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 :
Sana IKLI

le lundi 12 juillet 2021

Titre :

Méthodes exactes et heuristiques pour l'ordonnancement des atterrissages d'avions

---

École doctorale et discipline ou spécialité :
ED AA : Mathématiques et Applications

Unité de recherche :
ENAC LAB Laboratoire de Recherche ENAC

Directeur(s) de Thèse :

M. Marcel MONGEAU (directeur de thèse)
M. Emmanuel RACHELSON (co-directeur de thèse)

Jury :
M. Paul ARMAND Professeur Université de Limoges - Président
Mme Luce BROTCORNE Directrice de recherche INRIA Lille - Rapporteure
M. Alain FAYE Maître de conférences ENSIIE - Rapporteur
M. Nicolas JOUANDEAU Maître de conférences Université Paris 8 - Examinateur
Mme Catherine MANCEL Maître de conférences ENAC - Examinatrice
M. Marcel MONGEAU Professeur ENAC - Directeur de thèse
M. Xavier OLIVE Ingénieur de recherche ONERA - Examinateur
M. Emmanuel RACHELSON Professeur ISAE-SUPAERO - Co-directeur de thèse

Remerciements

Je tiens tout d'abord à remercier les personnes qui m'ont aidé à mener à bien cette thèse, qui fut une expérience enrichissante et passionnante. Mes tous premiers remerciements vont à mes parents et à ma sœur. Cette thèse n'aurait pas vu le jour sans leur soutien.

J'exprime ma gratitude aux membres du jury de ma thèse qui ont accepté de participer à ma soutenance. Je remercie madame Luce Brotcorne et monsieur Alain Faye pour avoir accepté d'être rapporteurs pour ma thèse. Je remercie également messieurs les examinateurs Paul Armand et Nicolas Jouandeau qui m'ont fait l'honneur d'assister à ma présentation. Je ne remercierai jamais assez monsieur Paul Armand pour m'avoir encouragé pour venir à l'ENAC, et pour le travail remarquable qu'il fait au master ACSYON de Limoges.

J'adresse une pensée particulière à mes collègues et amis de l'ENAC, qui ont contribué à la réussite de cette thèse et à rendre cette expérience agréable. Tout d'abord Sana (1) pour nos longues discussions passionnantes, Roni pour la " pause marche " de 16h et ma compatriote Hasna pour nos longues discussions au téléphone vers la fin de ma thèse. Je pense aussi à mes amis de Chine : Ying, MaJi, Xiao, Shangrong, Bug, Ziqing, mes collègues au bureau Gaby, Philippe, Thinh, Sylvain, sans oublier mes collègues de l'ISAE-SUPAERO : Zoé, Luca, Erwan, Valentin et Franco. Merci à vous tous !

Je tiens à remercier Serge Roux, notre "sauveur", qui a toujours des solutions à nos problèmes informatiques, et qui fait un travail remarquable dans le SI de l'ENAC. J'apprécie énormément sa bonne humeur et son humour qui rend les pauses déjeuner agréable. Je pense également à Hélène Weiss, assistante à la recherche à l'ENAC, qui nous aide énormément dans les procédures administratives.

Mes derniers remerciements, et non des moindres, vont à mes amis qui travaillent à l'accueil des résidences ENAC où je réside. Je pense à Antonella, Jiby et Naissa qui étaient un réel soutien psychologique pendant toute la durée de la thèse, et surtout pendant les périodes de confinement. Merci à vous et je vous souhaite le meilleur !

Table des matières

Table des figures

Liste des tableaux

Table des sigles, acronymes et abréviations

ALP	Aircraft Landing Problem - Problème d'ordonnancement d'atterrissages
ASP	Aircraft Sequencing Problem - Problème d'ordonnancement d'avions
ATP	Aircraft Take-off Problem - Problème d'ordonnancement de décollages
CPS	Constrained Position Shifting - Changement du position contraint
DP	Dynamic Programming - Programmation dynamique
FCFS	First Come First Serve - Premier arrivé premier servi
IAF	Initial Approach Fix – Point repère d'approche initiale
kt	Knot - nœud, mesure de vitesse (1 kt = 1 nm/h)
PLNE	Programmation Linéaire mixte en Nombres Entiers
ML	Machine Learning - Apprentissage automatique
NM	Nautical Mile - mile nautique, mesure de distance (1 NM = 1,852 km)
OACI	Organisation de l'Aviation Civile Internationale
TMA	Terminal Manoeuvring Area – Zone de contrôle terminale

Introduction générale

Les pistes des aéroports sont reconnues comme l'un des principaux goulots d'étranglement du système de gestion du trafic aérien et l'un des facteurs qui déterminent la capacité des aéroports. Optimiser l'utilisation des infrastructures aéroportuaires, en particulier la piste, motive les nombreuses études sur les trois problèmes d'ordonnancement d'avions : ALP (*Aircraft Landing Problem*), ATP (*Aircraft Take-off Problem*) et ASP (*Aircraft Scheduling Problem*). De manière générale, l'ALP (respectivement l'ATP) consiste à affecter d'abord une piste disponible à chaque avion demandant l'atterrissage (respectivement le décollage), puis à allouer à chacun une date cible d'atterrissage (respectivement de décollage). Lorsque l'on considère l'ordonnancement simultané des atterrissages et des décollages, le problème est appelé *Aircraft Scheduling Problem*. Nous adoptons pour la suite de ce manuscrit la nomenclature « problème d'ordonnancement d'avions » pour désigner indifféremment l'ALP, l'ATP ou l'ASP. Nous préciserons quel problème est concerné quand cela sera pertinent.

Dans la formulation mathématique d'un problème d'ordonnancement d'avions, plusieurs contraintes opérationnelles doivent être prises en compte. Les contraintes les plus importantes comprennent la séparation réglementaire entre les paires d'aéronefs consécutifs afin de garantir la sécurité des vols et les fenêtres de temps, définies par une date au plus tôt et une date au plus tard d'opération sur la piste. Ces dates sont définies selon le type d'avion en question et la quantité de carburant à bord. Diverses fonctions-objectif peuvent être envisagées en fonction du point de vue de la partie prenante considérée : aéroport, compagnie aérienne ou contrôleur aérien. Les points de vue les plus fréquemment étudiés dans la littérature sont celui de l'aéroport et celui de la compagnie aérienne. Le premier cherche généralement à optimiser la capacité des pistes ; le deuxième vise à maximiser la ponctualité et à minimiser la consommation du carburant.

Dans cette thèse, nous nous intéressons au problème d'ordonnancement des atterrissages d'avions. Notre objectif est de minimiser les coûts de retard, ce qui correspond au point de vue des compagnies aériennes. Nous considérons une représentation du coût de retard réaliste, mais rarement considérée dans la littérature : une fonction convexe, linéaire par morceaux. Nous présentons d'abord une méthode exacte basée sur la programmation linéaire mixte en nombres entiers, puis une nouvelle méthode heuristique basée sur un algorithme de planification optimiste, issu du domaine de l'apprentissage par renforcement. Nous proposons également dans cette thèse des bases de données et des instances tests pour notre problème qui sont difficiles à résoudre et qui impliquent des coûts de retard réalistes.

Le manuscrit est divisé en six chapitres. Dans le premier chapitre, nous mettons en évidence le cadre opérationnel du problème d'ordonnancement d'avions en zone d'approche aéroportuaire. Nous commençons par une introduction à la gestion du trafic aérien, en nous focalisant sur les aspects aéronautiques liés au problème considéré. Nous donnons ensuite dans ce chapitre un aperçu des techniques utilisées par les contrôleurs aériens pour séquencer les avions. Finalement, nous présentons un tour d'horizon des outils d'aide à la décision mis à la disposition des contrôleurs aériens. Le deuxième chapitre est consacré à la présentation des formulations mathématiques du problème, avec les différentes contraintes opérationnelles impliquées et les objectifs considérés selon les acteurs concernés : aéroports, compagnies aé-

riennes et contrôleurs aériens. Plusieurs variantes du problème peuvent être considérées : atterrissages et/ou décollages, pistes uniquement, pistes et autres points *repères* de l'espace aérien aéroportuaire. Ces variantes sont aussi présentées dans ce chapitre. Le troisième chapitre présente la première contribution de la thèse. Il s'agit d'une revue complète des méthodes de résolution les plus pertinentes de la littérature pour le problème d'ordonnancement d'avions. Cette contribution a été publiée sous la forme d'un article intitulé *The aircraft runway scheduling problem : A survey*, publié dans la revue scientifique *Computers & Operations Research*. Dans le quatrième chapitre, nous proposons une méthode exacte pour modéliser et résoudre le problème d'ordonnancement d'atterrissages d'avions. Il s'agit d'un modèle de programmation mixte en nombres entiers. L'objectif est de minimiser le coût total de retard. Nous considérons dans notre modèle une fonction coût convexe et linéaire par morceaux. Nous présentons également dans ce chapitre notre deuxième contribution de la thèse : la proposition de nouvelles instances réalistes et difficiles pour l'ALP, construites à partir du trafic réel sur l'aéroport de Paris-Orly. Nos données et instances sont disponibles en ligne, à l'adresse : `http://data.recherche.enac.fr/ikli-alp/`. Le cinquième chapitre présente la troisième contribution de la thèse : il s'agit d'une nouvelle méthode heuristique pour résoudre le problème d'ordonnancement d'atterrissages d'avions. Nous nous intéressons au contexte particulier où l'on dispose de ressources de calcul finies (temps de calcul limité par exemple) pour trouver des solutions qui soient de bonne qualité. Nous proposons un nouveau modèle inspiré des processus de décision markoviens pour formuler le problème d'ordonnancement d'atterrissages d'avions. Ensuite, nous présentons notre approche heuristique pour résoudre un tel modèle et trouver des solutions en des temps de calcul très courts. Le sixième et dernier chapitre s'intéresse au problème d'ordonnancement d'arrivées d'avions au niveau de points critiques de l'espace aérien aéroportuaire ainsi qu'au seuil des pistes d'atterrissage. Dans ce chapitre, nous introduisons le contexte et la motivation de l'étude de ce problème. Ensuite, nous présentons un modèle d'optimisation mixte en nombres entiers pour modéliser et résoudre ce problème. Puis nous proposons une étude numérique préliminaire, dans laquelle nous comparons notre modèle mathématique à une technique traditionnelle utilisées par les contrôleurs aériens. Nous terminons le chapitre par des perspectives d'amélioration du modèle et de poursuite de travaux.

Chapitre 1

Introduction à la gestion du trafic aérien

Le problème d'ordonnancement d'avions en atterrissage relève du domaine plus large de la gestion du trafic aérien dans la zone d'approche entourant les aéroports. Dans ce chapitre, nous présentons d'abord le cadre général de la gestion de trafic aérien, en nous focalisant sur les aspects liés à notre problématique. Ensuite, nous donnons un aperçu des différentes contraintes opérationnelles liées à notre problématique, qui permettent de garantir la sécurité des vols, puis nous décrivons les principales techniques utilisées par les contrôleurs pour séquencer les avions. Enfin, nous présentons les outils d'aide à la décision mis à disposition des services de contrôle aérien pour optimiser les séquences d'avions.

Sommaire

1.1 La gestion du trafic aérien

La gestion du trafic aérien (*Air Traffic Management* – ATM) – dont relève notre problématique d'ordonnancement d'avions en atterrissage – est un terme aéronautique englobant tous les systèmes et activités disponibles pour guider un aéronef, depuis le moment où il quitte sa position de parking dans l'aéroport de départ, en passant par toutes les phases de vol (figure 1.1), jusqu'à l'arrivée à sa nouvelle position de parking dans l'aéroport de destination. L'objectif de tels systèmes est d'assurer la sécurité des vols et la fluidité du trafic aérien.

Un des défis majeurs des systèmes ATM est la gestion du trafic aérien dans la zone d'approche aéroportuaire appelée *Terminal Maneuvering Area* – TMA, car elles englobent

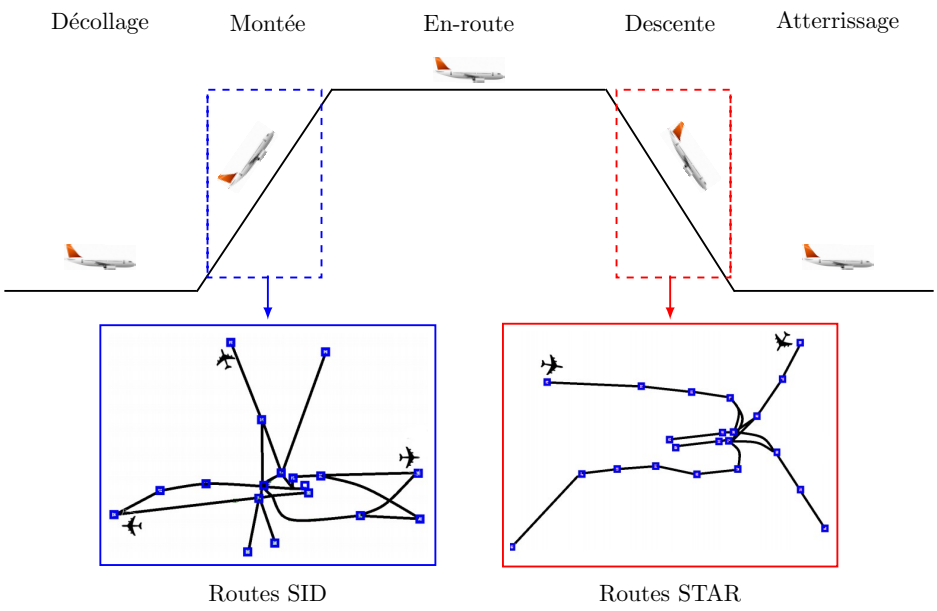


FIGURE 1.1 – Illustration des cinq principales phases de vol d'un aéronef (décollage, montée, en-route, descente et atterrissage), avec exemples de routes de départ SID et d'approche STAR de l'aéroport de Paris Charles De-Gaulle.
Source (pour SID et STAR) : [73], page 26.

une densité importante du trafic aérien. En effet, cette zone comporte des flux de départs et d'arrivées d'aéronefs depuis et vers un aéroport de départ et de destination. Les flux d'aéronefs arrivant de différentes directions doivent être fusionnés et les avions séquencés dans un flux ordonné, en suivant des routes prédéfinies appelées *Standard Terminal Arrival Routes* – STAR. Un point de référence caractérisant une route STAR est l'IAF (*Initial Approach Fix*), qui marque le début de la procédure d'approche préparant l'aéronef à l'atterrissage. Les flux de départs d'aéronefs doivent être fusionnés dans les flux d'arrivées, en suivant généralement une route de départ prédéfinie et propre à l'aéroport de départ, appelée *Standard Instrument Departure* – SID (figure 1.1, cadre bleu).

Dans la zone TMA comme dans tout l'espace aérien contrôlé, les aéronefs sont guidés par les contrôleurs. Afin de simplifier leur travail et de bien répartir les responsabilités, l'espace aérien est divisé en plusieurs centres de contrôle délimités géographiquement [28]. Selon la phase de vol de l'aéronef (figure 1.1), on peut distinguer trois grands centres de contrôle de l'espace aérien comme illustré dans la figure 1.2 :

— le **contrôle en-route** se charge d'assurer les services de contrôle pour les aéronefs en phase d'en-route, en dehors de la proximité des aéroports. Ces contrôleurs sont en contact radio avec les pilotes ; ils leur communiquent généralement des ordres relatifs au changement d'altitude, de cap ou de vitesse ;

— le **contrôle d'approche** assure les services de contrôle pour les aéronefs à proximité des aérodromes, pour les phases de montée et d'approche initiale[1]. Lors de la phase d'approche initiale, le contrôleur peut retarder l'approche finale vers la piste en utilisant une des techniques d'attente, que nous allons détailler dans la partie 1.3.2 ;

1. La phase d'approche se compose de deux étapes principales : l'approche initiale et l'approche finale [28].

— le **contrôle d'aérodrome**, ou la tour de contrôle, se charge des aéronefs en approche finale ou en attente de décollage, en plus de la gestion des aéronefs au sol le long des voies de circulation (*taxiway*). Le contrôleur d'aérodrome se charge également du séquencement des départs et d'arrivées d'aéronefs au niveau des pistes disponibles.

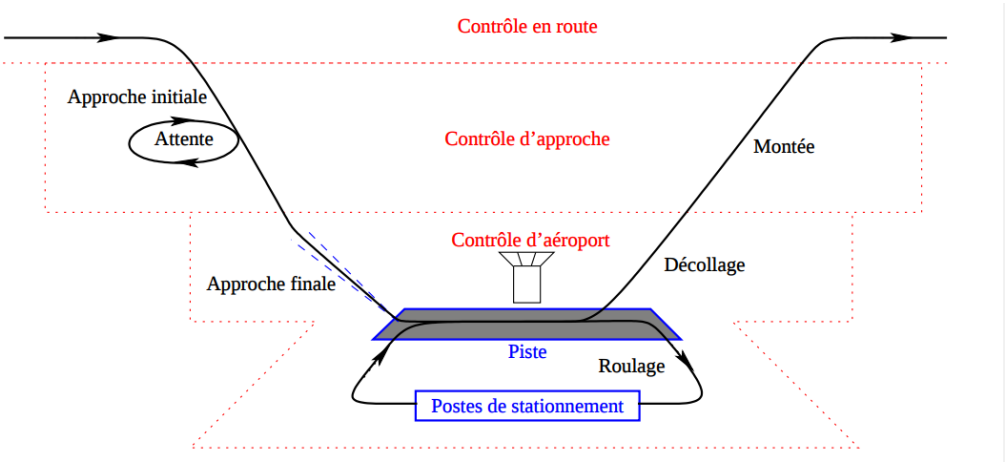


FIGURE 1.2 – Les trois principaux services de contrôle de la circulation aérienne.
Source : [28], page 6.

La responsabilité principale d'un contrôleur (d'en-route, d'approche ou d'aérodrome) est de garantir la sécurité des vols, en assurant que les aéronefs sous son contrôle respectent des normes de séparation que nous détaillons dans la section suivante.

1.2 Les règles de séparation

Pour le contrôle aérien, la séparation est la distance minimale à garantir entre deux aéronefs. C'est un moyen utilisé par le contrôleur aérien pour garantir la sécurité des vols et réduire le risque de collisions. Les deux normes de séparation les plus utilisées sont la **séparation radar** et la **séparation de turbulence de sillage** [37].

1.2.1 La séparation radar

La séparation radar est une norme de séparation entre deux avions qui s'applique dans la phase de vol en-route (plus généralement, lorsque les renseignements sur la position des aéronefs sont tirés de sources radar), où la distance de séparation verticale est de 1000 pieds et la distance horizontale est de 5 milles nautiques (ou 3 milles nautiques dans les zones congestionnées) comme illustré dans la figure 1.3.

1.2.2 La séparation de turbulence de sillage

Tous les aéronefs en vol génèrent de la turbulence de sillage (figure 1.4) qui prend essentiellement la forme de deux tourbillons tournant en sens inverse l'un de l'autre et évoluant

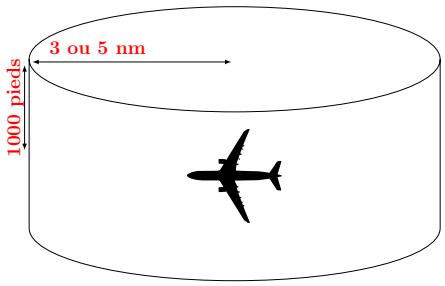


FIGURE 1.3 – Cylindre de séparation radar.

derrière l'avion (*leader*) qui les génère. L'aéronef qui pénétrerait dans la turbulence de sillage et en subirait les effets est nommé le suiveur (*follower*). La circulation du tourbillon varie en fonction du temps : elle est maximale au moment de la formation du tourbillon (circulation initiale) puis elle décroît jusqu'à un niveau de circulation qui se confond avec la turbulence atmosphérique.

FIGURE 1.4 – La turbulence de sillage d'un avion en vol.
Source : [95].

Pour minimiser les effets des turbulences de sillage qui sont plus notables dans les phases d'atterrissage et de décollage, des normes de séparation longitudinale entre aéronefs sont définies, dont la valeur dépend directement des catégories des aéronefs en jeu. L'organisation de l'aviation civile internationale (OACI) définit quatre catégories d'avion selon leur masse maximale au décollage :

— **super gros porteur** – *Super Heavy* ou S (cette catégorie contient uniquement l'Airbus A380) ;
— **gros porteur** – *Heavy* ou H ;
— **moyen tonnage** – *Medium* ou M ;
— **faible tonnage** – *Light* ou L.

En s'appuyant principalement sur cette catégorisation, les services de contrôle aérien en approche assurent une séparation minimale entre paires d'aéronefs en atterrissage et/ou en décollage afin de prévenir tout risque d'incident qui serait causé par les turbulences de sillage.

Concernant l'atterrissage, ces séparations sont données en unités de distance mais peuvent être converties en unités de temps en utilisant des vitesses représentatives, comme expliqué dans l'annexe A de [60]. La table 1.1 montre la matrice de séparation par catégorie de turbulence de sillage exprimée en milles nautiques, sur une piste unique.

		Avion suiveur		
		H	M	L
Avion générateur	H	4	5	6
	M	2.5	2.5	4
	L	2.5	2.5	2.5

TABLE 1.1 – Matrice de séparation (NM) au seuil de piste selon les catégories de turbulences de sillage de l'OACI
Source : de Neufville et al. [26].

Concernant les décollages et les cas mixtes décollages/atterrissages, des minima de séparation sont définis par les autorités de l'aviation civile. La table 1.2 présente la matrice de séparation temporelle exprimée en secondes, sur une piste unique, et pour les quatre combinaisons d'opérations possibles : deux décollages, deux atterrissages, un décollage suivi d'un atterrissage et vice versa.

			Avion suiveur					
			atterrissage			décollage		
			H	M	L	H	M	L
Avion générateur	atterrissage	H	96	157	196	75	75	75
		M	60	69	131	75	75	75
		L	60	69	82	75	75	75
	décollage	H	60	60	60	90	120	120
		M	60	60	60	60	60	60
		L	60	60	60	60	60	60

TABLE 1.2 – Matrice de séparation (secondes) au seuil de piste, selon les catégories de turbulences de sillage de l'OACI, pour les quatre combinaisons d'opérations possibles : deux décollages, deux atterrissages, un décollage suivi par un atterrissage et l'inverse.
Source : Lieder et Stolletz [72].

En pratique, les aéroports les plus fréquentés au monde gèrent plusieurs pistes à la fois. Par conséquent, les opérations sur une piste peuvent influencer, voire restreindre les mouvements sur les autres pistes. Aux États-Unis par exemple, les avions peuvent atterrir simultanément sur les pistes parallèles, à condition que leurs axes soient distants de plus de d pieds ($d = 1200$ pour les vols à vue[2] et $d = 4300$ pour les vols aux instruments[3]) [9]. Si cette condition n'est pas satisfaite, une séparation entre paires d'avions effectuant leurs opérations (atterrissages et/ou décollages) sur deux pistes différentes est alors nécessaire. La table 1.3 montre un exemple de séparation entre aéronefs opérant sur des pistes parallèles, pour les quatre combinaisons d'opérations possibles (deux décollages, deux atterrissages, décollage

2. Un vol à vue est un vol qui respecte certaines conditions de visibilité et de distance horizontale et verticale par rapport aux nuages. La prévention des collisions dans un vol à vue repose essentiellement sur le principe *voir et éviter*.

3. Un vol aux instruments est un vol effectué à l'aide des indications données par les instruments de bord de l'aéronef et les directives des contrôleurs aériens.

puis atterrissage et inversement), en fonction de l'espacement de ces pistes.

Espacement des pistes	**Décollage ⟶Décollage**	**Décollage ⟶ Atterrissage**	**Atterrissage ⟶ Décollage**	**Atterrissage ⟶ Atterrissage**
Jusqu'à 2500 pieds (Jusqu'à 760 mètres)	comme sur piste unique (table 1.2)	comme sur piste unique	Pas de séparation	comme sur piste unique
2500 - 4300 pieds (760 - 1310 mètres)	Pas de séparation	Pas de séparation	Pas de séparation	40 secondes
Plus de 4300 pieds (plus de 1310 mètres)	Pas de séparation	Pas de séparation	Pas de séparation	Pas de séparation

TABLE 1.3 – Matrice de séparation entre aéronefs opérant sur deux pistes parallèles. *Source : Lieder et Stolletz [72].*

Les normes de séparation au seuil de piste basées sur les trois catégories de turbulence de sillage (expliquées ci-dessus) ont été développées il y a plus de 40 ans. De nos jours elles sont considérées trop protectrices, voire même obsolètes. Pour cette raison, des organisations européennes et américaines pour la sécurité de la navigation aérienne (EUROCONTROL) et la Federal Aviation Administration (FAA) ont mené des recherches pour redéfinir les catégories de turbulence de sillage et la séparation qui leur est associée, en se basant sur les dernières avancées dans la compréhension du phénomène de tourbillon de sillage.

Le programme RECAT (Re CATegorization) de la FAA affine les catégories classiques de l'OACI en six nouvelles catégories. Il a pour objectif de définir une séparation dynamique entre aéronefs, c'est-à-dire une séparation qui est mise à jour en temps réel, en se basant sur les données de l'avion et les conditions météorologiques [21].

Son homologue européen est RECAT-EU [31], développé par EUROCONTROL. Dans le cadre de ce projet, la nouvelle catégorisation établie comporte également six nouvelles catégories :

— **super gros porteur** – *Super Heavy* ou SH ;

— **gros porteur classe supérieure** – *Upper Heavy* ou UH ;

— **gros porteur classe inférieure** – *Lower Heavy* ou LH ;

— **moyen tonnage classe supérieure** – *Upper Medium* ou UM ;

— **moyen tonnage classe inférieure** – *Lower Medium* ou LM ;

— **faible tonnage** – *Light* - L.

La table 1.4 présente les minima de séparation associés à cette nouvelle catégorisation. Le symbole * correspond au minimum de séparation réglementaire, qui est de $2,5$ ou 3 NM, selon les cas.

En pratique, plusieurs facteurs peuvent jouer un rôle dans l'application de la séparation entre aéronefs, comme les conditions météorologiques et la configuration des pistes[4] d'aéroports. Par exemple, dans l'aéroport international de Milan Malpensa, les deux pistes parallèles 17R/35L et 17L/35R sont utilisées à la fois pour les décollages et les atterrissages.

4. La configuration de pistes représente le nombre de pistes et leur disposition géométrique ; par exemple, une piste unique, plusieurs pistes parallèles, plusieurs pistes croisées, etc.

		Avion suiveur					
		SH	UM	LH	UM	LM	L
Avion générateur	SH	*	6	6	7	7	8
	UH	*	3	4	5	5	6
	LH	*	4	3	5	5	6
	UM	*	*	*	*	*	5
	LM	*	*	*	*	*	5
	*	*	*	*	*	*	*

TABLE 1.4 – Matrice de séparation (NM) au seuil de piste selon le programme RECAT-EU d'EUROCONTROL.
Source : [63].

La séparation d'approche finale vers la piste 17L/35R est la distance de séparation de l'OACI indiquée dans la table 1.1. En revanche, pour la piste 17R/35L, la séparation minimale entre deux atterrissages est toujours de 6 NM, quel que soit le type d'aéronef concerné, car il faut plus de temps aux avions pour libérer cette piste après l'atterrissage [15].

1.3 Techniques de séquencement d'avions

Les contrôleurs aériens assurent la sécurité des vols, en vérifiant que les aéronefs sous leur contrôle respectent bien les normes de séparation discutées dans la section précédente. Comme les règles de sécurité imposent que la piste ne peut être utilisée que par un seul avion à la fois, les aéronefs au départ des grands aéroports comme les aéronefs à l'arrivée dans la zone TMA – comprenant les vols à 30-45 minutes de la piste d'atterrissage – sont séquencés selon des critères tels que l'équité et la capacité des pistes [37]. La technique la plus fréquemment utilisée en pratique pour séquencer les avions est la règle du « *premier arrivé, premier servi* », que nous présentons dans la section 1.3.1. Par ailleurs, la régulation du trafic au départ ou à l'arrivée des grands aéroports peut conduire les contrôleurs à retarder certains avions au décollage ou à l'atterrissage. Pour ce faire, ils utilisent des *techniques d'attente* que nous détaillons dans la section 1.3.2.

1.3.1 Premier arrivé, premier servi

La règle du « premier arrivé, premier servi » (*First-Come First-Served* – FCFS) est la technique de séquencement la plus utilisée par les contrôleurs aériens. Elle consiste à planifier les atterrissages (respectivement les décollages) suivant leurs dates estimées d'arrivée (respectivement de décollage).

Pour les atterrissages, un séquenceur FCFS calcule des dates programmées d'atterrissage (*scheduled landing time*) selon l'ordre donné par les dates estimées d'arrivée *(Estimated Time of Arrival* – ETA) et en prenant en considération les contraintes de séparation à la piste. Les dates estimées d'arrivée sont obtenues lorsqu'un avion entre dans la couverture radar de

(a) Représentation schématique (source [99]) (b) Aéroport international de Zurich

FIGURE 1.5 – Une représentation schématique du trombone (a) et son illustration sur le trafic réel de l'aéroport international de Zurich (b).

l'aéroport de destination [82]. Pour les décollages, le séquenceur calcule des dates prévues de décollage (*scheduled take-off time*) suivant l'ordre donné par les avions en attente dans les positions d'attente des aéroports prévues à cet effet [82].

La technique du FCFS est facile à mettre en œuvre en pratique et nécessite généralement peu de charge de contrôle. En revanche, il en résulte rarement des solutions optimales (en terme de capacité de pistes par exemple) en cas de congestion, à cause des contraintes de séparation qui doivent être satisfaites mais sans autoriser de changement d'ordre d'atterrissage. Néanmoins, il existe dans la littérature des techniques qui permettent d'améliorer les séquences données par la FCFS, comme le concept du « **changement de position contraint** » que nous introduisons dans la partie 2.1.1.

1.3.2 Techniques d'attente

Les modèles de séquencement d'avions ont pour objectif de planifier des dates d'atterrissage (ou de décollage) en optimisant une fonction-objectif, tout en respectant un ensemble de contraintes opérationnelles, dont la contrainte de séparation au seuil de piste introduite dans la partie 1.2.2. Ces modèles supposent que les avions à séquencer sont *en attente* en l'air (ou au sol) et capables d'atterrir (ou de décoller) à tout moment. En cas de congestion, des retards surviennent et certains avions peuvent être mis en attente par les contrôleurs avant atterrissage (ou décollage) [3]. Cette attente est particulièrement critique pour les avions en atterrissage qui, contrairement aux avions demandant le décollage, ne peuvent pas être arrêtés dans l'espace aérien.

Pour les décollages, la mise en attente est relativement simple : les avions peuvent *attendre* dans leur position de parking, ou à certains points spécifiques de l'aéroport prévus à cet effet. Pour les atterrissages, certaines routes d'arrivée d'avions dans les aéroports sont construites sous forme de trombone, comme illustré sur la figure 1.5. Il s'agit de routes d'arrivée que les avions suivent, jusqu'à ce qu'ils reçoivent une autorisation (*clearance*) de la part des contrôleurs aériens. Cette structure de route est très présente dans plusieurs aéroports dans le monde, car elle permet un flux de trafic fluide vers les pistes d'atterrissage [99].

(a) Vector-For-Spacing (b) Circuit d'attente

FIGURE 1.6 – Représentation schématique de deux techniques d'attente : la technique du *vector-for-spacing* (a) et le circuit d'attente (b). La ligne pointillée représente la trajectoire directe.

Dans la revue de littérature de Bennell *et al.* [11], d'autres techniques utilisées par les contrôleurs pour la mise en attente d'avions sont présentées. Les plus fréquemment utilisées sont :

— le ***vector-for-spacing*** illustré dans la figure 1.6a. Il consiste à rallonger la trajectoire d'un avion de la manière suivante : au lieu de suivre le chemin direct entre deux points, l'avion est dévié pendant une courte durée, puis il rejoint sa trajectoire initiale ;

— les ***circuits d'attente*** (*holding pattern*) ressemblent à des hippodromes superposés (figure 1.6b) construits pour retarder les avions. En effet, en cas de trafic important, l'avion entre dans le circuit d'attente et se met à *tourner* jusqu'à ce qu'il reçoive l'autorisation du contrôleur de quitter l'hippodrome pour entamer son approche finale.

1.4 Outils d'aide à la décision

Pour assister les contrôleurs aériens dans l'ordonnancement des arrivées et des départs d'avions, plusieurs outils d'aide à la décision sont mis à leur disposition. L'un de ces outils est le *Center-TRACON Automation System* (CTAS), développé par la NASA et la FAA [29]. Il regroupe trois outils, à savoir : le *traffic management advisor* qui fournit des dates prévues d'atterrissage ainsi qu'une piste disponible, le *descent advisor* qui aide les contrôleurs dans le guidage d'avions vers un point spécifique de l'espace aérien (*metering fix*[5]) et l'outil *final approach sequencing tool* qui fournit des recommandations de vitesse et de cap. Le CTAS contient une composante appelée *the expedite departure path* qui aide à gérer les décollages d'avions [29].

Les analogues européens de CTAS sont les outils *Arrival MANager* (AMAN) et *Departure MANager* (DMAN), présentés dans les deux sections suivantes.

5. Points spécifiques dans l'espace aérien le long d'une route aérienne, vers lesquels les avions sont guidés afin d'entrer dans l'espace aérien terminal entourant les aéroports.

1.4.1 Arrival MANager

Tout comme CTAS, les outils AMAN visent à aider les contrôleurs aériens à séquencer les flux d'arrivée d'avions vers des points spécifiques, tels que le seuil de piste ou les *balises* (points repères) de la zone TMA. Le principal objectif de ces outils est d'optimiser la capacité des pistes et/ou de réguler les flux d'aéronefs dans l'espace aéroportuaire.

Pour fournir des recommandations aux contrôleurs, AMAN se base sur plusieurs types de données tels que le plan de vol, les données radar, les modèles de performance d'avions et les données météorologiques [77]. Dans les versions basiques d'AMAN, deux types de recommandations sont fournies :

— une séquence d'atterrissage qui optimise la capacité des pistes, comme illustré dans la figure 1.7;

— une gestion des retards pour chaque avion dans la séquence, sous forme de temps à gagner (*Time To Gain*) ou de temps à perdre (*Time To Lose*).

Les versions les plus développées d'AMAN peuvent fournir des actions de contrôle plus avancées, telles que le guidage radar (recommandations d'altitude, vitesse et cap). Il existe une version d'AMAN appelée *Extended AMAN* (E-AMAN), dont l'horizon opérationnel considéré est étendu jusqu'à 500 NM de l'aéroport, au lieu de 100 à 200 NM pour les versions d'AMAN actuelles [79]. La motivation d'une telle extension est de commencer le séquencement d'avions plus tôt afin de réduire la congestion et la gêne sonore dans l'espace aérien à proximité des aéroports.

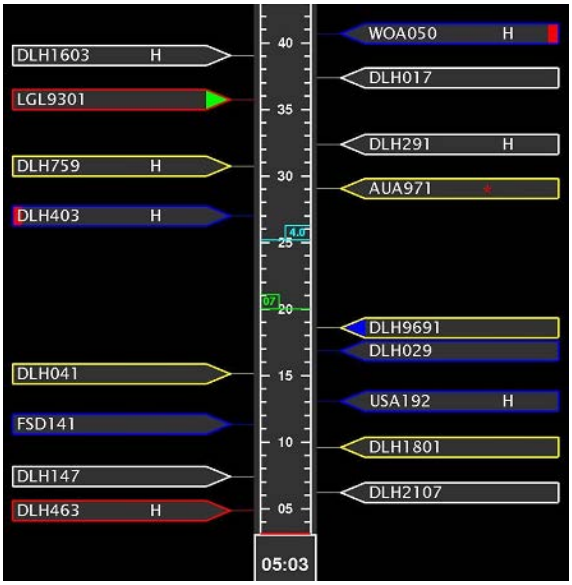


FIGURE 1.7 – Une capture d'écran d'une partie de l'outil AMAN.
Source : [77].

1.4.2 Departure MANager

DMAN est un outil d'aide à la décision pour les aéronefs au sol. Il a été conçu de façon flexible pour pouvoir s'adapter facilement à la configuration de l'aéroport dans lequel il est

utilisé, dans l'objectif d'optimiser l'utilisation de l'espace aéroportuaire et plus particulièrement les pistes. DMAN fournit également deux recommandations pour chaque avion [30] :

— un horaire programmé de décollage (*scheduled take-off time*) ;
— une trajectoire de montée sans conflit (*conflict-free climbing trajectories*).

En raison de la difficulté supplémentaire que représente les atterrissages (pas d'arrêt possible en l'air), comparés aux décollages, AMAN est prioritaire par rapport à DMAN. En effet, ce dernier programme les départs d'avions en fonction des séquences proposées par AMAN, c'est-à-dire qu'il place les décollages dans les intervalles entre deux atterrissages, en respectant les contraintes de séparation.

Maintenant que nous avons présenté le cadre opérationnel dans lequel s'insère le problème d'ordonnancement (d'atterrissages et/ou de décollages) d'avions, nous discutons dans le chapitre suivant comment ce problème est considéré dans la littérature (énoncés mathématiques, fonctions-objectifs, contraintes, modélisation).

Chapitre 2

Le problème d'ordonnancement d'avions en zone d'approche aéroportuaire

Dans ce chapitre, nous présentons l'énoncé classique d'un problème d'ordonnancement d'avions en zone d'approche aéroportuaire, avec les différentes contraintes opérationnelles impliquées, ainsi que ses objectifs, selon la partie prenante considérée. Puis nous détaillons les variantes de ce problème, selon l'opération à ordonnancer (atterrissages et/ou décollages), et selon les aspects de l'espace aérien impliqués (pistes uniquement, piste et autres points de la zone TMA). Nous présentons ensuite le modèle mathématique le plus cité de la littérature ; il s'agit de la formulation de Beasley [9] pour l'ordonnancement d'atterrissages. Enfin, nous mettons en évidence le lien entre le problème d'ordonnancement d'avions et d'autres problèmes classiques en optimisation combinatoire.

Sommaire

2.1 Énoncé du problème

Le problème d'ordonnancement d'avions en zone d'approche aéroportuaire a pour objectif l'optimisation de l'utilisation des infrastructures aéroportuaires, et plus particulièrement la

piste, connue pour être le goulot d'étranglement des aéroports. Dans la littérature, ce problème est généralement divisé en trois types de problèmes, impliquant soit les atterrissages, soit les décollages, soit les deux.

— Le problème d'ordonnancement des atterrissages (ALP). Il consiste à attribuer aux avions demandant l'atterrissage une piste disponible ainsi qu'une heure cible d'atterrissage.
— Le deuxième problème s'intéresse uniquement aux décollages (ATP). Il consiste à allouer aux avions en position de parking des heures cibles de décollage aux pistes disponibles.
— Quand il s'agit d'ordonnancer simultanément les atterrissages et les décollages, le problème est alors appelé : problème d'ordonnancement (ou de séquencement) d'avions (ASP).

Depuis la publication de la première approche pour résoudre le problème d'ordonnancement d'avions par Dear [27] en 1976, plusieurs travaux ont traité les différents sous-problèmes mentionnés ci-dessus. Certains de ces travaux ne considèrent que les pistes, ce qui est le plus fréquent dans la littérature, d'autres travaux impliquent d'autres aspects de l'espace aérien aéroportuaire comme les points d'entrée dans la zone TMA, les routes d'approche ou de départ et les voies de circulation. Plusieurs énoncés et formulations mathématiques correspondantes sont proposés dans ces travaux, suivant le problème considéré (ALP, ATP ou ASP) et selon les aspects de l'espace aérien aéroportuaire pris en compte.

Commençons d'abord par définir l'énoncé de base d'un problème d'ordonnancement d'avions, commun aux trois problèmes mentionnés ci-dessus. Nous verrons plus en détail chacun de ces problèmes (ALP, ATP ou ASP) avec les différents aspects de l'espace aérien pris en compte dans les sections 2.2 et 2.3.

Considérons un ensemble de pistes (*runways*) $\mathcal{R} = \{1, 2, ..., R\}$ et un ensemble d'avions (*aircraft*) $\mathcal{A} = \{1, 2, ..., N\}$ demandant l'atterrissage (ou le décollage) dans (depuis) le même aéroport de destination (départ). Chaque avion $i \in \mathcal{A}$ a une date préférentielle d'atterrissage (décollage), notée T_i, attribuée par les autorités de gestion de flux aériens[1], tels que le *Network Manager Operations Centre* – NMOC en Europe. Chaque avion possède aussi une fenêtre de temps qui correspond à un intervalle temporel dans lequel l'aéronef doit absolument atterrir (décoller). Notons $[E_i, L_i]$ (*earliest*, *latest*) cette fenêtre de temps.

Un problème d'ordonnancement d'avions consiste d'abord à attribuer à chaque avion une piste disponible, $r \in \mathcal{R}$, puis à allouer pour chaque avion, $i \in \mathcal{A}$, une date cible d'opération (atterrissage ou décollage), notée x_i, tout en respectant un ensemble de contraintes opérationnelles, dans le but d'optimiser un objectif donné. Les contraintes à respecter concernent principalement la séparation entre les aéronefs pour assurer la sécurité des opérations et les restrictions de *fenêtres de temps* pour éviter que les avions subissent des retards excessifs. Les objectifs à optimiser varient selon le point de vue de la partie prenante considérée : aéroports, compagnies aériennes, contrôleurs aériens ou organismes gouvernementaux. Les différents objectifs de chacune de ces parties prenantes ainsi que les fonctions-objectif typiques de la littérature seront détaillés dans la section 2.1.2.

1. En pratique, dans le cas des décollages, les autorités de gestion de flux aérien attribuent souvent un intervalle temporel de 15 minutes autour de la date préférentielle T_i (par exemple, $[T_i - 5, T_i + 10]$) [42].

Dans la suite de cette section, nous commençons par présenter les contraintes puis les objectifs communs aux différents énoncés d'un problème d'ordonnancement d'avions. Ensuite, nous détaillons chaque problème selon les aspects de l'espace aérien aéroportuaire pris en compte : piste uniquement (section 2.2) ; pistes et autres aspects de la zone TMA (section 2.3).

Rappelons que dans de ce manuscrit, nous adoptons la nomenclature "problème d'ordonnancement d'avions" pour désigner indifféremment l'ALP, l'ATP ou l'ASP. Nous préciserons quel problème est concerné quand cela sera pertinent.

2.1.1 Contraintes

Un certain nombre de contraintes s'imposent quand on considère un problème d'ordonnancement d'avions, indépendamment de la variante considérée (ALP, ATP ou ASP) et les aspects de l'espace aérien pris en compte (piste unique, pistes multiples, IAF, routes d'approche et/ou de départ). La contrainte la plus importante dans ce genre de problème est la *séparation* qui garantit la sécurité des vols en évitant des rapprochements dangereux entre aéronefs. Il est pertinent d'imposer aussi des contraintes dites de *fenêtre de temps* (*time-window constraints*), qui évitent que les avions subissent des retards importants. D'autres contraintes peuvent s'ajouter au problème, pour le simplifier (*changement de position contraint – constrained position shifting*) ou pour considérer les priorités des compagnies aériennes (contraintes de *précédence*). Nous détaillons chacune de ces contraintes dans ce qui suit.

— La **séparation** est une contrainte fondamentale qui garantit la sécurité des opérations sur les pistes ou dans l'espace aérien. Dans l'espace aérien, c'est souvent la *séparation radar* qui est concernée. C'est une distance de séparation verticale (1000 pieds) et horizontale (3 ou 5 miles nautiques selon la congestion) que les aéronefs en vol doivent maintenir.

  Pour la piste, la séparation concernée est relative aux catégories de turbulence de sillage, détaillée dans la partie 1.2.2. Dans ce contexte, plusieurs types de séparation ont été modélisées dans la littérature, selon le nombre et la configuration des pistes. On trouve principalement trois types de séparation au seuil de piste : *successive*, *complète* ou *diagonale*.

  La séparation *successive*, comme son nom l'indique, concerne deux avions successifs dans une séquence opérant sur la même piste, tandis que la séparation *complète* touche toutes les paires d'avions dans cette séquence.

  Dans le cas de pistes multiples, la question de séparation entre avions opérant sur deux pistes différentes se pose. Il s'agit de la séparation dite *diagonale*. Les valeurs de cette séparation en fonction des pistes et leur espacement on été présentées dans la table 1.3 de la partie 1.2.2.

— La **fenêtre de temps** est un intervalle $[E, L]$ défini par une heure d'atterrissage (ou de décollage) au plus tôt E et au plus tard L, compte tenu (respectivement) d'éventuelles accélérations et de la disponibilité de carburant à bord. En effet, quand un avion arrive à proximité de la zone aéroportuaire TMA, les outils d'aide à la décision calculent une heure estimée d'arrivée – *Estimated Time of Arrival* (ETA) au seuil d'une piste disponible. Si l'avion accélère, la date d'atterrissage réelle (*Actual Landing Time*) peut

être antérieure à l'ETA. Inversement, si l'avion est retardé après être entré dans la zone du centre de contrôle, la date réelle d'atterrissage sera plus tardive que l'ETA mais sera limitée par la disponibilité de carburant à bord [69].

Prakash *et al.* [85] proposent de définir la fenêtre de temps pour les avions demandant l'atterrissage comme suit :

- $E = T - 60$ secondes car c'est le plus économique pour les avions à l'arrivée ;
- $L = T + 1800$ secondes comme moyenne pour la disponibilité de carburant.

— Le **changement de positions contraint** (*Constrained Position Shifting* – CPS) est un concept théorique introduit pour la première fois en 1976, dans la thèse de doctorat de Dear [27]. Il permet aux avions de dévier de leur position initiale dans la séquence FCFS jusqu'à un nombre maximal de changements de positions, appelé *Maximum-Position Shift* (MPS) en anglais. Par exemple, si un avion occupe la position 5 dans une séquence FCFS, et que le nombre maximal de changements de positions est égal à 2, alors cet avion ne peut être décalé que dans les positions $3, 4, 5, 6$ ou 7. Le changement de position contraint a un double avantage : théorique, car il permet d'améliorer la complexité et de réduire le temps de calcul des méthodes exactes, et pratique puisqu'il garantit une certaine équité entre les avions, tout en améliorant les séquences initiales données par la séquence FCFS.

Bien qu'il limite le nombre de séquences réalisables, le CPS peut considérablement améliorer la séquence initiale du FCFS et augmenter la capacité des pistes (figure 2.1) ; il permet d'éviter la situation la plus indésirable dans laquelle un atterrissage d'avion gros porteur (*Heavy*) est suivi d'un atterrissage d'un avion de faible tonnage (*Light*), nécessitant la plus grande séparation de 6 NM, soit environ 196 secondes. Les modèles qui adoptent le concept du CPS sont donc fréquents dans la littérature [6, 85, 89] pour toutes les raisons évoquées ci-dessus.

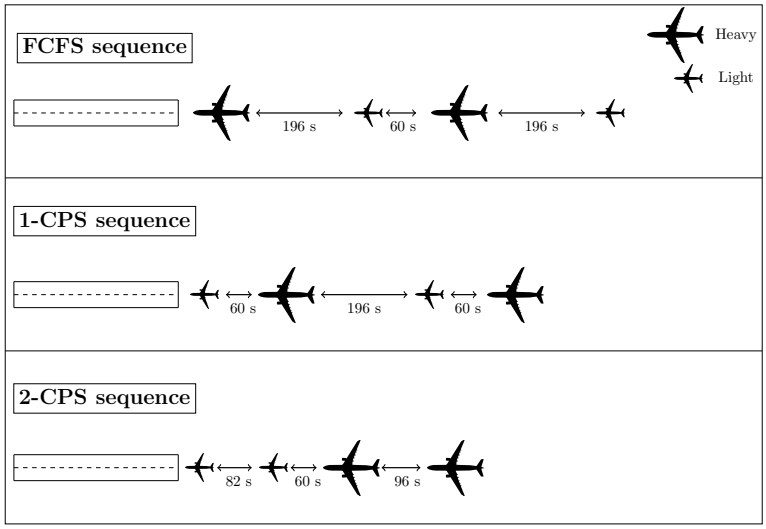


FIGURE 2.1 – Comparaison entre trois séquences d'atterrissage composées de quatre avions.

Sur un simple exemple présenté en figure 2.1, on observe qu'en utilisant les contraintes de séparation temporelle présentées dans la table 1.2, la FCFS nécessite un total de

452 secondes pour faire atterrir les quatre avions alors que la FCFS avec un nombre maximal de changements de positions égal à 1 requière 316 secondes pour faire atterrir les mêmes avions, et la FCFS avec un nombre maximal de changements de position égal à 2 nécessite seulement 238 secondes pour faire atterrir la totalité des avions dans la séquence.

— Les **contraintes de précédence** (*precedence constraints*) permettent de modéliser la priorité de certains avions par rapport à d'autres. En effet, pour les aéronefs demandant l'atterrissage et provenant d'une même route aérienne, les contrôleurs ne peuvent pas autoriser de dépassements. Par conséquent, il est pertinent de considérer un ordre de priorité selon l'ordre d'arrivée des avions sur la route aérienne, ce qui favorise ainsi une certaine équité. Une autre motivation pour l'ajout de contraintes de précédence provient des compagnies aériennes elles-mêmes, en fonction de leurs schémas de connections [6].

2.1.2 Fonctions-objectifs

Le transport aérien implique plusieurs parties prenantes : les contrôleurs aériens, les compagnies aériennes, les aéroports et les organismes gouvernementaux qui ont des objectifs différents, parfois contradictoires, ce qui mène souvent à considérer des éventuels compromis [11]. Nous résumons dans la table 2.1 les principaux objectifs des quatre parties prenantes ; cette classification (deux premières colonnes seulement) est tirée de la revue de littérature de Bennell *et al.* [11]. Nous ajoutons à cette classification une troisième colonne avec, pour chaque objectif, un exemple de la littérature qui lui correspond.

Selon l'objectif de la partie prenante adopté, la formulation mathématique du problème d'ordonnancement d'avions diffère. Néanmoins, on peut distinguer dans la littérature scientifique des fonctions-objectifs typiques telles que :

— le coût total de déviations (notée x_i^- et x_i^+) par rapport aux dates préférentielles T_i à minimiser : $\min \sum_{i \in \mathcal{A}} (c_i^- x_i^- + c_i^+ x_i^+)$, ce qui revient à maximiser la ponctualité (point de vue des compagnies aériennes) ;

— la moyenne des heures cibles d'opération (atterrissage ou décollage) : $\min \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} x_i$;

— la date d'opération du dernier avion dans la séquence (*makespan*) :
$\min \max_{i \in \mathcal{A}} x_i$, ce qui favorise des séquences de longueur minimale, et par conséquent maximise le rendement des pistes (point de vue des aéroports).

2.2 Ordonnancement au seuil de piste

Le problème d'ordonnancement d'avions mettant en jeu uniquement les pistes permet de mieux connaître les capacités de ces dernières, et par conséquent de les exploiter de façon optimale. Il est donc pertinent de considérer un problème d'ordonnancement d'avions en mettant en jeu seulement les pistes. En pratique par exemple, dans les deux aéroports

Parties prenantes	Objectifs	Exemple de la littérature
Gouvernement	◦ Minimiser les effets environnementaux (pollution, gêne sonore...).	◦ Evertse et Visser [32]
Aéroport	◦ Maximiser la ponctualité ; ◦ Maximiser le rendement des pistes ; ◦ Minimiser le nombre de changements de porte dû aux retards des départs.	◦ Beasley *et al.* [9] ◦ Shohel et Alam [1] ◦ Mohleji et Tene [78]
Compagnie aérienne	◦ Maximiser la ponctualité ; ◦ Minimiser les retards au départ et à l'arrivée ; ◦ Minimiser les coûts, en particulier le coût du carburant ; ◦ Maximiser le respect des priorités des compagnies aériennes ;	◦ Salehipour *et al.* [93] ◦ Furini *et al.* [37] ◦ Bennell *et al.* [12] ◦ Balakrishnan et Chandran [6]
Contrôleur aérien	◦ Maximiser la capacité de la piste ; ◦ Maximiser l'équité entre les avions ; ◦ Minimiser le temps de roulage au sol ; ◦ Minimiser les retards au départ et à l'arrivée ; ◦ Minimiser la charge de travail des contrôleurs ; ◦ Maximiser l'équilibre entre les départs et les arrivées d'avions.	◦ Balakrishnan et Chandran [6] ◦ Prakash *et al.* [85] ◦ Soares *et al.* [19] ◦ Furini *et al.* [37] ◦ Kim *et al.* [67] ◦ Kim *et al.* [66]

TABLE 2.1 – Résumé des principaux objectifs pour les quatre parties prenantes impliquées dans le transport aérien, avec exemples de la littérature.

londoniens Heathrow et Gatwick, un modèle d'optimisation mixte en nombres entiers est utilisé par les contrôleurs aériens, afin d'avoir un aperçu de la capacité des pistes [9].

Dans la suite de cette section, nous commençons par présenter le problème d'ordonnancement d'avions qui ne considère que les atterrissages (ALP) sur une piste unique. Ensuite, nous passons au problème considérant simultanément les atterrissages et les décollages (ASP). Finalement, nous abordons le problème mettant en jeu plusieurs pistes.

2.2.1 Ordonnancement des atterrissages sur une piste unique

L'ordonnancement des atterrissages d'avions (ALP) sur une piste unique correspond à l'énoncé de base présenté au début de cette section, sans la décision relative à l'allocation de piste. Rappelons que ce problème consiste à attribuer à chaque avion une date cible d'atterrissage, tout en respectant un ensemble de contraintes (Section 2.1.1) et en optimisant un objectif donné (Section 2.1.2). D'après la revue de littérature de Bennell *et al.* [11], cet énoncé est le plus fréquemment étudié dans la littérature de l'ordonnancement d'avions. Ce problème est étudié par exemple dans [5, 85, 92].

La contrainte fondamentale à respecter est la contrainte de séparation au seuil de piste entre toutes les paires d'avions, afin de garantir la sécurité des opérations. Rappelons que l'OACI classifie les avions principalement en trois catégories selon leur masse maximale au

décollage. Des normes de séparation minimale, dites de turbulence de sillage, sont alors définies en fonction de la catégorie de chaque paire d'aéronefs atterrissant. Dans le cas ALP sur une piste unique, on suppose généralement que ces normes de séparation vérifient une propriété importante, qui est l'*inégalité triangulaire* (définition 2.2.1). Par conséquent, il n'est plus nécessaire de vérifier la séparation entre toutes les paires d'avions dans une séquence d'atterrissage, mais seulement entre paires d'avions successifs. Autrement dit, la séparation *successive* implique la séparation *complète*.

Définition 2.2.1

La séparation de turbulence de sillage entre paires d'avions $(i, j) \in \mathcal{A} \times \mathcal{A}$, notée S_{ij}, satisfait l'inégalité triangulaire si pour tout $(i, j, k) \in \mathcal{A} \times \mathcal{A} \times \mathcal{A} : i \neq j, j \neq k, k \neq i$, on a :

$$S_{ik} \leq S_{ij} + S_{jk}.$$

2.2.2 Ordonnancement simultané des atterrissages et des décollages sur une piste

Certains aéroports dans le monde disposent d'une seule piste, dédiée à la fois aux atterrissages et aux décollages. Tel est le cas par exemple de l'aéroport de Gatwick, le deuxième plus grand aéroport de Londres, connu pour être l'aéroport le plus fréquenté dans le monde parmi ceux n'ayant qu'une seule piste pour le trafic commercial [89]. Un deuxième exemple est l'aéroport de Milan-Linate, qui, comme l'aéroport de Gatwick, dispose d'une seule piste pour le trafic commercial [37]. En conséquence, le problème d'ordonnancement simultané d'atterrissages et des décollages (ASP) est aussi un problème dont l'étude est pertinente. L'ASP a motivé plusieurs travaux dans la littérature [37, 85, 89].

L'énoncé de ce problème correspond à l'énoncé de base présenté en début de cette section. Il s'agit de trouver pour chaque avion $i \in \mathcal{A}$, une date cible d'opération (atterrissage ou décollage), en respectant les contraintes opérationnelles, principalement la *séparation* et les *fenêtres de temps*, dans le but d'optimiser un objectif donné, comme par exemple maximiser la capacité des pistes. La grande différence avec le cas d'ordonnancement des atterrissages uniquement est que la matrice de séparation ici (table 1.2) ne vérifie pas nécessairement l'inégalité triangulaire (définition 2.2.1). Par conséquent, il ne suffit plus de vérifier les séparations entre paires d'avions successifs pour assurer la séparation *complète* entre toutes les paires d'avions. Autrement dit, il faut imposer la séparation de turbulence de sillage entre toutes les paires d'avions dans la séquence.

Bien qu'ils traitent le même problème (ASP), les quatre exemples cités ci-dessus ne considèrent pas tous les mêmes objectifs et contraintes. En effet, Prakash *et al.* [85] et Rodriguez-Diaz *et al.* [89] considèrent, en plus de la séparation et les *fenêtres de temps*, la contrainte du *changement de position* (voir Section 2.1.1), ce qui n'est pas le cas des deux autres articles de Furini *et al.* [36, 37]. Cependant, les travaux de [85, 89] abordent deux objectifs différents : le premier vise à minimiser la date du dernier avion posé dans la séquence, le deuxième minimise le retard total de la séquence. Nous faisons une revue complète des approches de résolution utilisées dans chacun de ces travaux dans le chapitre 3.

2.2.3 Ordonnancement sur pistes multiples

Le problème d'ordonnancement d'avions mettant en jeu plusieurs pistes nécessite, en plus de l'attribution d'une date cible d'opération (atterrissage ou décollage), l'allocation d'une piste disponible pour chacune de ces opérations.

Le cas multipiste a motivé plusieurs travaux dans la littérature ; on peut citer par exemple [9, 18, 71, 72]. Cependant, l'énoncé n'est pas toujours le même : il dépend de la *configuration* des pistes considérées, c'est-à-dire de la disposition géométrique des pistes entre elles et les types d'opérations (atterrissages et/ou décollages) autorisées. On peut distinguer principalement quarte configurations de pistes [63] :

— pistes **indépendantes** : les opérations sur une piste ne sont pas affectées par les opérations sur les autres pistes. Cette configuration est étudiée par exemple dans [18] ;

— pistes **interdépendantes** : par opposition à la première configuration, les opérations sur une piste sont affectées par les opérations sur les autres pistes. Un exemple notable de la littérature considérant ce cas est [72] ;

— pistes **homogènes** : pistes acceptant les mêmes types d'avions et les mêmes types d'opérations (décollage et/ou atterrissage). Ce cas est étudié par exemple dans [9], où les auteurs considèrent jusqu'à cinq pistes réservées uniquement aux atterrissages, et acceptant tous les types d'avions ;

— pistes **hétérogènes** : pistes n'acceptant qu'un certain type d'opérations (décollage ou atterrissage), ou un certain type d'avions (les plus lourds par exemple). Cette configuration est étudiée par exemple dans [72], en même temps que la configuration de pistes interdépendantes.

Si l'on considère la configuration de pistes indépendantes, on peut avoir deux opérations simultanément sur deux pistes différentes. Par conséquent on a pas besoin d'imposer les contraintes de séparation *diagonale*. C'est le cas par exemple des pistes parallèles suffisamment espacées (leurs axes sont distants de plus de 1300 mètres). Dans le cas contraire (pistes interdépendantes), les contraintes de séparation *diagonale* sont nécessaires. C'est le cas par exemple de pistes parallèles espacées de moins de 760 mètres ; les valeurs de la séparation diagonale en fonction de l'espacement des pistes on été présentées dans la table 1.3. Par ailleurs, la configuration de pistes la plus générale est celle des pistes interdépendantes et hétérogènes, qui d'après [63], n'a été considérée qu'une seule fois dans la littérature dans [72], dans lequel l'interdépendance est spécifique non seulement à la paire d'avions mais aussi à la paire de pistes utilisées.

2.3 Ordonnancement au seuil de piste et aux balises de la zone TMA

Le problème d'ordonnancement d'avions (ALP, ATP ou ASP) impliquant uniquement les pistes est intéressant, car il permet de mieux connaître la capacité des pistes, et par conséquent de mieux les exploiter. Néanmoins, prendre aussi en considération d'autres aspects

du problème propres à chaque aéroport demeure plus réaliste. Un des aspects de l'espace aérien aéroportuaire les plus importants à considérer est les IAFs, qui sont des balises (points repère) de la zone TMA où les flux d'arrivées d'avions sont fusionnés et les avions séquencés, afin de préparer leur atterrissage. En effet, pendant les périodes de congestion, des goulots d'étranglement peuvent se produire ailleurs dans cette zone, ce qui entraîne des retards importants, non seulement au niveau des pistes, mais aussi aux balises de cette zone [40].

Ce problème requiert deux types de décision : une première décision au niveau des balises de la zone terminale ; elle concerne l'attribution d'un point d'entrée dans la TMA ainsi qu'une date cible de passage à ce point. La deuxième décision concerne les pistes, c'est-à-dire l'affectation d'une piste ainsi qu'une heure cible d'opération sur cette piste. Les contraintes principales d'un tel problème concernent – comme auparavant – la séparation afin de garantir la sécurité des vols et des opérations sur la piste, et la contrainte de fenêtre de temps afin d'éviter que les avions subissent des retards importants. La contrainte de séparation doit être vérifiée non seulement au seuil des pistes, mais aussi au niveau des balises de la zone TMA considérés. Une façon de prendre en compte cette séparation est proposée dans [64], dont les auteurs considèrent la séparation radar de 5 miles nautiques pour des avions régulés à une vitesse commune de 250 nœuds, ce qui donne une séparation temporelle de 72 secondes à vérifier au niveau des balises de la zone TMA, en l'occurrence les IAF dans cet article.

L'ordonnancement d'avions prenant en considération des balises de la zone TMA a motivé certains travaux dans la littérature [64, 65, 66]. Dans les deux articles de Kim *et al.* [65, 66], le problème considéré est l'ordonnancement d'avions au niveau des pistes et des balises dans la zone TMA, dans le but de minimiser les émissions dans cette zone. La différence entre [65] et [66] est que dans [66], les auteurs considèrent en plus des atterrissages, l'étape de roulage au sol (*taxiing*). Dans [64], l'énoncé est différent : il s'agit d'un ordonnancement « étendu » des arrivées d'avions, appelée *Extended Arrival Management* dans la littérature. Ce problème consiste à commencer l'ordonnancement d'avions quelques heures avant leur atterrissage, dans le but d'optimiser l'utilisation des pistes tout en minimisant les actions de dernière minutes (rappelons que dans l'ordonnancement des atterrissages classique, les avions sont pris en compte lorsqu'ils sont à environ 45-60 minutes de l'aéroport de destination [69]). Les auteurs de [64] proposent alors un modèle de programmation stochastique à deux étapes : en première étape, les avions sont ordonnancés sur un même IAF, c'est-à-dire un ordre et une date cibles de passage par ce point sont attribués à chaque avion tout en respectant les contraintes opérationnelles. En deuxième étape, les avions sont séquencés au niveau de la piste, dans le but de minimiser les déviations entre les dates cibles d'atterrissage et les dates non contraintes d'atterrissage, correspondant aux dates d'atterrissage dans un espace aérien terminal décongestionné. Ces deux étapes " successives " de la programmation stochastique sont en fait prises en compte dans une seule optimisation.

Un autre énoncé encore plus réaliste est considéré dans Bianco *et al.* [15]. Il s'agit d'ordonnancer les avions dans la zone TMA, en prenant en considération toutes les étapes par lesquelles passent l'avion dans cette zone : l'approche de l'aéroport avec les différentes routes d'approche, l'atterrissage, le roulage au sol et le décollage avec les différentes routes de départ. La contrainte principale considérée dans cet article est la séparation, non seulement entre aéronefs dans certaines régions spécifiques de la zone TMA, mais aussi pendant le roulage au sol et au niveau des pistes. Ce problème est encore plus complexe que l'énoncé

considérant uniquement la piste, qui est déjà connu pour être NP-difficile (voir section 2.5). Afin de diminuer la complexité du problème, les auteurs de [15] ont considéré – en plus de la séparation – le *changement de position contraint* (CPS) introduite dans la section 2.1.1. Nous présentons une revue complète des méthodes de résolution utilisées dans cet article ainsi que tous les articles cités dans cette section dans le chapitre 3.

Après avoir étudié le problème d'ordonnancement d'avions en zone d'approche aéroportuaire, avec les différentes variantes, énoncés et les contraintes qu'il implique, nous présentons maintenant la modélisation mathématique du problème. Nous choisissons de présenter la formulation mathématique de Beasley *et al.* [9] pour l'ordonnancement des atterrissages d'avions (ALP) sur des pistes interdépendantes, cet article étant le plus cité de la littérature (plus de 500 citations selon *Google scholar*).

2.4 Modélisation

Beasley *et al.* [9] présente un modèle d'aide à la décision pour l'ordonnancement des atterrissages d'avions qui propose des heures cibles d'atterrissage, mais qui laisse les contrôleurs décider des consignes à donner aux avions afin de respecter au mieux les dates cibles calculées. Le modèle proposé est une formulation mathématique d'optimisation mixte en nombres entiers pour le cas statique déterministe, en considérant le cas de pistes multiples interdépendantes. Nous présentons donc ici leur modèle.

Paramètres du modèle (données d'entrée)

N : nombre total d'avions à ordonnancer ;
R : nombre total de pistes disponibles ;
$\mathcal{A}$: ensemble des indices d'avions ($\mathcal{A} = \{1, 2, ..., N\}$) ;
$\mathcal{R}$: ensemble des indices de pistes ($\mathcal{R} = \{1, 2, ..., R\}$).

Pour chaque avion $i \in \mathcal{A}$:

E_i : date d'atterrissage au plus tôt de i ;
T_i : date préférentielle d'atterrissage de i ;
L_i : date d'atterrissage au plus tard de i ;
c_i^- : coût ($c_i^- \geq 0$) par unité de temps d'atterrir plus tôt que T_i ;
c_i^+ : coût ($c_i^+ \geq 0$) par unité de temps d'atterrir plus tard que T_i ;

Pour chaque paire d'avions $(i, j) \in \mathcal{A} \times \mathcal{A}$:

S_{ij} : séparation minimale temporelle (≥ 0) entre les deux avions i et j atterrissant sur la même piste (l'avion i atterrit avant l'avion j) ;
s_{ij} : séparation minimale temporelle (≥ 0) entre les deux avions i et j atterrissant sur deux pistes différentes (l'avion i atterrit avant l'avion j).

Type		Variable
Continu	t_i	heure cible d'atterrissage de l'avion i,
	t_i^-	temps d'avance de l'heure cible d'atterrissage t_i par rapport à l'heure préférentielle T_i,
	t_i^+	temps de retard de l'heure cible d'atterrissage t_i par rapport à l'heure préférentielle T_i.
Binaire	$\delta_{ij} =$	$\begin{cases} 1 \text{ si l'avion } i \text{ atterrit avant } j, \\ (i,j) \in \mathcal{A} \times \mathcal{A} : i \neq j, \\ 0 \text{ sinon.} \end{cases}$
	$z_{ij} =$	$\begin{cases} 1 \text{ si les avions } i \text{ et } j \text{ atterrissent sur la même piste,} \\ (i,j) \in \mathcal{A} \times \mathcal{A} : i \neq j, \\ 0 \text{ sinon.} \end{cases}$
	$y_{ir} =$	$\begin{cases} 1 \text{ si l'avion } i \text{ est affecté à la piste } r, \\ i \in \mathcal{A}, r \in \mathcal{R}, \\ 0 \text{ sinon.} \end{cases}$

TABLE 2.2 – Variables d'optimisation du modèle de Beasley *et al.* [9].

Variable de décision

Le modèle proposé par Beasley *et al.* [9] contient deux types de variables de décision : des variables continues pour affecter les dates d'atterrissages, et des variables binaires pour l'ordre dans la séquence et l'affectation de piste. Nous présentons ces variables dans la table 2.2.

Contraintes

Beasley *et al.* [9] considèrent deux types de contraintes opérationnelles : les *fenêtres de temps* et la *séparation* au seuil de piste pour les avions atterrissant sur une même piste ou sur deux pistes différentes. À cela s'ajoute d'autres contraintes pour le *séquencement*, l'*affectation des pistes* et les *liens de cohérence* entre les différentes variables de décision.

— Les contraintes de **fenêtre de temps** correspondent au respect des fenêtres de temps d'atterrissage pour chaque avion $i \in \mathcal{A}$:

$$E_i \leq t_i \leq L_i. \tag{2.1}$$

— La **séparation** au seuil de piste garantit, pour chaque paire d'avions $(i,j) \in \mathcal{A} \times \mathcal{A}$: $i \neq j$, non seulement la séparation S_{ij} entre avions atterrissant sur une même piste ($z_{ij} = 1$), mais aussi la séparation s_{ij} (*diagonale*) entre les avions atterrissant sur deux pistes différentes ($z_{ij} = 0$) :

$$t_j \geq t_i + S_{ij} z_{ij} + s_{ij}(1 - z_{ij}) - M(1 - \delta_{ij}), \tag{2.2}$$

où M est une constante positive choisie suffisamment grande (*big-M*), contraintes clas-

siquement utilisée en recherche opérationnelle pour modéliser les contraintes logiques.

— Les contraintes de **séquencement** (2.3) assurent que pour chaque $(i, j) \in \mathcal{A} \times \mathcal{A} : i \neq j$, les variables de séquencement δ_{ij} sont bien définies, c'est-à-dire que l'avion i atterrit avant j ou l'inverse, mais pas les deux :

$$\delta_{ij} + \delta_{ji} = 1. \tag{2.3}$$

— L'**affectation d'une et une seule piste** d'atterrissage pour chaque avion $i \in \mathcal{A}$ est garantie par la contraintes :

$$\sum_{r \in \mathcal{R}} y_{ir} = 1. \tag{2.4}$$

— Les **liens de cohérence** entre les différentes variables de décision sont garantis par les contraintes suivantes :

$$\begin{aligned}
0 \leq & t_i^- \leq T_i - E_i, && (2.5)\\
0 \leq & t_i^+ \leq L_i - T_i, && (2.6)\\
& t_i^- \geq T_i - t_i, && (2.7)\\
& t_i^+ \geq t_i - T_i, && (2.8)\\
& t_i = T_i - t_i^- + t_i^+, && (2.9)\\
& z_{ij} \geq y_{ir} + y_{jr} - 1, && (2.10)\\
& z_{ij} = z_{ji}, . && (2.11)\\
& && (2.12)
\end{aligned}$$

pour chaque avion $i \in \mathcal{A}$ et chaque piste $r \in \mathcal{R}$.

Fonction-objectif

Le modèle de Beasley *et al.* [9] minimise la somme des déviations des dates cibles par rapport aux heures préférentielles d'atterrissage pour chaque avion : Ce coût de déviations est linéaire par morceau pour chaque avion, comme illustré dans la figure 2.4. Minimiser le coût total des retards est un cas particulier de cette fonction-objectif.

$$\min \sum_{i \in \mathcal{A}} (c_i^- t_i^- + c_i^+ t_i^+) \tag{2.13}$$

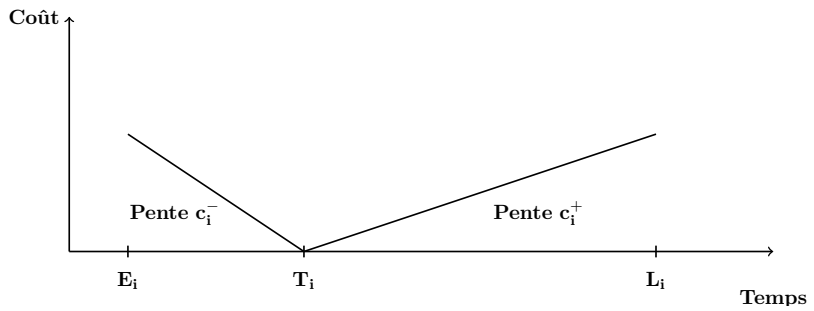


FIGURE 2.2 – Variation de la fonction coût du modèle de Beasley *et al.* [9], pour un avion $i \in \mathcal{A}$ ayant une fenêtre de temps $[E_i, L_i]$.

Résumé du modèle de Beasley *et al.* [9]

$$\min_{\delta,t,t^+,t^-,y,z} \quad \sum_{i\in\mathcal{A}} (c_i^- t_i^- + c_i^+ t_i^+)$$

$$\begin{aligned}
s.c \quad & E_i \le t_i \le L_i && i \in \mathcal{A} \\
& x_j \ge t_i + S_{ij} z_{ij} + s_{ij}(1 - z_{ij}) - M\delta_{ji} && (i,j) \in \mathcal{A}\times\mathcal{A} : i \ne j \\
& \delta_{ij} + \delta_{ji} = 1 && (i,j) \in \mathcal{A}\times\mathcal{A} : i < j \\
& \sum_{r\in\mathcal{R}} y_{ir} = 1 && i \in \mathcal{A} \\
& 0 \le t_i^- \le T_i - E_i && i \in \mathcal{A} \\
& 0 \le t_i^+ \le L_i - T_i && i \in \mathcal{A} \\
& t_i^- \ge T_i - t_i && i \in \mathcal{A} \\
& t_i^+ \ge t_i - T_i && i \in \mathcal{A} \\
& t_i = T_i - t_i^- + t_i^+ && i \in \mathcal{A} \\
& z_{ij} \ge y_{ir} + y_{jr} - 1 && (i,j) \in \mathcal{A}\times\mathcal{A} : i < j, r \in \mathcal{R} \\
& z_{ij} = z_{ji} && (i,j) \in \mathcal{A}\times\mathcal{A} : i < j \\
& t_i^-, t_i, t_i^+ \ge 0 && i \in \mathcal{A} \\
& \delta_{ij}, z_{ij}, y_{ir} \in \{0,1\} && i,j \in \mathcal{A} : i \ne j, \forall\ r \in \mathcal{R}
\end{aligned}$$

Cette formulation a une faible relaxation continue et atteint la valeur 0 si elle est implémentée telle quelle. Pour améliorer la valeur de la relaxation continue, plusieurs techniques sont proposées dans la littérature, comme par exemple :

— Adéquation de la valeur du *big-M*. Au lieu de choisir une (très grande) valeur arbitraire pour le paramètre M impliqué dans la modélisation des contraintes logiques (ici les contraintes de séparation (2.2)), il est recommandé de choisir M le plus petit possible, afin d'éviter les problèmes de convergence et adapté à chaque contrainte (ici, à chaque paire d'avions). On peut facilement montrer que cela est réalisable en choisissant :

$$M_{ij} = L_i + S_{ij} - E_j.$$

— Les stratégies de fixation de variables ou *Variable-Fixing Strategies*. Cette technique est introduite dans [18] (Lemme 1) ; elle consiste à fixer l'ordre de certains avions appartenant à la même classe (la catégorie de turbulence de sillage par exemple) suivant l'ordre lexicographique non décroissant de leurs fenêtres de temps. Si les contraintes CPS sont imposées, d'autres variables peuvent être fixées en se basant sur ces contraintes (voir par exemple, Prakash *et al.* [85]).

Beasley *et al.* [9] présentent aussi une liste d'inégalités supplémentaires valides (*Valid Inequalities*), qui peuvent être redondantes dans la formulation PLNE, mais s'avèrent utiles pour renforcer les relaxations continues.

2.5 Analogies et complexité

Le problème d'ordonnancement d'avions (sur une piste unique ou des pistes multiples) est lié à plusieurs problèmes d'optimisation combinatoire dont la complexité est bien connue, tels que le problème d'ordonnancement de tâches (*job-shop scheduling problem*), le problème de voyageur de commerce (*Traveling Salesman Problem* – TSP) et le problème de tournées de véhicules (*Vehicle Routing Problem* – VRP). Nous détaillons dans la suite ces analogies qui nous permettront de déduire la complexité NP-difficile du problème considéré. Nous présentons également des variantes polynomiales qui ont été considérées dans la littérature.

2.5.1 Première analogie : Problème d'ordonnancement de tâches

La relation entre un problème d'ordonnancement de tâches et un problème d'ordonnancement d'atterrissages d'avions (ALP) basique a été décrite dans [9]. Dans ce contexte, le lien se dessine comme suit :

— les **pistes** d'atterrissage représentent les **machines** ;

— les **avions** à séquencer représentent les **tâches** à ordonnancer ;

— la **séparation** de turbulence de sillage entre deux atterrissages successifs i et j représente la **somme de la durée de la tâche** i – *processing time* (occupation de la piste) et du **temps de réglage** – *set-up time* (temps d'inactivité) entre la tâche i et j (figure 2.3).

Selon cette analogie, l'ALP est alors équivalent à un problème d'ordonnancement de tâches avec des durées des tâches dépendantes de la séquence – *sequence-dependent processing times* (mais temps des réglages égaux à zéro), ou un problème d'ordonnancement de tâches avec des temps de réglages dépendant de la séquence – *sequence-dependent set-up times* (durées des tâches égales à zéro).

L'heure d'atterrissage au plus tôt (respectivement au plus tard) habituellement considérée dans l'ALP est interprétée comme la date d'arrivée ou de disponibilité de la tâche – *release date* (respectivement la date d'échéance –*due date*). Une fonction-objectif typique minimise l'heure d'atterrissage du dernier avion de la séquence (*makespan*). Dans ce cas, le problème

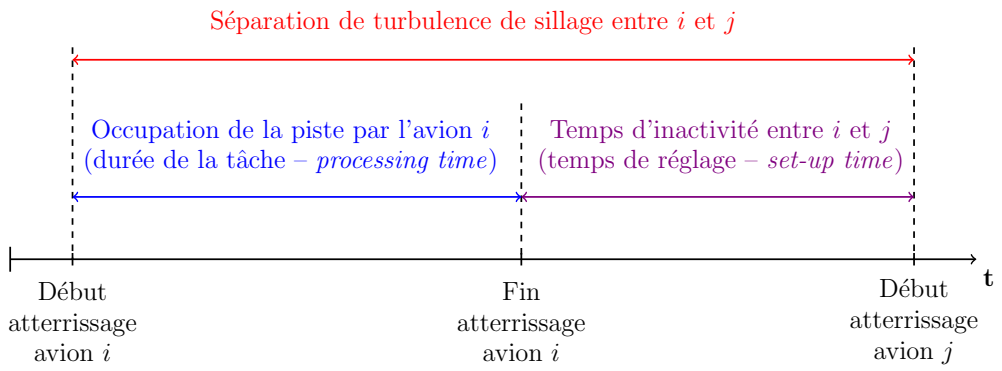


FIGURE 2.3 – Analogie entre évènements sur la piste et contrainte de séparation dans [9].

d'ordonnancement des atterrissages (avec une seule piste d'atterrissage) se réduit au problème de voyageur de commerce (*Traveling Salesman Problem* – TSP) qui est NP-difficile [70], ce qui démontre donc que l'ALP est aussi NP-difficile.

Dans un contexte plus réaliste, Bianco *et al.* [15] montrent que le problème d'ordonnancement d'atterrissages et de décollages d'avions, appelé *Aircraft Sequencing Problem* (ASP) dans la littérature, prenant en considération la configuration des pistes ainsi que la structure des routes de départ et d'arrivée est analogue à un problème de *no-wait job-shop*. Selon cette analogie, chaque route d'approche (ou de départ) est décomposée en plus petits segments de vol, d'une longueur de 5 miles nautiques environ ; ces segments de vol en plus des pistes (d'atterrissage et de décollage) correspondent à des machines. Les avions comme auparavant correspondent aux tâches à ordonnancer. Dans ce contexte, les séparations de turbulence de sillage correspondent aux temps de réglages dépendants de la séquence (*sequence-dependent set-up time*). La durée d'une tâche i est considérée constante et ne dépend que de la machine (piste ou segment de vol) et de la tâche elle-même. Les auteurs supposent que chaque avion a des heures d'atterrissage et de décollage au plus tôt, qui correspondent comme auparavant aux dates de disponibilité des tâches.

2.5.2 Seconde analogie : problème de tournées de véhicules

L'analogie entre un problème d'ordonnancement des atterrissages d'avions et le problème de tournées de véhicules est décrite dans [18]. Dans cet article, le lien se dessine comme suit :

- les **pistes** d'atterrissage correspondent aux **véhicules** à déployer ;
- les **avions** à séquencer correspondent aux **clients** à servir ;
- la **séparation** de turbulence de sillage entre deux avions successifs i et j représente la **distance** entre deux clients, qui est, dans ce contexte, asymétrique.

Les dates préférentielles d'atterrissage correspondent aux heures auxquelles les clients préfèrent être servis. Les bornes inférieures et supérieures sur ces dates préférentielles correspondent aux contraintes classiques des fenêtres de temps. La fonction-objectif vise alors à

servir (faire atterrir) chaque client (avion) dans sa fenêtre temporelle de manière à minimiser le coût total des déviations par rapport aux heures préférentielles.

Selon cette analogie, l'ALP mettant en jeu des pistes multiples indépendantes est alors équivalent à un problème de tournée de véhicules sans contraintes de capacité. Dans le cas de pistes multiples et hétérogènes, on peut considérer que la flotte de véhicules est aussi hétérogène, et que certains clients préfèrent être servis par certains types de véhicules. Cependant, le cas de pistes interdépendantes n'a pas d'analogie directe avec le problème de tournées de véhicules [63]. Par ailleurs, si l'on considère qu'on ne dispose que d'un seul véhicule, le problème est alors équivalent au problème de voyageur de commerce avec des contraintes de fenêtres de temps.

2.5.3 Variantes polynomiales

En conséquence des analogies présentées ci-dessus, on peut déduire que mêmes les variantes les plus simples présentées ci-dessus du problème d'ordonnancement d'avions sont des problèmes NP-difficiles. Néanmoins, certaines autres variantes, considérées dans la littérature peuvent être résolues en un temps polynomial :

— Le **problème d'ordonnancement d'atterrissages sur une piste unique, sous le « changement de position contraint »**. Rappelons que sous cette contrainte, un avion n'est autorisé à être dévié de sa position initiale dans la séquence « premier arrivé, premier servi » (FCFS) que d'un nombre fixe de positions. En imposant cette contrainte, Balakrishnan et Chandran [6] ont su proposer des algorithmes de programmation dynamique de complexité polynomiale. En effet, pour n avions et un nombre maximal, noté p, de changements de positions, la complexité de l'algorithme proposé est $O(n(2p+1)^{2p+2}))$, qui est cependant exponentielle en p.

Cette contrainte a deux avantages : l'un est théorique car elle réduit le nombre de séquences admissibles et réduit aussi la compléxité du problème. L'autre avantage est pratique, car elle garantit une certaine équité entre les avions.

Dans la littérature, les travaux de recherche considérant cette contrainte sont nombreux (par exemple : [5, 6, 85]). Nous faisons une revue complète de ces articles dans le prochain chapitre.

— Le **problème d'ordonnancement d'atterrissages sur une piste unique en regroupant les avions dans des classes** [18]. Cette variante suppose que les avions peuvent être regroupés dans des classes (selon leurs catégories de turbulence de sillage par exemple) et que les avions appartenant à une même classe sont similaires, c'est-à-dire qu'ils ont le même coût de retard. Briskorn et Stolletz [18] montrent alors que sous cette condition, les avions appartenant à la même classe peuvent être séquencés selon la règle FCFS. Ils proposent alors des algorithmes qui sont polynomiaux en le nombre total d'avions, noté n, mais exponentiel en le nombre total de classes, noté C. En effet, pour une variante simple de l'ALP impliquant une seule piste et une seule classe, l'algorithme proposé a une complexité $O(n^3)$. Cependant, si l'on considère trois classes comme dans les catégories de turbulence de sillage H, M et L, l'algorithme pourrait ne pas être applicable en pratique, car de complexité $O(n^{23})$ ([18], Théorème 4). D'autres

articles dans la littérature considèrent également des contraintes de classes d'avions, comme par exemple [71, 72].

Chapitre 3

Revue des méthodes de résolution du problème d'ordonnancement d'avions

Résumé du chapitre

Ce chapitre présente notre première contribution de la thèse. Il s'agit d'une revue complète des méthodes de résolution les plus pertinentes de la littérature pour le problème d'ordonnancement d'avions (ALP, ATP ou ASP). Il reprend notre article intitulé *The aircraft runway scheduling problem : A survey*, publié dans la revue scientifique *Computers & Operations Research*. Nous nous concentrons principalement sur les contributions récentes (de 2010 à aujourd'hui). Pour les contributions antérieures à 2010, nous orientons le lecteur vers la revue de Bennell *et al.* [11].

Sommaire

Pour chaque article étudié, nous présentons d'abord son contexte, c'est-à-dire le problème traité, les contraintes impliquées et l'objectif considéré. Ensuite, nous étudions la méthode de résolution choisie. Puis nous évoquons les instances sur lesquelles la méthode a été testée, et à la fin nous discutons des résultats. Notre revue de littérature couvre non seulement les méthodes exactes et les métaheuristiques, mais également deux nouvelles approches basées sur l'apprentissage par renforcement, qui semblent être prometteuses pour les problèmes d'ordonnancement d'avions. En outre, nous proposons une étude comparative de certaines approches exactes, où nous montrons comment la plupart des instances de référence de la littérature sont résolues facilement avec les versions actuelles des logiciels d'optimisation. Par

conséquent, nous proposons de nouvelles instances construites à partir de données réelles de l'aéroport de Paris-Orly.

3.1 Tour d'horizon des méthodes exactes pour l'ordonnancement d'avions

Plusieurs types de méthodes exactes sont proposées dans la littérature ; la majorité sont soit des méthodes basées sur l'optimisation linéaire mixte en nombres entiers – PLNE (*Mixed-Integer Linear Programming*), soit des approches de programmation dynamique (*Dynamic Programming* - DP). Remarquons qu'on peut trouver des travaux qui utilisent la programmation par contraintes comme approche de résolution. Néanmoins, ils restent très rares comparés aux approches basées la PLNE ou la DP. Pour cette raison, nous nous concentrons sur ces deux dernières approches dans cette section, et nous orientons le lecteur intéressé à consulter [2] pour une revue de littérature sur les travaux utilisant la programmation par contraintes pour des problèmes généraux de gestion de trafic aérien.

Nous présentons dans la table 3.1 un aperçu des différentes approches exactes étudiées dans cette section. Dans cette table, nous utilisons les abréviations Prec., Pr., Sep. et TW pour désigner respectivement les contraintes de précédence (*Preceeding constraints*), *Prunining*, les contraintes de séparation et les fenêtres de temps (*Time Windows*).

3.1.1 Programmation dynamique

Briskorn et Stolletz [18] considèrent le problème d'ordonnancement d'atterrissages d'avions (ALP) sur des pistes indépendantes, sous les contraintes de séparation et des fenêtres de temps, et dans l'objectif de minimiser la somme des déviations par rapport aux heures cibles. Tout d'abord, ils font l'hypothèse de *classe d'avions*, c'est-à-dire les avions peuvent être regroupés dans des classes (catégories de turbulence de sillage par exemple). Ils démontrent ensuite qu'il est optimal de planifier les avions appartenant à la même classe selon l'ordre « premier-arrivé, premier-servi » (FCFS). L'ALP est modélisé comme un problème de plus court chemin dans un graphe, et résolu à l'aide de la programmation dynamique. Dans le cas général multi-classes et multi-pistes, la variable d'état de la DP est le nombre d'avions déjà planifiés dans chaque classe et le profil d'occupation de pistes (*runway-occupation profile*) du dernier avion sur chaque piste, c'est-à-dire sa date et classe d'avions. L'algorithme proposé est démontré polynomial en le nombre total d'avions, noté N, mais exponentiel en le nombre de classes, noté, W : $O(RW^{R+1}N^{(R+1)W+R+1})$, où R est le nombre de pistes. Cependant, il n'a pas été implémenté, mais les auteurs ont adapté le modèle PLNE de [9] à leur fonction-objectif qui dépend de chaque classe d'avions, et concluent que le temps de calcul peut être diminué sous les hypothèses de classes d'avions et de fenêtre de temps ordonnées.

Méthode exacte	Source	Problème	Nb de pistes, configuration	Contraintes opérationnelles	Objectif	Modèle	Approche de résolution	Instances (Nb d'avions)
DP	Balakrishnan et Chandran [5]	ASP	1	CPS, Prec., Sep, et TW	min $\sum$ retards, min max retard	Graphe orienté	DP + CPS	Générées (10 à 50)
	Balakrishnan et Chandran [6]	ALP	1	CPS, Prec., Sep, et TW	min makespan	Graphe orienté	DP + CPS	Aéroport international de Denver (11 à 23), générées (10 à 50)
	Rathinam *et al.* [87]	ATP	1	Prec., Sep. et TW	min $\sum$ retards	Optimisation multi-objectifs	DP	Générées (6 à 40)
	Ravidas *et al.* [88]	ATP	2, indépendantes	Prec., Sep. et TW	min $\sum$ retards	Optimisation multi-objectifs	DP	Générées (8 à 25)
	Montoya *et al.* [80]	ATP	1	CPS, Prec., Sep. et TW	Multi-obj : min $\sum$ retards, max capacité piste	Problème de permutation	DP	Basée sur l'aéroport DFW (16 à 32)
	Bennell *et al.* [12]	ALP	1	Sep. et TW	min makespan + temps atterrissage moyen + carburant + violation fenêtre de temps	Files d'attente	DP	Aéroport de Londres Heathrow (21, 42 et 84), générées (selon des scénarios)
	Briskorn et Stolletz [18]	ALP	$\geq$ 1, indépendantes	Sep. et TW	min $\sum$ déviations	Graphe orienté + ELW, PLNE + ELW	DP, CPLEX	Bianco *et al.* [13] (30 et 44)
	Faye [33]	ALP (séquence fixée)	1	Sep. et TW	min $\sum$ déviations	LP	DP, FICO-Xpress	OR-library (10 à 500)
	Lieder *et al.* [71]	ALP	$\geq$ 1, indépendantes	Sep. et TW	min $\sum$ retards	PLNE + ELW, Graphe orienté + ELW + règle de dominance	CPLEX, DP	Bianco *et al.* [13] (30 et 44), Générées (50 à 100)
	Lieder et Stolletz [72]	ASP	$\geq$ 1, interdépendantes hétérogènes	Sep. et TW	min $\sum$ retards	PLNE + ELW, Graphe orienté + ELW + critère de dominance	CPLEX, DP	Générées (30 à 62)
PLNE	Faye [34]	ALP	$\geq$ 1, interdépendantes	Sep. et TW	min $\sum$ déviations	IP + time discretization	Matrix approx. + FICO Xpress	OR-library (10 à 44)
	Furini *et al.* [36]	ASP	1	Sep. et TW	min $\sum$ retards	PLNE	CPLEX + RH	OR-group Bologna (60)
	Malik *et al.* [76]	ASP	$\geq$ 1, interdépendantes	CPS, Prec., Sep. et TW	min $\sum$ retards	PLNE	GuRoBi	Basé sur l'aéroport CLT (10 à 35)
	Avella *et al.* [4]	ASP	1	Sep. et TW	min $\sum$ retards	BILP + VI	VFS + génération de colonnes + CPLEX	Aéroport de Stockholm (33 et 40), Aéroport de Hamburg (57, 58 et 72), OR-group Bologna (60)
	Ghoniem et Farhadi [39]	ALP	$\geq$ 1, indépendantes	Sep. et TW	min $\sum$ retards	PLNE + valid inequalities	CPLEX	OR-library (10 à 500), générées (15 à 25)
	Hancerliogullari *et al.* [44]	ALP	$\geq$ 1, indépendantes	Sep.,TW et équilibre de trafic	min $\sum$ retards	PLNE	Solver non-spécifié	Générées (15 à 25)
	Kim *et al.* [66]	ALP	$\geq$ 1, indépendantes	Sep. et TW	min $\sum$ emissions	PLNE	Solver non-spécifié	Aéroport métropolitain de Détroit (3 à 40)
	Prakash *et al.* [85]	ASP	1	CPS, Sep. et TW	min makespan	PLNE	GuRoBi + CPS + data-splitting	Générées (30 à 60)
	Pohl *et al.* [84]	ASP	$\geq$ 1, indépendantes	Sep. et TW	min retards	PLNE	GuRoBi + VI + Pr + Starting solution	Aéroport de Munich (30, 40, 60 et 75), OR-library (100 à 500)

TABLE 3.1 – Résumé des méthodes exactes de la littérature pour le problème d'ordonnancement d'avions.

Lieder *et al.* [71] utilisent le même modèle de programmation dynamique que [18], en considérant les mêmes contraintes, mais pas le même objectif. En effet, dans [71], il est question de minimiser le retard total, qui est un cas particulier de l'objectif considéré dans l'article de Briskorn et Stolletz [18]. Les auteurs de [71] développent des règles de dominance entre les états (*dominance criteria*) qui réduisent considérablement l'espace d'états, ce qui leur a permis d'implémenter efficacement l'algorithme de programmation dynamique de [18]. Les résultats montrent le bénéfice d'utiliser cette approche par rapport à une approche PLNE standard. Lieder et Stolletz [72] étendent ensuite cette méthode pour le cas le plus général : pistes hétérogènes et interdépendantes.

Un autre cadre de programmation dynamique est proposé par Balakrishnan et Chandran [5, 6] pour l'ALP, qui considère, en plus de la *séparation* et de la fenêtre de temps, le *changement de position contraint*. L'idée est encore ici de modéliser le problème comme un problème de plus court chemin dans un graphe orienté et de le résoudre à l'aide de la programmation dynamique. Le graphe est cette fois constitué de N étapes (N est le nombre total d'avions). Chaque étape représente la position des avions dans la séquence finale. Ces étapes sont constituées de nœuds, et chaque nœud représente une sous-séquence d'avions. Plusieurs fonctions-objectif sont considérées. Dans [6], la récurrence de la programmation dynamique vise à trouver la séquence de longueur minimale, ce qui revient à maximiser le rendement de la piste. D'autres objectifs sont étudiés dans [5], comme la minimisation du retard moyen et la minimisation du retard maximal.

Des algorithmes de programmation dynamique multi-objectif sont aussi proposés dans la littérature pour l'ATP [80, 87, 88] et l'ALP [12]. Dans [87], les auteurs considèrent l'ATP mettant en jeu une seule piste, avec l'objectif de minimiser la somme des retards. Ils reformulent ce problème comme un nouveau problème d'optimisation multi-objectif minimisant la somme des retards et la date du dernier avion décollé. La variable d'état de leur algorithme est une sous-séquence composée des indices d'avions et les dates de décollage correspondantes. Il est prouvé que l'algorithme proposé fournit l'ensemble des solutions non-dominées par rapport aux deux objectifs mentionnés ci-dessus. Les tests numériques sont effectués sur des instances générées artificiellement, impliquant 4 à 40 avions. Leurs résultats montrent que cet algorithme trouve les solutions optimales dans des temps de calculs très courts, ce qui a encouragé l'extension de l'approche au cas deux pistes indépendantes dans [88]. Le même problème est considéré dans Montoya *et al.* [80], mais les auteurs ajoutent en plus les contraintes CPS. L'algorithme proposé calcule les solutions Pareto-optimales par rapport aux deux objectifs mentionnés ci-dessus. Les tests numériques sont effectués sur des instances générées artificiellement, en imitant des scénarios réalistes de l'aéroport de Dallas - Fort Worth. L'algorithme est ensuite comparé à une heuristique de référence ; les auteurs montrent que les solutions extrêmes calculées par leur algorithme surpassent celles données par l'heuristique de base en terme de réduction de retard.

Bennell *et al.* [12] étudient l'ALP avec une piste unique. Les trois objectifs considérés sont la ponctualité, les coûts du carburant et la maximisation de la capacité de la piste. Étant donné la complexité du problème (rappelons que ce problème est NP-difficile), ils optent pour une somme pondérée de ces trois objectifs. Dans leur algorithme, la variable d'état est similaire dans sa définition à celle de [18] : c'est un tuple qui contient le nombre d'avions déjà atterris de chaque catégorie de turbulence de sillage, ainsi que la catégorie et la date

d'atterrissage du dernier avion. L'algorithme proposé est testé non seulement pour le cas statique, où tous les paramètres sont disponibles au début de l'optimisation, mais aussi pour le cas dynamique dans une approche d'horizon glissant. Les tests numériques sont effectués sur des instances générées artificiellement, mais aussi sur des instances construites à partir de trafic réel. Les auteurs ont comparé leur approche à la performance d'un contrôleur aérien sur la base de cinq indices de performance. Les résultats montrent que, pour le cas statique, leur algorithme présente des performances similaires à celles d'un contrôleur aérien en terme de capacité de piste. Cependant, la programmation dynamique atteint de meilleurs résultats sur les autres indices de performance, en particulier sur la consommation de carburant. Cet écart s'explique par le fait que les contrôleurs aériens visent surtout à maximiser la capacité de la piste et ne prennent pas forcément en compte les multiples autres objectifs considérés par Bennell *et al.* [12].

3.1.2 Programmation mixte en nombres entiers

Faye [34] considère le problème d'ordonnancement d'atterrissages (ALP) sur des pistes interdépendantes, dans l'objectif de minimiser les déviations par rapport aux dates préférentielles d'atterrissage. En supposant que les paramètres du problème sont entiers, l'auteur démontre que la solution est également entière. Il modélise ensuite le problème sous forme d'un problème d'optimisation linéaire avec variables binaires (0-1 *linear program*), construit sur la base de scénarios (heure d'atterrissage et affectation des pistes). Pour résoudre ce problème, l'auteur propose un algorithme appelé *dynamic constraints generation algorithm*. Ce dernier se décompose en deux blocs : le premier bloc résout un problème relâché associé au problème d'optimisation linéaire avec variables binaires mentionné ci-dessus qui n'est pas la relaxation linéaire classique : cette relaxation repose sur l'approximation de la matrice de séparation par une matrice de rang 2, de telle sorte que chaque entrée de la matrice de rang 2 soit inférieure ou égale à chaque entrée de la matrice de séparation. Le deuxième bloc détecte les paires d'avions qui violent la séparation en raison de la relaxation, puis les corrigent. Le deuxième bloc se termine par une solution du problème relâché qui satisfait toutes les contraintes de séparation, ce qui fournit une solution optimale. Les tests numériques sur les instances de la *OR-Library* montrent que cette approche donne de meilleurs résultats en terme de temps de calcul que l'approche de Beasley *et al.* [9] (section 2.4).

Furini *et al.* [36] présentent une formulation MILP pour l'ordonnancement simultané d'atterrissages et de décollages (ASP) sur une piste unique, pour minimiser le retard total pondéré. Les variables binaires de ce modèle mathématique sont utilisées pour indiquer la position d'un avion dans la séquence à la piste, contrairement à ce qui est fréquent dans la littérature, où les variables binaires indiquent plutôt les positions relatives entre deux avions. Le modèle est résolu par CPLEX avec une approche d'horizon glissant, pour trouver des solutions améliorées (pas nécessairement optimales) par rapport à l'heuristique FCFS. Plus tard, dans Furini *et al.* [37], comparent ce modèle à celui de Beasley *et al.* [9] sur certaines instances générées, et en imposant une limite de temps pour CPLEX. Les résultats montrent que la formulation de [9] est plus performante que celle de [36], tant en termes de temps de calcul que de qualité des solutions obtenues dans la limite de temps prédéfinie. Cependant, l'approche de recherche tabou qu'ils développent surpasse les résultats obtenus par CPLEX sur le modèle. Cette dernière approche sera étudiée dans la section 3.2.

Hancerliogullari *et al.* [44] proposent une formulation PLNE pour l'ordonnancement d'avions à l'atterrissage et au décollage (ASP). Ce modèle diffère de celui de [9] sur deux aspects : tout d'abord, les pistes sont supposées indépendantes, c'est-à-dire que les contraintes de séparation pour les avions atterrissant sur deux pistes différentes sont supposées être automatiquement satisfaites, tandis que le modèle [9] prend en compte des exigences de séparation supplémentaires (séparation *diagonale*) pour les avions qui atterrissent sur des pistes différentes. Le second aspect est lié à l'équilibre du trafic sur les pistes disponibles, que les auteurs intègrent dans le modèle, en ajoutant des contraintes de bornes inférieures et supérieures sur le nombre d'avions affectés à une piste. Les auteurs rapportent des résultats en termes de temps de calcul pour résoudre le MILP proposé (solveur non spécifié) sur des instances congestionnées, générés artificiellement, allant de 15 à 25 avions et impliquant 2 à 5 pistes. Les temps de calcul étant importants pour presque toutes les instances générées, les auteurs ont eu recours à des heuristiques que nous étudierons dans la section 3.2.

Kim *et al.* [66] proposent une nouvelle approche PLNE pour ordonnancer les avions sur la piste, en prenant en considération des points repères de l'espace aérien entourant un aéroport (*TRACON*[1] *fixes*). Ces points repères incluent les points d'entrée dans la zone terminale et des points de sortie de cette zone. Le modèle calcule l'affectation de piste et de point repère de la zone terminale pour chaque avion, ainsi que des dates cibles d'atterrissage (ou décollage) et de passage par ces points. L'objectif est de minimiser les émissions de carburant dans cette zone, y compris les émissions dans les voies de circulation (*taxiways*). Notons que cette fonction-objectif est rarement considérée dans la littérature d'ordonnancement d'avions. Au moyen de tests numériques sur des données de l'aéroport métropolitain de Détroit, les auteurs montrent que leur approche permet non seulement de réduire les émissions de $29,9\%$, mais aussi d'augmenter la capacité des pistes, car elle rééquilibre le trafic entre les pistes et les différents points repère de la TMA.

Ghoniem et Farhadi [39] adaptent la formulation de [9] pour y incorporer les atterrissages (ASP). Ils y ajoutent ensuite des inégalités valides pour renforcer la formulation. Par le biais de tests numériques sur la *OR-library* et sur des instances générées, les auteurs montrent que même en ajoutant ces inégalités, les temps de calcul restent élevés pour résoudre des instances modestes (20 avions). Pour cette raison, ils proposent un nouveau modèle (*set-partitioning model*) et le résolvent avec la génération de colonnes, ce qui leur permet de résoudre des instances plus grandes en des temps de calcul raisonnables.

Malik *et al.* [76] proposent une approche PLNE pour planifier simultanément les atterrissages, les décollages et les traversées de la piste, pour l'aéroport de Charlotte Douglas (CLT). La particularité de ce travail est qu'il prend en compte, en plus des contraintes classiques du CPS, de séparation et de fenêtres de temps, des contraintes spécifiques liées à la configuration de l'aéroport considéré. Un exemple de telles contraintes est la fixation d'un ordre FCFS pour les avions partants vers (ou arrivant à) un même point de référence (*metering fix*). Une autre particularité est liée à la séparation de turbulence de sillage utilisée : il s'agit de la matrice de séparation réduite des projets RECAT introduits en section 1.2.2. Les tests numériques sur des instances de l'aéroport de Charlotte Douglas montrent que, en comparaison avec la FCFS, leur approche améliore le retard total de la séquence. En revanche, les temps de calcul deviennent très importants à partir de la valeur 3 pour le nombre maximal de changements

1. TRACON : *Terminal Radar Approach CONtrol*, c'est l'équivalent aux États-Unis de la TMA en Europe.

de position contraints.

Avella *et al.* [4] optent pour une formulation linéaire en nombres entiers pour l'ASP, prenant en compte les deux contraintes classiques de séparation et de fenêtre de temps. L'objectif est de minimiser le retard total. Ils proposent ensuite un algorithme exact pour résoudre leur modèle, qui s'appuie sur des stratégies de fixation de variables et de resserrage de fenêtres de temps. Plusieurs jeux de données sont utilisées par les auteurs pour valider leur algorithme : (i) instances de l'aéroport de Stockholm avec 33 et 40 avions, (ii) instances de l'aéroport de Hamburg avec 57, 58 et 72 avions, et (iii) les instances de la OR-group Bologna, chacune avec 60 avions. Ils comparent ensuite la performance de leur algorithme à celle de CPLEX (sans inégalités valides ni fixation de variables), en terme de qualité des solutions obtenues dans une limite de temps de calcul de 600 secondes. Ils concluent que leur algorithme est supérieur à CPEX, tant en terme de temps de calcul que de nombre d'instances résolues dans la limite de temps prédéfinie.

Récemment, Prakash *et al.* [85] ont proposé une approche basée sur un MILP pour ordonnancer simultanément les atterrissages et les décollages (ASP) sur une piste unique. En plus des contraintes de séparation et de fenêtres de temps, les auteurs considèrent les contraintes de changements de positions et les incorporent explicitement dans le modèle. À notre connaissance, c'est la première fois dans la littérature que ces contraintes de CPS sont intégrées explicitement dans un modèle PLNE. La fonction-objectif minimise la date du dernier avion posé. L'approche proposée est appelée *data-splitting method*. Elle est basée sur la subdivision de la séquence originale (donnée par la règle du FCFS) en toutes les paires possibles de sous-séquences d'avions respectant les contraintes du CPS, puis la résolution indépendante de chaque paire de sous-problèmes correspondants en utilisant GuRoBi. Pour les tests numériques, les auteurs génèrent artificiellement des instances très congestionnées comportant 30 à 40 avions. Les résultats de cette étude montrent que cette approche surpasse les méthodes de programmation dynamique de la littérature en terme de temps de calcul.

Encore plus récemment, Pohl *et al.* [84] ont aussi opté pour une approche PLNE pour la planification simultanée des atterrissages, des décollages et des déneigements des pistes en période hivernale. Ils prennent en considération les contraintes de fenêtres de temps et de séparation entre les différentes opérations sur la piste (atterrissages, décollages, déneigements). La résolution directe de leur PLNE conduit à des temps de calcul très longs (au-delà d'une heure). Pour cette raison, ils effectuent un pré-traitement et ajoutent des inégalités valides (*valid inequalities*) pour réduire l'espace de recherche. De plus, ils construisent des solutions réalisables qu'ils intègrent au solver GuRoBi comme solution de départ. Les tests numériques sont effectués sur des instances réelles de l'aéroport de Munich, impliquant 30, 45, 60 et 75 avions, et 2 ou 3 pistes. Les auteurs concluent qu'avec le pré-traitement, les inégalités valides et la solution de départ pour GuRoBi, les temps de calcul sont réduits, et des solutions optimales sont obtenues en 60 secondes pour la plupart des instances. De plus, leur algorithme est plus performant – en terme de retard total – que l'heuristique utilisée par les contrôleurs de l'aéroport de Munich. Cependant, les tests supplémentaires sur les instances de la OR-library montrent que, pour certaines instances (par exemple `airland10,airland11`), les temps de calcul restent longs pour une application en temps réel.

3.1.3 Instances du problème et tests numériques

Dans cette sous-section, nous présentons une étude comparative de deux modèles MILP connus de la littérature, à savoir la formulation la plus citée, celle de Beasley *et al.* [9] et le modèle de Furini *et al.* [37]. Nous faisons référence à ces deux modèles dans la suite avec les acronymes des auteurs : BKSA pour [9] et FKPT [37]. Nous avons choisi ces deux modèles car ils représentent deux formulations mathématiques différentes : BKSA est modélisée comme un problème d'ordonnancement classique, où les variables binaires représentent l'ordre relatif entre deux avions, tandis que FKPT est un problème d'affectation où les variables binaires indiquent la position d'un avion dans la séquence. Nous testons ces deux modèles sur des benchmarks de la littérature, à savoir :

— Les instances de la *OR-library* [7], qui sont disponibles en ligne : `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`.

— Les instances du groupe *OR-group Bologna* [37], aussi disponibles en ligne : `http://or.dei.unibo.it/`.

Tous les tests numériques sont effectués sur un ordinateur personnel sous le système GNU/Linux, processeur Intel(R) Core(TM) i7-4700M avec 8 Go de RAM. Les modèles sont implémentés avec Docplex, l'interface Python du solver CPLEX, version 12.8.

Les résultats sont présentés dans la table 3.2. La première colonne indique l'origine des instances, la deuxième colonne présente le nom de l'instance testée. Les troisième et quatrième colonnes indiquent respectivement le nombre total d'avions, $|\mathcal{A}|$, et de pistes, $|\mathcal{R}|$, impliqués. Les autres colonnes donnent les temps de calcul, en secondes, pour trouver une solution optimale pour chacune des deux formulations considérées. Nous imposons une limite de temps sous CPLEX de 300 secondes (5 minutes) pour chaque test. Pour la formulation FKPT [37], le symbole '-' indique simplement que les tests impliquant plusieurs pistes ne sont pas pertinents, puisque ce modèle n'est pas adapté à ce cas.

La table 3.2 montre que, pour les instances de la *OR-library* impliquant 10 à 50 avions et pour les instances du *OR-group Bologna*, la formulation BKSA parvient à trouver des solutions optimales en des temps de calculs très courts, contrairement au modèle FKPT qui nécessite des temps de calcul longs, même pour les petite instances impliquant $|\mathcal{A}| = 15$ avions. Nous pouvons en déduire qu'avec une bonne formulation PLNE et les versions actuelles des solveurs (CPLEX 12.8 par exemple), même les grandes instances de la littérature impliquant jusqu'à 60 avions peuvent être résolues de façon optimale en des temps de calcul raisonnables. Les seuls instances difficiles sont les très grandes, impliquant 100 à 500 avions.

Pour cette raison, nous construisons 4 ensembles de données (*data sets*) qui serviront de base pour générer 12 nouvelles instances pour l'ALP (instances Ikli). Ces ensembles de données, les 12 instances ainsi que les implémentations de cette sous-section, sont disponibles à l'adresse `http://data.recherche.enac.fr/ikli-alp/`.

La construction de ces instances est expliquée dans la section 4.3, dans laquelle nous fournissons également des informations détaillées sur chaque instance, ainsi que sur chacune des quatre base de données. Les résultats des deux formulations BKSA et FKPT sur nos 12 instances sont présentés dans la table 3.2. On observe que seules quelques instances sont résolues à l'optimalité dans des temps de calcul courts, à savoir : `alp_11_30` avec deux

pistes, `alp_19_30` avec une et deux pistes, et `alp_19_40` avec deux pistes. Aucun des modèles BKSA et FKPT ne conduit à la résolution des instances restantes dans des délais de calcul raisonnables.

3.2 Tour d'horizon des méthodes stochastiques

La nature dynamique du problème d'ordonnancement d'avions motive le recours à des méthodes de résolution rapide qui permettent la mise à jour des solutions facilement quand un nouvel évènement se produit. Plusieurs chercheurs privilégient ainsi les heuristiques et les métaheuristiques. Dans la littérature antérieure à 2010, les algorithmes génétiques étaient fréquemment utilisés [8, 51, 52, 53, 54]. Récemment, la recherche tabou (*Tabu Search* – TS) et le recuit simulé (*Simulated Annealing* – SA) ont attiré beaucoup plus d'attention et sont devenus les métaheuristiques les plus utilisées [37, 43, 44, 74, 89, 93, 97]. D'autres métaheuristiques sont aussi utilisées dans la littérature, telles que l'optimisation par colonies de fourmis (*Ant Colony Optimization* – ACO) [10, 61, 103] et la recherche à voisinage variable (*Variable Neighborhood Search* – VNS) [93, 94].

La table 3.3 résume les approches stochastiques les plus pertinentes étudiées dans cette section, y compris les métaheuristiques, heuristiques et nouvelles approches basées sur l'apprentissage par renforcement (Reinforcement Learning – RL).

3.2.1 Algorithmes génétiques

Hu et Paolo [54] considèrent le problème d'ordonnancement d'avions à l'atterrissage (ALP) en mettant en jeu une seule piste, dans l'objectif de minimiser le retard total. Ils introduisent un nouveau cadre de représentation de *chromosomes* pour les algorithmes génétiques, qui diffère des représentations classiques basées sur les permutations d'avions ([52, 53]). En effet, la représentation de [54] consiste à projeter une séquence candidate sur un espace artificiel, où la représentation d'une solution est basée sur des valeurs numériques (points) plutôt que sur une permutation, puis à relier tous ces points pour former une séquence d'atterrissage. L'espace artificiel choisi est un espace bidimensionnel : l'axe des abscisses représente le temps, tandis que l'axe des ordonnées représente la catégorie de turbulence de sillage d'un avion, convertie à la même unité de temps de l'axe des abscisses, en utilisant les normes de séparation relatives aux turbulences de sillage (Table 1.1). Des tests numériques approfondis sont effectués sur des instances générées artificiellement pour trois scénarios de congestion (*sous-congestionné*, *normal* et *congestionné*) et contenant chacun des instances impliquant jusqu'à 60 avions. Les résultats démontrent une performance similaire à d'autres approches classiques d'algorithmes génétiques pour les deux scénarios *sous-congestionné* et *normal*, mais une meilleure performance de [54] pour les instances du scénario *congestionné*.

	Instance	$\|\mathcal{A}\|$	$\|\mathcal{R}\|$	Temps CPU (s)	
				BKSA [9]	FKPT [37]
OR-library	Airland1	10	1	0.05	43.05
			2	0.03	-
	Airland2	15	1	0.16	63.05
			2	0.07	-
	Airland3	20	1	0.07	> 300
			2	0.12	-
	Airland4	20	1	6.69	> 300
			2	154.90	-
	Airland5	20	1	20.38	> 300
			2	67.36	-
	Airland6	30	1	0.03	> 300
			2	29.16	-
	Airland7	44	1	1.46	> 300
			2	1.08	-
	Airland8	50	1	0.9	> 300
			2	3.66	-
	Airland9	100	1	> 300	> 300
			2	> 300	-
OR-group Bologna	FPT01	60	1	0.23	> 300
	FPT02	60	1	0.27	> 300
	FPT03	60	1	0.24	> 300
	FPT04	60	1	0.45	> 300
	FPT05	60	1	0.2	> 300
	FPT06	60	1	0.2	> 300
	FPT07	60	1	0.33	> 300
	FPT08	60	1	0.33	> 300
	FPT09	60	1	0.28	> 300
	FPT10	60	1	0.28	> 300
	FPT11	60	1	0.13	> 300
	FPT12	60	1	0.06	> 300
Instances Ikli	alp_7_30	30	1	> 300	> 300
			2	> 126	-
	alp_7_40	40	1	> 300	> 300
			2	> 286	-
	alp_7_50	50	1	> 300	> 300
			2	> 300	-
	alp_11_30	30	1	> 300	> 300
			2	8	-
	alp_11_40	40	1	> 300	> 300
			2	> 300	-
	alp_11_50	50	1	> 300	> 300
			2	> 300	-
	alp_15_30	30	1	> 300	> 300
			2	189	-
	alp_15_40	40	1	> 300	> 300
			2	117	-
	alp_15_50	50	1	> 300	> 300
			2	> 300	-
	alp_19_30	30	1	17	> 300
			2	2	-
	alp_19_40	40	1	> 300	> 300
			2	3	-
	alp_19_50	50	1	> 300	> 300
			2	> 300	-

TABLE 3.2 – Résultats numériques pour les deux principaux modèles MILP.

Méthode stochastique	Source	Problème	Nb de pistes, configuration	Contraintes opérationnelles	Objectif	Modèle	Approche de résolution	Instances (Nb d'avions)
Meta-heuristiques	Bencheikh *et al.* [10]	ALP	≥ 1, interdépendantes	Sep. et TW	min $\sum$ déviations	Graphe à 2 niveaux	ACO + Recherche locale	OR-library (10 à 50)
	Furini *et al.* [37]	ASP	1	Sep. et TW	min $\sum$ retards	Problème de permutation	TS + RH	OR-group Bologna (60)
	Hammouri *et al.* [43]	ALP	≥ 1, interdépendantes	Sep. et TW	min $\sum$ déviations	Problème de permutation	SA + Iterated local search	OR-library (10 à 500)
	Hancerliogullari *et al.* [44]	ALP	≥ 1, indépendantes	Équilibre de traffic, Sep. et TW	min $\sum$ retards	PLNE	SA + AATCSR, SA + ERT, SA + FPI	Générées (15 à 25)
	Hu et Paolo [54]	ALP	1	Sep. et TW	min $\sum$ retards	Espace artificiel 2D + Ripple spreading	GA	Générées (20 à 60)
	Rodriguez-Diaz *et al.* [89]	ASP	1	CPS, Sep. et TW	min $\sum$ retards	Problème de permutation	SA + CPS	OR-library (10 à 500), Générées (50 à 200), Aéroport de Gatwick (non spécifié)
	Hu [50]	ALP	1	Sep. et TW	min makespan	Problème de permutation	Matrix approximation + ACO	Hu *et al.* [51] (30), Générées (20 à 260)
	Shohel *et al.* [96]	ASP	≥ 1, différentes configurations	Sep.	max capacité piste	Prb optimisation avec contraintes	CPS + GA	Générées (50)
	Sabar et Kendall [90]	ALP	≥ 1, interdépendantes	Sep. et TW	min $\sum$ déviations	PLNE	ILS+VND	OR-library (10à 500)
	Salehipour [91]	ALP	≥ 1, indépendantes	Sep. et TW	min $\sum$ déviations	PLNE	Recherche locale + GuRoBi	OR-library (10 à500), Ghoniem *et al.* [39] (15 à 50)
	Salehipour *et al.* [93]	ALP	≥ 1, indépendantes	Sep. et TW	min $\sum$ déviations	PLNE	VND + SA , VNS + SA	OR-library (10 à 500)
	Salehipour *et al.* [94]	ALP	≥ 1, indépendantes	Sep. et TW	min $\sum$ retards	PLNE	VND + GA	Hansen [45] (12, 15 et 20)
	Soykan et Rabadi [97]	ASP	≥ 1, indépendantes	Sep. et TW	min $\sum$ retards	PLNE	TS + TTFGA	Ghoniem *et al.* [39](15 à 25)
	Zhan *et al.* [103]	ALP	Sep. et TW	1	min $\sum$ retards	Problème de permutation	ACO + RH	Générées (30 à 60), Bianco *et al.* [13] (20 et 30)
Heuristiques	Ikli *et al.* [58]	ALP	1	CPS, Sep. et TW	min $\sum$ retards	Arbre de recherche	Heuristique basée sur l'OP	Ikli [56] (18 à 40)
	Salehipour et Ahmadian [92]	ALP	1	Sep. et TW	min $\sum$ déviations	IP	Recherche locale + CPLEX	OR-library (10 à 500)
	Vadlamani et Seyedmohsen [101]	ALP	1	Sep. et TW	min $\sum$ déviations	PLNE	ALNS + CPLEX	OR-library (10 à 150)
RL	Soares *et al.* [19]	ATP	1	Sep. et TW	min retards taxiing	MDP	*Q-learning*	JFK airport
	Brittain et Wei [20]	Séquencer et séparer les avions	-	Sep.	min conflits	Formulation RL	*Hierarchical* deep RL	Jeu NASA Sector-33 [22] (2 à 5)

TABLE 3.3 – Résumé des méthodes stochastiques de la littérature pour le problème d'ordonnancement d'avions.

Shohel *et al.* [96] optent également pour une approche basée sur un algorithme génétique pour résoudre l'ASP. En plus de l'ASP, [96] considèrent le problème de trouver la configuration[2] de piste qui maximise sa capacité. Les contraintes considérées sont la séparation et les contraintes CPS. Ces dernières sont intégrées dans un algorithme génétique coopératif, où une population de séquences d'avions et de configurations de pistes candidates évoluent conjointement pour trouver la séquence la mieux adaptée et une configuration de piste qui maximise sa capacité. Des tests numériques sont effectués sur des instances simulant des heures de pointe à l'aéroport O'Hare de Chicago, qui comportent 8 pistes. Les configurations de pistes possibles sont également basées sur cet aéroport. Les auteurs observent qu'avec leur modèle intégré, ils obtiennent une meilleure utilisation des pistes par rapport à la capacité de base de l'aéroport considéré. En effet, la capacité de base est de 168 opérations (atterrissages et décollages) par heure, tandis que leur modèle atteint une capacité de 170 opérations par heure.

3.2.2 Colonies de fourmis

Zhan *et al.* [103] considèrent aussi l'ALP avec une seule piste, avec l'objectif de minimiser le retard total. Il présente pour la première fois un algorithme de colonies de fourmis dans une approche d'horizon glissant, appelée RHC-ACS-ASS (*Receding Horizon Control with Ant Colony System for Aircraft Sequencing and Scheduling*). Dans leur système de colonies de fourmis, l'heuristique FCFS est utilisée pour calculer la *phéromone* initiale. Des mises à jour sont ensuite effectuées pour favoriser la diversité de la population (diversification), et augmenter la désirabilité pour les meilleures solutions trouvées (intensification). Dans leur étude numérique, les auteurs ont généré des instances congestionnées, impliquant 30 et 60 avions. Des tests sont aussi effectués sur des instances de [14], afin de comparer leur approche à l'algorithme génétique de [51] qui utilise ces mêmes instances. Les résultats montrent que, dans une approche d'horizon glissant, l'algorithme de colonies de fourmis de [10] est plus performant que les algorithmes génétiques de [51].

Pour le cas multi-pistes et l'objectif plus général de minimiser les déviations par rapport aux heures préférentielles, Bencheikh *et al.* [10] présentent un nouveau modèle pour l'algorithme de colonies de fourmis, basé sur un graphe à deux niveaux. Le premier niveau choisit une piste disponible, tandis que le second sélectionne l'avion à atterrir sur cette piste. Dans cet algorithme, les fourmis partent d'un nœud initial artificiel, sélectionnent une piste, puis un avion à insérer sur cette piste, en fonction de la priorité de l'avion et de la mémoire de la colonie de fourmis. Le processus de sélection est répété jusqu'à ce que la liste des aéronefs disponibles soit vide. Des mises à jour de phéromones globales sont alors effectuées, et l'algorithme se termine lorsqu'un critère d'arrêt est rempli. Les tests numériques effectués sur les instances de la *OR-library* (10 à 50 avions) affichent des temps de calcul courts pour cet algorithme, capable de trouver les solutions optimales pour 80% des instances testées.

Un algorithme basé sur les colonies de fourmis est également proposé pour l'ALP dans [50], avec l'objectif de minimiser le makespan de la séquence. Les contraintes considérées sont les fenêtres de temps et la séparation. L'algorithme proposé comporte deux étapes principales.

2. Rappelons qu'une configuration de piste représente le nombre de pistes et leur disposition ; par exemple, une piste unique, plusieurs pistes parallèles, plusieurs pistes croisées, etc.

Dans la première, la matrice de séparation est approchée par une matrice de rang 2, similaire à celle de [34] présentée dans la section 3.1.2. Cette nouvelle matrice rend le problème indépendant de la séquence, et donc plus facile à résoudre. Ensuite, un algorithme de colonie de fourmis est utilisé pour compenser la perte de précision due à l'approximation de la matrice de séparation. Pour l'étude numérique, les auteurs utilisent une instance de [51]. Il génère également d'autres instances impliquant de 20 à 260 aéronefs. La comparaison avec l'algorithme ACO de [75] sur l'instance de [51] montre que son algorithme réussit à trouver la solution optimale en 15 secondes, tandis que l'algorithme de [75] ne trouvait qu'une solution réalisable, mais en moins d'une seconde. Pour les grandes instances générées, l'algorithme de [50] trouve de meilleures solutions – en termes de makepan de la séquence – que CPLEX dans les limites de temps prédéfinies, qui sont néanmoins discutables pour une application en temps réel (120 et 3600 secondes).

3.2.3 Recherche tabou

Furini *et al.* [37] considèrent l'ASP avec une piste unique. Les deux contraintes prises en compte sont la séparation et la fenêtre de temps ; l'objectif est de minimiser le retard total pondéré. Ils proposent une approche de résolution par horizon roulant, qui consiste à subdiviser la séquence originale en un ensemble de sous-séquences appelées *chunks*. Ensuite, ils ordonnent successivement les avions de chaque *chunk*, en utilisant soit leur PLNE vu dans la section 3.1.2, soit leur méthode de recherche tabou. Dans cette dernière, les solutions candidates sont définies par des permutations de l'ensemble d'avions. Le voisinage d'une solution est obtenue soit en échangeant la position de deux avions, soit en déplaçant un avion vers une nouvelle position. L'algorithme continue à améliorer les solutions candidates jusqu'à ce qu'un critère d'arrêt soit satisfait (nombre maximal d'itérations ou limite de temps). Les tests sont effectués sur les instance du *OR-group Bologna*, et montrent que cette approche est plus performante qu'une approche PLNE pour résoudre les chunks, dans la limite de temps prédéfinie (15 secondes).

Pour le cas multi-piste, Soykan et Rabadi [97] proposent aussi une approche de recherche *tabou*. Celle-ci peut être décomposée en deux étapes : la première calcule une solution gloutonne, de façon similaire à l'heuristique FCFS. La deuxième étape est le cœur de la recherche tabou où les solutions candidates sont améliorées. Ces solutions sont générées soit en échangeant deux avions sur la même piste ou sur deux pistes différentes, soit en supprimant/insérant un avion d'/dans une piste. L'algorithme s'arrête lorsqu'aucune amélioration n'est constatée après un nombre prédéfini d'itérations. L'algorithme est testé sur des instances de [39], impliquant 15, 20 et 25 avions, et jusqu'à 5 pistes. Les résultats donnent des temps de calcul très courts (moins d'une seconde) et un écart aux solutions optimales (*optimality gap*) moyen de $10,15\%$. Toutefois, l'écart peut être important pour certaines instances, comme par exemple l'instance 48 (25 avions, 4 pistes) où l'écart est de $94,75\%$.

3.2.4 Recherche à voisinage variable

La recherche à voisinage variable (*Variable Neighborhood Search* – VNS) est utilisée par Salehipour *et al.* [94] pour résoudre l'ALP avec plusieurs pistes. L'objectif est de minimiser le retard total de la séquence. Les contraintes classiques de séparation entre paires d'avions atterrissant sur la même piste sont prises en compte. Cependant, la séparation est supposée constante (2 unités de temps), contrairement à la séparation relative aux turbulences de sillage, qui dépend de la catégorie des avions. La séparation entre paires d'avions atterrissant sur deux pistes différentes est également constante (une unité de temps). Dans leur algorithme de recherche à voisinage variable, quatre structures de voisinage sont utilisées : échange de deux avions sur une même piste, échange d'avions sur deux pistes différentes ou échange des séquences entre deux pistes. La quatrième structure de voisinage supprime un avion d'une piste pour l'insérer dans une autre piste. Les tests numériques s'appuient sur des instances générées de [45], impliquant jusqu'à 20 avions et 5 pistes. Bien qu'il améliore l'heuristique de base construite par les auteurs, l'algorithme nécessite des temps de calcul élevés.

La recherche à voisinage variable est aussi utilisée par Salehipour *et al.* [93], mais cette fois-ci pour améliorer les solutions candidates au sein d'un algorithme de recuit simulé. L'algorithme est appelé SA+VNS et s'appuie sur des structures de voisinage similaires à celles définies ci-dessus. L'algorithme choisit un voisinage et tente d'améliorer la solution courante (du recuit simulé). Lorsqu'aucune amélioration n'est possible, la structure de voisinage est alors changée. Le processus est répété jusqu'à ce qu'un critère d'arrêt soit atteint. Les tests effectués sur les instances de la *OR-library* montrent que cet algorithme trouve les solutions optimales pour toutes les instances impliquant jusqu'à 50 avions. Pour les autres instances (100 à 500 avions), l'algorithme réussit à trouver des solutions de bonne qualité (en terme d'écart aux solutions optimales).

Le dernier article qui utilise la recherche à voisinage variable que nous étudions est [90] par Sabar et Kandall, qui considèrent le même problème que celui étudié dans [93]. Les mêmes structures de voisinage décrites ci-dessus sont utilisées dans [90]. Quant aux tests numériques, ils s'appuient aussi sur les instances de la *OR-library*. Les auteurs comparent ensuite leur algorithme à plusieurs heuristiques de la littérature, notamment, la métaheuristique hybride SA+VNS de [93] et les heuristiques de [8]. D'un point de vue du temps de calcul, l'approche de Sabar et Kandall [90] a les temps de calcul les plus faibles par rapport aux heuristiques mentionnées ci-dessus, pour toutes les instances de la OR-library.

3.2.5 Recuit simulé

Hancerliogullari *et al.* [44] présentent plusieurs heuristiques pour résoudre le modèle PLNE décrit dans 3.1.2, y compris un algorithme de recuit simulé. La solution initiale est construite en utilisant trois algorithmes gloutons : le premier est basé sur la règle du FCFS, les deux autres sont inspirés de la littérature de l'ordonnancement des tâches, et font atterrir les avions selon un indice de priorité. Les solutions candidates du recuit simulé sont obtenues en choisissant au hasard deux avions et en échangeant leurs positions, pourvu que les contraintes de fenêtres de temps ne soient pas violées. Les tests ont été effectués sur des instances générées impliquant 15, 20 et 25 avions et 2 à 5 pistes. Ils affichent des temps de calcul très courts pour

le recuit simulé, et montrent que ce dernier améliore considérablement les solutions initiales fournies par les trois algorithmes gloutons mentionnés ci-dessus.

Rodriguez-Diaz *et al.* [89] examinent le problème d'ordonnancement simultané d'atterrissages et de décollages (ASP) sur une piste unique, en prenant en considération – en plus de la séparation et de la fenêtre de temps – les contraintes CPS. L'objectif est de proposer de bonnes solutions qui améliorent le retard total de l'heuristique FCFS, sans trop dévier de celle-ci. Ils optent pour un recuit simulé, dont les solutions candidates sont générées de la même façon que [44], sauf que cette fois-ci, elles doivent respecter en plus les contraintes du CPS. Des tests numériques on été effectués sur trois types d'instances : des instances générées artificiellement, les instances de la *OR-library* et des instances construites à partir de données réelles de l'aéroport de Gatwick. Pour les instances générées, leur algorithme est capable de trouver les solutions optimales pour 828 sur les 2000 instances ($41,4\%$). Les instances de la OR-library sont utilisées afin de comparer leur approche aux deux approches [83] et [93]. Les résultats montrent que le recuit simulé de [89] surpasse les deux approches de [83] et [93] en terme de temps de calcul. Toutefois, en terme de qualité des solutions obtenues, les résultats des deux approches [83] et [93] sont meilleurs. Les tests supplémentaires effectués sur les données de Gatwick montrent qu'avec l'approche de [89], on peut atteindre un pourcentage d'amélioration allant jusqu'à 30% par rapport à la FCFS.

Récemment, Hammouri *et al.* [43] ont proposé une méthode hybride utilisant le recuit simulé et la recherche locale itérative (*iterated local search*) pour l'ALP avec une ou plusieurs pistes. L'algorithme proposé comporte deux boucles principales. La boucle interne utilise un recuit simulé pour améliorer les solutions courantes, dont les structures de voisinage sont similaires à celles de [93]. La boucle externe s'appuie sur une approche de recherche locale itérative pour perturber les solutions ou relancer la recherche, dans le but d'explorer d'autres régions de l'espace des solutions et ainsi s'échapper des minima locaux. Des tests numériques sont effectués sur les instances de la *OR-library* afin de comparer cette approche à la *scatter search* de [83] et la méthode hybride de [93] qui utilisent ces mêmes instances. Les résultats montrent que l'algorithme de [43] est capable de trouver les solutions optimales pour les petites et moyennes instances (10 à 50 avions) en des temps de calcul raisonnables. De plus, il détient le meilleur temps de calcul moyen, comparé aux deux méthodes [83] et [93].

3.2.6 Matheuristiques

Les matheuristiques sont des méthodes d'optimisation hybrides obtenues par la combinaison de méthodes en programmation mathématique et de métaheuristiques [16]. Les travaux de Salehipour et Ahmadian [92], Vadlamani et Seyedmohsen [101] et ceux, très récents, de [91] se réclament de telles approches.

Salehipour et Ahmadian [92] considèrent le problème d'ordonnancement d'atterrissage (ALP) avec une piste et l'objectif de minimiser la somme des déviations par rapport aux dates préférentielles. La matheuristique proposée se décompose en deux étapes. Une première étape génère une séquence basée sur la règle du FCFS, et une deuxième étape améliore cette séquence. L'algorithme d'amélioration est basé sur une recherche locale qui permet à une partie des avions de changer leur position initiale, jusqu'à ce qu'un critère d'arrêt soit satisfait.

La séquence étant fixée, un programme linéaire est ensuite résolu (à l'aide de CPLEX) pour trouver les dates cibles d'atterrissage. Les auteurs ont effectué des tests sur les instances de la *OR-library*, et ont comparé leur méthode à deux heuristiques de la littérature : la méthode hybride de [93] et la *scatter search* de [83]. Leurs résultats montrent que [92] a le meilleur écart aux solutions optimales pour 10 sur un total de 13 instances de la *OR-library*. Cette approche a été récemment étendue au cas de pistes multiples et indépendantes dans [91].

Vadlamani et Seyedmohsen [101] considèrent le même problème que Salehipour et Ahmadian [92] avec le même objectif et sous les mêmes contraintes. Leur méthode se décompose également en deux étapes : recherche d'une séquence d'atterrissage (sans les dates) et calcul de dates d'atterrissage pour les avions de cette séquence fixée (programme linéaire résolu par CPLEX). L'approche est testée sur les données de la *OR-library*, en la comparant au recuit simulé et à la méthode hybride de [93]. Leurs résultats concluent que l'approche de [101] peut trouver des solutions optimales pour les instances impliquant 10 à 150 avions. Cependant, les temps de calcul peuvent être longs pour certaines instances, notamment l'instance avec 150 avions où l'algorithme requiert 234 secondes pour trouver la solution optimale.

3.2.7 Apprentissage par renforcement

L'apprentissage par renforcement (*Reinforcement Learning* – RL) [98] est utilisé pour résoudre une grande variété de problèmes ; cela inclut les problèmes d'ordonnancement d'avions.

Soares *et al.* [19] s'intéressent au problème d'ordonnancement de décollages (ATP) sur une seule piste, dans l'objectif de maximiser le respect des fenêtres de temps de décollage imposées. Le modèle proposé est différent des formulations mathématiques classiques basées sur des PLNE ou des graphes, il s'agit ici d'un processus de décision markovien (*Markov Decision Process* – MDP), qui est résolu avec l'algorithme très connu d'apprentissage par renforcement appelé *Q-learning* [102]. Dans leur modèle, les *agents* correspondent aux avions, chaque *état* correspond à la position au sol de l'avion, en fonction de sa *phase* (stationné – *parked*, en roulage – *taxiing*, décollé – *taken off*). Les actions consistent généralement à retarder un avion et la *récompense* est définie de manière à minimiser le retard pendant le roulage au sol. Les tests sont effectués sur des données réelles de l'aéroport international John F. Kennedy, impliquant 698 décollages d'avions. Les résultats de [19] montrent que cette approche obtient des performances similaires à celles des contrôleurs au sol, en terme de pourcentage de fenêtres de temps respectées. Cependant, en présence de perturbations (cas stochastique), leur approche est meilleure.

Brittain et Wei [20] proposent un autre cadre d'apprentissage par renforcement pour modéliser le problème d'ordonnancement et de séparation d'avions entre deux points repères donnés : un point de départ et un *point-objectif*. L'ordonnancement à la piste peut être un cas particulier de ce problème, en prenant la piste comme un de ces deux points. Le problème plus général que considère [20] fournit des recommandations de vitesse et de route aériennes afin de bien séquencer et séparer les avions. Les tests sont effectués sur une application en ligne appelée *NASA Sector-33* [22], il s'agit d'un jeu de contrôle de trafic aérien contenant 35 instances à résoudre, impliquant jusqu'à 5 avions. Afin de s'adapter à l'environnement du jeu, ils proposent un modèle (semblable à un MDP) composé d'agents, d'états, d'actions et

de récompenses. Deux types d'agents sont proposés : les *agents parents* et les *agents enfants*. L'état d'un *agent parent* contient une capture d'écran de l'écran de jeu. L'état d'un *agent enfant* contient des informations sur le point-objectif, la vitesse et l'accélération de l'avion, un identificateur de route, ainsi que des informations sur les *agents* les plus proches. Les actions d'un agent parent/enfant consistent à modifier ou à maintenir la route/vitesse de l'avion. La récompense est conçue de manière à pénaliser les agents en conflit (séparés par moins de 3 NM). Les instances de ce modèle sont résolues par un algorithme d'apprentissage par renforcement profond hiérarchique (*hierarchical deep reinforcement learning*). Cet algorithme combine l'algorithme de Q-learning [102] et les réseaux de neurones artificiels [68]. Il est dit hiérarchique car les actions sont effectuées à deux niveaux : d'abord au niveau de l'état parent qui choisit la route, puis au niveau de l'état enfant qui sélectionne la vitesse de l'avion. Les tests effectués dans l'environnement de la NASA présentés ci-dessus montrent la viabilité de leur approche pour séquencer et séparer correctement les avions de et vers les points de repères du jeu.

Conclusions du chapitre

Nous avons présenté dans ce chapitre plusieurs modèles et approches de résolution pour les trois problèmes ALP, ATP et ASP, en nous concentrant principalement sur les travaux récents de la littérature (depuis 2010). Les méthodes et modèles étudiés incluent la programmation mixte en nombres entiers, la programmation dynamique et les approches stochastiques, y compris les méthodes hybrides combinant programmation mathématique et métaheuristiques. Nous avons étudié aussi deux nouvelles approches de la littérature basées sur l'apprentissage par renforcement [19, 20].

Nous constatons que les contributions de la littérature se concentrent principalement sur le problème d'ordonnancement d'atterrissages ; seuls quelques travaux abordent spécifiquement le problème d'ordonnancement de décollages. Nous constatons également que les instances de la littérature ont perdu de leur intérêt, car elles peuvent être résolues dans des temps de calcul très courts. Nous proposons dans le chapitre suivant des bases de données et des instances pour l'ALP qui sont difficiles à résoudre. Nous proposons également une formulation PLNE pour résoudre l'ALP, avec l'objectif de minimiser les coûts de retard. La particularité de ce modèle PLNE est qu'il intègre explicitement les contraintes de CPS et qu'il considère une représentation du coût de retard réaliste : il s'agit d'une fonction coût convexe et linéaire par morceaux.

Chapitre 4

Description du problème d'ordonnancement d'avions à l'atterrissage et résolution par PLNE

Nous nous intéressons dans ce chapitre à la résolution par PLNE du problème d'ordonnancement d'atterrissages d'avions (ALP). Nous prenons en considération les contraintes de séparation, de fenêtre de temps et de changement de positions (CPS). L'objectif est de minimiser le coût total de retard. Nous proposons d'abord une formulation PLNE pour modéliser ce problème. La particularité de ce modèle PLNE est qu'il intègre explicitement les contraintes de CPS et qu'il considère une représentation du coût de retard réaliste : il s'agit d'une fonction coût convexe et linéaire par morceaux. Nous présentons également dans ce chapitre notre deuxième contribution de la thèse : la proposition de nouvelles instances réalistes et difficiles pour l'ALP, construites à partir de quatre ensembles de données et mettant en jeu des coûts de retard linéaires par morceaux. Finalement, nous présentons une étude numérique de notre modèle PLNE, et nous examinons l'impact des contraintes de changement de positions sur la qualité des solutions obtenues.

Sommaire

4.1 Énoncé du problème

Notre problème peut être formulé comme suit : soit $\mathcal{A}$ un ensemble d'indices correspondant à $|\mathcal{A}|$ avions demandant l'atterrissage dans un même aéroport de destination. Les autorités

de gestion des flux, telles que le *Network Manager Operations Centre*, imposent une date préférentielle d'atterrissage pour chaque avion dans le but d'optimiser la sécurité et la capacité de l'espace aérien. Pour chaque avion $i \in \mathcal{A}$, notons T_i cette date préférentielle, et $[T_i, L_i]$ la fenêtre de temps d'atterrissage, où L_i représente la date d'arrivée au plus tard, dont la valeur dépend de contraintes techniques et opérationnelles, telles que la consommation de carburant et les vitesses minimale et maximale de l'avion i. Notre problème consiste alors à trouver des dates cibles d'atterrissage, $x_i \in [T_i, L_i]$ $(i \in \mathcal{A})$, qui minimisent le coût du retard et qui respectent les contraintes de séparation, de fenêtre de temps et de changement de positions.

4.2 Formulations mathématiques

La formulation mathématique que nous proposons est un modèle PLNE basé sur [9], que nous adaptons pour y incorporer explicitement les contraintes de CPS. Nous généralisons cette formulation au cas où la fonction-objectif est linéaire par morceaux convexe, de façon à obtenir une représentation plus réaliste des coûts de retard que les représentations linéaires fréquemment étudiées dans la littérature.

4.2.1 Formulation PLNE

Données

On définit les éléments suivants :

$\mathcal{A}$: ensemble des indices d'avions.

S_{ij} : séparation minimale temporelle (≥ 0) entre deux avions $i, j \in \mathcal{A}$, où l'avion i atterrit avant l'avion j. Nous utilisons la matrice de séparation au seuil de piste, suivant les catégories de turbulences de sillage de l'OACI, exprimée en secondes (table 4.1).

Pour chaque avion $i \in \mathcal{A}$, on défint de plus :

T_i : date préférentielle d'atterrissage de i.

L_i : date d'atterrissage au plus tard de i.

c_i : coût $(c_i \geq 0)$ par seconde d'atterrir plus tard que T_i.

Variables de décision

x_i : heure cible d'atterrissage de l'avion i.

$$\delta_{ij} = \begin{cases} 1 \text{ si l'avion } i \text{ atterrit avant l'avion } j, \ i, j \in \mathcal{A} : i \neq j, \\ 0 \ \text{ sinon.} \end{cases}$$

		Avion suiveur		
		H	M	L
Avion générateur	H	96	157	196
	M	60	69	131
	L	60	69	82

TABLE 4.1 – Matrice de séparation (en secondes) au seuil de piste selon les trois catégories de turbulences de sillage : *Heavy* (H), *Medium* (M) et *Light* (L).
Source : Balakrishnan et Chandran [6].

Remarque 4.2.1

Pour chaque avion $i \in \mathcal{A}$, les variables $\delta_{ij}, j \in \mathcal{A}$, permettent de calculer la position de l'avion i dans n'importe quelle séquence. Cette position est donnée par :

$$|\mathcal{A}| - \sum_{j \in \mathcal{A}} \delta_{ij}$$

Contraintes

Nous prenons en considération quatre types de contraintes opérationnelles :

— Les **fenêtres de temps**. Elle permettent d'éviter que les avions subissent des retards importants. Elles sont représentées par les inégalités suivantes :

$$T_i \leq x_i \leq L_i, \quad i \in \mathcal{A}. \tag{4.1}$$

— Les contraintes de **séquencement**. Elles permettent d'ordonner l'atterrissage de toute paire d'avions : pour tout $i, j \in \mathcal{A} : i \neq j$, soit l'avion i atterrit avant l'avion j, soit l'avion j atterrit avant l'avion i. Elle sont représentées par les équations suivantes :

$$\delta_{ij} + \delta_{ji} = 1, \quad i, j \in \mathcal{A} : i < j, \tag{4.2}$$

— Les contraintes de **séparation**. Elle garantissent la séparation minimale de turbulence de sillage entre deux atterrissages : pour tout $i, j \in \mathcal{A} : i \neq j$, la date d'atterrissage de l'avion j est supérieure ou égale à la date d'atterrissage de l'avion i plus la séparation minimale entre l'avion i et l'avion j, quand l'avion i atterrit avant l'avion j (c'est-à-dire, quand $\delta_{ij} = 1$). Elles sont modélisées par les inégalités suivantes :

$$x_j \geq x_i + S_{ij} - M_{ij}(1 - \delta_{ij}), \quad i, j \in \mathcal{A} : i \neq j, \tag{4.3}$$

où $M_{ij} > 0$ est un paramètre dont la valeur doit être choisie suffisamment grande pour que la contrainte soit toujours satisfaite lorsque $\delta_{ij} = 0$ (c'est-à-dire, quand l'avion i n'atterrit pas avant l'avion j). On peut facilement montrer que cela est le cas en choisissant la valeur $M_{ij} = L_i + S_{ij} - T_j$.

Les contraintes (4.3) peuvent être renforcées, comme suggéré dans [9], en partitionnant l'ensemble des paires d'avions $\mathcal{A}\times\mathcal{A}$ en trois sous-ensembles $\mathcal{E}, \mathcal{F}$ et $\mathcal{J}$ définis ci-dessous, et en définissant une séparation adaptée à chacune de ces sous-ensembles. Soient :

— $\mathcal{E} \subseteq \mathcal{A} \times \mathcal{A}$ l'ensemble des paires d'avions $(i, j) \in \mathcal{A} \times \mathcal{A} : i \neq j$, dont **l'ordre relatif des atterrissage n'est pas connu** à l'avance :

$$\mathcal{E} = \{i, j \in \mathcal{A} : i \neq j \mid T_j \leq T_i \leq L_j \text{ ou } T_j \leq L_i \leq L_j\};$$

— $\mathcal{F} \subseteq \mathcal{A} \times \mathcal{A}$ l'ensemble des paires d'avions $(i, j) \in \mathcal{A} \times \mathcal{A} : i \neq j$ pour lesquelles l'avion i doit **certainement atterrir avant** l'avion j, mais pour lesquelles la contrainte de **séparation** au seuil de piste n'est **pas nécessairement satisfaite** :

$$\mathcal{F} = \{i, j \in \mathcal{A} : i \neq j \mid L_i < T_j \text{ et } L_i + S_{ij} > T_j\};$$

— $\mathcal{J} \subseteq \mathcal{A} \times \mathcal{A}$ l'ensemble des paires d'avion $(i, j) \in \mathcal{A} \times \mathcal{A} : i \neq j$, telles que l'avion i doit **certainement atterrir avant** l'avion j, et pour lesquelles la contrainte de **séparation** au seuil de piste **est satisfaite** par conséquence des contraintes de fenêtres de temps :

$$\mathcal{J} = \{i, j \in \mathcal{A} : i \neq j \mid L_i < T_j \text{ et } L_i + S_{ij} \leq T_j\};$$

Les contraintes (4.3) peuvent alors être remplacées par les contraintes :

$$\delta_{ij} = 1, \qquad i, j \in \mathcal{F} \cup \mathcal{J} \tag{4.4}$$
$$x_j \geq x_i + S_{ij}, \qquad i, j \in \mathcal{F} \tag{4.5}$$
$$x_j \geq x_i + S_{ij} - M_{ij}(1 - \delta_{ij}), \qquad i, j \in \mathcal{E}. \tag{4.6}$$

— Les contraintes de **changement de positions**. Rappelons que ces contraintes empêchent les avions de dévier de leur position initiale dans la séquence FCFS de plus d'un nombre m de positions, choisi par l'utilisateur et appelé *Maximum-Position Shift*. Sans perte de généralité, supposons que pour chaque avion $i \in \mathcal{A}$, son indice représente également sa position dans la séquence FCFS. Les contraintes de CPS peuvent alors être modélisées par les inégalités suivantes :

$$i - m \leq |\mathcal{A}| - \sum_{\substack{j \in \mathcal{A} \\ j \neq i}} \delta_{ij} \leq i + m, \quad i \in \mathcal{A}. \tag{4.7}$$

Fonction-objectif Nous souhaitons minimiser le coût total du retard par rapport aux dates préférentielles d'atterrissage $T_i, i \in \mathcal{A}$. Si la variation du coût de retard est supposée linéaire, la fonction-objectif est alors donnée par :

$$\min_{x,\delta} \sum_{i \in \mathcal{A}} c_i(x_i - T_i). \tag{4.8}$$

Dans l'équation (4.8), la variable x représente le vecteur dont la $i^{\text{ième}}$ composante est x_i, et la variable δ représente la matrice dont la composante (i, j) est δ_{ij}.

Modèle complet

$$\min_{x,\delta} \quad \sum_{i\in\mathcal{A}} c_i(x_i - T_i) \tag{4.9}$$

$$\textbf{s.c}: \quad T_i \leq x_i \leq L_i \qquad i \in \mathcal{A} \tag{4.10}$$

$$\delta_{ij} + \delta_{ji} = 1 \qquad i,j \in \mathcal{A} : i < j \tag{4.11}$$

$$\delta_{ij} = 1 \qquad i,j \in \mathcal{F} \cup \mathcal{J} \tag{4.12}$$

$$x_j \geq x_i + S_{ij} \qquad i,j \in \mathcal{F} \tag{4.13}$$

$$x_j \geq x_i + S_{ij} - M_{ij}(1 - \delta_{ij}) \qquad i,j \in \mathcal{E} \tag{4.14}$$

$$i - m \leq |\mathcal{A}| - \sum_{\substack{j\in\mathcal{A} \\ j\neq i}} \delta_{ij} \leq i + m \qquad i \in \mathcal{A} \tag{4.15}$$

$$x_i \geq 0 \qquad i \in \mathcal{A} \tag{4.16}$$

$$\delta_{ij} \in \{0,1\} \qquad i,j \in \mathcal{A} : i \neq j \tag{4.17}$$

Ce choix de fonction-objectif est très fréquent dans la littérature. Pourtant, cette fonction ne représente pas bien les coûts réels encourus par un retard. En effet, en pratique le coût du retard n'est pas une fonction linéaire du retard. Par exemple, selon [23], le coût de 5 minutes de retard d'un avion B744 dans la zone TMA est de 710 euros, en revanche, le coût de 15, 30 et 60 minutes de retard sont respectivement 2 760, 7 780 et 29 000 euros. Il est donc plus pertinent de considérer une fonction coût (de retard) linéaire par morceaux, avec des pentes croissantes. Dans la sous-section suivante nous intégrons une telle fonction-objectif linéaire par morceaux, convexe dans notre formulation PLNE.

4.2.2 Modèle du coût de retard linéaire par morceaux, convexe

Nous utilisons dans la suite la notation f_i pour tout $i \in \mathcal{A}$, pour désigner la fonction coût spécifique à l'avion i. La fonction-objectif est alors la somme de ces fonctions coût. Par exemple, dans le modèle linéaire (4.9)-(4.17), la fonction coût pour chaque avion i est : $f_i(x_i) = c_i(x_i - T_i)$. La fonction-objectif (4.9) peut alors être écrite comme $\sum_{i\in\mathcal{A}} f_i(x_i)$.

Nous supposons dans cette sous-section que la fonction coût f_i est convexe et linéaire par morceaux pour chaque avion $i \in \mathcal{A}$, comme illustré dans la figure 4.1. La reformulation linéaire d'une fonction-objectif convexe et linéaire par morceaux est classique en recherche opérationnelle (voir par exemple [17], chapitre 9). Pour représenter une telle fonction sous forme linéaire, nous introduisons les notations suivantes :

Pour chaque avion $i \in \mathcal{A}$:

- $\mathcal{R}_i = \{p_i^1, p_i^2, ..., p_i^{|\mathcal{R}_i|}\}$ représente l'ensemble de points de rupture de f_i à l'intérieur de son domaine de définition $[T_i, L_i]$, comme illustré dans la figure 4.1 ;
- $c_i^k, k = 1, ..., |\mathcal{R}_i| + 1$ sont les pentes de f_i. La convexité de f_i implique que $c_i^k < c_i^{k+1}$,

pour $k = 1, ..., |\mathcal{R}_i|$.

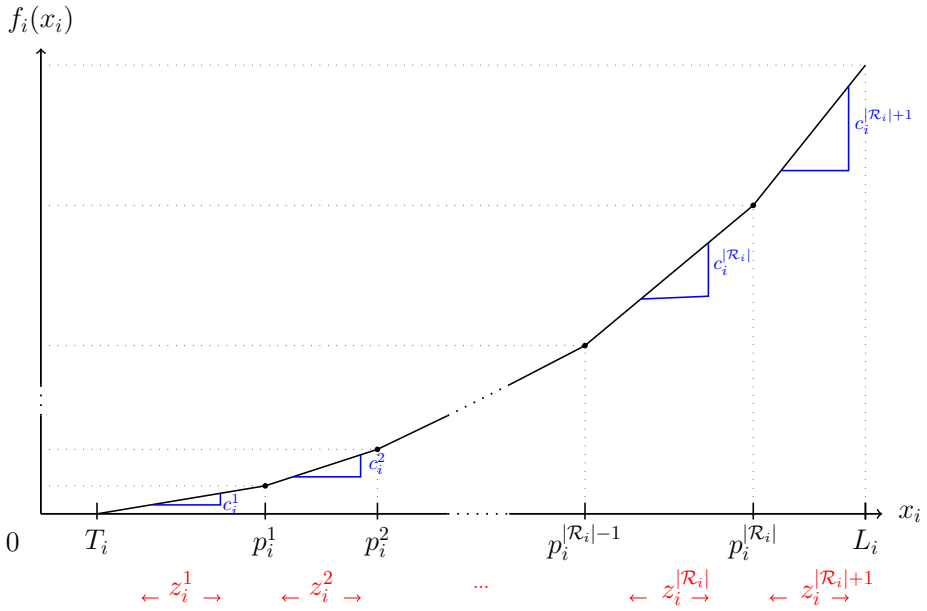


FIGURE 4.1 – Modèle de fonction coût f_i (pour un avion $i \in \mathcal{A}$) linéaire par morceaux et convexe sur son domaine de définition $[T_i, L_i]$.

Une telle fonction peut être linéarisée en utilisant différentes formulations [62]. Nous adoptons la formulation des « coûts incrémentaux » (*incremental cost formulation*) utilisée dans [9]. Nous introduisons alors, pour chaque avion $i \in \mathcal{A}$, $|\mathcal{R}_i| + 1$ variables auxiliaires, que nous notons $z_i^1, z_i^2, ..., z_i^{|\mathcal{R}_i|+1}$ (figure 4.1). Le coût linéaire par morceaux f_i peut alors être modélisé comme suit :

$$f_i(x_i) = \sum_{k=1}^{|\mathcal{R}_i|+1} c_i^k z_i^k \tag{4.18}$$

$$x_i = T_i + \sum_{k=1}^{|\mathcal{R}_i|+1} z_i^k \tag{4.19}$$

$$0 \leq z_i^1 \leq p_i^1 - T_i \tag{4.20}$$

$$0 \leq z_i^{|\mathcal{R}_i|+1} \leq L_i - p_i^{|\mathcal{R}_i|} \tag{4.21}$$

$$0 \leq z_i^k \leq p_i^k - p_i^{k-1}, \qquad k = 2, ..., |\mathcal{R}_i|. \tag{4.22}$$

Pour considérer une telle fonction coût dans notre formulation PLNE, nous remplaçons la fonction-objectif (4.9) par $\sum_{i \in \mathcal{A}} f_i(x_i)$ en utilisant l'équation (4.18), puis nous ajoutons au modèle PLNE les contraintes sur les nouvelles variables auxiliaires, exprimées par les équations (4.19)–(4.22) pour chaque avion $i \in \mathcal{A}$. Le programme linéaire mixte en nombres entiers avec une fonction-objectif linéaire par morceaux convexe est donc :

$$\min_{x,\delta,z} \quad \sum_{i\in\mathcal{A}} \sum_{k=1}^{|\mathcal{R}_i|+1} c_i^k z_i^k \tag{4.23}$$

$$\textbf{s.c}: \quad x_i = T_i + \sum_{k=1}^{|\mathcal{R}_i|+1} z_i^k \qquad i \in \mathcal{A} \tag{4.24}$$

$$0 \leq z_i^1 \leq p_i^1 - T_i \qquad i \in \mathcal{A} \tag{4.25}$$

$$0 \leq z_i^{|\mathcal{R}_i|+1} \leq L_i - p_i^{|\mathcal{R}_i|} \qquad i \in \mathcal{A} \tag{4.26}$$

$$0 \leq z_i^k \leq p_i^k - p_i^{k-1} \qquad k = 2, ..., |\mathcal{R}_i|, i \in \mathcal{A} \tag{4.27}$$

$$\text{ainsi que les contraintes (4.10)} - \text{(4.17)} \tag{4.28}$$

Remarques :

La formulation PLNE (4.9)–(4.17) implique $|\mathcal{A}|$ variables continues, au plus $|\mathcal{A}|(|\mathcal{A}|-1)$ variables binaires et au plus $[\frac{1}{2}|\mathcal{A}| + \frac{3}{2}|\mathcal{A}|^2]$ contraintes (sans compter les contraints de binarité (4.17), et de positivité (4.16)). On verra à la section 4.4.1 que la taille effective du problème peut être beaucoup plus petite. La formulation (4.23)–(4.28) ajoute en plus, pour chaque avion i : $(|\mathcal{R}_i|+1)$ variables (auxiliaires) continues et $(|\mathcal{R}_i|+2)|\mathcal{A}|$ contraintes. On verra qu'ici encore la taille effective du modèle (4.23)–(4.28) est généralement bien plus petite.

4.3 Construction de nouvelles instances

Les instances disponibles dans la littérature pour l'ALP (*OR-library* [7] et *OR-Group Bologna* [37] par exemple) ont perdu de leur intérêt, car elles peuvent être résolues à l'optimalité dans des temps de calcul très courts avec des solveurs tels que CPLEX [25]. Les seules instances encore difficiles sont les très larges instances de la *OR-library*, impliquant $|\mathcal{A}| = 100$ à 500 aéronefs, comme nous avons montré dans la section 3.1.3. En outre, les données de la littérature n'impliquent que des coûts de retard dépendant linéairement de l'amplitude du retard. Pour toutes ces raisons, nous avons construit — à partir de données de trafic réel — des bases de données et des instances pour l'ALP qui sont difficiles à résoudre. De plus, dans nos données, les coûts de retard sont plus réalistes, car ils dépendent à la fois du type de l'avion et de l'amplitude de son retard. Les calculs de ces coûts sont basés sur les travaux [23, 24], qui procèdent à une analyse détaillée des coûts de retard, en prenant en compte plusieurs facteurs : consommation de carburant, nombre de passagers, coût de maintenance, etc. Toutes nos données, instances et détails des calculs des coûts sont publiquement disponibles dans [55].

Nous présentons dans cette section nos ensembles de données et nos 12 instances de référence. Dans la section 4.3.1, nous expliquons le processus de construction de ces 12 instances. Ensuite, nous présentons dans la section 4.3.2 quelques statistiques importantes sur nos données, telles que le nombre d'avions de chaque catégorie de turbulence de sillage, qui donne une indication de l'hétérogénéité de l'instance, et le nombre d'avions par heure, qui reflète sa congestion.

4.3.1 Détails de la construction

Les données brutes proviennent de deux journées de trafic sur l'aéroport de Paris-Orly, provenant du *OpenSky Network* [100] : une journée de juillet 2018 (322 avions) et une journée d'avril 2019 (94 avions). Nous fusionnons d'abord ces deux journées de trafic afin d'obtenir un ensemble plus large et plus congestionné. Ensuite, nous divisons cet ensemble de données selon les intervalles horaires suivants : `07:00-11:00`, `11:00-15:00`, `15:00-19:00` et `19:00-23:00`. Cette subdivision engendre les quatre ensembles de données que nous avons nommé `data_7_11.csv`, `data_11_15.csv`, `data_15_19.csv` et `data_19_23.csv` dans [55]. Enfin, nous ajoutons artificiellement des avions de catégorie *Light* à chacun des quatre ensembles de données, afin d'obtenir un mélange de 40 % d'avions *Heavy*, 40 % d'avions *Medium* et 20 % d'avions *Light*, pour simuler des périodes de congestion réalistes, comme suggéré dans [85].

Ces quatre ensembles de données servent de base pour générer des instances pour le problème d'ordonnancement d'atterrissages d'avions. En l'occurrence, les 12 instances de référence utilisées dans la section 3.1.3 sont générées à partir de ces ensembles de données, en considérant simplement les $|\mathcal{A}|$ premières lignes de données ($|\mathcal{A}| \in \{30, 40, 50\}$) de chacun des quatre ensembles de données. Ces 12 instances peuvent donc servir de *benchmark* pour des études futures.

Les informations fournies pour chaque avion comprennent : le modèle de l'avion (`mdl`), parfois appelé type d'avion dans la littérature, la catégorie de turbulence de sillage (`category`), la date préférentielle d'atterrissage (`sta`) et la date réelle d'atterrissage (`ata`) au format `HH:MM:SS` et en secondes. Nous fournissons également des coûts de retard réalistes pour chaque avion, calculés à partir des travaux de Cook *et al.* [24] et de Cook et Tanner [23]. La fonction qui reflète la variation du coût est convexe et linéaire par morceaux. Cependant, les deux rapports mentionnés ci-dessus ne présentent des calculs de coût que pour 15 types d'avions, alors que nos données font intervenir 26 types d'avions différents. Afin d'extrapoler les coûts pour les types d'avions qui n'ont pas été considérés, nous faisons une régression linéaire sur les types d'avions étudiés dans [23]. Les détails de cette régression (variables, paramètres, coefficient de détermination R^2) ainsi que son implémentation sont disponibles dans `http://data.recherche.enac.fr/ikli-alp/`, dans le dossier `ikli_codes`.

La figure 4.2 montre la variation des coûts de retard pour l'ensemble des 26 types d'avions disponibles dans nos bases de données (*data sets*). Les abscisses 300, 900, 1800 et 3600 sont les points de rupture (*break points*) de la fonction coût, en secondes (correspondant à 5, 15, 30 et 60 minutes de retard, respectivement). À notre connaissance, c'est la première fois qu'une fonction coût linéaire par morceaux aussi détaillée et réaliste est fournie dans des données pour l'ALP, accessibles publiquement.

4.3.2 Description des instances

Nous résumons dans la Table 4.2 les principales caractéristiques de chacune des bases de données `data_7_11.csv`, `data_11_15.csv`, `data_15_19.csv`, et `data_19_23.csv`, à savoir : le nombre total d'avions, le nombre d'avions par heure et la répartition des catégories de

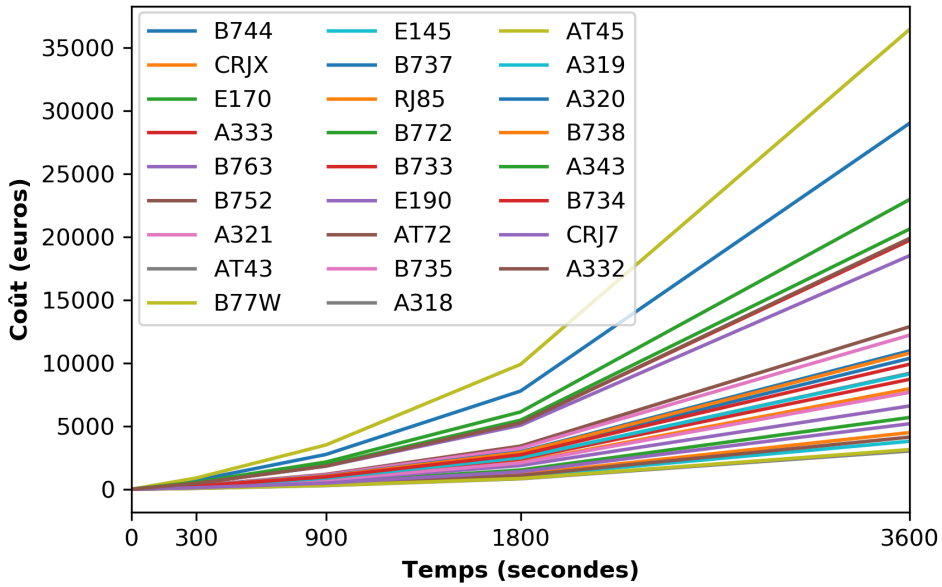


FIGURE 4.2 – Variation du coût de retard, en euros, pour les 26 types d'avions disponibles dans nos bases de données.

turbulence de sillage.

Les première et deuxième colonnes présentent le nom et l'intervalle temporel de la base de données. La troisième colonne indique le nombre total d'aéronefs, "$|\mathcal{A}|$". La quatrième colonne affiche le nombre moyen d'avions par heure, "Moyenne avions/heure", dans chaque base de données. Cette moyenne est égale au nombre total d'avions dans la base de données divisé par la longueur de l'intervalle (en heures). Enfin, la dernière colonne, "Mix d'avions", indique le nombre d'avions des trois catégories de turbulence de sillage : *Heavy* (H), *Medium* (M) et *Light* (L), introduites dans la section 2.1.1.

Base de donnée	Intervalle	$\|\mathcal{A}\|$	Moyenne avions/heure	Mix d'avions		
				H	M	L
`data_7_11.csv`	07:00-11:00	185	46.25	74	74	37
`data_11_15.csv`	11:00-15:00	207	51.75	83	83	41
`data_15_19.csv`	15:00-19:00	203	50.75	80	82	41
`data_19_23.csv`	19:00-23:00	167	41.75	67	67	33

TABLE 4.2 – Description des quatre base de données.

Nous résumons dans la table 4.3 les caractéristiques de chacune des 12 instances de référence : son intervalle temporel, le nombre d'avions par heure dans l'instance, qui reflète sa congestion, et le nombre d'avions de chaque catégories de turbulences de sillage. Dans la table 4.3, nous constatons que les instances les plus congestionnées sont `alp_15_30.csv` et `alp_15_40.csv`. Elles ont toutes les deux un nombre d'avions par heure égal à 60. Cela peut correspondre à des scénarios réels de forte demande. En effet, pendant les heures de congestion, les avions peuvent apparaître dans l'horizon de la planification toutes les 50 ou 60

secondes [85]. Les instances les moins congestionnées sont `alp_19_30.csv` et `alp_19_40.csv`, qui ont toutes les deux environ le même nombre d'avions par heure (≈ 36).

Origine de l'instance	Nom de l'instance	Intervalle	$\|\mathcal{A}\|$	Nombre avions/heure	Mix d'avions H	M	L
	`alp_7_30.csv`	`07:00-07:40`	30	45.0	13	11	6
`data_7_11.csv`	`alp_7_40.csv`	`07:00-07:59`	40	40.68	17	14	9
	`alp_7_50.csv`	`07:00-08:10`	50	42.86	22	17	11
	`alp_11_30.csv`	`11:00-11:45`	30	40.0	7	16	7
`data_11_15.csv`	`alp_11_40.csv`	`11:00-11:51`	40	47.06	11	18	11
	`alp_11_50.csv`	`11:00-11:58`	50	51.72	17	21	12
	`alp_15_30.csv`	`15:00-15:30`	30	60.0	15	11	4
`data_15_19.csv`	`alp_15_40.csv`	`15:00-15:40`	40	60.0	19	14	7
	`alp_15_50.csv`	`15:00-15:51`	50	58.82	22	19	9
	`alp_19_30.csv`	`19:00-19:50`	30	36.0	8	17	5
`data_19_23.csv`	`alp_19_40.csv`	`19:00-20:05`	40	36.92	9	21	10
	`alp_19_50.csv`	`19:00-20:15`	50	40.0	15	25	10

TABLE 4.3 – Description des nouvelles instances.

Enfin, nous présentons dans la table 4.4 un exemple de ces instances de référence. Il s'agit de l'instance `alp_15_30.csv`, qui contient $|\mathcal{A}| = 30$ avions. Dans la table 4.4, nous utilisons la notation "XXXX" pour désigner le modèle d'avions de type *Light*, qui ont été ajoutés artificiellement aux bases de données. Les colonnes "cost_300", "cost_900", "cost_1800" et "cost_3600" de cette table représentent les pentes de la fonction coût (en euros/seconde) pour 0 à 300 secondes, 300 à 900 secondes, 900 à 1800 secondes et 1800 à 3600 secondes de retard respectivement. Dans l'étude numérique de nos modèles PLNE, nous fixons, pour chaque avion $i \in \mathcal{A}$: $L_i = T_i + 3600$ (au plus une heure de retard). Lorsque nous considérons des coûts de retard linéaires, nous prenons pour chaque avion sa valeur dans la colonne 'cost_300' comme coût (par seconde) de retard. En revanche, quand la fonction coût considérée est linéaire par morceaux, nous considérons les trois points de rupture suivants : $T_i + 300$, $T_i + 900$ et $T_i + 1800$ secondes. Les coûts (par seconde) sur chaque morceaux correspondent respectivement aux colonnes 'cost_300', 'cost_900', 'cost_1800' et 'cost_3600'.

4.4 Étude numérique

Dans cette section, nous présentons les résultats d'implémentation des deux modèles PLNE : (4.9)–(4.17) et (4.23)–(4.28). Nous utilisons pour la suite les notations 'mod_lf' et 'mod_plf' pour désigner respectivement le modèle linéaire, (4.9)–(4.17), et le modèle linéaire par morceaux, (4.23)–(4.28). Ces deux formulations ont été implémentées avec DoCplex, l'interface Python du logiciel CPLEX. La résolution est ensuite faite par CPLEX, version 12.8.

Tous les tests de cette section ont été effectués sur un ordinateur personnel sous système

mdl	category	sta	sta_s	cost_300	cost_900	cost_1800	cost_3600
B738	Medium	15:00:00	54000	0.83	1.28	2.1	4.38
B738	Medium	15:00:00	54000	0.83	1.28	2.1	4.38
B744	Heavy	15:03:00	54180	2.37	3.42	5.58	11.79
A320	Medium	15:05:00	54300	0.83	1.25	2.02	4.19
A320	Medium	15:05:00	54300	0.83	1.25	2.02	4.19
B744	Heavy	15:05:00	54300	2.37	3.42	5.58	11.79
A343	Heavy	15:07:00	54420	1.63	2.39	3.94	8.42
B772	Heavy	15:09:00	54540	1.83	2.68	4.41	9.35
A321	Medium	15:10:00	54600	0.93	1.42	2.37	4.97
B738	Medium	15:10:00	54600	0.83	1.28	2.1	4.38
B772	Heavy	15:12:00	54720	1.83	2.68	4.41	9.35
XXXX	Light	15:13:00	54780	0.12	0.19	0.33	0.75
A320	Medium	15:15:00	54900	0.83	1.25	2.02	4.19
XXXX	Light	15:15:00	54900	0.12	0.19	0.33	0.75
B738	Medium	15:15:00	54900	0.83	1.28	2.1	4.38
A333	Heavy	15:15:00	54900	1.57	2.31	3.79	8.04
A343	Heavy	15:16:00	54960	1.63	2.39	3.94	8.42
B772	Heavy	15:17:00	55020	1.83	2.68	4.41	9.35
A319	Medium	15:20:00	55200	0.73	1.12	1.8	3.68
B744	Heavy	15:20:00	55200	2.37	3.42	5.58	11.79
A333	Heavy	15:20:00	55200	1.57	2.31	3.79	8.04
B744	Heavy	15:20:00	55200	2.37	3.42	5.58	11.79
B744	Heavy	15:21:00	55260	2.37	3.42	5.58	11.79
B772	Heavy	15:22:00	55320	1.83	2.68	4.41	9.35
XXXX	Light	15:23:00	55380	0.12	0.19	0.33	0.75
A333	Heavy	15:23:00	55380	1.57	2.31	3.79	8.04
B77W	Heavy	15:24:00	55440	2.97	4.37	7.09	14.74
CRJX	Medium	15:25:00	55500	0.35	0.51	0.85	1.84
XXXX	Light	15:29:00	55740	0.12	0.19	0.33	0.75
A320	Medium	15:30:00	55800	0.83	1.25	2.02	4.19

TABLE 4.4 – Instance avec $|\mathcal{A}| = 30$ avions : `alp_15_30.csv`.

d'exploitation GNU/Linux, processeur Intel(R) Core(TM) i7-4700M avec 8 Go de RAM.

Les instances de référence utilisées dans cette section correspondent aux 12 instances de [55], disponibles dans le dossier `ikli_instances.zip`.

Les résultats d'implémentation des deux modèles présentés en sections 4.2.1 et 4.2.2 sont rapportés dans la section 4.4.1. Dans cette sous-section, nous évaluons l'approche PLNE décrite dans ce chapitre sur la base de plusieurs indicateurs de performance. Nous discutons également de l'influence du paramètre m, qui correspond au nombre maximal de changements de position, sur la qualité des solutions obtenues et sur les temps de calcul.

4.4.1 Résultats principaux pour les instances de référence

Dans cette partie, nous évaluons l'approche PLNE décrite dans la section 4.2, sur la base des deux indicateurs de performance suivants :

— le temps de calcul (qui sera noté *cpu*), en secondes, requis pour que le logiciel d'optimi-

sation (CPLEX) trouve une solution optimale, dans une limite de temps égale à 1000 secondes ;

— le *pourcentage d'amélioration* (%improv) en termes de valeur de fonction-objectif, par rapport à la valeur de la solution de base FCFS. Le pourcentage d'amélioration obtenu par une méthode, M, est calculé comme suit :

$$\%\text{improv (M)} = \frac{C_{\text{FCFS}} - C_{\text{M}}}{C_{\text{FCFS}}} \times 100, \tag{4.29}$$

où C_{FCFS} and C_{M} sont respectivement le coût de la séquence FCFS et celui de la solution fournie par la méthode M.

La solution de base FCFS a été construite, pour chaque instance, selon la procédure suivante :

Étape 1 : Tri des avions de l'instance selon l'ordre croissant des heures préférentielles d'atterrissage, T_i. Cette étape fixe la séquence que nous notons Π.
Le $i^{\text{ième}}$ élément de cette séquence est noté $\Pi(i)$. Par exemple, $\Pi(3) = 1$ veut dire que le 3^{e} avion dans la séquence Π est l'avion d'indice 1.

Étape 2 : Calcul des heures cibles d'atterrissage, $x_{\Pi(i)}$ ($i \in \mathcal{A}$). Étant donné que l'ordre est fixé et que nous minimisons le retard, nous faisons alors atterrir les avions au plus tôt après leurs dates préférentielles d'atterrissage T_i, tout en respectant les contraintes de séparation. Les dates cibles d'atterrissage sont calculées comme suit :

$$x_{\Pi(i)} = \begin{cases} T_{\Pi(i)}, & \text{si } i = 1, \\ \max(T_{\Pi(i)},\ x_{\Pi(i-1)} + S_{\Pi(i-1)\Pi(i)}), & \text{si } 2 \leq i \leq |\mathcal{A}|, \end{cases} \tag{4.30}$$

où $S_{\Pi(i-1)\Pi(i)}$ est la séparation au seuil de piste entre deux atterrissages consécutifs $\Pi(i-1)$ et $\Pi(i)$, présentée dans la table 4.1.

Nous rapportons dans la table 4.5 les résultats de la résolution du modèle `mod_lf` pour nos 12 instances de référence. Nous donnons également, pour chacune de ces instances, la taille du problème associé, en nombre total de variables et de contraintes. Étant donné que le nombre maximal de changements de position est un paramètre arbitraire, nous avons choisi dans cette simulation deux valeurs pour ce paramètre : $m = 2$ et $m = 3$ (correspondant respectivement à 2-CPS et 3-CPS dans la table 4.5). Dans cette table, les première et deuxième colonnes présentent respectivement le nom de l'instance et sa taille (nombre d'avions). La troisième et quatrième colonnes affichent respectivement le nombre de variables (binaires et continues) et le nombre de contraintes du problème associé. Les colonnes '2-CPS' et '3-CPS' donnent les résultats de la résolution de chaque instance en termes des deux indicateurs de performance introduits ci-dessus : le pourcentage d'amélioration '%improv' et le temps de calcul 'cpu'. Nous pouvons constater dans cette table que même avec des petites valeurs du paramètre m, la PLNE peut améliorer la solution de base FCFS. En effet, pour $m = 2$, la PLNE atteint un pourcentage d'amélioration de $27,20\%$ en moyenne, et de $34,01\%$ en moyenne pour $m = 3$. Cela signifie qu'en autorisant les avions à dévier de seulement 2 positions par rapport à leurs positions d'origine, on peut réduire les retards de $27,20\%$ en moyenne. Nous constatons également que la plupart des instances ne sont pas résolues à l'optimalité dans la limite de temps de 1000 secondes. De plus, seule l'instance `alp_19_30.csv` est résolue à l'optimalité dans des temps de calcul courts, pour les deux valeurs de m choisies.

Instance	$\|\mathcal{A}\|$	Nb. var	Nb. cons	2-CPS		3-CPS	
				%improv	cpu	%improv	cpu
`alp_7_30.csv`	30	900	870	21.20	982.35	27.83	1000.02
`alp_7_40.csv`	40	1600	1560	21.29	1000.00	33.42	1000.00
`alp_7_50.csv`	50	2500	2450	24.75	1000.00	31.93	1000.00
`alp_11_30.csv`	30	900	870	35.47	503.50	48.88	418.52
`alp_11_40.csv`	40	1600	1560	34.48	1000.00	46.02	1000.00
`alp_11_50.csv`	50	2500	2450	26.84	1000.00	33.44	1000.00
`alp_15_30.csv`	30	900	870	18.35	1000.00	21.87	1000.00
`alp_15_40.csv`	40	1600	1560	19.09	1000.00	23.12	1000.00
`alp_15_50.csv`	50	2500	2450	19.19	1000.00	21.78	1000.00
`alp_19_30.csv`	30	900	870	33.96	8.44	38.81	16.69
`alp_19_40.csv`	40	1600	1560	36.97	1000.00	42.09	1000.00
`alp_19_50.csv`	50	2500	2450	34.81	1000.00	38.94	1000.02
min				18.35	8.44	21.78	16.69
max				36.97	1000.00	48.88	1000.00
moyenne				27.20	874.52	34.01	869.60

TABLE 4.5 – Résultats du modèle PLNE (4.9)–(4.17) pour résoudre les 12 instances de référence, pour les deux valeurs du paramètre du CPS : $m = 2$ et $m = 3$.

4.4.2 Influence du CPS

Nous avons vu dans la section 2.5 que l'ALP est un problème NP-difficile. Par conséquent, résoudre le problème avec des méthodes exactes risque de se heurter à l'explosion combinatoire de l'ensemble des solutions pour les problèmes de grande taille. L'ajout des contraintes CPS réduit la complexité du problème, et potentiellement les temps de calcul. Pour évaluer l'impact du paramètre, m, du changement de positions contraint, sur la qualité des solutions obtenues, et déterminer ainsi une bonne valeur pour ce paramètre, nous avons procédé à des tests empiriques.

La figure 4.3 montre l'évolution du pourcentage d'amélioration (%improv) selon les valeurs de m, pour différentes instances. Ces instances sont obtenues en considérant les $|\mathcal{A}|$ premières lignes de données de l'ensemble `data_7_11.csv` de [55]. Nous constatons que le pourcentage d'amélioration peut atteindre 50% pour certaines instances, lorsque $m = 5$. Pour cette même valeur de m, le pourcentage d'amélioration est supérieur à 45% pour toutes les instances. Rappelons que le pourcentage d'amélioration est calculé par rapport à la séquence FCFS. Cela signifie qu'en autorisant les avions à dévier de 5 positions par rapport à leurs positions d'origine, on peut réduire les retards de 45% pour ces instances. Toutefois, autoriser les avions à beaucoup s'éloigner de la séquence FCFS n'impliquera pas nécessairement une amélioration. Par exemple, pour l'instance avec $|\mathcal{A}| = 22$, le pourcentage d'amélioration de $53,59\%$ pour $m = 10$, ne s'améliore pas avec des valeurs de m supérieures à 10. Il n'est donc pas utile d'augmenter m d'avantage, étant donné le surcoût en temps de calcul qu'une grande valeur de m implique.

La figure 4.4 montre l'évolution du temps de calcul sous CPLEX, pour différentes valeurs de m, et pour les instances présentées ci-dessus, tirées de l'ensemble de données `data_7_11.csv`.

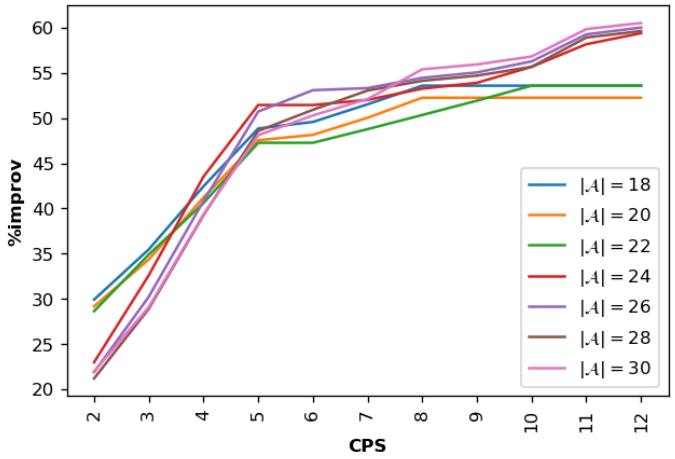


FIGURE 4.3 – Impact de différentes valeurs du CPS sur le pourcentage d'amélioration, pour différentes instances de tailles $|\mathcal{A}| = 18, 20, ..., 30$, générées à partir de l'ensemble `data_7_11.csv`.

Nous constatons que la PLNE permet de résoudre l'ALP dans des temps de calcul raisonnables, seulement pour des valeurs du paramètre m du CPS inférieures à 3. Au-delà de cette valeur, les temps de calcul deviennent prohibitifs. Concernant l'effet de saturation que nous observons, il est seulement dû à la limite de temps imposée sous CPLEX (1800 secondes). Il convient de mentionner que comme pour la PLNE, les algorithmes de programmation dynamique de la littérature requirent également des temps de calcul prohibitifs pour des valeurs de m supérieures à 3, comme montré dans [85].

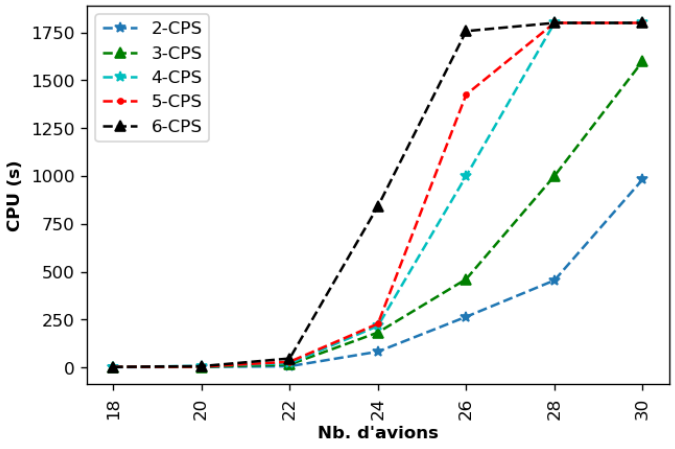


FIGURE 4.4 – Évolution du temps de calcul de l'approche PLNE, pour $m = 2, 3, ..., 6$.

4.4.3 Analyse des solutions obtenues avec les deux modèles

Pour les instances congestionnées, les avions risquent de subir des retards de plus de 300 secondes. C'est le cas par exemple de certains avions de l'instance `alp_15_50.csv`. En effet, si l'on résout le modèle `mod_lf` pour cette instance, pour un nombre maximal de changements de positions $m = 3$, certains avions risquent de subir des retards allant jusqu'à 3600 secondes

(voir figure 4.5). Par conséquent, il n'est pas pertinent de considérer un coût de retard linéaire pour cette instance.

Nous affichons dans la figure 4.5 la distribution des retards pour l'instance `alp_15_50.csv`, que nous avons résolue à l'aide de CPLEX, avec une limite de temps de 1000 secondes, en considérant la fonction coût linéaire (`mod_lf`) et la fonction coût linéaire par morceaux (`mod_plf`). Nous constatons que pour le modèle `mod_lf`, un avion de cette instance a subi un retard important (entre 1800 et 3600 secondes). De l'autre côté, avec le modèle `mod_plf`, aucun avion de cette instance n'a subi un tel retard.

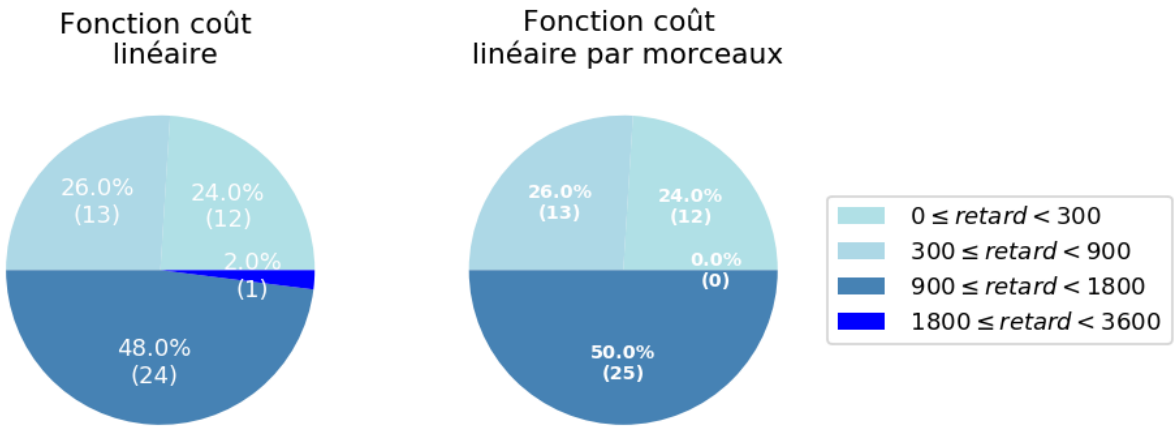


FIGURE 4.5 – Exemple de distribution des retards pour l'instance `alp_15_50.csv`, en considérant une fonction coût linéaire et linéaire par morceaux.

Pour mieux visualiser les solutions calculées par les deux modèles `mod_lf` et `mod_plf`, pour l'instance `alp_15_50.csv`, nous montrons dans la table 4.6 les séquences fournies par ces deux modèles, ainsi que la solution donnée par la règle du FCFS. Nous constatons que les solutions calculées par ces deux modèles sont différentes. De plus, 17 avions ont des positions différentes dans les deux solutions ; nous affichons en gras ces avions avec leurs dates cibles correspondantes. Par ailleurs, l'avion qui subi des retards importants avec le modèle `mod_lf` est l'avion 43, avec 2420 secondes de retard. Tandis qu'avec le modèle `mod_plf`, le retard de cet avion est de 1605 secondes.

Finalement, nous présentons dans la table 4.7 les résultats de la résolution du modèle `mod_plf` pour nos 12 instances de référence, pour les mêmes valeurs de m considérées pour le modèle `mod_lf` dans la table 4.5, et la même limite de temps de 1000 secondes. En comparant la table 4.7 à la table 4.5, qui correspond aux résultats du modèle `mod_lf`, nous constatons que pour toutes les instances résolues à l'optimalité avant la limite de temps, les temps de calcul du modèle `mod_plf` sont tous inférieurs à ceux du modèle `mod_lf`. Nous pouvons en déduire alors que considérer une fonction coût linéaire par morceaux n'augmente pas forcément les temps de calcul de la PLNE.

Conclusions du chapitre

Nous avons présenté dans ce chapitre une approche de programmation linéaire mixte en nombres entiers pour modéliser et résoudre le problème d'ordonnancement d'avions à l'atterrissage (ALP). Les contraintes considérées incluent la séparation, les fenêtres de temps et les contraintes de changement de positions. Notre objectif est de minimiser le coût du retard. Nous avons adapté notre modèle pour considérer une fonction coût convexe et linéaire par morceaux, qui rend compte de façon réaliste de la variation des coûts opérationnels des retards d'atterrissage. Nous avons présenté également dans ce chapitre la construction et mise à disposition publique de quatre ensembles de données servant de base pour générer des instances difficiles et réalistes pour l'ALP. Dans l'étude numérique de nos modèles PLNE, nous avons constaté que malgré l'ajout des contraintes de changement de positions, les temps de calcul de cette méthode restent prohibitifs sur certaines instances, et par conséquent, la résolution directe du modèle PLNE n'est pas adaptée à une application en temps réel.

Dans le chapitre suivant, nous présentons une nouvelle méthode heuristique pour résoudre l'ALP, qui permet de trouver des solutions de bonne qualité en des temps de calcul raisonnables.

FCFS			mod_lf			mod_plf		
Indice	mdl	Heure	Indice	mdl	Heure	Indice	mdl	Heure
1	B738	15 :00 :00	**1**	**B738**	**15 :00 :00**	**2**	**B738**	**15 :00 :00**
2	B738	15 :01 :09	**2**	**B738**	**15 :01 :09**	**1**	**B738**	**15 :01 :09**
3	B744	15 :03 :00	3	B744	15 :03 :00	3	B744	15 :03 :00
4	A320	15 :05 :37	**4**	**A320**	**15 :05 :37**	**5**	**A320**	**15 :05 :37**
5	A320	15 :06 :46	**5**	**A320**	**15 :06 :46**	**4**	**A320**	**15 :06 :46**
6	B744	15 :07 :46	6	B744	15 :07 :46	6	B744	15 :07 :46
7	A343	15 :09 :22	8	B772	15 :09 :22	8	B772	15 :09 :22
8	B772	15 :10 :58	7	A343	15 :10 :58	7	A343	15 :10 :58
9	A321	15 :13 :35	11	B772	15 :12 :34	11	B772	15 :12 :34
10	B738	15 :14 :44	9	A321	15 :15 :11	9	A321	15 :15 :11
11	B772	15 :15 :44	13	A320	15 :16 :20	10	B738	15 :16 :20
12	XXXX	15 :19 :00	10	B738	15 :17 :29	13	A320	15 :17 :29
13	A320	15 :20 :09	15	B738	15 :18 :38	15	B738	15 :18 :38
14	XXXX	15 :22 :20	14	XXXX	15 :20 :49	12	XXXX	15 :20 :49
15	B738	15 :23 :29	12	XXXX	15 :22 :11	14	XXXX	15 :22 :11
16	A333	15 :24 :29	18	B772	15 :23 :11	18	B772	15 :23 :11
17	A343	15 :26 :05	20	B744	15 :24 :47	20	B744	15 :24 :47
18	B772	15 :27 :41	16	A333	15 :26 :23	16	A333	15 :26 :23
19	A319	15 :30 :18	22	B744	15 :27 :59	22	B744	15 :27 :59
20	B744	15 :31 :18	17	A343	15 :29 :35	17	A343	15 :29 :35
21	A333	15 :32 :54	23	B744	15 :31 :11	23	B744	15 :31 :11
22	B744	15 :34 :30	19	A319	15 :33 :48	19	A319	15 :33 :48
23	B744	15 :36 :06	21	A333	15 :34 :48	21	A333	15 :34 :48
24	B772	15 :37 :42	27	B77W	15 :36 :24	27	B77W	15 :36 :24
25	XXXX	15 :40 :58	24	B772	15 :38 :00	24	B772	15 :38 :00
26	A333	15 :41 :58	26	A333	15 :39 :36	26	A333	15 :39 :36
27	B77W	15 :43 :34	25	XXXX	15 :42 :52	25	XXXX	15 :42 :52
28	CRJX	15 :46 :11	**30**	**A320**	**15 :44 :01**	**29**	**XXXX**	**15 :44 :14**
29	XXXX	15 :48 :22	**28**	**CRJX**	**15 :45 :10**	**30**	**A320**	**15 :45 :23**
30	A320	15 :49 :31	**31**	**CRJX**	**15 :46 :19**	**28**	**CRJX**	**15 :46 :32**
31	CRJX	15 :50 :40	**33**	**XXXX**	**15 :48 :30**	**31**	**CRJX**	**15 :47 :41**
32	A343	15 :51 :40	**29**	**XXXX**	**15 :49 :52**	**35**	**A321**	**15 :48 :50**
33	XXXX	15 :54 :56	32	A343	15 :50 :53	32	A343	15 :49 :50
34	A333	15 :55 :56	36	A343	15 :52 :29	36	A343	15 :51 :26
35	A321	15 :58 :33	34	A333	15 :54 :05	34	A333	15 :53 :02
36	A343	15 :59 :33	**39**	**A343**	**15 :55 :41**	**33**	**XXXX**	**15 :56 :18**
37	A320	16 :02 :10	**35**	**A321**	**15 :58 :18**	**37**	**A320**	**15 :57 :27**
38	XXXX	16 :04 :21	**37**	**A320**	**15 :59 :27**	**39**	**A343**	**15 :58 :27**
39	A343	16 :05 :21	**42**	**B738**	**16 :00 :36**	**40**	**XXXX**	**16 :01 :43**
40	XXXX	16 :08 :37	38	XXXX	16 :02 :47	38	XXXX	16 :03 :05
41	XXXX	16 :09 :59	40	XXXX	16 :04 :09	41	XXXX	16 :04 :27
42	B738	16 :11 :08	**41**	**XXXX**	**16 :05 :31**	**42**	**B738**	**16 :05 :36**
43	A318	16 :12 :17	**44**	**B772**	**16 :06 :31**	**43**	**A318**	**16 :06 :45**
44	B772	16 :13 :17	47	B77W	16 :08 :07	47	B77W	16 :07 :45
45	A343	16 :14 :53	**45**	**A343**	**16 :09 :43**	**44**	**B772**	**16 :09 :21**
46	B738	16 :17 :30	**43**	**A318**	**16 :12 :20**	**45**	**A343**	**16 :10 :57**
47	B77W	16 :18 :30	48	A321	16 :12 :20	48	A321	16 :13 :34
48	A321	16 :21 :07	46	B738	16 :13 :29	46	B738	16 :14 :43
49	CRJX	16 :22 :16	49	CRJX	16 :15 :47	49	CRJX	16 :15 :52
50	XXXX	16 :24 :27	50	XXXX	16 :17 :58	50	XXXX	16 :18 :03
Coût total linéaire		54963.15			43830.81			43688.75
Coût total linéaire par morceaux		95363.81			67626.07			66905.17

TABLE 4.6 – Comparaisons des solutions calculées par les deux modèles `mod_lf` (fonction coût linéaire) et `mod_plf` (fonction coût linéaire par morceaux).

Instance	$\|\mathcal{A}\|$	Nb. var	Nb. cons	2-CPS		3-CPS	
				%improv	cpu	%improv	cpu
`alp_7_30.csv`	30	1020	870	26.01	570.42	39.07	1000.00
`alp_7_40.csv`	40	1760	1560	30.97	1000.00	42.35	1000.00
`alp_7_50.csv`	50	2700	2450	38.28	1000.00	49.26	1000.00
`alp_11_30.csv`	30	1020	870	45.40	434.13	61.50	159.71
`alp_11_40.csv`	40	1760	1560	50.51	1000.00	63.60	1000.00
`alp_11_50.csv`	50	2700	2450	44.47	1000.00	54.11	1000.00
`alp_15_30.csv`	30	1020	870	33.15	1000.00	37.47	1000.00
`alp_15_40.csv`	40	1760	1560	35.22	1000.00	43.69	1000.00
`alp_15_50.csv`	50	2700	2450	38.43	1000.00	43.79	1000.00
`alp_19_30.csv`	30	1020	870	40.09	2.93	46.58	3.21
`alp_19_40.csv`	40	1760	1560	53.89	277.73	57.71	649.19
`alp_19_50.csv`	50	2700	2450	51.32	1000.00	54.68	1000.00
min				26.01	2.93	37.47	3.21
max				53.89	1000.00	63.60	1000.00
moyenne				40.64	773.76	49.48	817.67

TABLE 4.7 – Résultats du modèle `mod_plf` pour résoudre les 12 instances de référence, pour les deux valeurs de nombre maximal de changements de positions $m = 2$ (2-CPS) et $m = 3$ (3-CPS).

Chapitre 5

Approche de résolution par un algorithme de planification optimiste

Ce chapitre présente la troisième contribution de la thèse : une nouvelle méthode heuristique pour résoudre le problème d'ordonnancement d'avions à l'atterrissage (ALP), en prenant en considération les contraintes de changement de positions. L'objectif est de minimiser la somme des coûts des retards. Nous supposons ici que le coût des retards pour chaque avion est une fonction convexe et linéaire par morceaux. Nous nous intéressons au contexte particulier où l'on dispose de ressources de calcul finies (temps de calcul limité par exemple) pour trouver des solutions de bonne qualité. Notre méthode est basée sur l'algorithme de *planification optimiste* [49], introduit par Munos et Hren pour résoudre le problème de planification dans les processus de décision markoviens [86]. Nous présentons dans la section 5.1 le contexte initial pour lequel l'algorithme de planification optimiste a été proposé. Ensuite, nous détaillons dans la section 5.2 un nouveau modèle pour l'ALP, inspiré des processus de décision markoviens et notre méthode pour résoudre ce modèle, basée sur l'algorithme de planification optimiste. Enfin, nous comparons la performance de notre algorithme à celle des logiciels de programmation mathématique.

Ce chapitre reprend principalement notre article intitulé *An optimistic planning approach for the aircraft landing problem*, accepté au séminaire ENRI International Workshop on ATM/CNS (EIWAC2019) et sélectionné pour publication dans *Air Traffic Management and Systems IV, Lecture Notes in Electrical Engineering* [58].

Sommaire

5.1 Introduction à l'algorithme de planification optimiste

La planification consiste à décider à chaque pas de temps d'une action à entreprendre lors de l'interaction avec un système dynamique. Le plan peut être représenté sous forme séquentielle (une suite d'actions) ou sous forme d'un contrôle en boucle fermée (*politique*). Dans le contexte des processus de décision markoviens (*Markov Decision Process* – MDP), le plan optimal est celui qui maximise un critère quantitatif correspondant à l'accumulation d'un signal de *récompenses*. L'algorithme de planification optimiste [49] permet de résoudre ce problème de planification, étant donné des ressources de calcul finies (temps de calcul limité par exemple). Avant de décrire l'algorithme de planification optimiste, nous présentons d'abord le contexte dans lequel il a été proposé et nous définissons les termes qui lui sont associés.

5.1.1 Contexte

L'algorithme de planification optimiste a été introduit pour la première fois par Hren et Munos [49] dans le cadre de la thèse de Hren [48] afin de résoudre un problème de planification dans un processus de décision markovien déterministe. Formellement, un processus de décision markovien est défini par un quadruplet (S, A, T, r) où :

- S est un ensemble d'*états*, qui représente le système à un instant donné. Il peut être fini, dénombrable ou continu. Si nous considérons par exemple que le système étudié est une voiture, l'état contiendrait par exemple la position de la voiture et sa vitesse.
- A est un ensemble d'*actions* définissant les interactions possibles avec le système. Il peut être aussi fini on continu. Si nous reprenons l'exemple de la voiture, les actions possibles seraient par exemple l'accélération ou le freinage.
- $T : S \times A \times S \to [0, 1]$ est une fonction de *transition* qui définit l'effet des actions sur le système. Étant donnés un état et une action, elle définit la probabilité de transiter vers un état résultant. Cette fonction peut aussi être déterministe (un seul état résultant).
- $r : S \times A \times S \to \mathbb{R}$ est la fonction *récompense* qui représente ce qu'on gagne lors de la prise d'une décision d'action permettant de passer d'un état à un autre. Elle peut être déterministe.

Dans les processus de décision markoviens, on suppose souvent que les récompenses diminuent avec le temps, suivant un *facteur de dépréciation* $\gamma \in [0, 1[$. L'objectif de la planification dans un processus de décision markovien est de trouver, à partir de n'importe quel état, l'action à effectuer afin de maximiser la somme (dépréciée) des récompenses futures. L'algorithme de planification optimiste [49] s'intéresse au contexte particulier où le nombre d'états est très grand, possiblement infini et que l'on dispose de ressources de calcul finies pour trouver cette action. Il propose alors une stratégie d'exploration optimiste, qui, une fois les ressources de calcul épuisées, fournit l'action qui soit la plus « proche » possible de l'action optimale. Cette proximité de l'action optimale est évaluée en terme de *regret*, qui représente la différence de qualité entre l'action optimale et l'action proposée par l'algorithme pour un état donné. L'algorithme a été initialement proposé pour les processus de décision markoviens déterministes,

c'est-à-dire avec des fonctions de transition et des récompenses déterministes. Il a ensuite été généralisé dans [81], pour le cas des processus de décision markoviens à transitions et récompenses stochastiques.

5.1.2 Description de l'algorithme

Supposons que l'on dispose d'un modèle génératif du processus de décision markovien étudié, c'est-à-dire d'un mécanisme pour générer les transitions et les récompenses. Supposons de plus, sans perte de généralité, que toute récompense obtenue avec ce modèle génératif appartient à l'intervalle $[0, 1]$. Fixons un état initial et considérons les séquences d'états (trajectoires) réalisables à partir de cet état initial. Considérons enfin un arbre, dans lequel les nœuds correspondent aux états et les arcs aux actions et récompenses. Dans cet arbre, le nœud racine correspond à l'état initial et tout nœud de profondeur d correspond à un état obtenu après une séquence de d actions effectuées depuis l'état initial. On définit pour chaque nœud une *fonction d'utilité* qui est une mesure de la qualité de ce nœud ; cette fonction d'utilité est la somme des récompenses obtenues du nœud racine au nœud courant plus une borne supérieure sur la somme des récompenses futures. Le problème que l'on cherche à résoudre est le suivant : comment explorer l'arbre décrit ci-dessus de façon à proposer, une fois les ressources de calcul expirées, une action (ou une séquence d'actions) qui soit la plus proche possible de l'action (ou de la séquence d'actions) optimale. L'idée de l'algorithme de planification optimiste consiste à explorer à chaque itération les nœuds les plus prometteurs, c'est-à-dire les nœuds ayant la plus grande borne supérieure sur la somme des récompenses (d'où le terme « optimiste »). Nous définissons ci-dessous quelques notations nécessaires pour la présentation de l'algorithme.

Considérons l'arbre, $\mathcal{T}$, composé du nœud racine et de tous les nœuds atteignables (via une séquence d'action) depuis ce dernier. On dira qu'un nœud est *étendu* quand des ressources de calcul ont été allouées au calcul des transitions vers ses enfants. Notons $\mathcal{T}_n$ l'ensemble des nœuds déjà étendus jusqu'à l'itération n et $\mathcal{S}_n$ l'ensemble des nœuds susceptibles d'être étendus à l'itération suivante (*leaf nodes*). La figure 5.1 illustre un exemple. Nous définissons pour chaque nœud i appartenant à $\mathcal{S}_n$ la somme des récompenses (dépréciées par un facteur $\gamma \in [0, 1[$) obtenues le long du chemin depuis la racine jusqu'au nœud i, cette somme est notée g_i et est définie de manière récursive pour les nœuds $i \in \mathcal{T}_n$: $g_i = \max_{j \in \{\text{enfants de } i\}} g_j$. Finalement, nous associons à chaque nœud i appartenant à $\mathcal{T}_n \cup \mathcal{S}_n$, de profondeur d_i, une *estimation*, notée v_i, qui est une mesure de la qualité de ce nœud. Plus précisément, $v_i = g_i + h_i$, où h_i est une heuristique estimant une borne supérieure de la somme des récompenses futures qui peuvent être obtenues depuis le nœud i. Dans [49], $h_i = \frac{\gamma^{d_i}}{1-\gamma}$ pour tout i. Afin de voir qu'il s'agit bien d'une borne supérieure, rappelons d'abord que toutes les récompenses sont dans $[0, 1]$ et que le facteur de dépréciation, γ, est dans $[0, 1[$. La somme des récompenses futures depuis le nœud de profondeur d_i est $\sum_{n \geq d_i} r_n \gamma^n$. Une borne supérieure sur cette somme est alors $\sum_{n \geq d_i} \gamma^n = \gamma^{d_i} + \gamma^{d_i+1} + ... = \frac{\gamma^{d_i}}{1-\gamma}$.

La stratégie de planification optimiste est présentée dans l'algorithme 1. L'arbre $\mathcal{T}$ est exploré progressivement, en choisissant d'abord les nœuds les plus prometteurs, afin d'identifier le plus rapidement possible une bonne branche avant l'épuisement des ressources de calcul.

L'algorithme étend d'abord le nœud racine, puis il construit les deux ensembles $\mathcal{T}_0$ et $\mathcal{S}_0$: le nœud racine est ajouté à l'ensemble $\mathcal{T}_0$ des nœuds déjà étendus et ses enfants sont ajoutés à l'ensemble $\mathcal{S}_0$ des nœuds susceptibles d'être étendus à l'itération suivante. Ensuite, à chaque itération n, l'algorithme étend un nœud $i \in \mathcal{S}_{n-1}$ possédant la plus grande valeur v_i, c'est-à-dire le nœud ayant la plus grande borne supérieure sur les valeurs des nœuds. Finalement, une fois le budget d'expansion de nœuds épuisé, l'action renvoyée par l'algorithme est celle qui a permis de générer le *meilleur* nœud-fils du nœud racine, au sens de g ; autrement dit, l'action qui correspond à $\underset{j \in \{\text{enfants du nœud racine}\}}{\operatorname{argmax}} g_j$.

Les auteurs montrent que le regret de l'algorithme 1 est borné supérieurement par $\frac{\gamma^{d^{max}}}{1-\gamma}$, où d^{max} correspond à la profondeur maximale des nœuds dans $\mathcal{T}_n$. Par conséquent, l'action renvoyée par leur algorithme n'est jamais pire que l'action trouvée par un algorithme dit de planification uniforme (*breadth-first search*), disposant des mêmes ressources de calcul.

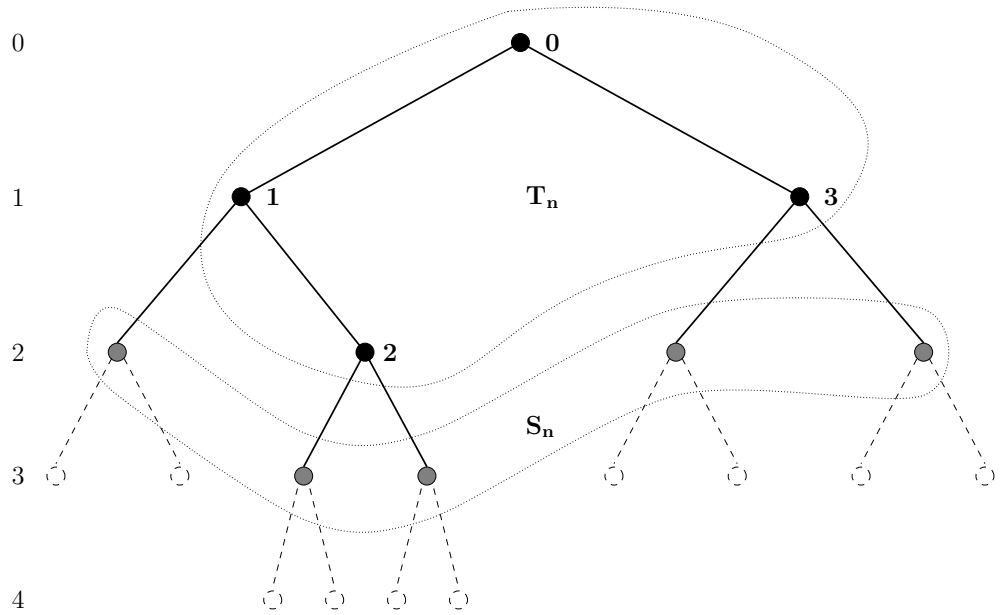


FIGURE 5.1 – Exemple d'un arbre binaire (deux enfants par nœud) avec l'ensemble $\mathcal{T}_n$ des nœuds déjà étendus (disques noirs) et de l'ensemble $\mathcal{S}_n$ des nœuds susceptibles d'être étendus (disques gris) à l'itération $n = 3$. Chaque nœud de $\mathcal{T}_n$ est étiqueté par le numéro de l'itération à laquelle il a été étendu ($0, 1, 2$ ou 3). La profondeur des nœuds est indiquée à gauche de la figure.

5.2 Présentation de l'approche

Le problème d'ordonnancement d'avions à l'atterrissage (ALP) est souvent formulé dans la littérature comme un modèle linéaire mixte en nombres entiers. Dans cette section, nous proposons d'abord un nouveau modèle pour l'ALP, inspiré des processus de décision markoviens. Ensuite, nous expliquons notre approche de planification optimiste basée sur l'algorithme 1

Algorithme 1 : Planification optimiste

$n \leftarrow 0$
Étendre le nœud racine
$\mathcal{T}_0 \leftarrow \{\text{nœud racine}\}$
$\mathcal{S}_0 \leftarrow \{\text{enfants du nœud racine}\}$
tant que *des ressources de calcul sont disponibles* **faire**
  $n \leftarrow n + 1$
  Étendre le nœud $i \in \mathcal{S}_{n-1}$ ayant la plus grande valeur v_i
  $\mathcal{T}_n \leftarrow \mathcal{T}_{n-1} \cup \{i\}$
  $\mathcal{S}_n \leftarrow \mathcal{S}_{n-1} \setminus \{i\} \cup \{\text{enfants de i}\}$
fin
Retourner l'action $\underset{j \in \{\text{enfants du nœud racine}\}}{\operatorname{argmax}} g_j$

pour résoudre l'ALP. Finalement, nous présentons différentes heuristiques d'estimation, adaptées à l'ALP, pour calculer les valeurs des nœuds.

Dans la suite de cette section, nous gardons les mêmes notations que dans le chapitre précédent : $\mathcal{A}$ est l'ensemble d'avions à ordonnancer, m est le nombre maximal de changements de position et f_i est la fonction coût d'un $i \in \mathcal{A}$. La notation Π désigne une séquence réalisable (permutation de l'ensemble d'avions). Rappelons que le i^{e} élément de la séquence Π est noté $\Pi(i)$. Par exemple, $\Pi(3) = 1$ veut dire que le 3^{e} avion dans la séquence Π est l'avion d'indice 1. Nous supposons également ici, sans perte de généralité, que pour chaque avion $i \in \mathcal{A}$, son indice représente également sa position dans la séquence FCFS.

5.2.1 Description du modèle

Comme pour un processus de décision markovien, notre modèle mathématique est composé d'états, d'actions, de transitions et de coûts instantanés. Le coût instantané d'une transition est l'équivalent de la récompense dans les MDPs. Nous montrons que ce modèle peut également être vu comme un arbre de recherche.

États

Chaque *état*, noté s, représente une partition de l'ensemble des avions, $\mathcal{A}$, en $\bar{I}$ et I :

— $\bar{I}$ est l'ensemble des indices des avions ayant déjà atterri.

— I est l'ensemble des indices des avions qui n'ont pas encore atterri.

Notons $\bar{\Pi}$ la séquence des avions dans l'ensemble $\bar{I}$. L'état s est défini comme le vecteur : $s = (\bar{\Pi}, x_{\bar{\Pi}(|\bar{I}|)}, I)$, où $x_{\bar{\Pi}(|\bar{I}|)}$ est la date du dernier avion posé dans l'ensemble $\bar{I}$.

Considérons un exemple avec $\mathcal{A} = \{1, 2, 3, 4, 5\}$ et supposons que les avions 1 puis 3 ont déjà atterri, c'est-à-dire : $\bar{I} = \{1, 3\}$ et $\bar{\Pi} = [1, 3]$, $I = \{2, 4, 5\}$. L'état actuel s est :

$$\begin{aligned} s &= ([1,3], x_{\bar{\Pi}(2)}, \{2,4,5\}) \\ &= ([1,3], x_3, \{2,4,5\}) \end{aligned}$$

Notons que l'état initial se distingue par une composante séquence $\bar{\Pi} = []$ (aucun avion n'a atterri) et une composante $I = \mathcal{A}$. À l'inverse, un état terminal se distingue par une composante séquence $\bar{\Pi}$ contenant tous les avions de $\mathcal{A}$ et une composante $I = \{\}$ (tous les avions ont atterri).

Actions

Chaque *action* est un indice d'avion $i \in I$ que nous décidons de faire atterrir, tout en respectant les contraintes de CPS.

Reprenons l'exemple précédent avec $\bar{\Pi} = [1, 3]$ et $I = \{2, 4, 5\}$ ($s = ([1, 3], x_3, \{2, 4, 5\})$) et supposons que $m = 1$, c'est-à-dire que les avions ne peuvent dévier de leur position d'origine que d'au plus une position. Alors dans cet exemple, le seul avion que nous pouvons faire atterrir depuis cet état, sans violer les contraintes de CPS, est l'avion d'indice 2. Par contre, si nous supposons que $m = 2$, les indices d'avions que nous pouvons faire atterrir tout en respectant les contraintes de CPS sont alors : $2, 4$ et 5.

Transitions

Chaque *transition* correspond à une mise à jour de la séquence $\bar{\Pi}$ et de l'ensemble I. En effet, si nous exécutons l'action $i \in I$ à partir d'un état donné $s = (\bar{\Pi}, x_{\bar{\Pi}(|\bar{I}|)}, I)$, le système génère l'état unique suivant, noté s', correspondant aux ensembles I' et $\bar{I}'$ suivants :

$$I' = I \setminus \{i\} \text{ et } \bar{I}' = \bar{I} \cup \{i\}, \quad \text{(l'avion } i \text{ a atterri)}.$$

Le nouvel état s' est alors défini par : $s' = (\bar{\Pi}', x_i, I')$, où $\bar{\Pi}'$ est la séquence $\bar{\Pi}$ à laquelle on ajoute l'avion i. Le coût de la transition correspond au coût de retard du dernier avion posé (l'avion i).

Coûts

Quand le système transite d'un état s à un nouvel état s', le coût instantané de la transition est l'équivalent de la récompense dans les MDPs. Ce coût instantané représente le coût de retard du dernier avion posé dans $\bar{\Pi}'$, c'est-à-dire $f_{\bar{\Pi}(|\bar{I}'|)}(x_{\bar{\Pi}(|\bar{I}'|)})$. En outre, nous définissions le coût estimé du nouvel état, qui, comme la valeur v présentée dans la section 5.1.2, est une mesure de la qualité de l'état. Ce coût estimé, noté c, est défini par :

$$c(s') = g(\bar{I}') + h(I'), \tag{5.1}$$

où

— $g(\bar{I}')$ est le coût total de la séquence $\bar{\Pi}'$, c'est-à-dire la somme des coûts instantanés de l'état initial jusqu'à l'état courant s' : $\sum_{i \in \bar{I}'} f_i(x_i)$.

— $h(I')$ est une heuristique qui estime le coût d'une séquence optimale d'atterrissages des avions de I', respectant les contraintes CPS.

Étant donné que nous minimisons le retard, nous faisons alors atterrir les avions le plus tôt possible après leurs dates préférentielles d'atterrissage. Par conséquent, les dates

cibles d'atterrissage pour chaque avion i de $\bar{I}'$ sont :

$$x_{\bar{\Pi}'(i)} = \begin{cases} T_{\bar{\Pi}'(i)} & \text{si } i = 1, \\ \max(T_{\bar{\Pi}'(i)},\ x_{\bar{\Pi}'(i-1)} + S_{\bar{\Pi}'(i-1)\bar{\Pi}'(i)}) & \text{si } 2 \leq i \leq |\bar{I}'|. \end{cases} \tag{5.2}$$

Si le coût est supposé linéaire, le coût de retard du i^{ieme} avion de la séquence $\bar{\Pi}'$ est alors : $c_{\bar{\Pi}'(i)}(x_{\bar{\Pi}'(i)} - T_{\bar{\Pi}'(i)})$.

Si les coûts sont supposés convexes et linéaires par morceaux, le coût de retard du i^{ieme} avion de la séquence $\bar{\Pi}'$ est $f_{\bar{\Pi}'(i)}(x_{\bar{\Pi}'(i)})$, qui peut être calculé en utilisant les équations (4.18)-(4.22) vues au chapitre précédent.

À la fin de cette section, nous proposons des candidats pour l'heuristique d'estimation h : la règle FCFS et des heuristiques d'estimation apprises par des algorithmes d'apprentissage automatique.

Si nous fixons un état initial et considérons les séquences d'états réalisables à partir de cet état initial, nous obtenons un arbre, dans lequel les nœuds correspondent aux états, les arcs aux transitions avec coûts instantanés associés. Cet arbre est constitué du nœud racine qui correspond à l'état initial (aucun avion n'est posé), puis de tous les nœuds atteignables depuis ce dernier. Chaque nœud de cet arbre est caractérisé par l'état auquel il correspond, ainsi que le coût estimé qui lui est associé (le coût c défini ci-dessus). Les nœuds terminaux correspondent à des états dans lesquels tous les avions sont posés.

L'ALP peut alors être vu comme un problème de parcours d'arbre, où l'on cherche le nœud de profondeur $d = |\mathcal{A}|$ le moins coûteux à atteindre depuis le nœud racine. Nous décrivons ci-dessous notre algorithme de planification optimiste, basé sur l'algorithme 1.

5.2.2 Algorithme

L'algorithme de planification optimiste que nous proposons est détaillé dans l'algorithme 2. De façon similaire à l'algorithme 1, l'algorithme 2 part du nœud racine dans lequel aucun avion n'a atterri. Au début, l'algorithme étend le nœud racine, puis il initialise les deux ensembles $\mathcal{T}_0$ et $\mathcal{S}_0$: le nœud racine est ajouté à l'ensemble $\mathcal{T}_0$ et ses enfants sont ajoutés à l'ensemble $\mathcal{S}_0$. Ensuite, à chaque itération, l'algorithme étend le nœud le moins coûteux de l'ensemble des feuilles $\mathcal{S}_{n-1}$, en se basant sur une *évaluation optimiste* des coûts des nœuds. Cette évaluation implique les deux valeurs g et h définies ci-dessus. La première calcule le coût de la séquence d'avions déjà posés, c'est-à-dire le coût du chemin depuis la racine jusqu'au nœud courant. La deuxième valeur h estime le coût le plus faible parmi toutes les séquences possibles du sous-ensemble d'avions qui n'ont pas encore atterri. L'algorithme 2 continue d'explorer cet arbre tant que les ressources de calcul sont disponibles et tant qu'il n'a pas atteint le dernier état terminal. Si la condition d'arrêt est satisfaite (ressources de calcul épuisées ou dernier état terminal atteint), l'algorithme renvoie le nœud le moins coûteux, au sens de la fonction coût c, parmi les nœuds de l'ensemble $\mathcal{S}_n$. Deux cas de figure se présentent. Si le nœud renvoyé est un nœud terminal : il correspond à un état dans lequel tous les avions ont atterri. Dans ce cas, la séquence solution est construite depuis ce nœud, en retraçant ses parents jusqu'au nœud racine. Si le nœud retourné n'est pas un nœud termi-

nal, nous ne disposons que d'une solution partielle. Dans ce cas, la solution est complétée en faisant atterrir les avions qui n'ont pas encore atterri dans l'ordre FCFS, pour obtenir ainsi une solution complète (voir l'exemple de la section 5.2.3).

Algorithme 2 : Construction de l'arbre de planification optimiste

$n \leftarrow 0$
Étendre le nœud racine
$\mathcal{T}_0 \leftarrow \{\text{nœud racine}\}$
$\mathcal{S}_0 \leftarrow \{\text{enfants du nœud racine}\}$
$d \leftarrow 1$ (la plus petite profondeur des nœuds dans $\mathcal{S}_n$)
tant que *condition d'arrêt pas satisfaite* **faire**
  $n \leftarrow n + 1$
  Étendre le nœud $i \in \mathcal{S}_{n-1}$ ayant le plus petit coût c_i
  $\mathcal{T}_n \leftarrow \mathcal{T}_{n-1} \cup \{i\}$
  $\mathcal{S}_n \leftarrow \mathcal{S}_{n-1} \setminus \{i\} \cup \{\text{enfants de i}\}$
  $d \leftarrow$ la plus petite profondeur des nœuds dans $\mathcal{S}_n$
fin
retourner nœud $i \in \mathcal{S}_n$ ayant le plus petit coût c_i. La solution est construite en retraçant les parents de ce nœud jusqu'au nœud racine.

5.2.3 Exemple d'illustration

Nous illustrons dans la figure 5.2 un exemple d'arbre construit lors des quatre premières itérations de l'algorithme 2, où $\mathcal{A} = \{1, 2, 3, 4, 5, 6\}$ et le nombre maximal de changements de position est égal à 1. Dans cette illustration, les nœuds sont désignés par les états qu'ils représentent. Plus précisément, la séquence $\bar{\Pi}$, l'ensemble I et la date d'atterrissage du dernier avion de $\bar{\Pi}$. Le coût estimé de chaque nœud est affiché en rouge. Les valeurs des coûts et des dates d'atterrissage de cette illustration ont été choisies de façon arbitraire, à titre d'exemple.

Au début de l'algorithme (itération $n = 0$), le nœud racine, caractérisé par $\bar{\Pi} = []$ et $I = \mathcal{A}$, est étendu. Les nœuds en pointillés représentent les nœuds éligibles à être étendus à l'itération suivante, c'est-à-dire les nœuds de l'ensemble $\mathcal{S}_n$. À chaque itération $n > 0$, le nœud de $\mathcal{S}_{n-1}$ ayant le plus petit coût c est étendu. L'algorithme continue jusqu'à ce que les ressources de calcul soient épuisées, ce qui correspond à l'itération $n = 4$ dans notre exemple. L'algorithme renvoie alors le nœud le moins coûteux de l'ensemble $\mathcal{S}_n$, qui correspond au nœud caractérisé par $\bar{\Pi} = [2, 1, 3, 4]$, $x_4 = 3$ et $I = \{5, 6\}$ (sur fond orange). L'algorithme remonte ensuite ses parents jusqu'au nœud racine, pour obtenir les dates d'atterrissage des avions qui ont déjà atterri. Étant donné que l'algorithme s'arrête avant de pouvoir obtenir une solution complète dans laquelle tous les avions sont posés, nous la complétons en faisant atterrir les avions restants (les avions 5 et 6) dans l'ordre FCFS.

5.2.4 Comparaison avec l'algorithme A*

Dans le domaine de l'intelligence artificielle, l'algorithme A* [46] est un algorithme de recherche de plus court chemin dans un graphe entre un nœud initial et un nœud final,

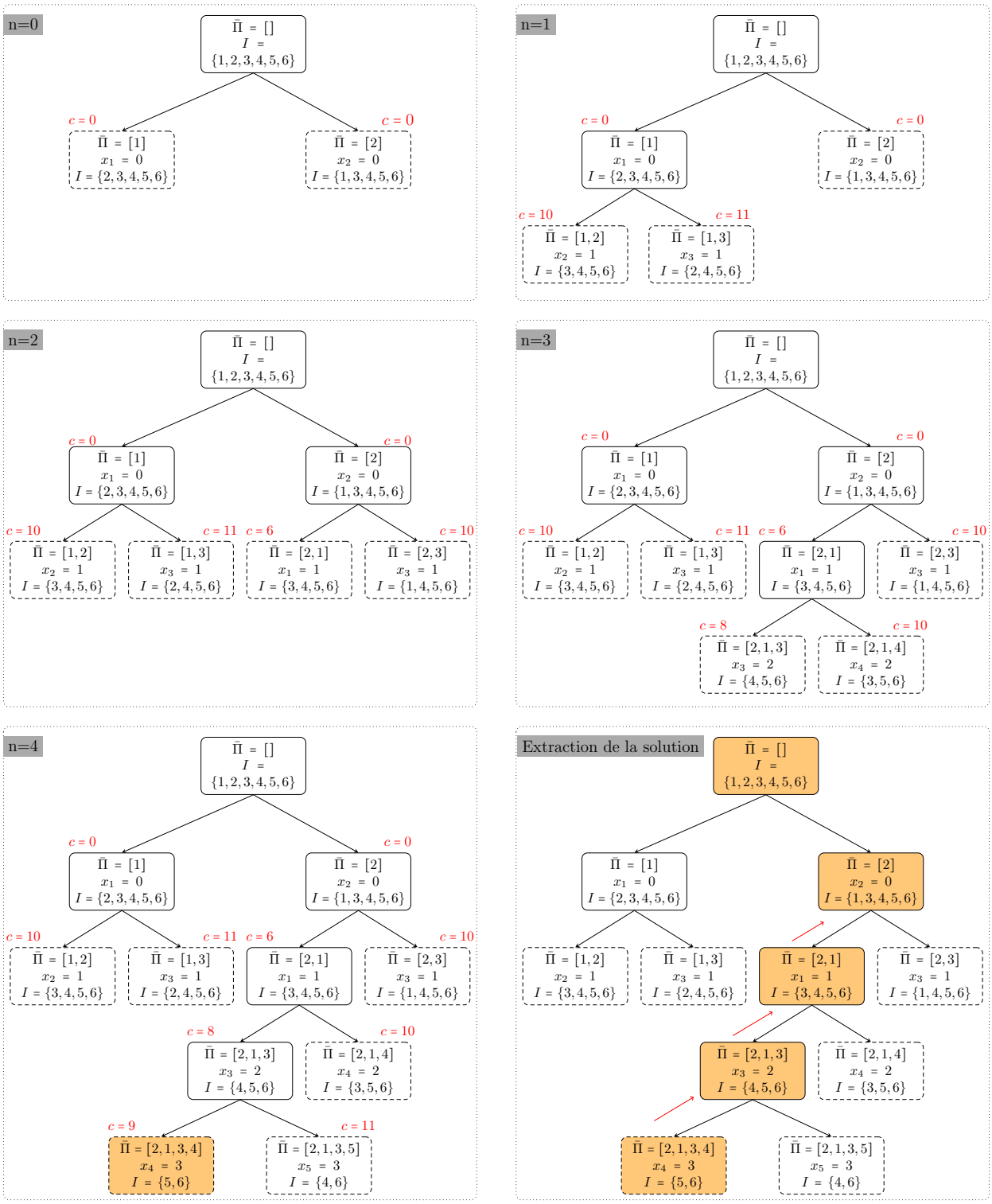


FIGURE 5.2 – Illustration de l'arbre construit à chaque itération de l'algorithme 2, pour $n = 0, 1, ..., 4$. Ici, $|\mathcal{A}| = 6$, $m = 1$ et la limite des ressources de calcul est atteinte à l'itération $n = 4$.

tous les deux connus. Cet algorithme de parcours de graphe trouve le plus court chemin en utilisant le coût du chemin déjà parcouru et une estimation du coût restant jusqu'au nœud final. Si nous considérons un graphe défini par un ensemble de nœuds, $\mathcal{N}$, et un ensemble

d'arcs, l'algorithme A* attribue à chaque nœud $n \in \mathcal{N}$, le coût :

$$c(n) = g(n) + h(n),$$

où $g(n)$ est le coût cumulatif du nœud initial jusqu'au nœud n et $h(n)$ est une estimation du coût restant jusqu'au nœud terminal. L'algorithme A* s'arrête dès qu'il rencontre le nœud final. Quand $h(n)$ est admissible, c'est-à-dire qu'elle ne surestime jamais le coût optimal du nœud courant jusqu'au nœud final, le chemin parcouru correspond au chemin le plus court. Quand $h(n) = 0, n \in \mathcal{N}$, l'algorithme A* correspond alors à l'algorithme de Dijkstra.

L'algorithme A* présente des similarités avec l'algorithme de planification optimiste, notamment dans le calcul des coûts des nœuds. En effet, pour calculer les coûts des nœuds, les deux algorithmes se basent sur un coût cumulatif (du nœud de départ au nœud courant) et sur une heuristique d'estimation du coût restant (jusqu'au nœud final). Cependant, cette heuristique n'est pas la même pour les deux algorithmes. Dans le cas de A*, un nœud final est obligatoire pour le calcul de l'heuristique, tandis que pour l'algorithme de planification optimiste, un nœud final n'est pas nécessaire pour le calcul de cette heuristique. Par exemple, dans le cas des processus de décision markoviens considérés dans [49], l'heuristique utilise la valeur $\frac{\gamma^d}{1-\gamma}$, qui constitue une borne supérieure de la somme des récompenses dépréciées accessibles depuis le nœud courant.

Par ailleurs, d'autres aspects distinguent un algorithme de planification optimiste de l'algorithme A*. Mentionnons qu'un algorithme de planification optimiste cherche à identifier une branche qui soit proche de l'optimale, dans un graphe qui peut être potentiellement infini (mais qui ne l'est pas dans le cas de l'ALP) tandis que A* cherche le plus court chemin dans un graphe fini, entre un nœud initial et un nœud final, tous les deux connus. Une autre différence entre ces deux algorithmes est qu'un algorithme de planification optimiste est contraint par des ressources de calcul limitées pour identifier cette branche, alors que l'algorithme A* ne l'est pas.

5.2.5 Heuristiques d'estimation

Comme nous avons vu dans la section 5.2.1, le calcul des coûts estimés pour chaque nœud nécessite le calcul de l'heuristique h, estimant le coût le plus bas lié au sous-ensemble d'avions qui n'ont pas encore atterri. Nous proposons dans cette section des candidats pour cette heuristique h. Nous étudions dans la section 5.3.1 l'effet de chacune de ces heuristiques sur la qualité des solutions obtenues.

Cette sous-section est basée sur notre article [59], intitulé *Coupling mathematical optimization and machine learning for the aircraft landing problem.*

5.2.5.1 Premier arrivé, premier servi

Le premier candidat pour l'heuristique h est le coût obtenu par la règle du FCFS. Rappelons que cette règle trie les avions à ordonnancer selon l'ordre croissant des heures préférentielles d'atterrissage (T_i), ce qui fixe la séquence. Ensuite, elle calcule pour chaque avion une

date cible d'atterrissage en le faisant atterrir au plus tôt après sa date préférentielle d'atterrissage, tout en respectant les contraintes de séparation (voir la section 4.4.1, équation (4.30)). Le coût de retard de chaque avion est alors calculé à l'aide des équations (4.18)-(4.22) et la somme de ces coûts représente le coût total de la séquence FCFS. La solution donnée par la règle du FCFS est toujours réalisable ; son coût est donc supérieur ou égal au coût optimal. Par conséquent, cette heuristique d'estimation n'est pas admissible au sens de l'algorithme A*.

5.2.5.2 Heuristique apprise

Nous proposons ici d'entraîner différents modèles d'apprentissage automatique (*Machine Learning* – ML) sur un grand nombre d'instances de l'ALP déjà résolues, afin de prédire le coût du retard d'une instance. Ces modèles ML pourront alors être utilisés comme heuristiques d'estimation h. Une idée similaire est utilisée dans [35] pour prédire la production optimale d'une centrale éolienne. Nous décrivons ci-dessous comment les ensembles d'entraînement et de test (*training and testing sets*) sont générés et nous détaillons les entrées et les sorties de ces modèles ML.

Génération des données

Les ensembles d'entraînement et de test sont générés à partir de nos quatre ensembles de données, `data_7_11.csv`, `data_11_15.csv`, `data_15_19.csv` et `data_19_21.csv`, présentés dans la section 4.3 et disponibles dans [55]. Le premier fichier, `data_7_11.csv`, est utilisé pour produire l'ensemble de test ; les autres fichiers servent à générer l'ensemble d'entraînement.

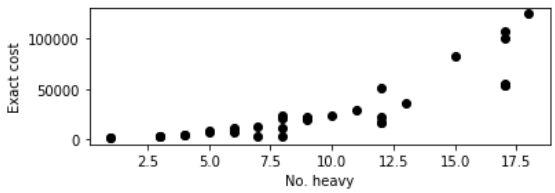
Définition des attributs

Le coût optimal d'une instance dépend de plusieurs facteurs tels que la congestion de l'instance et les catégories de turbulence de sillage des différents types d'avions impliqués. Pour utiliser les modèles ML, il est important de sélectionner des attributs ayant un impact significatif sur le coût du retard d'une instance, afin de mieux prévoir son coût optimal. La figure 5.3 illustre la relation entre certaines caractéristiques d'une instance et son coût optimal.

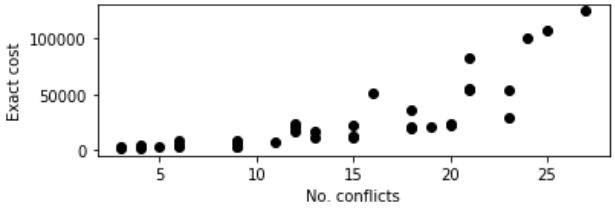
Tout d'abord, on observe que le coût optimal augmente avec le nombre d'avions de catégorie *Heavy* – H (figure 5.3a). Une explication possible est liée au fait que dévier des avions de catégorie H de leurs dates préférentielles est plus coûteux que dévier des avions de catégorie *Medium* – M ou *Light* – L.

En outre, le coût optimal dépend du nombre de conflits, c'est-à-dire le nombre de paires d'avions qui violeraient les contraintes de séparation si chacun se posait à sa date préférentielle d'atterrissage (figure 5.3b). En effet, plus le nombre de conflits augmente, plus le nombre d'avions qui vont être déviés de leurs dates préférentielles d'atterrissage sera élevé, ce qui entraînera l'augmentation du coût de l'instance. D'autre part, le coût optimal diminue lorsque le rapport entre la *longueur* de l'instance[1] et le nombre total d'avions augmente (figure 5.3c).

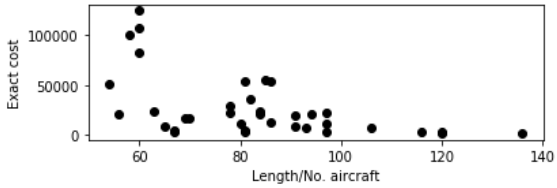
1. Différence (en secondes) entre les dates préférentielles la plus grande et la plus petite de l'instance.



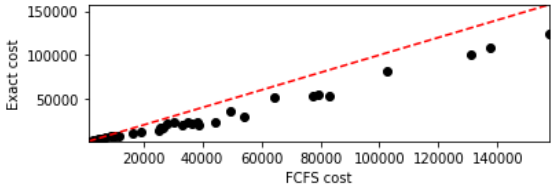
(a) Coût Optimal vs. nombre d'avions de catégories H



(b) Coût Optima vs. nombre de conflits dans l'instance



(c) Coût optimal vs. rapport [longueur de l'instance (secondes) / nombre total d'avions]



(d) Coût optimal vs. coût de la séquence FCFS

FIGURE 5.3 – Visualisation de la relation entre les attributs sélectionnés et le coût exact (optimal). La ligne rouge en pointillés représente la ligne d'identité.

Le coût optimal est également corrélé au coût de la séquence FCFS (figure 5.3d). En effet, si une instance est congestionnée et contient un grand nombre d'avions de catégorie H, tant la séquence FCFS que la séquence optimale auront des valeurs de coût élevées.

Sur la base de ces observations, nous sélectionnons les attributs suivants comme des données d'entrée des modèles ML :

- le nombre total d'avions de catégorie H ;
- le nombre total de conflits dans l'instance ;
- le rapport entre la longueur de l'instance (en secondes) et le nombre total d'avions : cela permet de connaître le degré de congestion de l'instance ;
- le coût de la séquence FCFS.

Modèles ML

Nous utilisons trois modèles pour estimer le coût de retard optimal d'une instance : la régression linéaire (*Linear Regression* – LR) [47] comme modèle de base et deux autres modèles d'apprentissage automatique, à savoir : les réseaux de neurones (*Neural Networks* – NN) [41] et la *Support Vector Regression* – SVR [47].

Pour chaque instance de l'ensemble d'entraînement, nous extrayons ses attributs et les affectons à un vecteur, noté X, qui est le vecteur d'entrée aux modèles ML. Quant au coût optimal de chaque instance i, noté Y_i, il est obtenu en utilisant notre modèle PLNE (4.23)-(4.28), résolu à l'aide de CPLEX. La sortie des modèles ML est l'heuristique d'estimation du coût optimal, notée h, qui dépend ici de certains paramètres inconnus, w (à déterminer), plus précisément $\hat{Y} = h(X, w)$ approche le coût optimal Y au sens des moindres carrés. Ces paramètres sont appris par les modèles ML pendant la phase d'entraînement, en minimisant la Mean Squared Error (MSE) (l'équation (5.3)) sur l'ensemble d'entraînement : $\{(X_1, Y_1), ..., (X_n, Y_n)\}$, où n est le nombre d'échantillons dans l'ensemble d'entraînement, X_i

est un vecteur d'attributs et Y_i est le coût optimal correspondant, $i = 1, 2, ..., n$:

$$MSE(w) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - h(x_i, w))^2 \tag{5.3}$$

5.3 Étude numérique

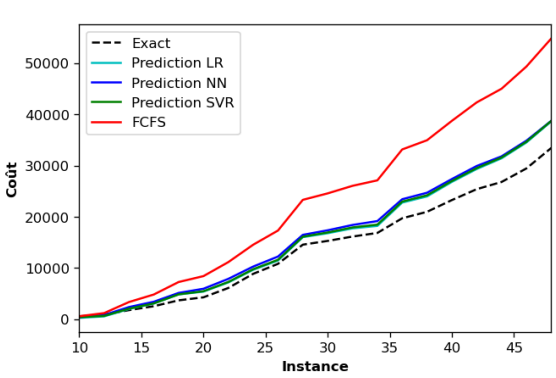
Nous présentons dans cette section les résultats d'implémentation de l'algorithme 2. Nous commençons d'abord par une étude comparative des différentes heuristiques d'estimation présentées dans la section 5.2.5, afin de déterminer laquelle est la plus appropriée pour prédire le coût optimal d'une instance. Ensuite, nous étudions la performance de l'algorithme 2, utilisant chacune de ces heuristiques d'estimation. Nous comparons enfin la performance de cet algorithme à celle des logiciels de programmation mathématique, tels que CPLEX.

5.3.1 Identification de la meilleure heuristique d'estimation

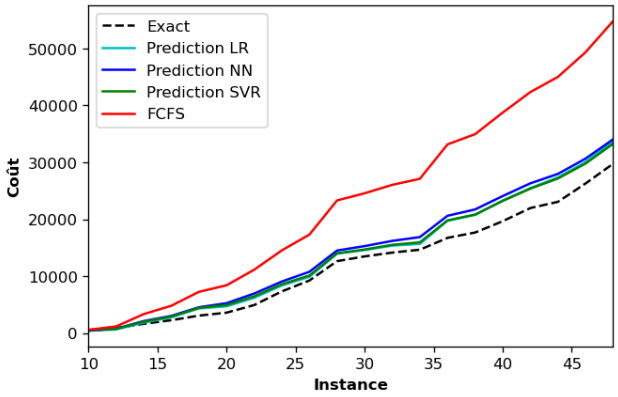
Afin de déterminer quelle est l'heuristique la plus appropriée pour estimer le coût optimal de l'ensemble d'avions qui n'ont pas encore atterri, nous effectuons une étude numérique sur différentes instances de l'ensemble test `data_7_11.csv`. Nous comparons dans cette étude la performance de l'algorithme 2 en utilisant chacune des heuristiques d'estimation présentées ci-dessus. Avant de présenter les résultats de la performance de l'algorithme 2 avec les trois heuristiques d'estimation, nous visualisons d'abord les résultats de prédiction des coûts optimaux avec le modèle de base LR, les deux modèles ML (NN et SVR) et nous les comparons avec les coûts obtenus avec la séquence FCFS. Les tests de cette sous-section sont effectués sur différentes instances de l'ensemble test `data_7_11.csv`, de tailles $|\mathcal{A}|$ allant de 5 à 40 avions et pour différentes valeurs du nombre maximal de changements de position, m. Rappelons que l'ensemble `data_7_11.csv` contient des données d'atterrissage de 185 avions (table 4.2). Ces données incluent, pour chaque avion, sa catégorie de turbulence de sillage, sa date préférentielle d'atterrissage et ses coûts (par seconde) de retard. Les instances sont obtenues en considérant les $|\mathcal{A}|$ premières lignes de données de cet ensemble.

La figure 5.4 montre les valeurs des meilleurs coûts connus (ligne pointillée noire), des coûts de la séquence FCFS (ligne rouge) et des coûts prédits par les modèles LR, NN et SVR pour les instances ci-dessus. Les meilleurs coûts connus sont obtenus en résolvant le modèle (4.23)-(4.28) à l'aide de CPLEX, dans une limite de temps fixée à 1000 secondes pour chaque instance. Ces graphiques révèlent que pour toutes les valeurs de m choisies, les estimations données par les trois modèles LR, NN et SVR sont plus précises que les coûts calculés par la règle du FCFS.

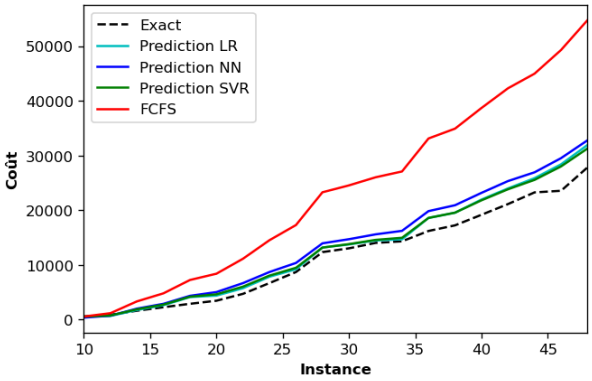
Nous présentons maintenant les résultats de l'implémentation de l'algorithme 2 avec les différentes heuristiques d'estimation présentées ci-dessus. Les temps de calcul sont critiques pour l'ALP, en raison de la nature dynamique du problème. Nous imposons alors une limite de temps de 5 secondes pour ces tests numériques. Les résultats sont affichés dans la table 5.1,



(a) 3-CPS



(b) 4-CPS



(c) 5-CPS

FIGURE 5.4 – Comparaison des coûts prédits par la régression linéaire et les modèles d'apprentissage sur les données test, avec la séquence FCFS est également représenté (en couleur rouge).

en terme de pourcentage d'amélioration. Rappelons que cet indicateur de performance reflète l'amélioration de la valeur de la fonction-objectif par rapport à la valeur de la solution de base FCFS. La définition de cet indicateur est donnée par l'équation (4.29).

Dans cette table, la première colonne présente la taille de l'instance, les autres colonnes indiquent le pourcentage d'amélioration pour différentes valeurs du nombre maximal de changements de position : $m = 3$ (3-CPS), $m = 4$ (CPS) et $m = 5$ (5-CPS). Pour chacune de ces valeurs, nous donnons le pourcentage d'amélioration de l'algorithme 2 en utilisant la séquence FCFS (OP-FCFS), la régression linéaire (OP-LR), les réseaux de neurones (OP-NN) et le support vector regression (OP-SVR) comme heuristiques d'estimation. Les meilleurs résultats sont indiqués en gras. Nous observons que pour toutes les instances, l'algorithme 2 (avec FCFS, LR, NN ou SVR) améliore significativement la séquence de base donnée par la règle FCFS, en des temps de calcul très courts (5 secondes). En effet, l'algorithme peut atteindre un pourcentage d'amélioration de $37,81\%$ pour $m = 3$, $45,43\%$ pour $m = 4$ et $49,75\%$ pour $m = 5$. En particulier, NN et FCFS atteignent tous les deux des résultats très compétitifs pour les trois valeurs de m étudiées, avec des améliorations légèrement meilleures en moyenne pour la FCFS. Motivés par ces résultats, nous adoptons la FCFS comme heuristique d'estimation dans la suite de cette section.

$\|\mathcal{A}\|$	%improv (3-CPS)				%improv (4-CPS)				%improv (5-CPS)			
	OP-FCFS	OP-LR	OP-NN	OP-SVR	OP-FCFS	OP-LR	OP-NN	OP-SVR	OP-FCFS	OP-LR	OP-NN	OP-SVR
20	**36.40**	32.88	31.38	35.99	**45.43**	44.13	40.61	40.68	**48.06**	46.02	46.09	41.84
22	**34.25**	32.71	31.88	32.47	**45.17**	37.22	38.46	39.79	42.82	42.86	**42.86**	38.88
24	**36.74**	33.73	36.11	36.26	36.95	39.02	**39.46**	39.08	**42.86**	28.97	40.69	39.63
26	**35.30**	32.58	34.12	33.07	36.00	35.70	**38.42**	37.34	42.14	39.15	42.46	**42.50**
28	34.79	26.23	**35.14**	32.67	39.27	37.28	**39.64**	39.64	41.69	39.14	41.23	**41.69**
30	34.60	25.82	**34.77**	34.32	39.04	37.82	**39.49**	36.82	39.69	38.50	**40.75**	40.70
32	34.05	31.82	32.16	**34.20**	38.84	35.19	37.26	**38.98**	**43.77**	39.44	42.02	39.75
34	34.47	34.72	**34.78**	29.90	**38.05**	37.06	37.07	35.85	43.31	39.12	34.76	**43.47**
36	35.90	35.00	35.19	**39.94**	34.19	33.74	34.12	**35.13**	**43.09**	36.64	39.99	36.81
38	35.25	31.08	**35.25**	30.08	26.83	34.66	35.78	**35.89**	**41.29**	36.66	37.59	36.58
42	33.67	30.36	**35.33**	33.23	**42.44**	35.22	33.76	34.43	44.09	39.75	**49.75**	39.49
44	36.59	27.07	**37.81**	32.39	**35.98**	32.96	33.78	33.78	47.78	34.33	**47.80**	35.65
46	25.51	**27.91**	27.87	32.24	29.73	32.33	**32.86**	32.31	39.06	38.37	**39.63**	38.38
48	**32.93**	27.63	21.44	27.63	31.65	**31.70**	31.32	31.60	47.48	32.68	**47.55**	47.47
50	**33.43**	23.69	23.69	23.69	30.72	29.25	**30.81**	29.86	32.29	36.27	**36.28**	36.23
min	**25.51**	23.69	21.44	23.69	26.83	29.25	**30.81**	29.86	32.29	28.97	34.76	**35.65**
max	36.74	35.00	**37.81**	36.26	**45.43**	44.13	40.61	40.68	48.06	46.02	**49.75**	43.47
moyenne	**34.26**	30.22	32.46	32.54	**36.69**	35.55	36.19	36.08	**42.63**	37.86	41.96	39.94

TABLE 5.1 – Comparaison des différentes heuristiques d'estimation pour l'algorithme de planification optimiste.

5.3.2 Comparaison avec des logiciels de programmation mathématique

Nous comparons dans cette sous-section la performance de l'algorithme 2 utilisant la règle du FCFS comme heuristique d'estimation avec la performance des logiciels de programmation mathématique pour résoudre le modèle (4.23)-(4.28). Les deux logiciels de programmation mathématique que nous choisissons sont :

— DoCplex, qui est l'interface Python du logiciel de programmation mathématique CPLEX. La résolution est ensuite faite par CPLEX, version 12.8.

— PySCIPOpt, qui est l'interface Python du logiciel de programmation mathématique SCIP [38]. La résolution est ensuite faite avec SCIP, version 7.0.1.

Les tests sont effectués cette fois-ci sur les 12 instances de référence introduites dans la section 4.3, pour un nombre maximal de changements de position égal à 5. La limite de temps imposée est de 5 secondes.

Les résultats sont présentés dans la table 5.2. Nous constatons qu'en moyenne, l'algorithme 2 atteint des pourcentages d'amélioration proches de ceux donnés par CPLEX. En revanche, comparé à SCIP, notre algorithme donne les meilleurs résultats sur presque toutes les instances étudiées.

Conclusions du chapitre

Nous avons présenté dans ce chapitre une nouvelle formulation basée sur les processus de décision markoviens pour modéliser le problème d'ordonnancement d'avions en atterrissage (ALP), en prenant en considération les contraintes de séparation et de changements de position. Nous avons ensuite présenté une nouvelle méthode heuristique pour résoudre l'ALP, basée sur l'algorithme de planification optimiste [49]. Dans l'étude expérimentale de notre

Instance	$\|\mathcal{A}\|$	% improv		
		CPLEX	OP-FCFS	SCIP
alp_7_30.csv	30	**41.05**	36.73	21.13
alp_7_40.csv	40	**41.34**	35.25	22.92
alp_7_50.csv	50	**35.38**	34.91	22.32
alp_11_30.csv	30	**43.87**	41.00	34.88
alp_11_40.csv	40	**42.55**	37.44	30.68
alp_11_50.csv	50	**32.82**	26.73	31.26
alp_15_30.csv	30	**28.13**	27.58	20.64
alp_15_40.csv	40	20.92	**27.58**	19.85
alp_15_50.csv	50	24.26	**35.80**	15.17
alp_19_30.csv	30	**42.87**	33.95	32.70
alp_19_50.csv	40	46.56	**48.49**	31.94
alp_19_50.csv	50	**38.07**	32.47	26.75
min		20.92	**26.73**	15.17
max		46.56	**48.49**	32.88
moyenne		**36.39**	34.82	25.85

TABLE 5.2 – Comparaison avec deux logiciels de programmation mathématique : CPLEX et SCIP.

méthode, nous avons montré qu'elle est capable de trouver des solutions de bonne qualité, en des temps de calcul très courts (5 secondes). Pour valider notre méthode, nous avons comparé sa performance à celle de deux logiciels de programmation mathématique : CPLEX et SCIP. Dans cette dernière comparaison, nous avons constaté que notre méthode aboutit à des résultats très compétitifs avec CPLEX et largement meilleurs que SCIP.

Dans le chapitre suivant, nous considérons le problème d'ordonnancement d'atterrissages d'avions dans un contexte plus réaliste : il s'agit de l'ordonnancement d'arrivées d'avions au niveau de points critiques de la zone TMA ainsi qu'aux seuils de pistes.

Chapitre 6

Problème d'ordonnancement d'avions aux balises de la zone terminale et au seuil de piste

Dans les aéroports les plus fréquentés et pendant les périodes de congestion, des goulots d'étranglement peuvent se produire ailleurs dans la zone d'approche aéroportuaire (TMA), surtout au niveau de certaines balises (*points repères*) de cette zone, ce qui engendre des retards importants à l'atterrissage. Nous nous intéressons dans ce chapitre au problème d'ordonnancement d'arrivées d'avions au niveau des balises de la zone TMA ainsi qu'au seuil de piste. Nous prenons en compte les contraintes de séparation pour garantir la sécurité des vols et les contraintes de fenêtres de temps. L'objectif est de minimiser le retard total. Nous proposons dans cette étude préliminaire une approche PLNE pour modéliser et résoudre ce problème. Dans l'étude numérique, nous comparons l'approche PLNE à une technique traditionnelle utilisée par les contrôleurs aériens.

Ce chapitre reprend notre communication intitulée *A mixed integer programming approach for scheduling aircraft arrivals at terminal airspace fixes and runway threshold* [57], publiée dans les actes de la conférence *Project Management and Scheduling, 2021*

Sommaire

6.1 Définition du problème

La gestion de trafic aérien dans la zone TMA est l'un des défis majeurs des systèmes ATM, de par la densité de trafic importante qu'elle comporte. Une des responsabilités des contrôleurs de cette zone est d'affecter les pistes d'atterrissage ou de décollage aux avions, de

façon à optimiser l'utilisation de ces pistes. La technique de base utilisée par les contrôleurs d'approche pour l'affectation des pistes consiste à attribuer à chaque avion la piste la plus proche de sa balise d'arrivée. Quand le volume de trafic venant des différentes directions est déséquilibré et que tous les avions empruntent le chemin le plus court de leur balise d'arrivée à la piste, certaines pistes risquent d'être surchargées, tandis que d'autres demeureront inutilisées. Par conséquent, cette stratégie peut entraîner de la congestion au niveau des balises ainsi que des retards importants à l'atterrissage.

Nous proposons dans cette étude préliminaire un modèle mathématique pour l'affectation de piste et de balise de la zone TMA, dans le but de rééquilibrer le trafic et réduire ainsi les retards.

6.1.1 Contexte et motivation

Les avions arrivant de différentes directions entrent dans la zone TMA via des balises de l'espace aérien aéroportuaire, appelés *Initial Approach Fix* – IAF. Au niveau de ces balises, certains avions peuvent être mis en attente par les contrôleurs avant l'atterrissage, comme nous avons vu dans la section 1.3.2. Les flux d'aéronefs arrivant des différents IAF doivent être fusionnés et les avions séquencés, en suivant des routes prédéfinies appelées *Standard Terminal Arrival Routes* – STAR. Une approche de type STAR est souvent constituée de trois segments définis par des points spécifiques, comme illustré dans la figure 6.1 :

— L'*approche initiale* débute au point IAF et se termine à l'IF (*Intermediate Fix*).

— L'*approche intermédiaire* est définie de l'IF au FAF (*Final Approach Fix*).

— L'*approche finale* débute à partir du FAF et se termine au seuil de piste, si l'avion complète son atterrissage.

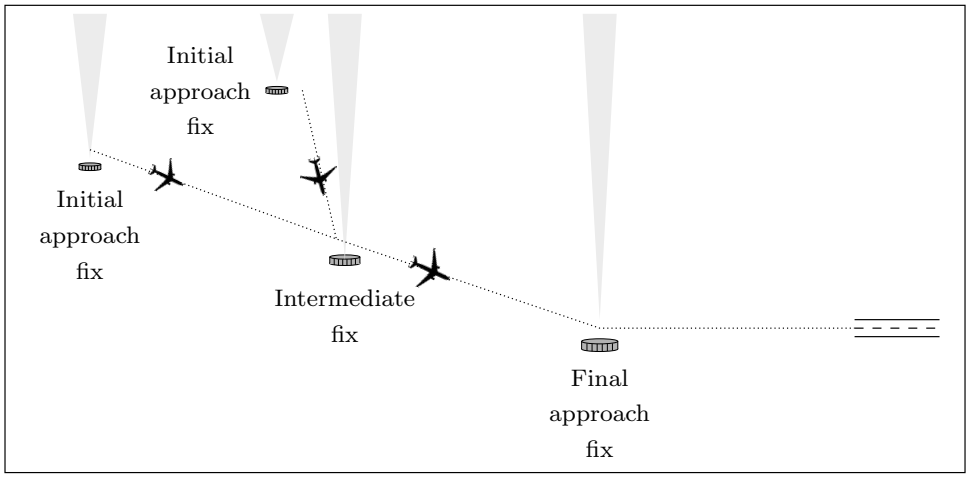


FIGURE 6.1 – Illustration des différentes phases d'approche.

En pratique, les contrôleurs aériens affectent à chaque avion un point IAF de passage, selon son aéroport de départ. Les contrôleurs font généralement passer les avions par l'IAF le plus proche de leur aéroport de départ. Ensuite, les avions atterrissent sur la piste la plus

proche de leur IAF de passage, selon la règle FCFS. Cette stratégie d'affectation d'IAF et de piste est simple à appliquer, mais elle risque d'engendrer des retards importants quand le volume de trafic arrivant des différentes directions est fortement déséquilibré. Ceci est parfois le cas dans certains aéroports, comme par exemple à l'aéroport de Détroit, comme l'illustre la figure 6.2. Nous observons dans cette figure que le flux de trafic arrivant de l'ouest est très important, ce qui risque de surcharger les pistes situées dans la partie ouest. En revanche, le volume de trafic en provenance de l'est est beaucoup plus faible. Pour remédier à ce problème, les contrôleurs les plus expérimentés détournent alors une partie du trafic d'un IAF vers un autre, ou d'une piste surchargée vers une autre [65].

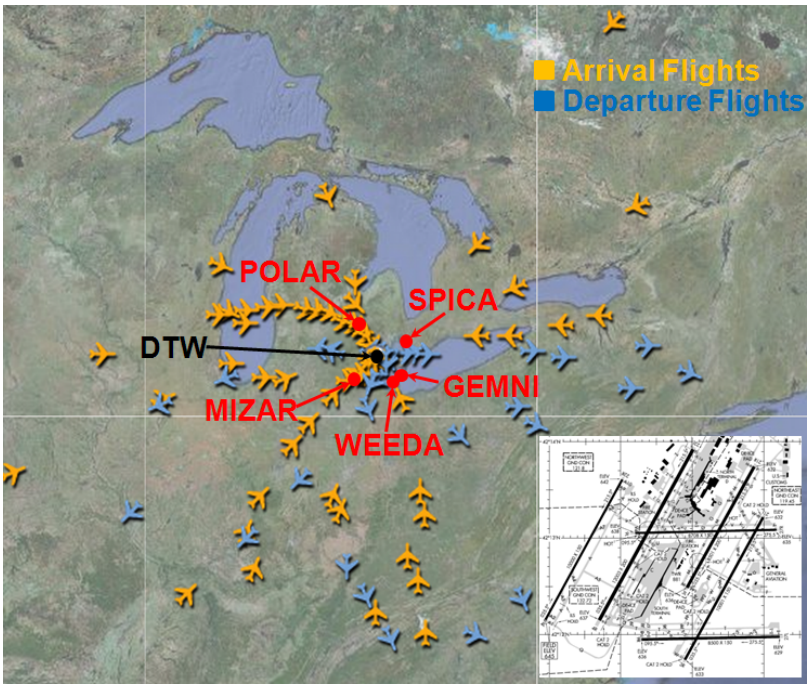


FIGURE 6.2 – Volume de trafic déséquilibré entre les cinq IAF (nommés : MIZAR, POLAR, SPICA, GEMNI et WEEDA) de l'aéroport de Détroit.
Source : Kim et al. [65].

Motivée par cette stratégie des contrôleurs expérimentés, nous proposons dans cette étude un modèle d'optimisation pour le problème d'ordonnancement d'arrivées d'avions au niveau des IAF et au seuil de piste. Ce problème requiert deux types de décision : une première décision au niveau des IAF, qui concerne l'attribution d'un IAF de passage pour chaque avion, ainsi qu'une date cible de passage à ce point. La deuxième décision concerne les pistes d'atterrissage, c'est-à-dire l'affectation d'une piste disponible pour chaque avion, ainsi qu'une heure cible d'atterrissage sur cette piste.

6.1.2 Contraintes opérationnelles

Afin de garantir la sécurité des vols et des mouvements sur les pistes, des normes de séparation entre paires d'aéronefs sont imposées. Nous considérons dans notre modèle les deux normes de séparation suivantes :

— Une séparation longitudinale minimale de 72 secondes à respecter au niveau des IAF. Nous supposons que tous les avions passent par l'IAF à la même altitude. Par conséquent, seule la séparation longitudinale de 5 NM entre les avions compte. Nous convertissons ensuite cette valeur basée sur la distance (5 NM) en une séparation basée sur le temps, en supposant que tous les avions ont une même vitesse à l'IAF : 250 kt [63].

— Une séparation relative aux turbulences de sillage, dont les valeurs varient entre $2,5$ NM et 6 NM (voir la table 1.1). À partir du point de début de l'approche finale (FAF) et jusqu'à la piste, les séparations minimales imposées dépendent donc de la catégorie de l'avion meneur et de celle de l'avion suiveur. Les valeurs de cette séparation sont converties en temps, comme expliqué dans l'annexe A de [60]. Après la conversion, nous obtenons les valeurs affichées dans la table 6.1.

		Avion suiveur		
		H	M	L
Avion générateur	H	96	157	196
	M	60	69	131
	L	60	69	82

TABLE 6.1 – Matrice de séparation (en secondes) au seuil de piste selon les trois catégories de turbulences de sillage : *Heavy* (H), *Medium* (M) et *Light* (L).
Source : Balakrishnan et Chandran [6].

Pour éviter que les avions subissent des retards importants, notre modèle impose également des contraintes de fenêtres de temps. C'est un intervalle temporel, $[T, L]$, défini par une heure préférentielle d'atterrissage, T, et une heure d'atterrissage au plus tard, L. Cette dernière est définie compte tenue de la disponibilité de carburant à bord. Dans la littérature, la valeur de L varie souvent entre 1800 et 3600 secondes après l'heure préférentielle d'atterrissage.

6.2 Formulation mathématique

Nous proposons dans cette section une formulation PLNE pour le problème d'ordonnancement d'arrivées d'avions aux balises de la zone TMA et au seuil de piste.

Données

On définit les éléments suivants :

$\mathcal{A}$: ensemble des indices d'avions.
$\mathcal{I}$: ensemble des indices d'IAFs.
$\mathcal{R}$: ensemble des indices des pistes d'atterrissage.
S^I : séparation minimale temporelle à l'IAF (≥ 0).
S_{ij} : séparation minimale temporelle à la piste (≥ 0) entre deux avions $i, j \in \mathcal{A}$, où l'avion i atterrit avant l'avion j.
τ_{kr} : temps de transition moyen entre l'IAF $k \in \mathcal{I}$ et la piste $r \in \mathcal{R}$.

Pour chaque avion $i \in \mathcal{A}$:

T_i : date préférentielle d'atterrissage.
L_i : date d'atterrissage au plus tard.

Variables de décision :

y_i : date cible de passage à l'IAF de l'avion i.

x_i : date cible d'atterrissage de l'avion i.

$$a_{ik} = \begin{cases} 1 & \text{si l'avion } i \text{ passe par l'IAF } k,\ i \in \mathcal{A}, k \in \mathcal{I}, \\ 0 & \text{sinon.} \end{cases}$$

$$b_{ir} = \begin{cases} 1 & \text{si l'avion } i \text{ est affecté à la piste } r,\ i \in \mathcal{A}, r \in \mathcal{R}, \\ 0 & \text{sinon.} \end{cases}$$

$$\delta_{ijk} = \begin{cases} 1 \text{ si les avions } i \text{ et } j \text{ passent par le même IAF } k \ (i \textbf{ avant } j),\ i, j \in \mathcal{A} : i \neq j, \\ \quad k \in \mathcal{I}, \\ 0 \quad \text{sinon.} \end{cases}$$

$$\Delta_{ijr} = \begin{cases} 1 \text{ si les avions } i \text{ et } j \text{ sont affectés à la même piste } r \ (i \ \textbf{ avant } j),\ i, j \in \mathcal{A}, \\ \quad i \neq j, r \in \mathcal{R}, \\ 0 \quad \text{sinon.} \end{cases}$$

Fonction-objectif :

Nous souhaitons minimiser le retard total par rapport aux dates préférentielles d'atterrissage $T_i, i \in \mathcal{A}$. La fonction-objectif est alors :

$$\sum_{i \in \mathcal{A}} (x_i - T_i). \tag{6.1}$$

Dans ce qui suit, les variables x et y représentent les vecteurs dont les $i^{\text{ième}}$ composantes sont respectivement x_i et y_i. La variable a (respectivement b) représente la matrice dont la composante $(i, k), i \in \mathcal{A}, k \in \mathcal{I}$ (respectivement $(i, r), i \in \mathcal{A}, r \in \mathcal{R}$) est a_{ik} (b_{ir} respectivement). Finalement, les variables δ et Δ représentent les matrices dont les composantes $(i, j), i, j \in \mathcal{A}$, sont respectivement δ_{ij} et Δ_{ij}.

On obtient donc la formulation suivante :

Modèle complet

$$\min_{x,y,a,b,\delta,\Delta} \quad \sum_{i\in\mathcal{A}}(x_i - T_i) \tag{6.2}$$

s.c

$$T_i \le x_i \le L_i \qquad i \in \mathcal{A} \tag{6.3}$$

$$x_i \ge y_i + \sum_{r\in\mathcal{R}} b_{ir}\tau_{kr} - M(1-a_{ik}) \qquad i \in \mathcal{A}, k \in \mathcal{I} \tag{6.4}$$

$$y_j \ge y_i + S^I - M^I(1-\delta_{ijk}) \qquad i,j \in \mathcal{A} : i \ne j, k \in \mathcal{I} \tag{6.5}$$

$$x_j \ge x_i + S_{ij} - M^R(1-\Delta_{ijr}) \qquad i,j \in \mathcal{A} : i \ne j, r \in \mathcal{R} \tag{6.6}$$

$$\sum_{k\in\mathcal{I}} a_{ik} = 1 \qquad i \in \mathcal{A} \tag{6.7}$$

$$\sum_{r\in\mathcal{R}} b_{ir} = 1 \qquad i \in \mathcal{A} \tag{6.8}$$

$$\sum_{k\in\mathcal{I}} (\delta_{ijk} + \delta_{jik}) \le 1 \qquad i,j \in \mathcal{A} : i < j \tag{6.9}$$

$$\sum_{r\in\mathcal{R}} (\Delta_{ijr} + \Delta_{jir}) \le 1 \qquad i,j \in \mathcal{A} : i < j \tag{6.10}$$

$$\delta_{ijk} + \delta_{jik} \ge a_{ik} + a_{jk} - 1 \qquad i,j \in \mathcal{A} : i < j, k \in \mathcal{I} \tag{6.11}$$

$$2(\delta_{ijk} + \delta_{jik}) \le a_{ik} + a_{jk} \qquad i,j \in \mathcal{A} : i < j, k \in \mathcal{I} \tag{6.12}$$

$$\Delta_{ijr} + \Delta_{jir} \ge b_{ir} + b_{jr} - 1 \qquad i,j \in \mathcal{A} : i < j, r \in \mathcal{R} \tag{6.13}$$

$$2(\Delta_{ijr} + \Delta_{jir}) \le b_{ir} + b_{jr} \qquad i,j \in \mathcal{A} : i < j, r \in \mathcal{R} \tag{6.14}$$

$$y_i, x_i \ge 0 \qquad i \in \mathcal{A} \tag{6.15}$$

$$a_{ik}, b_{ir}, \delta_{ijk}, \Delta_{ijr} \in \{0,1\} \qquad i,j \in \mathcal{A} : i \ne j, k \in \mathcal{I}, r \in \mathcal{R} \tag{6.16}$$

Dans la formulation ci-dessus, les contraintes (6.3) représentent les restrictions de fenêtres de temps. Les contraintes (6.4) lient les dates d'atterrissage, x_i, aux dates de passage à l'IAF, y_i. Les contraintes (6.5) et (6.6) garantissent la séparation entre paires d'aéronefs à l'IAF et à la piste (respectivement). Les contraintes (6.7) et (6.8) assurent que chaque avion passe par un seul IAF et atterrit sur une seule piste (respectivement). Les contraintes (6.9) et (6.10) représentent les contraintes de séquencement pour les paires d'avions passant par le même IAF et atterrissant sur la même piste (respectivement). Les contraintes (6.11) et (6.12) (respectivement (6.13) et (6.14)) préservent la relation logique entre les variables δ_{ijk} et a_{ik} (Δ_{ijr} et b_{ir} respectivement). Les contraintes (6.15) et (6.16) garantissent que les variables de décision prennent leur valeur dans un ensemble adéquat (variables continues positives et variables binaires).

Indice	Catégorie	IAF	T (s)	T (HH:MM:SS)	Piste
1	Heavy	ODILO	22 800	06 :20 :00	26
2	Medium	MOLBA	22 800	06 :20 :00	26
3	Heavy	MOLBA	23 100	06 :25 :00	26
4	Medium	ODILO	23 700	06 :35 :00	06

TABLE 6.2 – Exemple d'instance avec quatre avions.

6.3 Résultats préliminaires

Dans cette section, nous présentons les résultats d'implémentation du modèle (6.2)-(6.16). Cette formulation PLNE a été implémentée avec DoCplex, l'interface Python du logiciel CPLEX. La résolution est ensuite faite par CPLEX, version 12.8.

Nous décrivons dans la section 6.3.1 les instances utilisées pour les tests numériques. Ensuite, nous présentons dans la section 6.3.2 les résultats d'implémentation du modèle PLNE (6.2)-(6.16), que nous comparons avec la solution traditionnelle utilisée par les contrôleurs sur la base de plusieurs indicateurs de performance.

6.3.1 Contexte expérimental et instances du problème

Nous utilisons dans cette étude préliminaire l'aéroport de Paris-Orly comme cas d'étude. Cet aéroport comporte deux pistes (06/24 et 08/26) comme le montre la figure 6.3b, considérées comme indépendantes[1]. La piste 02/20 est rarement utilisée pour le trafic commercial. L'aéroport a trois IAFs nommés : MOLBA, ODILO et VEBEK, comme illustré dans la figure 6.3a. Les temps de transition moyens entre ces IAF et les deux pistes utilisées pour le trafic commercial sont présentés dans la table 6.3.

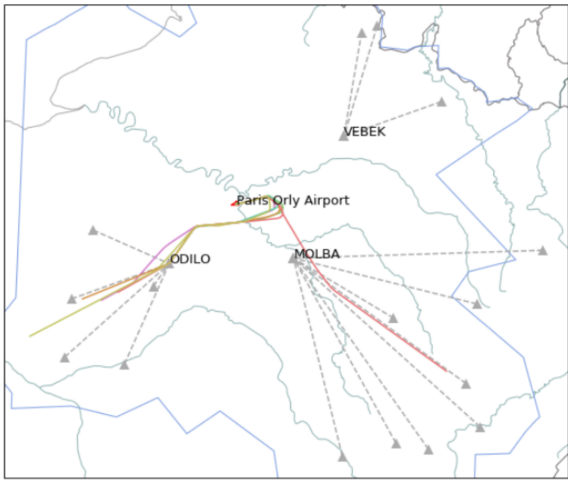
Les données brutes proviennent d'une journée de trafic aérien en juillet 2018 sur cet aéroport, avec 322 avions. Cet ensemble de données sert de base pour générer les instances de ce chapitre. Chaque instance est obtenue en considérant un intervalle temporel d'une heure, entre `12:00` et `22:00`. Nous présentons dans la table 6.2 les données correspondant à nos instances. Pour chaque avion, nous avons les informations suivantes : sa catégorie de turbulence de sillage (catégorie), l'IAF par lequel il est passé, sa date préférentielle d'atterrissage (T) en secondes et au format `HH:MM:SS` et sa piste d'atterrissage. L'intégralité des données (pour les 322 avions) se trouve en annexe A.

6.3.2 Résultats

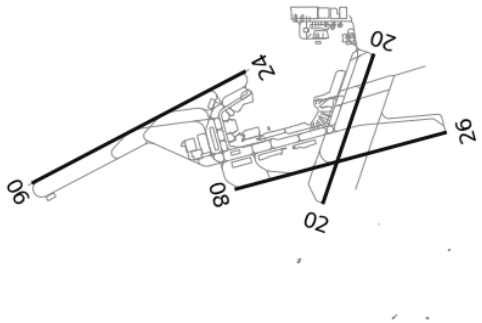
Nous comparons dans cette étude les trois solutions suivantes :

— **FCFS** (*First-Come, First-Served*) : la technique traditionnelle utilisée par les contrô-

1. Pistes indépendantes : les opérations (atterrissage et/ou décollage) sur une piste ne sont pas affectées par les opérations sur les autres pistes.



(a) Les trois IAF de l'aéroport Paris-Orly.



(b) Les pistes de l'aéroport Paris-Orly.

FIGURE 6.3 – Représentation de l'aéroport Paris-Orly : IAF et pistes.

		Pistes	
		06	26
IAF	MOLBA	1 036	854
	ODILO	816	1 200
	VEVEK	1 085	1 085

TABLE 6.3 – Temps de transition moyen entre les IAF et les pistes de l'aéroport de Paris-Orly (en secondes).

leurs aériens pour l'affectation des IAF et des pistes. Elle consiste à affecter à chaque avion l'IAF le plus proche en fonction de son aéroport d'origine, puis la piste la plus proche selon l'IAF affecté. Les séquences d'avions à l'IAF et au seuil de piste sont définies selon l'ordre « premier arrivé, premier servi » ;

— **RIAS-MILP** (*Runway and IAF Assignment and Sequencing*) : la solution donnée par notre modèle (6.2)-(6.16) ;

— **RAS-MILP** (*Runway Assignment and Sequencing*) : la solution donnée par notre modèle (6.2)-(6.16), mais en fixant l'affectation de l'IAF selon l'aéroport d'origine, de la même façon que la solution FCFS. Cette solution demande moins de charge de travail de la part du contrôleur aérien que celle obtenue avec RIAS-MILP, car elle ne requiert plus de détourner le trafic d'un IAF vers un autre.

Les trois solutions ci-dessus sont comparées en termes des indicateurs de performances suivants :

— cpu : le temps de calcul en secondes, requis pour que le logiciel d'optimisation (CPLEX) trouve une solution optimale ;

— Avg Del : le retard moyen de chaque solution calculé comme suit :

$$\text{Avg Del} = \frac{\sum_{i \in \mathcal{A}} (x_i - T_i)}{|\mathcal{A}|}. \tag{6.17}$$

Intervalle	$\|\mathcal{A}\|$	FCFS			RAS-MILP			RIAS-MILP		
		cpu	Avg Del	Max Del	cpu	Avg Del	Max Del	cpu	Avg Del	Max Del
12 :00-13 :00	28	<1	53	145	5	23	86	99	17	69
13 :00-14 :00	20	<1	58	166	1	15	75	5	7	69
14 :00-15 :00	15	<1	61	235	1	20	147	2	14	69
15 :00-16 :00	17	<1	59	246	5	18	77	1	8	69
16 :00-17 :00	22	<1	101	378	1	52	288	365	25	138
17 :00-18 :00	21	<1	65	248	1	18	89	3	1	17
18 :00-19 :00	19	<1	59	251	1	16	100	2	10	69
19 :00-20 :00	22	<1	48	217	1	18	148	7	7	69
20 :00-21 :00	18	<1	63	216	1	25	184	2	5	69
21 :00-22 :00	26	<1	66	242	3	24	151	106	11	69
Min	15	<1	48	145	1	15	75	1	1	17
Max	28	<1	589	378	5	52	288	365	25	138
Moyenne	21	<1	57	234	2	23	130	59	11	71

TABLE 6.4 – Comparaison des solutions calculées par nos modèles (RAS-MILP et RIAS-MILP) avec la solution donnée par la règle FCFS.

Cet indicateur de performance correspond à la valeur de notre fonction-objectif, à une constante multiplicative près.

— Max Del : le retard maximal de chaque solution calculé comme suit :

$$\text{Max Del} = \max\{(x_i - T_i), i = 1, 2, ..., |\mathcal{A}|\}. \tag{6.18}$$

La table 6.4 représente les résultats de la résolution du modèle PLNE (6.2)-(6.16) pour les 10 instances décrites dans la section 6.3.1. Pour tous les tests, nous considérons $|\mathcal{R}| = 2$ pistes, ce qui correspond au nombre de pistes utilisées pour le trafic commercial à l'aéroport de Paris-Orly. Nous considérons également les $|\mathcal{I}| = 3$ IAF de cet aéroport.

Dans la table 6.4, les première et deuxième colonnes présentent respectivement l'intervalle temporel de l'instance et sa taille, $|\mathcal{A}|$. Les autres colonnes donnent les résultats de la résolution de chaque instance en termes des indicateurs de performance introduits ci-dessus, à savoir le temps de calcul (cpu), le retard moyen (Avg Del) et le retard maximal (Max Del) de chaque solution. Nous constatons d'abord que les temps de calculs requis par FCFS sont très courts (< 1 seconde) pour toutes les instances. Nous constatons également que le modèle RAS-MILP exige des temps de calcul très courts par rapport au modèle RIAS-MILP. Ceci peut être expliqué par le fait que pour le modèle RAS-MILP, les variables binaires relatives à l'affectation d'IAF (δ) sont fixées, ce qui réduit la complexité du problème. En revanche, le modèle RIAS-MILP donne les meilleurs résultats en termes de retard moyen et de retard maximal (plus de degrés de liberté).

La table 6.4 quantifie le bénéfice d'une approche d'optimisation pour des instances réalistes du problème d'ordonnancement des arrivées d'avions, même dans le cas RAS-MIP – dans lequel l'affectation des IAF est fixée en fonction de l'aéroport d'origine – par rapport à la technique traditionnelle utilisée par les contrôleurs (FCFS) et donne l'ordre de grandeur des temps de calcul requis. Notre modèle PLNE complet (RIAS-MILP) donne, comme prévu, des retards moyens qui sont encore plus réduits. Le gain en termes des retards moyens reste modéré pour ces instances, mais nous comptons travailler sur des aéroports plus congestionnés pour lesquels une approche d'optimisation sera plus bénéfique.

6.4 Conclusion et perspectives

Nous nous sommes intéressée dans ce chapitre au problème d'ordonnancement des arrivées d'avions dans un contexte plus réaliste. Il s'agit d'un ordonnancement au niveau des IAFs de la zone d'approche aéroportuaire ainsi qu'au seuil de piste. Nous avons présenté une nouvelle approche de programmation mixte en nombres entiers (PLNE) pour modéliser et résoudre ce problème. Nous avons quantifié le bénéfice de cette approche avec la solution traditionnelle utilisée par les contrôleurs aériens pour l'affectation d'IAF et de piste. Les résultats préliminaires montrent qu'une approche PLNE est bénéfique, même dans le cas où l'affectation des IAF est fixée en fonction de l'aéroport d'origine. Nos pistes futures de recherche incluent :

— la prise en compte d'une fonction-objectif convexe et linéaire par morceaux, comme celle étudiée dans la section 4.2.2, car une telle fonction permet de bien représenter les coûts réels encourus par un retard ;

— la validation du modèle PLNE sur des données venant de d'autres aéroports plus congetionnés, pour lesquels une approche d'optimisation sera plus bénéfique ;

— l'intégration de la PLNE dans une approche d'horizon glissant, afin de résoudre des instances de plus grandes tailles et plus difficiles à résoudre. En effet, si nous considérons le cas particulier de notre problème sans l'étape de l'IAF, le problème se réduit à un problème de tournées de véhicules (VRP) sans contraintes de capacité, qui est connu pour être NP-difficile [18]. Par conséquent, l'approche PLNE risque de se heurter à des temps de calcul prohibitifs pour les instances de grandes tailles.

Nous prévoyons également de discuter de notre modèle avec des contrôleurs aériens, afin de l'améliorer et de le rendre plus réaliste, dans le but d'évaluer la viabilité de cette approche PLNE dans une application en temps réel.

Conclusion et perspectives

Optimiser l'utilisation des infrastructures aéroportuaires, en particulier la piste qui est considérée comme l'un des principaux goulots d'étranglement du système aéroportuaire, est un enjeu majeur du secteur du transport aérien. Dans cette optique, on étudie le problème d'ordonnancement des atterrissages d'avions. Ce problème consiste à affecter d'abord une piste disponible à chaque avion demandant l'atterrissage, puis à allouer à chacun une date cible d'atterrissage tout en respectant plusieurs contraintes opérationnelles. L'objectif diffère selon le point de vue de la partie prenante considérée : aéroport, contrôleur aérien ou compagnie aérienne. Dans la littérature, ce sont les points de vue des aéroports et des compagnies aériennes qui sont généralement pris en compte. Le premier vise à maximiser la capacité des aéroports ; le deuxième cherche à minimiser les coûts.

Dans le cadre de cette thèse, nous avons présenté une revue des modèles et méthodes de résolution les plus pertinents pour le problème d'ordonnancement d'avions (ALP – *Aircraft Landing Problem*, ATP – *Aircraft Take-off Problem* et ASP – *Aircraft Scheduling Problem*). Les méthodes et modèles étudiés incluent les méthodes exactes et les approches stochastiques, y compris les méthodes hybrides combinant programmation mathématique et métaheuristiques. Nous avons également étudié les nouvelles approches basées sur l'apprentissage par renforcement.

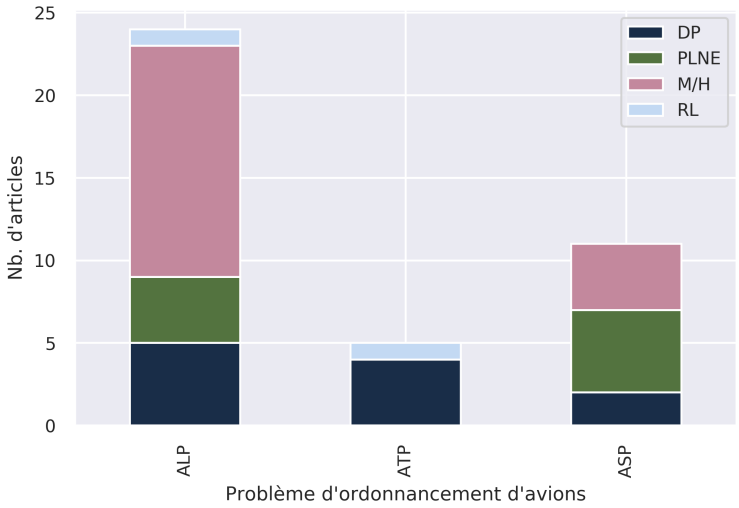


FIGURE 6.4 – Décompte des articles abordant les problèmes d'ordonnancement d'avions, regroupés en quatre types de méthodologies : programmation dynamique (DP), programmation linéaire mixte en nombres entiers (PLNE), métaheuristiques/heuristiques (M/H) et apprentissage par renforcement (RL).

Nous décomptons dans la figure 6.4 les articles de la littérature récente, en fonction du problème qu'ils considèrent (ALP, ATP ou ASP). Ils sont regroupés en quatre types de méthodologies : programmation dynamique, programmation mixte en nombres entiers, métaheuristiques/heuristiques et apprentissage par renforcement. Nous observons à partir de cet histogramme que la plupart des travaux ont recours à des approches stochastiques.

Ceci peut être expliqué par la nature dynamique du problème, qui incite à privilégier des méthodes permettant l'adaptation rapide de solutions courantes lorsqu'un nouvel évènement se produit. Nous observons également que les contributions se concentrent principalement sur le problème d'ordonnancement d'atterrissages d'avions.

Par le biais d'une étude comparative, nous avons montré que la plupart des instances de la littérature sont résolues dans des temps de calcul très courts. Par conséquent, elles ont perdu de leur intérêt pour les versions actuelles des logiciels d'optimisation. Dans cette thèse, nous avons présenté des bases de données et des instances issues de données de trafic réel pour l'ALP qui sont difficiles à résoudre. Dans nos données, nous proposons des calculs de coûts de retard réalistes qui dépendent de chaque avion et de l'amplitude de son retard.

Dans le cadre de cette thèse, nous nous sommes intéressée au problème d'ordonnancement des atterrissages d'avions dans deux contextes différents : un ordonnancement à la piste considérée comme une ressource indépendante, puis un ordonnancement des arrivées d'avions en considérant plusieurs pistes d'atterrissage précédées par des balises de l'espace aérien aéroportuaire.

Nous avons d'abord proposé une approche de programmation mixte en nombres entiers pour modéliser et résoudre le problème d'ordonnancement des atterrissages d'avions. Le modèle proposé intègre plusieurs contraintes opérationnelles : la séparation de sécurité entre les paires d'avions, les fenêtres de temps et des contraintes de changements de position pour garantir une certaine équité entre les avions. L'objectif est de minimiser le coût de retard, modélisé par une fonction convexe, linéaire par morceaux. Dans l'étude numérique de ce modèle, nous avons constaté que malgré l'ajout des contraintes de changement de positions (diminuant la complexité du problème), les temps de calcul de la programmation linéaire mixte en nombres entiers restent prohibitifs sur certaines instances. Pour remédier à ce problème, nous avons proposé une méthode heuristique capable de fournir des solutions de bonne qualité en des temps de calcul très courts. Cette méthode est basée sur un algorithme de planification optimiste, issu du domaine de l'apprentissage par renforcement. Dans l'étude numérique de cette méthode, nous avons comparé sa performance à celle de deux logiciels de programmation mathématique : CPLEX et SCIP. Dans cette dernière comparaison, nous avons constaté que notre méthode aboutit à des résultats comparables à ceux de CPLEX et meilleurs que ceux de SCIP.

Le deuxième problème consiste à ordonnancer les arrivées d'avions au niveau des balises de la zone d'approche aéroportuaire ainsi qu'au seuil de piste. Les balises considérées sont les points de début de l'approche initiale, appelés IAF – *Initial Approach Fix*. Pour modéliser et résoudre ce problème, nous avons proposé une approche de programmation linéaire mixte en nombres entiers. Dans notre modèle, nous considérons les contraintes de séparation et de fenêtre de temps. L'objectif est de minimiser le retard total. Dans l'étude numérique de ce modèle, nous avons comparé notre approche à la solution de base utilisée par les contrôleurs aériens pour séquencer les avions aux IAF et au seuil de piste. Les résultats préliminaires montrent qu'une approche d'optimisation est bénéfique pour réduire les retards, même dans le cas où l'affectation d'IAF n'est plus une décision à prendre et qu'elle est fixée selon l'aéroport de provenance de l'avion. Cependant, les temps de calcul pour trouver une solution optimale avec ce modèle PLNE deviennent longs pour certaines instances (jusqu'à 365 secondes).

Pour prolonger le travail de cette thèse, plusieurs perspectives sont envisagées. Tout d'abord, il serait intéressant de considérer la minimisation des coûts de déviations par rapport aux dates préférentielles d'atterrissage (en permettant des arrivées en avance), qui est plus général que la minimisation des coûts de retard que nous avons considérée. De plus, cette généralisation nous permettra de comparer notre méthode avec des approches pertinentes de la littérature qui considèrent cette fonction-objectif. Une autre généralisation possible est de considérer le cas multi-piste. En effet, les aéroports les plus congestionnés utilisent plusieurs pistes qui sont exploitées simultanément.

Une deuxième perspective de ce travail serait d'étudier le cas dynamique, dans lequel les avions à ordonnancer évoluent progressivement au cours du temps. Dans ce contexte, la connaissance de certaines données du problème, comme par exemple les dates préférentielles d'atterrissage, évolue au cours du temps. Le cas dynamique est plus réaliste, car en pratique, les avions arrivent de façon continue dans l'horizon de la planification. Une approche possible est d'intégrer notre méthode exacte et/ou heuristique dans une approche par horizon glissant (*receding horizon*), comme proposé dans [12]. Cette approche subdivise l'intervalle temporel de la séquence initiale en plusieurs sous-intervalles temporels qui se chevauchent, puis résout séquentiellement les problèmes statiques sur chaque sous-intervalle.

Enfin, une troisième piste future de recherche concerne le problème d'ordonnancement des arrivées d'avions au niveau des pistes précédées par les balises de la zone TMA. Pour ce problème, il serait pertinent de prendre en compte une fonction-objectif convexe et linéaire par morceaux, car une telle fonction permet de bien représenter les coûts réels des retards. Il serait aussi intéressant d'intégrer la méthode PLNE proposée pour ce problème dans une approche d'horizon glissant, afin de résoudre des instances de plus grande taille.

Annexe A

Ensemble de données de l'aéroport de Paris-Orly

Nous présentons dans cette annexe l'intégralité de l'ensemble de données qui a servi de base pour générer les instances du chapitre 6. Pour chaque avion, nous avons les informations suivantes : sa catégorie de turbulence de sillage (cat), l'IAF par lequel il est passé, sa date préférentielle d'atterrissage en secondes (T(s)) et au format `HH:MM:SS` (T(`HH:MM:SS`)) et sa piste d'atterrissage (rwy). Cet ensemble de données est représenté par les sept tables ci-dessous.

index	cat	IAF	T(`HH:MM:SS`)	T (s)	rwy
1	Heavy	ODILO	06 :20 :00	22800	26
2	Medium	MOLBA	06 :20 :00	22800	26
3	Heavy	MOLBA	06 :25 :00	23100	26
4	Medium	ODILO	06 :35 :00	23700	26
5	Heavy	ODILO	06 :50 :00	24600	26
6	Heavy	ODILO	07 :15 :00	26100	26
7	Heavy	ODILO	07 :15 :00	26100	26
8	Medium	ODILO	07 :20 :00	26400	26
9	Medium	MOLBA	07 :30 :00	27000	26
10	Medium	ODILO	07 :35 :00	27300	26
11	Medium	MOLBA	07 :35 :00	27300	26
12	Medium	MOLBA	07 :35 :00	27300	26
13	Medium	ODILO	07 :40 :00	27600	26
14	Medium	MOLBA	07 :40 :00	27600	26
15	Medium	MOLBA	07 :50 :00	28200	26
16	Medium	ODILO	07 :50 :00	28200	26
17	Medium	ODILO	07 :55 :00	28500	26
18	Medium	ODILO	08 :00 :00	28800	26
19	Medium	MOLBA	08 :00 :00	28800	26
20	Medium	ODILO	08 :00 :00	28800	26
21	Heavy	ODILO	08 :10 :00	29400	26
22	Medium	ODILO	08 :10 :00	29400	26
23	Medium	ODILO	08 :10 :00	29400	26
24	Medium	ODILO	08 :10 :00	29400	26
25	Medium	ODILO	08 :15 :00	29700	26
26	Medium	ODILO	08 :15 :00	29700	26
27	Heavy	ODILO	08 :15 :00	29700	26
28	Medium	ODILO	08 :20 :00	30000	26
29	Medium	MOLBA	08 :25 :00	30300	26
30	Medium	MOLBA	08 :25 :00	30300	26

Table A.1 – Ensemble de données de l'aéroport de Paris-Orly (1/7).

index	cat	IAF	T	T (s)	rwy
31	Medium	ODILO	08 :25 :00	30300	26
32	Heavy	ODILO	08 :30 :00	30600	26
33	Medium	MOLBA	08 :30 :00	30600	26
34	Medium	MOLBA	08 :35 :00	30900	26
35	Heavy	ODILO	08 :35 :00	30900	26
36	Medium	MOLBA	08 :35 :00	30900	26
37	Medium	ODILO	08 :35 :00	30900	26
38	Medium	MOLBA	08 :40 :00	31200	26
39	Medium	MOLBA	08 :45 :00	31500	26
40	Medium	ODILO	08 :50 :00	31800	26
41	Medium	ODILO	08 :50 :00	31800	26
42	Heavy	ODILO	08 :55 :00	32100	26
43	Heavy	ODILO	09 :00 :00	32400	26
44	Heavy	ODILO	09 :00 :00	32400	26
45	Medium	ODILO	09 :10 :00	33000	26
46	Medium	ODILO	09 :20 :00	33600	26
47	Medium	ODILO	09 :25 :00	33900	26
48	Medium	ODILO	09 :25 :00	33900	26
49	Medium	ODILO	09 :25 :00	33900	26
50	Medium	ODILO	09 :35 :00	34500	26
51	Medium	MOLBA	09 :40 :00	34800	26
52	Medium	ODILO	09 :50 :00	35400	26
53	Heavy	ODILO	09 :50 :00	35400	26
54	Medium	MOLBA	09 :55 :00	35700	26
55	Medium	ODILO	09 :55 :00	35700	26
56	Medium	MOLBA	10 :00 :00	36000	26
57	Medium	ODILO	10 :05 :00	36300	26
58	Heavy	ODILO	10 :10 :00	36600	26
59	Medium	ODILO	10 :10 :00	36600	26
60	Medium	MOLBA	10 :10 :00	36600	26
61	Medium	MOLBA	10 :10 :00	36600	26
62	Medium	ODILO	10 :20 :00	37200	26
63	Heavy	MOLBA	10 :20 :00	37200	26
64	Medium	MOLBA	10 :25 :00	37500	26
65	Medium	ODILO	10 :30 :00	37800	26
66	Medium	MOLBA	10 :30 :00	37800	26
67	Heavy	ODILO	10 :35 :00	38100	26
68	Medium	ODILO	10 :40 :00	38400	26
69	Medium	ODILO	10 :40 :00	38400	26
70	Medium	ODILO	10 :45 :00	38700	26
71	Medium	ODILO	10 :45 :00	38700	26
72	Medium	ODILO	10 :50 :00	39000	26
73	Medium	ODILO	10 :50 :00	39000	26
74	Medium	ODILO	10 :55 :00	39300	26
75	Medium	MOLBA	10 :55 :00	39300	26
76	Medium	ODILO	11 :00 :00	39600	26
77	Medium	ODILO	11 :00 :00	39600	26
78	Medium	MOLBA	11 :05 :00	39900	26
79	Medium	ODILO	11 :10 :00	40200	26
80	Medium	MOLBA	11 :23 :00	40980	26

TABLE A.2 – Ensemble de données de l'aéroport de Paris-Orly (2/7).

index	cat	IAF	T	T (s)	rwy
81	Medium	ODILO	11 :25 :00	41100	26
82	Medium	ODILO	11 :25 :00	41100	26
83	Medium	ODILO	11 :25 :00	41100	26
84	Heavy	ODILO	11 :29 :00	41340	26
85	Medium	MOLBA	11 :30 :00	41400	26
86	Medium	ODILO	11 :30 :00	41400	26
87	Medium	ODILO	11 :30 :00	41400	26
88	Medium	MOLBA	11 :30 :00	41400	26
89	Medium	MOLBA	11 :35 :00	41700	26
90	Medium	ODILO	11 :40 :00	42000	26
91	Medium	MOLBA	11 :40 :00	42000	26
92	Medium	ODILO	11 :45 :00	42300	26
93	Medium	MOLBA	11 :50 :00	42600	26
94	Medium	ODILO	11 :55 :00	42900	26
95	Medium	ODILO	11 :55 :00	42900	26
96	Medium	ODILO	11 :55 :00	42900	26
97	Medium	MOLBA	11 :55 :00	42900	26
98	Medium	ODILO	11 :55 :00	42900	26
99	Medium	MOLBA	12 :00 :00	43200	26
100	Medium	ODILO	12 :00 :00	43200	26
101	Medium	ODILO	12 :00 :00	43200	26
102	Medium	ODILO	12 :00 :00	43200	26
103	Medium	MOLBA	12 :05 :00	43500	26
104	Medium	ODILO	12 :10 :00	43800	26
105	Medium	MOLBA	12 :10 :00	43800	26
106	Medium	ODILO	12 :15 :00	44100	26
107	Medium	MOLBA	12 :20 :00	44400	26
108	Medium	ODILO	12 :20 :00	44400	26
109	Medium	ODILO	12 :20 :00	44400	26
110	Medium	ODILO	12 :30 :00	45000	26
111	Medium	MOLBA	12 :30 :00	45000	26
112	Medium	MOLBA	12 :35 :00	45300	26
113	Medium	ODILO	12 :35 :00	45300	26
114	Medium	MOLBA	12 :35 :00	45300	26
115	Medium	ODILO	12 :40 :00	45600	26
116	Medium	ODILO	12 :40 :00	45600	26
117	Medium	MOLBA	12 :45 :00	45900	26
118	Medium	ODILO	12 :50 :00	46200	26
119	Medium	MOLBA	12 :50 :00	46200	26
120	Medium	ODILO	12 :50 :00	46200	26
121	Medium	MOLBA	12 :55 :00	46500	26
122	Medium	ODILO	12 :55 :00	46500	26
123	Medium	ODILO	12 :55 :00	46500	26
124	Medium	ODILO	12 :55 :00	46500	26
125	Medium	ODILO	13 :00 :00	46800	26
126	Medium	ODILO	13 :00 :00	46800	26
127	Medium	ODILO	13 :10 :00	47400	26
128	Medium	MOLBA	13 :10 :00	47400	26
129	Medium	MOLBA	13 :10 :00	47400	26
130	Medium	MOLBA	13 :15 :00	47700	26

TABLE A.3 – Ensemble de données de l'aéroport de Paris-Orly (3/7).

index	cat	IAF	T	T (s)	rwy
131	Medium	ODILO	13 :20 :00	48000	26
132	Medium	ODILO	13 :20 :00	48000	26
133	Medium	MOLBA	13 :25 :00	48300	26
134	Medium	ODILO	13 :25 :00	48300	26
135	Medium	ODILO	13 :25 :00	48300	26
136	Heavy	ODILO	13 :35 :00	48900	26
137	Medium	MOLBA	13 :35 :00	48900	26
138	Heavy	MOLBA	13 :40 :00	49200	26
139	Medium	MOLBA	13 :45 :00	49500	26
140	Medium	MOLBA	13 :50 :00	49800	26
141	Medium	MOLBA	13 :50 :00	49800	26
142	Medium	ODILO	13 :55 :00	50100	26
143	Medium	VEBEK	14 :00 :00	50400	26
144	Medium	MOLBA	14 :00 :00	50400	26
145	Medium	MOLBA	14 :10 :00	51000	26
146	Medium	ODILO	14 :10 :00	51000	26
147	Medium	MOLBA	14 :25 :00	51900	26
148	Medium	ODILO	14 :25 :00	51900	26
149	Medium	MOLBA	14 :25 :00	51900	26
150	Medium	MOLBA	14 :35 :00	52500	26
151	Medium	MOLBA	14 :35 :00	52500	26
152	Medium	MOLBA	14 :35 :00	52500	26
153	Medium	ODILO	14 :35 :00	52500	26
154	Medium	ODILO	14 :40 :00	52800	26
155	Medium	MOLBA	14 :40 :00	52800	26
156	Medium	MOLBA	14 :50 :00	53400	26
157	Medium	ODILO	15 :00 :00	54000	26
158	Medium	ODILO	15 :10 :00	54600	26
159	Medium	ODILO	15 :10 :00	54600	26
160	Heavy	MOLBA	15 :20 :00	55200	26
161	Medium	MOLBA	15 :20 :00	55200	26
162	Medium	MOLBA	15 :25 :00	55500	26
163	Medium	ODILO	15 :30 :00	55800	26
164	Medium	ODILO	15 :30 :00	55800	26
165	Medium	MOLBA	15 :40 :00	56400	26
166	Medium	ODILO	15 :40 :00	56400	26
167	Medium	MOLBA	15 :45 :00	56700	26
168	Medium	ODILO	15 :55 :00	57300	26
169	Medium	MOLBA	15 :55 :00	57300	26
170	Medium	ODILO	15 :55 :00	57300	26
171	Medium	ODILO	16 :00 :00	57600	26
172	Medium	ODILO	16 :00 :00	57600	26
173	Medium	ODILO	16 :00 :00	57600	26
174	Medium	ODILO	16 :15 :00	58500	26
175	Medium	ODILO	16 :15 :00	58500	26
176	Medium	ODILO	16 :15 :00	58500	26
177	Medium	MOLBA	16 :25 :00	59100	26
178	Medium	ODILO	16 :25 :00	59100	26
179	Medium	ODILO	16 :30 :00	59400	26
180	Medium	MOLBA	16 :30 :00	59400	26

TABLE A.4 – Ensemble de données de l'aéroport de Paris-Orly (4/7).

index	cat	IAF	T	T (s)	rwy
181	Medium	MOLBA	16 :30 :00	59400	26
182	Medium	ODILO	16 :35 :00	59700	26
183	Medium	ODILO	16 :45 :00	60300	26
184	Medium	MOLBA	16 :45 :00	60300	26
185	Medium	MOLBA	16 :45 :00	60300	26
186	Medium	MOLBA	16 :55 :00	60900	26
187	Medium	MOLBA	16 :55 :00	60900	26
188	Medium	MOLBA	16 :55 :00	60900	26
189	Medium	MOLBA	16 :55 :00	60900	26
190	Medium	MOLBA	16 :55 :00	60900	26
191	Medium	ODILO	17 :00 :00	61200	26
192	Medium	MOLBA	17 :00 :00	61200	26
193	Medium	MOLBA	17 :05 :00	61500	26
194	Medium	MOLBA	17 :05 :00	61500	26
195	Medium	ODILO	17 :10 :00	61800	26
196	Medium	ODILO	17 :10 :00	61800	26
197	Medium	MOLBA	17 :15 :00	62100	26
198	Medium	ODILO	17 :25 :00	62700	26
199	Medium	MOLBA	17 :25 :00	62700	26
200	Medium	MOLBA	17 :30 :00	63000	26
201	Medium	ODILO	17 :30 :00	63000	26
202	Medium	MOLBA	17 :35 :00	63300	26
203	Heavy	MOLBA	17 :40 :00	63600	26
204	Medium	ODILO	17 :40 :00	63600	26
205	Medium	MOLBA	17 :45 :00	63900	26
206	Medium	MOLBA	17 :45 :00	63900	26
207	Medium	MOLBA	17 :50 :00	64200	26
208	Medium	ODILO	17 :50 :00	64200	26
209	Medium	VEBEK	17 :55 :00	64500	26
210	Medium	MOLBA	17 :55 :00	64500	26
211	Medium	ODILO	18 :00 :00	64800	26
212	Medium	MOLBA	18 :05 :00	65100	26
213	Medium	ODILO	18 :05 :00	65100	26
214	Medium	MOLBA	18 :05 :00	65100	26
215	Medium	MOLBA	18 :10 :00	65400	26
216	Medium	ODILO	18 :15 :00	65700	26
217	Heavy	MOLBA	18 :15 :00	65700	26
218	Medium	MOLBA	18 :15 :00	65700	26
219	Medium	MOLBA	18 :20 :00	66000	26
220	Medium	ODILO	18 :20 :00	66000	26
221	Medium	ODILO	18 :20 :00	66000	26
222	Medium	ODILO	18 :25 :00	66300	26
223	Medium	MOLBA	18 :30 :00	66600	26
224	Medium	ODILO	18 :30 :00	66600	26
225	Medium	MOLBA	18 :35 :00	66900	26
226	Heavy	MOLBA	18 :40 :00	67200	26
227	Medium	ODILO	18 :50 :00	67800	26
228	Medium	ODILO	18 :55 :00	68100	26
229	Medium	MOLBA	19 :00 :00	68400	26
230	Medium	MOLBA	19 :05 :00	68700	26

TABLE A.5 – Ensemble de données de l'aéroport de Paris-Orly (5/7).

index	cat	IAF	T	T (s)	rwy
231	Medium	VEBEK	19 :05 :00	68700	26
232	Medium	ODILO	19 :10 :00	69000	26
233	Medium	ODILO	19 :10 :00	69000	26
234	Medium	ODILO	19 :15 :00	69300	26
235	Medium	ODILO	19 :15 :00	69300	26
236	Medium	MOLBA	19 :25 :00	69900	26
237	Medium	ODILO	19 :25 :00	69900	26
238	Medium	ODILO	19 :25 :00	69900	26
239	Medium	ODILO	19 :30 :00	70200	26
240	Medium	MOLBA	19 :35 :00	70500	26
241	Medium	ODILO	19 :35 :00	70500	26
242	Medium	ODILO	19 :40 :00	70800	26
243	Medium	MOLBA	19 :40 :00	70800	26
244	Medium	ODILO	19 :45 :00	71100	26
245	Medium	ODILO	19 :45 :00	71100	26
246	Medium	ODILO	19 :50 :00	71400	26
247	Medium	ODILO	19 :55 :00	71700	26
248	Medium	MOLBA	20 :00 :00	72000	26
249	Medium	MOLBA	20 :00 :00	72000	26
250	Medium	MOLBA	20 :00 :00	72000	26
251	Medium	ODILO	20 :05 :00	72300	26
252	Medium	ODILO	20 :10 :00	72600	26
253	Medium	ODILO	20 :15 :00	72900	26
254	Medium	MOLBA	20 :17 :00	73020	26
255	Medium	MOLBA	20 :20 :00	73200	26
256	Medium	MOLBA	20 :20 :00	73200	26
257	Medium	MOLBA	20 :25 :00	73500	26
258	Medium	ODILO	20 :35 :00	74100	26
259	Medium	MOLBA	20 :35 :00	74100	26
260	Medium	MOLBA	20 :40 :00	74400	26
261	Medium	VEBEK	20 :45 :00	74700	26
262	Medium	ODILO	20 :50 :00	75000	26
263	Medium	MOLBA	20 :50 :00	75000	26
264	Medium	ODILO	20 :55 :00	75300	26
265	Medium	ODILO	21 :00 :00	75600	6
266	Medium	VEBEK	21 :05 :00	75900	26
267	Medium	VEBEK	21 :05 :00	75900	26
268	Medium	MOLBA	21 :10 :00	76200	26
269	Medium	MOLBA	21 :10 :00	76200	26
270	Medium	MOLBA	21 :10 :00	76200	26
271	Medium	MOLBA	21 :15 :00	76500	26
272	Medium	VEBEK	21 :20 :00	76800	26
273	Medium	ODILO	21 :22 :00	76920	26
274	Medium	ODILO	21 :25 :00	77100	26
275	Medium	ODILO	21 :25 :00	77100	26
276	Medium	ODILO	21 :25 :00	77100	26
277	Medium	MOLBA	21 :30 :00	77400	26
278	Medium	MOLBA	21 :30 :00	77400	6
279	Medium	ODILO	21 :35 :00	77700	26
280	Medium	MOLBA	21 :40 :00	78000	6

TABLE A.6 – Ensemble de données de l'aéroport de Paris-Orly (6/7).

index	cat	IAF	T	T (s)	rwy
281	Heavy	ODILO	21 :45 :00	78300	26
282	Medium	ODILO	21 :45 :00	78300	26
283	Medium	ODILO	21 :50 :00	78600	26
284	Medium	ODILO	21 :50 :00	78600	26
285	Medium	ODILO	21 :55 :00	78900	26
286	Medium	MOLBA	21 :55 :00	78900	6
287	Medium	MOLBA	21 :55 :00	78900	26
288	Medium	MOLBA	22 :00 :00	79200	6
289	Medium	MOLBA	22 :00 :00	79200	26
290	Medium	MOLBA	22 :00 :00	79200	26
291	Medium	MOLBA	22 :05 :00	79500	26
292	Medium	MOLBA	22 :05 :00	79500	26
293	Medium	ODILO	22 :10 :00	79800	26
294	Medium	ODILO	22 :10 :00	79800	26
295	Medium	ODILO	22 :15 :00	80100	6
296	Medium	MOLBA	22 :15 :00	80100	6
297	Medium	MOLBA	22 :15 :00	80100	6
298	Medium	MOLBA	22 :15 :00	80100	6
299	Medium	MOLBA	22 :15 :00	80100	26
300	Medium	MOLBA	22 :20 :00	80400	26
301	Medium	VEBEK	22 :25 :00	80700	26
302	Medium	MOLBA	22 :25 :00	80700	26
303	Medium	MOLBA	22 :25 :00	80700	6
304	Medium	ODILO	22 :25 :00	80700	26
305	Medium	ODILO	22 :30 :00	81000	6
306	Medium	ODILO	22 :30 :00	81000	6
307	Medium	MOLBA	22 :30 :00	81000	26
308	Medium	MOLBA	22 :35 :00	81300	26
309	Medium	MOLBA	22 :35 :00	81300	6
310	Medium	ODILO	22 :35 :00	81300	6
311	Medium	ODILO	22 :35 :00	81300	26
312	Medium	MOLBA	22 :40 :00	81600	6
313	Medium	MOLBA	22 :40 :00	81600	6
314	Medium	ODILO	22 :40 :00	81600	26
315	Medium	ODILO	22 :40 :00	81600	6
316	Medium	MOLBA	22 :40 :00	81600	6
317	Medium	MOLBA	22 :45 :00	81900	26
318	Medium	ODILO	22 :50 :00	82200	6
319	Medium	ODILO	22 :50 :00	82200	26
320	Medium	MOLBA	22 :55 :00	82500	26
321	Medium	ODILO	22 :55 :00	82500	26
322	Medium	ODILO	23 :05 :00	83100	6

TABLE A.7 – Ensemble de données de l'aéroport de Paris-Orly (7/7).

Bibliographie

[1] Ahmed, S. et Alam, S. « An evolutionary optimization approach to maximize runway throughput capacity for hub and spoke airports ». In : *Australasian conference on artificial life and computational intelligence*. Springer, 2016, p. 313-323.

[2] Allignol, C., Barnier, N., Flener, P. et Pearson, J. « Constraint programming for air traffic management : A survey ». *The Knowledge Engineering Review* 27.3 (2012), p. 361-392.

[3] Artiouchine, K., Baptiste, P. et Dürr, C. « Runway sequencing with holding patterns ». *European Journal of Operational Research* 189.3 (2008), p. 1254-1266.

[4] Avella, P., Boccia, M., Mannino, C. et Vasilyevc, I. « Time-indexed formulations for the runway scheduling problem ». *Transportation Science* 51.4 (2017), p. 1196-1209.

[5] Balakrishnan, H. et Chandran, B. « Algorithms for scheduling runway operations under constrained position shifting ». *Operations Research* 58.6 (2010), p. 1650-1665.

[6] Balakrishnan, H. et Chandran, B. « Scheduling aircraft landings under constrained position shifting ». In : *AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, CO, USA*. 2006.

[7] Beasley, J. E. « OR-Library : Distributing test problems by electronic mail ». *Journal of the Operational Research Society* 41.11 (1990), p. 1069-1072.

[8] Beasley, J. E., Sonander, J. et Havelock, P. « Scheduling Aircraft Landings at London Heathrow using a population heuristic ». *Journal of the Operational Research Society* 52.5 (2001), p. 483-493.

[9] Beasley, J. E., Krishnamoorthy, M., Sharaiha, Y. M. et Abramson, D. « Scheduling aircraft landings–The static case ». *Transportation Science* 34.2 (2000), p. 180-197.

[10] Bencheikh, G., Boukachour, J. et Alaoui, A. « Improved ant colony algorithm to solve the aircraft landing problem ». *International Journal of Computer Theory and Engineering* 3 (2011), p. 224-233.

[11] Bennell, J. A., Mesgarpour, M. et Potts, C. N. « Airport runway scheduling ». *4OR* 9.2 (2011), p. 115-138.

[12] Bennell, J. A., Mesgarpour, M. et Potts, C. N. « Dynamic scheduling of aircraft landing ». *European Journal of Operational Research* 258.1 (2017), p. 315-327.

[13] Bianco, L., Dell'Olmo, P. et Giordani, S. « Minimizing total completion time subject to release dates and sequence-dependent processing times ». *Annals of Operations Research* 86 (1999), p. 393-415.

[14] Bianco, L., Dell'Olmo, P. et Giordani, S. « Scheduling models and algorithms for TMA traffic management ». In : *Modelling and Simulation in Air Traffic Management*. Springer, 1997, p. 139-167.

[15] Bianco, L., Dell'Olmo, P. et Giordani, S. « Scheduling models for air traffic control in terminal areas ». *Journal of Scheduling* 9.3 (2006), p. 223-253.

[16] Boschetti, M., Maniezzo, V., Roffilli, M. et Bolufé Röhler, A. « Matheuristics : Optimization, Simulation and Control ». In : *Proceedings of 2009 Hybrid Metaheuristics, Lecture Notes in Computer Science, Springer*. 2009, p. 171-177.

[17] Bradley, S. P., Hax, A. C. et Magnanti, T. L. *Applied mathematical programming*. Addison-Wesley, 1977.

[18] Briskorn, D. et Stolletz, R. « Aircraft landing problems with aircraft classes ». *Journal of Scheduling* 17.1 (2014), p. 31-45.

[19] Brito Soares, I., De Hauwere, Y.-M., Januarius, K., Brys, T., Salvant, T. et Nowe, A. « Departure management with a reinforcement learning approach : Respecting CFMU slots ». In : *IEEE 18th International Conference on Intelligent Transportation Systems, Las Palmas de Gran Canaria, Spain*. 2015.

[20] Brittain, M. et Wei, P. « Autonomous aircraft sequencing and separation with hierarchical deep reinforcement learning ». In : *Proceedings of the International Conference for Research in Air Transportation, Barcelona, Spain*. 2018.

[21] Cheng, J., Hoff, A., Tittsworth, J. et Gallo, W. A. « The Development of wake turbulence re-categorization in the United States ». In : *8th AIAA Atmospheric and Space Environments Conference, Washington, D.C., USA*. 2016.

[22] Colen, J. *NASA Sector 33 application*. https://www.nasa.gov/centers/ames/Sector33/iOS/index.html. En ligne ; consulté le 21 juillet 2021.

[23] Cook, A. J. et Tanner, G. *European airline delay cost reference values*. Rapport technique. Transport Studies Group University of Westminster, London UK, 2011.

[24] Cook, A. J., Tanner, G. et Anderson, S. *Evaluating the true cost to airlines of one minute of airborne or ground delay*. Rapport technique. Transport Studies Group, EUROCONTROL et University of Westminster, London UK, 2004.

[25] Cplex, I. I. « V12. 1 : User's Manual for CPLEX ». *International Business Machines Corporation* 46.53 (2009), p. 157.

[26] De Neufville, R., Odoni, A. R., Belobaba, P. et Reynolds, T. *Airport systems : Planning, design, and management*. McGraw-Hill, 2nd edition, 2013.

[27] Dear, R. G. *The dynamic scheduling of aircraft in the near terminal area*. Rapport technique. R76-9, Flight Transportation Laboratory, MIT Cambridge MA USA, 1976.

[28] Deau, R. « Optimisation des séquences de pistes et des mouvements au sol sur les grands aéroports ». Thèse de doctorat. Institut National Polytechnique de Toulouse, France, 2010.

[29] Erzberger, H., Davis J, T. et Green, S. « Design of Center-TRACON Automation System ». In : *Proc. AGARD Guidance and Control Symposium on Machine Intelligence in Air Traffic Management, Berlin, Germany*. 1993.

[30] EUROCONTROL. *Departure Management*. https://www.eurocontrol.int/phare/public/standard_page/Departure_Mgt.html. En ligne ; consulté le 21 juillet 2021.

[31] EUROCONTROL. *RECAT-EU*. http://recat-project.eu/solutions/recat-eu. En ligne ; consulté le 21 juillet 2021.

[32] Evertse, C. et Visser, H. « Real-time airport surface movement planning : Minimizing aircraft emissions ». *Transportation Research Part C : Emerging Technologies* 79 (2017), p. 224-241.

[33] Faye, A. « A quadratic time algorithm for computing the optimal landing times of a fixed sequence of planes ». *European Journal of Operational Research* 270.3 (2018), p. 1148-1157.

[34] Faye, A. « Solving the aircraft landing problem with time-discretization approach ». *European Journal of Operational Research* 242.3 (2015), p. 1028-1038.

[35] Fischetti, M. et Fraccaro, M. « Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks ». *Computers & Operations Research* 106 (2019), p. 289-297.

[36] Furini, F., Persiani, C. A. et Toth, P. « Aircraft Sequencing Problems via a Rolling Horizon Algorithm ». In : *International Symposium on Combinatorial Optimization*. Springer, 2012, p. 273-284.

[37] Furini, F., Kidd, M. P., Persiani, C. A. et Toth, P. « Improved rolling horizon approaches to the aircraft sequencing problem ». *Journal of Scheduling* 18.5 (2015), p. 435-447.

[38] Gamrath, G., Anderson, D., Bestuzheva, K., Chen, W.-K., Eifler, L., Gasse, M., Gemander, P., Gleixner, A., Gottwald, L., Halbig, K., Hendel, G., Hojny, C., Koch, T., Bodic Le, P., Maher, S. J., Matter, F., Miltenberger, M., Mühmer, E., Müller, B., Pfetsch, M. E., Schlösser, F., Serrano, F., Shinano, Y., Tawfik, C., Vigerske, S., Wegscheider, F., Weninger, D. et Witzig, J. *The SCIP Optimization Suite 7.0*. Rapport technique. Zuse Institute Berlin Allemagne 2020. http://www.optimization-online.org/DB_HTML/2020/03/7705.html.

[39] Ghoniem, A. et Farhadi, F. « A column generation approach for aircraft sequencing problems : A computational study ». *Journal of the Operational Research Society* 66.10 (2015), p. 1717-1729.

[40] Gilbo, E. P. « Optimizing airport capacity utilization in air traffic flow management subject to constraints at arrival and departure fixes ». *IEEE Transactions on Control Systems Technology* 5.5 (1997), p. 490-503.

[41] Goodfellow, I., Bengio, Y. et Courville, A. *Deep learning*. MIT press, 2016.

[42] Gotteland, J.-B., Durand, N. et Alliot, J.-M. « Handling CFMU slots in busy airports ». In : *Proceedings of the Fifth Air Traffic Management R&D Seminar, Budapest, Hungary*. 2003.

[43] Hammouri, A. I., Braik, M. S., Azmi Al-Betar, M. et Awadallah, M. A. « ISA : A hybridization between iterated local search and simulated annealing for multiple-runway aircraft landing problem ». *Neural Computing and Applications* (2019), p. 1-21.

[44] Hancerliogullari, G., Rabadi, G., Al-Salem, A. H. et Kharbeche, M. « Greedy algorithms and metaheuristics for a multiple runway combined arrival-departure aircraft sequencing problem ». *Journal of Air Transport Management* 32 (2013), p. 39-48.

[45] Hansen, J. V. « Genetic search methods in air traffic control ». *Computers & Operations Research* 31.3 (2004), p. 445-459.

[46] Hart, P. E., Nilsson, N. J. et Raphael, B. « A formal basis for the heuristic determination of minimum cost paths ». *IEEE transactions on Systems Science and Cybernetics* 4.2 (1968), p. 100-107.

[47] Hastie, T., Tibshirani, R. et Friedman, J. *The elements of statistical learning, second edition*. Springer Series in Statistics, 2009.

[48] Hren, J.-F. « Planification optimiste pour systemes déterministes ». Thèse de doctorat. Université de Lille, France, 2012.

[49] Hren, J.-F. et Munos, R. « Optimistic planning of deterministic systems ». *European Workshop on Reinforcement Learning. Springer* (2008), p. 151-164.

[50] Hu, B. « An efficient Ant Colony algorithm based on wake-vortex modeling method for aircraft scheduling problem ». *Journal of Computational and Applied Mathematics* 317 (2017), p. 157-170.

[51] Hu, X.-B. et Chen, W.-H. « Genetic algorithm based on receding horizon control for arrival sequencing and scheduling ». *Engineering Applications of Artificial Intelligence* 18.5 (2005), p. 633-642.

[52] Hu, X.-B. et Di Paolo, E. « An efficient genetic algorithm with uniform crossover for air traffic control ». *Computers & Operations Research* 36.1 (2009), p. 245-259.

[53] Hu, X.-B. et Di Paolo, E. « Binary-representation-based genetic algorithm for aircraft arrival sequencing and scheduling ». *IEEE Transactions on Intelligent Transportation Systems* 9.2 (2008), p. 301-310.

[54] Hu, X.-B. et Paolo, E. A. D. « A ripple-spreading genetic algorithm for the aircraft sequencing problem ». *Evolutionary Computation* 19.1 (2010), p. 77-106.

[55] Ikli, S. *Library of codes, instances and data sets for the Aircraft Landing Problem (ALP)*. `http://data.recherche.enac.fr/ikli-alp/`. En ligne ; consulté le 21 juillet 2021.

[56] Ikli, S. *The aircraft landing problem instances*. `https://personnel.isae-supaero.fr/emmanuel-rachelson/alp-instances.html`. En ligne ; consulté le 21 juillet 2021.

[57] Ikli, S., Mancel, C., Mongeau, M., Olive, X. et Rachelson, E. « A mixed integer programming approach for scheduling aircraft arrivals at terminal airspace fixes and runway threshold ». In : *Proceedings of the International Conference on Project Management and Scheduling, Toulouse, France*. 2020/2021, (4 pages).

[58] Ikli, S., Mancel, C., Mongeau, M., Olive, X. et Rachelson, E. « An optimistic planning approach for the aircraft landing problem ». In : *Air Traffic Management and Systems IV, Lecture Notes in Electrical Engineering 731*. 2021, p. 173-188.

[59] Ikli, S., Mancel, C., Mongeau, M., Olive, X. et Rachelson, E. « Coupling mathematical optimization and machine learning for the aircraft landing probleme ». In : *ICRAT : 9th International Conference for Research in Air Transportation, Tampa, FL, USA*. 2020.

[60] Ikli, S., Mancel, C., Mongeau, M., Olive, X. et Rachelson, E. « The aircraft runway scheduling problem : A survey ». *Computers & Operations Research* 132 (20 pages), ID : 105336 (2021).

[61] Jiang, Y., Xu, Z., Xu, X., Liao, Z. et Luo, Y. « A schedule optimization model on multi runway based on ant colony algorithm ». *Mathematical Problems in Engineering* Volume 2014, Article ID 368208 (2014), 11 pages.

[62] Keha, A. B., Farias JR., I. R. de et Nemhauser, G. L. « Models for representing piecewise linear cost functions ». *Operations Research Letters* 32.1 (2004), p. 44-48.

[63] Khassiba, A. « Two-stage stochastic programming for aircraft arrival scheduling under uncertainty ». Thèse de doctorat. Université Toulouse 3 - Paul Sabatier, France, 2020.

[64] Khassiba, A., Bastin, F., Gendron, B., Cafieri, S. et Mongeau, M. « Extended aircraft arrival management under uncertainty : A computational study ». *Journal of Air Transportation* 27.3 (2019), p. 131-143.

[65] Kim, B., Li, L. et Clarke, J.-P. « Runway assignment by minimizing emissions in terminal airspace ». In : *AIAA Guidance, Navigation, and Control Conference, Toronto, Ontario, Canada*. 2010.

[66] Kim, B., Li, L. et Clarke, J.-P. « Runway assignments that minimize terminal airspace and airport surface emissions ». *Journal of Guidance, Control, and Dynamics* 37.3 (2014), p. 789-798.

[67] Kim, J., Kroller, A. et Mitchell, J. « Scheduling aircraft to reduce controller workload ». In : *9th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'09)*. 2009.

[68] LeCun, Y., Bengio, Y. et Geoffrey, H. « Deep learning ». *Nature* 521.7553 (2015), p. 436-444.

[69] Lee, H. et Balakrishnan, H. « A study of tradeoffs in scheduling terminal-area operations ». *Proceedings of the IEEE* 96.12 (2009), p. 2081-2095.

[70] Lenstra, J. K. et Kan, A. R. « Complexity of vehicle routing and scheduling problems ». *Networks* 11.2 (1981), p. 221-227.

[71] Lieder, A., Briskorn, D. et Stolletz, R. « A dynamic programming approach for the aircraft landing problem with aircraft classes ». *European Journal of Operational Research* 243.1 (2015), p. 61-69.

[72] Lieder, A. et Stolletz, R. « Scheduling aircraft take offs and landings on interdependent and heterogeneous runways ». *Transportation Research Part E : Logistics and Transportation Review* 88 (2016), p. 167-188.

[73] Ma, J. « Optimisation du trafic aérien dans de grands aéroport ». Thèse de doctorat. Université Toulouse 3 - Paul Sabatier, France, 2019.

[74] Ma, J., Delahaye, D., Sbihi, M., Scala, P. et Mujica Mota, M. A. « Integrated optimization of terminal maneuvering area and airport at the macroscopic level ». *Transportation Research Part C : Emerging Technologies* 98 (2019), p. 338-357.

[75] Ma, W., Bo, X., Ming, L. et Hui, H. « An efficient approximation algorithm for aircraft arrival sequencing and scheduling problem ». *Journal of Computational and Applied Mathematics, Article ID 236756* 2014 (2014), 8 pages.

[76] Malik, W., Lee, H. et Jung, Y. C. « Runway scheduling for Charlotte Douglas international airport ». In : *16th AIAA Aviation Technology, Integration, and Operations Conference, Washington D.C. USA*. 2016.

[77] Martinie, C., Palanque, P., Pasquini, A. et Ragosta, M. « Using complementary models-based approaches for representing and analysing ATM systems ». In : *2nd International Conference on Application and Theory of Automation in Command and Control Systems-ATACCS'12, London, UK*. 2012, p. 146-157.

[78] Mohleji, S. et Noam, T. « Minimizing departure prediction uncertainties for efficient RNP aircraft operations at major airports ». In : *6th AIAA Aviation Technology, Integration and Operations Conference (ATIO), Wichita, KA, USA*. 2006.

[79] Montlaur, A. Villardi de et Delgado Munoz, L. « Delay assignment optimization strategies at pre-tactical and tactical levels ». In : *Fifth SESAR Innovation Days, Bologna, Italy*. 2015.

[80] Montoya, J., Rathinam, S. et Wood, Z. « Multiobjective departure runway scheduling using dynamic programming ». *IEEE Transactions on Intelligent Transportation Systems* 15.1 (2014), p. 399-413.

[81] Munos, R. et Busoniu, L. « Optimistic planning for markov decision processes ». In : *Proceedings 15th International Conference on Artificial Intelligence and Statistics (AISTATS-12)*. 2012, p. 182-189.

[82] Neuman, F. et Erzberger, H. *Analysis of delay reducing and fuel saving sequencing and spacing algorithms for arrival traffic*. Rapport technique. TM-103880, Ames Research Center NASA USA, 1991.

[83] Pinol, H et Beasley, J. E. « Scatter Search and Bionomic Algorithms for the Aircraft Landing Problem ». *European Journal of Operational Research* 171 (2006), p. 439-462.

[84] Pohl, M., Kolisch, R. et Schiffer, M. « Runway Scheduling during Winter Operations ». *Omega* 102, Reference : 102325 (2020), (16 pages).

[85] Prakash, R., Piplani, R. et Desai, J. « An optimal data-splitting algorithm for aircraft scheduling on a single runway to maximize throughput ». *Transportation Research Part C : Emerging Technologies* 95 (2018), p. 570-581.

[86] Puterman, M. L. *Markov decision processes : discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[87] Rathinam, S., Wood, Z., Sridhar, B. et Jung, Y. « A generalized dynamic programming approach for a departure scheduling problem ». In : *AIAA Guidance, Navigation, and Control Conference, Chicago ILL, USA*. 2009.

[88] Ravidas, A., Rathinam, S. et Wood, Z. « An optimal algorithm for a two runway scheduling problem ». *Proceedings of the Institution of Mechanical Engineers, Part G : Journal of Aerospace Engineering* 227.7 (2013), p. 1122-1129.

[89] Rodríguez-Díaz, A., Adenso-Díaz, B. et González-Torre, Pilar, P. L. « Minimizing deviation from scheduled times in a single mixed-operation runway ». *Computers & Operations Research* 78 (2017), p. 193-202.

[90] Sabar, N. R. et Kendall, G. « An iterated local search with multiple perturbation operators and time varying perturbation strength for the aircraft landing problem ». *Omega* 56 (2015), p. 88-98.

[91] Salehipour, A. « An algorithm for single-and multiple-runway aircraft landing problem ». *Mathematics and Computers in Simulation* 175 (2020), p. 179-191.

[92] Salehipour, A. et Ahmadian, M. « A heuristic algorithm for the aircraft landing problem ». In : *The 22nd International Congress on Modelling and Simulation (MODSIM), Tasmania, Australia*. 2017.

[93] Salehipour, A., Modarres, M. et Moslemi Naeni, L. « An efficient hybrid metaheuristic for aircraft landing problem ». *Computers & Operations Research* 40.1 (2013), p. 207-213.

[94] Salehipour, A., Naeni, L. et Kazemipoor, H. « Scheduling aircraft landings by applying a variable neighborhood descent algorithm : Runway-dependent landing time case ». *Journal of Applied Operational Research* 1.1 (2009), p. 39-49.

[95] Service Technique de l'Aviation Civile (STAC). *La Turbulence de sillage : Note d'information technique*. https://www.stac.aviation-civile.gouv.fr/sites/default/files/nit_turbulence_de_sillage_18-10-2016_light.pdf. En ligne ; consulté le 21 juillet 2021.

[96] Shohel, A. M., Alam, S. et Barlow, M. « A cooperative co-evolutionary optimisation model for best-fit aircraft sequence and feasible runway configuration in a multi-runway airport ». French. *Aerospace* 5.3 (2018), 26 pages.

[97] Soykan, B. et Rabadi, G. « A tabu search algorithm for the multiple runway aircraft scheduling problem ». In : *Heuristics, Metaheuristics and Approximate Methods in Planning and Scheduling*. Springer, 2016, p. 165-186.

[98] Sutton, R. S. et Barto, A. G. *Reinforcement learning : An introduction*. MIT press, 1998.

[99] The Civil Aviation Authority, Gatwick Airport South. *Airspace design guidance : Noise mitigation considerations when designing PBN departure and arrival procedures*. Rapport technique. West Sussex UK, 2016.

[100] The OpenSky Network. *Open air traffic data for research*. https://opensky-network.org. En ligne ; consulté le 21 juillet 2021.

[101] Vadlamani, S. et Hosseini, S. « A novel heuristic approach for solving aircraft landing problem with single runway ». *Journal of Air Transport Management* 40 (2014), p. 144-148.

[102] Watkins, C. J. et Dayan, P. « Q-learning ». *Machine Learning* 8 (1992), p. 279-292.

[103] Zhan, Z.-H., Zhang, J., Li, Y., Liu, O., Kwok, S., Ip, W. et Kaynak, O. « An efficient ant colony system based on receding horizon control for the aircraft arrival sequencing and scheduling problem ». *IEEE Transactions on Intelligent Transportation Systems* 11.2 (2010), p. 399-412.

Publications

Articles de revue

— Sana Ikli, Catherine Mancel, Marcel Mongeau, Xavier Olive et Emmanuel Rachelson. *The aircraft runway scheduling problem : A survey.* Computers & Operations Research 132 (20 pages), ID : 105336 (2021).

— Sana Ikli, Catherine Mancel, Marcel Mongeau, Xavier Olive et Emmanuel Rachelson. *An optimistic planning approach for the aircraft landing problem.* In : Air Traffic Management and Systems IV, Lecture Notes in Electrical Engineering 731, 173-188 (2021).

Articles de conférence

— Sana Ikli, Catherine Mancel, Marcel Mongeau, Xavier Olive et Emmanuel Rachelson. *A mixed integer programming approach for scheduling aircraft arrivals at terminal airspace fixes and runway threshold.* In : Proceedings of the International Conference on Project Management and Scheduling (PMS), Toulouse, France. (2020/2021), 4 pages.

— Sana Ikli, Catherine Mancel, Marcel Mongeau, Xavier Olive et Emmanuel Rachelson. *Coupling mathematical optimization and machine learning for the aircraft landing probleme.* In : 9th International Conference for Research in Air Transportation (ICRAT), Tampa, FL, USA (2020), 7 pages.

— Sana Ikli, Catherine Mancel, Marcel Mongeau, Xavier Olive et Emmanuel Rachelson. *Approche de programmation mixte en nombres entiers pour le séquencement d'avions en atterrissage-cas statique.* In : 20ème congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Le Havre, France (2019).

Résumé — Les pistes des aéroports sont considérées comme l'un des principaux goulots d'étranglement du système aéroportuaire et l'un des facteurs qui déterminent la capacité des aéroports. Optimiser l'utilisation des pistes afin de réduire les retards motive les études du problème d'ordonnancement des atterrissages d'avions. Dans le cadre de cette thèse, nous nous intéressons au problème d'ordonnancement d'atterrissages d'avions dans deux contextes différents : un ordonnancement à la piste considérée comme une ressource indépendante, puis un ordonnancement d'arrivées d'avions en considérant plusieurs pistes d'atterrissage précédées par des balises de l'espace aérien aéroportuaire. Notre objectif est de minimiser les coûts de retard des avions. Pour le premier problème, nous considérons une représentation du coût de retard réaliste mais rarement considérée dans la littérature : une fonction convexe, linéaire par morceaux. Pour la résolution du problème, deux méthodes d'optimisation sont proposées dans cette thèse : une méthode exacte basée sur la programmation linéaire mixte en nombres entiers et une méthode heuristique basée sur un algorithme de planification optimiste issu du domaine de l'apprentissage par renforcement. Nous proposons aussi dans cette thèse des nouvelles instances réalistes et difficiles pour le problème d'ordonnancement d'atterrissage, car les instances de la littérature sont de nos jours facilement résolues avec les versions actuelles des logiciels d'optimisation. Les tests numériques effectués sur les instances proposées montrent que les retards peuvent significativement être réduits quand une approche d'optimisation est adoptée pour ordonnancer les atterrissages. Cependant, la méthode exacte requiert des temps de calcul qui deviennent prohibitifs avec la taille du problème (ici le nombre d'avions). Pour le deuxième problème, nous proposons une étude préliminaire dans laquelle nous adoptons une approche de programmation linéaire mixte en nombres entiers. La comparaison de cette approche avec la technique traditionnellement utilisée par les contrôleurs aériens révèlent encore une fois à quel point une approche d'optimisation peut être bénéfique pour réduire les retards.

Mots clés : Aide à la décision, optimisation, ordonnancement d'atterrissages d'avions, modélisation, programmation linéaire mixte en nombres entiers

Abstract — Airport runways are considered to be one of the main bottlenecks in the airport system and one of the key factors that determine airport capacity. Optimizing the utilization of the runways to reduce delays motivates the numerous studies of the aircraft landing problem. In this thesis, we focus on the problem of scheduling aircraft landings in two different contexts : scheduling landings on one runway considered as an independent resource, and scheduling aircraft arrivals on critical airspace *fixes* and at the runway threshold. The objective is to minimize the total delay cost. For the first problem, we consider a realistic delay-cost representation that is rarely considered in the literature : a convex, piecewise linear function. To solve this problem, two optimization methods are proposed in this thesis : an exact method based on mixed-integer linear programming and a heuristic method based on an optimistic planning algorithm used in reinforcement learning. We also propose in this thesis new realistic and challenging instances for the aircraft landings problem, as the instances of the literature are nowadays easily solved with current versions of optimization software. Numerical tests performed on our instances show that the delays can be significantly reduced when an optimization approach is adopted to schedule aircraft landings. However, the exact method requires prohibitive computation times with the increasing size of the instance (increasing numbers of aircraft). For the second problem, we propose a preliminary study in which we adopt a mixed-integer linear programming approach. The comparison of this approach with the technique traditionally used by air traffic controllers reveals once again how beneficial an optimization approach can be for reducing delays.

Keywords : Decision making, optimization, scheduling aircraft landings, modeling, mixed-integer linear programming

Laboratoire ENAC, 7 Avenue Edouard Belin
31400 Toulouse