Titre: Title:	Fabrication d'horaires dans les forces armées canadiennes : approche par génération de missions et réduction de réseaux
Auteur: Author:	Alexis Guigue
Date:	2000
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Guigue, A. (2000). Fabrication d'horaires dans les forces armées canadiennes : approche par génération de missions et réduction de réseaux [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8762/

UNIVERSITÉ DE MONTRÉAL

FABRICATION D'HORAIRES DANS LES FORCES ARMÉES CANADIENNES : APPROCHE PAR GÉNÉRATION DE MISSIONS ET RÉDUCTION DE RÉSEAUX

ALEXIS GUIGUE
DÉPARTEMENT DE MATHÉMATIQUES
ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
MAI 2000

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

FABRICATION D'HORAIRES DANS LES FORCES ARMÉES CANADIENNES : APPROCHE PAR GÉNÉRATION DE MISSIONS ET RÉDUCTION DE RÉSEAUX

présenté par : GUIGUE Alexis

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. SOUMIS François, Ph.D., président

M. SAVARD Gilles, Ph.D., membre et directeur de recherche

M. DESAULNIERS Guy, Ph.D., membre

DÉDICACE

À ma mère,

Heureux l'homme qui a rencontré la femme
dévouée, énergique et fière de son amour
qui lui rendra toujours présente sa jeunesse,
qui empêchera son âme de jamais se contenter,
qui saura lui rappeler sans cesse les obligations
de sa tâche, et qui, parfois même lui révèlera
son génie.

Georges Sorel.

REMERCIEMENTS

Cette page est de loin la plus difficile à écrire, principalement parce que c'est la plus ingrate. Il m'est impossible en effet d'énumérer de manière exhaustive toutes les personnes qui ont contribué au succès du long cycle de mes études supérieures, entamé voici maintenant 6 ans, et qu'achève ce mémoire. Je me contenterai donc d'en citer quelques unes, tout en m'excusant sincèrement auprès de celles qui en sont absentes.

L'amour de mes parents et de mes deux soeurs, Valérie et Myriam a été le fil d'Ariane de cette période de ma vie. Je leur dédie ma réussite, qu'ils ont forgée par toutes les multiples et intenses démonstrations de leur sentiment à mon égard.

Je remercie l'École Nationale Supérieure de l'Aéronautique et de l'Espace (ENSAE) pour m'avoir permis, dans le cadre de son programme d'échange, de venir terminer mon cursus universitaire au GERAD.

Ma présence à Montréal trouve aussi sa raison d'être dans les encouragements de M. Daniel Delahaye, chercheur à l'École Nationale de l'Aviation Civile (ENAC), qui m'a initié au monde des mathématiques appliquées et qui m'a motivé à me reconvertir dans ce domaine.

Je remercie mon directeur de recherches, M. Gilles Savard, pour son aide financière et ses nombreux conseils.

Je tiens à mentionner tout particulièrement la contribution de M. Eric Rancourt, étudiant au doctorat, à ce mémoire. Il m'a grandement aidé à mettre en place tout le support informatique de ma recherche.

Je remercie enfin toute la joyeuse équipe du GERAD auprès de laquelle j'ai passé deux années formidables.

RÉSUMÉ

L'objectif principal de ce mémoire est le développement d'une approche de résolution efficace pour le problème de fabrication d'horaires des avions de transport rencontré dans la division 1-CAD des Forces Armées Canadiennes. Ce problème présente à la fois des caractéristiques de problème de tournées de véhicules avec collectes et livraisons et fenêtres de temps et de problème de rotations d'équipage pour les transporteurs aériens.

Nous avons retenu deux formulations mathématiques équivalentes pour notre problème : une formulation multi-commodités et une formulation de partitionnement d'ensemble généralisé. Nous construisons les réseaux sous-jacents à ces deux formulations à partir d'un ensemble de missions qu'un générateur fournit *a priori*. Les missions vérifient un sous-ensemble de contraintes que nous n'avons alors plus à prendre en compte. Cependant, la cardinalité de l'ensemble des missions considérées, qu'il est possible de contrôler en ajustant les paramètres du générateur, peut s'avérer élevée, ce qui suggère de construire les réseaux de manière efficace. Nous présentons un réseau de référence et développons différents algorithmes exacts (en ce sens que nous prouvons que les arcs et les noeuds éliminés ne peuvent apparaître dans une solution réalisable du problème) pour réduire sa taille.

Pour résoudre les modèles mathématiques, nous utilisons une procédure de séparation et évaluation progressive où les bornes inférieures sont évaluées par génération de colonnes. Nous proposons quatre méthodes de branchement. Pour finir, nous considérons trois scénarii et présentons quelques résultats numériques visant à mettre en évidence la validité de notre approche et à démontrer l'efficacité des algorithmes de réduction de réseaux.

ABSTRACT

The main goal of this master's thesis is to develop an efficient solution approach for the problem of building flight schedules for the Air Transport Group of the Canadian Armed Force. This problem exhibits characteristics of Pickup and Delivery Problems with time windows and Crew Pairing Problems.

We give, for this problem, two equivalent mathematical formulations : a multi-commodity flow formulation and a generalized set partitioning formulation. The networks induced by these formulations are built upon a set of missions which is provided *a priori* by a mission generator. All these missions satisfy a subset of constraints that do not need to be considered anymore. The number of missions, which can be controled by the mission generator's parameters, should be important, so that suggests to build our networks efficiently. We present a reference network and develop some exact algorithms to reduce it (by exact, we mean that we prove that the arcs and nodes that are discarded cannot be part of any optimal solution of our problem).

To solve these problems, we propose a branch-and-bound algorithm where lower bounds are evaluated by column generation. We propose four branching methods. Finally, we present some numerical results on scenarios derived from a valid flight schedule. These results illustrate the efficiency of the proposed models and the reduction algorithms.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES

GERAD : Groupe d'Études et de Recherche en Analyse des Décisions

GTA : Groupe de Transport Aérien

PFH : Problème de Fabrication d'Horaires

PVM : Plan de Vol Mensuel

CHAPITRE 1

Introduction

Dans ce mémoire, nous abordons le problème de planification des vols pour le groupe de transport aérien de la division 1-CAD des Forces Armées Canadiennes que nous désignons par Groupe de Transport Aérien (GTA).

Ce chapitre constitue une introduction générale à ce problème. La section 1.1 décrit le contexte dans lequel se place le problème et met en évidence ses caractéristiques. L'objectif et le plan du mémoire sont exposés dans la section 1.2.

1.1 Présentation du problème

La présentation qui suit s'inspire largement de celle développée par Rancourt ([13]). Pour plus de détails, le lecteur intéressé pourra consulter aussi Rancourt ([14]) et Rancourt et Savard ([15]).

Le Groupe de Transport Aérien (GTA) a pour mission d'apporter un soutien logistique en aéronefs et en personnel navigant à l'ensemble des demandes qui lui sont soumises. Ces demandes sont de nature extrêmement variée. Elles peuvent consister, par exemple, en requêtes de transport de matériel et de personnel entre différents aéroports ou bases militaires. Il peut s'agir aussi de missions de recherche et de sauvetage. Pour satisfaire ces demandes, le GTA possède un ensemble de ressources en transport aérien réparties sur plusieurs bases sur le territoire canadien. Le GTA ne disposant pas de toutes les ressources nécessaires à la satisfaction de toutes les demandes,

celles-ci se voient attribuées un niveau de priorité. Les huit niveaux de priorité sont :

- niveau 1 : les vols d'urgence et de transport de dignitaires comme par exemple le premier ministre du Canada.

- niveau 2 : la formation des membres du GTA. Il s'agit de mission d'entraînement du personnel navigant du GTA.

- niveau 3 : les vols réguliers de réapprovisionnement dans le Nord et de réapprovisionnement des forces de maintien de la paix.

- niveau 4 : le transport aérien spécial des Forces Armées Canadiennes pour répondre aux objectifs nationaux.

- niveau 5 : les exercices conjoints de différents commandements des Forces Armées Canadiennes.

- niveau 6 : les vols réguliers de transport de passagers.

- niveau 7 : les vols réguliers de transport de fret.

- niveau 8 : les vols spéciaux n'appartenant à aucune des catégories précédentes.

Les règles concernant l'acceptation des demandes soumises au GTA sont les suivantes :

- La satisfaction d'une demande ne doit jamais se faire aux dépens d'une demande

de plus haute priorité.

- La satisfaction de demandes de même niveau de priorité se fait selon le principe de premier arrivé premier servi.

Les demandes peuvent éventuellement être combinées entre elles, et ce, bien entendu, dans un souci de réduction des coûts et d'augmentation du taux global d'acceptation. Par exemple, des demandes de transport et de personnel sont régulièrement couplées avec des missions d'entraînement du personnel navigant.

Compte tenu de toutes les demandes, le GTA produit, pour chaque mois, un Plan de Vol Mensuel (PVM) qui doit respecter idéalement les contraintes de priorité concernant ces demandes. Cette tâche est présentement dévolue à des planificateurs expérimentés qui construisent le PVM manuellement. Pour la suite de ce mémoire, nous désignons par *requête* une demande.

Un PVM détaille un ensemble de *lignes d'affectation*. Une *ligne d'affectation* représente la suite de missions effectuées par un aéronef générique d'un certain type et localisé à une base déterminée. Remarquons que les missions affectées à une même ligne d'affectation ne sont pas nécessairement toutes réalisées par le même aéronef mais elles sont toutes réalisées par des aéronefs de même type et localisés à la même base (d'où l'emploi de la terminologie aéronef générique). Le nombre de lignes d'affectation est déterminé par le commandement du GTA en fonction de la taille de la flotte, des équipages disponibles et des contraintes d'entretien des aéronefs. Il est à noter que certaines de ces lignes sont exclusivement réservées pour des missions de recherche et de sauvetage.

Les planificateurs doivent donc construire un ensemble de missions qu'ils affectent ensuite aux différentes lignes d'affectation, une mission étant une suite de segments de vol définis par une heure de début et une heure de fin. Les missions doivent respecter un certain nombre de contraintes comme les restrictions sur les combinaisons pos-

sibles de matériel, la réglementation des différentes agences canadiennes et internationales concernant le personnel navigant ([18]) et les heures d'opération des aéroports. Précisons que certaines missions sont prédéfinies et n'ont pas à être construites. C'est le cas de la plupart des missions réalisant des requêtes de priorité 1 à 5.

Pour résumer, un PVM fournit donc l'horaire de chaque aéronef générique, en plus de la description de chacune des missions. Le PVM, enfin, ne précise pas quel équipage volera chaque mission, cette responsabilité étant du ressort de chaque base. Parmi l'ensemble de tous les PVM existants, nous retenons celui de coût minimum, la définition de la fonction objectif étant dévolue au planificateur.

La figure 1.1 donne un exemple de PVM, construit à partir des quatre missions détaillées dans le tableau 1.1.

Tableau 1.1 - Exemple de missions

Missions	Requête(s) couverte(s)
Mission 1	_080206A _080206B _080206C
Mission 20	SARTRNR_080560A1
Mission 45	_080213A _080213B _080213C
Mission 343	NAT413_08560C _080202

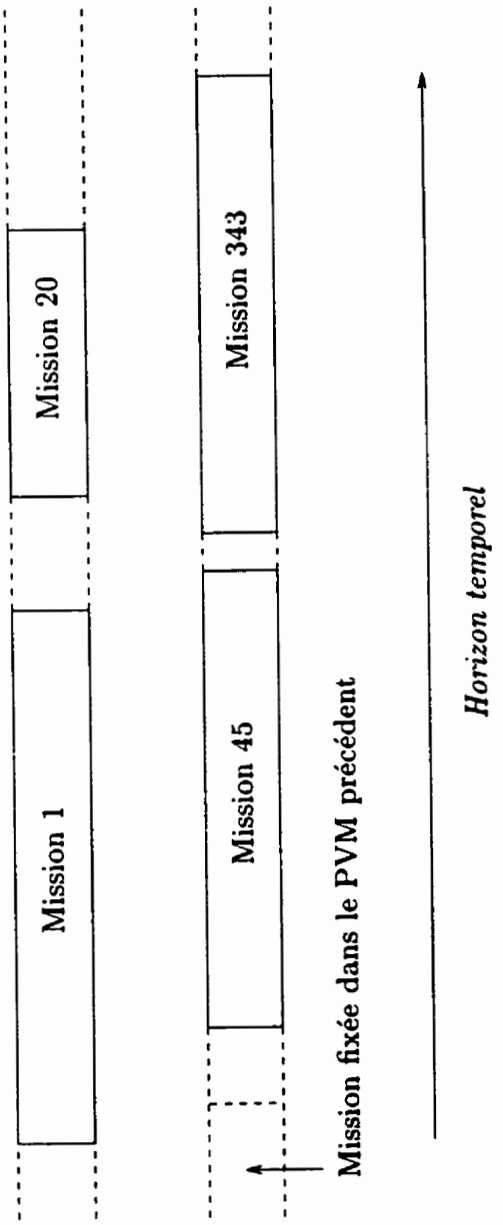

Figure 1.1 – Exemple de lignes d'affectation

1.2 Objectifs et organisation du mémoire

Le problème tel qu'exposé dans la section 1.1 possède des caractéristiques de problème de tournées de véhicule avec collectes et livraisons et de problèmes de fabrication de rotations d'équipage pour les transporteurs aériens. Desaulniers *et al.* ([5]) présentent une formulation unifiée pour les problèmes de ces classes. Cette formulation consiste en un modèle non linéaire en nombres entiers dans un réseau multi-commodités. Rancourt ([13]) dérive un modèle pour la planification des vols pour le GTA à partir de cette formulation.

Pour résoudre le modèle unifié, Desaulniers *et al.* ([5]) proposent une méthode de séparation et évaluation progressive où les bornes inférieures sont évaluées en utilisant l'algorithme de décomposition de Dantzig-Wolfe ([2]). Cet algorithme est aussi connu sous le nom de méthode de génération de colonnes. La structure des contraintes du modèle unifié se prête bien à une approche de décomposition. Les contraintes couplantes de partitionnement des tâches constituent le problème maître et chaque commodité engendre un problème de routage, modélisé sous la forme d'un réseau espace-temps. C'est à ce niveau qu'apparaît notre contribution. En effet, Rancourt ([13]) modélise les réseaux espace-temps à partir de segments de vol. Nous modélisons les réseaux à partir de missions (le chapitre 2 explicite la terminologie employée dans ce mémoire). Un générateur de missions fournit *a priori* l'ensemble de toutes les missions envisageables (c'est-à-dire réalisables par une commodité donnée), à partir desquelles nous construisons les réseaux qui interviennent dans la formulation mathématique du problème. L'objectif de cette approche est triple :

- Simplifier la modélisation : en effet, les missions vérifient déjà un certain nombre de contraintes qui n'ont donc plus à être considérées dans le modèle proposé.
- Compte tenu de cette simplification, obtenir des temps de résolution raisonnables et meilleurs que ceux de Rancourt ([13]).

- Apporter un intérêt pratique : cette modélisation s'intègre facilement dans un système d'aide à la décision. Il devient aisé pour le planificateur d'imposer ou d'interdire une ou des missions, de rajouter ou d'enlever des contraintes.

Ce mémoire est organisé comme suit. Le chapitre 2 donne une description concise du problème considéré et présente une revue de littérature pertinente à ce sujet. Le chapitre 3 détaille les modèles mathématiques développés. Nous donnons également une définition claire du concept de mission : nous précisons en particulier l'ensemble des contraintes qu'elle prend en compte. Nous présentons ensuite l'algorithme de construction des missions et terminons en comparant notre approche avec celle de Rancourt ([13]). Dans le chapitre 4, nous expliquons la manière effective avec laquelle les réseaux sont construits et montrons comment il est possible d'utiliser les propriétés intrinsèques du problème dans la construction même de ces réseaux. Dans le chapitre 5, nous appliquons cette méthode à quelques scénarii afin de valider notre approche et comparons les résultats obtenus à ceux de Rancourt ([13]). Finalement, le chapitre 6 tire quelques conclusions de cette recherche.

CHAPITRE 2

Présentation du problème

Ce chapitre a pour but essentiel de présenter le problème. À cette fin, nous détaillons dans la section 2.1 l'ensemble de la terminologie utilisée ainsi que les notations employées tout au long de ce mémoire. La section 2.2 définit les caractéristiques du problème, et en particulier, les objectifs et les contraintes. Enfin, la section 2.3 propose une revue de la littérature pertinente au sujet traité.

2.1 Définitions préalables

Nous donnons dans cette section une liste aussi complète que possible du vocabulaire employé dans ce mémoire.

Un *aéroport* correspond à tout lieu où un aéronef est susceptible de décoller ou d'atterrir dans le cadre de la mission qui lui est assignée. Il peut s'agir d'un aéroport civil, d'une base militaire appartenant aux Forces Armées Canadiennes ou aux Forces Armées d'un pays étranger. Une *base* constitue un aéroport particulier et désigne généralement une base militaire des Forces Armées Canadiennes. Les avions y sont stationnés lorsqu'ils ne sont pas en mission. Dans le chapitre 5 consacré aux résultats numériques, nous traitons l'exemple de la flotte aérienne des CC130 Hercules dont les trois bases sont Winnipeg, Greenwood et Trenton. La distinction entre *base* et *aéroport* intervient dans la définition d'une mission.

Un *service de vol* est une suite de segments et de connexions effectués par un même équipage durant une journée de travail. Un même service de vol peut participer à la couverture de plusieurs requêtes. Une *connexion* est une période séparant deux segments consécutifs faisant partie de la même journée de travail. Enfin, un *segment* est un vol sans arrêt entre deux aéroports. C'est la plus petite entité qu'il est donc possible de considérer. Rancourt ([13]) développe d'ailleurs sa modélisation à partir de segments de vol. Une *mission* est une suite de services de vol et de repos débutant et se terminant à une base donnée et réalisée par un ou plusieurs équipages opérant un même aéronef. En conséquence, une mission peut couvrir plusieurs requêtes. Le concept de mission est fondamental dans ce mémoire. Toute la modélisation, présentée dans le chapitre 3, est basée là-dessus.

Pour chaque mois, le GTA doit produire un PVM qui doit satisfaire le plus grand nombre de *requêtes* possibles. Nous dirons d'une mission qui réalise une requête qu'elle *couvre* cette requête. Notons qu'une mission peut couvrir plusieurs requêtes.

Les requêtes sont de deux types :

1. Elles peuvent être obligatoires, auquel cas elles doivent être satisfaites.
2. Elles peuvent être optionnelles.

Par hypothèse, nous supposons que chaque requête nécessite un et un seul aéronef. Cette restriction n'impose aucune perte de généralité. En effet, une requête nécessitant plusieurs aéronefs peut être décomposée en autant de requêtes. Si la requête initiale est obligatoire, les requêtes dupliquées le sont aussi. Si la requête initiale est optionnelle, alors nous introduisons le concept de *groupe de requêtes optionnelles*. Un *groupe de requêtes optionnelles* est un ensemble de requêtes optionnelles qui doivent être toutes acceptées ou rejetées. Remarquons que cette modélisation s'adapte au cas où une requête ne peut être réalisée que si une autre requête doit être effectuée au préalable. Par exemple, on pourrait avoir à transporter du matériel d'un aéroport A

à un aéroport B s'il avait préalablement été transporté à l'aéroport A.

De même, il existe deux types de mission :

1. Les *missions prédéfinies* sont des missions déjà construites et qui sont fournies en entrée au problème. Elles peuvent être soit *obligatoires*, et dans ce cas feront nécessairement partie de toute solution de notre problème, soit *optionnelles*, et dans ce cas couvrent des requêtes optionnelles.
2. Les *missions non prédéfinies.* Une mission de ce type ne peut couvrir que des requêtes de transport. Dans Rancourt ([13]), elles étaient susceptibles d'être générées au cours du processus de résolution. Dans le cadre de ce mémoire, elles sont construites *a priori* par un générateur de missions. Remarquons que les requêtes de transport peuvent être soit obligatoires, soit optionnelles.

Pour finir, considérons deux requêtes optionnelles A et B. On dit que la requête A est prioritaire à la requête B si et seulement si son niveau de priorité est plus élevé ou si elle est de même niveau mais qu'elle a été enregistrée par le GTA avant, suivant le principe du premier arrivé premier servi.

Précisons que la priorité d'un groupe de requêtes optionnelles correspond à la plus haute priorité des requêtes qui le composent (généralement les niveaux de priorité des requêtes qui composent un groupe de requêtes optionnelles sont identiques).

2.2 Définition du problème

Le problème considéré dans ce mémoire est appelé le problème de fabrication d'horaires et est abrégé PFH. Dans cette section, nous décrivons les principales caractéristiques du problème.

2.2.1 Les contraintes

Nous pouvons distinguer deux types de contraintes :

1. Les contraintes dites *locales* qui n'impliquent aucune coordination entre les lignes d'affectation. Pour fins de clarté, elles sont énumérées ci-dessous. Le lecteur trouvera plus de détails dans Rancourt ([13]) et Rancourt et Savard ([15]).

   (a) Les contraintes concernant le personnel navigant.

   (b) Les contraintes d'ouverture des aéroports.

   (c) Les contraintes de compatibilité de fret.

   (d) Les contraintes de préséance et de couplage.

   Dans la modélisation adoptée, nous ne tenons pas compte explicitement de ces contraintes. Elles sont toutes prises en compte par le générateur de missions lors de la construction de l'ensemble de missions retenues pour le PFH considéré. Précisons enfin que toutes les missions prédéfinies respectent ces contraintes.

2. Les contraintes dites *globales*. Une contrainte est dite *globale* si elle n'est pas *locale*. Par exemple, pour un aéroport civil, le nombre d'avions qui peuvent atterrir ou décoller durant une certaine période de temps est limité, ce qui n'est pas le cas des aéroports militaires. Ou encore, il faut pouvoir assurer la continuité du PVM d'un mois donné avec celui du mois précédent.

2.2.2 L'objectif

Le GTA n'étant pas une entreprise commerciale mais une unité des Forces Armées Canadiennes dont la mission est d'assurer la sécurité du territoire et de défendre la souveraineté nationale, l'objectif principal du problème est de maximiser le nombre de requêtes optionnelles acceptées tout en respectant les priorités. Une solution d'une

instance du PFH est dite optimale du point de vue des priorités si et seulement si chaque requête optionnelle acceptée n'empêche pas l'acceptation d'une requête optionnelle plus prioritaire.

Parmi toutes les solutions possibles qui maximisent cet objectif, on préférera celle qui minimise les coûts d'opération. Parmi ces coûts, on retrouve principalement les coûts d'opération des aéronefs tels que le carburant, les coûts d'entretien et les coûts associés à l'atterrissage dans certains aéroports. Le générateur de missions prend en compte l'ensemble de tous ces coûts lors de la construction des missions.

Une solution d'une instance du PFH est dite optimale si et seulement si elle est de coût minimum parmi toutes les solutions optimales du point de vue des priorités.

2.2.3 Caractéristiques du problème

Dans un premier temps, nous introduisons dans les tableaux 2.1 et 2.2 une partie des notations utilisées dans ce mémoire. Nous reprenons, dans un souci de facilité de compréhension, les notations de Rancourt ([13]). La *fenêtre de temps* d'une mission est, par définition, l'intervalle de temps durant lequel la commodité devant accomplir cette mission est susceptible de décoller de la base.

Précisons que nous supposons que toute mission prédéfinie est réalisable. Au cours de la résolution d'une instance du PFH, aucune vérification ne sera faite quant à la validité d'une mission prédéfinie. De plus, nous imposons que chaque requête optionnelle appartienne à un et un seul groupe de requêtes optionnelles. Ce qui s'exprime d'un point de vue mathématique par :

$$\bigcup_{o \in O} o = T^{OP} \text{ et } \forall (o_1, o_2) \in O^2 \text{ avec } o_1 \neq o_2, o_1 \cap o_2 = \emptyset.$$

Tableau 2.1 – Notations générales

K	l'ensemble des lignes d'affectation ou commodités
I	l'ensemble des ensembles de commodités identiques
M	l'ensemble des missions
M^{POB}	l'ensemble des missions prédéfinies obligatoires
M^k	l'ensemble des missions associées à la commodité k
M^i	l'ensemble des missions associées à l'ensemble de commodités identiques i
M^l	l'ensemble des missions dont la durée est strictement supérieure à la longueur de la fenêtre de temps
$M^c = M \setminus M^l$	l'ensemble des missions dont la durée est inférieure ou égale à la longueur de la fenêtre de temps
T	l'ensemble des requêtes
T^{OB}	l'ensemble des requêtes obligatoires
$T^{OP} = T \setminus T^{OB}$	l'ensemble ordonné des requêtes optionnelles
$O \subset P(T^{OP})$	l'ensemble ordonné des groupes de requêtes optionnelles, $P(T^{OP})$ désignant l'ensemble de tous les ensembles de requêtes optionnelles

Tableau 2.2 – Notations relatives à une mission

d_m	la durée de la mission
c_m	le coût de la mission
$[a_m, b_m]$	la fenêtre de temps pour le début de la mission $m \in M$
$T_m \subset T$	l'ensemble des requêtes couvertes par la mission $m \in M$

Cela signifie que tous les éléments de O sont disjoints deux à deux. Les requêtes optionnelles sont ordonnées suivant leur niveau de priorité respectif et les groupes de requêtes optionnelles sont ordonnés suivant le niveau de priorité de la requête la plus prioritaire du groupe.

2.3 Revue de la littérature

Nous présentons dans cette section une revue de la littérature pertinente sujet traité. La complexité du PFH vient du fait qu'il présente en même temps les caractéristiques de deux problèmes connus et eux-mêmes déjà difficiles à résoudre.

D'une part, les contraintes concernant le personnel navigant apparentent le PFH au problème de fabrication de rotations d'équipage. L'intérêt pour ce problème date des années 70. En effet, dans un contexte de concurrence forte, les compagnies aériennes se sont vigoureusement lancées à cette époque dans une politique de réduction des coûts, ce qui les a amenées à s'intéresser à la gestion des équipages. Desaulniers *et al.* ([4]) détaillent l'ensemble des approches qui ont été développées tant du point de vue de la modélisation que du point de vue algorithmique sur le problème de fabrication des rotations d'équipage. Les auteurs suggèrent un modèle non linéaire de flots dans un réseau multi-commodités avec contraintes de ressources et résolvent le problème dans le contexte de la compagnie Air France. Desaulniers *et al.* ([3]) proposent une approche de résolution dans le cadre plus général de planification et de gestion des opérations en transport aérien dont le problème de fabrication des rotations d'équipage est un cas particulier. Ils présentent un modèle de partionnement d'ensemble généralisé et proposent une méthode de résolution par séparation et évaluation progressive où les bornes sont évaluées par génération de colonnes.

D'autre part, le concept de lignes d'affectation et l'existence de requêtes de transport donnent au PFH des caractéristiques de problèmes de tournées de véhicules avec

collectes et livraisons et fenêtres de temps. Ce problème est présenté dans le détail par Desrosiers *et al.* ([9]) tandis que Desaulniers *et al.* ([7]) font une synthèse sur les plus récents développements le concernant. Les auteurs proposent un modèle multi-flots avec contraintes de temps et de capacité et donnent une liste assez complète des différentes heuristiques développées dans la littérature pour obtenir une solution réalisable d'assez bonne qualité. Parmi celles-ci, on retrouve principalement :

- Les heuristiques classiques : il s'agit principalement des méthodes constructives (*route construction*) et des méthodes de recherche locale autour de la solution courante (*route improvement*). Ces deux méthodes peuvent éventuellement être combinées entre elles.
- Les métaheuristiques comme les algorithmes génétiques ou la méthode tabou.
- Les algorithmes de regroupement (*clustering*).

Les auteurs présentent ensuite différentes méthodes de résolution exactes proposées dans la littérature pour les cas avec un seul véhicule et avec plusieurs véhicules : entre autres, une approche de génération de colonnes imbriquée dans un processus de séparation et évaluation progressive. Ils concluent en énumérant certaines applications de ce problème et en rapportant les plus récents résultats numériques.

Le lecteur non familier avec la problématique militaire pourra se référer aux articles de Solanki *et al.* ([17]) et de Rappoport *et al.* ([16]) qui donnent un bon aperçu de ses spécificités. En résumé, les unités de transport militaire doivent satisfaire des critères de flexibilité, de souplesse et de disponibilité. En effet, certaines requêtes peuvent arriver de manière imprévue. La principale conséquence est que ces unités sont généralement de taille relativement modeste. De plus, il n'y a pas ou peu de vols pour lesquels le chargement est défini et c'est une des sources de difficulté de notre problème.

Rancourt ([13]) propose pour le PFH une modélisation dans laquelle les commodités accomplissent des séquences, réalisables bien entendu, de segments de vol et de

missions (si celles-ci ont été prédéfinies). Nous proposons, comme nous l'avons déjà précisé, de générer pour chaque ligne d'affectation une séquence de missions, ce qui suppose l'existence d'un générateur automatique de mission. Cette idée de génération de missions s'apparente à la génération de *duty* ou de *services de vol* dans le domaine civil. Elle se distingue par le fait que les missions possèdent en général une fenêtre de temps qui peut être relativement large et qu'elles peuvent être d'une durée supérieure à plusieurs jours.

La modélisation que nous avons retenu pour le PFH s'inscrit directement dans la formulation unifiée proposée par Desaulniers *et al.* ([5]). Les auteurs présentent une formulation générale pour les problèmes de tournées de véhicules et d'horaires d'équipages. Cette formulation est représentée par un modèle non linéaire de flots dans un réseau multi-commodités avec contraintes de ressources. La résolution de ce modèle utilise une méthode de séparation et évaluation progressive pour laquelle les bornes inférieures sont calculées à l'aide d'une extension du principe de décomposition de Dantzig-Wolfe ([2]). Ce modèle est un problème de partionnement d'ensemble avec contraintes supplémentaires où les colonnes correspondent aux points extrêmes des sous-problèmes de la formulation originale.

La décomposition de Dantzig-Wolfe est aussi connue sous le nom de méthode de génération de colonnes. Celle-ci est à l'origine d'une nombreuse littérature. Citons par exemple Barnhart *et al.* ([1]) qui proposent un état de l'art dans le domaine de la génération de colonnes. Les auteurs présentent une formulation générique d'un problème en nombres entiers et détaillent les différentes étapes d'une méthode de génération de colonnes appliquée à ce problème. Cette formulation comprend comme cas particuliers le problème de partionnement, de partionnement généralisé ou encore de recouvrement. La solution obtenue après la résolution de la relaxation continue n'étant pas nécessairement entière, les auteurs discutent de différentes stratégies de branchement pertinentes.

D'un point de vue pratique, parmi les difficultés rencontrées par une méthode de génération de colonnes figure la convergence relativement lente du processus de résolution. De nombreuses itérations sont nécessaires pour obtenir une bonne approximation du polyèdre dual, d'autant plus que les problèmes que nous traitons en pratique sont souvent fortement dégénérés. du Merle *et al.* ([10]) proposent un algorithme de stabilisation des variables duales qui combine une procédure de perturbation des contraintes primales et une procédure de pénalisation des variables duales. La particularité de cette approche tient dans le fait que le programme mathématique modifié reste linéaire.

La modélisation des réseaux espace-temps dans la formulation unifiée peut s'avérer très complexe. Par complexe, nous entendons un grand nombre de ressources. Chaque itération d'une méthode de génération de colonnes nécessite la résolution d'un oracle qui renvoie une (ou des) colonne(s) de coût réduit négatif. Dans ce mémoire, l'oracle consiste en un problème de plus court chemin avec ressources qui est résolu par un algorithme de programmation dynamique dont la complexité augmente considérablement avec le nombre de ressources. Nagih et Soumis ([12]) proposent une approche heuristique basée sur une réduction de l'espace des états par agrégation des ressources. Cette approche semble prometteuse, mais n'a pas encore été validée par des résultats numériques.

Nous terminons cette section par l'article de Desaulniers *et al.* ([6]) qui résume l'ensemble des techniques qui ont été développées pour accélérer le processus de résolution d'une méthode de génération de colonnes. En fait, les auteurs se placent plus spécifiquement dans le cadre de la formulation unifiée de Desaulniers *et al.* ([5]) qui, nous le rappelons, modélise l'ensemble des problèmes déterministes de tournées de véhicules et de confections d'horaires d'équipages. Les stratégies d'accélération interviennent au niveau des cinq phases du processus de résolution : le pré-traitement, la

résolution du (des) sous-problème(s), la gestion du problème-maître, le branchement et la post-optimisation.

CHAPITRE 3

Modélisation

Dans ce chapitre, nous détaillons l'ensemble de l'approche qui a été retenue pour modéliser le PFH. Dans la section 3.1, deux modèles mathématiques sont présentés pour le PFH, soit un modèle de multiflots en nombres entiers mixte et un modèle de partitionnement d'ensemble généralisé. Chacun de ces modèles nécessite la construction de plusieurs réseaux espace-temps qui décrivent l'ensemble des tournées réalisables par un aéronef générique. Nous expliquons la manière dont sont construits ces réseaux dans la section 3.2. À la section 3.3, nous comparons ces modèles à ceux de Rancourt ([13]).

3.1 Modèles mathématiques

Les modèles que nous présentons supposent l'existence de réseaux espace-temps $G^k = (V^k, A^k)$ représentant l'ensemble des chemins (un chemin est une séquence de missions) réalisables par la commodité k. La construction de ces réseaux est détaillée à la fin de ce chapitre et dans le chapitre 4. Les notations $o(k)$ et $d(k)$ désignent respectivement la source et le puits associés au réseau G^k.

3.1.1 Modèle de multiflots

À chaque aéronef générique k, à chaque arc (i,j) appartenant à A^k et à chaque requête $t \in T$ est associée la constante a_{ij}^{kt} qui prend la valeur 1 si l'arc couvre la requête et 0 sinon, et la variable de flots X_{ij}^k qui prend la valeur 1 si l'arc est retenu dans la solution entière courante, et 0 sinon. c_{ij}^k représente le coût de l'arc (i,j).

Pour chaque requête $t \in T$, nous définissons la constante b_o^t qui vaut 1 si la requête t appartient au groupe de requêtes optionnelles o et 0 sinon. À chaque groupe de requêtes optionnelles $o \in O$ est associée la variable binaire Z_o qui prend la valeur 1 si le groupe de requêtes optionnelles est rejeté, 0 sinon.

Nous rappelons que l' objectif consiste à obtenir une solution optimale du point de vue des priorités. Nous utilisons donc, à l'instar de Rancourt ([13]), des pénalités pour le rejet des groupes de requêtes optionnelles. Ces pénalités sont choisies de telle sorte que toute solution optimale du programme mathématique le soit aussi d'un point de vue des priorités. Rancourt ([13]) propose deux algorithmes pour calculer ces pénalités, qui sont notées $c_o, o \in O$.

Le rôle de la variable Y est de compter le nombre de groupes de requêtes optionnelles rejetées. Cette variable est introduite pour définir le branchement sur le nombre de requêtes acceptées qui est présenté au chapitre 5.

Enfin, pour chaque aéronef générique, nous introduisons la ressource T^k. Dans le modèle proposé, il n'y a effectivement qu'une ressource, qui représente un instant sur l'horizon temporel. Et pour chaque noeud i du réseau G^k, la valeur prise par T^k est notée T_i^k.

Compte tenu de ce qui précède, nous obtenons le modèle suivant :

$$\min \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij}^k X_{ij}^k + \sum_{o \in O} c_o Z_o \tag{3.1}$$

sujet à :

$$\sum_{k \in K} \sum_{(i,j) \in A^k} a_{ij}^{kt} X_{ij}^k + \sum_{o \in O} b_o^t Z_o = 1, \quad \forall t \in T \tag{3.2}$$

$$\sum_{o \in O} Z_o = Y, \tag{3.3}$$

$$Y \quad \text{entier} \tag{3.4}$$

$$Z_o \in \{0,1\}, \quad \forall o \in O \tag{3.5}$$

$$\sum_{j:(o(k),j) \in A^k} X_{o(k)j} = 1, \quad \forall k \in K \tag{3.6}$$

$$\sum_{j:(j,d(k)) \in A^k} X_{jd(k)} = 1, \quad \forall k \in K \tag{3.7}$$

$$\sum_{j:(j,i) \in A^k} X_{ji}^k - \sum_{j:(i,j) \in A^k} X_{ij}^k = 0, \quad \forall k \in K, \forall i \in V^k \setminus \{o(k), d(k)\} \tag{3.8}$$

$$X_{ij}^k (f_{ij}^k(T_i^k) - T_j^k) \leq 0, \quad \forall k \in K, \forall (i,j) \in A^k \tag{3.9}$$

$$a_i^k \leq T_i^k \leq b_i^k, \quad \forall k \in K, \forall i \in V^k \tag{3.10}$$

$$X_{ij}^k \in \{0,1\}, \quad \forall k \in K, \forall (i,j) \in A^k. \tag{3.11}$$

Les contraintes (3.2) assurent que chaque requête obligatoire et chaque groupe de requêtes optionnelles accepté est affecté à une ligne d'affectation. La contrainte (3.3) ajuste la variable Y. Les contraintes (3.4), (3.5) et (3.11) imposent l'intégrité des variables de décision. Les contraintes (3.6)-(3.8) assurent la conservation de flot ainsi que l'envoi d'une unité de flot de l'origine à la destination dans chaque réseau G^k. Les contraintes (3.9) traduisent la compatibilité entre les variables de flots et de ressources. La fonction d'extension f_{ij}^k est définie par :

$$f_{ij}^k(T_i^k) = T_i^k + d_{ij}^k \ \forall k, \forall (i,j) \in A^k. \tag{3.12}$$

où d_{ij}^k désigne la durée associée à l'arc (i,j) pour la commodité k. Enfin, les contraintes (3.10) imposent le respect des fenêtres de ressources.

La fonction objectif (3.1) comprend deux termes. Le premier permet d'assurer que parmi toutes les solutions optimales du point de vue des priorités, nous retenons celle qui minimise les coûts d'opération. Le deuxième calcule la pénalité totale associée au rejet des groupes de requêtes optionnelles

Détaillons l'ensemble des différences entre le modèle (3.1)-(3.11) et celui de Rancourt ([13]). Principalement, nous notons qu'un ensemble de contraintes a disparu. Pour simplifier la modélisation des réseaux, déjà fort complexes, Rancourt transfère une partie des contraintes dans le problème maître (ce qui ne pose aucunes difficultés d'un point de vue théorique, la décomposition de Dantzig-Wolfe s'applique en effet à n'importe quelle partition de l'ensemble des contraintes). Ces contraintes assurent que la livraison du fret d'une requête de transport acceptée est effectuée exactement une fois. Dans le cadre de notre modélisation qui utilise des missions construites *a priori*, ces contraintes deviennent redondantes avec les contraintes (3.2). De plus, nous pouvons remarquer que nous n'utilisons qu'une seule ressource, ce qui réduit considérablement la complexité des réseaux. Rappelons en effet que la résolution des sous-problèmes, dans notre cas, consiste en la résolution d'un problème de plus court chemin avec ressources réalisée à l'aide d'un algorithme de programmation dynamique dont la complexité augmente considérablement avec le nombre de ressources.

3.1.2 Modèle de partionnement d'ensemble

Le PFH peut aussi être formulé en utilisant des variables associées aux chemins réalisables des réseaux G^k. Par définition, un chemin p du réseau G^k est dit réalisable si et seulement si il existe, dans l'ensemble défini par les contraintes (3.6)-(3.11), un vecteur X^k avec $X^k_{ij} = 1$ si l'arc (i, j) est dans le chemin p et 0 sinon.

Pour toute commodité $k \in K$, nous définissons Ω^k comme l'ensemble des chemins réalisables du réseau G^k. Pour tout chemin $p \in \Omega^k$, nous notons c^k_p le coût du chemin p qui est égal à la somme des coûts des arcs qui le composent. Pour tout $k \in K$, $p \in \Omega^k$ et $t \in T$, nous notons a^{kt}_p la constante qui est égale au nombre de fois où la requête t est couverte par le chemin p. En général, a^{kt}_p prend la valeur 0 ou 1. Si la constante a^{kt}_p est supérieure à 1, alors le chemin p contient un cycle.

Enfin, nous introduisons, pour tout $k \in K$ et pour tout $p \in \Omega^k$, les variables binaires de chemin θ^k_p. Cette variable prend la valeur 1 si le chemin p est retenue par la solution entière courante et 0 sinon.

Il vient alors :

$$\min \sum_{k \in K} \sum_{p \in \Omega^k} c^k_p \theta^k_p + \sum_{o \in O} c_o Z_o \tag{3.13}$$

sujet à :

$$\sum_{k\in K}\sum_{p\in\Omega^k} a_p^{kt}\theta_p^k + \sum_{o\in O} b_o^t Z_o = 1, \quad \forall t \in T \tag{3.14}$$

$$\sum_{o\in O} Z_o = Y, \tag{3.15}$$

$$Y \quad \text{entier} \tag{3.16}$$

$$Z_o \in \{0,1\}, \quad \forall o \in O \tag{3.17}$$

$$\sum_{p\in\Omega^k} \theta_p^k = 1, \quad \forall k \in K \tag{3.18}$$

$$\theta_p^k \in \{0,1\}, \quad \forall k \in K, \forall p \in \Omega^k. \tag{3.19}$$

L'interprétation de la fonction objectif est identique à celle de la formulation multi-flots. Les contraintes (3.14) assurent que chaque requête obligatoire et chaque groupe de requêtes optionnelles accepté est couvert exactement par une ligne d'affectation. Les contraintes (3.18) imposent qu'un seul chemin soit choisi par ligne d'affectation. Enfin, les contraintes (3.19) imposent l'intégrité des variables de chemin.

Tel qu'expliqué dans Desaulniers *et al.* ([5]), cette formulation est équivalente à celle donnée par les équations (3.1)-(3.11). Elle s'obtient en appliquant une généralisation du principe de décomposition de Dantzig-Wolfe.

Lorsque plusieurs commodités sont identiques, Desaulniers *et al.* ([5]) montrent qu'il est possible de les agréger afin de réduire le nombre de variables de chemin et de contraintes. Soit $K' = \{k_1, ..., k_m\} \subset K$ un ensemble de m commodités identiques, nous éliminons de la formulation les variables θ_k^p, $k \in K' \setminus \{k_1\}$, nous éliminons aussi les contraintes (3.19), $k \in K' \setminus \{k_1\}$ et nous les remplaçons par la contrainte (3.20) pour $k = k_1$, soit :

$$\sum_{p\in\Omega^{k_1}} \theta_p^{k_1} = m. \tag{3.20}$$

Pour la suite du mémoire, nous préciserons lorsque nous considérons que les commodités sont désagrégées.

3.2 Le générateur de missions

L'ensemble des requêtes de transport obligatoires et optionnelles ainsi que leurs caractéristiques sont fournies en entrée au générateur de missions qui produit alors un ensemble de missions satisfaisant les contraintes dites locales et qui sont rappelées dans la section 3.2.1.

3.2.1 Les contraintes locales

Les contraintes locales sont :

1. Les contraintes d'équipages : elles ont trait à la réglementation concernant le travail du personnel navigant. Elles concernent principalement la durée maximale d'un service de vol et la durée minimale d'un repos entre deux services de vol.
2. Les contraintes d'ouverture des aéroports restreignent les heures auxquelles un aéronef peut atterrir ou quitter certains aéroports.
3. Les contraintes de compatibilité de fret imposent qu'en aucun temps le fret chargé dans un aéronef ne soit incompatible avec celui-ci. De plus, nous dirons que deux requêtes de transport sont compatibles si c'est la même configuration qui les supporte. Remarquons qu'il est possible de modifier la configuration d'un aéronef au cours d'une mission, mais cela se fait rarement, principalement à cause du coût en temps d'une telle opération.

4. Les contraintes de préséance imposent que le chargement précède le déchargement. De plus, pour certains types de fret, l'ordre de chargement doit être pris en compte. Par exemple, si le fret A a été chargé avant le fret B, alors le fret B doit être déchargé avant le fret A.
5. Les contraintes de couplage imposent que le chargement et le déchargement soient effectués par le même aéronef.
6. Les contraintes de capacité limitent la quantité de passagers et de fret à bord de l'aéronef.

3.2.2 La construction des missions

La construction des missions s'effectue en deux étapes. Dans un premier temps, nous déterminons l'ensemble des patrons de mission. Puis à partir de chaque patron, nous construisons, si cela est possible les missions associées. Nous détaillons maintenant ces deux étapes.

1. **Construction des patrons de missions.** Tout d'abord, définissons ce que nous appelons un patron. Un patron est une séquence de collectes et de livraisons respectant les contraintes de compatibilité, de préséance et de couplage qui sont détaillées dans la section 3.2.1. Par exemple, considérons $R1$ et $R2$, deux requêtes de transport, et $L1$ et $L2$, les livraisons associées et $C1$ et $C2$ les collectes associées.

   Si $R1$ et $R2$ sont compatibles, alors les patrons possibles sont :

$$C1 \longrightarrow L1$$

$$C2 \longrightarrow L2$$

$$C1 \longrightarrow L1 \longrightarrow C2 \longrightarrow L2$$

$$C2 \longrightarrow L2 \longrightarrow C1 \longrightarrow L1.$$

Si de plus la capacité de l'aéronef permet de charger le fret des collectes $C1$ et $C2$, alors les patrons suivants se rajoutent :

$$C1 \longrightarrow C2 \longrightarrow L1 \longrightarrow L2$$

$$C1 \longrightarrow C2 \longrightarrow L2 \longrightarrow L1$$

$$C2 \longrightarrow C1 \longrightarrow L1 \longrightarrow L2$$

$$C2 \longrightarrow C1 \longrightarrow L2 \longrightarrow L1.$$

Dans le cas où $R1$ et $R2$ ne sont pas compatibles (et en excluant un changement de configuration éventuellement possible), alors les seuls patrons envisageables sont :

$$C1 \longrightarrow L1$$

$$C2 \longrightarrow L2.$$

L'exemple ci-dessus montre que la contrainte de compatibilité est très restrictive. Elle limite grandement le nombre de patrons. Cette contrainte est présente entre plusieurs paires de requêtes en pratique.

Une fois cette étape terminée, nous vérifions la réalisabilité du patron, compte tenu des fenêtres de temps des requêtes et de la durée des segments de vol (c'est une condition nécessaire pour retenir le patron pour la phase suivante).

2. **Construction des missions**

   À toute mission est associée une base. Ce qui signifie qu'un aéronef qui effectue une mission doit partir de la base qui lui est associée et y revenir. La construction d'une mission suit donc les étapes suivantes :

   (a) Considérer un patron.

(b) Associer une base au patron.

(c) Construire une mission ou plusieurs (si elles existent) respectant les contraintes d'équipage et d'ouverture des aéroports.

Remarquons qu'un couple patron-base ne donne pas nécessairement une mission, mais peut en donner plusieurs. En effet, dans la séquence de segments de vol, nous pouvons être amenés à insérer (dans le but de respecter les contraintes d'équipage) des changements d'équipage ou des périodes de repos. Notons que les périodes de repos ne peuvent avoir lieu qu'en dehors de la base fixée initialement (en effet, imposer un repos à la base de départ revient à construire une nouvelle mission), et que les changements d'équipage, eux ne peuvent avoir lieu qu'à la base. La possibilité d'insérer des repos et des changements d'équipage augmente légèrement la combinatoire et peut donner lieu à la génération de plusieurs missions. Cette insertion est bien entendu faite de manière pertinente.

Nous supposons pour la suite que la fenêtre de temps de départ d'une mission se réduit à un intervalle. Si tel n'est pas le cas, nous dupliquons autant de fois que nécessaire la mission considérée.

D'un point de vue pratique, la génération des patrons et des missions se réalise par l'énumération explicite des chemins dans des réseaux avec contraintes de ressources. Plus précisément, dans le cas des patrons, les noeuds du graphe représentent les collectes ou les livraisons. Pour assurer une représentation "équitable" des requêtes dans les patrons générés, chacune des collectes devient le noeud source. Nous fixons, pour chaque graphe ainsi constitué, une limite supérieure sur le nombre de patrons générés ainsi que sur le nombre de requêtes contenues dans un patron. Dans le cas des missions, les noeuds du graphe représentent les collectes et les livraisons du patron considéré et le noeud source et le noeud puits la base.

Nous avons donné dans cette section un bref aperçu du générateur de missions. À ce stade, nous disposons donc d'un ensemble de missions dont nous pouvons faire varier de manière assez importante la cardinalité, en jouant sur les différents paramètres du générateur. Les caractérisques de chacune des missions sont la fenêtre de départ, la durée, le coût et la liste des requêtes couvertes. Il reste à construire maintenant les réseaux, ce que nous présentons au chapitre 4.

3.3 Discussion sur la modélisation retenue

Dans cette section, nous comparons les deux modèles qui ont été proposés pour résoudre le PFH, c'est-à-dire celui présenté par Rancourt ([13]) et celui présenté dans ce mémoire. Ces deux modèles sont résolus en utilisant la décomposition de Dantzig-Wolfe. Les contraintes couplantes constituent le problème maître et les réseaux espace-temps G^k, la base des sous-problèmes. Nous concentrons notre discussion sur ces deux points.

3.3.1 Le problème maître

Notre modélisation réduit, en termes de contraintes, la taille du problème maître d'autant de requêtes de transport que le scénario considéré en comporte. Ce nombre peut être très variable. Dans les scénarii considérés par Rancourt ([13]), le nombre total de requêtes de transport par rapport au nombre total de requêtes peut atteindre 50 pour cent.

3.3.2 Les sous-problèmes

La modélisation des contraintes locales au niveau des réseaux espace-temps G^k se fait par l'intermédiaire de ressources. À chaque itération d'un processus de résolution d'une décomposition de Dantzig-Wolfe, l'oracle est appelé. La résolution du sous-problème dans notre cas, consiste en la résolution d'un problème de plus court chemin avec ressources sur le réseau G^k. L'algorithme implanté dans l'optimiseur que nous utilisons est un algorithme d'étiquetage basé sur le principe d'optimalité de Bellman. La dimension d'une étiquette est égale au nombre de ressources plus une unité (cette dernière conserve le coût réduit). Partant du noeud source $o(k)$, l'algorithme construit de manière itérative des étiquettes qui correspondent aux chemins partiels du noeud source $o(k)$ au noeud courant. Si les fonctions d'extension des ressources sont croissantes, alors il est possible d'appliquer à chaque noeud, le principe de dominance à l'ensemble des étiquettes (en fait, la dominance peut n'être appliquée que sur un sous-ensemble des ressources, ce qui constitue alors une stratégie de résolution heuristique), ce qui permet de réduire leur nombre. Les étiquettes non dominées sont conservées. Desrochers *et al.* ([8]), Desrosiers *et al.* ([9]) et enfin Desaulniers *et al.* ([5]) fournissent plus de détails sur cet algorithme.

La complexité d'un tel algorithme et la place mémoire occupée au cours de la résolution d'une instance augmentent avec le nombre de ressources. Dans le cadre de notre modélisation, nous n'avons qu'une seule ressource et ce, quelque soit le scénario considéré. Rancourt ([13]) prend en compte explicitement et non implicitement comme nous le faisons les contraintes dites locales dans la modélisation des réseaux. Cela l'amène à considérer, en fonction du scénario, un ensemble de ressources dont la cardinalité varie de 1 à 20. Nous diminuons donc le nombre de ressources mais en contrepartie, la taille des réseaux est augmentée (c'est-à-dire le nombre d'arcs et de noeuds), compte tenu du fait que toutes les missions sont énumérées en théorie.

Il se peut très bien qu'il ne soit pas envisageable, pour des raisons d'espace mémoire ou de temps de calcul, d'énumérer toutes les missions. Il est également possible que la cardinalité de l'ensemble des missions soit trop importante pour garantir une résolution efficace en terme de temps de calcul. Ces deux éventualités risquent d'être fréquentes en pratique et ce, pour trois raisons :

- Le nombre élevé de requêtes de transport.
- Les fenêtres de temps souvent très large de ces requêtes.
- Le fait qu'une mission puisse s'étaler sur plusieurs jours et couvrir plusieurs requêtes.

Nous sommes alors obligés de ne considérer qu'un sous-ensemble de l'ensemble des missions, dont nous contrôlons la cardinalité par l'ajustement des différents paramètres du générateur de missions. Nous sommes donc sous-optimal. Être sous-optimal n'est pas nécessairement trop préjudiciable puisque durant le processus de séparation et évalutation progressive, l'arbre de branchement est rarement exploré dans sa totalité. Toutefois, pour ne pas avoir une différence relative trop importante par rapport à la valeur de la solution optimale entière, nous sommes amenés à considérer un sous-ensemble de missions de cardinalité suffisante. La taille des instances du problème peut donc être relativement grande, d'où l'intérêt de construire les réseaux G^k de manière efficace.

CHAPITRE 4

Construction des réseaux espace-temps

Nous proposons dans ce chapitre différents algorithmes pour construire les réseaux espace-temps. Nous partons d'un algorithme de référence et expliquons la manière dont nous l'avons amélioré.

La section 4.1 présente l'approche qui a été retenue pour modéliser les réseaux. Chacune des sections 4.2, 4.3, 4.4 et 4.5 détaille un algorithme de construction de réseaux que nous notons respectivement par A_1, A_2, A_3 et A_4 . La section 4.2 décrit l'algorithme de référence qui servira de base comparative pour les résultats numériques. La section 4.3 propose une agrégation des réseaux construits à l'aide de l'algorithme de la section 4.2. Dans les sections 4.4 et 4.5, nous mettons en évidence les propriétés intrinsèques du PFH et montrons comment nous pouvons les utiliser dans le cadre de la construction des réseaux. Enfin, dans la section 4.6, nous faisons une synthèse du chapitre.

Précisons que ce chapitre est assez technique puisque les algorithmes sont présentés dans le détail, mais le lecteur trouvera au début de chaque section une présentation qui se veut aussi claire que possible des idées sous-jacentes à ces algorithmes.

4.1 Modélisation des réseaux

Soit M l'ensemble de toutes les missions, soit M^k l'ensemble des missions associées à la commodité k. Nous avons bien entendu :

$$M = \bigcup_{k \in K} M^k.$$

À chaque commodité $k \in K$ est associé un réseau espace-temps $G^k = (V^k, A^k)$, où V^k représente l'ensemble des noeuds et A^k l'ensemble des arcs. Le qualificatif espace-temps signifie que chaque noeud du graphe G^k représente, entre autres, un lieu (dans le problème considéré, il s'agit d'un aéroport) et une période de temps ou fenêtre de temps. Nous explicitons maintenant les ensembles V^k et A^k.

4.1.1 Description des ressources

Nous avons, avec la modélisation choisie, intégré toutes les contraintes locales dans la construction des missions. Il reste seulement maintenant à représenter des séquences de missions. La ressource T^k a été introduite dans cette intention.

4.1.2 Description des noeuds

L'ensemble V^k se partitionne en deux sous-ensembles, $V^k_{mission}$ et V^k_{base}, auxquels se rajoutent le noeud source $o(k)$ et le noeud puits $d(k)$:

$$V^k = V^k_{base} \cup V^k_{mission} \cup \{o(k), d(k)\}.$$

À chaque mission $m \in M^k$ est associé un noeud *mission* et deux noeuds *base*. Par la suite, nous verrons qu'en fait, à un noeud *base*, ne correspond pas nécessairement

une et une seule mission. Le premier noeud *base* représente le départ de la base à laquelle la commodité k est rattachée et le deuxième noeud représente le retour à cette même base.

Le tableau 4.1 précise enfin la fenêtre de ressources pour chacun de ces types de noeuds.

Tableau 4.1 – Fenêtre de ressources pour un noeud $i \in V^k$

Type du noeud $i \in V^k$	Fenêtre de la ressource T^k
mission	Fenêtre de la mission m, $[a_m, b_m]$
base	$]-\infty, +\infty[$

Plus précisément, la fenêtre de temps des noeuds *base* correspond à l'horizon temporel considéré H, c'est-à-dire :

$$H = [\min_{m \in M}(a_m), \max_{m \in M}(b_m + d_m)].$$

4.1.3 Description des arcs

L'ensemble A^k se partitionne en trois sous-ensembles, $A^k_{mission}$, A^k_{repos} et $A^k_{preparation}$. Donc,

$$A^k = A^k_{mission} \cup A^k_{repos} \cup A^k_{preparation}.$$

Un arc de type *mission* représente la réalisation d'une mission. Un arc de type *repos* signifie que la commodité reste à sa base pendant une certaine période de temps. Enfin, un arc de type *preparation* signifie que la commodité s'apprête à effectuer une mission. Soit alors $(i, j) \in A^k$, nous précisons son type dans le tableau 4.2.

Tableau 4.2 – Relation entre les types d'arcs et les types de noeuds

Type du noeud i	Type du noeud j	Type de l'arc (i,j)
mission	*base*	*mission*
base	*mission*	*preparation*
base	*base*	*repos*
mission	*mission*	*

L'astérisque * dans le tableau 4.2 signifie que nous ne considérons aucun arc entre deux missions. Une mission ayant été effectuée, la commodité k se retrouve à sa base et se rend disponible pour effectuer une autre mission.

Le tableau 4.3 précise la consommation de ressources de chacun de ces types d'arcs.

Tableau 4.3 – Consommation de ressources pour un arc $(i,j) \in A^k$

Type de l'arc $(i,j) \in A^k$	Consommation de la ressource T^k
mission	durée de la mission m, d_m
repos	0
preparation	0

La consommation du type d'arc *preparation* est nulle car le coût en temps de la préparation d'une mission est inclus dans la durée de cette mission.

Le tableau 4.4 précise le coût pour chacun de ces types d'arcs.

4.1.4 La modélisation d'une mission

La mission $m \in M^k$, compte tenu de ce qui précède, peut se modéliser comme indiqué dans la figure 4.1.

Tableau 4.4 – Coût pour un arc $(i, j) \in A^k$

Type de l'arc $(i, j) \in A^k$	Coût
mission	coût de la mission m, c_m
repos	0
preparation	0

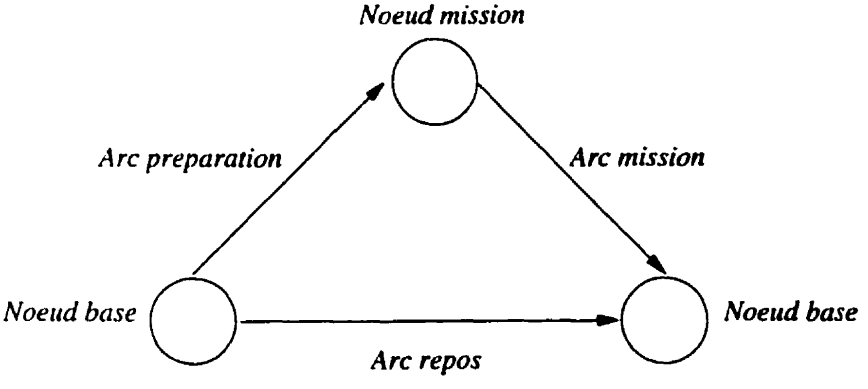

Figure 4.1 – Modélisation d'une mission (1)

Il reste maintenant à construire le réseau, c'est-à-dire à faire le lien entre les missions $m \in M^k$. La figure 4.1 montre que faire le lien entre deux missions consiste à faire le lien entre les noeuds *base* qui leur sont associés. C'est ce que font les algorithmes des sections 4.2, 4.3, 4.4 et 4.5 en plus de préciser dans le détail les ensembles V^k et A^k.

4.2 Algorithme de référence

Nous cherchons à caractériser les deux noeuds *base* associés à une mission m. Pour cela, rappelons que le premier noeud représente le départ de la commodité k dans le but d'effectuer la mission m et le deuxième le retour à la base. À chacun de ces deux évènements, nous faisons correspondre un instant. Cet instant, sur l'horizon temporel de l'instance du problème, correspond pour le premier noeud *base* au temps au plus tard auquel la commodité k peut effectuer la mission et le deuxième, le temps au plus tôt auquel la commodité k peut revenir. Cette manière de procéder assure de conserver "l'ordre" des missions. Nous notons A_1 l'algorithme développé dans cette section.

Il reste donc à déterminer ces deux instants. Nous détaillons maintenant les deux cas qui se présentent. Soit $m \in M^k$, $[a_m, b_m]$ sa fenêtre de temps et d_m sa durée.

- **Premier cas** : $b_m - a_m < d_m$, autrement dit, $m \in M^k \cap M^l$. Alors, les deux instants que nous retenons sont :

  1. b_m.
  2. $a_m + d_m$.

  La figure 4.2 présente alors la modélisation complète de la mission m.

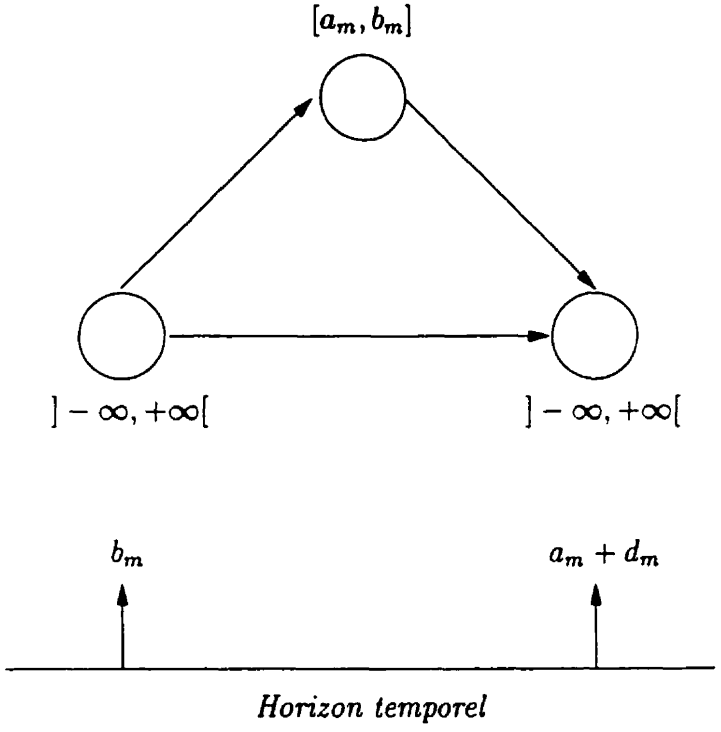

Figure 4.2 - Modélisation d'une mission (2)

- **Deuxième cas** : $b_m - a_m \geq d_m$, c'est-à-dire $m \in M^k \cap M^c$. Dans ce cas, la commodité k peut accomplir la mission m et revenir à la base à un instant appartenant à l'intervalle $[a_m, b_m]$. Les missions de ce type engendrent donc des cycles. Dans un souci d'efficacité algorithmique, nous désirons que le graphe soit acyclique. Pour y parvenir, nous utilisons l'astuce suivante. Nous réalisons une partition de la fenêtre de temps de la mission m de telle sorte que chaque intervalle I_m^i de cette partition ait une longueur strictement inférieure à d_m. Nous nous retrouvons alors pour chacun des intervalles dans le premier cas. Attention, les cycles existent toujours (comme le montre bien d'ailleurs la figure 4.3), ils ont juste été éliminés de la représentation physique des réseaux. Nous proposons la partition suivante.

  Posons $I_m^i = [a_m + i * d_m, \min\{b_m, a_m + (i+1) * d_m - 1\}]$, alors :

$$[a_m, b_m] = \bigcup_{i=0}^{\infty} (I_m^i \cap [a_m, b_m]).$$

La figure 4.3 présente alors la modélisation complète de la mission m.

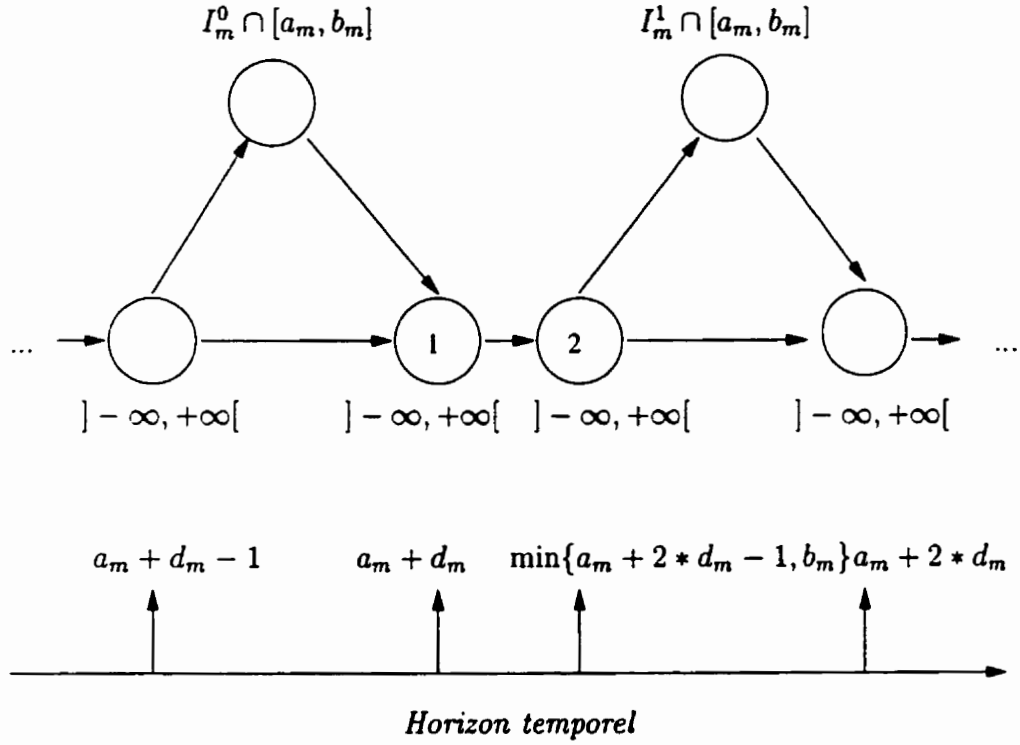

Figure 4.3 – Modélisation d'une mission (3)

Donnons un exemple de cette représentation. Considérons la mission $m \in M^k \cap M^c$ décrite dans le tableau 4.5.

Tableau 4.5 – Exemple d'une mission $m \in M^k \cap M^c$

Fenêtre de temps	Durée
[0,100]	50

Alors, la partition proposée donne trois intervalles à considérer pour la mission m :

1. $[0, 49]$
2. $[50, 99]$
3. $[100, 100]$.

Remarquons qu'il aurait été possible de considérer d'autres façons de partitionner l'intervalle $[a_m, b_m]$, mais celle qui a été choisie minimise le nombre de noeuds *base*. En effet, nous aurions pu partitionner la fenêtre de temps de m, soit l'intervalle $[0, 100]$, de la manière suivante :

1. $[0, 29]$
2. $[30, 59]$
3. $[60, 89]$
4. $[90, 100]$.

À chaque mission ont été associés deux noeuds *base* qui correspondent chacun à un événement : départ de la base et retour à la base. Ces événements ont été caractérisés par un instant. Il reste donc à ordonner ces instants (en agrégeant les noeuds possédant le même instant au besoin) et à les relier par des arcs *repos*. Le dernier noeud *base* est relié au noeud *puits* et le premier au noeud *source*.

En résumé, l'algorithme A_1 que nous proposons procède à une discrétisation de l'horizon temporel. L'ordre de préséance entre missions est respecté. Le réseau ainsi constitué est acyclique. Le nombre de noeuds se trouve être dans le pire cas, $3\times \mid M^k \cap M^l \mid +3 \times \chi \times \mid M^k \cap M^c \mid +2$, où χ est la borne supérieure du nombre d'intervalles à considérer pour une mission $m \in M^k \cap M^c$ (et il faut tenir en plus compte du noeud puits et du noeud source). Le nombre d'arcs, quant à lui, se trouve être, toujours dans le pire cas, $4\times \mid M^k \cap M^l \mid +4 \times \chi \times \mid M^k \cap M^c \mid +1$.

4.3 Algorithme d'agrégation

Dans la section précédente, nous avons considéré toutes les missions de manière indépendante. Chacune d'entre elles génère au moins deux noeuds *base*. Une fois tous les noeuds *base* créés, nous les ordonnons et nous les relions entre eux. La question que pose cette section est la suivante : ces noeuds sont-ils tous indispensables ? Autrement dit, est-il possible d'arriver à éliminer des noeuds *base* tout en s'assurant que, d'une part, aucun chemin n'est supprimé ni ajouté et que, d'autre part, le caractère acyclique est conservé ? L'algorithme A_2 présenté dans cette section vise à réduire le nombre de noeud de type *base*. Nous agissons ainsi directement sur la taille du réseau, ce qui permet d'accélérer le temps de résolution des sous-problèmes. Nous notons A_2 l'algorithme que nous développons dans cette section.

Nous détaillons maintenant un petit exemple qui va donner une idée de l'algorithme développé et qui sera précisé plus tard dans la section. Considérons trois missions m_1, m_2 et m_3 dont nous donnons les caractéristiques dans le tableau 4.6.

Tableau 4.6 – Missions considérées dans le cadre de l'exemple

Mission	Fenêtre de temps	durée
m_1	[0,2]	5
m_2	[1,3]	5
m_3	[4,5]	3

D'après la section 4.2, les instants que nous retenons sur l'horizon temporel sont :

- Pour la mission m_1, 2 et 5.
- Pour la mission m_1, 3 et 6.
- Pour la mission m_1, 5 et 7.

La figure 4.4 donne une représentation de la situation.

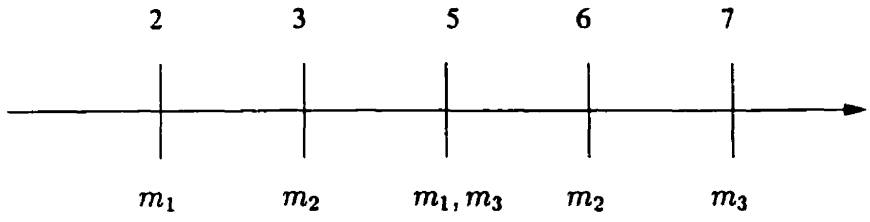

Figure 4.4 – Illustration de l'exemple (1)

Nous voyons facilement sur la figure 4.4 qu'il est possible d'agréger le noeud associé à l'instant 2 et le noeud associé à l'instant 3. Par contre, il n'est pas possible de leur agréger le noeud associé à l'instant 5, parce qu'il correspond notamment au noeud retour à la base de la mission m_1. Or, nous avons précisé au début de la section que nous voulions conserver le caractère acyclique du réseau. En continuant le raisonnement de manière analogue, nous obtiendrions la situation détaillée dans la figure 4.5.

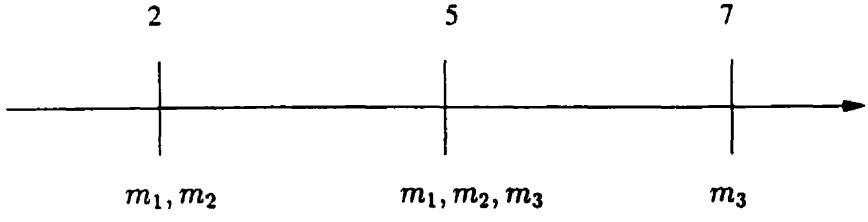

Figure 4.5 – Illustration de l'exemple (2)

Tout chemin dans le graphe de la figure 4.4 se retrouve dans le graphe de la figure 4.5. La réciproque est valable à cause des fenêtres de temps. Le graphe de la figure 4.5 donne la possibilité d'effectuer m_3 après m_2, ce qui n'est pas le cas du graphe de la figure 4.4, mais ce chemin n'est pas réalisable.

Nous sommes donc capables de réduire le nombre de noeuds de 5 à 3. L'idée de l'algorithme est alors assez claire. Les noeuds étant ordonnés, (suivant l'instant qui leur est associé), il s'agit d'opérer une agrégation tant qu'aucun cycle n'apparaît. Lorsque nous cherchons à agréger un noeud avec un ensemble de noeuds, il faut vérifier qu'aucune des missions associées à ce noeud n'est déjà présente dans l'ensemble des missions associées à cet ensemble de noeuds. Remarquons que dans le cas d'une mission $m \in M^k \cap M^c$, nous pouvons considérer qu'il s'agit d'une mission différente pour tous les intervalles de la partition (cela permet d'agréger plus de noeuds en théorie) et c'est ce que nous supposons pour les algorithmes 4.1, 4.2 et 4.4.

L'algorithme A_2 proposé maintenant est une mise en oeuvre possible des idées que nous venons de développer. Il se compose de trois parties. Dans la première (c'est-à-dire l'algorithme 4.1), essentiellement, sont différenciées à un noeud donné les missions pour lesquelles ce noeud est un noeud correspondant à un retour de mission et celles pour lesquelles ce noeud est un noeud correspondant à un début de mission (c'est le rôle des paramètres α et β où $\alpha > \beta > 0$, α et β entiers). Un doublet d est constitué de deux entiers : le premier caractérise la mission associé à ce doublet, le deuxième est un score qui permet de classer l'ensemble des doublets. Puis, dans la deuxième (c'est-à-dire l'algorithme 4.2), nous procédons à l'agrégation en tant que tel. Dans cet algorithme, L représente l'ensemble de tous les noeuds *base* considérés, L_{retenu} l'ensemble des noeuds *base* retenus et L_{temp} l'ensemble des noeuds éliminés à chaque itération de la boucle principale. Enfin, dans la dernière (c'est-à-dire l'algorithme 4.3), sont générés les noeuds *base* et les arcs *preparation* et *mission*.

Reprenons notre petit exemple avec $\alpha = 10$ et $\beta = 5$. La liste des doublets générés par l'algorithme 4.1 est la suivante :

$$(m_1, 25), (m_2, 35), (m_1, 50), (m_3, 55), (m_2, 60), (m_3, 70).$$

L'application de l'algorithme 4.2 redonne le résultat présenté dans la figure 4.5.

Si maintenant nous prenions $\alpha = 1$ et $\beta = 0$ (c'est-à-dire que nous ne faisions pas la différence entre un instant correspondant à un début de mission et un instant correspondant à une fin de mission) et que nous considérions la liste des doublets ordonnée de la manière suivante :

$$(m_1, 2), (m_2, 3), (m_3, 5), (m_1, 5), (m_2, 6), (m_3, 7),$$

l'application de l'algorithme 4.2 donnerait le résultat présenté dans la figure 4.6.

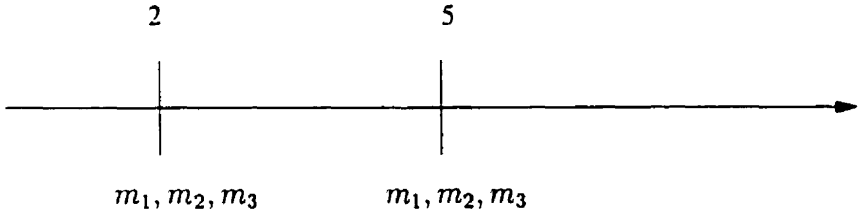

Figure 4.6 - Illustration de l'algorithme

Cette représentation n'est évidemment pas correcte dans la mesure où la mission m_1 et la mission m_3 sont associées aux mêmes noeuds, ce qui signifie qu'il n'est pas possible d'effectuer la mission m_3 après la mission m_1. Il faut respecter impérativement "l'ordre" des missions. Les instants correspondant à une arrivée ou à un départ ne sont pas strictement identiques et c'est ce que traduit l'algorithme 4.1.

Dans les algorithmes présentés maintenant, nous introduisons un certain nombre de notations : $d.x$ désigne la première valeur du doublet d, $d.y$ la deuxième valeur du doublet d et $L.x$ l'ensemble des valeurs $d.x$ où d appartient à L. De plus, à chaque mission $m \in M^k$ est associé un indice, noté $indice(m)$, qui caractérise de manière univoque la mission m. Dans le cas d'une mission $m \in M^k \cap M^c$, chaque copie de

la mission initiale possède un indice différent, ce qui différencie les copies entre elles. Cela se justifie sur la figure 4.3 où nous pouvons agréger les noeuds notés 1 et 2.

Algorithme 4.1 Détermination des doublets pour une mission $m \in M^k$

Soit $m \in M^k$

si $m \in M^k \cap M^l$ **alors**

$\Rightarrow$ Créer deux doublets d_1 et d_2

$d_1 = (indice(m), \quad (a_m + d_m) * \alpha)$

$d_2 = (indice(m), \quad b_m * \alpha + \beta)$

sinon

Poser $J = [a, b] = [a_m, a_m + d_m - 1]$

tant que J non vide **effectuer**

$\Rightarrow$ Créer deux doublets d_1^i et d_2^i

$d_1^i = (indice(m), \quad (a + d_m) * \alpha)$

$d_2^i = (indice(m), \quad b * \alpha + \beta)$

$indice(m) = indice(m) + 1$

$J \Leftarrow [a_m, b_m] \cap [b + 1, b + d_m]$

Algorithme 4.2 Détermination des noeuds *base*

Soit L une liste de doublets d

Soit L_{retenu} une liste d'instants sur l'horizon temporel

pour tout $m \in M^k$ **effectuer**

Appliquer l'algorithme 4.1 à m

Ajouter les doublets à L

Trier la liste L suivant les scores croissants

tant que L non vide **effectuer**

Soit d le premier élément de L, rajouter $[d.y/\alpha]$ à L_{retenu}

Enlever d à L

Soit L_{temp} une liste vide de doublet

Considérer le premier élément d_{temp} de L

tant que ($d_{temp}.x \neq d.x$) et ($d_{temp}.x \notin L_{temp}.x$) **effectuer**

Ajouter d_{temp} à L_{temp}

Enlever d_{temp} à L

Considérer le premier élément d_{temp} de L

Algorithme 4.3 Construction des arcs *preparation* et *mission*

pour tout $m \in M^k$ **effectuer**

si $m \in M^k \cap M^l$ **alors**

Chercher le dernier instant inférieur ou égal à $a_m + d_m$

Chercher le dernier instant inférieur ou égal à b_m

sinon

Poser $J = [a, b] = [a_m, a_m + d_m - 1]$

tant que J non vide **effectuer**

Chercher le dernier instant inférieur ou égal à $a + d_m$

Chercher le dernier instant inférieur ou égal à b

$J \Leftarrow [a_m, b_m] \cap [b + 1, b + d_m]$

Construire les arcs *mission* et *preparation* associés à la mission m à partir de ces deux instants.

Nous prouvons maintenant que le graphe obtenu après l'application de la procédure d'agrégation est acyclique.

Proposition 4.1 : Les algorithmes 4.1, 4.2 et 4.3 produisent un graphe acyclique.

Preuve : Raisonnons par l'absurde. Supposons que la mission $m \in M^l$ produise un cycle (rappelons que pour une mission $m \in M^c$, nous avons considéré qu'il s'agissait d'une mission différente pour tous les intervalles de la partition : en conséquence, toutes les copies sont des éléments de M^l), c'est-à-dire que le dernier instant inférieur ou égal à $a_m + d_m$ est identique au dernier instant inférieur ou égal à b_m. La figure 4.7 résume la situation.

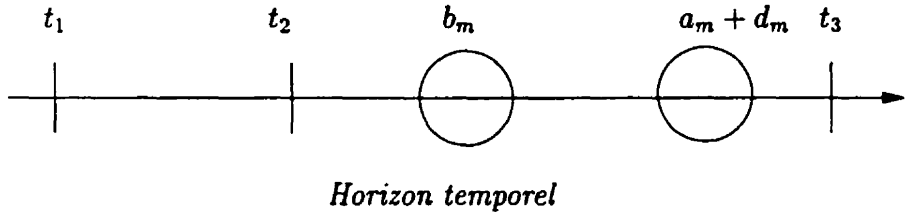

Figure 4.7 – Situation dans le cas d'un cycle (1)

Remarquons tout d'abord que $t_2 = b_m$. En effet, l'algorithme 4.2 agrège nécessairement le noeud associé à l'instant t_2 avec le noeud associé à l'instant b_m puisque *indice*(m) n'est pas présent dans la liste d'indices associée à l'instant t_2. La nouvelle situation est décrite dans la figure 4.8.

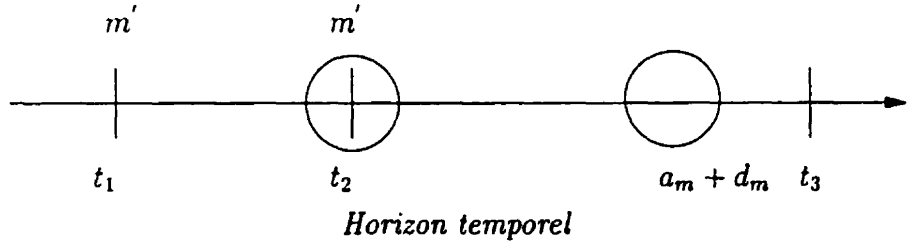

Figure 4.8 – Situation dans le cas d'un cycle (2)

Soit m' une mission dont l'indice est présent à l'instant t_1 et t_2. À l'instant t_2, *indice*(m) est placé dans la liste des doublets construite par l'algorithme 4.1 après *indice*(m') (puisque à cet instant, *indice*(m') correspond à une arrivée et *indice*(m) à un départ), ce qui signifie que dans l'algorithme 4.2, le noeud associé à l'*indice*(m') ne peut être agrégé avec le noeud associé à l'*indice*(m). *indice*(m) appartient donc à l'ensemble de noeuds agrégés suivant celui de *indice*(m'). Cela implique nécessairement que $t_3 \leq a_m + d_m$, d'où la contradiction.

□

Il est assez difficile d'évaluer l'efficacité de notre algorithme (à part d'un point de vue pratique bien sûr). Il s'applique sur l'ensemble des noeuds *base*, soit sur au plus $2\times \mid M^k \cap M^l \mid +2 \times \chi\times \mid M^k \cap M^c \mid +2$ noeuds et, par conséquent, sur au plus $2\times \mid M^k \cap M^l \mid +2 \times \chi\times \mid M^k \cap M^c \mid +1$ arcs.

4.4 Premier algorithme de coupes dans le réseau

Supposons que le réseau G^k associé à la commodité k ait été construit comme indiqué dans la section 4.2. Dans la section 4.3, nous avons modifié sa représentation physique en agrégeant certains noeuds. Dans cette section, nous nous proposons de modifier sa représentation polyédrale en montrant qu'il est possible d'éliminer *a priori* un sous-ensemble des chemins admissibles à ce réseau. Pour ce faire, nous allons d'abord mettre en évidence certaines propriétés du PFH compte tenu de la modélisation. Nous notons A_3 l'algorithme développé dans cette section.

Soit M l'ensemble des missions, notons M^u l'ensemble de toutes les missions m possédant les deux propriétés suivantes :

1. La fenêtre de temps de la mission m est réduite à un instant.
2. Il existe une requête obligatoire r couverte par la mission m qui n'est couverte par aucune autre mission $m' \in M$.

Il est très fréquent pour les instances des problèmes traités que $M^u \neq \emptyset$. Les missions $m \in M^u$ correspondent par exemple à une demande de mise en disponibilité d'un avion pour une activité quelconque (e.g., exercices de parachutage). En général, de telles missions sont des missions prédéfinies obligatoires. La particularité des missions $m \in M^u$ est qu'elles appartiennent nécessairement à toute solution entière réalisable.

Soit R^u l'ensemble des requêtes couvertes par les missions de $m \in M^u$.

Notons $M^{u'}$ l'ensemble des missions m possédant la propriété suivante :

1. m couvre au moins une requête de R^u.

Remarquons que, en général, $M^u \subset M^{u'}$. En effet, il se peut qu'une mission m couvre deux requêtes r_1 et r_2, r_1 n'étant couverte que par m et r_2 étant couverte par d'autres missions. Nous pouvons éliminer de l'ensemble des missions que nous considérons initialement toutes les missions $m \in M^{u'} \setminus M^u$. En pratique, cette réduction n'est pas significative.

Nous travaillons désormais sur l'ensemble de missions $M^u \cup (M \setminus M^{u'})$.

Pour la suite de cette section, nous raisonnons sur les ensembles de commodités identiques que nous notons i. Deux commodités sont identiques si elles sont de même type et si elles sont associées à la même base. Soit $i \in I$ un ensemble de commodités identiques, toutes les commodités $k \in i$ possèdent le même ensemble M^k. Notons M^i l'ensemble des missions associées à i, alors :

$$M^i = M^k, \forall k \in i.$$

Notons M_i^u l'ensemble des missions de M^u associées à i, alors :

$$M_i^u = M^i \cap M^u$$

et

$$M^u = \bigcup_{i \in I} M_i^u.$$

Venons en maintenant à l'idée de l'algorithme A_3. Considérons un élément $m \in M_i^u$, nous savons que cette mission doit nécessairement apparaître dans toute solution

entière réalisable. De plus, cette mission a une fenêtre de temps réduite à un point. Nous pouvons donc affirmer que l'intervalle $[a_m, a_m + d_m]$ (rappelons que $a_m = b_m$) est bloqué pour un élément de $k \in i$. Nous déterminons alors l'ensemble de tous les intervalles bloqués et construisons ce que nous appelons l'horizon de disponibilité pour tous les ensembles de commodités identiques. L'horizon de disponibilité permet de savoir à tout instant sur l'horizon temporel considéré le nombre de commodités utilisées pour accomplir une mission $m \in M_i^u$.

Soit $i \in I$, notons n_t^i, le nombre de commodités $k \in i$ nécessaires à l'instant t, alors trois cas se présentent :

1. $n_t^i > \mid i \mid$: le problème est non réalisable.
2. $n_t^i = \mid i \mid$: nous dirons que l'instant t est bloqué. En ce sens que les commodités $k \in i$ n'ont d'autre choix que de couvrir les missions $m \in M_i^u$ telles que $t \in [a_m, a_m + d_m]$.
3. $n_t^i < \mid i \mid$: $(\mid i \mid - n_t^i)$ commodités sont libres pour effectuer d'autres missions.

Nous allons exploiter le deuxième cas. Supposons que nous ayons déterminé l'ensemble B^i des intervalles bloqués pour un ensemble de commodités identiques $i \in I$. Un intervalle est dit bloqué si tous les instants qu'il couvre le sont (au sens de la définition précédente). Cela nous amène à reconsidérer l'ensemble des missions $m \in (M^i \setminus M_i^u) \cap (M^u \cup (M \setminus M^{u'}))$ et à vérifier que leur réalisation n'entre pas en conflit avec un de ces intervalles. Certaines de ces missions vont voir leur fenêtre de départ restreinte tandis que d'autres vont tout simplement être éliminées. Jusqu'à présent, nous nous sommes contentés de réduire la cardinalité de M et modifier les missions m de cet ensemble.

Considérons maintenant un intervalle bloqué $b \in B^i$, la borne inférieure correspond à l'instant associé à un début de mission et la borne supérieure à l'instant associé à une fin de mission. Notre algorithme de référence appliqué au nouvel ensemble de

missions crée donc un noeud *base* pour chacune des bornes de l'intervalle b. Et, nous sommes assurés, compte tenu des modifications que nous avons opérées sur les missions à considérer pour la construction du réseau associé à l'ensemble de commodités i qu'il n'existe aucun noeud *base* dont l'instant est compris dans l'intervalle b. L'intervalle b étant bloqué, nous omettons l'arc *repos* entre ces deux noeuds *base*. Par cette procédure, nous éliminons un sous-ensemble de l'ensemble des chemins réalisables des réseaux G^k.

Dans ce qui suit, nous voyons le détail algorithmique des considérations que nous venons de développer. Avec l'algorithme 4.4, nous construisons l'horizon de disponibilité de l'ensemble des commodités i. Un doublet d est constitué de deux entiers, le premier représente un instant sur l'horizon de temps et le deuxième le nombre de commodités utilisées à cet instant et jusqu'à l'instant suivant dans la liste de doublets.

Algorithme 4.4 Détermination d'un horizon de disponibilité (1)

Soit L une liste de doublets ordonnés suivant la composante x

Soient $d_1 = (-\infty, 0)$ et $d_2 = (+\infty, 0)$

Ajouter d_1 et d_2 à L

pour tout $m \in M_i^u$ **effectuer**

Considérer a_m (Rappel : $a_m = b_m$)

Soit l'intervalle au plus tard $J_1 = [t_1, t_2]$ tel que : $a_m \in J_1$

Soit l'intervalle au plus tôt $J_2 = [t_1', t_2']$ tel que : $a_m + d_m \in J_2$

avec $t_1', t_2', t_1, t_2 \in L.x$

pour tout $d \in L$ et $d.x \in [t_2, t_1']$ **effectuer**

Incrémenter $d.y$

si $a_m = t_1$ **alors**

Incrémenter $d.y$ où d est tel que $d.x = t_1$

sinon

Créer le doublet $d = (a_m, d'.y + 1)$ où d' est tel que $d'.x = t_1$

si $a_m + d_m \neq t_2'$ **alors**

Créer le doublet $d = (b_m, d'.y - 1)$ où d' est tel que $d'.x = t_1'$

Les algorithmes 4.5 et 4.6 détaillent la manière dont nous traitons les missions compte tenu du fait que nous connaissons les horizons de disponibilité.

Algorithme 4.5 Réduction de la fenêtre de temps d'une mission

Soit $b = [t_1, t_2] \in B_i$ et $m \in (M \setminus M^{u'}) \cap M^i$

$[a_m, b_m] \leftarrow [a_m, b_m] \setminus ([a_m, b_m] \cap b)$

Soit $J = [bi, bs] = [a_m + d_m, b_m + d_m] \setminus ([a_m + d_m, b_m + d_m] \cap b)$

$[a_m, b_m] \leftarrow [bi - d_m, bs - d_m] \cap [a_m, b_m]$

si $[a_m, b_m] = \emptyset$ **alors**

La mission m est rejetée

si $t_1 \geq b_m$ et $t_2 \leq a_m + d_m$ **alors**

La mission m est rejetée

Dans l'algorithme 4.5, il y a deux conditions pour rejeter des missions : ou bien la fenêtre de la mission traitée devient vide après le traitement, ou bien la mission débute avant le début de l'intervalle bloqué et s'achève après. Certains cas posent problèmes, nous les détaillons dans les figures 4.9 et 4.10.

1. Premier cas :

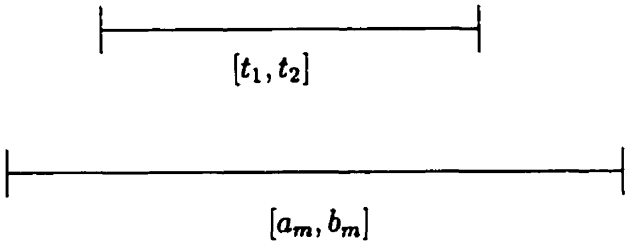

Figure 4.9 – Premier cas particulier

Il faut, dans le cas décrit à la figure 4.9 créer deux missions, la première avec une fenêtre de temps $[a_m, t_1]$ et la deuxième avec une fenêtre de temps $[t_2, b_m]$, les inclure dans $(M \setminus M^{u'}) \cap M^i$ et leur réappliquer l'algorithme 4.5.

2. Deuxième cas :

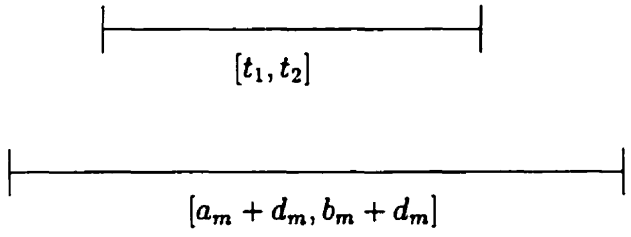

Figure 4.10 – Deuxième cas particulier

Il faut, dans le cas décrit à la figure 4.10 créer deux missions, la première avec une fenêtre de temps $[a_m, t_1 - d_m]$ et la deuxième avec une fenêtre de temps $[t_2 - d_m, b_m]$, les inclure dans $(M \setminus M^{u'}) \cap M^i$ et leur réappliquer l'algorithme 4.5.

Nous déterminons le nouvel ensemble $(M \setminus M^{u'}) \cap M^i$ avec l'algorithme 4.6.

Algorithme 4.6 Détermination du nouvel ensemble $(M \setminus M^{u'}) \cap M^i$

pour tout $m \in (M \setminus M'_u) \cap M^i$ **effectuer**

pour tout $b \in B_i$ **effectuer**

Appliquer l'algorithme 4.5 à m et b et rajouter des missions si il y a lieu.

Enfin, il est possible de procéder à une agrégation des noeuds *base* en s'inspirant de ce qui a été fait dans la section 4.3.

4.5 Deuxième algorithme de coupes dans le réseau

Dans cette section, nous montrons comment nous pouvons envisager une décomposition suivant l'horizon temporel (en plus de la décomposition suivant les commodités qui est déjà réalisée) en utilisant les horizons de disponibilité que nous avons construits dans la section 4.4. Nous notons A_4 l'algorithme développé dans cette section.

Pour la suite de l'exposé de cette section, nous introduisons l'ensemble d'intervalles $H^i, i \in I$:

$$H^i =]-\infty, \infty[\setminus (\bigcup_{b \in B^i} b).$$

H^i correspond à l'ensemble des intervalles pour lesquels aucun instant n'est bloqué.

Soit maintenant $i \in I$ un ensemble de commodités identiques, nous avons déterminé un ensemble d'intervalles bloqués B^i. Considérons un intervalle bloqué $b \in B^i$, soit $h_1 \in H^i$ l'intervalle dont la borne supérieure coïncide avec la borne inférieure de b et $h_2 \in H^i$ l'intervalle dont la borne inférieure coïncide avec la borne supérieure de b. Toute commodité $k \in i$ réalise un sous-chemin sur l'intervalle de temps h_1, accomplit ensuite une mission m qui contribue au blocage de l'intervalle b, et enfin réalise un sous-chemin sur l'intervalle de temps h_2. Cela suggère donc de créer un réseau pour chaque intervalle $h \in H^i$. Il s'agit en plus d'assurer que nous pouvons reconstituer le chemin emprunté par une commodité, cela oblige à imposer des conditions initiales et finales pour ces réseaux. Ces conditions traduisent le fait qu'une commodité pénètre dans un intervalle de temps $h \in H^i$ en ayant accompli une mission $m \in M^i$ et sort de cet intervalle en accomplissant une autre mission $m \in M^i$.

Voyons maintenant le détail algorithmique. L'algorithme 4.7 est une extension de l'algorithme 4.4. Il construit les horizons de disponibilité et détermine en même temps les missions qui contribuent au blocage des intervalles $b \in B^i, \forall i \in I$.

Pour chaque intervalle $h \in H^i$, l'algorithme 4.8 détaille la manière avec laquelle nous construisons le réseau associé. Auparavant, nous avons besoin de définir le concept de missions de *liaison*. Une mission est une mission de liaison si elle participe à la constitution d'un intervalle de blocage. Pour trouver les missions de liaison associées à un type de commodité i, il suffit de reconsidérer l'algorithme 4.4 et de travailler avec des triplets au lieu de doublets. Le troisième champ correspond à la liste des missions effectuées à l'instant considéré.

Pour chaque type de commodité i, l'ensemble des missions de liaison est noté $M^i_{liaison}$. Pour chaque intervalle $h \in H^i$, les ensembles de missions de liaison qui couvrent les deux intervalles bloqués encadrant h sont notés $M^{i,b}_{liaison,h}$ et $M^{i,f}_{liaison,h}$. Remarquons plusieurs choses :

- Premièrement, $\mid M^{i,f}_{liaison,h} \mid = \mid M^{i,b}_{liaison,h} \mid = \mid i \mid$, sauf dans le cas où l'on considère le premier intervalle de l'ensemble B^i (auquel cas $\mid M^{i,b}_{liaison,h} \mid = 0$) ou le dernier intervalle (auquel cas $\mid M^{i,f}_{liaison,h} \mid = 0$).
- Soient h^1 et h^2 deux intervalles consécutifs de H^i, alors $M^{i,f}_{liaison,h^1} = M^{i,b}_{liaison,h^2}$.
- Possiblement $M^{i,b}_{liaison,h} \cap M^{i,f}_{liaison,h} \neq \emptyset$, ce qui signifie qu'une mission de liaison couvre en fait l'intervalle h.

Voyons maintenant comment est construit un réseau associé à un intervalle $h \in H^i$. Tout d'abord, nous cherchons l'ensemble des missions de $(M \backslash M^{u'}) \cap M^i$ dont la fenêtre de temps est incluse dans h. Nous construisons alors avec les missions ainsi retenues l'horizon temporel comme indiqué dans la section 4.2. Il reste alors à incorporer au réseau les missions de liaison appartenant à l'ensemble $M^{i,b}_{liaison,h}$ et à l'ensemble $M^{i,f}_{liaison,h}$. L'algorithme 4.8 détaille la manière dont nous procédons. Précisons que dans ce réseau, les seuls arcs qui rentrent au puits ou qui sortent de la source sont les arcs de mission de liaison. Autrement dit, il n'y a pas d'arc repos entre la source et le premier noeud base ni d'arc repos entre le dernier noeud base et le puits, sauf dans le cas où l'on considère le premier intervalle de l'ensemble H^i (auquel cas il y a un arc

Algorithme 4.7 Détermination d'un horizon de disponibilité (2)

Soit L une liste de triplets ordonnés suivant la composante x

Soient $d_1 = (-\infty, 0, NULL)$ et $d_2 = (+\infty, 0, NULL)$

Ajouter d_1 et d_2 à L

pour tout $m \in M_i^u$ **effectuer**

Considérer a_m (Rappel : $a_m = b_m$)

Soit l'intervalle au plus tard $J_1 = [t_1, t_2]$ tel que : $a_m \in J_1$

Soit l'intervalle au plus tôt $J_2 = [t_1', t_2']$ tel que : $a_m + d_m \in J_2$

avec $t_1', t_2', t_1, t_2 \in L.x$

pour tout $d \in L$ et $d.x \in [t_2, t_1']$ **effectuer**

Incrémenter $d.y$

$d.z = d.z + m$

si $a_m = t_1$ **alors**

Incrémenter $d.y$ où d est tel que $d.x = t_1$

Ajouter m à la liste $d.z$

sinon

Créer le triplet $d = (a_m, d'.y, d'.z \cup \{m\})$ où d' est tel que $d'.x = t_1$

si $a_m + d_m \neq t_2'$ **alors**

Créer le triplet $d = (b_m, d'.y - 1, d'.z \setminus \{m\})$ où d' est tel que $d'.x = t_1'$

repos entre le premier noeud base et la source) et le dernier intervalle de l'ensemble H^i (auquel cas il y a un arc repos entre le dernier noeud base et le puits).

Pour satisfaire les conditions initiales ou finales du réseau, à chacune des missions des ensembles $M^{i,b}_{liaison,h}$ et $M^{i,f}_{liaison,h}$ est associée une tâche à effectuer, ce qui revient à rajouter une contrainte de partitionnement dans le problème maître. À chaque mission contribuant à la constitution d'un intervalle de blocage correspond autant de tâches différentes que le nombre de fois où cette mission apparaît dans les ensembles $M^{i,b}_{liaison,h}$ et $M^{i,f}_{liaison,h}$, $\forall h \in H^i$.
La procédure que nous venons de développer revient à reconsidérer la décomposition initiale. Nous ne réalisons plus une décomposition suivant les types de commodité, mais suivant les types de commodité et suivant l'horizon de temps. Cela a pour conséquence :

1. Le nombre de réseaux augmente considérablement, mais la taille totale des réseaux (c'est-à-dire le nombre d'arcs et de noeuds de l'ensemble des réseaux) devient à peine supérieure à celle des réseaux engendrés par l'algorithme de référence. Pour une itération donnée, le temps de résolution du sous-problème sera donc moins élevé car nous considérons des réseaux beaucoup plus petits.
2. La taille du problème maître augmente en nombre de rangées, mais le nombre total de chemins dans les réseaux est beaucoup moins élevé.

Remarquons pour finir qu'il est possible d'envisager une agrégation des réseaux de la manière présentée à la section 4.3.

Algorithme 4.8 Construction du réseau associé à h

Déterminer les missions $m \in (M \setminus M^{u'}) \cap M^i$ telles que $[a_m, b_m] \subset h$ (nécessairement $[a_m + d_m, b_m + d_m] \subset h$, d'après l'algorithme 4.6).

Construire l'horizon temporel comme indiqué à la section 4.2.

pour tout $m \in M^{i,b}_{liaison,h}$ **effectuer**

si $m \notin M^{i,f}_{liaison,h}$ **alors**

Rajouter l'instant $a_m + d_m$ à l'horizon temporel.

Créer un noeud base associé à $a_m + d_m$ avec la fenêtre $[a_m + d_m, a_m + d_m]$.

Créer un arc mission entre la source et le noeud base créé.

pour tout $m \in M^{i,f}_{liaison,h}$ **effectuer**

si $m \notin M^{i,b}_{liaison,h}$ **alors**

Rajouter l'instant a_m à l'horizon temporel.

Créer un noeud base associé à a_m avec la fenêtre $[a_m, a_m]$.

Créer un arc mission entre le noeud base créé et le puits.

sinon

Créer un arc mission entre la source et le puits.

Terminer la construction du réseau comme indiqué à la section 4.2

4.6 Conclusion

De ce chapitre, nous pouvons dégager deux idées essentielles. Tout d'abord, le principe de construction des réseaux G_k est relativement simple (section 4.2). Ensuite, il est possible de modifier ces réseaux de référence et ce, de deux manières différentes (que nous pouvons conjuguer) :

- En réalisant une agrégation des noeuds *base*, ce qui ne modifie pas l'ensemble Ω^k (section 4.3), mais ce qui permet d'accélérer le temps de résolution des sous-problèmes.
- En montrant que nous pouvons éliminer un certain nombre d'arcs *repos* (section 4.4 et section 4.5), ce qui diminue la cardinalité de Ω^k.

CHAPITRE 5

Résultats numériques

Ce chapitre présente les résultats numériques obtenus sur quelques instances du PFH. Il vise à mettre en évidence la validité du modèle développé au chapitre 3. Dans la section 5.1, nous expliquons la méthode de résolution que nous avons retenue. Puis, dans la section 5.2, nous détaillons les différents scénarii utilisés dans le cadre de nos tests. Les sections 5.3 et 5.4 présentent les résultats obtenus en appliquant la méthode de résolution sur les scénarii. Plus précisément, la section 5.3 s'intéresse à la relaxation linéaire et montre l'importance du choix de l'algorithme de construction de réseaux, tandis que dans la section 5.4, le PFH est résolu au complet. Enfin, dans la section 5.5, nous faisons une synthèse de ce chapitre.

5.1 Méthode de résolution

Pour résoudre la formulation unifiée, de laquelle nous avons dérivé le modèle (3.1)-(3.11), Desaulniers *et al.* [5] proposent un algorithme de séparation et évaluation progressive (communément appelé Branch-and-Bound). Nous reprenons dans ce mémoire cette approche de résolution.

5.1.1 Évaluation des bornes inférieures

À un noeud de l'arbre de branchement, une borne inférieure est obtenue en résolvant une relaxation de la formulation (3.13)-(3.19) à laquelle on a appliqué les décisions de branchement prises à chacun des ancêtres du noeud considéré. La relaxation choisie est la relaxation linéaire qui consiste à relaxer les contraintes d'intégrité (3.16), (3.17) et (3.19). Le programme mathématique à résoudre est alors le suivant :

$$\min \sum_{k \in K} \sum_{p \in \Omega^k} c_p^k \theta_p^k + \sum_{o \in O} c_o Z_o \tag{5.1}$$

sujet à :

$$\sum_{k \in K} \sum_{p \in \Omega^k} a_p^{kt} \theta_p^k + \sum_{o \in O} b_o^t Z_o = 1, \quad \forall t \in T \tag{5.2}$$

$$\sum_{o \in O} Z_o - Y = 0, \tag{5.3}$$

$$0 \leq Z_o \leq 1, \quad \forall o \in O \tag{5.4}$$

$$\sum_{p \in \Omega^k} \theta_p^k = 1, \quad \forall k \in K \tag{5.5}$$

$$0 \leq \theta_p^k \leq 1, \quad \forall k \in K, \forall p \in \Omega^k. \tag{5.6}$$

Bien entendu, nous devons tenir compte de l'ensemble des décisions de branchement déjà prises. Celles-ci affectent les contraintes du problème maître et les réseaux G_k (donc l'ensemble Ω^k).

Le nombre de chemins réalisables étant généralement beaucoup trop grand pour les énumérer explicitement, le programme (5.1)-(5.6) est résolu par génération de colonnes. Cette méthode est utilisée pour résoudre les programmes linéaires de grande taille, c'est-à-dire possédant un très grand nombre de variables. Dans le cadre de cette

méthode, le programme linéaire à résoudre est appelé problème maître et toute restriction de ce programme ne considérant qu'un sous-ensemble des variables est appelée problème maître restreint. L'algorithme de génération de colonnes est un processus itératif qui résout une suite de problèmes maîtres restreints et de sous-problèmes jusqu'à l'obtention d'une solution optimale pour un problème maître restreint qui soit aussi optimale pour le problème maître. Une itération de l'algorithme se déroule de la façon suivante. Le problème maître restreint est résolu, permettant d'obtenir un ensemble de variables duales optimales pour ce problème. Ces variables sont alors transférées aux sous-problèmes pour leur permettre de prouver l'optimalité du problème maître restreint courant ou de générer au moins une variable de coût réduit négatif. Une variable (ou colonne) du problème maître restreint ayant été générée par un sous-problème est appelée colonne dynamique. Pour le PFH, à chaque réseau G^k correspond un sous-problème qui est le suivant :

$$\min \sum_{(i,j)\in A^k} (c_{ij}^k - \sum_{t\in T} a_{ij}^{kt}\delta_t) X_{ij}^k - \xi_k \tag{5.7}$$

sujet à :

$$\sum_{j:(o(k),j)\in A^k} X_{o(k)j} = 1, \quad \forall k \in K \tag{5.8}$$

$$\sum_{j:(j,d(k))\in A^k} X_{jd(k)} = 1, \quad \forall k \in K \tag{5.9}$$

$$\sum_{j:(i,j)\in A^k} X_{ji}^k - \sum_{j:(i,j)\in A^k} X_{ij}^k = 0, \quad \forall k \in K, \forall i \in V^k \setminus \{o(k), d(k)\} \tag{5.10}$$

$$X_{ij}^k (f_{ij}^k(T_i^k) - T_j^k) \leq 0, \quad \forall k \in K, \forall (i,j) \in A^k \tag{5.11}$$

$$a_i^k \leq T_i^k \leq b_i^k, \quad \forall k \in K, \forall i \in V^k \tag{5.12}$$

$$X_{ij}^k \in \{0,1\}, \quad \forall k \in K, \forall (i,j) \in A^k. \tag{5.13}$$

Dans cette formulation, $\delta_t, t \in T$ et ξ_k sont les variables duales associées aux

contraintes (5.2) et à la contrainte (5.5) spécifique à k respectivement. Chaque sous-problème correspond à un problème de plus court chemin avec contraintes de ressource et peut se résoudre par un algorithme pseudo-polynomial lorsque les fonctions f_{ij}^k sont non-décroissantes (Desaulniers *et al.* [5]).

5.1.2 Les stratégies de résolution

Parmi les difficultés rencontrées par une méthode de génération de colonnes figure la convergence relativement lente du processus de résolution. De nombreuses itérations sont nécessaires pour obtenir une bonne approximation du polyèdre dual, d'autant plus que les problèmes que nous traitons en pratique sont souvent fortement dégénérés. L'article de Desaulniers *et al.* ([6]) présente un état de l'art de l'ensemble des techniques qui ont été développées pour accélérer le processus de résolution d'une méthode de génération de colonnes. En fait, les auteurs se placent plus spécifiquement dans le cadre de la formulation unifiée de Desaulniers *et al.* ([5]) dont le PFH, rappelons-le, est un cas particulier.
Il est assez fréquent, lorsque l'on résout un problème de partitionnement d'ensemble par génération de colonnes, de relaxer la relaxation linéaire de ce problème afin d'accélérer sa résolution. Ainsi, pour le PFH, il est possible de remplacer les équations (5.2) par :

$$\sum_{k \in K} \sum_{p \in \Omega_k} a_p^{kt} \theta_p^k + \sum_{o \in O} b_o^t Z_o \geq 1, \; \forall t \in T. \tag{5.14}$$

Cette relaxation consiste à résoudre le problème comme un modèle de recouvrement d'ensemble et permet de borner inférieurement par 0 les variables duales associées à ces contraintes. Pareillement, il est possible de remplacer les contraintes (5.5) par :

$$\sum_{p \in \Omega^k} \theta_p^k \leq 1, \ \forall k \in K. \tag{5.15}$$

Les variables duales associées à ces contraintes sont alors bornées supérieurement par 0.

De plus, il est possible d'envisager une stratégie de perturbation des contraintes. Celle-ci consiste à introduire des variables d'écart pour les contraintes de partitionnement. Ces variables d'écart ont un coût nul, sont bornées inférieurement par 0 et supérieurement par un nombre suffisamment petit et différent pour chaque contrainte. Les contraintes (5.2) deviennent alors :

$$\sum_{k \in K} \sum_{p \in \Omega_k} a_p^{kt} \theta_p^k + \sum_{o \in O} b_o^t Z_o + y^t = 1, \ \forall t \in T \tag{5.16}$$

ou (si on permet le sur-recouvrement) par :

$$\sum_{k \in K} \sum_{p \in \Omega_k} a_p^{kt} \theta_p^k + \sum_{o \in O} b_o^t Z_o + y^t \geq 1, \ \forall t \in T \tag{5.17}$$

et à introduire les contraintes suivantes supplémentaires dans le problème maître

$$0 \leq y^t \leq \epsilon^t, \ \forall t \in T, \tag{5.18}$$

où ϵ^t est un nombre suffisamment petit.

Dans le cadre de la génération de colonnes, du Merle *et al.* ([10]) proposent un algorithme de stabilisation des variables duales qui combine une procédure de perturbation des contraintes primales (suivant le principe détaillé ci-dessus) et une procédure de pénalisation des variables duales. Nous utilisons l'algorithme de perturbation dans le cadre de nos tests.

Pour finir, soulignons que lorsque de telles techniques de perturbation de contraintes sont employées, la méthode de résolution doit contenir une ou des stratégies de branchement permettant de remettre les contraintes sous leur forme originale.

5.1.3 Stratégies de branchement

Nous détaillons dans cette section les différentes stratégies de branchement que nous avons utilisées pour la résolution complète du PFH. Les méthodes de branchement se classent en deux catégories : les méthodes de branchement optimales et les méthodes de branchement heuristiques. Nous précisons entre parenthèses le nom que nous leur avons donné.

Les stratégies heuristiques :

- La fixation de variables de chemin (*cfix*) : cette stratégie consiste à fixer à 1 une ou plusieurs variables de chemin ayant une valeur fractionnaire au noeud courant supérieure à un seuil donné. Rien ne permet d'établir que le problème reste réalisable. Cependant, nous pouvons nous assurer que chaque contrainte de couverture de tâche est couverte au plus une fois dans l'ensemble des colonnes fixées à 1, ce qui constitue une condition nécessaire de réalisabilité.
- La fixation d'une mission (*mfix*) : cette stratégie consiste à imposer la réalisation d'une mission m dont le flot est supérieur à un seuil donné. Pour appliquer cette décision, il suffit d'enlever de tous les réseaux les arcs couvrant au moins une tâche couverte par m (sauf les arcs représentant m bien entendu). Dans le cas d'une mission $m \in M^c$, nous calculons le flot pour chaque arc couvrant cette mission. Nous retenons l'arc de flot le plus élevé et éliminons les autres, ce qui revient finalement à retenir une fenêtre de temps en particulier.

Les stratégies optimales :

- Branchement sur le nombre de requêtes acceptées : cette décision consiste à resteindre Y à être plus petit ou égal à $\lfloor Y \rfloor$ sur une branche et à être plus

grand ou égal à $\lceil Y \rceil$ sur l'autre branche. Ces deux décisions sont imposées par l'ajout d'une borne sur la variable Y dans le problème maître.

- Acceptation ou rejet de requêtes : cette décision consiste à fixer Z_o à 1 sur une branche (ce qui revient à rejeter toutes les requêtes du groupe de requêtes optionnelles o) et à 0 sur l'autre (ce qui revient à accepter toutes les requêtes du groupe de requêtes optionnelles o).
- Fixation d'une inter-tâche (*itfix*) : cette stratégie consiste à sélectionner deux tâches t_1 et t_2 et de créer deux branches. Dans la première, on impose que t_1 soit immédiatement suivie de t_2. Dans la deuxième, on impose que t_2 ne puisse pas être effectuée immédiatement après t_1. Une variante de cette stratégie consiste à choisir une seule tâche t et à imposer sur une branche que cette tâche soit la première (respectivement la dernière) d'une ligne d'affectation et sur l'autre branche qu'elle soit précédée (respectivement suivie) d'une autre tâche.
- Enlèvement de la perturbation des contraintes : cette stratégie consiste à remettre les contraintes de couverture des requêtes sous leur forme originale (contraintes (5.2)).
- Choix d'une commodité pour couvrir une requête (*afix*) : soit t une requête couverte par au moins deux commodités, soit k une de ces commodités, alors nous créons deux branches. La première décision interdit que la requête t soit couverte par k. La deuxième décision impose que la requête t soit couverte par k. Pour appliquer ces décisions, il suffit d'éliminer les arcs et les noeuds qui couvrent t dans les réseaux correspondants.

L'ordre des priorités entre les différentes méthodes de branchement est présenté dans la section 5.4.

5.2 Les différents scénarii

Les scénarii ont été construits à partir du PVM du mois d'août 96 pour les aéronefs de type CC130. Ceux-ci sont des avions-cargo pour le transport de fret et de marchandises. Nous ne considérons qu'un seul type d'avion, ce qui fait que les ensembles de commodités identiques se confondent avec les ensembles de commodités associées aux bases. Nous disposons de treize lignes d'affectation, soit neuf lignes pour la base de Trenton, trois pour celle de Winnipeg et une pour celle de Greenwood. Nous prenons en compte toutes les requêtes du PVM du mois d'août 1996. Nous sommes assurés compte tenu de l'existence de ce PVM de l'existence d'une solution entière réalisable, à condition, bien entendu, d'avoir généré au préalable toutes les missions à l'aide du générateur.

Nous proposons les trois scénarii suivants :

Scénario 1 : Ce scénario considère toutes les requêtes du PVM de août 1996. Toutes les requêtes sont obligatoires.

Scénario 2 : Nous reprenons le scénario 1, mais nous restreignons l'horizon à l'intervalle de temps du 1 au 15 août.

Scénario 3 : Nous reprenons le scénario 1, mais nous restreignons l'horizon à l'intervalle de temps du 15 au 31 août.

Les scénarii 2 et 3 ont pour but de mettre en évidence l'impact de la longueur de l'horizon d'optimisation sur les temps de résolution.

Dans le jeu de données considérées dans ce chapitre, toutes les requêtes sont obligatoires, ce qui est le cas en général, en pratique. En effet, quelques mois avant la construction du PVM, les requêtes affluent et les requêtes optionnelles sont alors traitées *a priori* : ces dernières deviennent alors obligatoires ou elles sont rejetées.

5.3 Tests numériques : la relaxation linéaire

Le logiciel GENCOL 4.3 est une implémentation informatique de la méthode de génération de colonnes. Celui-ci a été développé au GERAD depuis voici près de 20 ans et est utilisé par plusieurs compagnies aériennes à travers le monde pour résoudre, entre autres, des problèmes de rotations d'équipage. Il intègre un module complet de branchement et l'optimiseur auquel il fait appel est le logiciel CPLEX 6.5. Enfin, l'ensemble des réseaux a été construit avec la librairie de classes NetGen 5.3 qui elle-même a été développée à l'interne au GERAD. Tous les tests que nous effectuons dans ce chapitre ont été réalisés sur une machine SPARC Ultra-1/200. Tous les temps sont exprimés en secondes.

5.3.1 Comparaison des différents algorithmes

Nous avons noté au chapitre 4 par A_1, A_2, A_3 et A_4 les quatre algorithmes que nous avons développés. Désignons par S un ensemble de missions. Nous considérons pour nos tests trois ensembles S_1, S_2 et S_3 de cardinalité croissante. Plus précisément :

- $| S_1 | = 439$. S_1 contient l'ensemble des missions prédéfinies auquel nous avons rajouté des missions produites par le générateur.
- $| S_2 | = 6439$, les missions considérées ne comportent pas plus de trois requêtes.
- $| S_3 | = 43969$, les missions considérées ne comportent pas plus de cinq requêtes.

Bien sûr, $S_1 \subset S_2 \subset S_3$. Précisons de plus que, sauf mention contraire, les commodités sont agrégées et que les tests répertoriés dans les tableaux qui suivent ont été effectués sur le scénario 1.

Les résultats des tests pour la relaxation linéaire des problèmes sont présentés dans les tableaux 5.1, 5.2 et 5.3.

Tableau 5.1 – Comparaison des différents algorithmes : $S = S_1$

	A_1	A_2	A_3	$A_3 \cap A_2$	A_4	$A_4 \cap A_2$
Nombre de réseaux	3	3	3	3	59	59
Contraintes du PM	187	187	187	187	359	359
Nombre de noeuds	1269	668	1234	681	1382	874
Nombre d'arcs	1773	1172	1698	1145	1920	1412
Nombre de colonnes générées	3644	3875	1541	1669	615	613
Nombre d'itérations	2499	2660	934	1086	173	170
CPU_{PM}	255.4	276.0	88.7	98.1	4.1	4.6
CPU_{SP}	49.8	26.2	10.8	6.2	2.8	2.3
CPU_{TOT}	306.7	303.7	100.0	104.9	7.0	7.1

Tableau 5.2 – Comparaison des différents algorithmes : $S = S_2$

	A_1	A_2	A_3	$A_3 \cap A_2$	A_4	$A_4 \cap A_2$
Nombre de réseaux	3	3	3	3	57	57
Contraintes du PM	187	187	187	187	353	353
Nombre de noeuds	11448	7017	11317	6993	11619	7252
Nombre d'arcs	18142	13711	17937	13613	18369	14002
Nombre de colonnes générées	5558	6400	1984	1669	906	895
Nombre d'itérations	3892	4443	1061	1086	260	304
CPU_{PM}	441.3	493.9	142.7	132.4	7.8	7.7
CPU_{SP}	1002.3	667.0	191.0	100.2	56.2	46.3
CPU_{TOT}	1446.6	1164.2	334.7	233.4	65.2	54.8

Tableau 5.3 – Comparaison des différents algorithmes : $S = S_3$

	A_1	A_2	A_3	$A_3 \cap A_2$	A_4	$A_4 \cap A_2$
Nombre de réseaux	3	3	3	3	57	57
Contraintes du PM	187	187	187	187	353	353
Nombre de noeuds	54088	44547	53959	44523	54259	44782
Nombre d'arcs	98312	88771	98109	88673	98539	89062
Nombre de colonnes générées	4148	3973	1894	1963	982	851
Nombre d'itérations	2446	2329	987	993	362	261
CPU_{PM}	365.8	353.1	143.9	144.2	9.4	7.3
CPU_{SP}	3103.3	2410.2	903.9	755.5	722.6	476.8
CPU_{TOT}	3472.7	2766.0	1050.1	901.5	737.6	488.5

L'intersection $\cap$ signifie que nous combinons les algorithmes. Dans le cas des tableaux précédents, nous avons juste appliqué l'algorithme d'agrégation des réseaux avec les algorithmes A_3 et A_4. Nous pouvons faire un certain nombre de remarques concernant les tableaux 5.1, 5.2 et 5.3.

- L'algorithme d'agrégation permet de réduire la taille des sous-problèmes. Le tableau 5.4 détaille la réduction en pourcentage pour le nombre d'arcs et le nombre de noeuds, ce qui réduit en conséquence le temps de résolution total des sous-problèmes.

Tableau 5.4 – Efficacité de l'algorithme d'agrégation

	S_1	S_2	S_3
Arcs	31%	24%	10%
Noeuds	43%	38%	18%

- Le premier algorithme de coupes permet de diviser le nombre d'itérations par

un facteur 2.8 en moyenne sur l'ensemble des tests par rapport à l'algorithme de référence. Remarquons que la différence de taille entre les réseaux produits par A_1 et A_3 est très peu significative, ce qui suggère que peu d'arcs *repos* ont été éliminés par l'algorithme A_3. Chaque chemin généré par les sous-problèmes contient nécessairement les missions des intervalles bloqués du type de commodité identique qui réalise ce chemin, ce qui accélère la stabilisation des variables duales.

- Le deuxième algorithme de coupes permet de diviser le nombre d'itérations par un facteur 10 en moyenne sur l'ensemble des tests par rapport à l'algorithme de référence. Nous pouvons constater que la taille du problème maître augmente considérablement (elle double), ainsi que le nombre de réseaux (il est multiplié par 20), tandis que la taille des sous-problèmes, elle, ne change pratiquement pas.

5.3.2 Stratégies de résolution

Les sous-problèmes

À la vue des tableaux, nous pouvons constater que plus la cardinalité de l'ensemble des missions mises en jeu augmente, plus la part relative du temps de résolution des sous-problèmes augmente. Cette remarque s'applique plus particulièrement dans le cas de l'algorithme A_4 : cela provient du nombre élévé de réseaux. Remarquons que pour assurer la convergence de l'algorithme de génération de colonnes, il n'est pas nécessaire de résoudre les sous-problèmes de façon optimale à chaque itération. Ainsi, la résolution des sous-problèmes de façon heuristique permet parfois d'accélérer la résolution. Une telle approche est couramment utilisée dans des problèmes de fabrication de rotations d'équipage et d'horaires personnalisés. Les articles de Gamache *et al.* ([11]) et Desrosiers *et al.* ([6]) présentent des exemples de telles stratégies. De

plus, il n'est pas nécessaire de résoudre tous les sous-problèmes à chaque itération. Une itération peut prendre fin dès qu'un sous-problème a réussi à générer au moins une colonne de coût réduit négatif. Enfin, sans perte d'optimalité, il est possible, avant la résolution d'un sous-problème, d'éliminer du réseau correspondant l'ensemble des arcs missions de coût réduit positif. En effet, pour une mission dont le coût réduit est positif à une itération donnée, l'existence des arcs *repos* assure qu'il existe un meilleur chemin qui ne réalise pas cette mission (attention, l'existence de ces arcs n'est pas toujours assurée dans le cas des algorithmes A_3 et A_4). Lorsqu'au cours du branchement, nous prenons des décisions d'inter-tâches, alors, cette stratégie n'est plus valide. Le tableau 5.5 présente les résultats obtenus en appliquant les deux dernières stratégies proposées dans ce paragraphe. L'indice *ref* signifie que nous prenons pour référence les résultats obtenus dans le tableau 5.2.

Tableau 5.5 – La résolution des sous-problèmes : $S = S_2$

	Nbr d'itérations	CPU_{PM}	CPU_{SP}	CPU_{TOT}	$(CPU_{TOT})_{ref}$
A_1	5108	878.0	463.4	1345.0	1446.6
A_2	7410	933.5	343.8	1282.3	1164.2
A_3	2447	369.8	139.3	511.0	334.7
A_4	997	30.8	18.9	51.2	65.2

En fait, ce qui nous intéresse est le temps de calcul par itération, ce que détaille le tableau 5.6.

Celui-ci montre que ce genre de stratégies est bien plus efficace dans le cas de l'algorithme A_4, ce qui s'explique par le nombre élevé de réseaux qu'il construit (une cinquantaine au lieu de trois).

Tableau 5.6 – Le temps de calcul par itération

	Temps par Itération	$(\text{Temps par Itération})_{ref}$
A_1	0.26	0.37
A_2	0.17	0.26
A_3	0.21	0.32
A_4	0.05	0.25

Le problème-maître

Pour la suite des tests numériques, nous n'allons considérer que l'algorithme A_4 dans sa version non agrégée. Or, pour cet algorithme et pour les trois ensembles de missions considérés, le temps de résolution du problème-maître n'excède jamais 10 secondes (voir tableau 5.1, tableau 5.2 et tableau 5.3), ce qui ne justifie pas l'utilisation d'une perturbation ou le remplacement des contraintes de partitionnement par des contraintes de recouvrement.

Impact de la taille de S

Diminuer la cardinalité de S revient à restreindre le PFH, c'est ce que traduit le tableau 5.7. La valeur de la fonction objectif se dégrade tandis que le temps de résolution chute. Ce sera donc au planificateur de définir ses priorités par rapport au temps de résolution et à la qualité de la solution.

Influence de l'horizon temporel

Le tableau 5.8 suggère qu'une stratégie de découpage de l'horizon temporel peut se révéler éventuellement intéressante, en terme de temps de calcul, pour résoudre une instance du PFH. Le scénario 2 comporte 456 missions, tandis que le scénario 3

Tableau 5.7 – Impact de la taille de S : $A = A_4$

	S_1	S_2	S_3
Valeur de la fonction objectif	41145.3	39454.6	39245.14
CPU_{TOT}	7.0	65.2	737.6

en comporte exactement 32230. Rappelons que le scénario 2 couvre la période du 1 au 15 août et le scénario 3 la période du 15 au 31 août. La différence de cardinalité entre les deux ensembles vient du fait que durant la deuxième partie du mois, il y a beaucoup plus de requêtes de transport.

Tableau 5.8 – Impact de la durée de l'horizon temporel

	Sce 1, $S = S_1$	*Sce* 2	*Sce* 1, $S = S_3$	*Sce* 3
Valeur de la fonction objectif	41145.3	21876.5	39245.14	21990.65
CPU_{PM}	4.1	1.7	9.4	5.5
CPU_{SP}	2.8	1.3	722.6	248.6
CPU_{TOT}	7.0	3.1	737.6	255.8

5.4 Tests numériques : le branchement

Pour les scénarii considérés, toutes les requêtes sont obligatoires. Les deux branchements que nous avons mentionnés et qui traitent explicitement les requêtes optionnelles, à savoir le branchement sur le nombre de requêtes acceptées et l'acceptation ou rejet de requêtes ne sont donc pas pris en compte. De plus, les tests sont réalisés en tenant compte des stratégies que nous avons proposées pour la résolution des sous-problèmes.

5.4.1 Perturbation de la fonction objectif

Le tableau 5.9 montre que le nombre de colonnes fractionnaires à la fin de la relaxation est élevé et ce d'autant plus que le nombre de missions en jeu augmente. La profondeur dans l'arbre de branchement à laquelle nous sommes susceptibles d'obtenir une solution entière risque de s'avérer élevée. Rancourt ([13]) propose pour diminuer le nombre de colonnes fractionnaires de perturber la fonction objectif. Cette méthode consiste à perturber légèrement le coût de tout chemin réalisable d'un graphe G^k, tout en s'assurant que la solution optimale du PFH reste la même. Soulignons que l'emploi de cette méthode nécessite que les commodités ne soient pas agrégées.

Tableau 5.9 – Nombre de colonnes fractionnaires : commodités agrégées, $A = A_4$

	S_1	S_2	S_3
Nombre de colonnes fractionnaires	112	129	133
CPU_{TOT}	12.4	54.7	209.1

Tableau 5.10 – Nombre de colonnes fractionnaires : commodités désagrégées, $A = A_4$

	S_1	S_2	S_3
Nombre de colonnes fractionnaires	17	39	49
Nbr de réseaux	101	98	98
CPU_{PM}	18.4	34.7	36.7
CPU_{SP}	4.8	66.1	234.3
CPU_{TOT}	24.0	104.1	282.5

Le tableau 5.10 montre qu'il est possible de diminuer de manière significative le nombre de variables fractionnaires. Cela se fait au détriment du temps de calcul. Nous

constatons que l'augmentation du temps de résolution provient des sous-problèmes et d'une augmentation du phénomène de dégénérescence au niveau du problème maître.

5.4.2 Stratégie de branchement

Nous proposons dans le tableau 5.11 quatre stratégies de branchement. Pour chacune de ces stratégies, les méthodes de branchement retenues ont été ordonnées. Ainsi, lorsqu'une décision de branchement doit être prise, la première méthode qui peut être appliquée est choisie pour la prise d'une décision.

Tableau 5.11 – Stratégies de branchement

Stratégie	$itfix$	$cfix$	$afix$	$mfix$
1	1	-	-	-
2	2	1	-	-
3	3	2	1	-
4	3	2	1	2

Dans les tableaux qui suivent, la colonne noeud associé donne le numéro du noeud où a été obtenue la meilleure solution, sa valeur étant indiquée dans la première colonne.

Dans les tableaux 5.12 et 5.13, les seuils pour les méthodes $mfix$ et $cfix$ ont été fixés à 0.8. L'astérisque signifie que le seuil a été abaissé à 0.6. La stratégie d'exploration que nous avons retenue est la stratégie dite profondeur d'abord (*depthfirst*). Les deux critères d'arrêt sont les suivants :

- Le nombre de noeuds explorés dans l'arbre égale 500.
- Le saut d'intégrité ou gap est inférieur à 1%.

Le tableau 5.12 nous permet de tirer quelques conclusions. Tout d'abord, nous sommes capables d'obtenir une solution entière à moins de 1.5 % de la valeur de

Tableau 5.12 – Les différentes stratégies : commodités agrégées, $S = S_2$

Stratégie	Solution	Noeud associé	CPU_{TOT}	Gap
1	40077	200	429.4	1.577
2	40000	88	234.0	1.382
2*	40074	89	297.3	1.570
3	39861	298	2144.6	1.030
3*	39830	106	430.8	0.951
4	39830	103	834.0	0.951
4*	39838	57	496.3	0.972

la relaxation et ce en un temps relativement raisonnable (moins de 10 minutes en moyenne), compte tenu de la taille du problème. Pour les stratégies faisant appel à $cfix$ ou à $mfix$, plus le seuil est bas, plus vite nous obtenons une solution entière, mais avec le risque d'avoir pris des mauvaises décisions, d'où un saut d'intégrité plus élevé. Finalement, nous retenons la troisième stratégie qui présente un bon compromis : elle fournit des meilleures solutions entières que les stratégies 1 ,2 et 2* et les temps sont inférieurs aux stratégies 4 et 4*.

Tableau 5.13 – Les différentes stratégies : commodités désagrégées, $S = S_2$ (1)

Stratégie	Solution	Noeud associé	CPU_{TOT}	Gap
1	40008.18	57	409.3	1.318
2	39925.46	21	267.8	1.115
2*	40408.34	24	317.2	2.343
3	39885.01	19	404.3	0.998
3*	39869.19	40	1216.8	0.973
4	39868.93	64	2107.1	0.972
4*	39864.08	45	1394.8	0.950

Dans le tableau 5.13, les commodités ont été désagrégées. Nous constatons que la profondeur à laquelle nous trouvons la meilleure solution entière s'avère beaucoup moins élevée que dans le cas où les commodités sont agrégées. Mais le temps de calcul augmente considérablement, à cause de la multiplication des réseaux et de la convergence beaucoup plus lente du processus de résolution. Gamache *et al.* ([11]) propose une stratégie d'échantillonnage de réseaux. À une itération donnée, le jeu des variables duales a tendance à faire produire des colonnes qui couvrent sensiblement les mêmes tâches. L'idée consiste donc à ne résoudre que les réseaux prometteurs. Le nombre total d'itérations croît, mais cela est compensé par la diminution du temps de calcul par itération. Dans le cadre spécifique de notre problème, nous pourrions aussi utiliser cette stratégie dans le cas agrégé : nous constatons en effet en pratique que le nombre de colonnes générées par rapport au nombre de problèmes de plus court chemin résolus est faible. Dans le cas désagrégé, nous proposons la stratégie suivante. Nous introduisons tout d'abord le concept de *groupe de réseaux identiques* : deux réseaux appartiennent au même *groupe de réseaux identiques* si ils sont tous les deux la copie du même réseau de la version agrégée. L'idée est la suivante : à une itération donnée, nous choisissons un réseau pour un *groupe de réseaux identiques* donné. Si celui-ci est résolu avec succès, alors nous passons à l'itération suivante, sinon, nous considérons un autre *groupe de réseaux identiques* et procédons à la même opération. Cette procédure, à la perturbation près, nous assure d'être $\epsilon - optimal$ à la fin de la relaxation. Pour garantir l'optimalité, nous devons, si aucun des réseaux que nous avons choisi n'a été résolu avec succès, considérer tous les autres réseaux. Nous présentons les résultats dans le tableau 5.14, sachant que tous les noeuds sont résolus à l'optimalité.

En fait, pour apprécier la valeur de notre stratégie, il faut comparer le nombre de problème de plus court chemin résolus par rapport au nombre de colonnes générées. C'est ce que présente le tableau 5.15.

Tableau 5.14 – Les différentes stratégies : commodités désagrégées, $S = S_2$ (2)

Stratégie	Solution	Noeud associé	CPU_{TOT}	Gap
1	39933.89	37	252.1	1.132
2	39877.16	20	200.5	0.987
2*	40784.46	19	308.6	3.283
3	39867.98	96	998.3	0.956
3*	39840.88	33	631.1	0.897
4	39895.48	35	767.2	1.033
4*	39861.52	45	727.0	0.953

Tableau 5.15 – Nombre de problèmes de plus court chemin résolus sur nombre de colonnes générées, $S = S_2$

Stratégie	Cas agrégé	Cas désagrégé	Cas désagrégé + stratégie
1	12.0	15.7	14.7
2	11.7	17.6	8.5
2*	13.9	21.8	18.7
3	13.1	15.5	12.0
3*	10.7	18.3	11.9
4	11.2	17.5	14.4
4*	10.7	19.2	10.5

Nous présentons dans le tableau 5.16 (respectivement 5.17) les résultats obtenus pour les trois ensembles de missions que nous avons considérés dans le cas où les commodités sont agrégées (respectivement dans le cas où la fonction objectif a été perturbée). Les critères d'arrêt n'ont pas changé.

Tableau 5.16 – Commodités agrégées : résultats pour S_1, S_2 et S_3

Ensemble S	Solution	Noeud associé	CPU_{TOT}	Gap
S_1, 3	41273	64	42.9	0.310
S_1, 3*	41273	37	29.5	0.310
S_2, 3	39861	298	2144.6	1.030
S_2, 3*	39830	106	430.8	0.951
S_3, 3	39830	139	2628.8	1.490
S_3, 3*	39862	71	1915.4	1.572

Tableau 5.17 – Perturbation de la fonction objectif : résultats pour S_1, S_2 et S_3

Stratégie	Solution	Noeud associé	CPU_{TOT}	Gap
S_1, 3	41410.45	5	34.5	0.554
S_1, 3*	41408.69	3	35.0	0.553
S_2, 3	39867.98	96	998.3	0.956
S_2, 3*	39840.88	33	631.1	0.897
S_3, 3	39749.90	22	3767.4	1.19 *
S_3, 3*	39620.35	31	5673.4	0.870

$*$: dans le tableau 5.17, pour $S = S_3$ et avec la stratégie 3, précisons que la meilleure solution entière est obtenue en 39032.8 secondes au noeud 200, la valeur de la fonction objectif associée étant 39668.41 (soit un saut d'intégrité de 0.999 %).

Nous constatons que ce que nous gagnons en terme de coût en augmentant la cardinalité de S (surtout lorsque nous passons de l'ensemble S_2 à S_3) ne se justifie pas

compte tenu de l'augmentation considérable du temps de calcul. Dans les tableaux 5.18 et 5.19 , nous présentons les résultats obtenus pour un nouvel ensemble de missions S_4 ($| S_4 | = 19297$). Les missions considérées dans S_4 ne couvrent pas plus de quatre requêtes.

Tableau 5.18 – Commodités agrégées : résultats pour S_4

Stratégie	Solution	Noeud associé	CPU_{TOT}	Gap
3	39836	333	2409.7	1.385
3*	39829	65	804.4	1.368

Tableau 5.19 – Perturbation de la fonction objectif : résultats pour S_4

Stratégie	Solution	Noeud associé	CPU_{TOT}	Gap
3	39619.13	21	1256.7	0.751
3*	39682.67	17	752.5	0.912

Les temps obtenus sont très variables et dépendent fortement de la cardinalité de S. Il est difficile de les comparer avec ceux obtenus par Rancourt ([13]) car les scénarii que nous avons définis ne sont pas strictement identiques. Cependant, le scénario retenu par Rancourt se rapprochant le plus du scénario 1 est résolu en 3143 secondes sur la même machine avec laquelle nous avons réalisé nos tests. Cependant, il est assez difficile de comparer la qualité des solutions obtenues car l'ensemble des contraintes prises en compte n'est pas strictement identique.

5.5 Conclusion

Dans la partie de ce chapitre consacrée aux résultats numériques, nous avons mis en évidence la supériorité des réseaux construits par l'algorithme A_4. Cela s'explique

par le fait que ces réseaux intègrent des propriétés intrinsèques au PFH, comme l'existence d'intervalles bloqués (voir chapitre 4).

Au cours de la résolution, à chaque itération, peu de colonnes sont générées. Cela nous a suggéré, comme stratégie de résolution, d'arrêter une itération dès que l'oracle trouve une colonne de coût réduit négatif. Cela nous a permis de diminuer considérablement le temps de calcul. Mais, le nombre de problèmes de plus court chemin résolus par rapport au nombre de colonnes générées est encore élevé. Il serait intéressant de pouvoir identifier *a priori* les réseaux prometteurs, c'est-à-dire les réseaux susceptibles de générer des colonnes. La stratégie de perturbation de la fonction objectif permet effectivement de diminuer le nombres de variables fractionnaires à la fin de la relaxation. Mais la multiplication des réseaux augmente le temps de résolution des sous-problèmes donc le temps de résolution total. Nous avons proposé une stratégie efficace pour contrer ce problème. De plus, la convergence, du fait de la désagrégation et de la perturbation que nous avons introduite, devient plus lente, ce qui augmente le temps de résolution par noeud. Les résultats que nous présentons dans ce mémoire ne permettent pas de conclure quant à la supériorité de la stratégie de perturbation par rapport à la stratégie initiale (c'est-à-dire lorsque nous considérons les commodités agrégées). Il semblerait toutefois que la stratégie de perturbation permette d'obtenir de meilleures solutions.

Les stratégies de branchement que nous avons développées (c'est-à-dire $mfix$ et $afix$) diminuent légèrement le saut d'intégrité en contrepartie d'une augmentation sensible du temps de calcul.

CHAPITRE 6

Conclusion

Dans ce mémoire, nous avons développé un modèle de programmation mathématique pour assister les planificateurs du GTA dans la production d'horaires de vol valides. Nous avons poursuivi les travaux de maîtrise de Rancourt ([13]). Ce dernier développe un modèle mathématique qui construit les missions à partir de segments de vol, ce qui signifie que les missions sont générées au cours du processus de résolution. Nous proposons une approche différente où les missions sont construites *a priori* par un générateur et deviennent des entrées au problème.

Cette modélisation présente des avantages indéniables dans le contexte où ce projet a été défini, c'est-à-dire un système d'aide à la décision. En effet, il devient facile d'ajouter ou d'enlever des missions ou des contraintes au PFH. Cela se répercute sur l'ensemble des missions retenues pour l'instance considérée et n'affecte pas le modèle mathématique. Le temps de résolution ne dépend pratiquement que de la cardinalité de l'ensemble des missions retenues.

Nous proposons des algorithmes efficaces pour construire nos réseaux à partir d'un ensemble de missions. Nous avons repris la méthode de perturbation de la fonction objectif proposé par Rancourt ([13]). Cependant, il serait intéressant pour bénéficier des avantages de cette méthode de développer des stratégies d'échantillonnage des réseaux. Nous pourrions envisager aussi un procédé itératif d'échantillonnage des missions. Le principe serait le suivant : le PFH serait résolu avec un ensemble de missions données, puis à cet ensemble, nous rajouterions des missions prometteuses

et le PFH serait résolu à nouveau.

RÉFÉRENCES

[1] BARNHART, C., JOHNSON, E.L., NEMHAUSER, G.L., SAVELSBERGH, M.W.P. et VANCE, P.H. (1960). Branch and Price : Column Generation for Solving Huge Integer Programs. Operations Research, 46, 316–329.

[2] DANTZIG, G.B. et WOLFE, P. (1960). Decomposition Principle for Linear Programs. Operations Research, 8, 101–111.

[3] DESAULNIERS, G., DESROSIERS, J., GAMACHE, M. et SOUMIS, F. (1997a). Crew Scheduling in Air Transportation. Dans : T.G. Crainic et G. Laporte (eds.), *Fleet Management and Logistics*, Kluwer, Boston, 169–185.

[4] DESAULNIERS, G., DESROSIERS, J., DUMAS, Y., MARC, S., RIOUX, B., SOLOMON, M.M. et SOUMIS, F. (1997b). Crew Pairing at Air France. European Journal of Operational Research, 97, 245–259.

[5] DESAULNIERS, G., DESROSIERS, J., IOACHIM, I., SOLOMON, M.M., SOUMIS, F. et VILLENEUVE, D. (1998). A Unified Framework for Deterministic Time-Constrained Vehicle Routing and Crew Scheduling Problems. Dans : T.G. Crainic et G. Laporte (eds.), *Fleet Management and Logistics*, Kluwer, Boston, 57–93.

[6] DESAULNIERS, G., DESROSIERS, J. et SOLOMON, M.M. (1999). Accelerating Strategies in Column Generation Methods for Vehicle Routing and Crew Scheduling Problems. Les cahiers du GERAD, G–99–36, École des Hautes Études Commerciales, Montréal.

[7] DESAULNIERS, G., DESROSIERS, J. et SOLOMON, M.M. (2000). The VRP with Pickup and Delivery. Les cahiers du GERAD, G–00–25, École des Hautes Études Commerciales, Montréal.

[8] DESROCHERS, M. et SOUMIS, F. (1988). A Generalized Permanent Labelling Algorithm for the Shortest Path Problem with Time Windows. INFOR, 26, 193–214.

[9] DESROSIERS, J., DUMAS, Y., SOLOMON, M.M. et SOUMIS, F. (1995). Time Constrained Routing and Scheduling. Handbooks in OR & MS, Network Routing, vol. 8, M.O. Ball et al., Elsevier Science B.V., Amsterdam, 35–139.

[10] DU MERLE, O., VILLENEUVE, D., DESROSIERS, J. et HANSEN, P. (1999). Stabilized Column Generation. Discrete Mathematics, 194, 229–237.

[11] GAMACHE, M., SOUMIS, F., MARQUIS, G. et DESROSIERS, J. (1997). A Column Generation Approach for Large Scale Aircrew Rostering Problems. Les cahiers du GERAD, G–94–20, École des Hautes Études Commerciales, Montréal.

[12] NAGIH, A. et SOUMIS, F. (1999). L'aggrégation des contraintes de ressources dans un problème de plus court chemin. Les cahiers du GERAD, G–99–02, École des Hautes Études Commerciales, Montréal.

[13] RANCOURT, E. (1998a). Planification des vols pour le Groupe de Transport Aérien des Forces Armées Canadiennes. Mémoire de maîtrise (École Polytechnique de Montréal).

[14] RANCOURT, E. (1998b). Decision Support Systems for Simultaneous Aircraft and Crew Scheduling, Overview of the Canadian Forces and the Air Force. Document interne du GERAD.

[15] RANCOURT, E. et SAVARD, G. (1999). Decision Support Systems for Simultaneous Aircraft and Crew Scheduling, Statement of the problem. Document interne du GERAD.

[16] RAPPOPORT, H.K., LEVY, L.S., GOLDEN, B.L. et TOUSSAINT, K.J. (1992). A Planning Heuristic for Military Airlift. INTERFACES, 22, 73–87.

[17] SOLANKI, R.S. et SOUTHWORTH, F. (1991). An Execution Planning Algorithm for Military Airlift. INTERFACES, 21, 121–131.

[18] 1 CAD Orders - Volume 2, Book 3, Chapter 1.