") between the objective functions in the cases when all models return an optimal status. Figure 4.6 compares the distribution of such degradations for each case. Only optimal solutions are counted and

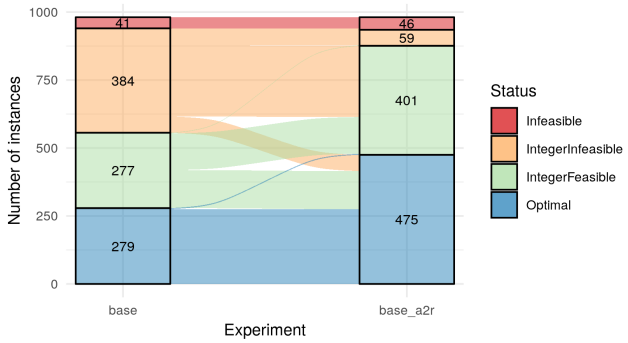

Figure 4.4: Number of instances per status returned in "base" model and the changes of status when applying learned cuts.

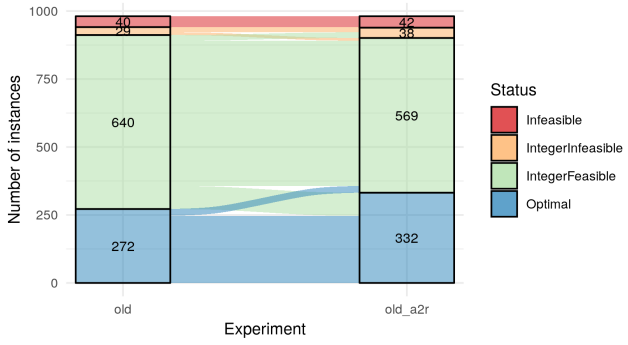

Figure 4.5: Number of instances per status returned in "old" model and the changes of status when applying learned cuts.

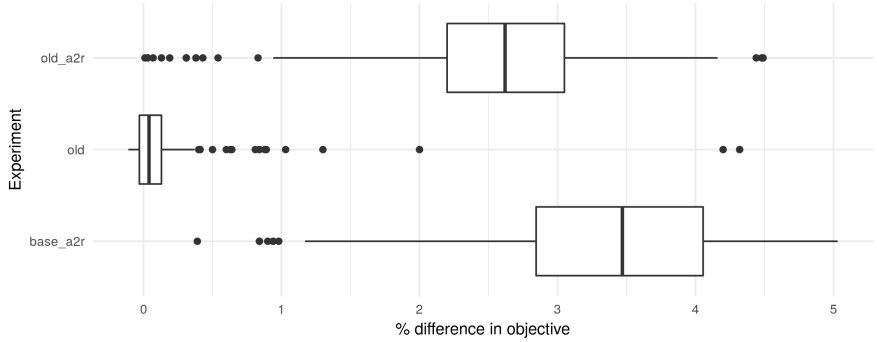

Figure 4.6: Relative difference between the objective function value for each model, compared with the "base".

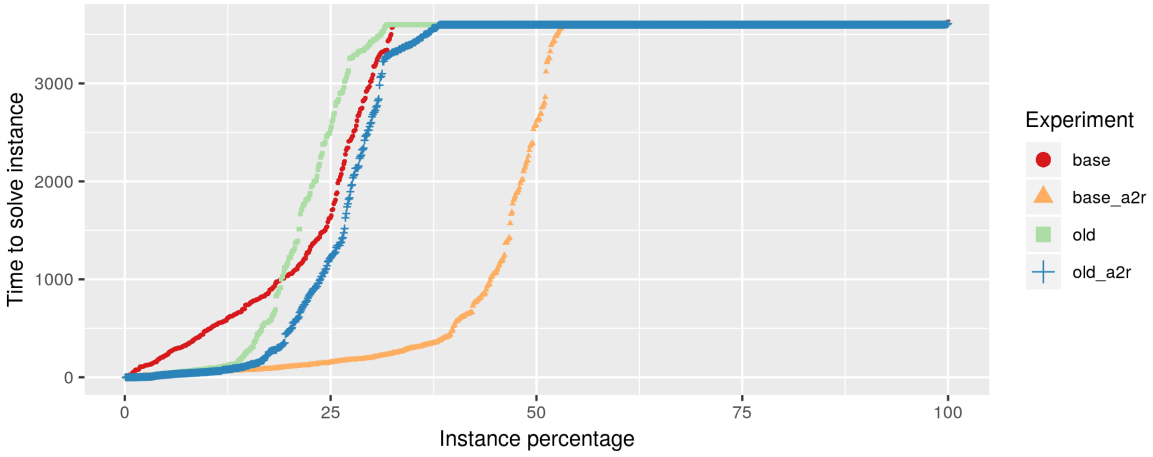

Figure 4.7: The distribution of solution times of each method for all instances.

the right and left side tails representing 10% of the sample are removed for better display. The degradations of the learned cuts are almost entirely lower than 5% from the "base" optimal. The "old" model is expected to have a degradation of 0% or close to 0%, which it does.

We measure the performance of each method by comparing the solution time. Figure 4.7 shows the distribution of solution times for each model for all instances. The x-axis represents the percentile of instances from 0 to 100 and the y-axis the time it takes to solve the slowest instance in that percentile. It is possible to see how adding the cuts increases performance and the greatest performance gains are obtained in the "base" model.

Another consequence of these pseudo-cuts is eliminating the complete solution space for an instance. We quantify this possibility in the following way: the number of new infeasible instances obtained and the increase on the number of soft constrains violations. Table 4.6 presents the following statistics on feasibility: "IntegerInfeasible $\rightarrow$ Infeasible" is the number of additional new infeasible instances which had this status in the "base" model; "errors_mean" is the average of new soft constraints violations and "errors_new" is the percentage of new instances with at least one soft constraint violation, among optimal solutions.

Indicator	base_a2r	old	old_a2r
IntegerInfeasible → Infeasible	5.00	0	2.00
errors_mean	0.06	0	0.06
errors_new	2.42	0	4.35

Table 4.6: The impact on feasibility for each method on all instances.

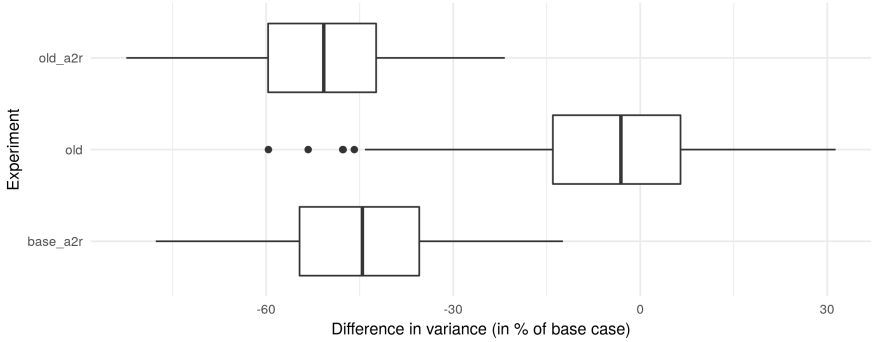

Figure 4.8: The percentage difference in variance for the distance between checks in alternative models with respect to the base model.

This shows the number of additional infeasible instances is almost non existent and there is very little additional instances with soft constraints violations in both models. All new infeasible instances are instances that are not proven feasible ("IntegerInfeasible → Infeasible") and could potentially be indeed infeasible.

As stated previously, the uniform usage of the fleet is also a factor to take into account when choosing a correct maintenance planning. Given that the pattern selection is oriented towards constraining the check patterns that are too far from the predicted mean distance between checks, it is expected for the variance of this measure to decline. For the cases when there is more than one fleet type (i.e., for larger instances), the variances of each fleet type are calculated individually and summed into one single indicator. Figure 4.8 shows how the variance is greatly reduced in most of the cases, to around half, which results in a more stable and balanced planning.

4.5.3 Comparison with other matheuristics

The three alternative matheuristics were compared with the learned cuts model in terms of performance, loss of optimality and loss of feasibility.

Table 4.7 shows the following indicators for optimal solutions: new infeasible instances per method and per status in the "base" model; the average of new soft constraints violations ("errors_mean") and the percentage of new instances with at least one soft constraint violation ("errors_new").

Indicator	base_a2r	base_flp	base_flp2	base_flp3
IntegerFeasible → Infeasible	0.00	10.00	8.00	8.00
IntegerInfeasible → Infeasible	5.00	76.00	12.00	10.00
Optimal → Infeasible	0.00	3.00	3.00	3.00
errors_mean	0.05	1.97	1.00	1.05
errors_new	2.31	31.54	23.85	24.23

Table 4.7: Comparison of the impact on feasibility with matheuristic methods.

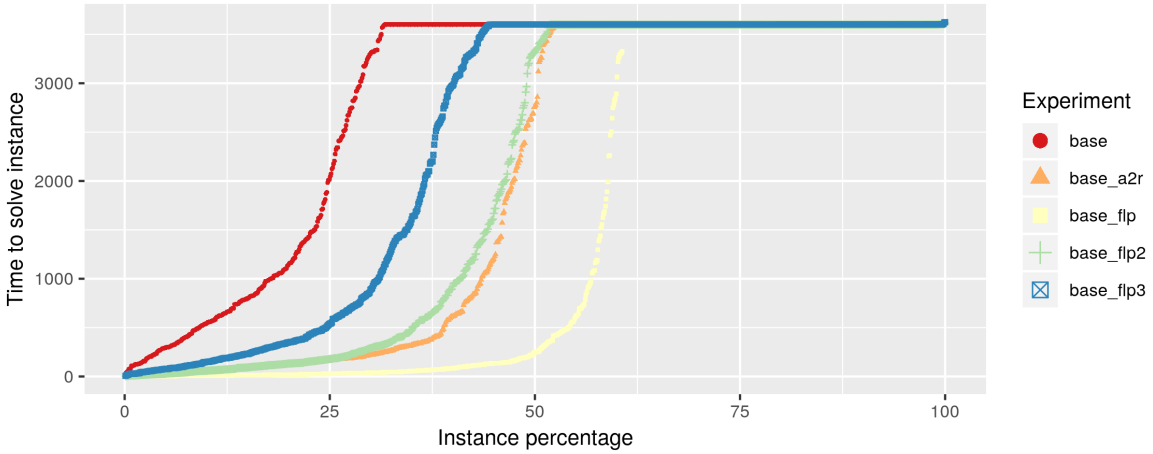

Figure 4.9: The distribution of solution times for each method for all 994 instances solved.

One can see that some previously "Optimal" and "IntegerFeasible" solutions in the "base" model become now "Infeasible", in contrast with the "base_a2r" model. Also, these matheuristics introduce many more new soft constraints violations. Figure 4.9 shows the solution time for each method compared to the base model and to the learned cuts model. In the x-axis is the percentile of instances from 0 to 100. In the y-axis is the time it took to solve the slowest instance in that percentile: only the most aggressive of the matheuristics beats the previously shown learned cuts.

These two results highlight the advantage of using learned cuts over other more standard matheuristics: they offer a good combination of less degradation in new infeasible solutions and better performance in solution times.

4.5.4 Summary

Tables 4.8, 4.9, and 4.10 show a summary for each dataset of the gains and costs of applying different degrees of cuts in models "base" and "old". Each statistic ("Stat") is a comparison between the named model and the "base" model. As a reference, the "old" model results are also shown and compared accordingly. It is important to note that the "old" model is similar in feasibility (E_μ, $E_\%$ and $Infeas$), quality degradation (Q_μ, Q_m and Q_{95}) and variance (V_μ)

Stat	base_a1	base_a2r	base_a3r	old	old_a1	old_a2r	old_a3r
E_μ	0.16	0.07	2.28	**-0.01**	2.27	0.07	2.65
$E_\%$	1.02	2.54	10.66	**0**	10.66	4.57	13.2
$Feas$	25.96	32.5	33.1	**36.22**	32.5	35.01	35.11
$Infeas$	1.21	0.61	1.51	**-0.1**	1.41	0.21	1.61
Q_μ	12.97	9.86	27.65	**-0.18**	56.58	6.3	14.4
Q_m	2.8	3.65	8.08	**0.02**	6.36	2.67	5.48
Q_{95}	4.82	7.83	11.17	**0.92**	9.93	4.14	7.81
T_μ	-16.16	-29.5	**-49.53**	0.54	-14.17	-5.2	-30.39
V_μ	-41.31	-42.56	-60.19	-1.83	**-93.2**	-48.77	-68.45

Table 4.8: Summary table comparing the performance of several options of cuts in scenario of size $|\mathcal{I}| = 30$.

Stat	base_a1	base_a2r	base_a3r	old	old_a1	old_a2r	old_a3r
E_μ	**0**	0.05	1.49	**0**	0.68	0.14	2.11
$E_\%$	**0**	2.7	10.81	**0**	2.7	8.11	10.81
$Feas$	37.01	48.55	54.56	**60.58**	49.55	55.07	58.48
$Infeas$	2	0.8	2.3	**0**	2.4	0.1	1.6
Q_μ	3.14	4.54	15.68	**0.37**	7.39	3.73	6.48
Q_m	2.8	3.77	9.2	**0**	7.42	2.69	5.69
Q_{95}	6.69	10.21	24.39	**1.79**	11.49	9.74	15.31
T_μ	-8.9	-15.48	**-31.64**	0.64	-6.62	-1.6	-10.65
V_μ	-41.62	-45.04	-62.71	-2.48	**-95.29**	-48.84	-68.16

Table 4.9: Summary table comparing the performance of several options of cuts in scenario of size $|\mathcal{I}| = 45$

as the "base" model, since both share the same solution space. Thus, the gains in performance (T_μ and $Feas$), and variance reduction (V_μ) offered by the learned cuts models need to be weighted against trade-offs on the former indicators.

All comparisons are done against the "base" model for each option dataset size. E_μ and $E_\%$ refer to the percentage difference in average number of soft constraint violations per instance and the proportion of new instances with at least one violation, among optimal solutions. $Feas$ ($Infeas$) refers to the average additional number of feasible (infeasible) instances obtained as a percentage of total instances (1000). Q_μ, Q_m and Q_{95} are the average, median and 95-percentile difference in the objective function when comparing optimal solutions (as a percentage of the "base"). T_μ is the difference in average solution times (as a percentage of the "base") for all instances. V_μ is the difference (as a percentage of the "base") in the variance of the distance between checks along the fleet for all instances.

Regarding optimality degradation (Q), solutions with learned cuts tend to be 5-6% away from the real optimal (or the best known solution). By recycling some excluded patterns the gap can be reduced to less than 4%.

Stat	base_a2	base_a2r	base_a3r	old	old_a1	old_a2r	old_a3r
E_μ	**0**	**0**	**0**	**0**	1.92	**0**	3.92
$E_\%$	**0**	**0**	**0**	**0**	7.69	**0**	15.38
$Feas$	32.59	36.02	43.18	**53.88**	37.23	44.7	49.44
$Infeas$	5.76	1.22	3.44	**0.11**	3.54	0.61	2.33
Q_μ	4.57	3.97	7.74	**0.63**	7.13	2.54	5.28
Q_m	3.97	3.15	7.47	**0**	6.29	2.58	5.3
Q_{95}	8.38	7.81	13.66	**3.42**	14.05	4.33	9.5
T_μ	-10.19	-8.26	**-18.54**	0.74	-4.48	-0.44	-5.67
V_μ	-50.9	-43.28	-61.98	1.13	**-94.89**	-47.54	-66.8

Table 4.10: Summary table comparing the performance of several options of cuts in scenario of size $|\mathcal{I}| = 60$

Regarding performance, all learned cuts increase the number of instances where a feasible solution is found in the "base" model: $Feas$ improves by an average difference of 20% to 50% (measured in % of total instances). This is not so in the case of the "old" model, where performance is lost in this sense. Gains in solution times are also substantial. The average time (T_μ) usually improves between 10% to 30%.

Additional infeasible solutions and soft constraints violations ($Infeas$ and E) increase with the aggressiveness of the cuts and depending on whether we allow the possibility of recycling or not. In the cases of less aggressive cuts, most new infeasible instances are not proven feasible by the "base" model. The impact of recycling in reducing the number of infeasible instances while keeping most of the performance is to be noted.

Finally, variance reduction (V) between 40% and 60% is usually obtained with most cut strategies, although recycling reduces slightly the strength of the reduction. The more aggressive a cut, the most the gain in variance reduction.

Larger instances allow for more aggressive cuts without losing too much feasibility or quality, as can be seen by comparing the impact of "base_a3r" across different sizes of instance. The cuts in the "base" model have greater reductions in solution time than those in the "old" model, while the cuts in the latter have slightly lower feasibility and quality degradation and slightly greater variance reductions. This can be explained as the nature of the cuts differs in each formulation: in the "base" model it consists in reducing the number of variables while in the "old" model it consists in adding constraints.

The best compromise seems to be reached when adding recycling to aggressive cutting ("base_a2r" or "base_a3r") depending on the size of the instance. In most of the cases, a low or very low optimality degradation (Q) can be seen together with a low feasibility change ($Infeas$, E), compared to both the "base" and "old" models. The performance gains can be seen in both time to reach an optimal solution (T) and the reduction of the variance (V) of the usage of the fleet. Finally, compared to the "base" model, the amount of feasible solutions ($Feas$) is increased.

These results encourage the design of more sophisticated ways of predicting patterns in solutions. Ideally, a function that returns a probability distribution of patterns for each instance could be trained and then used to sample promising patterns. Our learned cuts model is a special case where we give a very high priority (a probability of 1) to patterns in the range of tolerance for the distance between checks and a very low probability (dependent on the recycling parameter and the total amount of available patterns) to the rest of the patterns.

4.6 Conclusions

This chapter presents an alternative MIP formulation for the long-term Military Flight and Maintenance Planning problem. The performance is compared with previous formulations using cases inspired by the French Air Force. Valid bounds are formulated and tested. A forecasting model is designed to predict characteristics of optimal solutions based on the input data and used to create pseudo-cuts. For comparison, several matheuristic that use the LP relaxation are also applied to reduce the solution space of the problem.

The study shows that predicting characteristics of the optimal solution is a powerful method to obtain very good solutions that are close to the optimal, in less time and with very little loss of feasibility. In addition, the prediction also allows consideration of a second objective without hindering performance. The performance gains of these pseudo-cuts will depend heavily on the implementation, i.e, on the mathematical model employed and the way the pseudo-cuts are added to the model.

A comparison with more classical matheuristic techniques highlights the potential benefits of doing a good trade-off between optimality and infeasibility degradation in the search for performance.

Further work includes, first of all, researching better ways to predict the optimal patterns in a solution. Secondly, the application of this technique into problems where pattern can potentially be used, such as workforce scheduling and cutting stock problems. Furthermore, this technique can be generalized into a random sampling of patterns where each pattern is picked with a probability equal to the potential it has to appear in the optimal solution.

In this chapter, patterns were created exclusively from check decisions. Also, the gains in performance decrease as the size of instances to apply the cuts becomes too large with respect to the size of the instances used to train the predictions. As a consequence very large scale instances could not be solved efficiently. Chapter 5 presents an extension of this pattern concept that includes mission assignment decisions. The result is an explicit graph-based pattern generation technique that is explored with Dynamic Programming and implemented into a Variable Neighborhood Descent matheuristic. The goal is to obtain very good performance when solving very large scale instances.

Chapter 5

Graph-based Variable Neighborhood Descent matheuristic

Contents

This chapter presents an alternative VND matheuristic for the MFMP problem that combines Rolling Horizon (RH) heuristics and graph-based Dynamic Programming (DP) algorithms for local optimizations. This method produces near-optimal solutions faster than previous exact methods from Chapters 3 and 4.

The chapter is structured as follows. Section 5.1 outlines the solution method, including the implemented neighborhoods. Section 5.2 formulates a graph representation of the solution

space. Section 5.3 describes the scenario generation and the implementation details and Section 5.4 provides the results. Finally, Section 5.5 shows the conclusions.

The contributions of this chapter are to be submitted for publication as: F. Peschiera, N. Dupin, A. Haït, and O. Battaïa. Novel graph-based matheuristic to solve the flight and maintenance planning problem. Forthcoming.

5.1 Solution approach

Variable Neighborhood Descent (VND) is a variant of the more common metaheuristic Variable Neighborhood Search (VNS) [123] presented Section 2.3.3. The main difference is VND omits the random perturbation phase present in VNS and thus applies the local search phase to the unmodified current solution.

In our case, the resulting VND approach consists of a series of large local searches, randomly selected, for two neighborhood types. We call these two neighborhood types "SPA" and "RH". Let a candidate move be a relatively small part of the solution space that is chosen to be explored and optimized. In our problem we define a candidate move as a combination of: (1) a set of consecutive periods $\mathcal{T}_c \subset \mathcal{T}$ and (2) a subset of aircraft $\mathcal{I}_c \subset \mathcal{I}$.

All neighborhoods have in common the use of a slice and repair approach where (1) a valid solution is taken as input, (2) a candidate move is selected, (3) an exact method is run on the candidate to repair it and, finally, (4) the solution to the candidate is incorporated to the solution. The choice of the candidate move in (2) is determined partly randomly and partly based on the number of soft constraints violations on the current solution. Figure 5.1 summarizes the general solution approach.

The section is structured as follows. Sections 5.1.1 and 5.1.2 describe the solution encoding and a pattern representation of the solution, respectively. Section 5.1.3 presents the two neighborhoods: a RH heuristic that solves a MIP model and that we call "RH"; and a DP algorithm that solves a Shortest Path Problem (SPP) and that we name "SPA". Finally, Section 5.1.4 presents several methods, some based on said neighborhoods, to create an initial solution.

5.1.1 Solution encoding

Let I (T) be $|\mathcal{I}|$ ($|\mathcal{T}|$). Any solution x for the FMP is represented by a matrix $A = \mathbb{Z}^{I \times T}$.

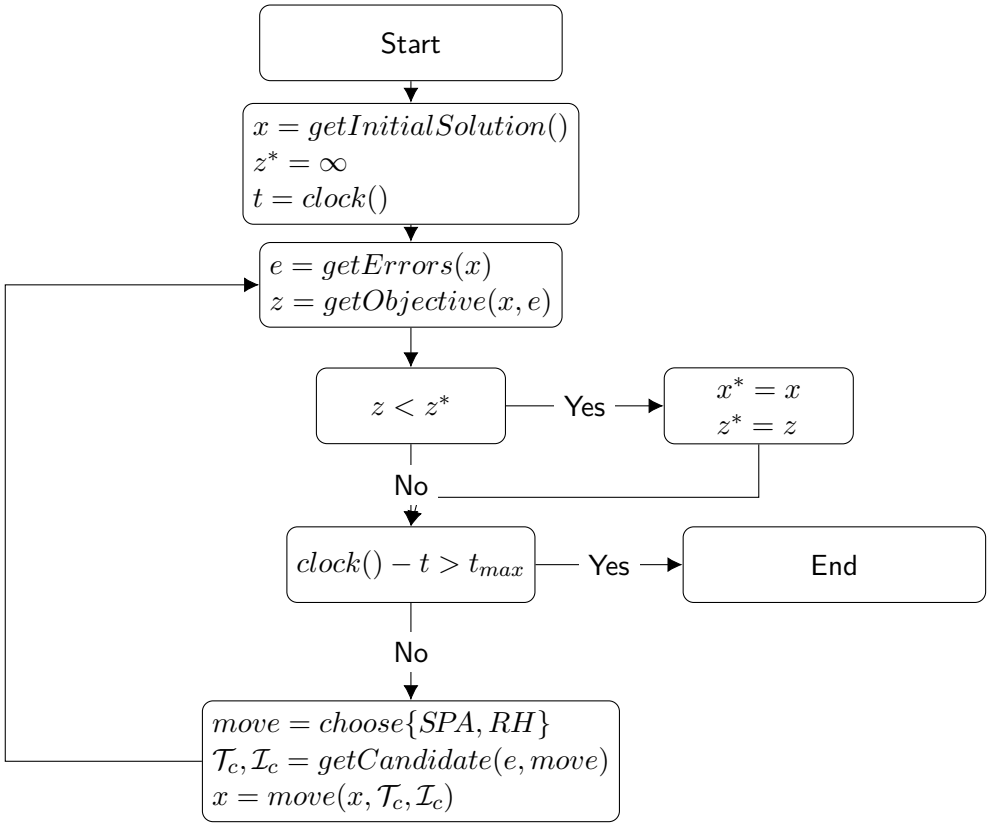

Figure 5.1: VND matheuristic with two neighborhoods "SPA" and "RH" that returns the best solution found x^* with cost z^*after t_{max} seconds.

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,t} & \cdots & a_{1,T} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,t} & \cdots & a_{2,T} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i,1} & a_{i,2} & \cdots & a_{i,t} & \cdots & a_{i,T} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{I,1} & a_{I,2} & \cdots & a_{I,t} & \cdots & a_{I,T} \end{pmatrix}$$

Each cell $a_{it} \in A$ in the matrix contains one integer value representing the assignment value of aircraft i at period t.

$$a_{it} = \begin{cases} -1 & check \\ 0 & no\ assignment \\ j & mission\ j \end{cases}$$

In order to know the complete state of a particular aircraft i in period t (i.e., calculate the rft_{it} and rct_{it}), one needs to explore each cell $a_{it'}\ \forall t' < t$.

5.1.2 Pattern representation

We define a pattern as a sequence of missions assignments and checks for aircraft i between periods t and t'. For a pattern to be feasible, it should comply with maintenance rules regarding flight hours and calendar periods between checks, the initial status of the aircraft, the periods in which each mission is available, the minimum and maximum durations of assignments to missions and the capacity of the aircraft to do a particular mission.

Let $\mathcal{P}_{itt'}$ represent all feasible patterns for aircraft i between the start of period t and the end of period t'. If pattern $p \in \mathcal{P}_{itt'}$ belongs to solution x, then p and a_{it} are related as follows:

$$p = \{a_{it}, a_{it+1}, ..., a_{it'}\} \Leftrightarrow p \in \mathcal{P}_{itt'} \wedge p \in x$$

In particular, any solution x can be completely represented by a set of size $|\mathcal{I}|$ patterns $p_i \in \mathcal{P}_{i0T}$, one for each aircraft $i \in \mathcal{I}$. In this case, each pattern p_i corresponds to a row in matrix A. The generation of patterns is explained in detail in Section 5.2, where a pattern is modeled as a path in a graph representation of the solution.

5.1.3 Neighborhoods

Given a valid initial solution x and a candidate move c, two neighborhoods are available to produce a new solution. Section 5.1.3.1 defines an exact algorithm that solves a SPP, Section 5.1.3.2 describes a generalized RH MIP model.

5.1.3.1 Shortest Path Algorithm

We name this neighborhood "SPA" and consists of a HC heuristic that changes the assignments for a set of aircraft, one aircraft at a time, in order to improve the objective function. This neighborhood takes advantage of a graph representation of an aircraft's solution, detailed in Section 5.2.1.

It does so by finding the shortest path between the extreme nodes of the candidate move using the existing graph representation for the aircraft. The weights for the edges in the graph are modified so that the solution to the SPP is also the optimal pattern swap for that aircraft.

Algorithm 6 presents the logic to find a solution for candidate move c by modifying one aircraft at a time. First, in Line 3 we calculate the violations of soft constraints without counting the present aircraft and only for the relevant periods (i.e., $t \in \mathcal{T}_c$). Then, the state of the aircraft at periods t^S and t^E is mapped with the equivalent nodes in the aircraft's graph in Lines 4 and 5. Function *get_weight* in Line 7 assigns a weight to each arc a according to the marginal impact on the objective function from including a in the current solution for aircraft i. The state of aircraft i in period t^E should be compatible with the periods that follow t^E and this implies filtering out some nodes from graph G_i. This is done in Line 8, where a subgraph G' is created. A new pattern is obtained by solving the SPP on graph G' in Line 9. Finally, the pattern is applied to the solution in Line 10, replacing the previous solution for aircraft i.

More detail on the implementation for the *get_weight* function can be found in Algorithm 7. Four sources of weights are identified for a given arc a, one for each component of the objective function. The weights depend on the destination node $n = a.Head$. Lines (6-8) increase the weight by PR if aircraft i is not assigned to an unsatisfied mission j at period t. Lines (9-11) increase the weight by PC if the aircraft will be in maintenance during a period when there is no more free maintenance capacity left. Lines (12-16) increase the weight by PH for each extra flight hour that the aircraft's clusters $k \in \mathcal{K}_i$ need to reach their minimum sustainability constraint, at each period of node n. Finally, Lines (17-18) increase the weight by $T - n.t^s$ in case node n corresponds to a check.

Figure 5.2 shows an example of a candidate move c of one aircraft being modified by "SPA". The candidate move is $\mathcal{T}_c = \{1, 2, 3, 4, 5\}$, $\mathcal{I}_c = \{2\}$.

Algorithm 6: $SPA(x, \mathcal{I}_c, \mathcal{T}_c)$

Data:
x: current solution.
G_i: graph for aircraft i.
G'_i: filtered graph for aircraft i.
a: arc of graph.
t^S: first period in $\mathcal{T}_c$.
t^E: last period in $\mathcal{T}_c$.

begin
  for $i \in shuffle(\mathcal{I}_c)$ **do**
    $err \leftarrow get\_errors\_subset(x, \mathcal{I} \setminus \{i\}, \mathcal{T}_c)$
    $source \leftarrow get\_node(G_i, i, t^S, x)$
    $sink \leftarrow get\_node(G_i, i, t^E, x)$
    for $a \in G_i$ **do**
      $weight[a] \leftarrow get\_weight(i, a, err)$
    $G'_i \leftarrow filter\_graph(G_i, x)$
    $pattern \leftarrow shortest\_path(G'_i, source, sink, weight)$
    $x.Apply(pattern)$

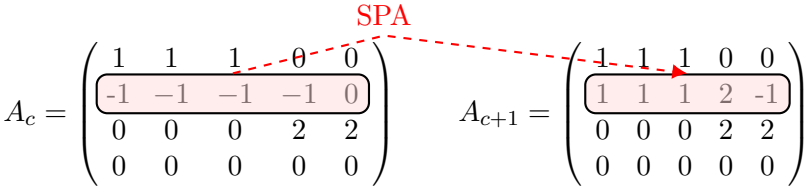

Figure 5.2: Example of a "SPA" neighborhood applied to solution A_c in order to obtain solution A_{c+1}.

Algorithm 7: *get_weight*(i, a, err)

Data:
i: aircraft.
a: arc of graph.
err: errors per category.
w: total weight of arc.
n: head node of arc a.
T: number of periods in planning horizon.

begin
  $n \leftarrow a.Head$
  $periods \leftarrow \{n.t^s, ..., n.t^f\}$
  $n.rfts \leftarrow map(t \rightarrow rft_{it}) \forall t \in periods$
  $w \leftarrow 0$
  for $(j, t) \in err[R_j]$ **do**
    if $i \in \mathcal{I}_j \wedge t \in periods \wedge n.a <> j$ **then**
      $w \leftarrow w + PR$
  for $t \in err[C^{max}]$ **do**
    if $t \in periods \wedge n.a == -1$ **then**
      $w \leftarrow w + PC$
  for $(k, t, rft_{rem}) \in err[H_{kt}^{Clust}]$ **do**
    if $k \in \mathcal{K}_i \wedge t \in n.rfts$ **then**
      $rft \leftarrow n.rfts[t] + rft_{rem}$
      if $rft < 0$ **then**
        $w \leftarrow w + PH \times -rft$
  if $n.a = -1$ **then**
    $w \leftarrow w + T - n.t^s$
  return w

5.1.3.2 Rolling Horizon

We name this neighborhood "RH" and consists of a generalization of the model presented in Section 4.1 to handle rolling horizons of any size.

In order to adapt it to solve candidate moves, we first build the complete MIP model with the whole set of variables and constraints for the whole planning horizon and the whole fleet of aircraft, as we would normally do to solve the entire problem. Then, at each iteration we (1) initialize the set of variables with the current solution, (2) fix all non-empty assignments outside of the candidate move, and (3) add additional temporary cuts for the assignments that cross the boundaries of the candidate move so that the part that falls outside is fixed while the part inside is set free. Note that this reduced model can schedule checks and assign missions outside of the boundaries of the candidate move as long as they comply with all the additional cuts and variable fixes. In practice, the solver pre-solve step eliminates a large part of the problem, reducing it to an equivalent instance of small or medium size with a potentially good initial solution.

Given that we use the same objective function as the original problem, an optimal solution to this neighborhood provides an optimal change to the solution for a given candidate move.

Let t^S (t^E) be the first (last) period of $\mathcal{T}_c$. Let $\mathcal{D}(r)_i$ be the set of variables of type $r \in \{m, a\}$ that refer to aircraft i. Let $\hat{D}(r)_i \subset \mathcal{D}(r)_i$ be the subset of variables that are part of the current solution, i.e., where $m_{itt'} = 1$ or $a_{ijtt'} = 1$, indexed by aircraft i. Finally, let $t^s(d)$ denote the first period index t and $t^e(d)$ denote the last period index t' for some variable $d \in \mathcal{D}(r)_i$.

For each $d \in \hat{D}(r)_i$, if $t^s(d)$ and $t^e(d)$ fall outside the candidate move, i.e., $t^s(d) \notin \mathcal{T}_c$ and $t^e(d) \notin \mathcal{T}_c$, then the variable is fixed. If both fall inside the candidate move, then the variable is initialized but is not fixed. If only one of the two falls inside the candidate move, two cases arise: one at each side of the boundary.

If $t^s(d)$ falls before the candidate move (i.e, $t^s(d) < t^S$) and $t^e(d)$ falls somewhere inside the candidate move (i.e., $t^e(d) \in \mathcal{T}_c$), Constraints (5.1) guarantee that an assignment r (check or mission) crossing the beginning of the boundary will only change its end period $t^e(d)$ to somewhere inside $\mathcal{T}_c$ while fixing its start period $t^s(d)$.

$$
\begin{aligned}
&B(r)_i^s = \{t^s(d) | t^s(d) < t^S \wedge t^e(d) \in \mathcal{T}_c \ \ \forall d \in \hat{D}(r)_i\} && i \in \mathcal{I}, r \in \{m, a\} \\
&DB(r)_i^s = \{d \in \mathcal{D}(r)_i | t^s(d) \in B(r)_i^s \wedge (t^S - 1 \leq t^e(d) \leq t^E)\} && i \in \mathcal{I}, r \in \{m, a\} \\
&\sum_{DB(r)_i^s} d \geq 1 && i \in \mathcal{I}, r \in \{m, a\} \qquad (5.1)
\end{aligned}
$$

Similarly, the second case where $t^s(d) \in \mathcal{T}_c$ and $t^e(d) > t^E$ is formulated in Constraints (5.2).

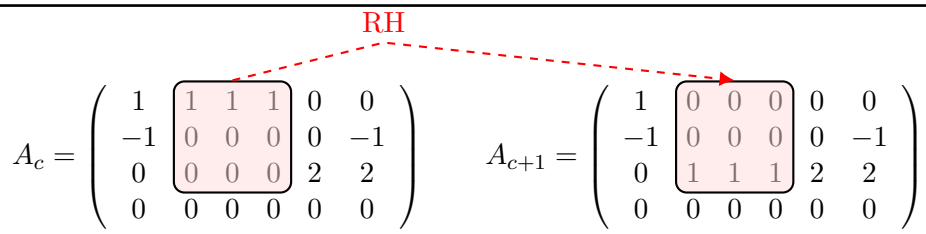

Figure 5.3: Example of a "RH" neighborhood applied to solution A_c in order to obtain solution A_{c+1}.

$$
\begin{aligned}
&B(r)_i^e = \{t^e(d) | t^s(d) \in \mathcal{T}_c \wedge t^e(d) > t^E \ \ \forall x \in \hat{D}(r)_i\} && i \in \mathcal{I}, r \in \{m, a\} \\
&DB(r)_i^e = \{d \in \mathcal{D}(r)_i | (t^S \leq t^s(d) \leq t^E + 1) \wedge t^e(d) \in B(r)_i^e\} && i \in \mathcal{I}, r \in \{m, a\} \\
&\sum_{DB(r)_i^e} d \geq 1 && i \in \mathcal{I}, r \in \{m, a\}
\end{aligned} \tag{5.2}
$$

Figure 5.3 shows an example of a candidate move being modified by "RH". The candidate move is $\mathcal{I}_c = \{1, 2, 3\}$, $\mathcal{T}_c = \{2, 3, 4\}$. $t^S = 2$, $t^E = 4$. $m_{216} \in \hat{D}(m)_2$ and $a_{3256} \in \hat{D}(a)_3$ are fixed. $a_{1114} \in \hat{D}(a)_1$ is modified with $B(a)_1^s = \{1\}$, $DB(a)_1^s = \{a_{1111}, a_{1112}, a_{1113}, a_{1114}\}$.

5.1.4 Initial solution

Three initialization methods were conceived.

The first, a heuristic, consists of calling the "SPA" neighborhood for the whole problem, i.e., $SPA(\varnothing, \mathcal{I}, \mathcal{T})$. The second, a MIP model with a time limit, consists of calling the "RH" neighborhood for the whole problem, i.e., $RH(\varnothing, \mathcal{I}, \mathcal{T})$.

The third consists of calling the constructive heuristic we call Greedy Maintenance-First ("maintFirst"), presented in Chapter 3.5.

5.2 Pattern construction

We will use a network to represent the solution space for each aircraft. Let the state s of an aircraft be determined by the values of its rft and rct and let an assignment v be defined by its value a, its start period t^s and its end period t^f. An assignment v can consist of a check, a mission assignment or an empty assignment. In this network, each node n represents an assignment v to an aircraft that ends the assignment in state s (i.e., has state s at the end of period $v.t^f$). An arc (n_1, n_2) represents a transition between two consecutive periods (i.e., between $n_1.t^f$ and $n_2.t^s$). Contrary to other network-based representations in the FMP literature, we keep count of the state of the aircraft in each node with respect to its

maintenance needs, thus creating a much larger graph that can be explored more efficiently. This increased level of detail incurs in costs in terms of time and memory during its generation. These limitations and ways of tackling them are discussed in Section 5.3.4. The concepts of state and assignment are similar to the definitions of state and decision space in Deng et al. [60] for the AMS problem. The key differences are: (1) each state in our network includes the status of a single aircraft instead of the concatenation of status for all aircraft in the fleet, (2) our assignments include mission assignments in addition to check scheduling, (3) we only deal with one check type.

In the next sections, the graph representation will be presented in detail. Section 5.2.1 formally describes the definition of each node and arc, Section 5.2.2 presents the way to create such a graph. Finally, Section 5.2.3 explains how to obtain paths between two nodes by extracting all paths, randomly sampling them or finding good paths.

5.2.1 Graph representation

A Directed Acyclic Graph (DAG) $G(X, V)$ is created for each aircraft. Each node $x \in X$ in the graph represents the concatenation of a state and an assignment, which we name "status". Thus, the node is represented by a unique combination of:

t^s	assignment starting period.
t^f	assignment ending period.
a	assignment value for each period between the start of t^s and the end of t^f.
rft	remaining flight time after the end of period t^f.
rct	remaining calendar time after the end of period t^f.

Each outbound neighbor of a node represents a new status for the aircraft on the immediately next period. This guarantees that the graph will never have any cycles, since we only consider neighbors that are in the future of the current node and, by construction, that have not been visited yet.

Figure 5.4 shows an example of a graph. Nodes are plotted with time in the horizontal axis. Each node shows $t^s/rft/rct$ and a color determined by a: white (empty), gray (check) and green (mission). Each arc shows a of its head node. The leftmost node is the source and represents the initial status of the aircraft at the beginning of the first period. The rightmost node is the sink node and represents the state of the aircraft at the end of the last period (it is unknown).

5.2.2 Graph creation

In order to generate an explicit graph with all possible status for an aircraft, an exhaustive exploration of its states and transitions needs to be realized. This search starts in a node that represents the initial status of the aircraft and that we call source. From this node, we

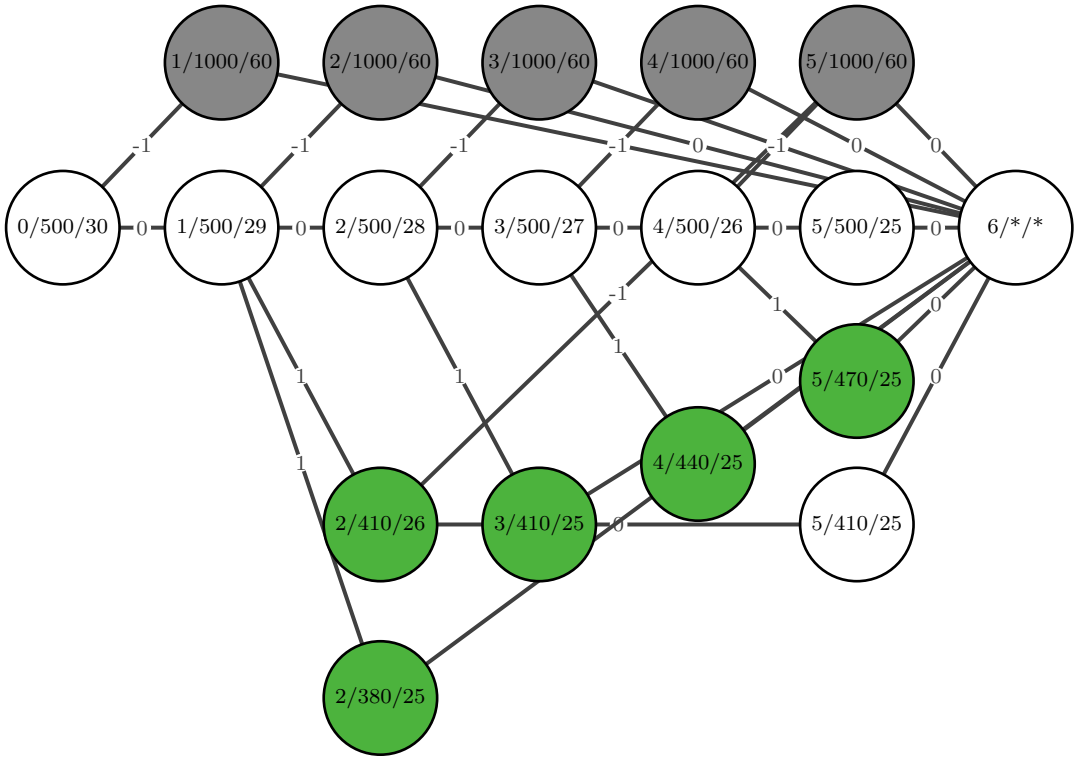

Figure 5.4: DAG of status transitions in a horizon of 5 periods.

apply a depth-first search (DFS) and calculate at each step all the neighbors of the node.

Algorithm 8 shows the procedure executed to create the graph for every aircraft i. The source node is created in Line 2 from the initial status of aircraft i. Then, if the aircraft has any fixed assignments during the initial periods, graph G is initialized with those states in Line 5. We also initialize our list of remaining nodes R with the source node or with the fixed nodes in case they exist. From there, a DFS is started in Line 8: at each iteration, a node n is removed from the list R. If n belongs to the graph G, it means we have already visited this node and nothing is done. If n does not belong to G, all its neighbor nodes are calculated in Line 11 based on maintenance needs and feasible mission assignments. These neighbors are then added as outbound arcs of node n in G and are added to list R. This is repeated until list R is empty. Finally, Line 13 adds one artificial node for each assignment combination (t^s, t^f, a) and one new arc between each existing node in G and one of these artificial nodes, as long as they share the same assignment information. By convention, the state of these new nodes is set to $rft = -1, rct = -1$. In practice this function creates $|X|$ new arcs, since each existing node in G shares the same assignment with one and only one new node.

Algorithm 8: $CreateGraph(i)$

Data:
i: aircraft.
n: current node.
R: list of remaining nodes to visit.
f: fix initial states.
Result:
G: graph, represented by a mapping between nodes and a list of their neighbors.

```
1  begin
2      n ← get_source_node(i)
3      f ← get_fixed_states(i)
4      if |f| > 0 then
5          R ← G[n] ← get_fixed_nodes(f)
6      else
7          R ← [n]
8      while |R| > 0 do
9          n ← R.pop()
10         if n ∉ G then
11             G[n] ← get_neighbors(n)
12             R.Add(G[n])
13     G ← add_artificial_end_nodes(G)
14     return G
```

1	2	3	4	5
-1	-1	-1	-1	-1
0	-1	-1	-1	-1
0	0	-1	-1	-1
0	0	0	-1	-1
0	0	0	0	-1
0	0	0	0	0
0	0	0	0	1
0	0	0	1	1
0	0	1	1	1
0	1	1	1	-1
0	1	1	1	0
0	1	1	1	1

Table 5.1: All possible patterns (paths) between source and sink nodes in Figure 5.4.

5.2.3 Path extraction

Given a Graph $G(X, V)$, a single path between two nodes $x_1 \in X$ and $x_2 \in X$ is equivalent to a pattern, as defined in Section 5.1.2. The set of all paths between two nodes can be obtained with a DFS. Let t^S (t^E) be the first (last) period in time window $\mathcal{T}_c$. Given an existing feasible solution $a_{it} \in \mathbb{Z}$, we map the status of an aircraft i at the period t^S (t^E) with node x_1 (x_2).

Take t^S and x_1 first. Since the assignment a_{it^S} does not necessarily start and end in period t^S, the assignment start $t^{S\prime} \leq t^S$ and end $t^{S\prime\prime} \geq t^S$ periods need to be obtained from the solution. Finally, these periods are used to produce the status of the aircraft: $rft = rft_{it^{S\prime\prime}}$, $rct = rct_{it^{S\prime\prime}}$, $t^s = t^{S\prime}$, $t^f = t^{S\prime\prime}$, $a = a_{it^S}$. In the special case where the assignment is an empty assignment of duration 1 (i.e., $a_{it^S} = 0$), the status mapping is simplified: $rft = rft_{it^S}$, $rct = rct_{it^S}$, $t^s = t^f = t^S$, $a = 0$. Node x_1 is thus obtained using this status mapping.

For node x_2 this procedure needs to be modified since we do not want to fix the end state of the aircraft. Instead, we obtain the artificial node that corresponds to choosing all nodes where the assignment in t^E is the same as the one in the current solution. The status used for x_2 is thus the following: $rft = -1$, $rct = -1$, $t^s = t^{E\prime}$, $t^f = t^{E\prime\prime}$, $a = a_{it^E}$.

Using the example in Figure 5.4, there exist 12 unique paths from source to sink. Table 5.1 shows all those possible patterns for that aircraft: mission assignments (1), checks (-1) and empty assignments (0).

The DFS procedure can be easily modified to render a limited number of paths at each time, by stopping the DFS process after a number of paths is returned. This, nevertheless, results in a biased sampling of the paths since, by construction, the DFS guarantees that consecutive paths share most of the sequence. In addition to this, by default, most DFS implementations use the same sequence of choices given the same graph. This approach is, thus, limited when the size of the sample is small relative to the size of the whole set of paths

to sample.

Unbiased sampling techniques exist. Take a DAG and pre-calculate the number of paths that pass through each node in their trip from x_1 to x_2. This operation can be done in $\mathcal{O}(V)$, where V is the number of edges in between the two nodes. Then, we use this information to sample one neighbor at each node when iterating from x_1. This implies not doing a DFS but starting each path at x_1, which could be inefficient.

Of course, unbiased does not mean good. Another possibility is to take biased samples of paths in a graph, by giving more probability to arcs that are similar to the paths we are looking for (i.e., have nodes that are potentially more useful for the solution). This implies having information about the paths we are looking for from the solution (e.g., the number of errors in the solution). Doing this avoids having to generate very large samples to get a good solution.

Finally, instead of sampling for good paths, one can directly choose the best path. This is easily achieved by giving weights to the arcs so as to obtain the best path that maximizes the current solution. This implies setting the right weights for the arcs and solving a SPP (e.g. with Djikstra). This approach is explained in detail in Section 5.1.3.1.

5.3 Experimentation

This section first presents the instances being used for this study in Section 5.3.1. Then, implementation details regarding the model (Section 5.3.2) and graph creation (Section 5.3.3) are listed. Next, some time and memory considerations when solving instances and generating the graphs are explained in Section 5.3.4. Lastly, the general configuration of the neighborhoods is described in Section 5.3.5.

5.3.1 Instance characteristics

Following the method in Section 3.6 which produces random instances of different sizes, several scenarios are studied, focusing in large instances (i.e., fleets of 60 or more aircraft). The size of the planning horizon for all instances is set to 90 monthly periods.

In order to make comparisons compatible with the previous mathematical models (that can only schedule a maximum of two maintenances), the distance between maintenances is not allowed to be less than half the size of the planning period to guarantee that there will never exist the possibility to plan three maintenances for one aircraft. As a result, the minimum (maximum) distance between maintenances is set to 45 (60).

We group instances with the same fleet size in scenarios that are identified by the size of the fleet. Scenarios for fleet sizes of 60 to 255 aircraft are studied. For each scenario, a number of random instances is solved. These instances are identified by the seed that is used

to generate the input data. For studying the initial solution and the quality of neighbors a single scenario (60) of 10 instances (seeds 81 to 90) is used. For comparing large instances, 6 scenarios (90, 105, 120, 195, 225, 255) of 5 instances each (seeds 8001 to 8005) are studied.

5.3.2 Mathematical model implementation

The mathematical models are generated using python 3.7 and the PuLP library [122] and are solved using CPLEX 12.10 running on a 12-core, 64 GB RAM machine running Linux Fedora 20 with a CPU speed (in MHz) of 2927.000. CPLEX parameters are optimized for the problem using the CPLEX Tuner tool. Threads are limited to 1.

5.3.3 Graph implementation

Graphs are generated once at the beginning of the solving process. These graphs are implemented using python 3.7 and exploited with the graph-tool python library [128]. This library includes several graph algorithms that are executed using efficient implementations in C++ and offers fast numpy array modification of nodes' and edges' attributes. A mapping is done between the nodes in the graph and the status of any given aircraft in order to communicate graph results to and from the graph representation.

5.3.4 Time and memory considerations

Instances for the initial solution study are solved for 10 seconds. Instances for comparing neighborhoods are solved for 5 minutes. Larger instances (90 - 255) are solved for 20 minutes.

Generating and storing one graph per aircraft is too expensive both in terms of time and memory consumption, even after parallelizing the creation of each individual graph. In order to come across these issues, two changes are implemented. The first one re-uses the same graph for all aircraft that belong to the same cluster $k \in \mathcal{K}$ while maintaining a unique source node for the initial conditions of each aircraft. The second is joining nodes with similar rft (by rounding down). This rounding down eliminates possible nodes and edges and so removes parts of a valid solution but does not add any non-valid solutions.

The example in Table 5.2 summarizes the size of three graphs in instance 8001 for 90 periods and 30 aircraft with different rounding coefficients. The original size (i.e., no rounding) is shown with $x = 0$. It is clear that the reduction in the size of the graphs can be considerable: a 90% reduction in the number of nodes and edges when rounding down rft in each node to the nearest 50. The final implementation uses a rounding of 50. Since most missions assignments demand between 10 and 100 flight hours per period and can last several periods, this aggregation is not too excessive.

These modifications reduce the time to generate the graphs from an average of 10 minutes

cluster	N_0	N_5	N_{10}	N_{50}	E_0	E_5	E_{10}	E_{50}
1	1.154.924	607.637	405.120	125.269	3.578.970	1.894.645	1.267.978	393.549
2	624.470	288.380	185.126	57.759	2.010.881	932.348	598.700	185.074
3	1.520.081	705.793	457.281	147.586	4.794.562	2.230.633	1.448.954	470.375

Table 5.2: Graph sizes in number of nodes (N_x) and edges (E_x) for a selected number of rounding coefficients x.

to an average of 2. These times are excluded from the performance comparison in Section 5.4.

5.3.5 Choice of parameters for neighborhood implementations

In the case of the "SPA" method, there is limited additional gain to have more than one aircraft included in the candidate move, so we limit it to one, i.e., we set $|\mathcal{I}_c| = 1$. Because of the good performance of the method, the time window of the candidate is set to the maximum possible, i.e. $\mathcal{T}_c = \mathcal{T}$.

In the case of the "RH" method, the number of aircraft in medium size instances is limited, following previous experiments in Chapter 4, to between 30 and 60. With respect to the size of the time window, the limit is set so as to have a chance of including a complete cycle. This implies having between 20 and 66 periods. Furthermore, we provide the current solution to warm-start the solving process and we stop the solving after 30 seconds or at 2% gap, whichever happens first.

5.4 Results

Several methods are tested following the implementations of the neighbors described in 5.1.3: "SPA", "RH", "maintFirst". The combination of "SPA" and "RH" is called "VND". When applying the method "RH" to the whole problem, i.e., solving the exact mathematical model with CPLEX, we use "MIP".

The section is structured as follows. Section 5.4.1 evaluates each method in terms of the speed and quality of the initial solution obtained, Section 5.4.2 compares the quality of neighborhoods "SPA" and "RH". Finally, Section 5.4.3 compares the performance of "VND" with "MIP" in very large instances.

5.4.1 Finding an initial solution

As a consequence of having elastic constraints for all four coupling constraints (maintenance capacity, serviceability, sustainability and mission assignments), it is relatively easy to find a

instance	maintFirst (%)	MIP (%)	SPA (%)
81	244204	663550	12922
82	770	1957	87
83	905	3191	138
84	1261	5663	306
85	594	2502	150
86	429	3323	94
87	2163	7813	162
88	2412	12189	537
89	54225	656024	11449
90	1606	11057	511

Table 5.3: Comparison on the relative gap of the initial solution for 10 instances and for each method.

feasible solution, even if the quality of that solution may be low.

We compare three constructing heuristics in their quality of initial solution. They are: "SPA", "maintFirst" and "MIP", presented in Section 5.1.4. In the case of "maintFirst" and "MIP" we stop the execution at the time "SPA" takes to find an initial solution, which is 10 seconds for tested instances of size 60. Table 5.3 measures the quality of the initial solution in each method by comparing it with the best solution found for each of the 10 instances. Each value is the relative difference between the objective and the best objective found for that instance, e.g., $\frac{MIP-bestFound}{bestFound} \times 100$ for "MIP".

In 10 seconds it is difficult for a solver to start the cutting phase, yet alone the branching phase for a large sized problem. The consequence is the particularly low quality of the initial integer solutions found by "MIP". It is worth mentioning that, if given more time and for this same size of instances, "MIP" is capable of obtaining high quality integer solutions during the cutting phase, sometimes proving optimality without the need to branch at all. The relative low performance of "maintFirst" confirms findings by [37, 92]: a decomposition based on deciding iteratively high-level maintenance decisions and low-level operations is not well suited to these types of planning problems. As with CG techniques used in [147], "SPA" provides good neighborhoods that balance the high-level maintenances with the low-level mission assignments.

As a result, "SPA" beats both "RH" and "maintFirst" by a large margin in obtaining good quality initial solutions.

5.4.2 Comparing neighborhoods

In order to achieve good quality solutions, a metaheuristic needs to be able to explore the solution space as much as possible, avoiding local minima. In the case of the VND, this depends on the size and types of neighborhoods it uses and how well they complement each

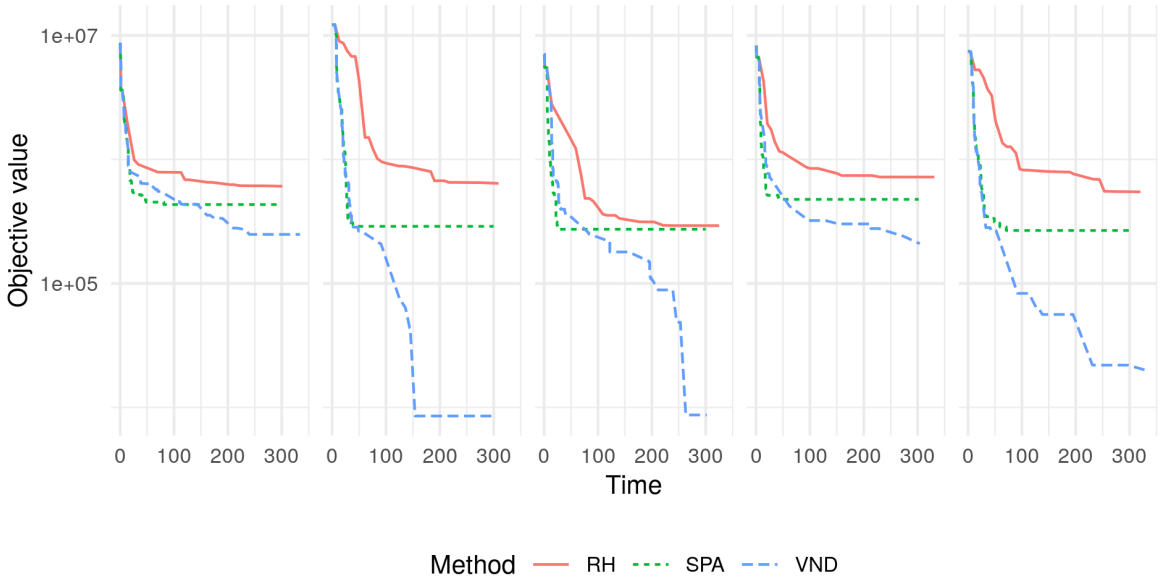

Figure 5.5: The best solution found in the y-axis (in log scale) and the solving time in the x-axis for each method, stopped at 300 s.

other.

In order to evaluate the "RH" and the "SPA" neighborhoods independently, we generate an initial solution using "maintFirst" and then proceed to test three cases: (1) only "SPA", (2) only "RH", and (3) both neighborhoods, i.e., "VND", with a 50:1 probability weight. Each instance is executed 5 times for each case: each of the five executions had a different seed in order to control the random number generators for the initial solution and the neighborhoods. These tests are done in a fleet of 60 aircraft with a solution time limit of 300 seconds. As a proxy for local minimum, we declare reaching a "stable solution" after 70 seconds without any improvement in the best solution found. The neighborhoods are configured in the following way for all tests: "SPA" uses $|\mathcal{I}_c| = 1$ and $|\mathcal{T}_c| = T$. "RH" uses $|\mathcal{T}_c| = 40$ and $|\mathcal{I}_c| = 10$.

Figure 5.5 shows how "SPA" quickly gets into a local minimum, usually of good quality. Each column represents one of 5 random executions for instance 81. "RH" takes longer to stabilize due to the number of possible neighborhoods being greater and also because each iteration takes longer. The quality of "SPA" over "RH" neighborhoods is consistent with [65]. "VND" usually provides a good compromise of the two by sharing the speed of "SPA" and the avoidance of local minima from "RH".

Figures 5.6 and 5.7 show a comparison on the time it takes to reach a stable solution for each method and the quality of that solution. In both cases the x-axis represents each instance. In Figure 5.6 the y-axis is the time. In Figure 5.7 the y-axis is the normalized objective function value in log scale. Each point corresponds to one of the 5 times the instance is solved.

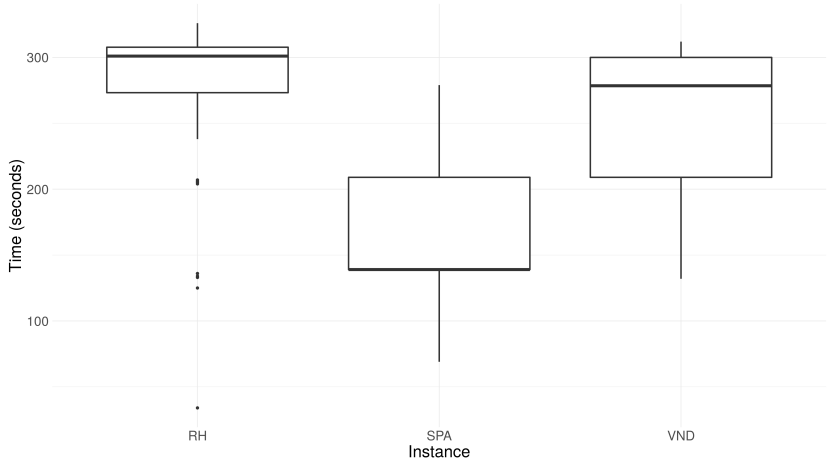

Figure 5.6: Comparison on the time it takes to reach a stable solution for 10 instances solved 5 times each. The x-axis is each method and the y-axis is the time.

It can be seen that "VND" improves substantially the quality of the solution, compared to "RH" and "SPA". In addition, "RH" and "VND" usually avoid falling into local minima, reaching the time limit without falling into a stable solution, while "SPA" does fall into a local minima in less than the time limit. The y-axis in Figure 5.7 has been normalized by dividing each value over the smallest one for each instance.

As in [65], this particular combination of neighborhoods provides gains that surpass the individual advantages of each independent neighborhood. This is achieved because the size of the neighborhoods is large and because the local minima are not correlated between the two neighborhoods.

5.4.3 Very large instances

It is uncommon for a fleet of military aircraft to surpass several dozen aircraft. Nevertheless, in order to test very large instances that could resemble real-life instances of large fleets in FMP problems, we increase the fleet size to hundreds of aircraft to compare the performance of "VND" with "MIP".

Table 5.4 compares the best solutions found after 20 minutes on 5 instances for each large scenario. The dif column represents $\frac{MIP-VND}{MIP} \times 100$. In more than half of the instances, "VND" obtains a solution that is at least 25% smaller. In the two instances where it does not obtain a better solution, "VND" is less than 5% worse than "MIP".

One of the advantages of using an exact method as "MIP" is the possibility of producing a global lower bound. We can then compare how far is each method from the lower bound obtained from "MIP". Table 5.5 shows the percentage gap of each method from the best lower bound found in "MIP". The gap is measured using the solution as a reference, e.g., $\frac{MIP-bestBound}{MIP} \times 100$ for "MIP". In some of the instances, the difference can be quite small for

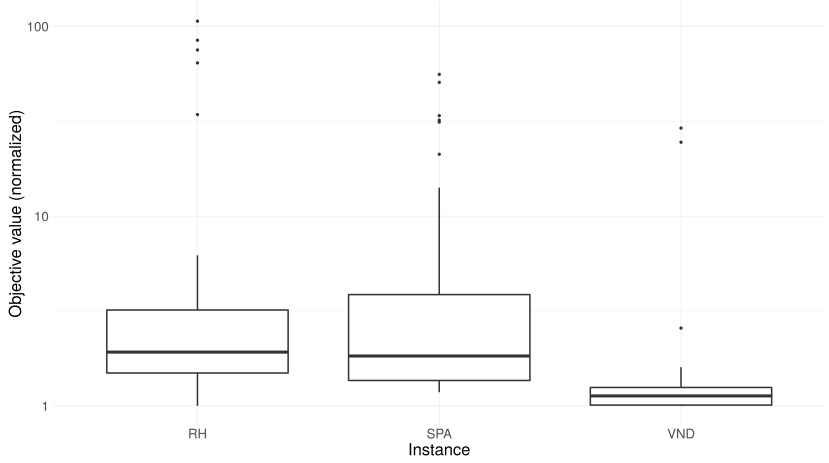

Figure 5.7: Comparison on the quality of the solution when reaching a stable solution for 10 instances solved 5 times each. The x-axis is each method and the y-axis is the normalized objective function value in log scale.

scenario	instance	MIP	VND	dif (%)
90	8001	306658	298621	-2,62
	8002	2104420	2035755	-3,26
	8003	2514723	988814	-60,68
	8004	326232	239640	-26,54
	8005	60550	62571	3,34
105	8001	1018262	734363	-27,88
	8002	1717329	1171060	-31,81
	8003	1833870	14652	-99,20
	8004	285501	270783	-5,16
	8005	630426	633133	0,43
120	8001	874174	727721	-16,75
	8002	1168682	1188695	1,71
	8003	974591	696745	-28,51
	8004	190291	65045	-65,82
	8005	1771422	985650	-44,36

Table 5.4: Comparison on the best solution found for each instance of each scenario by each method.

scenario	instance	MIP (%)	VND (%)
195	8001	73,40	6,57
	8002	13,73	4,24
	8003	12,92	15,46
	8004	36,07	16,98
	8005	96,48	32,34
225	8001	83,76	21,15
	8002	96,00	36,12
	8003	41,14	18,99
	8004	99,28	28,12
	8005	97,45	40,37
255	8001	67,06	33,38
	8003	74,89	24,67
	8004	80,12	30,89
	8005	70,76	28,85

Table 5.5: Percentage gaps between each method and the highest lower bound found by "MIP".

"VND" (5%). In most cases the gaps are reduced considerably with the average gap falling from 67.4% to 24.2% and reducing the maximum gap over all instances from 99% to 40%.

As is expected: as the size of instances gets larger, the increase in relative performance increases too. Figure 5.8 shows the solving process for "MIP" and "VND". The gap in the y-axis is measured using the best integer as a reference, i.e., $\frac{bestFound - bestBound}{bestFound} \times 100$. Most of the times, the initial solution provided by "SPA" is better than the best solution found by "MIP". Even then, the solution is constantly being improved by the combination of the two complementary neighborhoods: the small continuous improvements (especially at the beginning) are usually done by the "SPA" neighborhood, while the discrete jumps later in the solving process are the work of the "RH" neighborhood.

5.5 Conclusions

This chapter describes a matheuristic method to solve the long-term MFMP problem. The method is based on the combination of a RH technique and a Graph-based DP algorithm. The performance is compared with other heuristics and with an exact MIP model using CPLEX.

The results show the advantage of formulating a hybrid approach to solving large-scale combinatorial problems. The method takes advantage of two very different, and thus complementary, neighborhoods to feed a VND. This results in improvements in the quality of the initial solution, the overall solution time and the quality of the best solution found.

As a consequence of the general approach taken to design the technique, it can be adapted

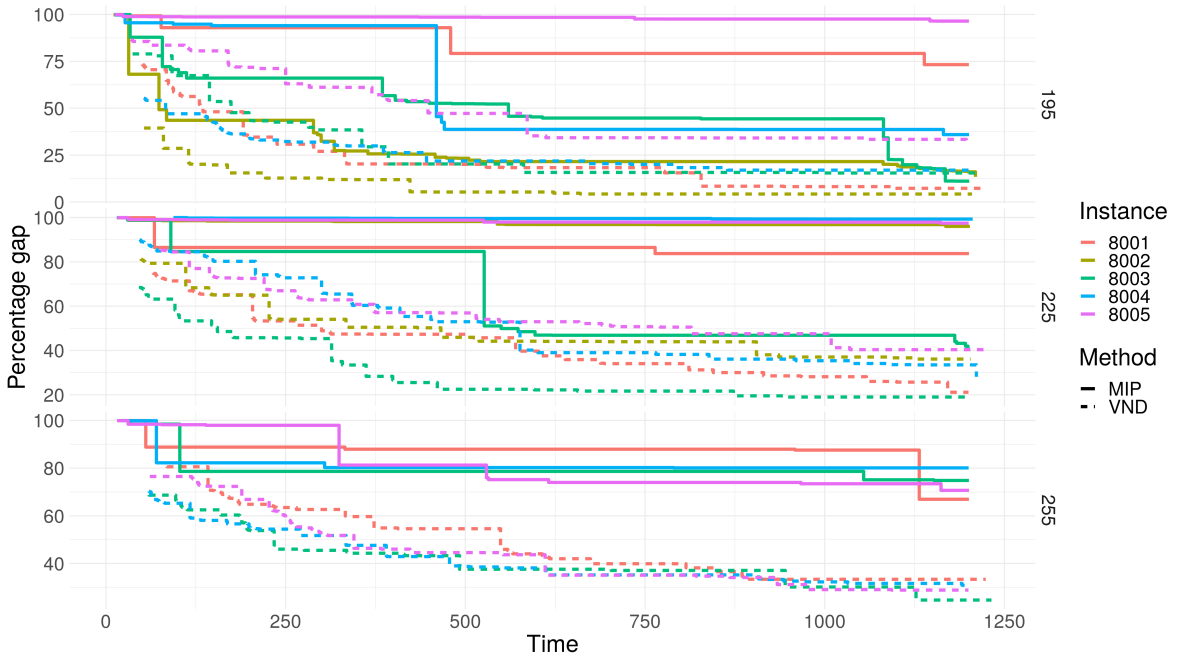

Figure 5.8: The percentage gap for the best solution found so far and the solving time for very large instances between 195 to 255 aircraft.

and applied to the regular (commercial) FMP problem as well as other similar problems such as the NRP and Rolling Stock Assignment and Maintenance Planning problem.

Further work includes different ways to exploit the graph representation of the solution space. For example, obtaining the K shortest paths instead of using the shortest (optimal) one and then sampling one of those good candidate solutions. This sampling could be part of a GRASP-like heuristic. Another possible extension is to use a fast, biased sampling method (as in Chapter 4) to obtain enough patterns to build a set covering problem formulation and solve it as a neighbor alternative to "RH".

As with other heuristics, this methodology is compatible with parallel computing. For example, a parallel multi-start algorithm can be used to generate a better initial solution. Furthermore, each iteration can be turned into a "Best of K" parallel neighbors. Finally, in order to handle the large amounts of memory required to store the graphs, and in order to permit a smoother parallel code, a graph database can be used and queried by several processes in parallel.

Chapter 6

Conclusions and future research

Conclusions

This thesis studies the Military Flight and Maintenance Planning problem and provides several new methods to solve it efficiently. The MFMP problem has not been frequently considered in the literature and in this thesis a deep analysis of its structure, complexity and properties was conducted. A new mathematical formulation was proposed, based on the French Air Force's requirements, and included more sophisticated constraints never before studied together in the MFMP problem.

The first solution approach for this new problem, presented in Chapter 3, is based on a MIP formulation. The model was able to solve small and medium instances up to optimality or near to optimality and to provide solutions with reasonable quality for large instances. A sensitivity analysis was conducted in order to evaluate the impact of the problem's characteristics and size on the final performance of the model. In order to improve the performance of this model, a Simulated Annealing metaheuristic was implemented to find fast initial solutions that are given as a starting point for the MIP model. The experimental tests provided insights on the characteristics that most influence the performance of the solution method. Furthermore, the tests showed the reduction in solution time due to the use of heuristic solutions to warm-start MIP models.

The second solution approach, presented in Chapter 4, combines a new MIP model, efficient valid cuts, and Machine Learning (ML)-based learned cuts. The new MIP model used a pattern-like formulation. Such a formulation generates a great number of variables, making the problem considerably larger, but due to tighter constraints, the linear relaxation is improved. The valid cuts used the initial status of the fleet and the relevant information on the mission requirements. The ML-based learned cuts were trained on thousands of small instances to find relationships between the input data features and specific characteristics of the optimal solutions. This learned information was used to apply invalid cuts to the solution space of large instances. These cuts reduced considerably the problem size without excluding good quality solutions from the solution space. The numerical tests showed significant improvements in the performance of the solution process. This work is one of the first in the field that successfully uses supervised learning techniques to predict characteristics of optimal solutions on new instances of well-known problems and applies it to MIP models.

The third and last solution approach, presented in Chapter 5, is conceived to handle very

large instances efficiently. It is based on a Variable Neighborhood Descent (VND) matheuristic that combines two types of exact neighborhoods: a Rolling Horizon (RH) technique and a Dynamic Programming (DP) algorithm. The DP algorithm used a graph-based explicit enumeration of states for each aircraft with respect to the next maintenance. As a consequence, good quality patterns were efficiently generated for each aircraft by solving a Shortest Path Problem in a Directed Acyclic Graph. The combination of time-based partitions (RH) and aircraft-based ones (DP) was shown to be particularly successful at avoiding local minima and reaching near-optimal solutions in very short time with comparison to exact methods. This work demonstrates the potential of combining efficient exact methods and metaheuristics to successfully solve large instances of the MFMP problem.

Future research

The MFMP studied in this thesis is inspired by the French Air Force actual needs of the Mirage 2000 fleet maintenance scheduling and mission planning. In this regard, many of the characteristics of the problem are a result of discussions with the DGA and Armée de l'Air. Nevertheless, some variants of this base problem exist that could be explored. The first one is the possibility of long term storage of aircraft. This option lets the aircraft its calendar-based check frequency requirements, i.e., an stored aircraft cannot fly and is "paused" in time and so does not deteriorate. The second one is using a flexible man-hour maintenance capacity. This technique, often used in the FMP and medium term MFMP problems implies modeling the capacity of the maintenance workshop as a total number of working hours per period.

The proof of complexity in Chapter 3 was specific for the long term problem, using minimum consecutive assignments of missions to aircraft. This proof is not compatible with short and medium term formulations where missions are usually continuous assignments of flight hours per period. Nevertheless, alternative proofs using the scheduling of maintenances could be explored for these problems. In particular, special cases where the problem becomes polynomial can be identified. This is probably the case when the mission requirements are continuous flight hours and the maintenance operations are already planned (a flight planning problem under maintenance constraints).

When dealing with planning horizons in the size of several years, consideration of uncertainties is crucial. By guaranteeing certain levels of availability and sustainability at each period per sub-fleet, the solutions found by our methods provide already a certain level of robustness. Nevertheless, if more information can be available on the uncertainty sources, the robustness of a solution can be improved using techniques tailored to the nature of that uncertainty.

The potential of the application of Machine Learning techniques to optimization methods is considerable. In Chapter 4, our attention was centered on a very specific application of well known techniques of supervised learning to gain information on the optimal solutions. Other techniques should also be tested. For example, logistic regression can be used for the estimation of the probability that a given solution pattern (i.e., variable) takes part in the

optimal solution. This probability can then be used to sample patterns and, thus, solutions. Another example is the use of automatic feature-selection techniques to obtain, faster and easier, a better set of input features to predict the optimal patterns.

Concerning the application of the ML information, a clear separation between the prediction part and the optimization one was maintained: good patterns are first predicted for a given instance and then used to solve the problem with these insights. An alternative is to apply the prediction inside of the solution process by sampling patterns as part of a broader decomposition technique where sampling is used as a first step to solve a subproblem. In this case, sampling can be adapted by new information coming from the current solution, such as constraint violations or shadow prices.

Decomposition techniques that can benefit from this kind of sampling are the ones where the number of decision variables grows exponentially with respect to the problem size. An evident candidate for this are string representations of variables such as the ones used in CG decompositions. Another potential good candidate is the one were a very large explicit graph is built for each possible state of each aircraft.

Directed Acyclic Graphs can actually be used to sample potentially interesting patterns. By assigning weights to edges and setting a maximum distance K for the extracted patterns, one can put a floor on the quality of the extracted paths. Then, by carefully choosing the probabilities to sample neighbors of a node, the result becomes an unbiased sampling of paths that have a quality better than K.

Finally, such a sampling can be also used for different purposes. One is to pass the sampled patterns to a set-covering master model that is compatible with the routing constraints found in FMP and VRP problems. A second option is to integrate the sampled patterns in the constructive phase of a GRASP heuristic in order to produce many fast and reasonably good solutions that can later be improved with local search.

Appendix A

Time-related index sets

$$
\begin{aligned}
\mathcal{T}_i^{m_{Init}} &= \Big\{ t \in \{1, ..., \max\{0, Rct_i^{Init} - E^{max} + E^{min}\}\} \Big\} \\
\mathcal{T}_{t'}^{m} &= \Big\{ t \in \{t', ..., \min\{T, t + M + E^{min} - 1\}\} \Big\} \\
\mathcal{T}_{t'}^{MM} &= \begin{cases} t \in \{t' + M + E^{min} - 1, ..., t' + M + E^{max} - 1\} & t' \leq T - E^{max} - M \\ \emptyset & t' > T - E^{max} - M \end{cases} \\
\mathcal{T}_i^{M_{Init}} &= \Big\{ t \in \{ \max\{0, Rct_i^{Init} - E^{max} + E^{min}\}, ..., Rct_i^{Init}\} \Big\} \\
\mathcal{T}_{t'}^{s} &= \Big\{ t \in \{ \max\{1, t' - M + 1\}, ..., t'\} \Big\} \\
\mathcal{T}_{t'}^{M} &= \{t \in \{t' + M + E^{min} - 1, ..., t' + M + E^{max} - 1\} \cap \{T - E^{max} + 1, ..., T - 1\}\} \\
\mathcal{T}_{t'}^{M+} &= \begin{cases} \mathcal{T}_{t'}^{M} & t' + M > T \vee t' + M + E^{max} - 1 < T \\ \mathcal{T}_{t'}^{M} \cup \{T\} & t' + M \leq T \leq t' + M + E^{max} - 1 \end{cases} \\
\mathcal{TT}_t &= \{(t_1, t_2) \mid t_1 \in \mathcal{T}_t^s \vee t_2 \in \mathcal{T}_t^s\} \\
\mathcal{T}_{jt'}^{MT} &= \Big\{ t \in \{ \max\{1, t' - MT_j^{min} + 1\}, ..., t'\} \Big\} \\
\mathcal{TT}_j &= \{t, t' \in \mathcal{T}_j \mid t + MT_j^{min} - 1 \leq t' \leq t + MT_j^{max} - 1\} \\
\mathcal{TTJ}_{jt} &= \{t_1, t_2 \in \mathcal{TT}_j \mid t_1 \leq t \leq t_2\}. \\
\mathcal{JTT}_{it_1t_2} &= \{(j, t, t') | j \in \mathcal{J}_i \wedge (t, t') \in \mathcal{TT}_j \wedge t \geq t_1 + M \wedge t' < t_2\}
\end{aligned}
$$

Consider a small example where $M = 2, Rct_1^{Init} = 5, E^{max} = 7, E^{min} = 4, MT_1^{min} = 3, MT_1^{max} = 4, T = 15, \mathcal{T}_1 = \{4, ..., 10\}, \mathcal{I}_1 = \{1\}$ then, the example solution in Figure A.1 should comply with the following:

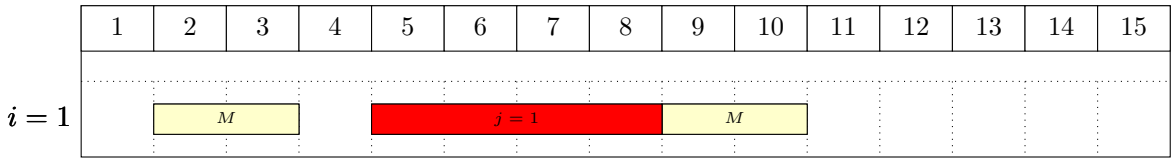

Figure A.1: A solution to the small example problem that is feasible with respect to calendar constraints on assignments and checks.

$$
\begin{aligned}
\mathcal{T}_1^{M_{Init}} &= \{2, ..., 5\} \\
\mathcal{T}_3^{s} &= \{2, 3\} \\
\mathcal{T}_2^{M} &= \{7, ..., 10\} \\
\mathcal{T}_7^{M} &= \{12, ..., 14\} \\
\mathcal{T}_9^{M+} &= \{14, 15\} \\
\mathcal{TTT}_3 &= \{(2,7), ...(2,10), (3,8)....(3,11)\} \\
\mathcal{TT}_1 &= \{(4,6), (4,7), (5,7), (5,8), ..., (8,10)\} \\
\mathcal{TTJ}_{15} &= \{(4,6), (4,7), (5,7), (5,8)\} \\
\mathcal{JTT}_{129} &= \{(1,4,6), (1,4,7), (1,5,7), (1,5,8), (1,6,8)\}
\end{aligned}
$$

Figure A.1 shows an example solution that complies with the time-related indexed sets. Aircraft 1 has a first check in period $2 \in \mathcal{T}_1^{M_{Init}}$. A second check is done in period $9 \in \mathcal{T}_2^{M}$. Also, since $(2, 9) \in \mathcal{TTT}_3$, aircraft 1 is considered in maintenance in period 3. The aircraft has an assignment to mission 1 in periods $(5, 8) \in \mathcal{TT}_1$. Since $(5, 8) \in \mathcal{TTJ}_{15}$, aircraft 1 is considered assigned to mission 1 during period 5. Finally, since maintenances are done in periods $(2, 9)$, all possible mission assignments between (e.g., $(1, 5, 8)$) should be in $\mathcal{JTT}_{129}$.

Appendix B

Instance data specification

Each instance studied in this thesis has the same standard format specified in Figure B.1. The structure has been generalized so it can be used and compared with other similar problems: Tasks (missions), Resources (aircraft) and Maintenances (checks).

The structure allows for several different types of Maintenances, each one with its own frequency in terms of calendar periods and flight hours, duration and needs of maintenance resources. Maintenances are done in what we call a Workshop: each workshop has its own capacity that can vary according to the period in the year and each maintenance can only be done at its workshop. Finally, there is a dependency between the different maintenances: there are some (overhaul maintenances) that reset the counters for the others. For this, we have the 'affects' and 'depends' attributes.

The initial status of an aircraft "RInitial" allows for Rct_{im}^{Init} ('elapsed') and Rft_{im}^{Init} ('used') for each aircraft i and each maintenance m. The minimum amount of flight hours done in each period U_{it}^{min} can vary according to each aircraft i and each period t.

Regarding the non-dimensional parameters for an instance, they are the following: num_period, start, seed, min_hours_perc and min_avail_percent.

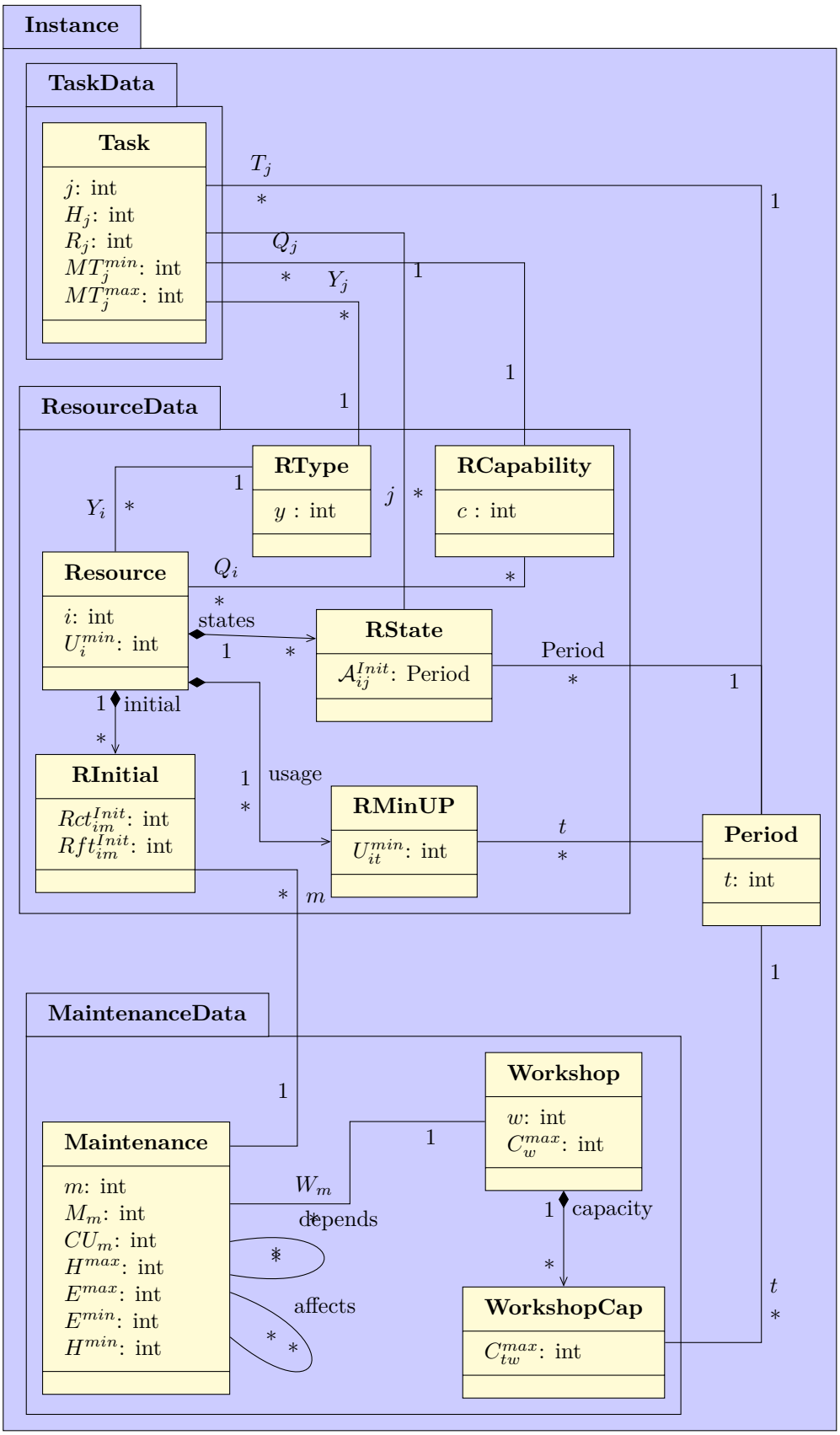

Figure B.1: Set of data objects that constitute an Instance.

Bibliography

[1] Un tournant dans le cycle d'entretien des Mirage 2000, 2011. URL https://www.defense.gouv.fr/actualites/economie-et-technologie/un-tournant-dans-le-cycle-d-entretien-des-mirage-2000. (Cited on pages v, xvi, and 6.)

[2] L'obsolescence programmée est un délit. *cnetfrance.fr*, 2015. URL https://www.cnetfrance.fr/news/l-obsolescence-programmee-est-un-delit-39822874.htm. (Cited on page 4.)

[3] Maintenance Reserves Need to Account for Realistic D Check Costs. *Aircraft Value News*, 2018. URL https://www.aircraftvaluenews.com/maintenance-reserves-need-to-account-for-realistic-d-check-costs/. (Cited on pages xvi and 9.)

[4] Participer au Défi "OptiPlan" pour l'amélioration de la planification de la maintenance des équipements militaires, 2018. URL https://www.defense.gouv.fr/dga/actualite/participer-au-defi-optiplan-pour-l-amelioration-de-la-planification-de-la-maintenance-des-equipements-militaires. (Cited on page 8.)

[5] Repair is as important as innovation. *The Economist*, 2018. URL https://www.economist.com/finance-and-economics/2018/10/20/repair-is-as-important-as-innovation. (Cited on page 3.)

[6] Why You Need To Adjust Your Monthly Budget For Home Maintenance. *Forbes*, 2018. URL https://www.forbes.com/sites/juliadellitt/2018/06/20/why-you-need-to-adjust-your-monthly-budget-for-home-maintenance. (Cited on pages xv and 3.)

[7] EU brings in 'right to repair' rules for appliances. *The BBC*, 2019. URL https://www.bbc.com/news/business-49884827. (Cited on page 3.)

[8] Artificial intelligence is changing every aspect of war. *The Economist*, 2019. URL https://www.economist.com/science-and-technology/2019/09/07/artificial-intelligence-is-changing-every-aspect-of-war. (Cited on pages xv, 4, and 8.)

[9] Airbus and COSLING provide software solution Optaforce for Mirage 2000 maintenance, 2019. URL https://www.airbus.com/newsroom/press-releases/en/2019/06/airbus-and-cosling-provide-software-solution-optaforce-for-mirage-2000-maintenance.html. (Cited on pages xix and 28.)

[10] T. Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009. ISSN 18672949. doi: 10.1007/s12532-008-0001-1. (Cited on page 28.)

[11] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, jan 2005. ISSN 01676377. doi: 10.1016/j.orl.2004.04.002. (Cited on page 26.)

[12] T. Achterberg, R. E. Bixby, Z. Gu, E. Rothberg, and D. Weninger. Presolve reductions in mixed integer programming. *INFORMS Journal on Computing*, 32(2):473–506, mar 2020. ISSN 15265528. doi: 10.1287/ijoc.2018.0857. (Cited on page 25.)

[13] T. Adamo, G. Ghiani, A. Grieco, E. Guerriero, and E. Manni. MIP neighborhood synthesis through semantic feature extraction and automatic algorithm configuration. *Computers and Operations Research*, 83:106–119, 2017. ISSN 03050548. doi: 10.1016/j.cor.2017.01.021. (Cited on page 31.)

[14] T. Adamo, G. Ghiani, E. Guerriero, and E. Manni. Automatic instantiation of a Variable Neighborhood Descent from a Mixed Integer Programming model. *Operations Research Perspectives*, 4:123–135, 2017. ISSN 22147160. doi: 10.1016/j.orp.2017.09.001. (Cited on page 31.)

[15] E. H. Aghezzaf and N. M. Najid. Integrated production planning and preventive maintenance in deteriorating production systems. *Information Sciences*, 178 (17):3382–3392, 2008. ISSN 00200255. doi: 10.1016/j.ins.2008.05.007. URL `https://ac.els-cdn.com/S0020025508001473/1-s2.0-S0020025508001473-main.pdf?{_}tid=adf118f0-c48b-11e7-856c-00000aacb35d{&}acdnat=1510149166{_}2a7a4a5c73ddd1f2ffaf83a8169de014`. (Cited on page 26.)

[16] R. K. Ahuja, Ö. Ergun, J. B. Orlin, and A. P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1-3):75–102, nov 2002. ISSN 0166218X. doi: 10.1016/S0166-218X(01)00338-9. (Cited on page 30.)

[17] R. K. Ahuja, Ö. Ergun, J. B. Orlin, and A. P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1-3):75–102, 2002. (Cited on pages xix and 26.)

[18] D. Anghinolfi, L. M. Gambardella, R. Montemanni, C. Nattero, M. Paolucci, and N. E. Toklu. A matheuristic algorithm for a large-scale energy management problem. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7116 LNCS, pages 173–181. Springer, Berlin, Heidelberg, 2012. ISBN 9783642298424. doi: 10.1007/978-3-642-29843-1_19. URL `https://link.springer.com/chapter/10.1007/978-3-642-29843-1{_}19`. (Cited on page 30.)

[19] K. R. Apt. Principles of Constraint Programming, 2000. (Cited on page 28.)

[20] E. M. Arkin and E. B. Silverberg. Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics*, 18(1):1–8, sep 1987. ISSN 0166218X. doi: 10.1016/0166-218X(87)90037-0. (Cited on pages xx and 38.)

[21] M. A. Awadallah, A. L. A. Bolaji, and M. A. Al-Betar. A hybrid artificial bee colony for a nurse rostering problem. *Applied Soft Computing Journal*, 35:726–739, aug 2015. ISSN 15684946. doi: 10.1016/j.asoc.2015.07.004. (Cited on page 29.)

[22] M. N. Azaiez and S. S. Al Sharif. A 0-1 goal programming model for nurse scheduling. *Computers and Operations Research*, 32(3):491–507, mar 2005. ISSN 03050548. doi: 10.1016/S0305-0548(03)00249-1. (Cited on page 27.)

[23] K. R. Baker. Workforce Allocation in Cyclical Scheduling Problems: A Survey. *Operational Research Quarterly (1970-1977)*, 27(1):155, 1976. ISSN 00303623. doi: 10.2307/3009134. (Cited on page 22.)

[24] P. Ball, K. Hoffman, R. R. I. Conference, U. Washington, U. DC, and U. 1996. The use of column generation in solving very large fleet assignment problems. In *INFORMS Conference*, Washington, DC, 1996. (Cited on pages xix and 26.)

[25] J. F. Bard and H. W. Purnomo. Preference scheduling for nurses using column generation. *European Journal of Operational Research*, 164(2):510–534, jul 2005. ISSN 03772217. doi: 10.1016/j.ejor.2003.06.046. (Cited on page 27.)

[26] C. Barnhart, N. L. Boland, L. W. Clarke, E. L. Johnson, G. L. Nemhauser, and R. G. Shenoi. Flight string models for aircraft fleeting and routing. *Transportation Science*, 32 (3):208–220, aug 1998. ISSN 00411655. doi: 10.1287/trsc.32.3.208. (Cited on pages xviii, 7, and 13.)

[27] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998. ISSN 0030364X. doi: 10.1287/opre.46.3.316. (Cited on pages xix and 26.)

[28] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le. Neural optimizer search with reinforcement learning, 2017. URL `https://dl.acm.org/citation.cfm?id=3305429`. (Cited on pages xix and 31.)

[29] Y. Bengio, A. Lodi, and A. Prouvost. Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon. *European Journal of Operational Research*, nov 2020. ISSN 03772217. doi: 10.1016/j.ejor.2020.07.063. URL `http://arxiv.org/abs/1811.06128`. (Cited on pages xix and 31.)

[30] P. Beraldi, G. Ghiani, A. Grieco, and E. Guerriero. Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling problem with sequence-dependent set-up costs. *Computers & Operations Research*, 35(11):3644–3656, 2008. (Cited on page 27.)

[31] I. Berrada, J. A. Ferland, and P. Michelon. A multi-objective approach to nurse scheduling with both hard and soft constraints. Technical Report 3, 1996. (Cited on page 27.)

[32] S. Bertels and T. Fahle. A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. *Computers & operations research*, 33(10):2866–2890, 2006. (Cited on page 31.)

[33] B. Bilgin, P. De Causmaecker, B. Rossie, and G. Vanden Berghe. Local search neighbourhoods for dealing with a novel nurse rostering model. *Annals of Operations Research*, 194(1):33–57, nov 2012. ISSN 15729338. doi: 10.1007/s10479-010-0804-0. URL https://link-springer-com.rev-doc.isae.fr/article/10.1007/s10479-010-0804-0. (Cited on page 29.)

[34] E. R. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. MIP: Theory and Practice — Closing the Gap. pages 19–49. 2000. doi: 10.1007/978-0-387-35514-6_2. (Cited on page 26.)

[35] R. Bixby, ·. E. Rothberg, R. Bixby, . E. Rothberg, and E. Rothberg. Progress in computational mixed integer programming-A look back from the other side of the tipping point. *Ann Oper Res*, 149:37–41, 2007. doi: 10.1007/s10479-006-0091-y. URL https://idp.springer.com/authorize/casa?redirect{_}uri=https://link.springer.com/content/pdf/10.1007/s10479-006-0091-y.pdf{&}casa{_}token=3F4SJIM{_}TawAAAAA:k6DFZ3fxeXRZo87R8VmI9GOLQYp4{_}IP7tmplU4o-s5KePji32HZw1e5cAXn-Ez67xylmJKrIuo-ilbvhJQ. (Cited on page 26.)

[36] S. Bourdais, P. Galinier, and G. Pesant. HIBISCUS: A constraint programming application to staff scheduling in health care. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 2833, pages 153–167. Springer, Berlin, Heidelberg, 2003. doi: 10.1007/978-3-540-45193-8_11. URL http://link.springer.com/10.1007/978-3-540-45193-8{_}11. (Cited on page 28.)

[37] F. Brandt, R. Bauer, M. Völker, and A. Cardeneo. A constraint programming-based approach to a large-scale energy management problem with varied constraints: A solution approach to the ROADEF/EURO Challenge 2010. *Journal of Scheduling*, 16(6): 629–648, 2013. ISSN 10946136. doi: 10.1007/s10951-012-0281-1. (Cited on pages 30 and 105.)

[38] P. Brucker, R. Qu, and E. Burke. Personnel scheduling: Models and complexity. *European Journal of Operational Research*, 210(3):467–473, may 2011. ISSN 03772217. doi: 10.1016/j.ejor.2010.11.017. URL https://www.sciencedirect.com/science/article/pii/S0377221710007897. (Cited on pages xviii and 22.)

[39] J. O. Brunner, J. F. Bard, and J. M. Köhler. Bounded flexibility in days-on and days-off scheduling. *Naval Research Logistics*, 60(8):678–701, dec 2013. ISSN 0894069X. doi: 10.1002/nav.21561. URL http://doi.wiley.com/10.1002/nav.21561. (Cited on pages xviii, 22, and 23.)

[40] E. K. Burke and T. Curtois. New approaches to nurse rostering benchmark instances. *European Journal of Operational Research*, 237(1):71–81, aug 2014. ISSN 03772217. doi: 10.1016/j.ejor.2014.01.039. (Cited on page 27.)

[41] E. K. Burke, P. De Causmaecker, G. V. Berghe, and H. Van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7(6):441–499, 2004. ISSN 10946136. doi: 10.1023/B:JOSH.0000046076.75950.0b. (Cited on page 22.)

[42] E. K. Burke, T. Curtois, R. Qu, and G. Vanden Berghe. A scatter search methodology for the nurse rostering problem. *Journal of the Operational Research Society*, 61(11): 1667–1679, nov 2010. ISSN 14769360. doi: 10.1057/jors.2009.118. URL `https://link.springer.com/article/10.1057/jors.2009.118`. (Cited on page 29.)

[43] E. K. Burke, J. Li, and R. Qu. A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems. *European Journal of Operational Research*, 203(2):484–493, jun 2010. ISSN 03772217. doi: 10.1016/j.ejor.2009.07.036. (Cited on page 30.)

[44] S. Çakırgil, E. Yücel, and G. Kuyzu. An integrated solution approach for multi-objective, multi-skill workforce scheduling and routing problems. *Computers and Operations Research*, 118:104908, jun 2020. ISSN 03050548. doi: 10.1016/j.cor.2020.104908. (Cited on page 30.)

[45] A. Caprara, M. Monaci, and P. Toth. Models and algorithms for a staff scheduling problem. *Mathematical Programming*, 98(1-3):445–476, sep 2003. ISSN 00255610. doi: 10.1007/s10107-003-0413-7. URL `http://link.springer.com/10.1007/s10107-003-0413-7`. (Cited on pages xviii and 22.)

[46] N. Chapados, M. Joliveau, and L.-M. Rousseau. Retail store workforce scheduling by expected operating income maximization. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pages 53–58. Springer, 2011. (Cited on page 28.)

[47] B. Cheang, H. Li, A. Lim, and B. Rodrigues. Nurse rostering problems - A bibliographic survey. *European Journal of Operational Research*, 151(3):447–460, dec 2003. ISSN 03772217. doi: 10.1016/S0377-2217(03)00021-3. (Cited on page 22.)

[48] P. Y. Cho. *Optimal scheduling of fighter aircraft maintenance*. PhD thesis, Massachusetts Institute of Technology, 2011. URL `https://dspace.mit.edu/handle/1721.1/67773?show=full`. (Cited on pages xviii, 16, 17, 20, 21, 26, 27, 31, and 45.)

[49] R. Cipriano, L. Di Gaspero, and A. Dovier. Hybrid approaches for rostering: A case study in the integration of constraint programming and local search. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4030 LNCS, pages 110–123. Springer Verlag, oct 2006. ISBN 3540463844. doi: 10.1007/11890584_9. (Cited on page 30.)

[50] L. Clarke, E. Johnson, G. Nemhauser, and Z. Zhu. The aircraft rotation problem. *Annals of Operations Research*, 69(0):33–46, 1997. ISSN 02545330. doi: 10.1016/0965-8564(96)81095-0. URL `http://link.springer.com/10.1023/A:1018945415148`. (Cited on page 13.)

[51] J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, J. C. Smith, M. Fischetti, and A. Lodi. Heuristics in Mixed Integer Programming. In *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc., feb 2011. doi: 10.1002/9780470400531.eorms0376. (Cited on page 26.)

[52] Cook, A.J. and Tanner, G. Innovative Cooperative Actions of R&D in EUROCONTROL Programme CARE INO III: Dynamic Cost Indexing: Aircraft maintenance – marginal delay costs. Technical report. URL https://westminsterresearch.westminster.ac.uk/item/9123x/innovative-cooperative-actions-of-r-d-in-eurocontrol-programme-care-ino-iii-dynamic-cost-indexing-aircraft-maintenance-marginal-delay-costs. (Cited on page 5.)

[53] M.-C. Côté, B. Gendron, C.-G. Quimper, and L.-M. Rousseau. Formal languages for integer programming modeling of shift scheduling problems. *Constraints*, 16(1):54–76, 2011. (Cited on page 31.)

[54] P. De Causmaecker and G. Vanden Berghe. A categorisation of nurse rostering problems. In *Journal of Scheduling*, volume 14, pages 3–16. Springer, feb 2011. doi: 10.1007/s10951-010-0211-z. (Cited on page 22.)

[55] P. De Chastellux, N. Dupin, and P. Bazot. Planification optimisée de maintenances d'aéronefs militaires. Technical report. URL http://roadef2017.event.univ-lorraine.fr/abstracts/ROADEF2017{_}paper{_}8.pdf. (Cited on pages xviii, 15, 17, 26, 27, and 31.)

[56] R. Dechter and Others. *Constraint processing*. Morgan Kaufmann, 2003. (Cited on page 28.)

[57] F. Della Croce and F. Salassa. A variable neighborhood search based matheuristic for nurse rostering problems. *Annals of Operations Research*, 218(1):185–199, nov 2014. ISSN 15729338. doi: 10.1007/s10479-012-1235-x. (Cited on page 30.)

[58] S. Demassey, G. Pesant, and L.-M. Rousseau. A cost-regular based hybrid column generation approach. *Constraints*, 11(4):315–333, 2006. (Cited on page 31.)

[59] S. den Hartog. On the Complexity of Nurse Scheduling Problems. may 2016. (Cited on page 22.)

[60] Q. Deng, B. F. Santos, and R. Curran. A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization. *European Journal of Operational Research*, 281(2):256–273, aug 2020. ISSN 03772217. doi: 10.1016/j.ejor.2019.08.025. URL https://www.sciencedirect.com/science/article/pii/S0377221719306782. (Cited on pages 13, 27, and 98.)

[61] K. Doganay and M. Bohlin. Maintenance plan optimization for a train fleet. In *WIT Transactions on the Built Environment*, volume 114, pages 349–358. WIT Press, jul 2010. ISBN 9781845644680. doi: 10.2495/CR100331. (Cited on pages xviii, 23, and 27.)

[62] K. A. Dowsland and J. M. Thompson. Solving a nurse scheduling problem with knapsacks, networks and tabu search. *Journal of the Operational Research Society*, 51(7):825–833, jul 2000. ISSN 14769360. doi: 10.1057/palgrave.jors.2600970. (Cited on page 30.)

[63] S. Dožić, A. Jelović, M. Kalić, and M. Čangalović. Variable Neighborhood Search to solve an airline fleet sizing and fleet assignment problem. *Transportation Research Procedia*, 37:258–265, 2019. ISSN 23521465. doi: 10.1016/j.trpro.2018.12.191. (Cited on page 29.)

[64] N. Dupin and E. ghazali Talbi. Parallel matheuristics for the discrete unit commitment problem with min-stop ramping constraints. *International Transactions in Operational Research*, 27(1):219–244, jan 2020. ISSN 14753995. doi: 10.1111/itor.12557. (Cited on page 26.)

[65] N. Dupin and E.-G. Talbi. Matheuristics to optimize maintenance scheduling and refueling of nuclear power plants. *Journal of heuristics*, pages 1–29, 2018. URL `http://arxiv.org/abs/1812.08598`. (Cited on pages 30, 106, and 107.)

[66] N. Dupin and E.-G. Talbi. Machine Learning-Guided Dual Heuristics and New Lower Bounds for the Refueling and Maintenance Planning Problem of Nuclear Power Plants. *Algorithms*, 13(8):185, jul 2020. ISSN 1999-4893. doi: 10.3390/a13080185. URL `https://www.mdpi.com/1999-4893/13/8/185`. (Cited on page 32.)

[67] W. El Moudani and F. Mora-Camino. A dynamic approach for aircraft assignment and maintenance scheduling by airlines. *Journal of Air Transport Management*, 6(4): 233–237, oct 2000. ISSN 09696997. doi: 10.1016/S0969-6997(00)00011-9. (Cited on page 13.)

[68] T. A. Feo and J. F. Bard. Flight Scheduling and Maintenance Base Planning. *Management Science*, 35(12):1415–1432, dec 1989. ISSN 0025-1909. doi: 10.1287/mnsc.35.12.1415. (Cited on page 13.)

[69] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995. (Cited on page 29.)

[70] J. A. Filar, P. Manyem, and K. White. How Airlines and Airports Recover from Schedule Perturbations: A Survey. *Annals of Operations Research*, 108(1-4):315–333, 2001. ISSN 15729338. doi: 10.1023/A:1016079600083. (Cited on pages 15 and 27.)

[71] M. Fischetti and M. Fraccaro. Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks. *Computers & Operations Research*, 106:289–297, jun 2019. ISSN 0305-0548. doi: 10.1016/J.COR.2018.04.006. URL `https://www.sciencedirect.com/science/article/pii/S0305054818300893`. (Cited on pages 31 and 32.)

[72] M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, sep 2005. ISSN 00255610. doi: 10.1007/s10107-004-0570-3. (Cited on page 26.)

[73] I. &. S. for Business. UK Aerospace Maintenance, Repair, Overhaul & Logistics Industry Analysis. *UK Government Department for Business, Innovation & Skills*, 2017. URL `https://assets.publishing.service.gov.uk/government/uploads/system/`

uploads/attachment{_}data/file/502588/bis-16-132-uk-mrol-analysis.pdf. (Cited on page 4.)

[74] S. Gabteni and M. Grönkvist. Combining column generation and constraint programming to solve the tail assignment problem. *Annals of Operations Research*, 171(1): 61–76, oct 2009. ISSN 02545330. doi: 10.1007/s10479-008-0379-1. URL https://link.springer.com/article/10.1007/s10479-008-0379-1. (Cited on pages xix and 28.)

[75] R. M. Garcia. Optimized Procurement and Retirement Planning of Navy Ships and Aircraft, 2001. URL https://apps.dtic.mil/docs/citations/ADA401106. (Cited on page 18.)

[76] F. Gardi, K. N. E. C. o. E. Computation, and undefined 2011. Local Search for Mixed-Integer Nonlinear Optimization: A Methodology and an Application. *Springer*, 6622 LNCS:167–178, 2014. doi: 10.1007/978-3-642-20364-0_15. URL https://www.researchgate.net/publication/225150342. (Cited on page 29.)

[77] A. Gavranis and G. Kozanidis. An exact solution algorithm for maximizing the fleet availability of a unit of aircraft subject to flight and maintenance requirements. *European Journal of Operational Research*, 242(2):631–643, apr 2015. ISSN 03772217. doi: 10.1016/j.ejor.2014.10.016. (Cited on pages 16 and 26.)

[78] A. Gavranis and G. Kozanidis. Mixed integer biobjective quadratic programming for maximum-value minimum-variability fleet availability of a unit of mission aircraft. *Computers and Industrial Engineering*, 110:13–29, aug 2017. ISSN 03608352. doi: 10.1016/j.cie.2017.05.010. URL https://linkinghub.elsevier.com/retrieve/pii/S0360835217302140. (Cited on pages 16, 20, 21, and 26.)

[79] H. Gavranović and M. Buljubašsić. A hybrid approach combining local search and constraint programming for a large scale energy management problem. *RAIRO Recherche Operationnelle*, 47(4):481–500, 2013. ISSN 12903868. doi: 10.1051/ro/2013053. URL www.rairo-ro.org. (Cited on page 30.)

[80] C. A. Glass and R. A. Knight. The nurse rostering problem: A critical appraisal of the problem structure. *European Journal of Operational Research*, 202(2):379–389, apr 2010. ISSN 03772217. doi: 10.1016/j.ejor.2009.05.046. (Cited on page 27.)

[81] R. Gopalan and K. T. Talluri. The aircraft maintenance routing problem. *Operations Research*, 46(2):260–271, 1998. ISSN 0030364X. doi: 10.1287/opre.46.2.260. (Cited on page 13.)

[82] M. Grönkvist. Accelerating column generation for aircraft scheduling using constraint propagation. *Computers and Operations Research*, 33(10):2918–2934, oct 2006. ISSN 03050548. doi: 10.1016/j.cor.2005.01.017. (Cited on page 28.)

[83] M. Grönkvist. The Tail Assignment problem. Technical report, 2011. (Cited on pages xix, 13, and 28.)

[84] Z. Gu, G. L. Nemhauser, and M. W. Savelsbergh. Lifted cover inequalities for 0-1 integer programs: Computation. *INFORMS Journal on Computing*, 10(4):427–437, 1998. ISSN 10919856. doi: 10.1287/ijoc.10.4.427. (Cited on page 25.)

[85] R. A. Hahn and A. M. Newman. Scheduling United States Coast Guard helicopter deployment and maintenance at Clearwater Air Station, Florida. *Computers and Operations Research*, 35(6):1829–1843, jun 2008. ISSN 03050548. doi: 10.1016/j.cor.2006.09.015. URL `https://www.sciencedirect.com/science/article/pii/S0305054806002346`. (Cited on pages xviii, 15, 17, 20, 21, and 26.)

[86] C. A. Hane, C. Barnhart, E. L. Johnson, R. E. Marsten, G. L. Nemhauser, and G. Sigismondi. The fleet assignment problem: Solving a large-scale integer program. *Mathematical Programming*, 70(1-3):211–232, oct 1995. ISSN 00255610. doi: 10.1007/BF01585938. (Cited on page 26.)

[87] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3):449–467, 2001. (Cited on page 30.)

[88] F. He and R. Qu. A constraint programming based column generation approach to nurse rostering problems. *Computers & Operations Research*, 39(12):3331–3343, 2012. (Cited on page 31.)

[89] Headquarters. Army Aviation Maintenance. Technical report, Department of the Army, 2017. URL `https://rdl.train.army.mil/catalog-ws/view/100.ATSC/574C586C-A989-425A-9F3C-C92C693D923F-1505223206762/atp3{_}04x7.pdf`. (Cited on pages 7 and 16.)

[90] A. Ikegami and A. Niwa. A subproblem-centric model and approach to the nurse scheduling problem. *Mathematical Programming, Series B*, 97(3):517–541, aug 2003. ISSN 00255610. doi: 10.1007/s10107-003-0426-2. (Cited on page 27.)

[91] N. Jafari and S. Hessameddin Zegordi. Simultaneous recovery model for aircraft and passengers. In *Journal of the Franklin Institute*, volume 348, pages 1638–1655. Pergamon, sep 2011. doi: 10.1016/j.jfranklin.2010.03.012. (Cited on pages 15 and 27.)

[92] V. Jost and D. Savourey. A 0-1 integer linear programming approach to schedule outages of nuclear power plants. *Journal of Scheduling*, 16(6):551–566, 2013. ISSN 10946136. doi: 10.1007/s10951-013-0322-4. (Cited on pages 27 and 105.)

[93] J. M. J. M. Kessler. United States Air Force fighter jet maintenance Models : effectiveness of index policies. 2013. URL `https://dspace.mit.edu/handle/1721.1/82873`. (Cited on page 18.)

[94] O. Khaled, M. Minoux, V. Mousseau, S. Michel, and X. Ceugniet. A multi-criteria repair/recovery framework for the tail assignment problem in airlines. *Journal of Air Transport Management*, 68:137–151, may 2018. ISSN 09696997. doi: 10.1016/j.jairtraman.2017.10.002. (Cited on pages 15 and 27.)

[95] O. Khaled, M. Minoux, V. Mousseau, S. Michel, and X. Ceugniet. A compact optimization model for the tail assignment problem. *European Journal of Operational Research*, 264(2):548–557, jan 2018. ISSN 03772217. doi: 10.1016/j.ejor.2017.06.045. (Cited on pages xviii, xix, 13, 14, and 26.)

[96] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, may 1983. ISSN 00368075. doi: 10.1126/science.220.4598.671. URL `https://science.sciencemag.org/content/220/4598/671https://science.sciencemag.org/content/220/4598/671.abstract`. (Cited on page 29.)

[97] N. Kohl, A. Larsen, J. Larsen, A. Ross, and S. Tiourine. Airline disruption management-Perspectives, experiences and outlook. *Journal of Air Transport Management*, 13(3):149–162, may 2007. ISSN 09696997. doi: 10.1016/j.jairtraman.2007.01.001. (Cited on pages 15 and 27.)

[98] G. Kozanidis. A multiobjective model for maximizing fleet availability under the presence of flight and maintenance requirements. *Journal of Advanced Transportation*, 43(2):155–182, 2009. ISSN 01976729. doi: 10.1002/atr.5670430205. (Cited on pages xviii, 15, 17, 20, 21, and 26.)

[99] G. Kozanidis, A. Gavranis, and E. Kostarelou. Mixed integer least squares optimization for flight and maintenance planning of mission aircraft. *Naval Research Logistics*, 59(3-4):212–229, apr 2012. ISSN 0894069X. doi: 10.1002/nav.21483. URL `http://doi.wiley.com/10.1002/nav.21483`. (Cited on pages 16 and 26.)

[100] G. Kozanidis, A. Gavranis, and G. Liberopoulos. Heuristics for flight and maintenance planning of mission aircraft. *Annals of Operations Research*, 221(1):211–238, oct 2014. ISSN 15729338. doi: 10.1007/s10479-013-1376-6. URL `http://link.springer.com/10.1007/s10479-013-1376-6`. (Cited on pages 16 and 26.)

[101] M. Krishnamoorthy and A. T. Ernst. The Personnel Task Scheduling Problem. pages 343–368. Springer, Boston, MA, 2001. doi: 10.1007/978-1-4757-3333-4_20. URL `http://link.springer.com/10.1007/978-1-4757-3333-4{_}20`. (Cited on page 38.)

[102] Y.-C. Lai, D.-C. Fan, and K.-L. Huang. Optimizing rolling stock assignment and maintenance plan for passenger railway operations. *Computers & Industrial Engineering*, 85:284–295, jul 2015. ISSN 0360-8352. doi: 10.1016/J.CIE.2015.03.016. URL `https://www.sciencedirect.com/science/article/pii/S0360835215001254`. (Cited on pages xviii, 23, and 27.)

[103] G. Laporte and G. Pesant. A general multi-shift scheduling system. *Journal of the Operational Research Society*, 55(11):1208–1217, 2004. (Cited on page 28.)

[104] E. Larsen, S. Lachapelle, Y. Bengio, E. Frejinger, S. Lacoste-Julien, and A. Lodi. Predicting Tactical Solutions to Operational Planning Problems under Imperfect Information. jul 2018. URL `http://arxiv.org/abs/1807.11876`. (Cited on pages xix, 31, 32, and 72.)

[105] H. C. Lau. On the complexity of manpower shift scheduling. *Computers and Operations Research*, 23(1):93–102, jan 1996. ISSN 03050548. doi: 10.1016/0305-0548(94)00094-O. (Cited on page 22.)

[106] J. Lazić, S. Hanafi, N. Mladenović, and D. Urošević. Variable neighbourhood decomposition search for 0-1 mixed integer programs. *Computers and Operations Research*, 37(6):1055–1067, jun 2010. ISSN 03050548. doi: 10.1016/j.cor.2009.09.010. (Cited on page 26.)

[107] H. Li and K. Womer. A decomposition approach for shipboard manpower scheduling. *Military Operations Research*, pages 67–90, 2009. (Cited on page 30.)

[108] Z. Li, J. Guo, and R. Zhou. Maintenance scheduling optimization based on reliability and prognostics information. In *Proceedings - Annual Reliability and Maintainability Symposium*, volume 2016-April. Institute of Electrical and Electronics Engineers Inc., apr 2016. ISBN 9781509002481. doi: 10.1109/RAMS.2016.7448069. (Cited on pages xviii, 17, 20, 21, and 31.)

[109] B. Lin, J. Wu, R. Lin, J. Wang, H. Wang, and X. Zhang. Optimization of high-level preventive maintenance scheduling for high-speed trains. *Reliability Engineering and System Safety*, 183:261–275, mar 2019. ISSN 09518320. doi: 10.1016/j.ress.2018.11.028. (Cited on pages xviii, 23, and 29.)

[110] Z. Liu, Z. Liu, Z. Zhu, Y. Shen, and J. Dong. Simulated annealing for a multi-level nurse rostering problem in hemodialysis service. *Applied Soft Computing Journal*, 64:148–160, mar 2018. ISSN 15684946. doi: 10.1016/j.asoc.2017.12.005. (Cited on page 29.)

[111] A. Lodi, L. Mossina, and E. Rachelson. Learning to Handle Parameter Perturbations in Combinatorial Optimization: an Application to Facility Location. jul 2019. URL `http://arxiv.org/abs/1907.05765`. (Cited on pages xix, 31, 32, and 72.)

[112] R. Lopes, V. W. Morais, T. F. Noronha, and V. A. Souza. Heuristics and matheuristics for a real-life machine reassignment problem. *International Transactions in Operational Research*, 22(1):77–95, jan 2015. ISSN 14753995. doi: 10.1111/itor.12063. URL `http://doi.wiley.com/10.1111/itor.12063`. (Cited on page 30.)

[113] S. Louet. Parly s'empare du chantier de la maintenance militaire. *Reuters*, 2017. URL `https://www.challenges.fr/top-news/parly-s-empare-du-chantier-de-la-maintenance-militaire_519362`. (Cited on pages xvi, 8, and 9.)

[114] R. Lusby, L. F. Muller, and B. Petersen. A solution approach based on Benders decomposition for the preventive maintenance scheduling problem of a stochastic large-scale energy system. *Journal of Scheduling*, 16(6):605–628, dec 2013. ISSN 10946136. doi: 10.1007/s10951-012-0310-0. (Cited on page 27.)

[115] B. Maenhout and M. Vanhoucke. Branching strategies in a branch-and-price approach for a multiple objective nurse scheduling problem. *Journal of Scheduling*, 13(1):77–93, feb 2010. ISSN 10946136. doi: 10.1007/s10951-009-0108-x. (Cited on page 27.)

[116] D. O. Marlow and R. F. Dell. Optimal short-term military aircraft fleet planning. *Journal of Applied Operational Research*, 9(1):38–53, 2017. ISSN 1927-0089. URL `www.orlabanalytics.ca`. (Cited on pages xviii, 8, 16, 17, 20, 21, 26, 27, and 31.)

[117] G. Maróti and L. Kroon. Maintenance routing for train units: The transition model. *Transportation Science*, 39(4):518–525, 2005. ISSN 15265447. doi: 10.1287/trsc.1050.0116. (Cited on pages 24 and 27.)

[118] K. Marriott, P. J. Stuckey, and P. J. Stuckey. *Programming with constraints: an introduction*. MIT press, 1998. (Cited on page 28.)

[119] V. Mattila, K. Virtanen, and T. Raivio. Improving maintenance decision making in the Finnish Air Force through simulation. *Interfaces*, 38(3):187–201, jun 2008. ISSN 00922102. doi: 10.1287/inte.1080.0349. URL `http://pubsonline.informs.org/doi/abs/10.1287/inte.1080.0349`. (Cited on page 18.)

[120] J.-P. Métivier, P. Boizumault, and S. Loudni. Solving nurse rostering problems using soft global constraints. In *International Conference on Principles and Practice of Constraint Programming*, pages 73–87. Springer, 2009. (Cited on page 28.)

[121] L. Mira, A. R. Andrade, and M. C. Gomes. Maintenance scheduling within rolling stock planning in railway operations under uncertain maintenance durations. *Journal of Rail Transport Planning and Management*, 14:100177, jan 2020. ISSN 22109706. doi: 10.1016/j.jrtpm.2020.100177. (Cited on pages 24 and 27.)

[122] S. Mitchell, M. OSullivan, and I. Dunning. PuLP: a linear programming toolkit for python. *The University of Auckland, Auckland, New Zealand, http://www.optimization-online. org/DB_FILE/2011/09/3178. pdf*, 2011. (Cited on page 103.)

[123] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24(11):1097–1100, nov 1997. ISSN 03050548. doi: 10.1016/S0305-0548(97)00031-2. (Cited on pages 29 and 90.)

[124] H. Murat Afsar, M. L. Espinouse, and B. Penz. A two-step heuristic to build flight and maintenance planning in a rolling-horizon. In *Proceedings - ICSSSM'06: 2006 International Conference on Service Systems and Service Management*, volume 2, pages 1251–1256. IEEE Computer Society, 2006. ISBN 1424404517. doi: 10.1109/ICSSSM.2006.320688. (Cited on pages xviii, xix, 13, 14, 27, and 29.)

[125] J. M. Newcamp, W. J. Verhagen, and R. Curran. Validation of the SmartBasing aircraft rotation and retirement strategy. *CEAS Aeronautical Journal*, 10(3):875–883, sep 2019. ISSN 18695590. doi: 10.1007/s13272-018-00356-z. URL `https://doi.org/10.1007/s13272-018-00356-z`. (Cited on page 18.)

[126] K. Nonobe. INRC2010: An approach using a general constraint optimization solver. *The First International Nurse Rostering Competition (INRC 2010)*, 2010. (Cited on page 28.)

[127] T. Osogami and H. Imai. Classification of various neighborhood operations for the nurse scheduling problem. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1969, pages 72–83. Springer Verlag, dec 2000. ISBN 3540412557. doi: 10.1007/3-540-40996-3_7. (Cited on page 29.)

[128] T. P. Peixoto. The graph-tool python library. *figshare*, 2014. doi: 10.6084/m9.figshare.1164194. URL `http://figshare.com/articles/graph{_}tool/1164194`. (Cited on page 103.)

[129] F. Peschiera, O. Battaïa, A. Haït, and N. Dupin. Bi-objective mip formulation for the optimization of maintenance planning on french military aircraft operations. 2018. URL `http://oatao.univ-toulouse.fr/20766/`. (Not cited.)

[130] F. Peschiera, A. Haït, N. Dupin, and O. Battaïa. A novel mip formulation for the optimization problem of maintenance planning of military aircraft. In *XIX Latin-Iberoamerican Conference on Operations Research*, pages 1–2, Lima, PE, 2018. (Not cited.)

[131] F. Peschiera, A. Haït, N. Dupin, and O. Battaïa. Maintenance planning on french military aircraft operations. In *Congrès annuel de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)*, pages 1–2, Lorient, FR, 2018. URL `http://oatao.univ-toulouse.fr/20036/`. (Not cited.)

[132] F. Peschiera, R. Dell, J. Royset, A. Haït, N. Dupin, and O. Battaïa. A novel solution approach with ML-based pseudo-cuts for the Flight and Maintenance Planning problem. *OR Spectrum*, pages 1–30, jun 2020. ISSN 0171-6468. doi: 10.1007/s00291-020-00591-z. URL `http://link.springer.com/10.1007/s00291-020-00591-z`. (Not cited.)

[133] F. Peschiera, N. Dupin, O. Battaïa, and A. Haït. An alternative mip formulation for the military flight and maintenance planning problem. In *Congrès annuel de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)*, pages 1–2, Montpellier, FR, 2020. URL `https://oatao.univ-toulouse.fr/26033/`. (Not cited.)

[134] F. Peschiera, A. Haït, N. Dupin, and O. Battaïa. Long term planning of military aircraft flight and maintenance operations. Technical report, ISAE-SUPAERO, Université de Toulouse, France, 2020. URL `https://arxiv.org/abs/2001.09856`. (Not cited.)

[135] F. Peschiera, N. Dupin, A. Haït, and O. Battaïa. Novel graph-based matheuristic to solve the flight and maintenance planning problem. Forthcoming. (Not cited.)

[136] B. W. Pippin. Allocating flight hours to army helicopters. Technical report, NAVAL POSTGRADUATE SCHOOL MONTEREY CA, 1998. (Cited on pages xviii, 15, and 17.)

[137] M. Porcheron, A. Gorge, O. Juan, T. Simovic, G. D. E. R&D, U. Clamart, U. France, and U. 2010. Challenge ROADEF/EURO 2010: A large-scale energy management problem with varied constraints. Technical report. (Cited on page 24.)

[138] C. Prud'homme, J.-G. Fages, and X. Lorca. *Choco Documentation*. TASC - LS2N CNRS UMR 6241, COSLING S.A.S., 2017. URL `http://www.choco-solver.org`. (Cited on pages xviii and 28.)

[139] R. Qu and F. He. A Hybrid Constraint Programming Approach for Nurse Rostering Problems. In *Applications and Innovations in Intelligent Systems XVI*, pages 211–224. Springer London, 2009. doi: 10.1007/978-1-84882-215-3_16. (Cited on page 30.)

[140] E. Rahimian, K. Akartunalı, and J. Levine. A hybrid Integer Programming and Variable Neighbourhood Search algorithm to solve Nurse Rostering Problems. *European Journal of Operational Research*, 258(2):411–423, apr 2017. ISSN 03772217. doi: 10.1016/j.ejor.2016.09.030. (Cited on page 30.)

[141] E. Rahimian, K. Akartunalı, and J. Levine. A hybrid integer and constraint programming approach to solve nurse rostering problems. *Computers and Operations Research*, 82:83–94, jun 2017. ISSN 03050548. doi: 10.1016/j.cor.2017.01.016. URL `https://www.sciencedirect.com/science/article/pii/S0305054817300163`. (Cited on page 30.)

[142] G. R. Raidl. Decomposition based hybrid metaheuristics. In *European Journal of Operational Research*, volume 244, pages 66–76. Elsevier, jul 2015. doi: 10.1016/j.ejor.2014.12.005. (Cited on page 30.)

[143] R. T. Rockafellar and J. O. Royset. On buffered failure probability in design and optimization of structures. *Reliability Engineering and System Safety*, 95(5):499–510, may 2010. ISSN 09518320. doi: 10.1016/j.ress.2010.01.001. URL `https://www.sciencedirect.com/science/article/pii/S0951832010000177`. (Cited on page 32.)

[144] R. T. Rockafellar and J. O. Royset. Measures of residual risk with connections to regression, risk tracking, surrogate models, and ambiguity. *SIAM Journal on Optimization*, 25(2):1179–1208, jan 2015. ISSN 10526234. doi: 10.1137/151003271. URL `http://epubs.siam.org/doi/10.1137/151003271`. (Cited on page 32.)

[145] R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *The Journal of Risk*, 2(3):21–41, 2000. ISSN 14651211. doi: 10.21314/jor.2000.038. URL `https://pdfs.semanticscholar.org/0df3/ccfb652189488337202933d4151fc20ac31d.pdf`. (Cited on page 32.)

[146] V. Rosenzweig Vojvodić and A. Domitrović. Planning of Training Aircraft Flight Hours. *Croatian Operational Research Review*, 1(1):170, dec 2011. ISSN 1848-0225. URL `https://hrcak.srce.hr/index.php?show=clanak{&}id{_}clanak{_}jezik=139831`. (Cited on pages xviii, 17, and 26.)

[147] A. Rozenknop, R. Wolfler Calvo, L. Alfandari, D. Chemla, and L. Létocart. Solving the electricity production planning problem by a column generation based heuristic. *Journal of Scheduling*, 16(6):585–604, 2013. ISSN 10946136. doi: 10.1007/s10951-012-0286-9. (Cited on pages 30 and 105.)

[148] D. T. Sanchez, B. Boyacı, and K. G. Zografos. An optimisation framework for airline fleet maintenance scheduling with tail assignment considerations. *Transportation Research Part B: Methodological*, 133:142–164, mar 2020. ISSN 01912615. doi: 10.1016/j.trb.2019.12.008. (Cited on pages xviii, xix, 13, 14, 15, and 26.)

[149] H. G. Santos, T. A. Toffolo, R. A. Gomes, and S. Ribas. Integer programming techniques for the nurse rostering problem. *Annals of Operations Research*, 239(1):225–251, apr 2016. ISSN 15729338. doi: 10.1007/s10479-014-1594-6. (Cited on page 27.)

[150] A. Sarac, R. Batta, and C. M. Rump. A branch-and-price approach for operational aircraft maintenance routing. *European Journal of Operational Research*, 175(3):1850–1869, dec 2006. ISSN 03772217. doi: 10.1016/j.ejor.2004.10.033. (Cited on pages xviii, xix, 13, 14, and 26.)

[151] M. W. P. Savelsbergh. Preprocessing and Probing Techniques for Mixed Integer Programming Problems. *ORSA Journal on Computing*, 6(4):445–454, nov 1994. ISSN 0899-1499. doi: 10.1287/ijoc.6.4.445. (Cited on pages 25 and 26.)

[152] J. Seif and A. J. Yu. An extensive operations and maintenance planning problem with an efficient solution method. *Computers and Operations Research*, 95:151–162, jul 2018. ISSN 03050548. doi: 10.1016/j.cor.2018.03.010. URL `https://www.sciencedirect.com/science/article/pii/S0305054818300790`. (Cited on pages xviii, 15, 16, 17, 20, 21, and 26.)

[153] A. Sgaslik. Planning German Army Helicopter Maintenance and Mission Assignment. 1994. ISSN 1098-6596. doi: 10.1017/CBO9781107415324.004. (Cited on pages xviii, 15, and 26.)

[154] A. I. J. Shah, N. M. Yusoff, and N. M. Noor. Optimization of Sukhoi Su-30MKM maintenance planning for maximum operational readiness. In *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, volume 2017-Decem, pages 2500–2503. IEEE, nov 2017. ISBN 9781509011339. doi: 10.1109/TENCON.2017.8228282. URL `http://ieeexplore.ieee.org/document/8228282/`. (Cited on pages xviii, 13, 17, 20, 21, and 31.)

[155] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1520, pages 417–431. Springer Verlag, oct 1998. ISBN 3540652248. doi: 10.1007/3-540-49481-2_30. (Cited on page 30.)

[156] P. Smet. *Nurse rostering: models and algorithms for theory, practice and integration with other problems.* PhD thesis, KU Leuven, jul 2016. URL `https://lirias.kuleuven.be/handle/123456789/496913?mode=full{&}submit{_}simple=Show+full+item+record`. (Cited on pages xviii, 22, 23, and 38.)

[157] P. Smet and G. V. Berghe. *A matheuristic approach to the shift minimisation personnel task scheduling problem.* 2012. ISBN 9788214052985. URL

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.308.4488{&}rep=rep1{&}type=pdf{#}page=146. (Cited on page 30.)

[158] M. Sniedovich and S. Voß. The corridor method: a dynamic programming inspired metaheuristic. *Control and Cybernetics*, 35:551–578, 2006. (Cited on page 26.)

[159] K. Sörensen. Metaheuristics-the metaphor exposed. *International Transactions in Operational Research*, 22(1):3–18, 2015. ISSN 14753995. doi: 10.1111/itor.12001. (Cited on page 28.)

[160] R. Soto, B. Crawford, R. Bertrand, and E. Monfroy. Modeling NRPs with Soft and Reified Constraints. *AASRI Procedia*, 4:202–205, jan 2013. ISSN 22126716. doi: 10.1016/j.aasri.2013.10.031. (Cited on page 28.)

[161] C. Sriram and A. Haghani. An optimization model for aircraft maintenance scheduling and re-assignment. *Transportation Research Part A: Policy and Practice*, 37(1):29–48, jan 2003. ISSN 09658564. doi: 10.1016/S0965-8564(02)00004-6. (Cited on pages xviii, 7, and 13.)

[162] M. Stølevik, T. E. Nordlander, A. Riise, and H. Frøyseth. A hybrid approach for solving real-world nurse rostering problems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6876 LNCS, pages 85–99. Springer, Berlin, Heidelberg, 2011. ISBN 9783642237850. doi: 10.1007/978-3-642-23786-7_9. URL http://www.sintef.no. (Cited on page 29.)

[163] P. Strandmark, Y. Qu, and T. Curtois. First-order linear programming in a column generation-based heuristic approach to the nurse rostering problem. *Computers and Operations Research*, 120:104945, aug 2020. ISSN 03050548. doi: 10.1016/j.cor.2020.104945. (Cited on page 27.)

[164] E. G. Talbi. *Metaheuristics: From Design to Implementation*, volume 74. John Wiley & Sons, 2009. ISBN 9780470278581. doi: 10.1002/9780470496916. (Cited on page 28.)

[165] E. G. Talbi. Combining metaheuristics with mathematical programming, constraint programming and machine learning. *Annals of Operations Research*, 240(1):171–215, may 2016. ISSN 15729338. doi: 10.1007/s10479-015-2034-y. (Cited on pages 29, 30, and 31.)

[166] I. X. Tassopoulos, I. P. Solos, and G. N. Beligiannis. A two-phase adaptive variable neighborhood approach for nurse rostering. *Computers and Operations Research*, 60:150–169, aug 2015. ISSN 03050548. doi: 10.1016/j.cor.2015.02.009. (Cited on page 29.)

[167] Transparency International. Global Corruption Report 2009: Corruption and the Private Sector, 2009. URL https://www.transparency.org/en/publications/global-corruption-report-2009. (Cited on page 3.)

[168] S. Tréfond, A. Billionnet, S. Elloumi, H. Djellab, and O. Guyon. Optimization and simulation for robust railway rolling-stock planning. *Journal of Rail Transport Planning*

and Management, 7(1-2):33–49, jun 2017. ISSN 22109706. doi: 10.1016/j.jrtpm.2017.02.001. (Cited on pages 24 and 27.)

[169] L. Trilling, A. Guinet, and D. Le Magny. Nurse scheduling using integer linear programming and constraint programming. *IFAC Proceedings Volumes*, 39(3):671–676, 2006. (Cited on page 28.)

[170] A. M. Turhan and B. Bilgen. A hybrid fix-and-optimize and simulated annealing approaches for nurse rostering problem. *Computers and Industrial Engineering*, 145: 106531, may 2020. ISSN 03608352. doi: 10.1016/j.cie.2020.106531. (Cited on pages 22 and 30.)

[171] R. Tyrrell Rockafellar and J. O. Royset. Engineering Decisions under Risk Averseness. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, 1(2):04015003, jun 2015. ISSN 23767642. doi: 10.1061/AJRUA6.0000816. URL `http://ascelibrary.org/doi/10.1061/AJRUA6.0000816`. (Cited on page 32.)

[172] J. Van Den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, and L. De Boeck. Personnel scheduling: A literature review. *European Journal of Operational Research*, 226(3):367–385, may 2013. ISSN 03772217. doi: 10.1016/j.ejor.2012.11.029. (Cited on page 22.)

[173] M. Vanhoucke and B. Maenhout. On the characterization and generation of nurse scheduling problem instances. *European Journal of Operational Research*, 196(2):457–467, jul 2009. ISSN 03772217. doi: 10.1016/j.ejor.2008.03.044. (Cited on page 22.)

[174] M. Verhoeff, W. J. Verhagen, and R. Curran. Maximizing operational readiness in military aviation by optimizing flight and maintenance planning. *Transportation Research Procedia*, 10(July):941–950, 2015. ISSN 23521465. doi: 10.1016/j.trpro.2015.09.048. URL `http://dx.doi.org/10.1016/j.trpro.2015.09.048`. (Cited on pages xviii, 15, 16, 17, 20, 21, and 26.)

[175] H. Vermuyten, J. Namorado Rosa, I. Marques, J. Beliën, and A. Barbosa-Póvoa. Integrated staff scheduling at a medical emergency service: An optimisation approach. *Expert Systems with Applications*, 112:62–76, dec 2018. ISSN 09574174. doi: 10.1016/j.eswa.2018.06.017. (Cited on pages 29 and 30.)

[176] C. P. Vielma and J. Pablo. Mixed Integer Linear Programming Formulation Techniques. *SIAM Review*, 57(1):3–57, 2015. doi: 10.1137/130915303. URL `http://dx.doi.org/10.1137/130915303http://hdl.handle.net/1721.1/96480`. (Cited on page 26.)

[177] L. Winata. *Heuristic approaches for flight and maintenance planning of large fleets.* PhD thesis, 2011. (Cited on pages 17, 20, 21, and 29.)

[178] C.-L. Wu and S. J. Maher. Airline scheduling and disruption management. *Air Transport Management*, pages 179–195, 2020. doi: 10.4324/9780429299445-12. (Cited on pages 15 and 27.)

[179] A. S. Xavier, F. Qiu, and S. Ahmed. Learning to Solve Large-Scale Security-Constrained Unit Commitment Problems. feb 2019. URL http://arxiv.org/abs/1902.01697. (Cited on pages xix and 31.)

[180] S. Yildiz and J. P. Vielma. Incremental and encoding formulations for Mixed Integer Programming. *Operations Research Letters*, 41(6):654–658, 2013. ISSN 01676377. doi: 10.1016/j.orl.2013.09.004. URL https://www.sciencedirect.com/science/article/pii/S0167637713001296?casa{_}token=nZSRz2kE9-AAAAAA:dPB6w7T8iWEbanyU4NxW{_}IzrRKfAgTDdfzrPolJwyczPuc2sPWBaBMA6sYzcWUuAjL4EcX8WhJs. (Cited on page 26.)

[181] T. H. Yunes, A. V. Moura, and C. C. De Souza. Hybrid column generation approaches for urban transit crew management problems. *Transportation Science*, 39(2):273–288, 2005. (Cited on page 31.)

Résumé — Cette thèse étudie le problème de planification de vol et de la maintenance des avions militaires. D'abord, nous étudions la complexité de ce problème d'optimisation. Puis, nous proposons un modèle de programmation linéaire en nombres entiers (PLNE) pour le résoudre. Nous construisons un générateur d'instances et une heuristique pour générer des solutions initiales. Ensuite, nous appliquons l'Apprentissage Automatique pour améliorer la performance des modèles PLNE en utilisant des coupes valides générées à partir des conditions initiales et des coupes apprises à partir de la prédiction des caractéristiques de solutions optimales. Ces coupes sont appliquées à un nouveau modèle PLNE. Le résultat est une réduction du temps de résolution avec peu de pertes d'optimalité et de faisabilité par rapport aux méthodes matheuristiques alternatives. Finalement, nous présentons une nouvelle matheuristique pour résoudre efficacement des grandes instances. La méthode utilise une descente à voisinage variable qui combine la programmation dynamique (DP) et l'horizon glissant. La DP exploite une représentation en graphe de l'espace des solutions de chaque avion. Le résultat est des solutions rapides et presque optimales, et un passage à l'échelle efficace pour des instances de très grande taille.

Mots clés : maintenance d'aéronefs, planification de vol et maintenance, programmation linéaire aux nombres entiers, matheuristiques, apprentissage automatique, algorithmes hybrides

Abstract — This thesis studies the long term Military Flight and Maintenance Planning problem. First, we evaluate the complexity of this optimisation problem. Then we propose a Mixed Integer Programming (MIP) model to solve it. We develop an instance generator and a heuristic to generate initial solutions. Furthermore, we apply Machine Learning to improve the performance of the MIP model by using valid cuts generated on the basis of initial conditions and learned cuts based on the prediction of characteristics of optimal solutions. These cuts are applied to a new MIP model. This results in reductions in the solution time with little losses in optimality and feasibility in comparison to alternative matheuristic methods. Finally, we present a new matheuristic to efficiently solve large instances. The method employs a Variable Neighborhood Descent that combines Dynamic Programming (DP) and Rolling Horizon neighborhoods. The DP is applied to a graph representation of the solution space for a single aircraft. This results in fast good quality solutions and an efficient scaling for very large instances.

Keywords: aircraft maintenance, flight and maintenance planning, mixed integer programming, matheuristics, machine learning, hybrid algorithms

Département d'Ingénierie des Systèmes Complexes, ISAE-SUPAERO, 10, avenue Édouard Belin
Toulouse