Titre: Title:	Générateur de missions de DSS-1-CAD : extensions et analyse des solutions
Auteur: Author:	Yasmine Yacef
Date:	2004
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Yacef, Y. (2004). Générateur de missions de DSS-1-CAD : extensions et analyse des solutions [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7520/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7520/
Directeurs de recherche: Advisors:	Gilles Savard, & Michel Gamache
Programme: Program:	Unspecified

UNIVERSITÉ DE MONTRÉAL

GÉNÉRATEUR DE MISSIONS DE DSS-1-CAD : EXTENSIONS ET ANALYSE DES SOLUTIONS

YACEF YASMINE
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)
AOÛT 2004

Your file Votre référence
ISBN: 0-612-97990-3
Our file Notre référence
ISBN: 0-612-97990-3

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

GÉNÉRATEUR DE MISSIONS DE DSS-1-CAD : EXTENSIONS ET ANALYSE DES SOLUTIONS.

présenté par : YACEF Yasmine
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. GAMACHE Michel, Ph.D., membre et codirecteur de recherche
M. ROUSSEAU Louis-Martin, Ph.D., membre

À ma mère, mon époux et mes enfants ...

REMERCIEMENTS

Je tiens à remercier mon directeur de recherche monsieur Gilles Savard pour ses conseils et son aide financière.

Je remercie monsieur Michel Gamache mon codirecteur de recherche pour son aide, ses conseils et son temps.

Je remercie monsieur François Soumis qui m'a dirigée au début de ma maîtrise.

Je remercie monsieur Eric Rancourt étudiant au doctorat pour son aide.

Je remercie tout particulièrement mon mari pour sa patience, son soutien et ses encouragements.

Et tous ceux qui m'ont aidée de près ou de loin dans mon travail.

RÉSUMÉ

Le travail de ce mémoire s'inscrit dans le cadre du problème de fabrication de plans de vol mensuels pour la division 1-CAD des forces armées canadiennes. Une formulation multi-commodités et une formulation de partionnement d'ensembles généralisé ont été proposées pour ce problème dans des travaux précédents (Rancourt (1998) et Guigue (2000)). La résolution se fait par séparation et évaluation progressive où les bornes sont calculées par génération de colonnes.

L'objectif de cette étude est dans un premier temps de prendre en compte de manière implicite des contraintes qui jusqu'à présent avaient été considérées de manière explicite. De ce fait, des changements ont été apportés à un module appelé générateur de missions, du système d'aide à la décision (DSS) développé pour résoudre le problème de fabrication de plans de vol mensuels.

La seconde partie de l'étude consiste à analyser les missions fournies par le générateur de missions dans le but de les classer selon certains critères afin d'en sélectionner un certain nombre que nous considérons être les «meilleures» au sens de ces critères. Cette sélection permet la réduction du nombre de missions à considérer durant le processus de résolution afin de réduire les temps de calculs tout en gardant une bonne qualité de solution.

ABSTRACT

The work of this master thesis is an extension of the previous works of Rancourt(1998) and Guigue(2000) for the problem of building flight schedules of the First Canadian Air Force Division (1-CAD). Two mathematical formualtions have been proposed in these works : a multi commodity flow formulation and a generalized set patitionning formulation. A branch and bound algorithm is used to solve the problem where the lower bounds are evaluated by column generation.

The aim of this study is first, to consider implicitly some explicit constraints which leads to make some changes to a module called the missions generator, of the desicion support systeme (DSS) elaborated to solve the flight scheduling problem.

The second part of the study is to analyse the missions obtained from the missions generator in order to classify them according to some criteria and then select the "best" missions. This selection is made to reduce the number of missions that we have to take into account when solving the problem in order to reduce the computation time and to keep a good solution.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

TABLE DES FIGURES

LISTE DES ANNEXES

CHAPITRE 1 : INTRODUCTION ET DÉFINITION DU PROBLÈME

1.1 Introduction

Le Groupe de Transport Aérien (GTA) de la division 1-CAD des Forces Armées Candiennes a pour mission de fournir des vols de transport stratégiques et tactiques à travers le monde ainsi que des vols pour les opérations de recherche et de sauvetage. Pour ce faire, le GTA doit gérer des ressources réparties sur différentes bases du territoire canadien afin de satisfaire des demandes ou *requêtes* de différentes natures qui lui sont soumises. Les ressources disponibles n'étant pas suffisantes pour satisfaire toutes les demandes, ces dernières se voient attribuées des priorités numérotée de 1 à 8 (la plus haute étant 1) de la façon suivante :

- priorité 1 : vols d'urgence et transport de dignitaires.
- priorité 2 : mission d'entraînement du personnel navigant du GTA.
- priorité 3 : vols réguliers de réapprovisionnement dans le nord et le réapprovisionnement des forces de maintien de la paix.
- priorité 4 : transport spécial des Forces Armées Canadiennes.
- priorité 5 : exercices conjoints des différents commandements des Forces Armées Canadiennes.
- priorité 6 : vols réguliers de transport de passagers.
- priorité 7 : vols réguliers de transport de fret.
- priorité 8 : vols spéciaux qui n'appartiennent à aucune des catégories précédentes.

L'acceptation des demandes se fait selon le niveau de priorité, c'est-à-dire une demande de priorité 1 est satisfaite avant celle de priorité 2 et ainsi de suite. Pour les

demandes de même niveau de priorité, la satisfaction se fait selon le principe du premier arrivé premier servi. Certaines demandes peuvent être combinées, par exemple une demande de transport de matériel avec un vol d'entraînement.
En fonction des demandes qui arrivent au GTA, les planificateurs construisent ce que l'on appelle un plan de vol mensuel (PVM) pour chacun des mois de l'année. Le PVM est constitué de *lignes d'affectation* qui sont associées aux appareils disponibles en tout temps à chacune des différentes bases. Le nombre d'appareils disponibles est déterminé en fonction de la taille de la flotte, des équipages disponibles et des contraintes d'entretien des appareils. Chaque ligne correspond à un aéronef générique d'un certain type. Durant la construction du PVM, les planificateurs affectent à chacune des lignes un certains nombres de suite de vols. Comme une panne d'appareil, une annulation de requêtes, une demande de dernière minute ou encore des conditions météorologiques difficiles peuvent survenir à tout moment, il faut pouvoir ajuster le PVM en conséquence.
Ce problème a été étudié par Rancourt (1998) et Guigue (2000), dans le cadre de leurs travaux de maîtrise. Rancourt (1998) propose une première modélisation à laquelle Guigue (2000) apporte des modifications. Nous allons dans le reste de ce chapitre rappeler ces modèles, mais auparavant nous donnons une définition détaillée du problème.

1.2 Définition du problème

Avant de donner une définition formelle du problème et afin de faciliter la lecture de ce qui suit on propose, dans la section ci-dessous, un lexique des termes utilisés.

1.2.1 Définitions

- *Base* : base militaire à laquelle un appareil est rattaché.

- *Aéroport* : base ou tout autre aéroport pouvant permettre le décollage et l'atterrissage d'un appareil.
- *Segment* : vol sans interruption entre deux aéroports.
- *Connexion* : arrêt entre deux segments dans la même journée de travail, dont la durée correspond au temps entre la fin du premier segment et le début du deuxième.
- *Service de vol* : suite de segments et de connexions effectués par un même équipage durant une journée de travail.
- *Briefing* : période de temps pour préparer le service de vol qui se termine au début du premier segment.
- *Debriefing* : période de temps pour finir le service de vol, dont la durée est le temps écoulé entre la fin du dernier segment et la fin du service de vol.
- *Repos* : temps entre deux services de vol consécutifs accomplis par le même équipage.
- *Demande ou requête de transport* : déplacement de fret (par fret on entend, passagers et/ou matériels).
- *Demande ou requête générique* : demande d'un appareil pour effectuer une mission d'entraînement, une mission régulière ou supporter un exercice des autres unités des Forces Armées Canadiennes.
  Les requêtes ou demandes peuvent être soit *obligatoires* (dans ce cas elles ne peuvent pas être annulées) soit *optionnelles.*
- *Groupe de requêtes optionnelles* : est un ensemble de requêtes optionnelles qui doivent être toutes acceptées ou toutes rejetées en même temps.
- *Mission* : suite de services de vol et de repos débutant et finissant à la même base réalisés par un ou plusieurs équipages sur un même aéronef.
  On distingue deux types de missions :

- *mission prédéfinie* : mission déjà construite et donnée en entrée au problème qui couvre une requête générique et qui peut être :
  1. soit *obligatoire*, c'est-à-dire elle doit obligatoirement faire partie de la solution comme par exemple, une mission qui est la seule mission à couvrir une requête obligatoire ;
  2. soit *optionnelle*.
- *mission non prédéfinie* : mission qui peut être construite durant le processus de résolution du problème afin de faire partie de la solution. Ce type de mission ne peut couvrir que des requêtes de transport.

Le tableau 1.1 résume certaines notations mathématiques utilisées dans le reste du mémoire. Pour une meilleure compréhension et pour faciliter le retour aux références, nous utiliserons dans ce qui suit, les mêmes notations que celles de Rancourt (1998).

Tableau 1.1 – Notations générales

K	l'ensemble des lignes d'affectation ou commodités
A	l'ensemble des aéroports.
f_{ij}^k	la durée d'un segment de $i \in A$ vers $j \in A$ pour $k \in K$
T	l'ensemble des requêtes
T^F	l'ensemble des requêtes de transport
C_k	l'ensemble des ensembles de requêtes de transport compatibles pour $k \in K$
O	l'ensemble des groupes de requêtes optionnelles.

1.2.2 Le problème

La construction de PVM est en fait un problème de fabrication d'horaires (PFH) qui a les mêmes caractéristiques que le problème de tournées de véhicules avec collectes et livraisons et que le problème de fabrication de rotations d'équipages.

L'objectif est de maximiser le nombres de requêtes acceptées tout en respectant les priorités. Pour des solutions équivalentes du point de vue des priorités, on favorisera celles qui minimisent les coûts d'opérations tels que le carburant et les coûts d'entretien. Donc une solution optimale du PFH sera une solution respectant les priorités, maximisant le nombres de requêtes couvertes et ayant un coût minimum.
Les contraintes que doivent respecter les planificateurs lors de la construction des PVM ou de leur modification en cas de nécessité sont les suivantes.

- *La réglementation concernant le personnel navigant.* Selon la convention de travail, la durée maximale d'un service de vol est de 18 heures. Dans le cas où le service de vol débute entre 18:00 et 07:59 heure local de l'aéroport de départ, alors la durée passe à 16 heures et elle est de 14 heures pour certaines missions. La durée minimale de repos entre deux services de vol est de 14 heures.
- *Les heures d'ouvertures des aéroports.* De manière évidente, si un aéroport est fermé, l'aéronef ne peut ni y atterrir, ni en décoller.
- *Les restrictions sur les combinaisons possibles de fret et la compatibilité avec l'appareil utilisé.* De part la nature du fret qui doit être transporté, il est clair que certaines combinaisons ne sont pas permises, comme par exemple le transport de matériel explosif avec du carburant. Par ailleurs, le type d'appareil utilisé peut ne pas convenir au transport de certain type de fret.
- *Les contraintes de préséance et de couplage.* La première assure que le chargement du fret d'une requête se fasse avant son déchargement et la seconde que le chargement et le déchargement se fassent par le même appareil.

1.2.3 Problèmes de fabrication d'horaires

Avant de présenter la modélisation du problème que nous verrons à la section suivante, nous allons citer quelques articles qui traitent de la classe de problèmes de

fabrication d'horaires (PFH) dont le problème de construction de PVM fait parti. Beaucoup d'articles ont traité des PFH, parmis lesquels on peut citer celui de Desaulniers et *al* (1997), dans lequel les auteurs proposent une formulation unifiée pouvant modéliser la plupart des PFH. Cette formulation est un modèle non linéaire de flots dans un réseau multi-commodités avec contraintes de ressource. La résolution de ce modèle se fait par une méthode de séparation et évaluation progressive où le le calcul des bornes inférieures se fait par la résolution d'un modèle équivalent obtenu par une extension du principe de Dantzig-Wolfe. Ce problème est un problème de partitionnement d'ensemble avec contraintes supplémentaires où les colonnes correspondent aux points extrêmes des sous-problèmes de la formulation originale.

Le principe de décomposition de Dantzig-wolfe ou encore la méthode de génération de colonnes est l'objet de plusieurs articles, dont celui de Barnhart et *al* (1998), qui présente une formulation générique d'un problème en nombres entiers et explique les différentes étapes de la méthode de génération de colonnes appliquées à ce problème.

La présence de contraintes d'équipages dans le problème de construction du PVM, nous amène à parler des problèmes de fabrication de rotation d'équipages qui ont été étudiés par exemple dans Desaulniers et *al* (1998) où les auteurs proposent deux modèles basés sur la formualtion unifiée de Desaulniers et *al* (1997).

Finalement nous citerons, du fait de l'existance de requêtes de transport, l'article de Dumas, Desrosiers et Soumis (1991) qui proposent un algorithme exact pour certains problèmes de tournées de véhicules avec collectes, livraisons et fenêtres de temps pour le transport de marchandises. Cet algorithme utilise la génération de colonnes avec pour sous-problème, un problème de plus court chemin qui prend en compte les contraintes de couplage, de préséance, de capacité et de fenêtres de temps.

Le lecteur intéressé pourra consulter Rancourt (1998) et Guigue (2000) pour une revue de littérature plus détaillée.

1.3 Modélisation

Pour le problème décrit plus haut, Rancourt (1998) propose deux modèles :
- un modèle de multiflots en nombres entiers mixte ;
- un modèle de partitionnement d'ensemble generalisé ;

Chaque modèle nécessite la construction de réseaux espace-temps $G^k = (V^k, A^k)$ où V^k est l'ensemble des nœuds qui en plus de représenter un endroit, en l'occurence un aéroport, représentent une période de temps ou fenêtre de temps et A^k est l'ensemble des arcs qui représentent des segments de vol.
On retrouve par la suite le travail de Guigue (2000) qui lui aussi propose le modèle de multiflots et celui de partitionnement d'ensemble, mais qui considère l'existence de missions générées *a priori* et dans son cas les arcs des réseaux représentent des missions.
Nous allons dans cette section, présenter de façon succincte, ces modèles et les différences entre eux. Le lecteur peut se rapporter à Rancourt (1998) et Guigue (2000) pour plus de détails.

1.3.1 Modélisation proposée par Rancourt(1998)

Dans cette sous-section ainsi que dans la suivante (celle qui présente les modèles de Guigue (2000)), nous nous contenterons de donner la formulation et de l'expliquer brièvement.

Modèle de multiflots

Avant de donner la formulation proprement dite, on définit les constantes et les variables nécessaires à l'écriture du modèle.

Pour chaque commodité $k \in K$, chaque arc $(i,j) \in A^k$ et chaque requête $t \in T$, soit la constante a_{ij}^{kt}, telle que :

$$a_{ij}^{kt} = \begin{cases} 1 & \text{si l'arc correspond à une mission prédéfine pour une requête générique ou correspond à la collecte d'une requête de transport,} \\ 0 & \textit{sinon.} \end{cases}$$

Pour chaque commodité $k \in K$, chaque arc $(i,j) \in A^k$ et chaque requête de transport $t \in T^F$, la constante d_{ij}^{kt} se définit comme suit :

$$d_{ij}^{kt} = \begin{cases} 1 & \text{si l'arc correspond à la livraison d'une requête de transport,} \\ 0 & \textit{sinon.} \end{cases}$$

Finalement, la constante b_o^t se définit comme suit :

$$b_o^t = \begin{cases} 1 & \text{si la requête } t \in T \text{ appartient au groupe de requêtes optionnelles } o \in O, \\ 0 & \textit{sinon.} \end{cases}$$

Si la requête t est obligatoire, alors on a $\sum_{o \in O} b_o^t = 0$; sinon on a $\sum_{o \in O} b_o^t = 1$.

Pour chaque réseau $G^k = (V^k, A^k)$, on associe $\forall (i,j) \in A^k$ une variable binaire de flot X_{ij}^k, telle que :

$$X_{ij}^k = \begin{cases} 1 & \text{si l'arc } (i,j) \text{ est utilisé par la commodité } k \\ 0 & \text{sinon.} \end{cases}$$

Pour l'ensemble des ressources $R^k, \forall k \in K$, représentant la durée de service de l'équipage, un instant de l'horizon temporel et l'état (chargé ou non) du fret d'une requête,

les variables de ressource $T_i^{kr}, \forall i \in V^k, \forall r \in R^k, \forall k \in K$, indiquent la valeur de la ressource r au nœud i si ce nœud est visité par la commodité k. Dans le cas contraire, la valeur prise par cette variable est sans importance. Le vecteur des variables de ressource au nœud $i \in V^k, k \in K$ est noté par T_i^k.

Il faut également ajouter les variables binaires $Z_o, \forall o \in O$ pour les groupes de requêtes optionnelles, qui prennent les valeurs :

$$Z_o = \begin{cases} 1 & \text{si toutes les requêtes du groupe de requêtes optionnelles } o \text{ sont rejetées,} \\ 0 & \text{si toutes les requêtes du groupe de requêtes optionnelles } o \text{ sont acceptées.} \end{cases}$$

Ainsi que la variable entière Y qui compte le nombre de groupes de requêtes optionnelles dont les requêtes sont rejetées. Cette dernière variable sert uniquement pour le branchement sur le nombre de requêtes acceptées dans un algorithme de séparation et d'évaluation progressive.

On associe un coût c_{ij}^k à chaque arc $(i,j) \in A^k$ qui est égal à $f_{l_i l_j}^k$ s'il s'agit d'un segment de vol, au coût de la missions c_m s'il s'agit d'une mission prédéfinie et à 0 sinon.

On attribue une pénalité c_o pour le rejet des requêtes du groupe de requêtes optionnelles o.

Le modèle du multiflots prend alors la forme suivante :

$$\min \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij}^k X_{ij}^k + \sum_{o \in O} c_o Z_o \tag{1.1}$$

s.c :

$$\sum_{k \in K} \sum_{(i,j) \in A^k} a_{ij}^{kt} X_{ij}^k + \sum_{o \in O} b_o^t Z_o = 1, \qquad \forall t \in T \tag{1.2}$$

$$\sum_{k \in K} \sum_{(i,j) \in A^k} d_{ij}^{kt} X_{ij}^k + \sum_{o \in O} b_o^t Z_o = 1, \qquad \forall t \in T^F \tag{1.3}$$

$$\sum_{o \in O} Z_o - Y = 0, \tag{1.4}$$

$$Y \quad \text{entier}, \tag{1.5}$$

$$Z_o \in \{0,1\}, \qquad \forall o \in O \tag{1.6}$$

$$\sum_{j:(o(k),j)\in A^k} X_{o(k),j} = 1, \qquad \forall k \in K \tag{1.7}$$

$$\sum_{j:(j,d(k))\in A^k} X_{j,d(k)} = 1, \qquad \forall k \in K \tag{1.8}$$

$$\sum_{j:(j,i)\in A^k} X^k_{ji} - \sum_{j:(i,j)\in A^k} X^k_{ij} = 0, \qquad \forall k \in K, \forall i \in V^k \backslash \{o(k), d(k)\} \tag{1.9}$$

$$X^k_{ij}(f^{kr}_{ij}(T^k_i) - T^{kr}_j) \leq 0, \qquad \forall k \in K, \forall (i,j) \in A^k, \forall r \in R^k \tag{1.10}$$

$$a^{kr}_i \leq T^{kr}_i \leq b^{kr}_i, \qquad \forall k \in K, \forall i \in V^k, \forall r \in R^k \tag{1.11}$$

$$X^k_{ij} \in \{0,1\}, \qquad \forall k \in K, \forall (i,j) \in A^k. \tag{1.12}$$

La fonction objectif (1.1) minimise la somme des coûts des lignes d'affectation ainsi que la somme pondérée des requêtes optionnelles rejetées. Les contraintes (1.2) assurent que chaque requête (obligatoire ou optionnelle) acceptée est affectée à une ligne d'affectation. Les contraintes (1.3) assurent que la livraison du fret d'une requête de transport acceptée est effectuée exactement une fois. La contrainte (1.4) ajuste la variable Y à sa valeur désirée. Les contraintes (1.5) et (1.6) imposent l'intégrité des variables supplémentaires. Les contraintes (1.7)-(1.9) imposent la conservation de flot ainsi que l'envoi d'une unité de flot de l'origine à la destination dans chaque réseau G^k. La compatibilité entre les variables de flot et de ressource est assurée par les contraintes (1.10). Le respect des fenêtres de ressource est assuré par les contraintes (1.11). Finalement, les contraintes (1.12) imposent l'intégrité des variables de flot.

La fonction $f^{kr}_{ij}(T^k_i)$, appelée fonction d'extension de ressource, est utilisée pour calculer une borne inférieure sur la variable T^{kr}_j lorsque l'arc $(i,j) \in A^k$ est utilisé dans la solution. Les fonctions d'extension de ressources (voir Rancourt (1998) pour la définition) ont la propriété de dépendre uniquement de T^k_i et d'être non décroissante. Ces propriétés permettent de faciliter le développement d'un algorithme efficace pour la résolution du modèle.

Modèle de partitionnement d'ensemble

Cette formulation fait intervenir de nouvelles variables qui sont les variables de chemins. On définit Ω^k l'ensemble des chemins réalisables du réseau G^k, pour tout $k \in K$. On note θ_p^k la variable associée à un chemin $p \in \Omega^k$, pour tout $k \in K$, telle que :

$$\theta_p^k = \begin{cases} 1 & \text{si le chemin } p \text{ est utilisé dans la solution} \\ 0 & \text{sinon} \end{cases}$$

De plus on définit :

- pour tout $p \in \Omega^k$, c_p^k le coût du chemin p, qui est la somme des coûts des arcs le composant.
- pour tout $k \in K$, $p \in \Omega^k$ et $t \in T$, a_p^{kt} la constante égale à 1 si la requête t est couverte par le chemin p et 0 sinon.
- pour tout $k \in K$, $p \in \Omega^k$ et $t \in T^F$, d_p^{kt} la constante égale à 1 si la livraison du fret de la requête de transport t est couverte par le chemin p et 0 sinon.

On retrouve également les variables supplémentaires $Z_o, \forall o \in O$ et Y définies pour le modèle précédent.

La formulation est alors :

$$\min \sum_{k \in K} \sum_{p \in \Omega^k} c_p^k \theta_p^k + \sum_{o \in O} c_o Z_o \tag{1.13}$$

s.c :

$$\sum_{k \in K} \sum_{p \in \Omega^k} a_p^{kt} \theta_p^k + \sum_{o \in O} b_o^t Z_o = 1, \qquad \forall t \in T \tag{1.14}$$

$$\sum_{k \in K} \sum_{p \in \Omega^k} d_p^{kt} \theta_p^k + \sum_{o \in O} b_o^t Z_o = 1, \qquad \forall t \in T^F \tag{1.15}$$

$$\sum_{o \in O} Z_o - Y = 0, \tag{1.16}$$

$$Y \quad \text{entier}, \tag{1.17}$$

$$Z_o \in \{0,1\}, \qquad \forall o \in O \tag{1.18}$$

$$\sum_{p \in \Omega^k} \theta_p^k = 1, \qquad \forall k \in K \tag{1.19}$$

$$\theta_p^k \in \{0,1\}, \qquad \forall k \in K, \forall p \in \Omega^k. \tag{1.20}$$

Comme pour la formulation multiflots, la fonction objectif (1.13), cherche à minimiser la somme des coûts des lignes d'affectation et la somme pondérée des requêtes optionnelles rejetées. Les contraintes (1.14) assurent que chaque requête (obligatoire ou optionnelle) acceptée est couverte par exactement une ligne d'affectation. Les contraintes (1.15) assurent que la livraison du fret d'une requête de transport acceptée est effectuée exactement une fois. La contrainte (1.16) ajuste la variable Y à sa valeur désirée. L'intégrité des variables supplémentaires est assurée par les contraintes (1.17) et (1.18). Les contraintes (1.19) imposent qu'au plus un chemin soit choisi pour chaque réseau. Finalement, les contraintes (1.20) imposent l'intégrité des variables de chemin.
Dans le cas où plusieurs commodités sont identiques, on peut appliquer le principe d'agrégation afin de réduire le nombre de variables de chemin et de contraintes, ce qui a été fait dans le cas de ce modèle.

1.3.2 Modélisation proposée par Guigue(2000)

Comme nous l'avons déjà mentionné, dans l'approche de Guigue (2000), on considére des missions construites *a priori* et les arcs des réseaux représentent alors des missions. La construction de telles missions permet de prendre en compte explicitement les contraintes liées à la livraison du fret des requêtes de transport qui sont les contraintes (1.3) du modèle de Rancourt (1998). Elles ne vont donc pas apparaître dans les modèles ci-dessous.
Pour ces modèles on associe un coût c_{ij}^k à chaque arc $(i,j) \in A^k$, correspondant au coût des missions.

Modèle de multiflots

Comme pour les modèles précédents on définit les constantes et les variables nécessaires à l'écriture du modèle.
Pour chaque commodité $k \in K$, chaque arc $(i,j) \in A^k$ et chaque requête $t \in T$, soit la constante a_{ij}^{kt}, telle que :

$$a_{ij}^{kt} = \begin{cases} 1 & \text{si et seulement si l'arc couvre la requête.} \\ 0 & \textit{sinon.} \end{cases}$$

Pour chaque requête $t \in T$, la constante b_o^t se définit comme suit :

$$b_o^t = \begin{cases} 1 & \text{si la requête } t \in T \text{ appartient au groupe de requêtes optionnelles } o \in O, \\ 0 & \textit{sinon.} \end{cases}$$

Pour chaque réseau $G^k = (V^k, A^k)$, on associe $\forall (i,j) \in A^k$ une variable binaire de flot X_{ij}^k, telle que :

$$X_{ij}^k = \begin{cases} 1 & \text{si l'arc } (i,j) \text{ est utilisé par la commodité } k \\ 0 & \text{sinon.} \end{cases}$$

On note T^k l'unique ressource qui représente un instant de l'horizon temporel. $T_i^k, \forall i \in V^k, \forall k \in K$, indique la valeur de la ressource au nœud i si ce nœud est visité par la commodité k.

Il faut également ajouter les variables binaires $Z_o, \forall o \in O$ pour les groupes de requêtes optionnelles, qui prennent les valeurs :

$$Z_o = \begin{cases} 1 & \text{si toutes les requêtes du groupe de requêtes optionnelles } o \text{ sont rejetées,} \\ 0 & \text{si toutes les requêtes du groupe de requêtes optionnelles } o \text{ sont acceptées.} \end{cases}$$

Ainsi que la variable entière Y qui compte le nombre de groupes de requêtes optionnelles dont les requêtes sont rejetées.

La pénalité associée au rejet des requêtes du groupe de requêtes optionnelles o, est notée c_o

Le modèle du multiflots prend alors la forme suivante :

$$\min \sum_{k\in K}\sum_{(i,j)\in A^k} c_{ij}^k X_{ij}^k + \sum_{o\in O} c_o Z_o \tag{1.21}$$

s.c :

$$\sum_{k\in K}\sum_{(i,j)\in A^k} a_{ij}^{kt} X_{ij}^k + \sum_{o\in O} b_o^t Z_o = 1, \qquad \forall t \in T \tag{1.22}$$

$$\sum_{o\in O} Z_o - Y = 0, \tag{1.23}$$

$$Y \quad \text{entier}, \tag{1.24}$$

$$Z_o \in \{0,1\}, \qquad \forall o \in O \tag{1.25}$$

$$\sum_{j:(o(k),j)\in A^k} X_{o(k),j} = 1, \qquad \forall k \in K \tag{1.26}$$

$$\sum_{j:(j,d(k))\in A^k} X_{j,d(k)} = 1, \qquad \forall k \in K \tag{1.27}$$

$$\sum_{j:(j,i)\in A^k} X_{ji}^k - \sum_{j:(i,j)\in A^k} X_{ij}^k = 0, \qquad \forall k \in K, \forall i \in V^k \backslash \{o(k), d(k)\} \tag{1.28}$$

$$X_{ij}^k (f_{ij}^k(T_i^k) - T_j^k) \leq 0, \qquad \forall k \in K, \forall (i,j) \in A^k \tag{1.29}$$

$$a_i^k \leq T_i^k \leq b_i^k, \qquad \forall k \in K, \forall i \in V^k \tag{1.30}$$

$$X_{ij}^k \in \{0,1\}, \qquad \forall k \in K, \forall (i,j) \in A^k. \tag{1.31}$$

La fonction objectif (1.21) est la même que celle des deux modèles précédents. Tout comme pour les modèles de Rancourt (1998), les contraintes (1.22) assurent que chaque requête (obligatoire ou optionnelle) acceptée est affectée à une ligne d'affectation. La contrainte (1.23) ajuste la variable Y à sa valeur désirée. Les contraintes (1.24),

(1.25) et (1.31) imposent l'intégrité des variables. Les contraintes (1.26)-(1.28) imposent la conservation de flot ainsi que l'envoi d'une unité de flot de l'origine à la destination dans chaque réseau G^k. La compatibilité entre les variables de flot et de ressource est assurée par les contraintes (1.29). Ici la fonction d'extension f_{ij}^k est définie par :

$$f_{ij}^k(T_i^k) = T_i^k + d_{ij}^k \quad \forall k, \forall (i,j) \in A^k \tag{1.32}$$

où d_{ij}^k est la durée de l'arc (i,j) pour la commodité k. Le respect des fenêtres de ressource est assuré par les contraintes (1.30).

En plus de la suppression des contraintes (1.3) du modèle de Rancourt (1998), on note une diminution du nombre de ressources puisque dans le modèle de Guigue (2000), on en compte seulement une.

Comme on peut le voir sur le schéma de la figure 1.1 pour l'approche de Guigue (2000), il y a une phase de génération de missions avant la phase d'optimisation qui ne se trouve pas dans l'approche de Rancourt (1998). Cette génération permet la suppression de contraintes liées à la livraison de fret, du problème maître.

Modèle de patitionnement d'ensemble

La définition des variables θ_p^k, Z_o et Y ainsi que la constante a_p^{kt} et le coût c_p^k sont identiques à celles de Rancourt(1998).

La formulation est alors :

$$\min \sum_{k \in K} \sum_{p \in \Omega^k} c_p^k \theta_p^k + \sum_{o \in O} c_o Z_o \tag{1.33}$$

s.c :

$$\sum_{k \in K} \sum_{p \in \Omega^k} a_p^{kt} \theta_p^k + \sum_{o \in O} b_o^t Z_o = 1, \qquad \forall t \in T \tag{1.34}$$

$$\sum_{o \in O} Z_o - Y = 0, \tag{1.35}$$

$$Y \quad \text{entier}, \tag{1.36}$$

$$Z_o \in \{0,1\}, \qquad \forall o \in O \tag{1.37}$$

$$\sum_{p \in \Omega^k} \theta_p^k = 1, \qquad \forall k \in K \tag{1.38}$$

$$\theta_p^k \in \{0,1\}, \qquad \forall k \in K,\ \forall p \in \Omega^k. \tag{1.39}$$

Comme pour la formulation multiflots, la fonction objectif (1.33), cherche à minimiser la somme des coûts des lignes d'affectation et la somme pondérée des requêtes optionnelles rejetées. Les contraintes (1.34) assurent que chaque requête (obligatoire ou optionnelle) acceptée est couverte par exactement une ligne d'affectation. La contrainte (1.35) ajuste la variable Y à sa valeur désirée. L'intégrité des variables supplémentaires est assurée par les contraintes (1.36) et (1.37). Les contraintes (1.38) imposent qu'au plus un chemin soit choisi pour chaque réseau. Finalement, les contraintes (1.39) imposent l'intégrité des variables de chemin.

On note une fois de plus que les contraintes (1.15) du modèle de Rancourt(1998) n'apparaîssent pas ici et cela pour la même raison que celle citée plus haut.

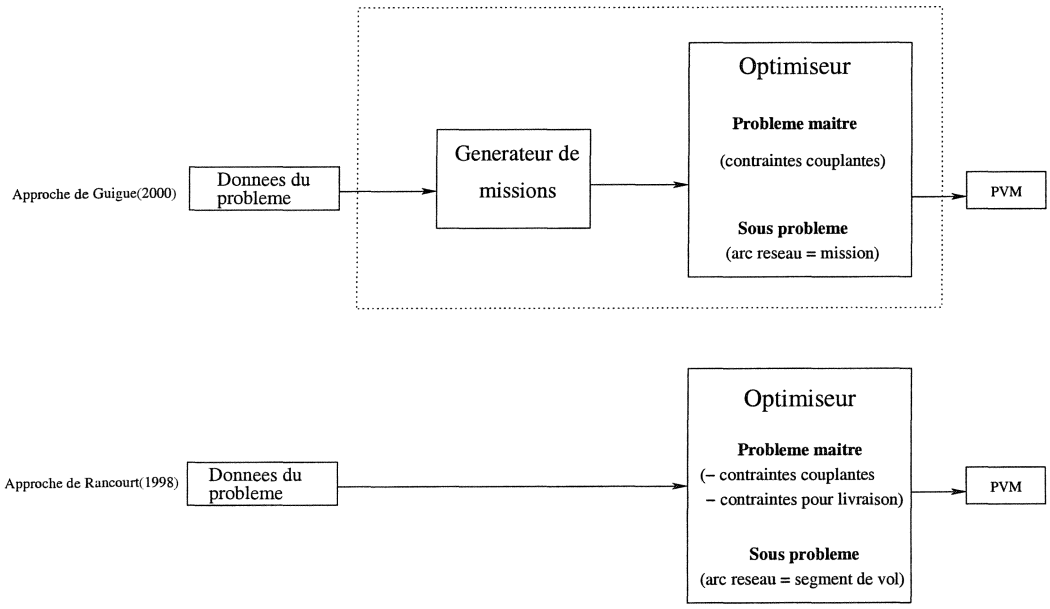

Figure 1.1 – Comparaison des approches de Rancourt (1998) et Guigue (2000)

1.4 Contribution et organisation du mémoire

L'approche de Rancourt (1998) présentée plus haut a l'inconvénient d'être complexe d'un point de vue de la modélisation des réseaux et du nombre de ressources nécessaires à la modélisation de certaines contraintes. Cette apporche quoique très détaillée ne permet pas d'ajouter facilement des contraintes opérationnelles telles que l'interdiction de la présence de deux requêtes dans une même mission. L'approche de Guigue (2000) par contre a l'avantage de réduire la taille du problème maître et de réduire le nombre de ressources au niveau des sous-problèmes et permet une certaine flexibilité quant à l'ajout de contraintes au problème, mais possède l'inconvénient d'avoir des réseaux de grande taille (grand nombre de nœuds et d'arcs) du fait que toutes les missions sont énumérées (en théorie). Evidemment, en pratique, seul un sous-ensemble des missions possibles sont générées *a priori*. D'où l'intérêt de notre étude dont le but est, dans un premier temps, de prendre en compte de façon implicite des contraintes du problème qui jusqu'à présent étaient considérées de manière explicite dans les modèles précédents et qui interviennent au niveau de la génération des missions. Et dans un deuxième temps, de classer, selon certains critères, les missions qui auront été générées dans le but d'en garder qu'un nombre restreint, i.e. celles qui sont susceptibles de se trouver dans la meilleure solution.

Dans le chapitre 2 nous allons décrire le module de l'optimiseur developpé pour générer les missions. Le chapitre 3 explique les changements apportés au générateur de missions. Dans le chapitre 4, il sera question de la deuxième partie de l'étude à savoir l'analyse et la réduction du nombre des missions générées. Finalement dans le chapitre 5, on retrouve les conclusions de cette étude.

CHAPITRE 2 : GÉNÉRATEUR DE MISSIONS

2.1 Introduction

Comme nous l'avons mentionné dans le chapitre 1, l'approche de Guigue (2000) qui améliore celle de Rancourt (1998), fait intervenir un processus de génération de missions avant la résolution. Dans l'optimiseur développé pour résoudre le problème défini au chapitre précédent, le module qui construit les missionns s'appelle le *générateur de missions.*

Nous allons dans ce chapitre définir le fonctionnement de ce module en commençant par décrire les données qui doivent être fournies en entrée, les résultats obtenus en sortie et finalement le procédé utilisé pour la construction des missions.

2.2 Les données d'entrée et les résultats

Le générateur de missions a besoin d'avoir comme données d'entrée les informations suivantes :

- liste des requêtes de transport, leur aéroport de chargement et de déchargement, leur durée de briefing et débriefing et leur durée de chargement et de déchargement ;
- les types d'avions, avec leur vitesse moyenne et la durée de vol maximale sans ravitaillement ;
- la liste des aéroports et leurs horaires d'ouverture ;
- les fenêtres de temps pour quitter l'aéroport de chargement et pour arriver à l'aéroport de déchargement, pour chaque requête de transport ;
- les temps de vol entre les différents aéroports en fonction du type d'appareil ;

– les combinaisons permises de fret.

En plus des données ci-dessus, durant le processus de résolution, l'utilisateur peut décider d'interdire des segments de vol, on a alors *l'ensemble des legs interdits* qui est également une donnée.

Le format sous lequel se présentent les données est décrit en annexe A.

Comme son nom l'indique le générateur de missions génère des missions. Les données récupérées en sortie sont donc des missions non prédéfinies, ayant chacune un identificateur et pour lesquelles on a l'itinéraire ainsi que les requêtes couvertes.

Le format sous lequel apparaissent les missions générées est donné en annexe A.

2.3 Mode de fonctionnement

Avant d'expliquer le fonctionnement du générateur de missions, nous introduisons quelques notations mathématiques utilisées dans le reste de ce chapitre qui sont détaillées dans les tableaux 2.1 et 2.2.

Tableau 2.1 – Notations pour les requêtes de transport

$ac(t)$	l'aéroport de chargement pour une requête t
$ad(t)$	l'aéroport de déchargement pour une requête t
b_t^k	la durée d'un briefing pour une requête t
d_t^k	la durée d'un debriefing pour une requête t
gl_t^k	la durée de chargement pour une requête t
gu_t^k	la durée de déchargement pour une requête t
U_t^k	l'ensemble des fenêtres de temps pour quitter $ac(t)$
V_t^k	l'ensemble des fenêtres de temps pour atterrir à $ad(t)$

La construction des missions se fait en deux étapes :

Tableau 2.2 – Notations pour l'équipage

ms_j	la durée maximale d'un service de vol dont le premier segment débute entre 08:00 et 17:59 heure locale
ms_n	la durée maximale d'un service de vol dont le premier segment débute entre 18:00 et 07:59 heure locale
st	le délai minimal entre deux missions affectées à la même ligne d'affectation
b	la durée normale d'un briefing
d	la durée normale d'un debriefing
$rest$	la durée minimale d'un repos entre deux services de vol

- construction de segments de journées de travail .
- génération de missions à partir des segments obtenus de la première étape.

Pour chaque étape, un réseau espace-temps $G = (V, A)$ est construit et dans lequel on cherche des chemins réalisables. Chaque réseau est caractérisé par son ensemble de nœuds V et d'arcs A, sur lesquels on définit des ressources R.

Pour chaque étape, le réseau construit est détaillé dans les sous-sections suivantes et illustré par les figures 2.1 et 2.2.

2.3.1 Description du réseau des journées de travail

1. Les ressources considérées sont :
   - la ressource *serv* : permet de cumuler le temps de travail accompli par l'équipage depuis le début du service de vol jusqu'à un certain nœud pour s'assurer de respecter la contrainte du nombre d'heures maximal dans une journée de travail.
   - la ressource *temps* : permet de représenter l'instant d'arrivée à un nœud. Elle permet de s'assurer que chaque nœud est visité à l'intérieur de sa fenêtre de temps et que chaque mission, chaque collecte ou chaque livraion se fasse dans

la période de temps permise. Elle permet également de respecter la contrainte d'ouverture des aéroports.

- la ressource *frt(t)* définie pour chaque requête de transport t, permet de savoir si le fret d'une requête est chargé ou pas. Cette ressource prend les valeurs 0 ou 1.

De plus, il faut ajouter les ressources qui assurent les contraintes de préséance et de couplage, qui vérifient que le fret d'une requête ne peut être déchargé que s'il a été préalablement chargé. Il existe également un autre type de ressource, dans le cas où l'on souhaite spécifier un nombre minimum ou maximum de requêtes à couvrir par une journée de travail, qui seront notées Min et Max, respectivement.

2. Les nœuds sont groupés en trois grandes classes :
   - Les nœuds *source* et *puits*, notés respectivement o et d.
   - Les nœuds *depart* et *arrivee* pour chaque aéroport correspondant respectivement au décollage et à l'atterissage à l'aéroport. Il y a autant de nœuds *depart* et *arrivee* qu'il y a de fenêtres de temps disjointes pour quitter/atterrir à l'aéroport.
   - Les nœuds *chargement* et *dechargement* pour chaque requête de transport correspondant respectivement à la fin du chargement et au début du déchargement du fret d'une requête. Il y a autant de nœuds *chargement* et *dechargement* qu'il y a de fenêtres de temps disjointes pour quitter/atterrir avec le fret à bord.

   Les fenêtres de ressources $[a_i, b_i]$ associées aux nœuds précédents sont définies dans le tableau 2.3. Ici les fenêtres pour les ressources Min et Max sont données dans le cas où ces deux ressources sont actives.

3. Les arcs. On retrouve les deux types d'arcs suivants :
   - arcs *segm* qui représente un segment de vol ;

Tableau 2.3 – Fenêtres de ressource des nœuds du réseau des journées de travail

i	$[a_i^r, b_i^r]$				
	$r = temps$	$r = serv$	$r = frt(t)$	$r = Min$	$r = Max$
o	$]-\infty, +\infty[$	$[0, 0]$	$[0, 0]$	$[0, 0]$	$[0, 0]$
d	$]-\infty, +\infty[$	$[0, ms_j]$	$[0, 0]$	$[-Min, -Min]$	$[0, Max]$
$arrivee$	(1)	$[0, ms_j]$	$[0, 0]$ ou $[0, 1]$	$[-Min, 0]$	$[0, Max - 1]$
$depart$	(2)	$[0, ms_j]$	$[0, 0]$ ou $[0, 1]$	$[-Min, 0]$	$[0, Max - 1]$
$chargement$	$h, h \in U_q$	$[0, ms_j]$	$[0, 0]$ ou $[0, 1]$	$[-Min, 0]$	$[0, Max - 1]$
$dechargement$	$h, h \in V_q$	$[0, ms_j]$	(3)	$[-Min, 0]$	$[0, Max - 1]$
(1) $h, h \in V_q$ s'il s'agit de $ad(q)$ et $h, h \in U_q$ avancé de gl s'il s'agit de $ac(q)$ (2)$h, h \in U_q$ s'il s'agit de $ac(q)$ et $h, h \in V_q$ reculé de gu s'il s'agit de $ad(q)$ (3) $[1, 1]$ ou $[0, 1]$ ou $[0, 0]$					

– arcs *au sol*

Il faut y ajouter des arcs reliant les nœuds *source* à *chargement* puisque un segment de journée de travail commence par un chargement et d'autres reliant les nœuds *dechargement* à *puits* puisqu'on termine un segment de journée de travail par un déchargement. La consommation des ressources sur ces arcs est nulle.

Les arcs *segm* doivent vérifier les conditions d'existance du tableau 2.4. La consommation de ressources pour ce type d'arcs est donnée par le tableau 2.5. Pour les arcs *au sol*, la consommation de ressources est détaillée dans le tableau 2.6.

Tableau 2.4 – Conditions d'existence des arcs *segm*.

Conditions
$a_i^{temps} + t_{ij}^{temps} \leq b_j^{temps}$
$b_i^{temps} + t_{ij}^{temps} \geq a_j^{temps}$

Tableau 2.5 – Consommation de ressources pour les Arcs $(i, j) \in A$ de type *segm*

i	j	t_{ij}^{temps}	t_{ij}^{serv}
depart	*arrivee*	f_{ij}	f_{ij}

Tableau 2.6 – Consommation de ressources pour les Arcs $(i, j) \in A$ *au sol*

i	j	t_{ij}^{temps}	t_{ij}^{serv}	$frt(t)$
arrivee	*chargement*	gl_q, $q \in T^F$	gl_q, $q \in T^F$	0
arrivee	*dechargement*	0	0	0
chargement	*depart*	0	0	+1
dechargement	*depart*	gu_q, $q \in T^F$	gu_q, $q \in T^F$	−1
chargement	*chargement*	0	0	+1
dechargement	*chargement*	gu_q, $q \in T^F(*)$	gu_q, $q \in T^F(*)$	−1
dechargement	*dechargement*	0	0	−1
(*) q première requête à décharger				

Illustrons la construction de ce réseau par un exemple. Pour simplifier, on considère que l'on a une seule requête de transport avec un aéroport de chargement A et un aéroport de déchargement B. Pour chacun des aéroports, on suppose qu'il y a trois fenêtres de temps pour le quitter ou y atterrir selon le cas. On peut décoller de A durant les intervalles $[8, 9]$, $[12, 13]$, $[18, 19]$, et atterrir en B durant les intervalles $[10, 11]$, $[14, 15]$, $[20, 21]$. On suppose que la durée de vol entre ces deux aéroports est de 2 unités de temps et que les durées de chargement et déchargement sont de $1/2$ unité chacune.

On obtient alors le réseau de la figure 2.1. Ici les différents arcs entre les nœuds chargement et déchargement n'apparaissent pas étant donné que l'on a qu'une seule requête de transport. Dans un réseau complet, on ne verra pas apparaître des nœuds isolés.

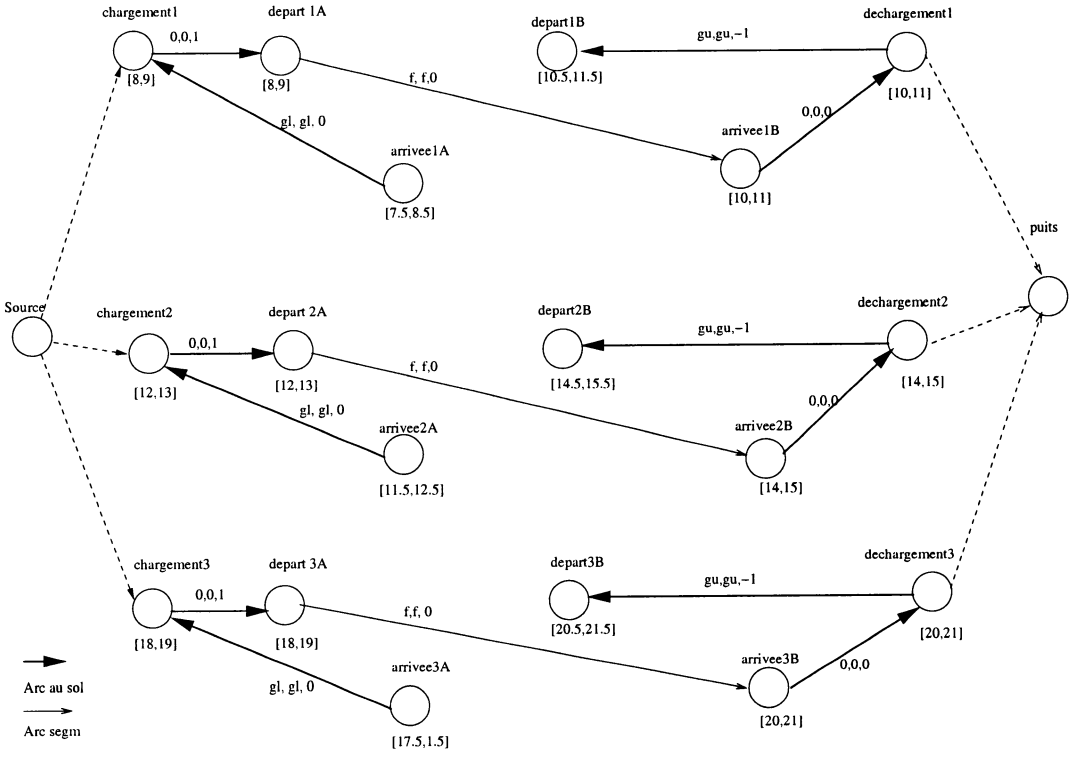

Figure 2.1 – Exemple d'un réseau de journée de travail

2.3.2 Description du réseau des missions

1. Les ressources de ce réseau sont
   - *serv* ;
   - *temps*

   Il est à noter qu'il existe une ressource commune aux deux réseaux qui permet de s'assurer qu'une requête est couverte une fois au plus.
2. Les différents types de nœuds qu'on retrouve dans le réseau se divisent en trois classes.
   - Les nœuds *source* et *puits*, notés respectivement o et d, qui dans certains cas peuvent représenter les bases, ce qui permet de réduire le nombre de nœuds dans le réseau.
   - Les nœuds *depart* et *arrivee* pour chaque aéroport. Il y a autant de nœuds de ce type qu'il y a d'intervalles de temps pour quitter/atterrir à l'aéroport ;
   - Les nœuds *Debutsegment* et *Finsegment*, pour chacun des segments de journée de travail obtenu dans l'étape précédente ;

   Ici et pour le reste de cette section, le terme segment désigne une portion de journée de travail.

   Les fenêtres de ressources associées aux nœuds précédents sont définies dans le tableau 2.7.
3. Les arcs du réseau sont :
   - arcs *segm* ;
   - arcs *au sol*;
   - arcs *repos* ;
   - arcs *segm-jrn* qui repésentent un segment de journée de travail.

   De plus, on a les arcs reliant les nœuds *source* à *Debutsegment* si le premier aéroport du segment de journée est la base et *source* à *arrivee* si le premier aéroport du segment de journée est différent de la base. De même on a des arcs

Tableau 2.7 – Fenêtres de ressource des nœuds du réseau des missions

i	$[a_i^r, b_i^r]$	
	$r = temps$	$r = serv$
o	$]-\infty, +\infty[$	$[0, 0]$
d	$]-\infty, +\infty[$	$[0, ms_j]$
$arrivee$	$h, h \in V_q$	$[0, ms_j]$
$depart$	$h, h \in U_q$	$[0, ms_j]$
$debutsegment$	(1)	$[0, ms_j - ds]$
$finsegment$	(2)	$[0, ms_j]$
ds est la durée du segment de journée (1) intervalle durant lequel le segment peut débuter (2) intervalle durant lequel le segment peut débuter auquel on a ajouté la durée du segment		

des nœuds $Finsegment$ à $puits$ si le dernier aéroport du segment est la base et des nœuds $depart$ aux nœuds $puits$, si le dernier aéroport du segment est différent de la base.

Comme pour le réseau précédent les arcs $segm$ doivent vérifier les conditions d'existance du tableau 2.4. La consommation de ressources pour ce type d'arcs est la même que celle donnée dans le tableau 2.5. Pour le reste des arcs les tableaux 2.8 à 2.11, donnent les consommations de ressource par type d'arcs.

Tableau 2.8 – Consommation de ressources pour les Arcs (i, j) de type $segm - jrn$

i	j	t_{ij}^{temps}	t_{ij}^{serv}
$debutsegment$	$finsegment$	ds	ds

Tableau 2.9 – Consommation de ressources pour les Arcs (i, j) de type *repos*

i	j	t_{ij}^{temps}	t_{ij}^{serv}
$arrivee$	$debutsegment$	$rest$	$-ms_j$
$finsegment$	$depart$	$rest$	$-ms_j$

Tableau 2.10 – Consommation de ressources pour les Arcs (i, j) de type *au sol*

i	j	t_{ij}^{temps}	t_{ij}^{serv}
$arrivee$	$debutsegment$	$\max(gl, a_j^{temps} - a_i^{temps})$	$\max(gl, a_j^{temps} - a_i^{temps})$
$finsegment$	$depart$	$\max(gu, a_j^{temps} - a_i^{temps})$	$\max(gu, a_j^{temps} - a_i^{temps})$

Tableau 2.11 – Consommation de ressources pour les Arcs reliant aux nœuds o et d

i	j	t_{ij}^{temps}	t_{ij}^{serv}
$source$	$debutsegment*$	$\max(0, a_j^{temps} - a_i^{temps})$	$\max(0, a_j^{temps} - a_i^{temps})$
$source$	$arrivee**$	f_{ij}	$\max(f_{ij}, a_j^{temps} - a_i^{temps})$
$finsegment***$	$puits$	$\max(0, a_j^{temps} - a_i^{temps})$	$\max(0, a_j^{temps} - a_i^{temps})$
$depart**$	$puits$	f_{ij}	$\max(f_{ij}, a_j^{temps} - a_i^{temps})$
(*) pour les segments dont le premier aéroport est la base (**) pour les aéroports différents de la base (***) pour les segments dont le dernier aéroport est la base			

Comme pour le réseau précédent, illustrons le tout par un exemple. Supposons qu'on ait trois segments de journées A, B et C et deux bases 1 et 2 tels que :

- le premier aéroport du segment A est la base 1, et son dernier aéroport est un aéroport $aeroAF$;
- le premier aéroport du segment B est un aéroport $aeroBD$, et son dernier aéroport est l'aéroport $aeroBF$;
- le premier aéroport du segment C est un aéroport $aeroCD$, et son dernier aéroport est la base 2 ;

On peut se rendre de $aeroAf$ vers $aeroBD$ et $aeroCD$ et de $aeroBF$ vers $aeroCD$. Pour simplifier l'exemple, on suppose qu'il n'y a qu'une seule fenêtre de temps pour quitter et atterrir aux différents aéroports. On aura alors un nœud *debutsegment* et $finsegment$, pour chacun des segments de journée A, B et C. On aura également un nœud *depart* pour $AeroAF$ et $AeroBF$ (puisqu'il n'y a qu'une seule fenêtre de temps), un nœud *arrivee* pour $aeroBD$ et $aeroCD$ et les nœuds *source* et *puits*, comme le montre la figure 2.2.

Résumé

On peut résumer la construction des missions («générateur de missions» de la figure 1.1) de la façon suivante. Dans un premier réseau avec ressources, on cherche tous les chemins réalisables qui représenteront des segments de journées. Par la suite dans un second réseau avec ressources, où les arcs représenteront les segments de journées obtenus lors de la première étape, on cherche pour chacune des requêtes tous les chemins (missions) qui la couvrent.

On peut limiter le nombre de missions générées par un paramètre que l'on introduira au chapitre 4 et par certaines régles de sélections qui permettent d'éliminer des missions jugées redondantes.

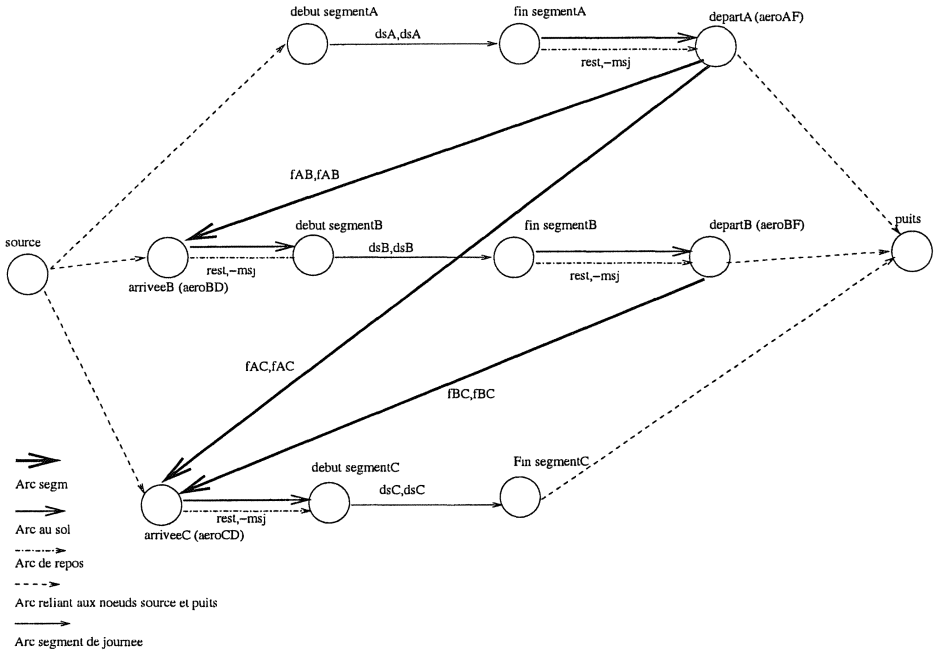

Figure 2.2 – Exemple d'un réseau de missions avec trois segments de journée.

Les missions ainsi générées représenteront les arcs du réseau utilisé dans le sous-problème de fabrication de plans de vol dont la relaxation sera résolue par génération de colonnes (voir «Optimiseur» approche de Guigue(2000) de la figure 1.1).

CHAPITRE 3 : MODIFICATION DU GÉNÉRATEUR DE MISSIONS

3.1 Introduction

Comme nous l'avons mentionné dans le premier chapitre, notre étude consiste dans un premier temps à prendre en compte de manière explicite les contraintes de capacité des appareils et de considérer les incompatibilités de fret (personne et/ou marchandise) plutôt que les compatibilités comme c'est le cas dans le problème étudié dans Guigue (2000) et Rancourt (1998) que nous désignons par *problème original* dans le reste de ce chapitre.

Les compatibilités de fret ainsi que les contraintes de capacités des appareils sont prises en compte de façon explicite dans le problème original de la manière suivante : il existe un *ensemble des requêtes de transport compatibles* qui est une donnée d'entrée du problème, qui contient des combinaisons de requêtes dont le fret est compatible et dont les poids et/ou les volumes respectent la capacité des appareils. Lors de la construction des missions non prédéfinies, si les fenêtres de temps le permettent et qu'on se trouve dans une situation où le fret de deux requêtes peut être transporté simultanément, il faut vérifier si cette combinaison est présente dans l'*ensemble des requêtes de transport compatibles*. Si tel est le cas, on peut les transporter en même temps ; sinon, on ne les met pas à bord du même appareil.

Puisqu' il n'est pas possible de considérer toutes les combinaisons possibles de fret, l'*ensemble des requêtes de transport compatibles* n'est pas complet et de ce fait on pourrait omettre lors de la construction des missions certaines combinaisons qui pourraient être meilleures.

Pour considérer le plus de combinaisons possibles, on remplace l'*ensemble des requêtes de transport compatibles* en introduisant dans le processus de génération des

missions, des contraintes de capacité qui ne concernent que les requêtes de transport. Par ailleurs pour les compatibilités on préfère les remplacer par des incompatibilités du fait que leur nombre est plus petit, ce qui permet de réduire l'aspect combinatoire. Nous allons dans ce chapitre définir les contraintes citées plus haut et expliquer les modifications apportées.

3.2 Définition des contraintes

3.2.1 Capacité des appareils

Les appareils peuvent avoir différentes configurations. Parmi celles-ci, on trouve entre autres :

- configuration cargo ;
- configuration passagers ;
- configuration VIP ;
- configuration cargo et passagers (il y a plusieurs possibilités de combiner les sièges de passagers avec les palettes de fret, principalement pour le HERCULES CC-130)
- configuration recherche et sauvetage (SAR).

En général, le changement de configuration se fait à la base d'attache de l'appareil et on opère rarement des changements de configuration à l'extérieur de la base, entre autres pour des raisons logistiques. Pour le reste de l'étude, nous allons donc considérer que la configuration reste fixe pour toute la durée d'une mission.

Les configurations des appareils ne sont pas fixées *a priori* et nous n'allons pas décider, dans le cadre de cette étude, de la configuration optimale. Nous nous assurons simplement qu'il existe une configuration permettant le transport des requêtes.

Pour ce faire et dans le cas où la requête consiste à transporter des passagers et du fret, pour éviter de considérer plusieurs combinaisons possibles, on va considérer

l'équivalent palette des passagers. Dans le cas qui nous intéresse, le transport des requêtes se fait par le CC-130, dont l'équivalence du nombre de passagers en nombre de palettes est le suivant :

- 31 sièges passagers occupent l'espace de 2 palettes.
- 48 sièges passagers occupent l'espace de 3 palettes.
- 66 sièges passagers occupent l'espace de 4 palettes.

3.2.2 Les incompatibilités

Pour cette contrainte, on considère dans un premier temps, des ensembles ou catégories de fret qui seraient définies par l'utilisateur selon ses normes établies. Par exemple :

- matériel dangereux ;
- matériel perrissable ;
- matériel de taille exceptionnelle ;
- passagers.

Cette classification du fret permet par la suite de définir pour chacune des catégories la liste de toutes les autres qui lui sont incompatibles. Ceci afin de tester par la suite si des requêtes de catégories différentes peuvent être transportées ensembles dans le cas où toutes les autres conditions sont vérifiées.

3.3 Les modifications

Comme seules les missions non prédéfinies sont concernées par l'ajout des nouvelles contraintes, les modifications se feront au niveau du générateur de missions, et plus particulièrement dans le réseau permettant la construction des segments de journée

de travail.
Le générateur de missions modifié aura besoin, en plus des données d'entrée définies au chapitre 2, de données supplémentaires. Quand aux résultats, ils demeurent les mêmes que ceux obtenus par le générateur de missions original, c'est-à-dire des missions non prédéfinies.

3.3.1 Les données d'entrée supplémentaires

Pour les types d'avions, en plus des données concernant le type d'appareil, sa vitesse et la durée maximale de vol sans ravitaillement, on a besoin de connaître sa capacité maximale en terme de palettes, de charge et de nombre de passagers.
Pour les requêtes de transport, on a besoin de connaître le poids, la longueur exprimée en nombre de palettes, le nombre de passagers, la nature du fret à transporter à savoir s'il s'agit de cargo, de passagers ou des deux, et finalement le type ou catégorie du fret.
Il faudra ajouter également deux nouveaux fichiers d'entrée, le premier pour la conversion du nombre de passagers en nombre de palettes par type d'appareil et le second pour la liste des incompatibilités de chacune des catégories de fret. Le format des fichiers d'entrée correspondants se trouve en annexe A.

3.3.2 Prise en charge des nouvelles contraintes

La contrainte de capacité sera modélisée par une ressource. On aura alors le même ensemble de nœuds V et d'arcs A, mais on ajoutera à l'ensemble R, les ressources suivantes :

- $PassCap$ pour vérifier la capacité de l'appareil en terme de passagers ;

- *Length* pour vérifier la capacité de l'appareil en terme de palettes ;
- *Weight* pour ne pas dépasser la charge maximale de l'appareil.

Les fenêtres de ressources correspondantes sont définies dans le tableau 3.1. La consommation des ressources sur les arcs de type *segm* et ceux sortant du nœud *source* est nulle. Le tableau 3.2 résume la consommation des ressources pour les arcs de type *au sol*.

Tableau 3.1 – Fenêtres des nouvelles ressources.

i	$[a_i^r, b_i^r]$		
	$r = Passcap$	$r = Length$	$r = Weight$
o	$[0, 0]$	$[0, 0]$	$[0, 0]$
d	$[0, maxp]$	$[0, maxl]$	$[0, maxw]$
$arrivee$	$[0, maxp]$	$[0, maxl]$	$[0, maxw]$
$depart$	$[0, maxp$	$[0, maxl]$	$[0, maxw]$
$chargement$	$[0, maxp]$	$[0, maxl]$	$[0, maxw]$
$dechargement$	$[0, maxp]$	$[0, maxl]$	$[0, maxw]$
$maxp$ capacité maximale de passagers $maxl$ nombre maximal de palettes $maxw$ charge maximale			

Remarque : *palt* (tableaux 3.2 et 3.3) est le nombre total de palettes de la requête, c'est-à-dire la somme du nombre de palettes du fret cargo et du nombre de palettes obtenu de la conversion du nombre de passagers car dans le fichier de données la longueur de la requête (voir annexe A) correspond uniquement au nombre de palettes du cargo.

Pour les incompatibilités, lorsque se présente le cas où deux ou plusieurs requêtes possédent le même aéroport de chargement avec des fenêtres de temps premettant

Tableau 3.2 – Consommation des nouvelles ressources pour les arcs *au sol*

i	j	$Passcap$	$Length$	$Weight$
arrivee	*chargement*	0	0	0
arrivee	*dechargement*	0	0	0
chargement	*depart*	$+pass$	$+palt$	$+poids$
dechargement	*depart*	$-pass$	$-palt$	$-poids$
chargement	*chargement*	$+pass$	$+palt$	$+poids$
dechargement	*chargement*	$-pass$	$-palt$	$-poids$
dechargement	*dechargement*	$-pass$	$-palt$	$-poids$
pass nombre de passagers de la requête *palt* nombre de palettes du fret de la requête *poids* poids du fret la requête				

Tableau 3.3 – Consommation des nouvelles ressources pour les arcs entre les nœuds *dechargement* et *puits*

i	j	$Passcap$	$Length$	$Weight$
dechargement	*puits*	$-pass$	$-palt$	$-poids$

de les charger à bord de l'appareil et que les contraintes de capacité sont respectées, on vérifie alors si leur fret est compatible. Si tel est le cas, on peut alors les charger en même temps.

3.4 Modifications apportées au code du générateur de missions original

Une liste des fichiers constituant le code original développé en langage C^{++}, ainsi qu'un bref descriptif de chacun d'eux se trouvent en annexe B. Seuls les fichiers concernant le générateur de missions y sont listés.

La construction des réseaux nécessaires à la génération des missions se fait par l'in-

termédiaire du logiciel *NetGen*, développé au GERAD, en 1997 et dont la révision et la mise à jour se font régulièrement.
La recherche des chemins réalisables (segments de journées de travail dans le réseau de la première étape et missions dans la seconde) se fait par le logiciel *GenCol*, lui aussi développé au GERAD depuis le début des années 80.

3.4.1 Modifications engendrées par les nouvelles données

L'ajout de nouvelles données d'entrée entraîne la modification de certains fichiers et la création d'autres.
Les fichiers touchés par la modification des données d'entrée sont :

- *AcsApplicDbClass/AircraftType.h* : on ajoute à l'objet *actype*, les arguments concernant la capacité de l'appreil en terme de passagers, nombre de palettes et charge maximale, ainsi que les méthodes correspondantes (ici les méthodes sont les fonctions membres des objets) ;
- *AcsParsers/AircraftTypeParser.cxx* : on doit permettre la lecture des nouvelles données concernant le type d'avion ;
- *LtpApplicDbClass/AirliftRequest.h* : on ajoute à l'objet *AirliftRequest* les arguments pour prendre en charge le poids de la requête, le nombre de palettes du fret, le nombre de passagers, la nature et le type de la requête ainsi que les méthodes correspondantes ;
- *LtpParsers/AirliftRequestParser.cxx* : on doit permettre la lecture des nouvelles informations.

Les fichiers ajoutés sont :

- *LtpsApplicDbClass/Conversionsr.h et .cxx* qui représentent une classe servant d'interface pour les informations concernant la conversion du nombre de passagers en nombre de palettes en fonction du type d'appareil ;

- *LtpParsers/ConversionsPsgrPalet.h et .cxx* : qui permettent la lecture des données pour l'interface précédente ;
- *LtpApplicDbClass/FretIncomp.h* qui est une classe pour les informations concernant les incompatibilités des différentes catégories de fret ;
- *LtpParsers/IncompCombParser.h et .cxx* qui permettent de lire les données concernant les incompatibilités.

Pour tenir compte de toutes ces nouvelles données dans la base de données de l'application, le fichier *LtpApplicDb/ApplicDbBuilder.cxx* qui permet la construction de la base de données a également été modifié. Et finalement les fichiers *LtpControl/Params.h et .cxx* prennent en compte les nouveaux fichiers.

3.4.2 Modifications engendrées par les nouvelles contraintes

Comme l'ajout de nouvelles ressources se fait au niveau du réseau des segments de journées, les fichiers *LtpOptStrategy/DutySegmGenNetDbBuilder.h* et *LtpOptStrategy/DutySegmGenNetDbBuilder.cxx* qui permettent la construction du réseau ont été modifiés. Il y a également les fichiers *LtpApplicDbClass/RequestCompatibility.h et .cxx* qui sont modifiés du fait que le test de compatibilité entre les requêtes ne se fait plus de la même façon (on considère les incompatibiltés plutôt que les compatibilités). Et finalement comme le test de compatibilité de requêtes ne se fait plus deux à deux mais entre une requête et un ensemble de requêtes, le fichier *LtpOptStrategy/DutySegmGenpathEnum.cxx* a été modifé.

CHAPITRE 4 : ANALYSE ET RÉDUCTION DES MISSIONS GÉNÉRÉES

4.1 Introduction

Plus le nombre de missions générées est grand, plus le temps de résolution est grand. On peut penser alors réduire la taille de l'ensemble des missions générées afin d'obtenir des résultats dans des temps de calcul relativement courts tout en s'assurant d'avoir une bonne solution.

L'objectif de cette deuxième partie du mémoire consiste à réaliser cette réduction de l'ensemble des missions générées de façon à garder les «meilleures »missions au sens de certains critères que l'on va définir plus loin.

4.2 Problématique

Étant donné un ensemble MG de missions générées de cardinalité M, avec M très grand et auquel correspond un temps de résolution assez grand, on souhaite constituer un sous-ensemble MR de cardinalité m, où $m < M$, qui permettrait de résoudre le problème en un temps plus court. La difficulté est de savoir comment déterminer ce sous-ensemble.

Pour tenter de répondre à cette question, nous allons faire une première analyse des missions générées puis nous allons définir des critères qui vont nous permettre de classer les missions pour les sélectionner par la suite.

4.3 Analyse des missions générées : étape 1

Le problème à résoudre correspond aux données du mois d'août 96. Ce mois représente l'un des plus complexes (Rancourt (1998)). Le nombre total de requêtes à couvrir est de 184, dont 77 requêtes de transport. Les tests ont été effectués sur un INTEL PENTIUM 4 CPU 2.66 GHz 1 GB RAM sous Linux. On rappelle que le coût des missions correspond à la somme des temps de vol de ces dernières.

Il existe un paramètre dans le générateur de missions, qu'on notera $NbrMax$, qui permet de gérer le nombre total de missions générées. Plus précisément, ce paramètre représente le nombre de missions que l'on génère pour qu'une requête soit considérée comme couverte. Ainsi, lors du processus de génération, on arrête de générer des missions pour une requête si on n'arrive plus à trouver de chemins réalisables dans le réseau ou si on a atteint la valeur que l'on aura fixée pour le paramètre $NbrMax$. Pour effectuer notre analyse, nous avons fait varier le paramètre $NbrMax$ de 100 à 10 000. Le tableau 4.1 donne la valeur de la relaxation linéaire du problème, les coûts en minutes des différentes solutions, le nombre total de missions générées ainsi que le nombre de missions nécessaires pour couvrir les 184 requêtes. La figure 4.1 illustre la variation du coût de la solution (courbe «Coût de la solution(1)») ainsi que de la valeur de la relaxation linéaire en fonction de $NbrMax$.

En observant les résultats, on constate qu'on améliore la solution quand on passe de 10 912 missions à 14 257 (les deux premières colonnes du tableau). Par la suite le coût de la solution ne change pas jusqu'à ce que le nombre total de missions générées passe à 44 667 ce qui correspond à $NbrMax = 3000$. Ici le coût de la solution augmente de 32 minutes ce qui correspond à une variation de 0.06, on peut donc encore considérer que c'est une bonne solution. Par la suite, on note qu'en passant de 44 667 missions à 55 536 ($nbrMax = 4000$) soit environ 10 000 missions de plus, que le coût de la solution est largement supérieur (presque 10 heures de plus). De nouveau le coût de la solution s'améliore pour $NbrMax = 6000$ et remonte d'environ

Tableau 4.1 – Coût de la solution et nombre total de missions générées en fonction de $NbrMax$.

$NbrMax$	**100**	**500**	**1000**	**1500**	**2000**	**2500**
Relaxation	69622.73	69406.07	69405.97	69399.96	69391.47	69391.61
Coût	69619	69433	69433	69433	69433	69433
Nbr missions générées	10912	14257	19597	24975	31592	38446
Nbr missions solution	133	132	132	132	132	132

$NbrMax$	**3000**	**3500**	**4000**	**6000**	**10000**
Relaxation	69423.58	69430.55	69403.83	69383.76	69249.44
Coût	69465	69484	70055	69437	69511
Nbr missions générées	44667	50742	55536	75302	114848
Nbr missions solution	130	132	133	129	131

0.10% pour $NbrMax = 10000$. Ces résultats plutôt inattendus sont du au fait que la résolution des problèmes utilisent des critères de branchement heuristiques en plus de critères d'arrêt de temps et du nombre de nœuds de branchement.

Maintenant si on observe la valeur de la relaxation linéaire, on peut noter qu'elle diminue quand le nombre de missions générées augmente, ce à quoi on pouvait s'attendre. On note toutefois une légère remontée pour $NbrMax = 3000$, 3500 et 4000 (à l'intérieur des limites des «modèles de résolution» utilisés au sens de GENCOL).

Dans le tableau 4.2, on retrouve les temps de calcul pour la procédure de séparation et évalution (Branch & Bound) en secondes ainsi que le nombre de nœuds explorés dans l'arbre de banchement et la profondeur atteinte dans ce même arbre. Pour ces résultats le paramètre qui contrôle le nombre de nœuds de branchement a été fixé à 1 000 et celui du temps de calcul à 3600 secondes soit 1 heure. On note que les temps de calcul les plus élévés correspondent à $NbrMax = 4000$ et 10 000. D'ailleurs le processus de résolution s'est arrêté car la limite du nombre de nœuds de branchement a été atteinte. Nous avons donc procédé à de nouveaux tests avec une limite de 10 000 nœuds de branchement. Les résultats se trouvent dans le tableau 4.3. Il n'y a pas une grande différences entre les résultats obtenus pour une limite de 1000 et 10 000 nœuds

de branchement si ce n'est que le coût de la solution pour $NbrMax = 10000$ passe de 69511 à 69389 et que le processus de résolution s'arrête à cause de la limite de temps, comme on peut le constater pour $NbrMax = 4000$ et 10000. Pour ces deux valeurs de $NbrMax$, nous avons testé une plus grande limite de temps soit 8h et un nombre de nœuds de branchement égal à 50 000. Pour $NbrMax = 10000$ nous obtenons une solution sans que le processus soit interrompu par le temps ou le nombre de nœuds de 69389 après que 6902 nœuds soient explorés et près de 2 heures de calcul pour la procédure de séparation et évaluation. Pour $NbrMax = 4000$, le processus de résolution s'arrête car il atteint la limite du nombre de nœuds avec un temps de calcul pour la procédure de séparation et évaluation de près de 5h30 et pour une solution de coût égale à 69920. Nous avons alors testé une autre méthode de branchement pour $NbrMax = 4000$ et on obtient une solution de coût égale à 69462 après 118 nœuds de branchement et 227 secondes de temps de calcul. Par contre la stratégie de branchement qui nous permet de trouver ces résultats pour $NbrMax = 4000$, ne donne pas d'aussi bons résultats pour les autres valeurs de $NbrMax$.

La courbe «Coût de la solution(2)»de la figure 4.1 donne la variation du coût de la solution avec cette nouvelle valeur pour $NbrMax = 4000$. Les coûts de la solution pour les autres valeurs de $NbrMax$ qui servent au tracé de cette courbe sont celles obtenues avec la première stratégie de branchement.

Étant donné ces différents résultats, on peut penser que réduire le nombres de missions générées aux meilleures missions pourrait possiblement conduire à de meilleures décisions de branchement.

Pour le reste de l'analyse on fixe le nombre de nœuds de branchement à 1 000 puisque dans la majorité des cas on n'améliore pas le coût de la solution quand on a un nombre de nœuds plus élevé.

Tableau 4.2 – Temps de calcul de la procédure de Branch & Bound pour 1000 nœuds de banchement.

	100	**500**	**1000**	**1500**	**2000**	**2500**	**3000**	**3500**	**4000**	**6000**	**10000**
Temps (s)	4	5.7	8.5	9.3	33.2	18.5	12.6	69.2	414.5	120	945.1
Nœuds	6	16	16	18	84	36	16	112	999	146	999
Profond.	3	5	5	5	13	7	5	13	40	12	39

Tableau 4.3 – Temps de calcul de la procédure de Branch & Bound pour 10 000 nœuds de banchement.

	100	**500**	**1000**	**1500**	**2000**	**2500**	**3000**	**3500**	**4000**	**6000**	**10000**
Temps (s)	3.5	5.7	7.5	33	38.6	18.1	12.3	67.8	3600.5	117.1	3600.3
Nœuds	6	16	20	84	104	36	16	112	9869	146	3302
Profond.	3	5	6	10	13	7	5	13	49	12	33

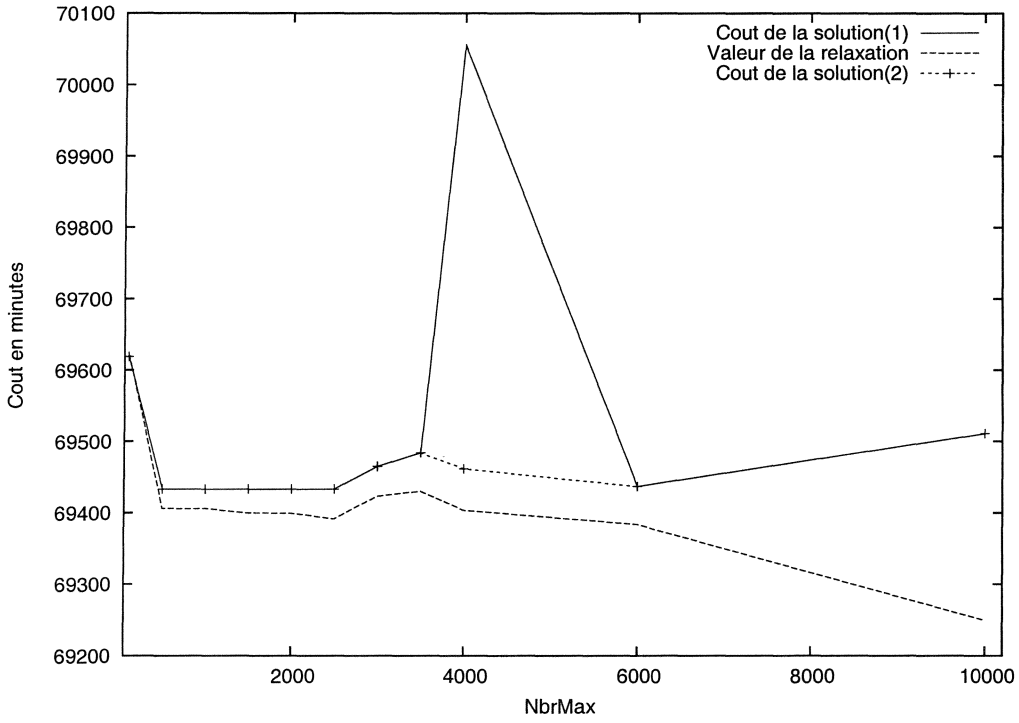

Figure 4.1 – Variation du coût de la solution et de la valeur de la relaxation en fonction de $NbrMax$.

4.4 Analyse des missions générées : étape 2

Puisque les coûts d'opération sont liés à la durée des missions, il pourrait être intéressant dans un premier temps de classifier les missions en fonction de leur durée en nombre de journées de travail afin de voir la contribution de chaque type de missions dans la solution. Un premier critère pourrait être établi sur la base de cette classification.

On regarde d'abord le nombre de missions de chaque type en faisant varier $NbMax$ de 100 à 10 000. Le tableau 4.4 résume ces différentes valeurs. Le tableau 4.5 présente les mêmes résultats exprimés cette fois ci en pourcentages. Les figures 4.2 à 4.5 illustrent ces résultats. Pour des raisons de clarté, les données servant au tracé des courbes de ces figures ont été séparées en deux groupes soit les missions dont la durée varie de 1 à 4 journées et celles dont la durée varie de 5 à 8 journées. Les missions de 9 journées ne sont pas représentées dans les figures étant donné qu'elles ne sont présentes que pour $NbrMax = 10000$. Nous avons également utilisé des échelles différentes pour chacun des groupes, pour que le tracé des courbes soit plus lisible.

Tableau 4.4 – Nombre de missions par type en fonction de $NbrMax$.

	100	**500**	**1000**	**1500**	**2000**	**2500**	**3000**	**3500**	**4000**	**6000**	**10000**
$1jrn$	1690	1712	1737	1737	1757	1767	1789	1794	1812	1856	2034
$2jrn$	4022	4258	4532	4594	4828	4883	5168	5479	5816	7140	10098
$3jrn$	4700	5856	7838	9919	12281	14452	16588	18735	20677	27658	41305
$4jrn$	323	1129	2852	4736	7425	10502	13202	15799	18114	27418	45698
$5jrn$	108	696	1591	2645	3594	4638	5493	6133	6291	7474	10148
$6jrn$	67	495	867	950	1249	1578	1725	2015	2102	2599	3474
$7jrn$	2	111	172	352	392	546	620	691	625	973	1571
$8jrn$	0	0	8	42	66	80	82	96	99	184	471
$9jrn$	0	0	0	0	0	0	0	0	0	0	49

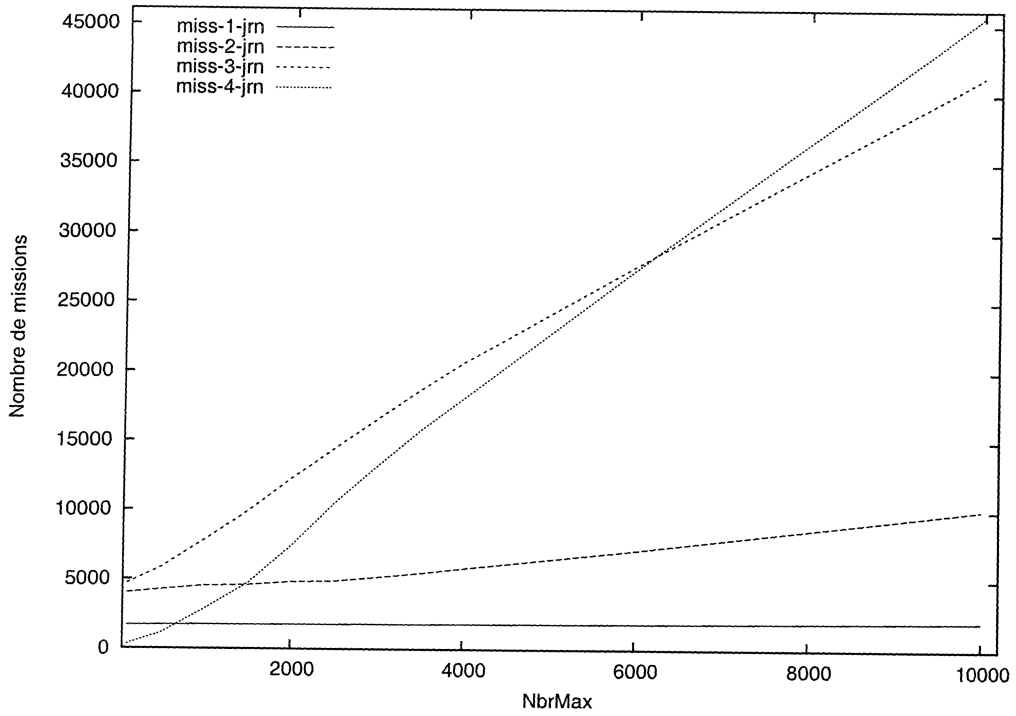

Figure 4.2 – Variation du nombre des missions générées dont la durée est 1 à 4 journées.

On observe également comment se répartissent les différents types de missions dans la solution. Le tableau 4.6 en donne un aperçu et la figure 4.6 l'illustre.

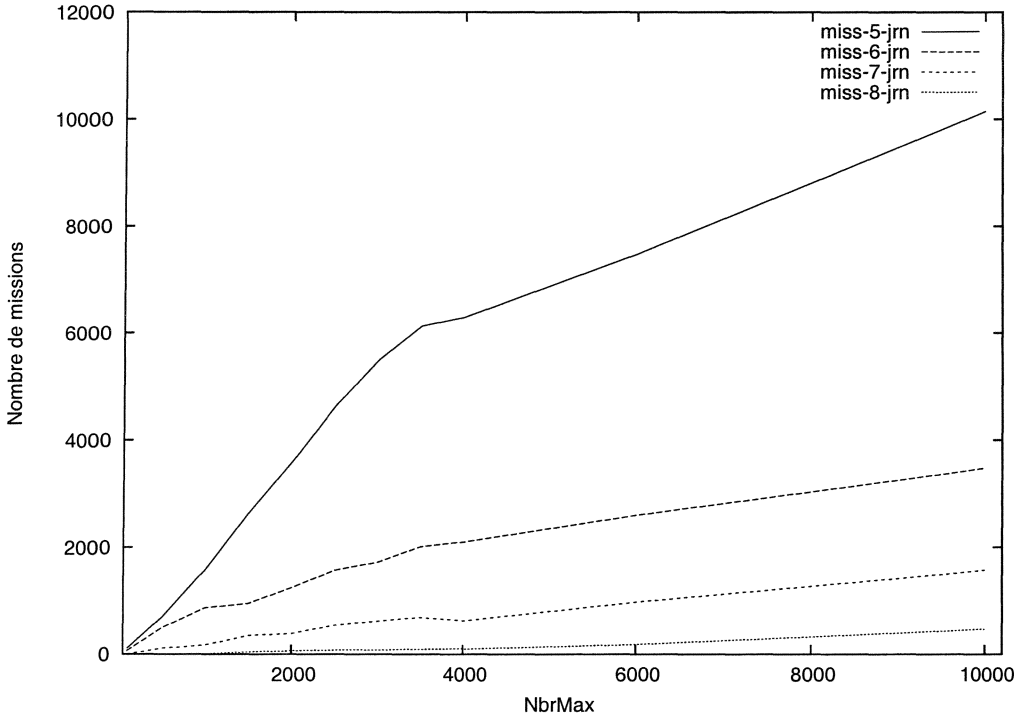

Figure 4.3 – Variation du nombre des missions générées dont la durée est 5 à 8 journées.

Tableau 4.5 – Pourcentage du nombre de missions par type.

	100	**500**	**1000**	**1500**	**2000**	**2500**	**3000**	**3500**	**4000**	**6000**	**10000**
$1jrn$	15.49	12.01	8.86	6.96	5.56	4.60	4.00	3.54	3.26	2.46	1.77
$2jrn$	36.86	29.87	23.13	18.39	15.28	12.70	11.57	10.80	10.47	9.49	8.90
$3jrn$	43.07	41.07	40.00	39.71	38.88	37.60	37.14	36.92	37.23	36.73	35.96
$4jrn$	2.96	7.92	14.55	18.96	23.50	27.31	29.56	31.13	32.62	36.41	39.79
$5jrn$	0.99	4.88	8.12	10.60	11.38	12.06	12.30	12.09	11.33	9.93	8.84
$6jrn$	0.61	3.47	4.42	3.80	3.95	4.10	3.86	3.97	3.78	3.45	3.02
$7jrn$	0.02	0.78	0.88	1.41	1.24	1.42	1.39	1.36	1.13	1.29	1.37
$8jrn$	0	0	0.04	0.17	0.21	0.21	0.18	0.19	0.18	0.24	0.41
$9jrn$	0	0	0	0	0	0	0	0	0	0	0.04

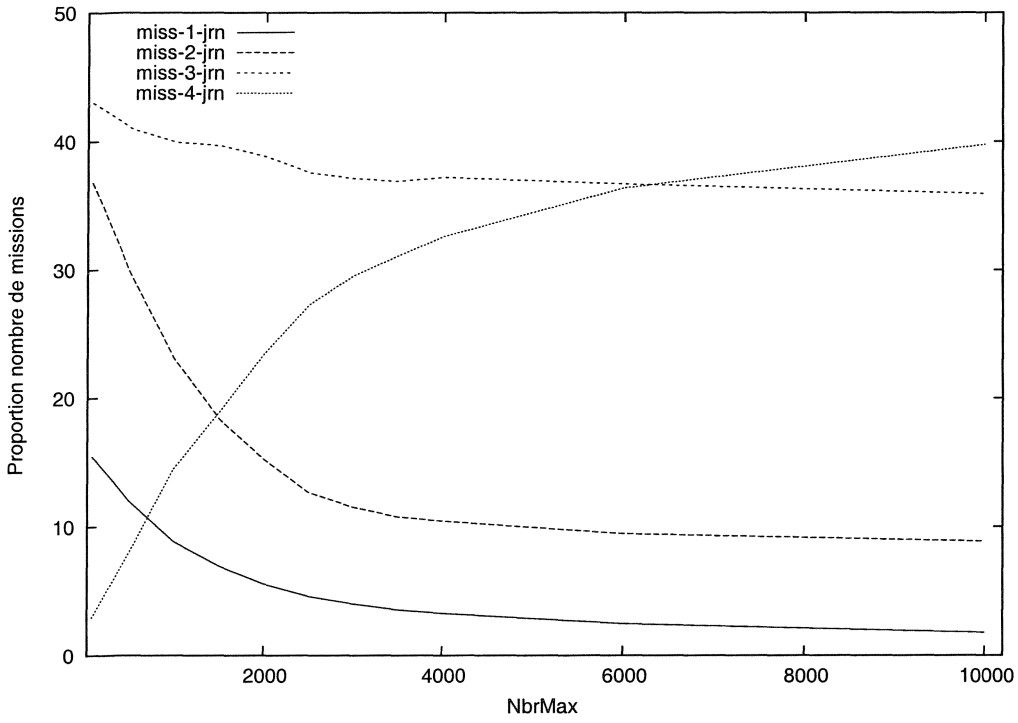

Figure 4.4 – Proportion des missions de 1 à 4 journées.

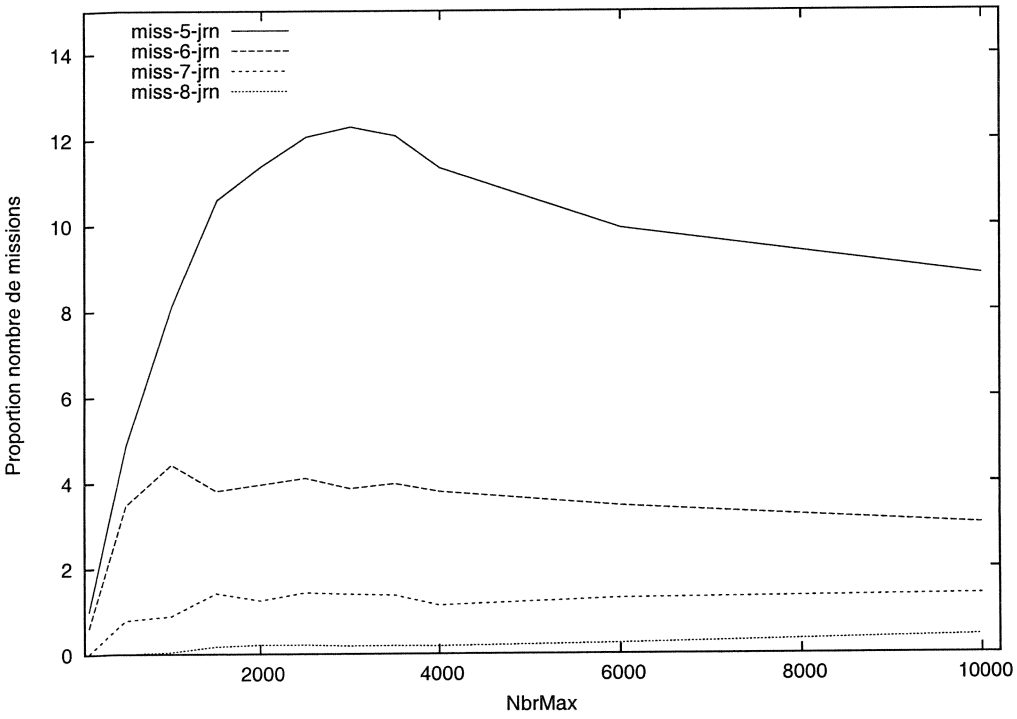

Figure 4.5 – Proportion des missions de 5 à 8 journées.

Tableau 4.6 – Composition de la solution.

	100	500	1000	1500	2000	2500	3000	3500	4000	6000	10000
$1jrn$	10	10	11	10	11	9	7	9	11	6	10
$2jrn$	5	4	4	4	4	4	5	4	7	5	7
$3jrn$	5	5	5	5	5	5	3	5	3	3	2
$4jrn$	1	1	1	1	1	1	1	1	1	2	1
$5jrn$	0	0	0	0	0	0	0	0	0	0	0
$6jrn$	0	0	0	0	0	0	0	0	0	0	0
$7jrn$	0	0	0	0	0	0	0	0	0	0	0
$8jrn$	0	0	0	0	0	0	0	0	0	0	0
$9jrn$	0	0	0	0	0	0	0	0	0	0	0
Total	**21**	**20**	**21**	**20**	**21**	**19**	**16**	**19**	**22**	**16**	**20**

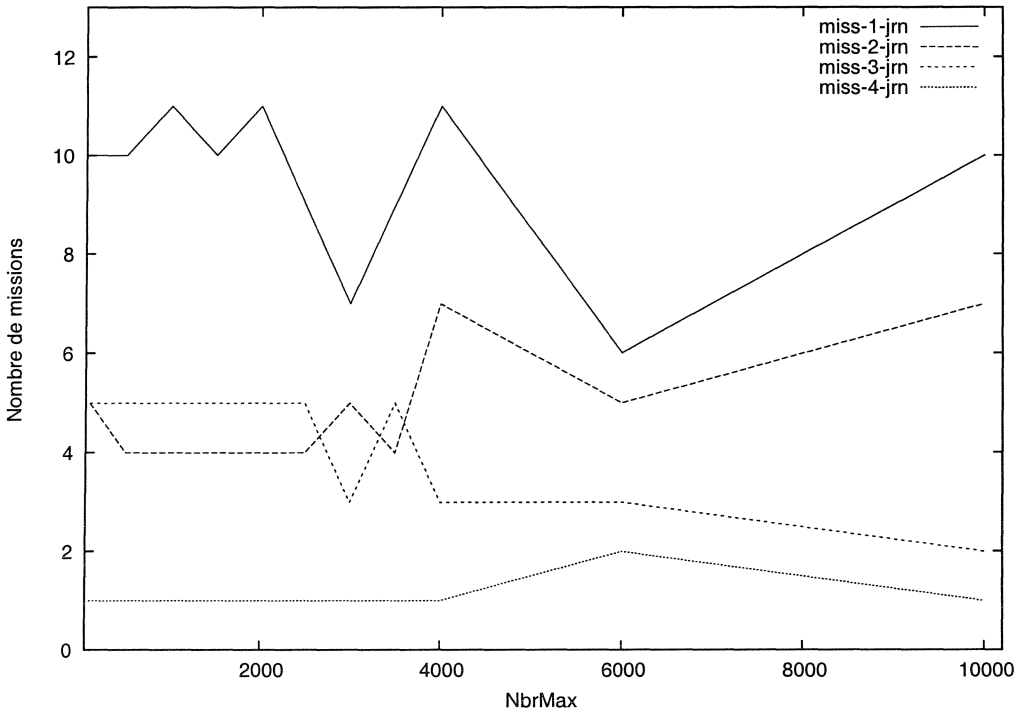

Figure 4.6 – Nombre de missions des différents types dans la solution.

Une première observation est que le nombre de missions dont la durée est d'une journée de travail est quasi-constant et donc plus le nombre de missions générées augmente plus le pourcentage de ce type de missions diminue comme on peut le constater dans le tableau 4.5. De plus si on regarde le tableau 4.6, on note que les missions d'une journée représentent le plus grand nombre dans la solution. On a donc tout interêt à garder toutes les missions d'une journée.
Une seconde observation est que les missions de cinq (5) journées de travail et plus ne font pas partie de la solution (voir tableau 4.6). On peut donc penser à les éliminer, ce qui réduirait déjà l'ensemble des missions à considérer. Différents tests ont été effectués en supprimant les missions de 5 journées et plus de l'ensemble de toutes les missions générées et les solutions trouvées sont les mêmes qu'avec l'ensemble des missions au complet. Mais comme leur nombre ne représente qu'environ 16% des missions totales, leur suppression ne réduit pas l'ensemble de manière significative.
Il est important de noter toutefois que, pour les données servant à notre analyse toutes les missions ont été effectuées sur le territoire canadien. Si des missions à l'étanger étaient prévues, il se peut alors que leur durée soit de l'ordre de 5 jours ou plus. Dans ce cas, il serait peut être nécessaire d'en garder un certain nombre.

4.5 Analyse des missions générées : étape 3

À partir des deux observations de la section précédente, on garde toutes les missions d'une journées de travail dans notre sous-ensemble MR et on ne garde aucune missions de 5 journées de travail et plus. Pour le reste des missions, soit les missions de 2, 3 et 4 journées de travail, on introduit un second critère qui est la proportion du temps de vol de la mission par rapport à sa durée totale que l'on notera *prod*. En effet, on peut s'attendre à ce qu'une mission ayant un temps de vol de 12 heures pour une durée total de 14 heures où la valeur de *prod* est d'environ 85 soit une mission

qui a plus de chance de faire partie de la solution qu'une mission de 7 heures de vol pour une durée totale de 170 heures où *prod* est d'environ 4.
On classe alors les missions de 2, 3 et 4 journées de travail selon la valeur du critère *prod* de la plus élevée à la plus faible. On retrouve dans le tableau 4.7, les valeurs min et max que prend le critère *prod* pour chaque type de missions quand $NbrMax$ varie. On note qu'à partir de $NbrMax = 1000$ les valeurs minimale et maximale de *prod* ne changent pas quand $NbrMax$ augmente. Nous nous sommes donc contentés de reproduire les résultats pour $NbrMax$ varient de 100 à 2000 uniquement.

Tableau 4.7 – Valeur max et min de *prod* pour chaque type de missions

$NbrMax$	**100**		**500**		**1000**		**1500**		**2000**	
$Prod$	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
$2jrn$	10	58	10	58	10	58	10	58	10	58
$3jrn$	4	45	2	45	2	45	2	45	2	45
$4jrn$	10	38	4	39	4	43	4	43	4	43

Pour réduire le nombre de missions on peut penser à ne garder que celles qui auraient une valeur pour le critère *prod* supérieure à un certain seuil. Deux valeurs de seuil différentes ont été testées. Une première valeur étant pour chaque type de missions t, t=2,3,4, la moyenne entre la plus grande valeur de *prod* et la plus faible i.e. $Seuil(t)_1 = (prod(t)_{max} + prod(t)_{min})/2$. La seconde valeur étant la moyenne entre $Seuil(t)_1$ et la valeur minimale de $prod(t)$ i.e. $Seuil_2 = (Seuil(t)_1 + prod(t)_{min})/2$.
Nous avons limité $NbrNax$ à 4 000 et nous n'avons gardé que quelques valeurs intermédiaires pour tester ces seuils. Les résultats obtenus sont donnés dans le tableau 4.8.

En observant les données du tableau 4.8, on peut constater qu'on détériore la qualité de la solution en réduisant l'ensemble des missions de cette manière sauf peut

Tableau 4.8 – Coût de la solution et nombre de missions pour seuils différents.

	Ensemble complet			$Seuil_1$			$Seuil_2$		
$NbrMax$	Coût	Total	Final	Coût	Total	Final	Coût	Total	Final
100	69619	10912	133	70102	5472	136	69956	10233	134
500	69433	14257	132	69996	6423	136	69852	11716	134
1000	69433	19597	132	69996	7145	136	69789	13826	134
3000	69465	44667	130	69996	10329	136	69770	22765	132
4000	70055	55536	133	69974	11817	134	69767	27006	132

être pour $NbrMax = 4000$ où on peut constater une amélioration pour $Seuil_2$ (voir remarque). On note également que $Seuil_2$ donne de meilleurs résultats comparativement à $Seuil_1$. On peut donc raisonnablement penser que les sous-ensembles obtenus pour $Seuil_1$ sont beaucoup trop réduits et que si on veut obtenir la meilleure solution ou tout du moins une bonne solution, il faut un nombre minimum de missions, sinon la réduction se fait au détriment de la qualité de la solution.
Tout ceci nous amène à la quatrième et dernière étape de l'analyse qui va être détaillée dans la section suivante.

4.6 Analyse des missions générées : étape 4 et fin

Les résultats des sections précédentes nous laissent penser qu'il faudrait envisager de définir les seuils différemment pour chaque type de missions. Les figures 4.7 à 4.9 nous donnent un aperçu du nombre de missions générées pour chacune des valeurs de $prod$ dans l'intervalle $[prod(t)_{min}, prod(t)_{max}]$ avec t=2,3,4, pour les missions de 2, 3 et 4 journées de travail réspectivement, quand $NbrMax$ varie. Nous avons représenté sur ces figures que les courbes pour certaines valeurs de $NbrMax$ pour plus de clarté. Le reste des courbes présentent le même profil.
Pour les missions de 2 journées de travail, on note que les courbes se situent sous la

barre des 1000 comme on peut le voir sur la figure 4.7, contrairement aux figures 4.8 et 4.9. De plus elles présentent un relief assez bas, ce qui pourrait motiver le choix de toutes les garder dans le sous ensemble que l'on souhaite construire. Des tests effectués en prenant les missions de 2 journées de travail au complet donne de meilleurs résultats.

Pour les missions de 3 journées de travail, on a pu noter à travers les différents tests que la plus petite valeur de *prod*, pour les missions se trouvant dans la solution, est 9. Si on observe la figure 4.8, on note que les courbes présentent un pic important pour des valeurs de $prod < 9$, c'est-à-dire qu'il y a une forte croissance du nombre de missions générées avec ces valeurs de *prod*. On note par contre pour les valeurs de $prod > 9$ que la croissance du nombre de missions est proportionnelle quand $NbrMax$ augmente. En moyenne le nombre de missions de 3 journées ayant une valeurs $prod < 9$ représente un quart du nombre total de missions de 3 journées de travail que l'on pourrait supprimer.

Pour les missions de 4 journées de travail, on note également la présence de pics en observant la figure 4.9, ainsi qu'une décroissance après chaque pic plus lente que pour les courbes des missions de 3 journées. Par ailleurs les tests ont montré que la valeur de *prod* pour les missions de 4 journées se trouvant dans la solution est 28 à l'exception de la solution pour $NbrMax = 6000$ où la valeur est de 21. On peut alors fixer le seuil de *prod* pour les missions de 4 journées à 28. On peut noter à partir des courbes de la figure 4.9 que le nombre de missions générées autour de cette valeur de *prod* est de l'ordre de 500. On a alors une très forte diminution du nombre de missions de 4 journées si on ne considére que les missions ayant une valeur $prod > 28$.

Nous avons réduit l'ensemble des missions générées pour $NbrMax = 4000$, 6000 et 10 000, en appliquant ces différents résultats et nous obtenons les données du tableau 4.9. On peut noter que pour $NbrMax$ =4000, le gain est significatif puisqu'on améliore

Tableau 4.9 – Résultats obtenus après réduction de l'ensemble des missions générées.

$NbrMax$	4000		6000		10000	
Réduction	Avant	Après	Avant	Après	Avant	Après
Coût	70055	69760	69437	69451	69511	69631
Temps (B&B)	414.5	25.8	120	62.8	945.1	405
Nbr missions	55536	21790	75302	30428	114848	44544

le coût de la solution en passant de 70055 à 69760 soit près de 5 heures (on avait obtenu un coût de 69920 après 5h30 de calcul et 50 000 nœuds de branchement). On réduit également considérablement le temps de calcul qui passe de 414 secondes à 26 secondes et finalement on réduit l'ensemble des missions considérées de près de 60%. Pour $NbrMax$ =6000, on réduit le temps de calcul de près de la moitié et le nombre de missions de près de 60% aussi. Par contre le coût de la solution est supérieur de 0.02%. On a donc une solution de même qualité en moitié moins de temps et avec moitié moins de missions à prendre en compte.

Pour $NbrMax$ =10000, on réduit aussi de moitié le temps de calcul et de près de 61% l'ensemble des missions. Le coût de la solution est 0.17% supérieur mais là encore il s'agit d'une solution tout à fait acceptable.

Remarque : Comme $NbrMax$ =4000 semble être un cas particulier, nous avons effectué d'autre tests en faisant varier les valeurs du seuil pour *prod* et nous avons trouvé une solution de 69462 après 84 secondes de temps de calcul pour la procédure de séparation et évaluation avec 26 600 missions. Ce qui représente une amélioration de près de 10 heures par rapport à 70055, avec une réduction de 80% du temps de calcul et de d'environ 54% du nombre de missions.

4.7 Conclusions

Cette analyse nous a permis de voir qu'il est possible de réduire l'ensemble des missions générées à un ensemble plus petit tout en retrouvant une bonne solution en des temps de calcul plus courts. Il en ressort également que de choisir de bonne missions peut permettre de prendre de meilleures décisions de branchement comme nous avons pu le monter pour $NbrMax = 4000$.

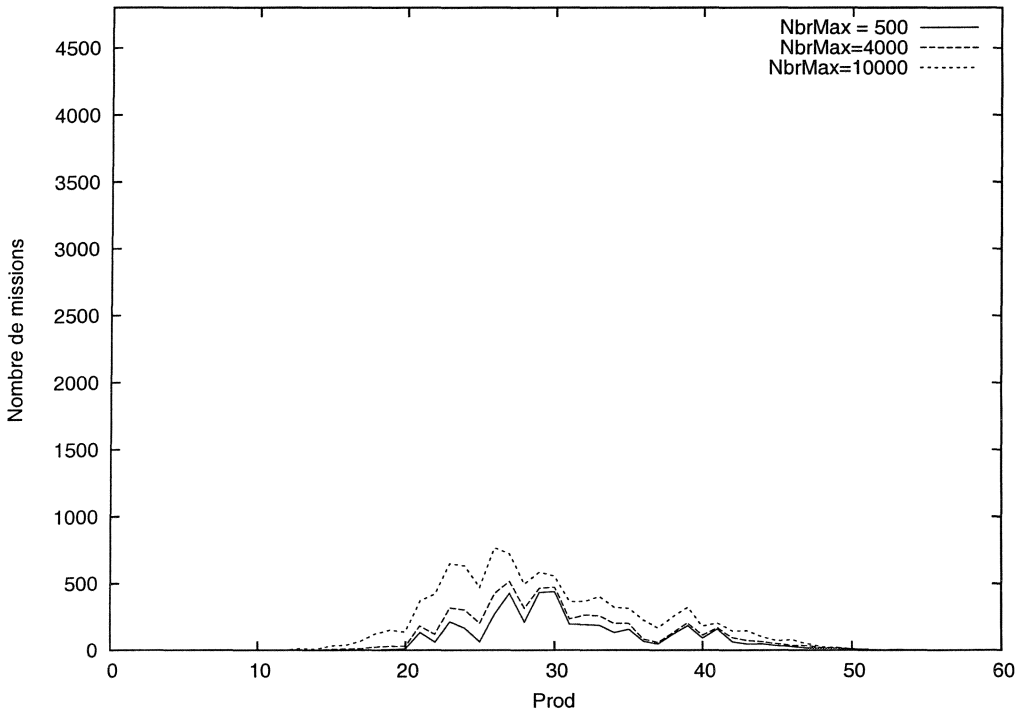

Figure 4.7 – Variation du nombre de missions par valeurs de *prod* quand $NbrMax$ varie pour les missions de 2 jours.

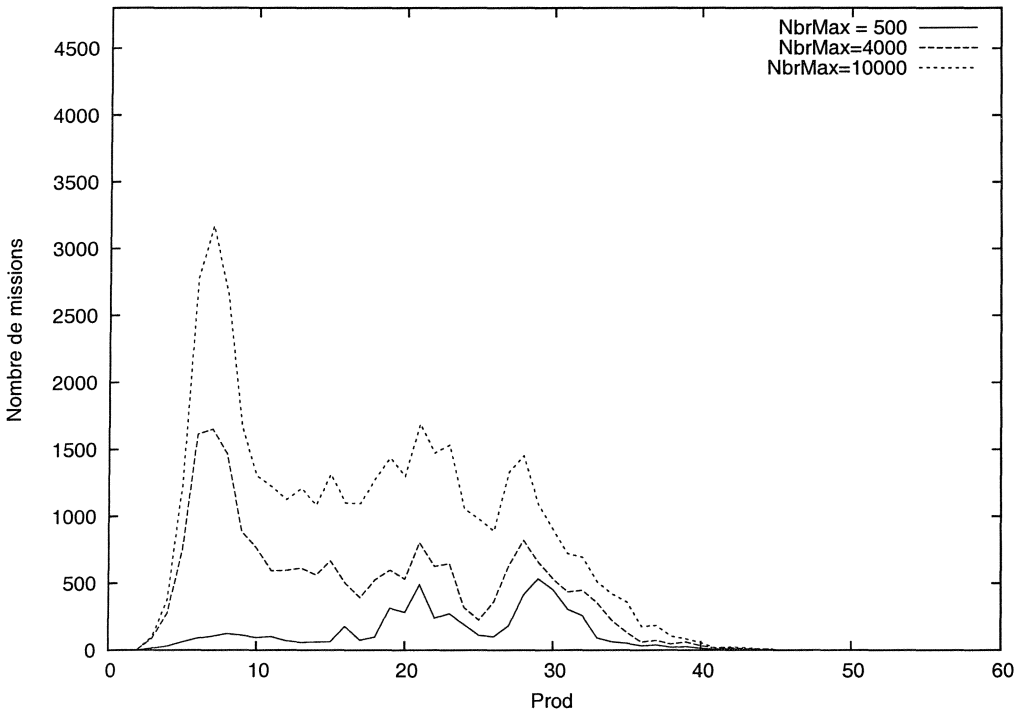

Figure 4.8 – Variation du nombre de missions par valeurs de *prod* quand $NbrMax$ varie pour les missions de 3 jours.

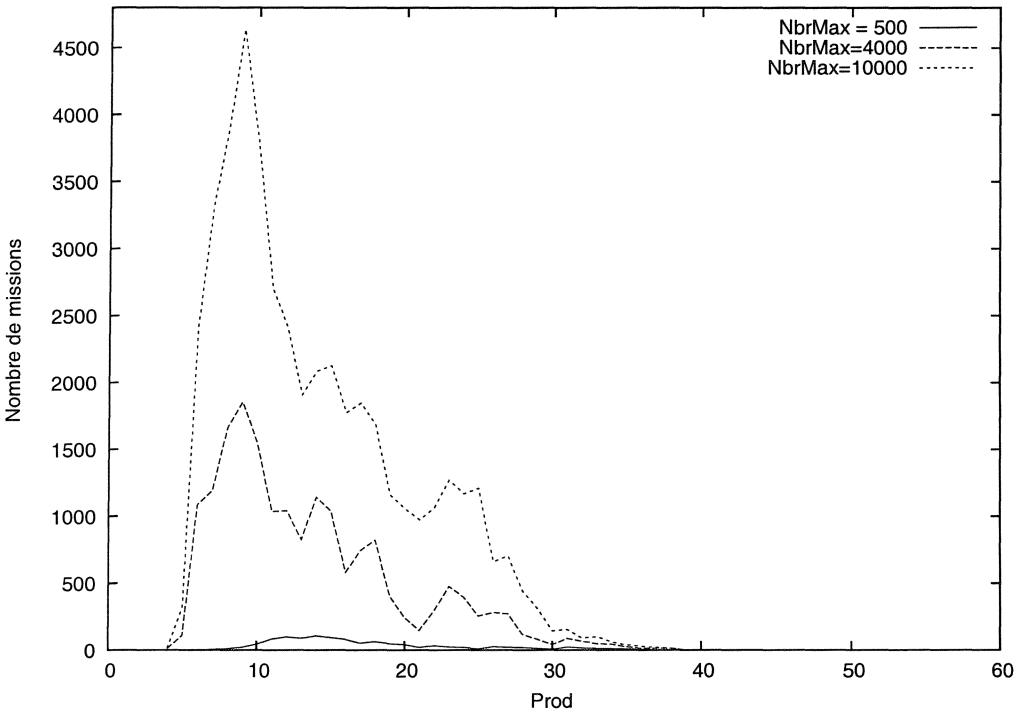

Figure 4.9 – Variation du nombre de missions par valeurs de *prod* quand $NbrMax$ varie pour les missions de 4 jours.

CONCLUSION

Nous avons dans ce mémoire fait suite aux travaux de Rancourt(1998) et Guigue(2000) qui présentaient pour le premier, un modèle mathématique pour résoudre le problème de fabrication de plan de vols de la division 1-CAD, en utilisant des segements de vol et pour le second, une autre approche qui considérait des missions générées *a priori*. Notre objectif a été dans un premier temps de modifier le générateur de missions, proposé dans l'approche de Guigue(2000) en y incluant des contraintes de capacités de manière implicite plutôt qu'explicite et de considérer les incompatibilités à la place des compatibilités. Ceci c'est fait par la modification de certains fichiers du code source du générateur de missions et l'ajout de certains autres.

Dans un deuxième temps, nous avons analysé les missions fournies par le générateur dans le but de pouvoir les classer et les sélectionner selon certains critères. Nous avons pu en fonction des critères choisis, réduire l'ensemble des missions générées ainsi les temps de calcul de plus de la moitié tout en retrouvant une bonne solution pour le scénario que nous avons étudié. Il serait toutefois intéressant de valider ces resultats sur d'autres scénarios.

Le but de notre étude étant d'étudier la pertinence de réduire l'ensemble des missions à considérer, le travail de sélection c'est fait une fois toutes les missions générées car le problème de séléction *a priori*, tout comme pour la génération de de rotations (problème de pairing), est un problème difficile. Mais il serait possible d'intégrer des paramètres au générateur de missions, pour que lors de processus de génération on puisse intervenir sur le nombre de missions à générer de chaque type en fonction des critères choisis.

BIBLIOGRAPHIE

AIR TRANSPORT GROUP.CC130 Hercules. Configurations/Floors plans. *CFACM 60-2631.*

BARNHART, C., JOHNSON, E.L, NEMHAUSEUR, G.L, SAVELSBERGH, M.W.P et VANCE, P.H.(1998). Branch and Price : Column Generation for solving Huge Interger Programs. *Operations Research, Vol. 46, No. 3.*

DESAULNIERS, G., DESROSIERS, J., IOACHIM, I., SOLOMON, M.M., SOUMIS, F. et VILLENEUVE, D.(1997). A Unified Framework for Deterministic Time-Constrained Vehicle Routing and Crew Scheduling Problems. *Les cahiers du GERAD, G94-46.*

DESAULNIERS, G., DESROSIERS, J., LASRY, A. et SOLOMON, M.M.(1998). Crew Pairing for a Regional Carrier. *Les cahiers du GERAD, G97-33.*

DUMAS, Y., DESROSIERS, J. et SOUMIS, F.(1991). The pickup and delivery problem with time windows. *European Journal of Operational Research, 54, 7-22.*

GUIGUE, A. (2000). Fabrication d'horaires dans les forces armées canadiennes : approche par gérération de missions et réduction de réseau. *Mémoire de maîtrise, École Polytechnique de Montréal, Canada.*

RANCOURT, E. (1998). Planification des vols pour le groupe de transport aérien des forces armées canadiennes. *Mémoire de maîtrise, École Polytechnique de Montréal, Canada.*

RANCOURT, E., SAVARD, G. (1999). Decision support systems for simultaneous aircraft and crew scheduling, statement of the problem.

ANNEXE A :

A.1 Format des entrées du générateur de missions original

Les données d'entrées dont a besoin le générateur de missions sont stockées dans des fichiers et ont le format suivant :

1. **Les requêtes de transport :**

Requête	Priorité	Départ	Arrivée	Brief.	Débrief.	Charg.	Décharg.	Avion

On retrouve sous la colonne :
- Requête, le numéro d'identification des requêtes ;
- Priorité, la priorité de la requête qui est un nombre compris entre 1 et 8. Cette information n'intervient pas au stade de la construction des missions mais plutôt au moment de l'affectation des missions au lignes d'affectations ;
- Départ, le nom de l'aéroport de chargement de la requête ;
- Arrivée, le nom de l'aéroport de déchargement de la requête ;
- Briefing, débriefing, chargement et déchargement, les durées correspondantes en heures ;
- Avions, le type d'avion pouvant transporter le fret de la requête.

2. **Types d'avion :**

Type d'avion	Vitesse moyenne	Durée de vol max. sans ravitaillement

On retrouve sous la colonne :

- Type d'avion, le type de l'appareil ;
- Vitesse moyenne, la vitesse de l'appareil en noeuds ;
- Durée de vol max, la durée maximale de vol en heures que peut effectuer l'avion sans ravitaillement.

3. **Liste des aéroports :**

Aéroport	Longitude	Latitude	Zone horaire	Heure ouverture	Heure fermeture

On retrouve sous la colonne :

- Aéroport, le nom de l'aéroport ;
- La position géographique, c'est-à-dire la longitude, latitude et zone horraire, sont nécessaires pour le calcul des temps de vol entre les différents aéroports, dans le cas où cette information n'est pas founie. Dans le cas contraire la position géographique n'est pas nécessaire pour la construction des missions.
- Heure ouverture et Heure fermeture, les heures locales d'ouverture et de fermeture des aéroports.

4. **Les fenêtres de temps pour quitter les aéroports de chargement et déchargement :**

Requête	début charg.	fin charg.	début déchrag.	fin décharg.

Sous chacune des colonnes on retrouve la date et l'heure pour le début ou pour la fin de la fenêtre de temps.

5. **Temps de vol entre les différents aéroports en fonction du type d'appareils :**

Type d'avion	Aéroport de départ	Aéroport d'arrivée	temps de vol

Les temps de vol sont donnés en heures.

6. **Combinaisons de fret permises :**

   Le type d'avion suivi des numéros des requêtes compatibles.

A.2 Format des sorties du générateur de missions original

Le résultat du générateur, à savoir les missions non prédéfinies, se présentent sous la forme suivante :

- Identificateur de la mission débutant par *ImpM* suivi d'un numéro.
- Type d'avion supportant la mission ;
- Fenêtre de temps comprenant la date et l'heure pour débuter la mission ;
- Le mot *START* signifiant que la mission débute à partir de ce point ;
- Liste des aéroports visités suivis des durées de briefing correspondantes ;
- *17 :00*, qui signifie période de repos ;
- Liste des aéroports visités suivis des durées de briefing correspondantes ;
- Durée du debriefing ;
- Le mot *END* signifiant la fin de la mission ;
- Numéro de la requête couverte ;
- Numéro de l'aéroport de chargement ;
- Numéro de l'aéroport de déchragement ;

La liste des aéroports visités peut se réduire à deux aéroports, à savoir l'aéroport de chargement et de déchargement.

La période de repos, peut ne pas apparaître si la mission peut se faire durant une

journée de travail.
Prenons par exemple la ligne suivante du fichier de sortie des missions implicites :

ImpM33|CC130|1996-08-03 8 :23|1996-08-03 10 :38|START|CYTR|0 :05|CYWK|1 :00|
CYYR|1 :00|CYZX|1 :00|CYFC|1 :00|CYZX|17 :00|CYTR|0 :05|END|080206-1/3|1|3|
080206-2/3|2|3|080206-3/3|4|5 ;

ImpM33 est l'identificateur de la mission. La mission peut commencer dans la fenêtre de temps |1996-08-03 8 :23|1996-08-03 10 :38|. L'aéroport de départ de la mission est *CYTR* (la base) qui est l'aéroport 0, pour lequel la durée de briefing est 5 minutes. Le prochain aéroport visité est *CYWK* et qui est l'aéroport 1, pour lequel la durée de briefing est 1 heure, viennent ensuite les aéroports *CYYR, CYZX,CYFC, CYZX* qui sont les aéroports 2, 3, 4 et 5. Les aéroports 2, 3 et 4 sont suivis de leur durée de briefing correspondantes. Le cinquième aéroport est suivi de 17 :00, ce qui veut dire que l'équipage s'y repose avant de repartir pour l'aéroport *CYTR* qui est le dernier aéroport et aussi la base. Ce dernier est suivi de la durée du debriefing. On voit par la suite le mot *END* qui veut dire que la mission est achevée. Il y a en suite le numéro de la requête ici 080206-1/3, qui est chargée à l'aéroport 1 et déchargée à l'aéroport 3, puis la requête 080206-2/3, qui est chargée à l'aéroport 2 et déchargée à l'aéroport 3 et finalement la requête 080206-3/3, qui est chargée à l'aéroport 4 et déchargée à l'aéroport 5.

A.3 Format des entrées du générateur de missions modifié

A.3.1 Les fichiers modifés

1. **Les requêtes de transport :**

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	Poids	Long	Pax	Nature	Type	(9)

Comme pour les donnés du Générateur originale les colonnes (1), (2), (3), (4), (5), (6), (7), (8) et(9) représentent les colonnes *Requête, Priorité, Départ, Arrivée, Brief., Débrief., Charg., Décharg* et *avion* respectivement et définissent les mêmes données. Pour les autres colonnes on retrouve, sous :

- Poids, le poids du fret de la requête. La mesure utilisée est le kilogrammes, mais pourrait tout aussi bien être des livres. La seule condition est que la capacité de l'appareil soit exprimée dans la même unité de mesure ;
- Long, le nombre de palette constituant le fret de la requête ;
- Pax, le nombre de passagers ;
- Nature, s'il s'agit de passagers, de cargo ou les deux et on notera p, c ou pc respectivement ;
- Type, le genre de fret à transporter, s'il s'agit par exemple de carburant ou de matière explosive ou autre.

2. **Les types d'avion :**

Avion	Vitesse	Durée vol max	Nbr plts max	charge max	Nbr pax max

Les trois permières colonnes sont les mêmes que pour les données du générateur de missions original et les trois dernières colonnes représentent la capacité maximale de l'appareil en terme de nombre de palettes, de poids et de nombre de passagers.

A.3.2 Les fichiers ajoutés

1. **Les incompatibilités de fret**

   Chaque type de fret possible est suivi de la liste des types avec lesquels il est

incompatible.

2. **La conversion du nombre de passagers en nombre de palettes**
   Pour chaque type d'avion, un nombre de passagers est suivi du nombre de palettes dont il occupe l'espace dans l'appareil.

ANNEXE B : LISTE DES FICHIERS DU CODE SOURCE

On retrouve dans cette annexe la liste des différents fichiers constituant le code source de l'optimiseur développé pour résoudre le problème de fabrication de PVM. Seuls les fichiers nécessaires à la génération de missions sont décrits brièvement. Ces fichiers peuvent être classés selon trois types à savoir spécifiques au générateur de missions, c'est-à-dire utilisé exclusivement dans le cadre de la génération de missions, ou utilisé par le générateur de missions dans ce cas il s'agit de fichiers qui sont également utilisés dans le processus de résolution ou finalement des ficheirs qui ont été ajoutés dans le cadre de cette étude. Le reste des fichiers sera simplement listé.

Il y existe trois principaux répertoires :

- *line-task*;
- *sacs-util*;
- *onecad*.

Les deux premiers seront détaillés plus loin. Le troisième contient le programme principal. Les figures B.1 à B.3 représentent l'organisation des répertoires.

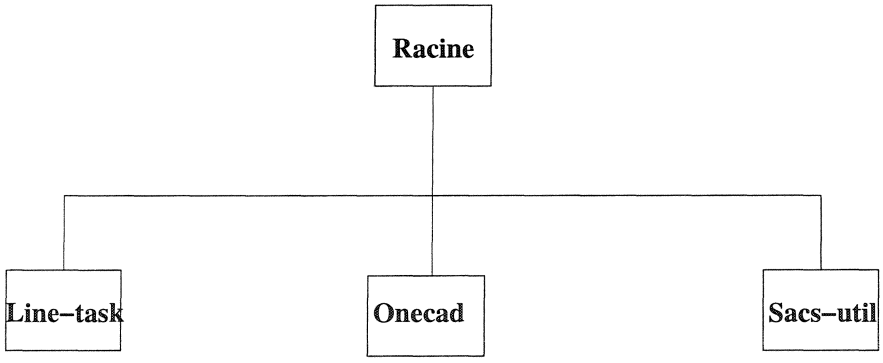

Figure B.1 – Organisation du répertoire principal

B.1 Détail du répertoire line-task

Dans ce répertoire on retrouve, les sous répertoires suivants :

- *LtpApplicDbClass*
- *LtpApplicDb*
- *LtpControl*
- *LtpGencol*
- *LtpOptStrategy*
- *LtpOptSupport*
- *LtpParsers*
- *LtpSolutionViews*

Les différents fichiers de ce répertoire définissent le *namespace LTP*. Les sous-répertoires ombragés dans la figure B.2, ne sont pas détaillés.
Les tableaux B.1 à B.6 précisent le type de chaque fichier.

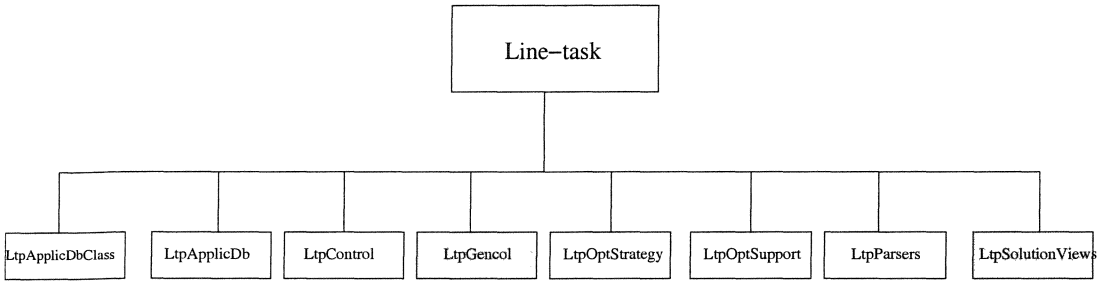

Figure B.2 – Organigramme du répertoire line-task

B.1.1 Détail du sous répertoire *LtpApplicDbClass*

- *AirliftRequest.h* définit une classe représentant une requête de transport ;
- *Conversions.h et .cxx* voir définition dans le chapitre 3 ;

- *DutySegment.h et .cxx* définissent une classe de base qui contient des information des segments de journée de travail ;
- *Duty SegmentBuider.h et .cxx* définissent une classe qui hérite de la classe précédente et permet de construire un objet qui représente un segment de journée ;
- *fretIncomp.h* voir définition dans le chapitre 3 ;
- *GenericRequest.h* définit une classe simple pour les objets représentants une requête ;
- *Mission.h et .cxx* définissent une classe de base pour les objets repésentants une mission ;
- *MissionBuilder.h et .cxx* permettent de construire les objets missions ;
- *Request.h* définit une classe de base pour les objets représentants différents types de requêtes ;
- *RequestCompatibility.h et .cxx* classe permettant de vérifier la compatibilité de requêtes entre elles ;

B.1.2 Détail du sous répertoire *LtpApplicDb*

- *ApplicDbBuilder.h et .cxx* permettent de construire la base de données de l'application.

B.1.3 Détail du sous répertoire *LtpControl*

- *DutySegmGenFactory.h et .cxx* permettent de construire les objets de type segment de journées ;
- *MissGenFactory.h et .cxx* utilisés pour la génération de missions au départ d'une base et couvrant un ensembles de segments de journées ;
- *MissionGeneratorFactory.h* génère les missions pour un ensemble de requêtes donné ;

Tableau B.1 – Liste des fichiers de *LtpApplicDbClass*

Nom du fichier	Spécifique	Utilisé	Ajouté
AirliftRequest.h		X	
Conversions.cxx			X
Conversions.h			X
CostCalculator.h			
DutySegment.cxx	X		
DutySegment.h	X		
DutySegmentBuilder.cxx	X		
DutySegmentBuilder.h	X		
FretIncomp.h			X
GenericRequest.h		X	
Mission.cxx		X	
Mission.h		X	
MissionBuilder.cxx	X		
MissionBuilder.h	X		
Request.h		X	
RequestCompatibility.cxx	X		
RequestCompatibility.h	X		
Solution.cxx			
Solution.h			

- *Params.h et .cxx* permettent de fixer les différents paramètres, comme les noms des fichiers d'entrée, des fichiers de résultats, le nombre de missions à générer par requête ;

B.1.4 Détail du sous répertoire *LtpGencol*

- BaseFix.cxx
- BaseFix.h
- BaseFixMethod.cxx
- BaseFixMethod.h

Tableau B.2 – Liste des fichiers de *LtpApplicDb*

Nom du fichier	Spécifique	Utilisé	Ajouté
ApplicDbBuilder.cxx		X	
ApplicDbBuilder.h		X	

- GencolFactory.cxx
- GencolFactory.h

B.1.5 Détail du sous répertoire *LtpOptStrategy*

- *DutySegmGenInterface.h et .cxx* définissent la classe qui sert d'interface au solver pour générer les segments de journées de travail pour un certain type d'avion ;
- *DutySegmGenNetDbBuilder.h et .cxx* définissent la classe permettant la construction du réseau des segments de journées de travail ;
- *DutySegmGenPathEnum.h et .cxx* définissent la classe qui permet d'énumérer les chemins réalisables dans le réseau des segments de journées de travail ;
- *MissGenInterface.h et .cxx* définissent la classe qui sert d'interface au solver pour générer les missions pour un certain type d'avion et un ensemble de segments de journées de travail ;
- *MissGenNetDbBuilder.h et .cxx* définissent la classe permettant la construction du réseau des missions ;
- *MissGenPathEnum.h et .cxx* définissent la classe qui permet d'énumérer les chemins réalisables dans le réseau des missions ;

B.1.6 Détail du sous répertoire *LtpOptSupport*

- *AirportTaskNameBuilder.h* classe permettant de donner un nom unique à la tâche correspondant à un aéroport ;

- *DutyTaskNameBuilder.h* classe permettant de donner un nom unique à la tâche correspondant à un segment de journée de travail ;
- *MissGenNetworkBldr.h et .cxx* définissent une classe utilisée pour la construction du réseau pour la génération des missions couvrant un ensemble spécifique de requêtes ;

B.1.7 Détail du sous répertoire *LtpParsers*

- *AirliftRequestParser.h et .cxx* classe permettant la lecture des données relatives aux requêtes de transport ;
- *ConversionPsgrPaletParser.h et .cxx* voir chapitre 3 ;
- *IncompCombParser.h et .cxx* voir chapitre 3 ;
- *LoadOffloadParser.h et .cxx* classe permettant la lecture des fenêtres de temps pour le chargement et déchargement des requêtes ;
- *PermCombParser.h et .cxx* permet la lecture des données concernant les combinaisons permises de fret des requêtes ;
- *RequestGroupParser.h et .cxx* classe permettant la lecteur des groupe de requêtes.

Tableau B.3 – Liste des fichiers de*LtpControl*

Nom du fichier	Spécifique	Utilisé	Ajouté
BestSolutionWriterFactory.cxx			
BestSolutionWriterFactory.h			
DeleteSolutionsFactory.cxx			
DeleteSolutionsFactory.h			
DutySegmGenFactory.cxx	X		
DutySegmGenFactory.h	X		
FindFeasSolFactory.cxx			
FindFeasSolFactory.h			
FindMissOptimSetFactory.cxx			
FindMissOptimSetFactory.h			
FindMissSetFactory.cxx			
FindMissSetFactory.h			
LinesBuilderFactory.cxx			
LinesBuilderFactory.h			
MissGenFactory.cxx	X		
MissGenFactory.h	X		
MissionGeneratorFactory.cxx	X		
MissionGeneratorFactory.h	X		
OptimFactory.cxx			
OptimFactory.h			
Params.cxx		X	
Params.h		X	
SolutionBuilderFactory.cxx			
SolutionBuilderFactory.h			
SolutionRegisterFactory.cxx			
SolutionRegisterFactory.h			
SolutionVerifierFactory.cxx			
SolutionVerifierFactory.h			
SolverFactory.cxx		X	
SolverFactory.h		X	

Tableau B.4 – Liste des fichiers de *LtpOptStrategy*

Nom du fichier	Spécifique	Utilisé	Ajouté
DutySegmGenInterface.cxx	X		
DutySegmGenInterface.h	X		
DutySegmGenNetDbBuilder.cxx	X		
DutySegmGenNetDbBuilder.h	X		
DutySegmGenPathEnum.cxx	X		
DutySegmGenPathEnum.h	X		
MissGenInterface.cxx	X		
MissGenInterface.h	X		
MissGenNetDbBuilder.cxx	X		
MissGenNetDbBuilder.h	X		
MissGenPathEnum.cxx	X		
MissGenPathEnum.h	X		
NetDbBuilder.cxx			
NetDbBuilder.h			
NetDbBuilderBasic.cxx			
NetDbBuilderBasic.h			
NetDbBuilderDecomp.cxx			
NetDbBuilderDecomp.h			
NetDbBuilderMIP.cxx			
NetDbBuilderMIP.h			

Tableau B.5 – Liste des fichiers de *LtpOptSupport*

Nom du fichier	Spécifique	Utilisé	Ajouté
AirportTaskNameBuilder.h		X	
AlgoBlocage.cxx			
AlgoBlocage.h			
AlgoDoublets.cxx			
AlgoDoublets.h			
AlgoFindCompMiss.cxx			
AlgoFindCompMiss.h			
AlgoKeepMission.cxx			
AlgoKeepMission.h			
AlgoPartition.cxx			
AlgoPartition.h			
AlgoStationEvent.cxx			
AlgoStationEvent.h			
ChosenMissionSet.h			
DutySequenceSet.h			
DutyTaskNameBuilder.h	X		
MissGenNetworkBldr.cxx	X		
MissGenNetworkBldr.h	X		
MissionEndTaskName.h			
MissionNetworkBldr.cxx	X		
MissionNetworkBldr.h	X		
MissionTaskNameBuilder.h			
RequestTaskAdder.cxx			
RequestTaskAdder.h			
RequestTaskNameBuilder.h			
SlackNetworkBuilder.cxx			
SlackNetworkBuilder.h			

Tableau B.6 – Liste des fichiers de *LtpParsers*

Nom du fichier	Spécifique	Utilisé	Ajouté
AirliftRequestParser.cxx		X	
AirliftRequestParser.h		X	
ConversionPsgrPaletParser.cxx			X
ConversionPsgrPaletParser.h			X
GenericMissionParser.cxx			
GenericMissionParser.h			
GenericRequestParser.cxx			
GenericRequestParser.h			
IncompCombParser.cxx			X
IncompCombParser.h			X
LoadOffloadParser.cxx	X		
LoadOffloadParser.h	X		
MissionStatusParser.cxx			
MissionStatusParser.h			
PermCombParser.cxx	X		
PermCombParser.h	X		
RequestGroupParser.cxx		X	
RequestGroupParser.h		X	
SolutionParser.cxx			
SolutionParser.h			

B.1.8 Détail du sous répertoire *LtpSolutionViews*

- FindMissSetOutSolutionView.cxx
- FindMissSetOutSolutionView.h
- OutSolutionView.cxx
- OutSolutionView.h

B.2 Détail du répertoire sacs-util

Ce répertoire est constitué des sous répertoires suivants :

- *AcsApplicDbClass*
- *AcsControl*
- *AcsGencol*
- *AcsNetDbSupport*
- *AcsOptions*
- *AcsOptStrategy*
- *AcsParsers*
- *AcsUtil*

Les différents fichiers de ce répertoire définissent le *namespace ACS*. Les sous-répertoires ombragés dans la figure B.3, ne sont pas détaillés.
Les tableaux B.7 et B.8 précisent le type de chaque fichier.

B.2.1 Détail du sous répertoire *AcsApplicDbClass*

- *AicraftType.h* sert à représenter un type d'avion avec ses caractéristiques.
- *AirForceBase.h* représente l'aéroport d'attache des appareils et des équipages.

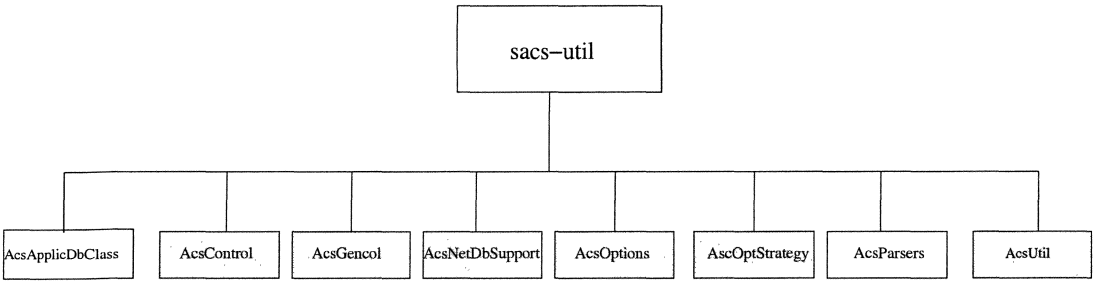

Figure B.3 – Organigramme du répertoire sacs-util

- *Airport.h* représente un aéroport où un appareil peut se poser ou duquel il peut décoller.
- *AlwaysOpenedAirport.h* dérive de la classe précédente et et repésente un type particulier d'aéroport qui est toujours ouvert.
- *Itinerary.h et .cxx* servent à stocker des informations concernant l'itinéraire d'une mission associé à un type d'avion et débutant et se terminant á une base.
- *Location.h et .cxx* servent à repésenter un point sur le globe caractérisé par sa latitude et sa longitude.
- *TravelTimes.h et .cxx* servent d'interface pour obtenir les temps de vol entre différents aéroports.

Tableau B.7 – Liste des fichiers de *AcsApplicDbClass*

Nom du fichier	Spécifique	Utilisé	Ajouté
AircraftType.h		X	
AirForceBase.h		X	
Airport.h		X	
AlwaysOpenedAirport.h		X	
ApplicDbObj.h			
ApplicDbObjPtrSet.h			
ApplicDbSingletonObj.h			
BaseAcTypeKey.h			
Itinerary.cxx		X	
Itinerary.h		X	
LineAvail.cxx			
LineAvail.h			
LineAvailOrder.h			
Location.cxx		X	
Location.h		X	
NameBuilder.cxx			
NameBuilder.h			
TravelTimes.cxx		X	
TravelTimes.h		X	

B.2.2 Détail du sous répertoire*AcsControl*

- ProgressFileCreator.cxx
- ProgressFileCreator.h
- ReuserFactory.h

B.2.3 Détail du sous répertoire *AcsGencol*

- GlobalParams.cxx
- GlobalParams.h
- Helpers.h
- ItfixOneParams.cxx
- ItfixOneParams.h
- ModelParams.cxx
- ModelParams.h

B.2.4 Détail du sous répertoire*AcsNetDbSupport*

- PathEnum.cxx
- PathEnum.h
- ResourceConsSetter.h
- ResourceCreator.h
- ResourceListGetter.cxx
- ResourceListGetter.h
- ResourceWinSetter.h
- RowCreator.h
- StationCreator.h
- TaskAdder.h

- TaskCreator.h
- TaskRowContributor.h
- TreeNode.cxx
- TreeNode.h

B.2.5 Détail du sous répertoire *AcsOptions*

- Options.cxx
- Options.h

B.2.6 Détail du sous répertoire *AcsOptStrategy*

- VoidSolutionSolver.cxx
- VoidSolutionSolver.h

B.2.7 Détail du sous répertoire*AcsParsers*

- *AircraftParser.h et cxx* permettent d'obtenir les disponibiltés des avions pour les différentes bases ;
- *AircraftTypeParser.h et .cxx* permettent la création de l'objet type d'avio ainsi que la lecture des données relatives à ce type d'objet ;
- *AirForceBaseParser.h* permet la lecture du nom des bases ;
- *AirportParser.h et .cxx* permettent la lecture des noms des aéroports et la création des objets de ce type ;
- *FeasibleLegParser.h et .cxx* permettent la lecture des temps de vol entre les différents aéroports ;
- *ForbiddenLegParser.h et .cxx* permettent la lecture des legs interdits.

Tableau B.8 – Liste des fichiers de *AcsParsers*

Nom du fichier	Spécifique	Utilisé	Ajouté
AbstractParser.cxx			
AbstractParser.h			
AircraftParser.cxx		X	
AircraftParser.h		X	
AircraftTypeParser.cxx		X	
AircraftTypeParser.h		X	
AirForceBaseParser.h		X	
AirportParser.cxx		X	
AirportParser.h		X	
ApplicDbObjIdParser.h			
ApplicDbObjIdParser2.h			
AvailExcParser.cxx			
AvailExcParser.h			
DateParser.h			
DoubleParser.h			
FeasibleLegParser.cxx		X	
FeasibleLegParser.h		X	
FileParser.cxx			
FileParser.h			
FileSetParser.cxx			
FileSetParser.h			
ForbiddenLegParser.cxx		X	
ForbiddenLegParser.h		X	
ItineraryParser.cxx			
ItineraryParser.h			
KeywordParser.cxx			
KeywordParser.h			
LongParser.h			
LongRangeParser.cxx			
LongRangeParser.h			
NewApplicDbObjIdParser.h			
ParserException.cxx			
ParserException.h			
StringParser.h			
TimeIntervalParser.h			
TimeParser.h			
TokenParser.h			
UntilSymbolParser.h			

B.2.8 Détail du sous répertoire *AcsUtil*

- AlgoTimeIntervalPartition.h
- CastPointerIter.h
- DiscSetSubset.h
- Error.cxx
- Error.h
- HorizonSubset.h
- LocalDateTime.cxx
- LocalDateTime.h
- Moment.h
- ProgressMsg.cxx
- ProgressMsg.h
- SecondCastIter.h
- Secure.h
- TimeInterval.cxx
- TimeInterval.h