POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Résolution du problème de construction des horaires mensuels d'une compagnie aérienne avec la méthode d'exploration de voisinage d'un ensemble initial de clusters

SALAH-EDDINE MAKHLOUFI
Département de mathématiques et de génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Mathématiques appliquées

Mars 2020

© Salah-Eddine Makhloufi, 2020.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Résolution du problème de construction des horaires mensuels d'une compagnie aérienne avec la méthode d'exploration de voisinage d'un ensemble initial de clusters

présenté par **Salah-Eddine MAKHLOUFI**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Issmaïl EL HALLAOUI, président
François SOUMIS, membre et directeur de recherche
Samuel ROSAT, membre

REMERCIEMENTS

Tout d'abord, je tiens à remercier mes parents et ma famille pour m'avoir supporté durant toutes les étapes de ma vie et pour m'avoir soutenu inconditionnellement durant les épreuves que j'ai traversé...

Je remercie très sincèrement mon directeur de recherche François Soumis pour le support et l'aide compétente qui m'a apporté durant mon projet de recherche. Il était toujours positif et à l'écoute. Cela m'a beaucoup aidé à surpasser les difficultés que j'ai rencontré durant mon travail de recherche.

Je remercie Issmail El Hallaoui d'avoir accepté la présidence de mon jury et Samuel Rosat qui a accepté d'en être membre.

J'aimerais également remercier mes collèges au sein du GERAD et AD OPT pour les discussions et les conseils qui m'ont offert.

RÉSUMÉ

Les compagnies aériennes ont eu recours à la recherche opérationnelle pour résoudre efficacement une partie de leurs problèmes de gestion et de planification. Parmi ces derniers, on trouve le PCHMME (problème de construction d'horaires mensuels des membres d'équipage). Ce problème consiste à construire un horaire mensuel pour chaque membre d'équipage en lui affectant un ensemble de rotations ainsi que d'autres tâches telles que les vacances et les formations afin de construire un horaire mensuel pour ce membre. Une rotation représente une succession de vols séparés par des périodes de repos. Une rotation débute et se termine dans la même base (aéroport). L'horaire construit doit respecter l'ensemble des règles gouvernementales, les règles de la convention collective, etc. L'ensemble de rotations est construit durant une étape antérieure.

Le problème de construction des horaires est un problème combinatoire et dont la complexité croît exponentiellement avec la taille du problème et le nombre de règles à respecter.

La compagnie AD OPT offre des produits d'optimisation dans le domaine du transport aérien. Un de ces produits est dédié à la résolution du PCHMME. Ce dernier est basé sur un solveur qui utilise la programmation linéaire, en particulier l'approche de la génération de colonnes.

L'objectif principal de ce travail est d'améliorer le temps de résolution du PCHMME pour une compagnie aérienne donnée. Cette dernière est classée parmi les plus grandes compagnies aériennes au monde suivant un classement basé sur la taille de la flotte et le nombre d'employés. La taille du PCHMME correspondant est alors très grande.

Le personnel de cabine de cette compagnie est classifié dans des catégories telles que la catégorie CM qui regroupe les chefs de cabine et la catégorie FA qui regroupe la grande majorité du reste des membres de cabine. La caractéristique la plus importante pour le présent travail qui distingue ces deux catégories est la taille. En effet, la taille de la catégorie FA est beaucoup plus grande que celle de la catégorie CM.

Le PCHMME est résolu d'une manière séquentielle, catégorie par catégorie. Dans ce travail, on propose l'utilisation de la solution de la catégorie CM afin d'accélérer la résolution de la catégorie FA, car la durée de résolution de cette dernière est beaucoup plus longue que celle de la catégorie CM.

À partir de la solution de la catégorie CM, on construit un ensemble de clusters qui regroupe les rotations du problème de FA. Chaque cluster représente une succession de rotations. L'ensemble de clusters servira de cible durant la résolution des sous-problèmes de la catégorie FA. Seulement les chemins qui sont au voisinage de cet ensemble seront générés, ce qui permet

de réduire le temps de résolution des sous-problèmes.
Plusieurs méthodes ont été proposées afin de construire cet ensemble de clusters. Une de ces méthodes est basée sur l'utilisation d'un algorithme de flot à coût minimum pour obtenir un ensemble de clusters avec une cardinalité minimale.
Au niveau de la résolution de la catégorie FA, on propose une approche améliorée de la génération de colonnes avec la MEV (méthode d'exploration de voisinage). Cette méthode est implémentée au niveau du sous-problème et permet de ne générer que les colonnes qui sont au voisinage de l'ensemble initial de clusters. On a également proposé plusieurs versions de la MEV pour déterminer la taille du voisinage à explorer durant la résolution des sous-problèmes. Parmi ces versions, on trouve la version dynamique de la MEV qui permet de déterminer une distance maximale permise par rapport à l'ensemble initial de clusters avant chaque résolution de sous problème. Cette valeur est déterminée suivant le nombre et la qualité des colonnes générées.
La MEV permet également aux membres de la catégorie FA d'avoir un nombre minimum de chefs durant le mois. Cela a un impact positif sur la productivité de l'employé.
La MEV n'a pas donné de bons résultats pour la compagnie étudiée, car cette dernière a plusieurs particularités qui nuisent à la MEV. Pour cette raison, on a proposé un raffinement de données afin de produire des données qui ressemblent plus aux celles des autres compagnies qui ont moins de caractéristiques qui vont à l'encontre de la MEV.
Après avoir testé avec les nouvelles données, on n'a pas obtenu l'amélioration attendue. Ceci est dû à une mauvaise interaction entre la MEV et des algorithmes utilisés au niveau de l'approche de résolution existante dans le produit d'AD OPT.

ABSTRACT

Airlines have used operational research to effectively resolve some of the management and planning issues they face. Among these, we find the crew rostering problem CRP which consists in assigning for each given crew member, a set of pairings as well as other tasks such as vacations and training in order to build a monthly schedule for this crew member. The schedule built must comply with all government rules and the rules of the collective agreement. The set of pairings is built during an earlier stage (a pairing represents a succession of flights and starts and ends at the same airport).

The CRS is a combinatorial problem and its complexity increases proportionally with the size of the problem and the number of rules to be respected.

The company AD OPT offers optimization products in the aviation sector. One of these products is dedicated to the resolution of CRP. The latter is based on a solver that uses linear programming, in particular the column generation approach.

The main objective of this work is to improve the time consumption of CRP for a given airline. This airline is ranked among the largest airlines in the world according to a ranking based on the size of the fleet and the number of employees. Therefore the size of the corresponding CRP is very large. The cabin crew members of this company are classified into categories and their CRP is solved sequentially category by category.

In this work, we focus on the category of the cabin managers and the FA category which represents the rest of the members. These two categories have a lot in common, for that reason, we propose to use the solution of the cabin managers category in order to speed up the resolution of the FA category because the size of the latter is larger and takes much longer to solve.

From the solution of the cabin managers category, we build a set of clusters which groups together the pairing of the FA problem. Each cluster represents a succession of pairings. The set of clusters is used as a target during the resolution of sub-problems of FA category. Only paths in the neighborhood of this set will be generated. This will reduce solving time of the subproblems.

Several methods have been proposed in order to build this set of clusters. One of these methods is based on the use of a minimum cost flow algorithm to obtain a set of clusters with minimum cardinality.

To resolve the FA category, we propose an improved approach of the column generation method. This approach is based on a neighborhood search method (NSM). This method is implemented at the subproblem level and only generates the columns which are in the

neighborhood of the initial set of clusters. Several versions of the NSM have been proposed in order to determine the size of the neighborhood to explore during the resolution of the sub-problems. Among these versions, there is the dynamic one which determines before each resolution of the sub-problems, a maximum distance allowed with respect to the initial set of clusters. This value represents the size of the neighborhood to explore and it is determined according to the number and quality of the columns generated.

After testing with the new data, we did not get the expected improvements. This is due to the harmful effects of the dominance heuristic. This heuristic eliminates paths in a quasi-random way during the resolution of the sub-problems.

The NSM also allows FA members to have a minimum of change of managers, during the month. This has a positive impact on the productivity of the employee.

The NSM did not give very improving results for the company studied because the latter has several characteristics which harm the process of the NSM. For this reason, a refinement of data has been proposed to produce data that have less characteristics which go against NSM which and it is more like other companies data.

After testing with the new data, we did not get the expected improvements. This is principally due to the harmful effect of the existing heuristics.

TABLE DES MATIÈRES

REMERCIEMENTS iii

RÉSUMÉ iv

ABSTRACT vi

TABLE DES MATIÈRES viii

LISTE DES TABLEAUX xi

LISTE DES FIGURES xii

LISTE DES SIGLES ET ABRÉVIATIONS xiii

CHAPITRE 1 INTRODUCTION 1
1.1 Contexte général du projet 1
1.1.1 L'utilisation de la recherche opérationnelle dans le domaine de transport aérien 2
1.1.2 Processus de planification au sein d'une compagnie aérienne 3
1.2 Description de la problématique et des objectifs poursuivis 4
1.3 Contexte de réalisation du travail 5
1.4 Structure du mémoire 5

CHAPITRE 2 REVUE DE LITTÉRATURE 6
2.1 Construction d'horaires personnalisés 6
2.2 Techniques de résolution 7
2.2.1 Décomposition de Dantzig-Wolfe 7
2.2.2 Méthode de génération de colonnes 8
2.2.3 Effets indésirables de la génération de colonnes 9
2.2.4 La méthode de résolution *Branch and Price* 9
2.2.5 Agrégation dynamique de contraintes 10
2.2.6 La relation entre les techniques de résolution présentées et le présent travail 12

CHAPITRE 3 ANALYSE DE L'EXISTANT 14
3.1 Modèle mathématique 14

3.1.1 Contraintes globales 14
3.1.2 Fonction objectif 16
3.1.3 Contraintes locales 17
3.2 Approche de résolution 17
3.2.1 Processus général 17
3.2.2 Description du réseau du sous-problème 18
3.3 Analyse des données d'entrée et taille du problème 20

CHAPITRE 4 MÉTHODE D'EXPLORATION DE VOISINAGE 22
4.1 Introduction de la MEV 22
4.2 Description de la MEV 23
4.2.1 Implémentation de la MEV 24
4.2.2 Version dynamique de la MEV 25
4.2.3 Stratégies d'accélération de la méthode d'exploration de voisinage . . 26
4.3 Construction d'un ensemble initial de clusters 27
4.3.1 Construction d'un ensemble initial de clusters non disjoints 27
4.3.2 Construction d'un ensemble de clusters disjoints en utilisant un algorithme glouton 28
4.3.3 Construction d'un ensemble de clusters disjoints en utilisant un algorithme de flot à coût minimum 30
4.3.4 Comparaison des résultats de l'algorithme glouton et de l'algorithme de flot 32

CHAPITRE 5 RÉSULTATS NUMÉRIQUES 34
5.1 Description des instances 34
5.2 Exécution avec les données originales 35
5.3 Données raffinées 37
5.3.1 Problèmes rencontrés 37
5.3.2 Solutions proposées 38
5.3.3 Exécution avec les données raffinées 38
5.4 Diagnostique de l'inefficacité de la méthode d'exploration de voisinage 40
5.4.1 Vérification de l'hypothèse 1 40
5.4.2 Vérification de l'hypothèse 2 42
5.5 Étude de la dominance 42
5.5.1 Effets indésirables de l'heuristique de dominance existante dans la version originale du produit 42
5.5.2 Solutions proposées 45

5.5.3 Évaluation de la version originale du produit 46

CHAPITRE 6 CONCLUSION ET RECOMMANDATIONS 48
6.1 Synthèse des travaux 48
6.2 Limitations de la solution proposée 49
6.3 Travaux futurs 49

RÉFÉRENCES 51

LISTE DES TABLEAUX

Tableau 1.1 Exemple de catégories de membres d'équipage de la compagnie étudiée. 4
Tableau 3.1 Exemple de taille des instances CS et FA. 20
Tableau 4.1 Les caractéristiques de chaque type d'arc du réseau $G(N, A)$ 31
Tableau 4.2 Comparaison des ensembles issus des différentes instances et des différentes méthodes de construction 32
Tableau 5.1 Description des valeurs de 'b' utilisées dans les noms des instances. . 35
Tableau 5.2 Paramètres qui caractérisent la MEV pour chaque version. 35
Tableau 5.3 Exécution du problème-cadre avec des données originales 36
Tableau 5.4 Données raffinées à exécuter. 39
Tableau 5.5 Résultat de l'exécution avec les données raffinées 39
Tableau 5.6 Exécution avec des instances pour tester l'hypothèse 1. 41
Tableau 5.7 Description de l'instance réduite. 41
Tableau 5.8 Résultats suite à la perturbation de la version originale 46

LISTE DES FIGURES

Figure 1.1	Revenu du transport aérien entre 2003 et 2019 en milliards USD [1] .	2
Figure 1.2	Évolution de nombre de passagers entre 2004 et 2019 [2]	2
Figure 2.1	Méthode de génération de colonnes.	9
Figure 2.2	Effets indésirables de la génération de colonnes. [3]	10
Figure 3.1	Description du réseau du problème	19
Figure 4.1	Exemples de quelques types de consommation de la ressource	25
Figure 4.2	La version dynamique de la MEV	26
Figure 4.3	Graphe intermédiaire $G'(N', A')$	31
Figure 4.4	Graphe $G(N, A)$	31
Figure 5.1	Illustration d'une partie du réseau	44

LISTE DES SIGLES ET ABRÉVIATIONS

AFIX	*Arc fixation*
BDCA	*Bi-dynamic constraint aggregation*
CEIC	Construction d'un ensemble initial de clusters
CFIX	*Column fixation*
CM	*Cabin managers*
CRP	*Crew rostering problem*
CRS	*Crew rostering system*
CS	*Cabin senior*
DCA	*Dynamic constraint aggregation*
DMT	Distance maximale tolérée
EIC	Ensemble initial de clusters
EICD	Ensemble initial de clusters disjoints
FA	*Flight attendants*
GC	Génération de colonnes
JNB	Aéroport international OR Tambo
MAJ	Mise à jour
MEV	Méthode d'exploration de voisinage
MPDCA	*Multi-phase dynamic constraint aggregation*
NSM	*Neighborhood search method*
OACI	Organisation de l'aviation civile internationale
PCHMME	Problème de construction des horaires mensuels des membres d'equipage
PCR	Problème de construction des rotations
PLNE	Programmation linéaire en nombres entiers
PM	Problème maître
PMR	Problème maître restreint
PMRA	Problème maître restreint agrégé
PNC	Personnel navigant commercial
PNT	Personnel navigant technique
PPCCCR	problème de plus court chemin avec contraintes de ressources
RMEV	Ressource de la méthode d'exploration de voisinage
RL	*Ressource level*
SP	Sous-problème

CHAPITRE 1 INTRODUCTION

Dans ce chapitre, on introduit le domaine du transport aérien, les défis rencontrés par ce dernier, ainsi que le rôle de l'utilisation de la recherche opérationnelle dans le processus de planification pour minimiser les coûts de la compagnie. On introduit également la problématique étudiée, le contexte de réalisation du travail et les objectifs poursuivis.

1.1 Contexte général du projet

Le transport aérien est le déplacement d'individus ou de marchandises par la voie des airs d'un endroit à un autre. Ce service est effectué par des compagnies aériennes qui utilisent essentiellement des avions de ligne qui opèrent des vols entre des aéroports. Un équipage technique est présent dans chaque vol pour assurer la sécurité et le confort des clients. Ce dernier peut être regroupé en deux groupes : les membres du cockpit, appelés aussi personnel navigant technique (PNT) et le personnel de cabine, nommé également personnel navigant commercial (PNC). Les individus qui voyagent à bord d'un avion de ligne sont appelés des passagers aériens.
Le secteur économique qui regroupe ce mode de transport est devenu un secteur majeur dans l'économie mondiale. En effet, les compagnies aériennes ont généré plus de 812 milliards USD de revenu en 2018 et elles ont transporté plus de 1,6 million de passagers. Ce nombre croît avec un taux annuel de 4% à 5% [4].
Les figures 1.1 et 1.2 illustrent l'évolution de revenu et de nombre de passagers dans les dernières années. En outre, le domaine du transport aérien génère plus de 28 millions d'emplois dans le marché mondial et transporte 40% de la valeur des échanges mondiaux de marchandises [4].

En raison de son expansion, le marché devient plus attirant pour de nouveaux investisseurs. Cependant, plusieurs facteurs rendent ce marché très compétitif et la marge de profitabilité devient de plus en plus étroite. Parmi ces facteurs, on trouve l'apparition des compagnies aériennes à bas prix, le coût élevé des dépenses fixes comme le carburant et les membres d'équipages, etc. Par conséquent, les compagnies aériennes adoptent des méthodes ciblant l'efficacité afin de maximiser leurs profits. Une de ces méthodes est la recherche opérationnelle. Cette dernière propose un moyen efficace pour l'optimisation des ressources et du processus de gestion de la compagnie, en particulier, les problèmes de planification.

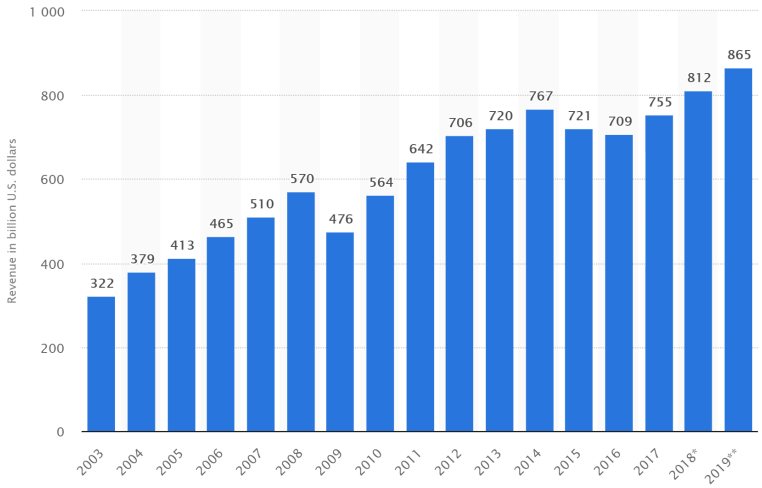

Figure 1.1 Revenu du transport aérien entre 2003 et 2019 en milliards USD [1]

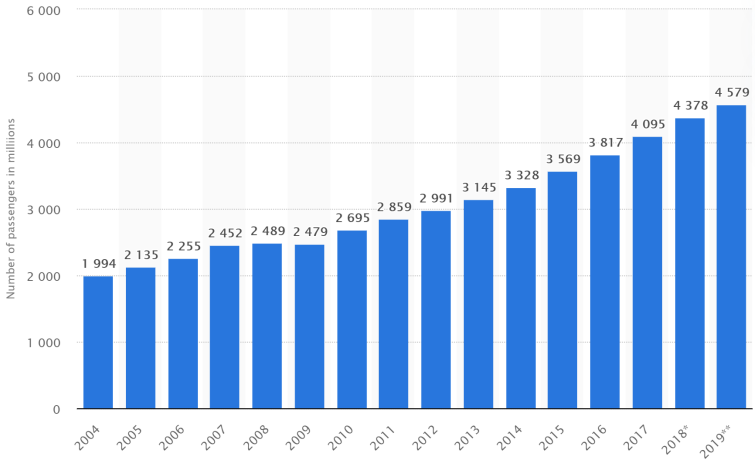

Figure 1.2 Évolution de nombre de passagers entre 2004 et 2019 [2]

1.1.1 L'utilisation de la recherche opérationnelle dans le domaine de transport aérien

Depuis les années 1950, les compagnies aériennes ont eu recours à la recherche opérationnelle pour faire face aux problèmes complexes de planification rencontrés. En 1961, le groupe aérien de recherche opérationnelle AGIFORS a été fondé. Il regroupe une multitude d'associations et des compagnies aériennes.

L'utilisation de la recherche opérationnelle a contribué significativement à la réduction des coûts liés aux opérations du transport aérien. En effet, selon l'association internationale du transport aérien, le coût réel du transport aérien a baissé de plus de 50 % dans les 20 dernières années. En 2018, l'efficacité de l'utilisation du carburant s'est améliorée de 12% par rapport à l'année 2010. En outre, un record d'efficacité de 81,9 % d'occupation de sièges disponibles

a été atteint.
Avec l'expansion ou la fusion des compagnies, ainsi que la croissance du marché, les problèmes sont de taille plus grande, de plus en plus combinatoires et donc plus complexes. Un autre point important qui pousse à l'utilisation de la recherche opérationnelle est la marge étroite du bénéfice des compagnies. Par exemple, il y a une dizaine d'années, une augmentation des coûts d'un certain pourcentage était tolérée. Cependant, actuellement, les compagnies ne peuvent pas se permettre cet excès. De nos jours, l'utilisation de la recherche opérationnelle est devenue cruciale pour la résolution des problèmes d'optimisation dans le domaine aérien.

1.1.2 Processus de planification au sein d'une compagnie aérienne

Plus de 100 articles d'application de la recherche opérationnelle dans le domaine du transport aérien existent dans la littérature. Les sujets les plus répandus portent sur la sécurité aérienne et la planification. La toute première étape de la planification consiste à définir les marchés à servir, la fréquence des vols et la programmation des vols pour répondre à la fréquence déterminée. La deuxième étape consiste à affecter la flotte aux vols prédéfinis. En particulier, il faut déterminer la taille de l'avion à affecter afin de maximiser le profit, en prenant en considération plusieurs facteurs comme la flotte existante. La troisième étape détermine la succession des vols pour chaque avion en respectant les exigences des demandes de maintenance. La quatrième étape détermine le membre ou les membres d'équipage à assigner à chaque vol afin de minimiser les coûts ainsi que de respecter des règles imposées. La dernière étape consiste à affecter des horaires pour les membres d'équipage. À cause de sa complexité, elle est subdivisée en deux sous problèmes : le problème de construction des rotations PCR et le problème de construction d'horaires mensuels des membres d'équipage PCHMME.
Le PCR a comme but de produire, tout en minimisant le coût, un ensemble de rotations qui couvrent une et une seule fois l'ensemble des segments de vols sur un horizon de temps donné en respectant l'ensemble des règles imposées à ce problème. Les rotations générées par le PCR sont par la suite utilisées dans le PCHMME. Ce dernier consiste à assigner ces rotations aux membres d'équipage en prenant en considération d'autres activités telles que les vacances et les formations, afin de construire un horaire mensuel pour chaque membre.
L'objectif du PCHMME varie d'une compagnie à l'autre en fonction de la stratégie adoptée. En général, ces stratégies peuvent être classifiées en trois modes : le premier mode consiste à construire des horaires anonymes. Les employés font des enchères sur un ensemble d'horaires mensuels construits préalablement d'une manière anonyme. Le deuxième mode est la construction des horaires personnalisés. Ce mode consiste à construire des horaires basés sur la maximisation de la satisfaction des préférences des employés en prenant en compte les règles imposées sur chaque employé. Le troisième mode est la construction des horaires

personnalisés en respectant la séniorité.

1.2 Description de la problématique et des objectifs poursuivis

Le présent travail traite un problème de construction des horaires mensuels des membres de cabine d'une compagnie aérienne donnée. Cette compagnie adopte la stratégie de construction des horaires personnalisés. L'objectif est de maximiser la moyenne de satisfaction des membres de cabine.
La compagnie en question subdivise son personnel de cabine en plusieurs catégories. Le tableau 1.1 illustre les catégories qui seront utiles pour décrire la problématique ainsi que le travail effectué.

Tableau 1.1 Exemple de catégories de membres d'équipage de la compagnie étudiée.

Type de la catégorie	Description de la catégorie	Nombre de membres
CM	Catégorie des managers de cabine. Ils sont responsables des membres de cabine dans les avions à large fuselage.	427
CS	Catégorie des superviseurs de cabine. Ils sont responsables des membres de cabine dans les avions à fuselage étroit et de la classe économique dans les avions à large fuselage.	611
FA	Le reste des membres de cabine.	1957

La compagnie résout le problème du personnel de cabine d'une manière séquentielle, catégorie par catégorie, en commençant par la résolution de la catégorie CM. Ensuite, celle de la catégorie CS et finalement, celle de la catégorie FA. Le temps de résolution des catégories des CM et CS est plus court par rapport à celui de la catégorie FA. En effet, le problème traité est le temps de résolution relativement élevé de la catégorie FA.
Avec l'expansion de la compagnie, la taille de cette catégorie devient de plus en plus grande et la résolution prend plus de temps.
L'objectif principal de ce travail est de proposer de nouveaux algorithmes afin d'améliorer le temps de résolution de la catégorie FA. Un deuxième objectif est de permettre à chaque membre de la catégorie FA de rester, dans la mesure du possible, avec le minimum de chefs. Pour ce faire, et en exploitant les similarités qui lient les catégories des chefs avec la catégorie

FA, une méthode qui utilise la solution des chefs afin de générer plus rapidement la solution de la catégorie FA est proposée. Cette méthode est basée sur l'exploration progressive de voisinages d'un ensemble initial de *clusters* construit à partir des solutions des catégories CM et CS. Un cluster représente une succession de rotations.

1.3 Contexte de réalisation du travail

Le travail de recherche s'est effectué dans le cadre d'un plus grand projet au GERAD. Le GERAD est un centre de recherche interuniversitaire créé en 1979 et spécialisé en recherche opérationnelle et en analyse décisionnelle. Ce dernier rassemble un grand nombre d'informaticiens théoriques, spécialistes de méthodes quantitatives en gestion, chercheurs opérationnels et ingénieurs-mathématiciens, issus principalement de l'École Polytechnique, l'Université McGill, l'Université du Québec à Montréal et HEC Montréal.
La compagnie AD OPT a été fondée en 1987 par des professeurs du GERAD. Elle fournit des solutions d'optimisation et de gestion de planification des horaires d'équipage dans le domaine du transport aérien. AD OPT est devenue une division de *Kronos Incorporated* en 2004, et une division d'*IBS Software* en 2019.
Le présent travail est basé sur un logiciel, développé par la compagnie AD OPT, de planification d'horaires mensuels des membres d'équipage. Il porte en particulier sur la spécialisation d'un produit *CRS* pour une compagnie aérienne donnée. *CRS* est un système de planification des horaires mensuels des membres d'équipage. Ce produit de planification d'équipages est articulé sur le logiciel d'optimisation GENCOL. Ce dernier est un solveur de génération de colonnes qui exploite le principe de décomposition de Dantzig-Wolfe, le logiciel CPLEX et d'autres algorithmes efficaces comme la programmation dynamique afin d'obtenir le meilleur résultat possible.

1.4 Structure du mémoire

Ce mémoire est subdivisé en cinq chapitres. Le premier chapitre est une introduction du travail. Le deuxième présente une revue de littérature concernant le problème de planification d'horaires mensuels ainsi que les techniques utilisées dans la résolution. Dans le troisième chapitre, on effectue une étude du solveur existant en présentant le modèle mathématique, l'approche de résolution adoptée ainsi qu'une analyse des instances et des points similaires entre les différentes catégories des membres d'équipage. Le chapitre quatre décrit les méthodes et les algorithmes proposés afin d'atteindre les objectifs du travail. Le chapitre cinq présente une analyse des résultats numériques de l'ensemble des expériences effectuées.

CHAPITRE 2 REVUE DE LITTÉRATURE

Dans ce chapitre, on commence par présenter les travaux existants qui portent sur la stratégie de construction d'horaires personnalisés. Ensuite, on décrit les techniques de résolution les plus pertinentes pour le travail effectué. Finalement, on identifie les principales idées provenant de la littérature qu'on compte utiliser dans le présent travail.

2.1 Construction d'horaires personnalisés

La stratégie de construction d'horaires personnalisés en maximisant un objectif global est plus populaire en Europe et en Asie qu'en Amérique du Nord. Elle consiste à construire des horaires personnalisés basés, en général, sur la maximisation de la moyenne de satisfaction des préférences des employés. Les employés expriment leurs préférences sous la forme d'enchères faites sur les jours de repos, les périodes de repos, les destinations désirées, les destinations à éviter. L'employé donne un score pour chaque choix effectué. La satisfaction d'un employé est alors liée à la somme des scores des choix obtenus dans l'horaire attribué. Les travaux relatifs à ce type de construction sont les plus abondants dans la littérature par rapport aux autres types de constructions.

Les travaux de Marchettini [5] et Glanert [6] sont basés sur la construction des horaires par l'affectation de la tâche de plus haute priorité à l'employé qui a la plus haute priorité. La difficulté de classifier toutes les tâches par ordre de priorité représente un des désavantages de cette méthode.

Les méthodes de Nicoletti [7], Buhr [8], Tingley [9], et Sarra [10] se basent sur une résolution de plusieurs problèmes d'affectations journaliers en affectant les rotations du jour j aux employés disponibles. Avec cette méthode, on ne peut pas avoir une vision globale du problème, ce qui conduit à une perte d'information et donc une détérioration dans la qualité de la solution.

Giafferri et al. [11] ont proposé une méthode à deux étapes. La première étape consiste à construire des horaires initiaux pour chaque employé. Ensuite, une ré-optimisation basée sur une résolution d'un ensemble de problèmes d'affectation journaliers pour améliorer ces horaires est effectuée. Cette ré-optimisation diminue quelques aspects indésirables des méthodes vues précédemment, mais elle ne les élimine pas.

Day et Ryan [12] ont proposé, pour la compagnie *Air New Zealand*, une approche à deux phases. La première phase de résolution de ce problème est l'utilisation d'une heuristique qui génère à priori, un ensemble d'horaires faisables pour chaque employé. Par raison d'équité,

on affecte d'abord les jours de repos, ensuite les rotations. L'objectif de cette heuristique est non seulement la génération des horaires de qualité, mais aussi de construire une matrice de contraintes qui facilite la génération d'une solution entière. Dans la deuxième phase, on utilise la stratégie de branchement *Branch and Bound* afin d'obtenir une solution entière.
Cette approche devient inefficace avec les problèmes de grandes tailles, car le nombre d'horaires générés pour chaque employé devient négligeable par rapport au nombre d'horaires possible.
Gamache et Soumis [13] ont proposé une méthode exacte basée sur la génération de colonnes. Le problème a été modélisé comme étant un problème de programmation linéaire en nombres entiers suivant la décomposition de *Dantzig-Wolfe*. Dans cette modélisation, on trouve un problème maître qui représente les contraintes liantes entre les employés et un sous-problème par employé. Ce dernier est représenté sous la forme d'un réseau. La relaxation linéaire de ce modèle est résolue par le processus de génération de colonnes (décrit plus bas). Ce processus est imbriqué dans un arbre de *Branch and Bound* pour trouver une solution entière.
Gamache et al. [14] ont implémenté la méthode introduite par Gamache et Soumis [13] pour la compagnie *Air France*. En outre, plusieurs stratégies d'accélération ont été mises en place afin de rendre la méthode plus performante. Ce travail a permis d'améliorer le temps de résolution par rapport au produit existant utilisé par la compagnie *Air France* par un facteur de 1000 et a également amélioré la qualité de la solution.
D'autres stratégies de résolutions existent dans la littérature. On trouve entre autres, les algorithmes génétiques par ElMoudani et al. [15]. Ces derniers ont proposé une méthode heuristique bi critère pour générer des horaires personnalisés. Cette méthode a été combinée avec les algorithmes génétiques afin de produire des horaires mensuels de coût minimum en respectant un seuil de satisfaction. La méthode a été testée sur des données industrielles d'une compagnie aérienne de taille moyenne.
D'autres méthodes ont été proposées dans la littérature, on trouve entre autres, la méthode de la programmation par contraintes, adoptée dans les stratégies proposées par Fahle et al. [16] et Sellmann et al. [17], la méthode de la recherche dispersée introduite par Maenhout et Vanhoucke [18].

2.2 Techniques de résolution

2.2.1 Décomposition de Dantzig-Wolfe

La majorité des méthodes citées plus haut utilisent un problème linéaire en nombres entiers (PLNE) afin de modéliser le problème de planification des horaires.
La méthode de décomposition de *Dantzig-Wolfe* [19] est adoptée afin de modéliser les PLNE

de grande taille dont les contraintes se séparent en deux groupes : Le problème suivant illustre la situation :

$$\min_{\boldsymbol{x}} \quad \mathbf{c}^T \boldsymbol{x} \tag{2.1}$$

$$\text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.2}$$

$$\boldsymbol{D}\boldsymbol{x} = \boldsymbol{e} \tag{2.3}$$

$$\boldsymbol{x} \in \mathbb{N}^n \tag{2.4}$$

$\boldsymbol{x}$ est le vecteur des variables ; $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{Z}^{m_1}$, $\boldsymbol{e} \in \mathbb{Z}^{m_2}$, $\boldsymbol{A} \in \mathbb{Z}^{m_1} \times \mathbb{Z}^n$ et $\boldsymbol{D} \in \mathbb{Z}^{m_2} \times \mathbb{Z}^n$
Soit $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ et $\boldsymbol{D}\boldsymbol{x} = \boldsymbol{e}$ deux groupes de contraintes tels que, le groupe $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ est considéré comme étant des contraintes liantes et $\boldsymbol{D}\boldsymbol{x} = \boldsymbol{e}$ comme des contraintes faciles. Alors, si le groupe des contraintes $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ n'est pas considéré, le problème devient facile à résoudre. Les contraintes du groupe $\boldsymbol{D}\boldsymbol{x} = \boldsymbol{e}$ peuvent avoir une structure d'un seul problème ou d'un ensemble de problèmes faciles à résoudre dans le cas où elles sont séparables et ont une structure de blocs angulaire. Chaque bloc peut représenter un problème facile tel qu'un problème de plus court chemin, un problème de flot, ou un problème de sac à dos. On peut aussi se retrouver avec des problèmes de plus court chemin avec contraintes de ressources. Les problèmes faciles qui constituent le groupe des contraintes $\boldsymbol{D}\boldsymbol{x} = \boldsymbol{e}$ constituent des sous-ensembles disjoints qui sont également nommés les sous-problèmes.
Le principe de décomposition de *Dantzig-Wolfe* [19] permet de réécrire ce problème sous la forme d'un autre problème linéaire en nombres entiers qui ne contient que les contraintes équivalentes aux contraintes liantes ($\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$). Ces dernières sont également appelées contraintes globales, car elles impliquent des variables provenant de plusieurs sous-ensembles disjoints.
Au final, un PLNE peut se réécrire sous la forme d'un autre PLNE en conservant seulement les contraintes liantes et une contrainte de convexité par sous-problème.

2.2.2 Méthode de génération de colonnes

Le problème de planification des horaires mensuels est un problème combinatoire. Le PLNE correspondant comporte un très grand nombre de variables.
La méthode de la génération de colonnes présentée à la figure 2.1 permet de réduire le nombre de ces variables et de n'utiliser que des variables qui sont susceptibles d'améliorer la solution. La génération de colonnes débute avec un problème maître restreint (PMR). C'est un problème qui ne comporte que les contraintes liantes et un petit ensemble de variables.
La résolution du PMR produit une solution duale. Cette dernière est utilisée au niveau des sous-problèmes pour générer, dans la mesure du possible, des variables de coût réduit négatif

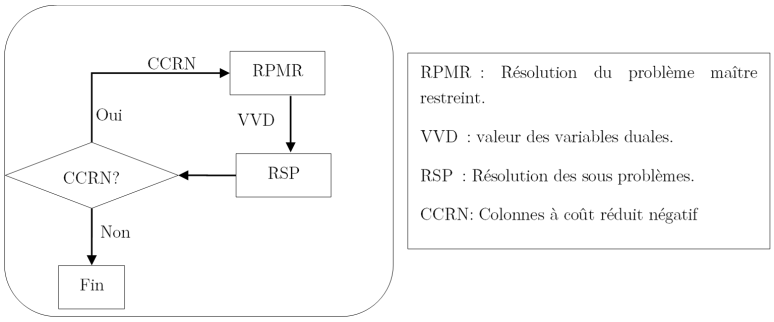

Figure 2.1 Méthode de génération de colonnes.

(dans le cas d'une minimisation). Ces variables seront ajoutées au PMR dans l'itération suivante. Ce processus est itératif et s'arrête une fois les sous-problèmes n'arrivent plus à générer des variables de coûts réduits négatifs. Dans ce cas, les variables non considérées ne peuvent pas améliorer le PMR. À ce niveau, l'optimalité est atteinte et la solution optimale du PMR est la même que la solution optimale du problème initial.

2.2.3 Effets indésirables de la génération de colonnes

La méthode de génération de colonnes souffre de plusieurs faiblesses qui nuisent à sa performance [19]. Parmi ces problèmes, on trouve l'effet *tailling-off* qui représente une convergence très lente. En effet, aux dernières itérations, l'amélioration de la valeur de la fonction objectif devient très petite.
Le deuxième inconvénient est l'effet *heading-in*. Cet effet reflète l'impact négatif des colonnes sans bonnes informations envoyées au problème maître à cause de la mauvaise qualité des valeurs duales. Cet effet est généralement présent dans les premières itérations. On trouve aussi le problème de dégénérescence représenté par l'effet plateau. Il y a aussi l'effet *bang-bang* qui représente l'instabilité des solutions duales qui varient d'une valeur extrême à une autre. La figure 2.2 tirée du livre *Column Generation* [3] résume ces effets indésirables.

2.2.4 La méthode de résolution *Branch and Price*

La méthode du *Branch and Price* (voir Barnhart et al. [20] pour plus de détails) est une méthode de résolution des problèmes en nombres entiers de grandes tailles. Elle est basée sur un arbre de *Branch and Bound*. Au niveau de chaque nœud de l'arbre, un problème linéaire en nombres réels est résolu. Ce dernier représente la relaxation linéaire du problème en nombres entier. La résolution de ce problème se fait avec la méthode de génération des colonnes. Pour le *Branch and Bound*, il faut déterminer une stratégie de branchement afin de choisir le nœud de branchement à résoudre en premier. Il faut également déterminer des règles

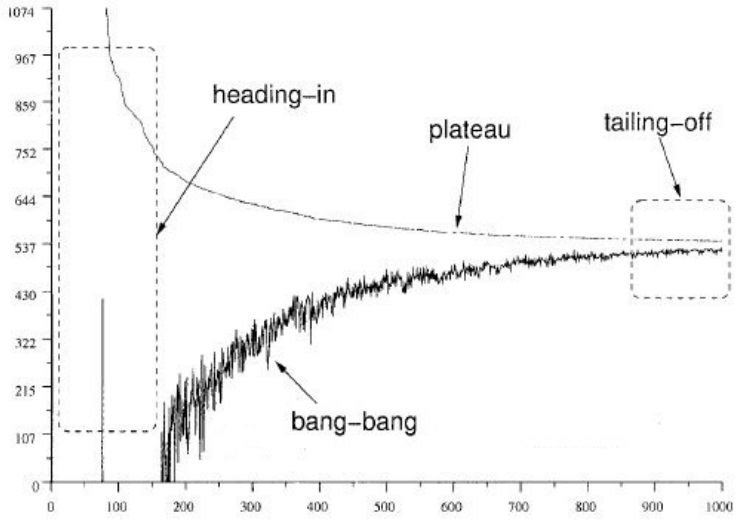

Figure 2.2 Effets indésirables de la génération de colonnes. [3]

de branchements pour créer les nœuds fils. La fixation des colonnes ($CFIX$) et la fixation des arcs ($AFIX$) (voir Savelsbergh [21]) sont parmi les règles de branchements les plus répandues dans la littérature.

2.2.5 Agrégation dynamique de contraintes

Plusieurs problèmes en nombres entiers traités dans la littérature contiennent des contraintes de partitionnements. Le nombre de ces contraintes est élevé ce qui complique davantage la résolution du problème.

L'agrégation dynamique des contraintes est une approche conçue par El Hallaoui et al. [22] afin d'améliorer le temps de résolution de la programmation linéaire et du processus de génération des colonnes standard. L'objectif de cette approche est de réduire la taille du problème maître, en particulier le nombre de contraintes de partitionnement. L'idée est d'agréger ces contraintes en les regroupant dans des sous-ensembles. Un sous-ensemble peut contenir une à plusieurs contraintes de partitionnement. Chaque sous-ensemble correspond à une classe d'équivalence et l'union de ces sous-ensembles donne une partition Q de l'ensemble des contraintes de partitionnement nommé W.

Cette méthode utilise une solution initiale comme solution de départ. Cette solution peut être trouvée par une heuristique et représente un ensemble de chemins. Les sous-ensembles de tâches de la partition Q sont obtenus à partir de la relation d'équivalence suivante : étant donné un ensemble de chemins C, on dit que deux tâches w_1 et w_2 sont équivalentes par rapport à C, si tous les chemins de C couvrent ces deux tâches ou aucune de ces tâches.

Des classes d'équivalences $l \in L$ sont créées pour les tâches de W par rapport à C. La parti-

tion Q est alors définie par : $Q = \{W_l : \ l \in L\}$. Cette partition donne un problème maître restreint agrégé (PMRA) dans lequel, chaque sous-ensemble de contraintes W_l est remplacé par une seule contrainte. En résolvant ce problème, une solution primale ainsi qu'une solution duale agrégée est obtenue. Cette solution doit être désagrégée avant qu'elle soit transmise au sous-problème. Pour ce faire, il faut résoudre le système suivant : $\sum_{w \in W_l} \alpha_w = \gamma_l \ \forall l \in L$ avec γ_l est la variable duale agrégée qui correspond au sous-ensemble de contraintes W_l et α_w est la variable duale désagrégée qui correspond à la tâche w. Une solution évidente de ce système est $\alpha_w = \frac{\gamma_l}{|w_l|}$. Cette solution n'est pas particulièrement intéressante. EL Hallaoui et al. [22] ont ajouté des contraintes duales à ce système pour plus d'efficacité.

Le sous-problème est résolu après l'obtention des variables duales désagrégées. L'optimalité est atteinte si cette résolution ne génère aucune colonne de coût réduit négatif. Sinon, dépendamment de ces colonnes, on décide soit de changer la partition Q si aucune colonne compatible n'est générée ou si le coût réduit de ces colonnes est très proche du zéro, ou soit de continuer avec la même partition Q en ajoutant au PMRA les colonnes compatibles avec Q.

Le fait de continuer avec la même partition est considéré comme une itération mineure, pendant que le changement de partition est considéré comme itération majeure.

Deux méthodes étaient adoptées pour changer de partition. La première est utilisée quand la solution du PMRA n'améliore pas la valeur de la fonction objectif ou quand la solution inclut des variables artificielles. L'ensemble C des chemins est augmenté par l'ajout des colonnes incompatibles produites dans les itérations mineures. Ensuite, une nouvelle partition Q est définie à l'aide de la relation d'équivalence. Cette méthode prend en considération toutes les colonnes incompatibles sans aucun critère ce qui peut, d'une part ré-augmenter la taille du PMRA et d'autre part s'éloigner de la solution initiale. En effet, le fait de prendre toutes les colonnes incompatibles sans se limiter aux colonnes dans le voisinage de la solution initiale peut nuire au temps de résolution, car on peut s'éloigner de la solution dès le début de la résolution.

Pour faire face à ces problèmes, des améliorations ont été apportées à la méthode originale. El Hallaoui et al. [23] ont introduit une version améliorée, nommée : agrégation dynamique des contraintes avec multiples phases (MPDCA). Cette version introduit la notion du degré d'incompatibilité d'une colonne par rapport à la partition courante pour bien contrôler la mise en place de la nouvelle partition. L'objectif est de générer les colonnes qui ont le degré d'incompatibilité minimal par rapport à la partition courante. Ceci se fait au niveau du sous-problème à l'aide d'une ressource quantitative qui cumule le nombre d'incompatibilités tout au long de construction du chemin. Soit k ce degré d'incompatibilité. En premier lieu, l'objectif est de générer des chemins de coût réduit négatif avec un degré d'incompatibilité $k = 0$.

Si c'est le cas, la partition demeure inchangée et les colonnes (correspondantes aux chemins générés) sont envoyées au problème maître restreint agrégé (PMRA). Sinon, la valeur de k est incrémentée et une deuxième résolution du sous-problème est effectuée. Si la résolution génère des colonnes à coût réduit négatif, la partition change, mais les classes d'équivalences ne changent pas beaucoup, car, avec un petit k, les colonnes incompatibles restent dans un voisinage proche aux colonnes de la partition courante. Sinon, si on n'arrive pas à générer des colonnes de coût réduit négatif, le degré d'incompatibilité augmente jusqu'à atteinte d'une borne supérieure préalablement imposée sur le degré d'incompatibilité. Ceci permettra la mise en place d'une nouvelle partition en limitant le nombre de désagrégations des classes d'équivalences par rapport à la partition ultérieure sans trop augmenter la taille de PMRA. Cette méthode s'intéresse en grande partie à la taille et au temps de résolution du problème maître.

Afin d'améliorer davantage la performance de la méthode DCA, El Hallaoui et al. [24] ont présenté une nouvelle version de DCA qui porte un intérêt particulier au sous-problème. Cette version améliorée, nommée agrégation bi-dynamique des contraintes (*BDCA*) introduit la notion d'un arc incompatible. Cette stratégie permet d'éliminer un ensemble d'arcs incompatibles au niveau du sous-problème ce qui produit un sous-problème de taille réduite. Les arcs incompatibles sont les arcs qui causent une désagrégation des classes d'équivalences. La résolution du sous-problème se fait en considérant un sous-ensemble d'arcs incompatibles et en éliminant le reste. Cette élimination est déterminée par un paramètre RL qui représente le niveau de réduction. Ce paramètre varie dans l'intervalle [0,1] et peut changer de valeur d'une itération à l'autre. Si RL égale à 0 le sous-problème est complètement réduit. Sinon, si RL égale à 1, le sous-problème est complet, c'est-à-dire, aucun chemin incompatible n'est omis.

L'objectif de la réduction est de garder les classes d'équivalence qui ont les meilleures valeurs duales agrégées, inchangées. Ceci réduit la taille du sous-problème. La valeur de RL détermine le nombre de classes d'équivalence à garder si on n'arrive pas à générer des colonnes de coût réduit négatif.

2.2.6 La relation entre les techniques de résolution présentées et le présent travail

Le produit utilisé pour résoudre le problème en nombres entiers qui correspond au PCHMME est basé sur la méthode de génération de colonnes ainsi que la méthode Branch and Price qui ont été présentées.

L'objectif de ce travail est d'améliorer le produit en utilisant des idées de DCA et MPDCA que nous venons de présenter. DCA, qui réduit le nombre de contraintes dans le problème

maître, a été développé pour les problèmes de partitionnement (des contraintes binaires avec les b_i=1). Il ne s'applique pas facilement au cas plus général des problèmes de recouvrement généralisé (des contraintes binaires avec les b_i entiers).
Par contre, nous utiliserons des idées de la version améliorée, MPDCA, qui travaille aussi sur les sous-problèmes. Nous utiliserons la contrainte de ressource sur le degré d'incompatibilité pour restreindre le domaine de recherche dans le sous-problème. L'ajustement dynamique de cette contrainte qui augmente graduellement le domaine de recherche devrait permettre de réduire fortement le temps de résolution des sous-problèmes dans les premières itérations de la génération de colonnes et d'atteindre une bonne précision dans les dernières.

CHAPITRE 3 ANALYSE DE L'EXISTANT

La structure de ce chapitre est la suivante : tout d'abord, on présente le modèle mathématique existant, la description du réseau du sous-problème et l'approche de résolution adoptée par la compagnie. Ensuite, on effectue une analyse des données d'entrée et de la taille du problème.

3.1 Modèle mathématique

Le problème de construction d'horaires mensuels est modélisé suivant la méthode de décomposition de *Dantzig-Wolfe* décrite dans le chapitre 2. Les contraintes pourraient être classées dans deux groupes : des contraintes globales et des contraintes locales. Pour ne pas alourdir le texte avec des informations non nécessaires et pour mieux comprendre le travail présenté, on se limitera à l'exposition d'une version simplifiée du modèle.

3.1.1 Contraintes globales

On définit les ensembles suivants :

- I : l'ensemble des membres d'équipage.
- Ω_i : l'ensemble des horaires mensuels possibles pour un membre d'équipage $i \in I$ en respectant ses contraintes locales.
- P : l'ensemble de toutes les rotations.
- R : l'ensemble de toutes les tâches de réserve.
- $x_j : \begin{cases} 1 & \text{si l'horaire } j \text{ est affecté au membre d'équipage } i. \\ 0 & \text{sinon.} \end{cases}$

Les contraintes globales sont liées aux règles suivantes :

- Contraintes affectant les membres d'équipage : ces contraintes imposent qu'un et un seul horaire mensuel doive être assigné à chaque membre d'équipage :

$$\sum_{j \in \Omega_i} x_j = 1 \qquad \forall i \in I \tag{3.1}$$

- Contraintes imposées sur la demande de chaque rotation (et tâche de réserve) : pour chaque rotation, on a une demande de membres d'équipage supérieures ou égale à 1. Cette contrainte modélise le respect de cette demande ainsi que le non-respect de la demande à l'aide des variables d'écart. Une contrainte sur la somme de ces variables d'écart sera décrite par la suite.

Soit :

— α_p : la demande de la rotation p.

— $a_{jp} : \begin{cases} 1 & \text{si la rotation } p \text{ est couverte par l'horaire } j. \\ 0 & \text{sinon.} \end{cases}$

— y_p : variable d'écart qui peut prendre une valeur supérieure ou égale à zéro et qui représente la non-couverture de la demande de la rotation p.

$$\sum_{i \in I} \sum_{j \in \Omega_i} a_{jp} x_j + y_p = \alpha_p \qquad \forall p \in P \cup R \tag{3.2}$$

— Contraintes sur le nombre de crédits non couverts permis pour les rotations (et les tâches de réserve) : le nombre de crédits non couverts des demandes des rotations ne doit pas dépasser la valeur d'un paramètre β. Cette dernière peut prendre la valeur zéro ou des valeurs positives selon des paramètres spécifiés par le planificateur.
Soit v_p le crédit qui correspond à la rotation (ou à la tâche de réserve). On a :

$$\sum_{p \in P \cup R} v_p y_p \leq \beta \tag{3.3}$$

— Contraintes sur la couverture des langues : la demande d'une langue dépend de l'équipement et la destination. Un exemple de cette demande est : 'Airbus' JNB 1 'AF' -1 1 ; cela est expliqué comme suit : il faut au moins un membre d'équipage qui parle la langue africaine dans chaque équipement Airbus à destination de Johannesburg (JNB). La présente contrainte modélise la couverture de la demande de chaque langue durant la période de planification.
Soit :

— a_{ij}^l : le nombre de fois que la demande de la langue $l \in L$ est satisfaite par le membre d'équipage $i \in I$ dans l'horaire $j \in \Omega_i$. Ce paramètre prend en considération les rotations que l'horaire mensuel contient, les stations desservies par chaque rotation et le type d'équipement utilisé.

— v_l : le nombre minimum des membres d'équipage qui doivent parler la langue l.

— y_l : variable d'écart qui représente la non-couverture de la langue l.

$$\sum_{i \in I} \sum_{j \in \Omega_i} a_{ij}^l x_j + y_l \geq v_l \qquad \forall l \in L \tag{3.4}$$

— Contrainte sur les jours de congés en excès : cette contrainte est imposée sur le nombre de jours de congé en excès. Si un horaire choisi dépasse le nombre de jours de congé toléré et prédéterminé par le planificateur, alors la différence est considérée comme étant des jours en excès. Soit a_j^{ex} ce nombre de jours de congés en excès pour l'horaire j. La somme des jours en excès correspond à une variable y^{ex}. La contrainte est modélisée

comme suit :

$$\sum_{i \in I} \sum_{j \in \Omega_i} a_j^{ex} x_j - y^{ex} = 0 \tag{3.5}$$

— Contraintes d'intégralité :

$$x_j \in \{0, 1\} \qquad \forall j \in \Omega_i \; \forall i \in I \tag{3.6}$$

$$y_p, \quad y_l, \quad y^{ex} \geq 0 \text{ et entiers} \qquad \forall p \in P \cup R, \; \forall l \in L \tag{3.7}$$

3.1.2 Fonction objectif

Plusieurs objectifs sont modélisés à l'aide de cette fonction. Des objectifs pour minimiser le nombre de violations des contraintes globales (présentées plus haut) et un objectif pour maximiser la moyenne des satisfactions des employés. Ces objectifs sont pondérés selon leur importance. Voici comment ils sont modélisés :

— Le nombre total de violations des exigences des langues : le respect de la couverture des langues est représenté par une contrainte globale. Une variable d'écart y_l correspond au nombre manquant de locuteurs de la langue l. Cet objectif sert à minimiser le nombre total de la non-couverture des langues. Il est pondéré avec le poids $W_1 \geq 0$ et se modélise comme suit :

$$W_1(\sum_{l \in L} c_l y_l) \tag{3.8}$$

La priorité de couverture diffère d'une langue à l'autre, d'où le coût c_l associé à chaque langue.

— Le coût total des pénalités des jours de congés en excès : cet objectif minimise le coût total des pénalités sur les jours de congés en excès. Il est pondéré par le poids $W_2 \geq 0$. Soit c^{ex}, le coût de pénalité pour chaque jour off en excès et y^{ex} la variable représentant le nombre de jours off en excès. L'objectif se modélise comme suit :

$$W_2(c^{ex} y^{ex}) \tag{3.9}$$

— La somme de la satisfaction des préférences des employés. Cet objectif maximise la satisfaction totale des employés et est ponderé par le poids $W_3 \geq 0$. Chaque horaire d'un employé j est associé à une valeur c_j qui représente la satisfaction de cet employé pour l'horaire j. L'objectif se modélise de la manière suivante :

$$- W_3(\sum_{i \in I} \sum_{j \in \Omega_i} c_j x_j) \tag{3.10}$$

Le modèle s'écrit alors comme suit :

$$\min_{x} \quad W_1(\sum_{l\in L} c_l y_l) + W_2(c^{ex} y^{ex}) - W_3(\sum_{i\in I}\sum_{j\in\Omega_i} c_j x_j)$$

$$\begin{aligned}
\text{s.t.} \quad \sum_{j\in\Omega_i} x_j &= 1 && \forall i \in I \\
\sum_{i\in I}\sum_{j\in\Omega_i} a_{jp} x_j + y_p &= \alpha_p && \forall p \in P \cup R \\
\sum_{p\in P} v_p y_p &\leq \beta \\
\sum_{i\in I}\sum_{j\in\Omega_i} a^l_{ij} x_j + y_l &\geq v_l && \forall l \in L \\
\sum_{i\in I}\sum_{j\in\Omega_i} a^{ex}_j x_j - y^{ex} &= 0 \\
x_j &\in \{0,1\} && \forall j \in \Omega_i \; \forall i \in I \\
y_p, \quad y_l, \quad y^{ex} &\geq 0 \text{ et entiers} && \forall p \in P \cup R, \; \forall l \in L
\end{aligned}$$

3.1.3 Contraintes locales

Les contraintes locales représentent les règles qui concernent chaque membre indépendamment du reste de l'équipage. Ces contraintes modélisent des règles imposées par l'organisation internationale de l'aviation civile, par la convention collective et aussi les tâches pré-assignées. En voici quelques exemples :

- Un minimum de jours de congé dans un horizon de huit, quatorze, vingt-huit et quatre-vingts jours.
- Une limite d'heures de crédit dans un horizon de temps de 28 jours.
- Une durée minimum de repos avant et après un vol long-courrier.

3.2 Approche de résolution

3.2.1 Processus général

Les membres d'équipages sont classés dans des catégories selon leurs séniorités. La résolution se fait d'une manière séquentielle, catégorie par catégorie en commençant par la catégorie la plus séniore. En effet, la catégorie la plus séniore (managers de cabine CM) est résolue en premier, et à la fin du processus de résolution, on trouve la catégorie des agents de bord FA. Les catégories sont résolues d'une manière similaire. Le même processus de résolution est appliqué d'une manière itérative pour toutes les catégories. La résolution se fait par phase. Au début, on retrouve une phase de pré-résolution. L'objectif de cette phase est de calculer

le meilleur et le pire score de chaque employé en se basant sur le réseau de l'employé (ces réseaux sont décrits dans la section 3.2.2). Ces deux paramètres (meilleur et pire score) seront par la suite utilisés dans le calcul de satisfaction de l'horaire de l'employé.
La deuxième phase consiste à résoudre un problème nommé *problème-cadre.* L'objectif de cette résolution est de déterminer la faisabilité du problème. Plusieurs contraintes du problème sont omises. Les contraintes restantes sont des contraintes qui affectent la faisabilité du problème. Ces contraintes sont (3.1), (3.2) et (3.3). Il faut spécifier que les contraintes d'intégralité sont relaxées durant la résolution du problème-cadre.
Si le problème-cadre est non réalisable, un rapport de non-faisabilité est généré et la résolution sera arrêtée. Sinon, la résolution continue et un problème contenant l'ensemble des contraintes sera résolu. Ce problème est nommé *le problème de satisfaction,* car l'objectif principal de ce problème est de maximiser la moyenne des satisfactions des employés. Contrairement au problème-cadre, ce problème sera résolu en nombres entiers avec la méthode de *Branch and Price*. Cette dernière utilise deux stratégies de branchement : la première est *Cfix*. Elle consiste à fixer une colonne pour un employé donné. La deuxième est *Tsplit*. Elle permet de fixer seulement une partie de la solution (quelques tâches) pour un employé donné.

3.2.2 Description du réseau du sous-problème

La résolution du problème-cadre et du problème de satisfaction est basée sur la stratégie de génération de colonnes décrite ci-dessus. Chaque employé correspond à un sous-problème. Ce dernier est décrit par le réseau montré dans la figure 3.1. Les tâches du problème sont représentées sur les arcs. Le réseau est structuré de telle sorte que les rotations soient dans la partie inférieure et les activités qui correspondent aux jours non travaillés soient dans la partie supérieure du réseau. Voici une description des arcs du réseau (on présente également d'autres précisions qui sont utiles pour mieux le comprendre) :

- Les activités qui correspondent aux jours non travaillés sont subdivisées en deux types : les jours de repos et les jours libres.
- Les jours libres sont des périodes demandées par la compagnie et qui sont liées à des règles de la convention collective.
- Arc de lien descendant : il lie un arc qui représente un jour non travaillé avec un autre représentant une tâche de travail. On le décrit comme descendant car les activités qui correspondent aux jours non travaillés sont modélisées en haut du réseau et les tâches de travail sont modélisées en bas du réseau.
- Arc de lien montant : il lie un arc qui représente une tâche de travail avec un autre représentant un jour non travaillé. On le décrit comme montant car les tâches de travail sont modélisées en bas du réseau et les activités qui correspondent aux jours

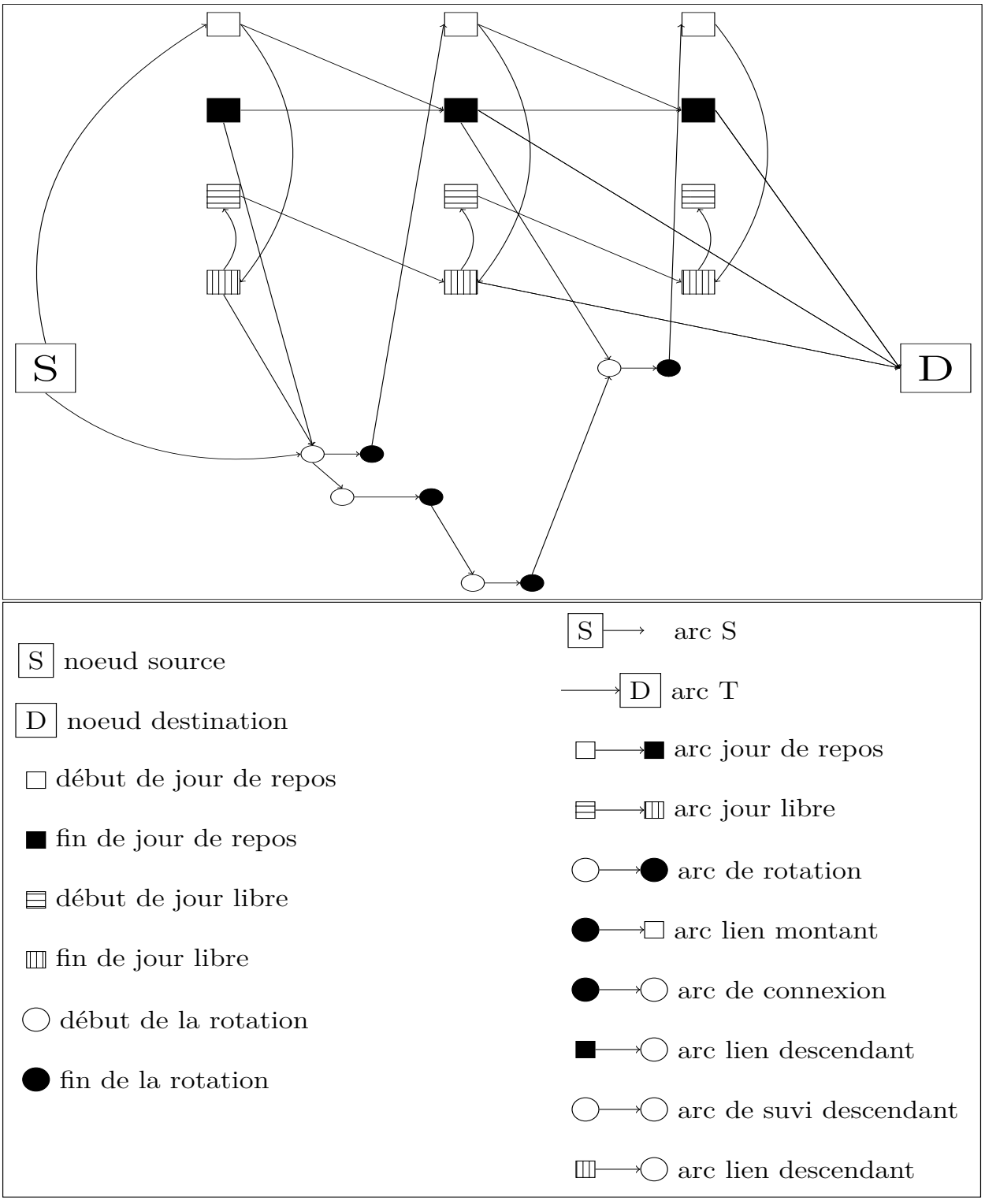

Figure 3.1 Description du réseau du problème

non travaillés en haut du réseau.

— Arc de rotation : il représente une tâche de travail.

— Arc de suivi descendant : il représente tout inter-arc qui lie des nœuds de début des arcs de rotation.
— Les arcs qui représentent les rotations et qui sont dans une même journée sont positionnés d'une manière verticale. L'arc le plus haut représente la rotation qui débute le plus tôt. Pour cette raison, on appelle les arcs de suivi des arcs de suivi descendants.
— Les arcs de type rotation peuvent également représenter d'autres tâches de travail comme les tâches de réserve et les activités de formation.
— Arc de connexion : il lie deux arcs de rotation. Cet arc permet d'effectuer deux rotations successives.

3.3 Analyse des données d'entrée et taille du problème

On détermine la taille d'une catégorie par les critères suivants : le nombre d'employés, la taille du réseau des sous problèmes, le nombre de rotations et le crédit total à couvrir. Le tableau 3.1, représente un exemple de la taille des catégories CS et FA d'une instance donnée.
La difficulté de ce problème est liée principalement à deux raisons. La première dépend de la taille du problème et sa nature combinatoire. En effet, une dizaine de milliers des contraintes et des millions de millions de variables définissent le problème pour une instance donnée. La deuxième difficulté est liée à la méthode de *Branch and Price* adoptée pour la résolution. Cette méthode est basée sur la méthode standard de génération des colonnes. Cette dernière demande plusieurs stratégies d'accélération et l'ajustement de beaucoup de paramètres pour les problèmes de grande taille.
La taille de la catégorie FA est plus grande par rapport à celle des catégories des chefs et donc, son temps de résolution est plus élevé. On a effectué une étude sur les données d'entrées des catégories des chefs et des FA et les règles qui les affectent afin d'analyser le niveau de similarité entre ces catégories. Pour ce faire, on a développé des scripts pour analyser les fichiers d'entrées et de sorties ainsi que les règles adoptées pour chaque catégorie. On a

Tableau 3.1 Exemple de taille des instances CS et FA.

	CS	FA
Nombre de nœuds du réseau	10 463	20 569
Nombre d'arcs du réseau	36 778	75 802
Nombre de ressources utilisées dans le réseau	31	31
Nombre de contraintes locales (dans un sous-problème)	147	138
Nombre de demandes de rotations à couvrir	4923	17359
Nombre de crédits à couvrir (en heure)	46 731	188 112
Nombre d'employés :	580	1867

remarqué les informations suivantes :

- 87,31% de rotations des chefs réapparaissent avec la même structure pour la catégorie des FA.
- La moyenne des heures de travail des catégories des chefs est 112,3 heures par mois et la moyenne des heures de travail de la catégorie FA est de 118,5 heures par mois.
- Le nombre de jours non travaillés moyen est de 13,37 pour les catégories des chefs et de 12,18 pour la catégorie des FA.
- Les catégories des chefs et des FA ont 110 règles similaires ce qui représente plus de 97,42% de règles imposées sur les chefs.

À partir de cette analyse, on peut croire que les résultats de chefs pourraient être exploités dans la résolution de la catégorie des FA afin de réduire son temps de résolution.

CHAPITRE 4 MÉTHODE D'EXPLORATION DE VOISINAGE

Dans ce chapitre, on commence par décrire la méthode d'exploration de voisinage (MEV), son utilité ainsi que ses différentes versions. Ensuite, on décrit les stratégies proposées pour construire un ensemble initial de clusters.

4.1 Introduction de la MEV

La méthode d'exploration de voisinage est un algorithme imbriqué dans l'approche de programmation dynamique utilisée pour résoudre le problème de plus court chemin avec contraintes de ressources. Ce dernier consiste à trouver l'ensemble des chemins pareto-optimaux entre le nœud source et le nœud destination sur un réseau donné, qui satisfont les contraintes sur les ressources. Une ressource est une quantité qui varie durant la construction du chemin suivant une fonction appelée fonction de prolongation. Cette dernière est définie pour chaque arc du réseau et concerne toutes les ressources. Elle sert à calculer, pour chaque arc, les nouvelles quantités des ressources pour chaque chemin après avoir traversé l'arc en question. Une contrainte de ressource est donnée sous forme d'une fenêtre spécifiée par un intervalle sur les nœuds du réseau. Une fenêtre d'intervalle limite les valeurs qu'une ressource peut prendre quand le chemin atteint un nœud donné. Le chemin est éliminé si ces valeurs de ressources ne respectent pas leurs contraintes de ressources correspondantes.
La programmation dynamique est basée sur deux étapes principales : la prolongation d'un chemin sur les arcs adjacents à un nœud et la stratégie de dominance. Cette dernière est utilisée pour garder seulement les chemins pareto-optimaux. Le processus de prolongation et de dominance se répète jusqu'à ce que tous les chemins qui ne sont pas éliminés, atteignent le nœud destination.
La taille des réseaux du problème de planification des horaires mensuels des membres d'équipages de la compagnie aérienne traitée est grande. Durant chaque itération de génération de colonnes, il faut résoudre un nombre élevé de PPCCCR, car on a un réseau pour chaque membre d'équipage. La résolution ne peut pas alors être faite d'une manière optimale dans des temps raisonnables. Par conséquent, au niveau de l'implémentation de la dominance, on garde seulement les chemins pareto-optimaux pour un sous-ensemble de ressource. Alors, les chemins éliminés ne seront pas forcément inutiles si on prend en considération toutes les ressources.
Même avec cet aspect heuristique de la dominance, le processus de génération de colonnes consomme la plus grande part du temps de résolution. Pour cette raison, on propose la mé-

thode d'exploration de voisinage pour réduire le temps de résolution global en réduisant le temps du processus de la génération de colonnes, en particulier le temps de résolution du PPCCCR.

L'objectif de la MEV est d'accélérer la résolution du PPCCCR et de diminuer le nombre de colonnes générées sans bonnes informations, qui seront envoyées au problème maître. Une colonne sans bonne information est une colonne qui ne contribue pas à l'amélioration de la valeur de la fonction objectif après la résolution du problème maître dans une itération donnée.
En effet, au lieu d'une résolution exhaustive du PPCCCR, on propose une méthode de résolution itérative, basée sur l'exploration progressive du voisinage d'un ensemble initial de clusters. Un cluster est un regroupement de tâches de type rotation. Les caractéristiques d'un cluster seront présentées ultérieurement.
La MEV sera appliquée pour résoudre les instances de la catégorie FA et l'ensemble initial de clusters sera construit à partir de solutions des instances de la catégorie des chefs. La section 4.3 décrit le processus de construction de cet ensemble...

4.2 Description de la MEV

Pour mieux expliquer la MEV, on définit les notions suivantes : on dit qu'un chemin a brisé un cluster (ou a effectué un brisement d'un cluster) s'il ne parcourt pas l'ensemble des arcs qui correspondent aux rotations contenues dans le cluster. En d'autres termes, pour un cluster C donné, un brisement a lieu si le chemin parcourt la rotation t_i du cluster sans parcourir la rotation t_{i-1} du même cluster (avec $i \neq 1$).
La distance d'un chemin par rapport à l'ensemble initial de clusters est calculée progressivement au cours de sa construction. Elle représente la somme des brisements des clusters au cours de la prolongation d'un chemin sur le réseau.
La valeur de la distance est mise à jour quand le chemin est prolongé sur un arc de type rotation. La valeur de cette distance est modélisée par une ressource nommée RMEV. On définit une contrainte sur la ressource RMEV pour garder seulement les chemins qui ont une distance inférieure à la distance maximale tolérée, DMT, qu'on spécifie à priori. L'implémentation du cœur de la méthode se fait au niveau de l'algorithme de résolution du PPCCCR. Deux versions de la méthode sont proposées : une version statique et une version dynamique. La différence entre les deux versions apparaît principalement au niveau de la détermination de la valeur de la DMT. La version statique, on fixe une valeur pour la DMT qui restera statique durant la résolution de tous les sous-problèmes qu'on aura besoin de résoudre du-

rant le processus de la génération de colonnes. La version dynamique, quant à elle, permet de spécifier la valeur de la DMT avant la résolution de chaque sous-problème. Plus de détails concernant la version dynamique et son utilité seront présentés après l'introduction de la partie commune des deux versions.

4.2.1 Implémentation de la MEV

Pour expliquer l'implémentation du cœur de la MEV, on introduit les informations suivantes :

- Sur chaque arc de type rotation, on marque préalablement l'information du cluster auquel cette rotation appartient.
- Les valeurs des paramètres de la MEV sont spécifiées avant la résolution du PPCCCR (fixes pour la version statique et variables pour la version dynamique).
- Les paramètres de la MEV sont :
  - NeighborhoodResCurVal : le nombre de brisements compté par la ressource (la distance courante d'un chemin donné par rapport à l'ensemble initial de clusters). Cette valeur est initialisée à zéro.
  - NeighborhoodResUBCurrVal : représente la valeur de la borne supérieure de la contrainte de ressource (distance maximale tolérée par rapport à l'ensemble initial des clusters).
  - D'autres paramètres qui concernent seulement la version dynamique seront introduits par la suite.

La résolution du PPCCCR utilise un algorithme d'étiquetage pour encoder les chemins. On va alors utiliser les termes *label* ou étiquette pour exprimer un chemin au cours de sa prolongation.

La ressource RMEV compte le nombre de brisements par rapport à l'ensemble initial de clusters pour chaque label prolongé. Ensuite, le test suivant est appliqué : si NeighborhoodResCurVal > NeighborhoodResUBCurrVal, alors, le label est rejeté. (La prolongation débute par le label qui contient seulement le nœud source, par la suite, le label sera prolongé en parcourant les arcs possibles pour produire d'autres labels et ainsi de suite jusqu'à ce que chaque label atteint le nœud destination ou se rejette au cours de la prolongation à cause d'une contrainte de ressource.)

Pour ce faire, la méthode compare le dernier arc de type rotation (ou l'arc S) visité avec l'arc de type rotation courant (ou l'arc T). Rappelant que le nœud de départ des arcs S est le nœud source du réseau et le nœud d'arrivé des arcs T est le nœud destination du réseau.

Pour chaque prolongation, le nombre de brisements est modélisé sous la forme de consommation de ressource RMEV. On parle d'une consommation de la ressource comptant le nombre de brisements. Les chemins générés avec la MEV seront alors à une distance inférieure ou

égale à la valeur de la distance maximale tolérée (DMT). Les cas suivants illustrent les types de consommation de ressource traités : soient C_i et C_j deux clusters de l'ensemble initial de clusters. Soit t_i^k et t_j^m deux tâches de type rotation telles que : $t_i^k \in C_i$ et $t_j^m \in Cj$.

- Cas pour une consommation nulle :
  - Prolongation d'un arc de type S vers le premier arc de type rotation d'un cluster C_i.
  - Prolongation de la dernière rotation du cluster C_i vers un arc de type T.
  - Prolongation de deux rotations successives du même cluster (t_i^k vers t_i^{k+1}).
  - Prolongation de la dernière rotation d'un cluster vers la première rotation d'un autre cluster (t_i^k vers t_j^1 avec $k = |C_i|$ et $i \neq j$).
- Cas pour une consommation d'une unité :
  - S vers t_i^k avec $k \neq 1$.
  - t_i^k vers t_j^1 avec $k \neq |C_i|$ et $i \neq j$.
- Cas pour une consommation de deux unités :
  - t_i^k vers t_j^m avec $i \neq j$, $k \neq 1 \neq |C_i|$, $i \neq j$, $m \neq 1 \neq |C_j|$.

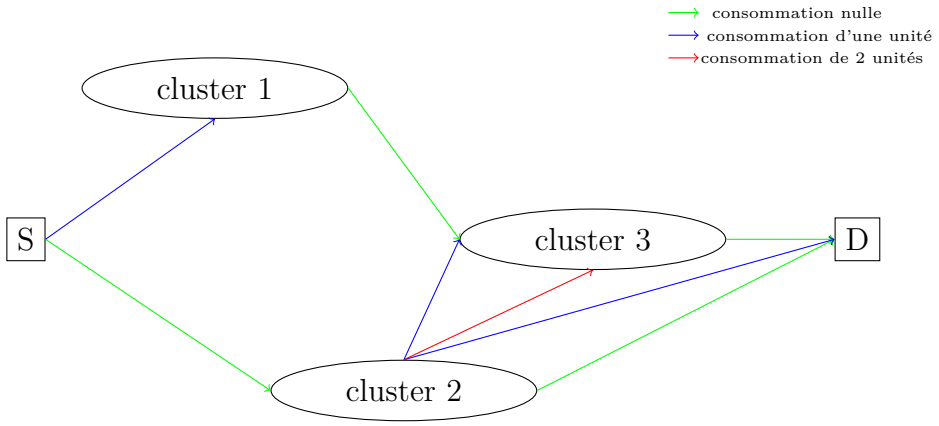

Figure 4.1 Exemples de quelques types de consommation de la ressource

La figure 4.1 représente quelques exemples pour les différents cas de consommation de ressource traités.

4.2.2 Version dynamique de la MEV

En plus des paramètres définis plus haut, la version dynamique en prend d'autres en considération. La valeur NeighborhoodResUBCurrVal doit être spécifiée avant la résolution de chaque PPCCCR. Le voisinage est déterminé suivant cette valeur. Une valeur petite implique un voisinage plus restreint et une valeur plus grande implique un voisinage plus large. Contrairement à la version statique dans laquelle cette valeur prend une valeur fixe qui ne change pas avec l'évolution de la résolution, la version dynamique attribut à cette valeur un

ensemble de valeurs qui varient entre une valeur minimale NeighborhoodResUBMinVal, et une valeur maximale NeighborhoodResUBMaxVal. Le changement de cette valeur se fait quand aucun chemin ne peut être généré dans le voisinage courant. On élargit alors le voisinage en incrémentant la valeur courante de la borne supérieure (NeighborhoodResUBCurrVal) jusqu'à atteinte de la borne maximale. La figure 4.2 représente le processus itératif de la version dynamique de la méthode d'exploration de voisinage. L'initialisation des paramètres de la méthode se fait durant l'appel au modèle de génération de colonnes. Elle n'apparaît alors pas dans la figure.

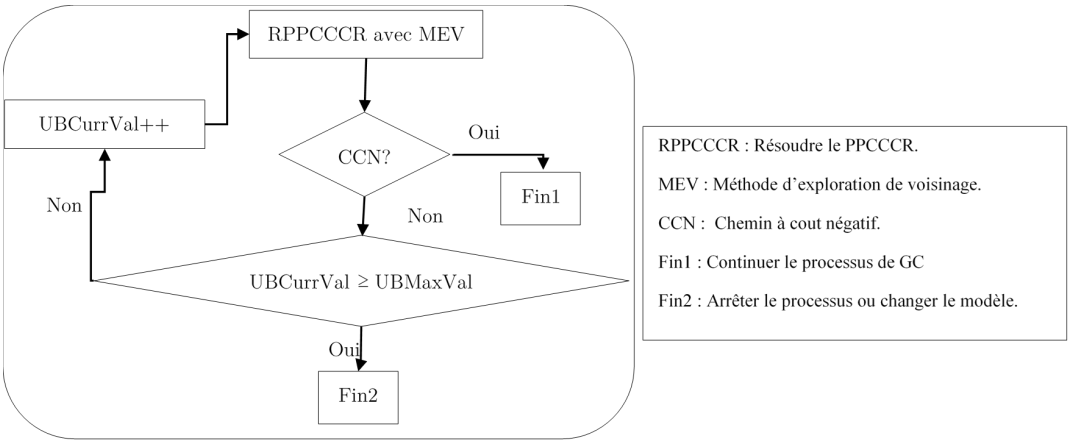

Figure 4.2 La version dynamique de la MEV

4.2.3 Stratégies d'accélération de la méthode d'exploration de voisinage

Plusieurs variantes et stratégies d'accélération ont été proposées afin d'améliorer l'efficacité de la méthode d'exploration de voisinage. Dans cette section, on en présente quelques-unes :

— Critère de changement de voisinage (NeigheborhoodchangeCriterea) : À partir des résultats préliminaires, on a remarqué que le nombre de colonnes générées et l'amélioration de la valeur de la fonction objectif dans les dernières itérations de la génération de colonnes dans un voisinage donné sont très petits. Pour cette raison, on a introduit un critère de changement de voisinage après chaque m itérations : si la valeur de la fonction objectif ne s'améliore pas avec un taux qu'on nomme Neighborhoodchange-Criterea alors on change de voisinage.

— Ajout du paramètre NeigheborhoodResUBStepVal : Le changement de voisinage de la MEV se faisait par l'augmentation de la borne supérieure de la ressource avec une unité. On constate que le nombre d'itérations est élevé pendant que l'amélioration de la valeur de la fonction objectif n'est pas importante. Un paramètre Neigheborhoo-

dResUBStepVal a été alors introduit dans le but de diminuer le nombre d'itérations et d'améliorer plus rapidement la valeur de la fonction objectif. La valeur de ce paramètre sera définie dépendamment de la taille de l'instance. Ce paramètre permet d'intégrer deux voisinages (ou plus) de l'ensemble initial de clusters dans la même résolution. Dans certains cas, il est plus important de résoudre les sous-problèmes sur un voisinage plus large que de prendre le risque de les résoudre plusieurs fois.

— Utiliser des valeurs dynamiques pour les paramètres de la méthode d'exploration de voisinage qui s'adaptent à la phase de résolution et à la profondeur du branchement dans la phase de branchement.

— Utiliser des valeurs dynamiques pour des paramètres de dominances dépendamment de la valeur de la DMT. En effet, pour un voisinage plus petit, on applique des conditions de dominance plus précises, car le domaine est plus petit.

— Adapter des fonctionnalités de génération de colonnes à la MEV. Par exemple, choisir le réseau à résoudre. On choisit le plus ancien réseau résolu. L'objectif de cette adaptation est de résoudre le maximum de réseaux différents pour générer le plus grand nombre de chemins qui suivent des clusters différents.

4.3 Construction d'un ensemble initial de clusters

La MEV permet également de générer des horaires pour les membres de la catégorie FA, qui ressemblent aux horaires des chefs. Cela permettra à un membre de la catégorie FA d'avoir le minimum de chefs possible durant son horaire mensuel. La MEV s'applique en explorant progressivement le voisinage de l'ensemble initial de clusters.
On a vu que la résolution des catégories des membres d'équipages pour la compagnie étudiée se fait d'une manière séquentielle et séparée, en commençant par la résolution de la catégorie des chefs et en finissant par la catégorie des FA. On va alors utiliser l'information produite par la résolution des chefs afin de construire un ensemble initial de clusters pour les FA.
Pour les catégories des chefs, on peut trouver des rotations avec une demande supérieure à 1, c'est-à-dire, une rotation qui doit être couverte par plus d'un chef.
Deux méthodes ont été alors proposées pour construire l'ensemble initial de clusters. La première produit un ensemble non disjoint tandis que les deux autres produisent un ensemble initial de clusters disjoints.

4.3.1 Construction d'un ensemble initial de clusters non disjoints

Pour cette méthode, chaque ligne de la solution des chefs est à l'origine d'un ou de plusieurs clusters. La ligne peut être considérée comme étant un seul cluster si la durée entre chaque

deux rotations successives est inférieure à un nombre spécifique de journées (qu'on appelle OffDaysMax). Sinon, la ligne sera brisée à un ou plusieurs clusters afin d'obtenir des clusters (parties de la ligne) avec une durée entre les rotations successives, inférieure au paramètre OffDaysMax. En effet, les lignes complètes extraites des solutions des chefs ne pourraient pas être considérées comme des clusters initiaux, car d'autres tâches telles que les vacances les entraînements ... pourraient détériorer la qualité du cluster si la durée entre deux rotations successives est élevée.

La méthode de construction d'un ensemble non disjoint est inefficace, voire inutile pour la méthode d'exploration de voisinages. Le calcul de la distance d'un chemin par rapport à un ensemble initial de clusters n'est pas cohérent à cause des clusters qui chevauchent entre eux (rotations similaires appartenant à deux clusters différents). La qualité des clusters obtenus à partir de la solution des chefs est alors susceptible à se détériorer à cause des clusters non disjoints. En effet, on peut s'éloigner du voisinage cible sans pouvoir contrôler ce phénomène avec la méthode implémentée. Le problème peut être illustré avec l'exemple suivant : Soit : C_1 et C_2 faisant partie de l'ensemble initial de clusters non disjoints et qui sont définis comme suit : C_1 $\{T_1, T_2, T_3, T_4\}$, C_2 $\{T_5, T_6, T_3, T_7\}$, Si on veut générer exactement des lignes qui suivent ces clusters, on devrait obtenir seulement deux lignes possibles. Cependant, des successions comme $\{T_1, T_2, T_3, T_7\}$ ou $\{T_5, T_6, T_3, T_4\}$, pourraient se produire durant la prolongation sans qu'elles ne soient pénalisées avec la méthode implémentée. Ces successions pourraient alors nuire au comportement de la MEV. Pour remédier à cette situation, on propose alors des méthodes pour construire un ensemble de clusters disjoints.

4.3.2 Construction d'un ensemble de clusters disjoints en utilisant un algorithme glouton

L'objectif de cet algorithme est de construire, à partir des tâches consécutives de lignes extraites des solutions des chefs, des clusters disjoints qui couvrent la totalité des rotations. L'algorithme 1 décrit cet algorithme.

Dans la ligne 6 de l'algorithme 1, on dit qu'un cluster est admis s'il constitue une succession de rotations consécutives ou une suite de rotations séparées par une durée inférieure à $OffDaysMax * 24h$ ($OffDaysMax$ est un paramètre qui représente le nombre de jours qu'on permet entre deux rotations consécutives).

Dans la ligne 10 de l'algorithme 1, on appelle une succession de rotations légales de l, une succession de rotation qui peut définir un cluster admis.

La procédure **MAJ**(L, r) consiste à éliminer les lignes de L qui ont une (ou plusieurs) rotation(s) en commun(s) avec la ligne r choisie comme cluster. Le but de cette procédure est

Algorithme 1 | Algorithme glouton pour la construction d'un ensemble C de clusters disjoints couvrant toutes les rotations.

1: $C \leftarrow \emptyset$
2: Initialiser l'ensemble L avec les lignes initiales des employés-chefs.
3: **TANT QUE** $L \neq \emptyset$ **FAIRE**
4: Marquer la première ligne r de L
5: Supprimer la ligne r de L
6: **SI** r construit un cluster admis **ALORS**
7: $C \leftarrow C \cup \{r\}$
8: **MAJ**(L, r)
9: **SINON**
10: Extraire un ensemble L_r des successions de rotations légales à partir de la ligne r et les ajouter à la fin de l'ensemble L. (c.-à-d. $L \leftarrow L \cup L_r$)
11: **FIN SI**
12: **FIN TANT QUE**

Algorithme 2 | Procédure **MAJ**(L, r)

1: **POUR TOUT** p in r **FAIRE**
2: **POUR TOUT** l *in* L **FAIRE**
3: **SI** p *in* l **ALORS**
4: Supprimer l de L
5: Extraire l'ensemble L_l des partitions légales de $l - p$
6: $L \leftarrow L \cup L_l$
7: **FIN SI**
8: **FIN POUR**
9: **FIN POUR**

d'assurer que les lignes restantes dans l'ensemble L sont disjointes avec la ligne r (i.e aucune rotation en commun). Pour les lignes de L qui ont au moins une rotation t en commun avec r, on supprime ces lignes de L, et on construit des clusters admis à partir des rotations (sauf la rotation t) de chacune de ces lignes et on les ajoute à la fin de l'ensemble L.

4.3.3 Construction d'un ensemble de clusters disjoints en utilisant un algorithme de flot à coût minimum

Les clusters disjoints de l'EIC ont été déterminés par un algorithme glouton. Même si ce dernier donne le meilleur choix local pour une rotation donnée, toutefois, un tel choix peut avoir un impact négatif sur l'ensemble initial de clusters obtenu. On va alors introduire une méthode permettant un choix plus global pour construire les clusters. En effet, le problème de construction de clusters initiaux peut être modélisé par un problème de flot à coût minimum. Ce problème consiste à répartir l'ensemble des rotations sur un nombre minimum de clusters disjoints trouvés à partir de la solution de la catégorie des chefs.
Le problème peut être modélisé sur un graphe orienté $G(N, A)$ (Figure 4.4) où N est l'ensemble des nœuds et A est l'ensemble des arcs. Chaque rotation p est représentée par un arc délimité par deux nœuds $p_{début}$ qui représente le début de la rotation et p_{fin} qui représente la fin de la rotation p. L'ensemble des nœuds N est l'union des ensembles $N_{début}$, N_{fin} et $\{o, d\}$ avec o est le nœud source et d est le nœud destination. $N_{début}$ Et N_{fin} sont les ensembles des nœuds de départs et d'arrivées des rotations p respectivement. L'ensemble A des arcs du graphe G est l'union des sous-ensembles suivants :
$A_o^{début}$: L'ensemble des arcs qui lient l'origine avec les nœuds j, tel que $j \in N_{début}$. Chaque arc de cet ensemble à un coût initial c_0.
A_d^{fin} : L'ensemble des arcs qui lient les nœuds j tels que $j \in N_{fin}$ avec le nœud destination d.
A_P : L'ensemble des arcs définis entre les nœuds de $N_{début}$ et les nœuds de N_{fin}. Ces arcs représentent les rotations du problème.
A_I : L'ensemble des inter-arcs définis entre les nœuds de N_{fin} et les nœuds de $N_{début}$.
$\{(o, d)\}$: L'arc qui lie l'origine et la destination.
$A_d = A_d^{fin} \cup \{(o, d)\}$
Pour construire le graphe $G(N, A)$ (figure 4.4), on construit d'abord un graphe intermédiaire $G'(N', A')$ représenté par la figure 4.3. Ce dernier est obtenu à partir de la solution d'une des catégories des chefs (CM ou CS) ou leur union. Les nœuds de ce graphe représentent les rotations et les arcs représentent les successions admissibles obtenues à partir des lignes de la solution. On rappelle qu'une succession admissible entre T_i et T_j existe si T_j suit T_i dans au moins une des lignes de la solution initiale et que la durée entre T_i et T_j ne dépasse pas la

Tableau 4.1 Les caractéristiques de chaque type d'arc du réseau $G(N, A)$.

Type d'arcs	Borne inférieure	Capacité	coût
$a \in A_o^{début}$	0	1	$c_0 >> 0$
$a \in A_d^{fin}$	0	1	0
$a \in A_p$	1	1	0
$a \in A_I$	0	1	$-c_1$
(o, d)	0	$\|P\|$	0

valeur du paramètre *OffDaysMax*. Le graphe $G'(N', A')$ est par la suite converti en $G(N, A)$ sur lequel on applique l'algorithme de flot à coût minimum.

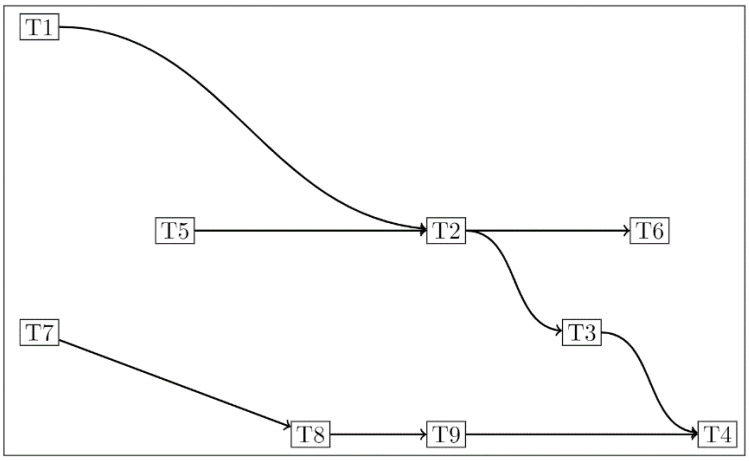

Figure 4.3 Graphe intermédiaire $G'(N', A')$

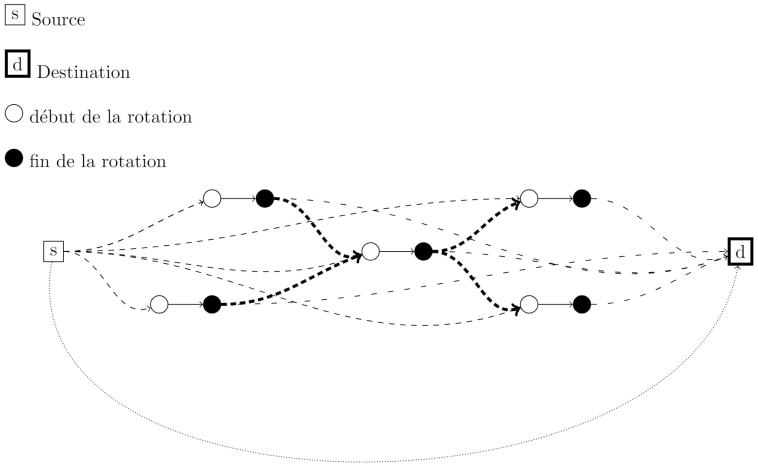

Figure 4.4 Graphe $G(N, A)$

En effet, la conversion se fait de la manière suivante : chaque nœud de graphe $G'(N', A')$ est dédoublé pour que la rotation soit modélisée avec deux noeuds, un noeud de départ et un

noeud d'arrivée. On ajoute également des arcs qui lient les nœuds de départ et les nœuds d'arrivée, un noeud source, un nœud destination, des arcs qui lient le noeud source avec les noeuds de départ de chaque rotation, des arcs qui lient les noeuds d'arrivée des rotations avec le noeud destination et un arc qui lie le noeud source et le noeud destination.

4.3.4 Comparaison des résultats de l'algorithme glouton et de l'algorithme de flot

On a utilisé le langage Python pour implémenter l'algorithme glouton proposé. En ce qui concerne l'algorithme de flot, on a utilisé un programme en Python pour définir le graphe $G'(N', A')$ et $G(N, A)$. Ce programme génère un code en langage C qui utilise la définition du réseau ainsi que la librairie du CPLEX pour l'optimisation du réseau (*CPLEX Network Optimizer*) afin de résoudre le problème de flot à coût minimum.
Afin de déterminer la méthode de construction de l'EIC disjoints à utiliser avec la MEV au niveau des résultats numériques, on effectue une comparaison entre les deux algorithmes présentés. Cela va simplifier la compréhension des résultats numériques présentés dans le chapitre suivant.

Tableau 4.2 Comparaison des ensembles issus des différentes instances et des différentes méthodes de construction

Instances	# de rotations/réseau	OffDaysMax	# de clusters		# moyen de rotations/cluster		# de clusters unitaires	
			flot	glouton	flot	glouton	flot	glouton
CM_O_3	1931	3	583	672	3,31	2,87	130	152
CM_O_4	1931	4	573	639	3,36	3,021	123	139
CS_O_3	3623	3	811	958	4,46	3,78	143	167
CS_O_4	3623	4	798	914	4,54	3,96	131	154
CM_CS_O_3	3624	3	738	901	4,9	4,022	111	126
CM_CS_O_4	3624	4	728	863	4,96	4,19	99	112
CS_M_4	3623	4	582	657	6,2	5,51	49	55
CS_M_3	3623	3	608	679	5,9	5,33	62	78

de rotation/réseau : nombre d'arcs de type rotation dans le réseau.
OffDaysMax : nombre de jours permis entre deux rotations consécutives.
de clusters : nombre de clusters.
flot : l'algorithme de flot.
glouton : l'algorithme glouton.
moyen de rotation/clusters : nombre moyen de rotations par clusters.
de clusters unitaires : nombre de clusters qui sont constitués d'une seule rotation.

Le tableau 4.2 représente une comparaison des ensembles initiaux de clusters obtenus par les deux algorithmes utilisés. On a testé les méthodes de construction de clusters sur les deux catégories des chefs (*CM, CS*), leur union ainsi que sur les données modifiées qu'on

obtient à partir du raffinement des données CS originales. On remarque que la qualité (en termes du nombre moyen de rotations par cluster, nombre de clusters et nombre de clusters unitaires) des clusters issus du problème de flot est meilleure que la qualité des clusters issus de l'algorithme glouton. On remarque également que la qualité des clusters issus des catégories CS et l'union des catégories CM et , est plus élevée que la qualité des clusters issus de la catégorie CM. En outre, les clusters issus des deux catégories CS et $CM \cup CS$ ont quasiment la même qualité. Donc, au niveau des exécutions dans la partie résultat, on optera pour les clusters issus de la catégorie CS, car les clusters issus proviennent en général de la même ligne (du même horaire) contrairement aux clusters issus de la catégorie CM union CS (un nombre important de tâches sont couvertes par des CM et des CS en même temps et donc on peut obtenir un cluster qui provient de plusieurs horaires initiaux).

CHAPITRE 5 RÉSULTATS NUMÉRIQUES

Dans ce chapitre, on commence par décrire les instances utilisées. Ensuite, on exécute ces dernières avec les différentes versions du logiciel. Un diagnostic des résultats trouvés est effectué. Dans ce diagnostic, on formule plusieurs hypothèses qu'on vérifie. À la base de ces hypothèses, une étude de la dominance sera effectuée. À la fin de ce chapitre, on présente une analyse de la version originale du logiciel.

5.1 Description des instances

On applique la méthode de construction de clusters sur la solution obtenue à la suite de la résolution des chefs de cabine afin d'obtenir l'ensemble initial de clusters. On utilise l'algorithme de flot à coût minimum pour construire l'ensemble initial de clusters afin d'obtenir un ensemble de clusters avec une cardinalité minimale. Par la suite, on résout l'instance des FA avec la nouvelle version du produit qui intègre la méthode d'exploration de voisinage expliquée plus haut. Pour l'exécution du code, on utilise des machines identiques avec 32 Go de mémoire et un processeur Intel Core i7-8700 CPU @ 3,20GHz×12.
Les noms des instances utilisées sont écrits en général sous la forme suivante : "NomDeLaCatégorie_x_a_y_b". Le terme "NomDeLaCatégorie" représente le nom de la catégorie des membres d'équipage (CS pour la catégorie des chefs et FA pour celle du reste des membres). La valeur 'x' prendra la lettre 'o' ou 'm'. Le 'o' pour indiquer que la solution utilisée pour construire l'ensemble initial des clusters provient des données originales (sans aucune modification) et le 'm' pour indiquer que les données ont été modifiées (appelées données raffinées et présentées plus bas). La valeur 'a' représente la distance maximale tolérée par rapport à l'ensemble initial des clusters (*NeighborhoodResUBMaxVal*, voir les paramètres de la section méthode d'exploration de voisinage pour plus de détails). La valeur 'y' représente la nature des données de l'instance à exécuter. Comme pour 'x', 'y' peut prendre la valeur 'o' si les données utilisées sont des données originales (sans aucune modification) ou la valeur 'm' si les données sont raffinées. La valeur 'b' reflète l'information de la valeur *NeighborhoodchangeCriterea* qui indique le critère de changement de voisinage de l'algorithme. Pour ne pas alourdir l'écriture, le terme *NeighborhoodchangeCriterea* sera remplacé par 'k' dans la suite du document.
Le tableau 5.1 met en évidence les valeurs de 'b' ainsi que les valeurs de 'k' correspondantes.

Une petite valeur de 'k' peut conduire à une amélioration faible de la valeur de la fonction objectif d'une itération à l'autre, ce qui causerait un nombre élevé d'itérations. On peut alors

Tableau 5.1 Description des valeurs de 'b' utilisées dans les noms des instances.

Valeur de b	Valeur du paramètre 'k' correspondante
'static'	Version statique du produit (valeur non existante).
'0'	Version statique du produit (valeur non existante).
'1'	0,001 (0,1%)
'2'	0,09 (9%)
'3'	0,25 (25%)

atteindre les critères d'arrêt de la version originale du solveur sans pour autant explorer tous les voisinages nécessaires. Une valeur très élevée de la valeur 'k' peut conduire à un élargissement de voisinage sans explorer suffisamment les voisinages les plus restreints. En général, le temps de résolution d'un PPCCCR dans un large voisinage est plus élevé qu'une résolution sur un voisinage plus restreint.

5.2 Exécution avec les données originales

On exécute quatre variantes de l'algorithme avec la version originale des données. Les résultats de ces dernières seront comparés à celui issu de l'exécution de la version originale du solveur (FA_original).
Le tableau 5.2 représente quelques paramètres de la méthode d'exploration de voisinage qui caractérisent l'exécution de chaque instance.

Tableau 5.2 Paramètres qui caractérisent la MEV pour chaque version.

Nom de l'instance	Nature de l'algorithme	Valeur de 'k'	Pas	DMT
FA_o_3_o_static	Statique	(Non existante)	(Non existant)	3
FA_o_3_o_1	Dynamique	0,001 (0,1%)	1	3
FA_o_3_o_2	Dynamique	0,09 (9%)	1	3
FA_o_3_o_3	Dynamique	0,25 (25%)	1	3

Dans le tableau 5.2, le paramètre "Pas" correspond à la valeur *NeighborhoodResUBStepVal*. On choisit une instance avec une petite valeur de 'k' (FA_o_3_o_1), une avec une valeur moyenne (FA_o_3_o_2), et une troisième avec une valeur de 'k' relativement grande (FA_o_3_o_3). En ce qui concerne les résultats trouvés, aucune des versions mentionnées plus haut n'a pu converger. Par conséquent, aucune solution entière réalisable ou fractionnaire n'a été trouvée. Le tableau 5.3 représente quelques caractéristiques des résultats obtenus après l'exécution du problème-cadre. Parmi ces caractéristiques, on trouve l'indice de non-faisabilité fractionnaire μ qui est défini comme la somme des variables d'écarts des contraintes

du problème maître (voir la section 3.1.1) :

$$\mu = \sum_{p \in P} y_p + \sum_{i \in I} (1 - \sum_{j \in \Omega_i} x_j)$$

En d'autres termes, cet indice représente la somme de la valeur fractionnaire nécessaire pour que chaque tâche soit couverte et pour que chaque employé ait un horaire. Avant d'entamer la résolution, la valeur initiale de l'indice est de 13 055. Ceci indique également le nombre de tâches et d'employés à couvrir dans le problème maître.

Tableau 5.3 Exécution du problème-cadre avec des données originales

Nom de l'instance	FA_original	FA_o_3_o_static	FA_o_3_o_2	FA_o_3_o_1	FA_o_3_o_3
μ	0	10,5	11	1639,87	7
Temps total (s)	16822,3	18655	94062,5	129315,6	24110,4
Temps SP (s)	11 822	15 008,8	78 761,3	110 398,3	20 306,8
Temps PM (s)	4661,5	3553,6	15 126	18 822,2	3722,7
# total d'iter.	33	168	172	1435	125
# SP résolus	23765	149832	142963	540815	96548

Temps total : représente le temps total de résolution.
Temps SP : représente le temps total consommé au niveau de la résolution des sous-problèmes.
Temps PM : représente le temps total consommé au niveau de la résolution du problème maître.
total diter : représente le nombre total d'itérations de la génération de colonnes.
SP résolus : nombre total de sous-problèmes résolus.

En analysant le résultat et en se basant sur l'indice de non-faisabilité, on constate que les trois instances FA_o_3_o_static, FA_o_3_o_2 et FA_o_3_o_3 sont proches d'une solution réalisable contrairement à l'instance FA_o_3_o_1. Cela est dû à la valeur de 'k' choisie pour chaque instance. En effet, pour l'instance FA_o_3_o_1, on a atteint les critères d'arrêt (de la version originale du logiciel) de l'algorithme de génération de colonnes avant d'explorer tous les voisinages de l'ensemble initial des clusters. Cependant, avec ces quatre instances, la MEV n'a pas pu générer des colonnes avec un coût réduit suffisamment négatif. Pour cette raison, on remarque dans le tableau 5.3 que le nombre d'itérations s'est élevé, de même pour celui du nombre de sous-problèmes résolus. On remarque également que le temps de résolution de ces instances est plus long par rapport aux données originales.
Les différentes exécutions ont été effectuées en générant des colonnes qui sont à une distance d'au plus de 3 unités par rapport à l'ensemble initial de clusters. La non-faisabilité peut alors être expliquée par le fait que l'ensemble initial de clusters construit à partir des chefs n'est pas une bonne cible pour la catégorie FA. En effet, en analysant davantage les données des chefs et des FA, on remarque des points d'incompatibilité qui pourraient causer ce phénomène.

Par exemple, dans la catégorie FA, la demande peut être différente d'une rotation à l'autre. Supposons que la rotation 1 nécessite 6 employés, on peut trouver une autre rotation qui nécessite seulement 3 employés et qui succède à la rotation 1 dans un cluster donné. On aura obligatoirement au moins 3 membres qui vont briser ce cluster, car la demande de la rotation 2 est de 3 pendant que la demande de la rotation 1 est de 6.
Plusieurs raisons peuvent causer une demande différente de personnel, en voici quelques-unes :

- Une première raison est que le type d'avion peut différer d'une rotation à une autre. On peut par exemple trouver une rotation avec des vols qui opèrent sur un *Boeing 737* et une autre rotation avec des vols qui opèrent sur un *Boeing 787*. Ces deux appareils n'ont ni la même taille ni le même nombre de sièges. Par conséquent le nombre de membres d'équipage dans chaque rotation sera potentiellement différent.
- Une deuxième raison est que la compagnie étudiée regroupe une partie des membres d'équipage dans d'autres catégories en plus de celles des chefs et des FA. Ainsi, une partie de la demande de quelques rotations sera satisfaite avant de résoudre la catégorie FA.
- On trouve également des rotations qui appartiennent aux chefs et qui n'appartiennent pas au FA et vice versa.
- Il faut noter que le planificateur peut aussi ajouter des rotations construites manuellement. Ces rotations sont généralement atypiques et durent entre 9 à 17 jours par mois.

Afin d'éviter ce problème, on va tester la méthode d'exploration de voisinage sur des données raffinées qui limitent les incompatibilités entre les données de la catégorie des chefs et celle de la catégorie FA.

5.3 Données raffinées

Les données de cette compagnie présentent des difficultés qui ne figurent pas dans les autres compagnies. Nous identifions ces difficultés et indiquons la méthodologie adoptée pour les éliminer de façon à travailler sur un problème représentatif de plusieurs compagnies.

5.3.1 Problèmes rencontrés

Afin d'obtenir une meilleure cible, on procède à un raffinement de données en éliminant les problèmes qui pourraient causer la non-faisabilité. En voici une liste de quelques problèmes rencontrés :

- Les rotations utilisées pour la catégorie des chefs ne sont pas identiques aux rotations utilisées au niveau de la catégorie FA.

— Le nombre de demandes α_p (voir la contrainte 3.2 du modèle) est différent pour chaque rotation dans la catégorie FA.
— Les règles imposées sur les activités de type réserve dans la catégorie des chefs diffèrent des règles imposées au niveau de la catégorie des FA.
— Les fenêtres de crédits (crédit minimum, crédit maximum) sont différentes d'une catégorie à une autre.
— Quelques employés ont un nombre élevé de tâches pré-assignées .
— L'aspect multi objectifs de la fonction objectif conduit à une perturbation du problème, car les exigences diffèrent d'une catégorie à une autre.

5.3.2 Solutions proposées

Ci-dessous, les améliorations proposées :

— On garde seulement les rotations en commun entre les chefs et les FA.
— On considère que la flotte est constituée d'appareils de même type ou de types homogènes (par exemple Boeing 737 et Airbus 320) et donc, a une demande égale de membres d'équipage pour chaque vol.
— On élimine les tâches de type réserve.
— Si un employé à un nombre élevé de tâches pré-assignée, on diminue ce nombre.
— On focalise seulement sur un seul objectif qui est la satisfaction des employés au lieu d'avoir un problème avec plusieurs objectifs (quelques exigences liées aux objectifs éliminés sont modélisées sous la forme de contraintes dures, par exemple la demande d'une langue spécifique pour une rotation donnée).

On développe alors un script *bash* qui fait appel à un ensemble de scripts Python pour ajuster les fichiers d'entrées du solveur (sol.in, act.in, staf.in ...) aux changements proposés.

5.3.3 Exécution avec les données raffinées

L'exécution avec les données originales n'a pas produit de solution réalisable fractionnaire ni de solution réalisable entière. À présent, on va exécuter avec les données raffinées, les mêmes variantes de l'algorithme utilisées avec les données originales. On exécute avec les 4 instances présentées dans le tableau 5.4.
La première instance sera traitée avec la version statique du produit pendant que les trois autres seront traitées avec la version dynamique en variant la valeur de 'k'. Ce traitement des instances produit les résultats mentionnés dans le tableau 5.5 :

On remarque qu'une solution entière a été obtenue pour 3 instances parmi 4. Cependant, pour l'instance FA_m_3_m_1, aucune solution réalisable entière n'a été générée. Ceci est

Tableau 5.4 Données raffinées à exécuter.

Nom de l'instance	Nature de l'algorithme	Valeur de 'k'	Pas	DMT
FA_m_3_m_static	Statique	(Non existante)	(Non existant)	3
FA_m_3_m_1	Dynamique	0,001 (0,1%)	1	3
FA_m_3_m_2	Dynamique	0,09 (9%)	1	3
FA_m_3_m_3	Dynamique	0,25 (25%)	1	3

Tableau 5.5 Résultat de l'exécution avec les données raffinées

Nom de l'instance	FA_m_initial	FA_m_3_m_static	FA_m_3_m_2	FA_m_3_m_1	FA_m_3_m_3
Indice de non réalisabilité	0	0	0	0	0
Solution BB	-4 006 282,5	-3 791 127,5	-3 187 065,5	-	-3 172 582,5
Satisfaction moy. (%)	79,33	72,97	59,65	-	59,41
Temps total (s)	3526,5	4115,02	11 219,69	-	4927,17
Temps SP (s)	1515,6	2220,7	6903,7	-	3154,8
Temps PM (s)	1840,1	1616,8	3933,3	-	1575,9
# d'itérations	1112	1053	1315	-	1196
# de SP résolus	21 387	61 170	100 840	-	62 378

Solution BB : représente la valeur de la fonction objectif associée à la solution entière.
Satisfaction moy. : représente la satisfaction moyenne des préférences des membres d'équipage.

dû à la valeur de 'k' choisie pour traiter cette instance. En effet, après chaque itération de génération de colonnes, la valeur de 'k' détermine si on doit explorer un voisinage plus large dans l'itération suivante ou garder le même voisinage. On évalue l'amélioration de la valeur de la fonction objectif dans les dernières m itérations et un test est effectué pour déterminer si on doit explorer un voisinage plus large ou non.

Le processus de résolution a pris alors beaucoup de temps durant l'exploration des voisinages les plus proches de l'ensemble initial des clusters. Le coût de la solution représenté par la valeur de la fonction objectif ne diminuait pas suffisamment et les colonnes générées ne sont pas de bonne qualité. Ainsi, le processus de résolution n'a pas abouti à une solution réalisable entière. Un problème identique se présente pour la catégorie FA_m_3_m_2. On n'explore pas suffisamment les voisinages plus larges, car on passe la grande partie du temps dans l'exploration des voisinages les plus proches et les critères d'arrêt du solveur se déclenchent avant l'exploration des voisinages les plus larges.

La catégorie FA_m_3_m_static produit la meilleure solution en termes de qualité avec une solution entière de -3 791 127 et une satisfaction moyenne de 72,9% par employé, et en termes de temps de résolution total avec 4115 secondes. Cependant, la qualité de cette solution reste assez faible par rapport à la solution obtenue avec la version originale de produit. En effet, on remarque que la qualité de la solution de l'instance FA_m_initial, obtenue avec l'exécution de la version originale du produit, est de -4 006 282,5 et une satisfaction moyenne de 79,33% par employé. On remarque également que cette résolution a été effectuée dans un temps plus petit

(3526 secondes) comparativement au temps d'exécution de la catégorie FA_m_3_m_static. La version statique du produit donne une meilleure solution par rapport aux autres versions, car à partir des premières itérations, on se permet de générer des chemins dans le plus large voisinage toléré (soit le voisinage de distance 3 par rapport à l'ensemble initial de clusters). Après avoir analysé les résultats des exécutions des données raffinées, on remarque que les voisinages les plus proches de l'ensemble initial de clusters ne génèrent pas assez de colonnes de bonne qualité pour permettre à la fonction objectif de converger. Pour cette raison, on propose l'hypothèse suivante, qu'on nomme hypothèse 1 : on suppose que l'ensemble initial de clusters construit à partir des chefs ne correspond pas à une bonne cible pour les FA et donc la génération de colonnes au voisinage de cet ensemble ne produit pas de solution réalisable. Par la suite, on vérifiera la validité de l'hypothèse 1. Dans le cas contraire, un diagnostic sera réalisé afin de déterminer d'autres raisons d'inefficacité de la méthode d'exploration de voisinage.

5.4 Diagnostique de l'inefficacité de la méthode d'exploration de voisinage

5.4.1 Vérification de l'hypothèse 1

L'hypothèse 1 consiste à supposer que l'ensemble initial de clusters construit à partir des chefs ne forme pas une bonne cible pour la catégorie FA. Pour vérifier cette hypothèse, on va construire un ensemble initial de clusters à partir de la solution des chefs pour résoudre la même catégorie des chefs en utilisant la version du produit avec la méthode d'exploration de voisinage. En d'autres termes, cela se résume par le processus suivant :

- Étape 1 : Résoudre la catégorie des chefs avec la version originale du produit.
- Étape 2 : Construire un ensemble initial de clusters à partir de la solution trouvée à l'étape 1.
- Étape 3 : Re-résoudre la catégorie des chefs avec la méthode d'exploration de voisinage en utilisant l'ensemble de clusters produit à l'étape 2.

On choisit cette approche particulière pour s'assurer que l'ensemble initial de clusters forme une bonne cible pour la catégorie à résoudre. Donc, on doit normalement pouvoir générer l'ensemble des colonnes qui font partie de la solution dans le voisinage le plus restreint de l'ensemble initial de clusters.

En ce qui concerne les tests, on va utiliser les mêmes variantes du produit que les tests précédents plus une version qu'on va appeler CS_m_0_m pour exprimer la résolution de l'instance à une distance de zéro de l'ensemble initial de clusters.

Le tableau 5.6 présente les résultats trouvés. Dans ce tableau, on ne va pas représenter les instances non réalisables :

En analysant le tableau 5.6 des résultats, on remarque qu'on obtient une solution entière

Tableau 5.6 Exécution avec des instances pour tester l'hypothèse 1.

Nom de l'instance	CS_m_initial	CS_m_1_m_static	CS_m_1_m_1
μ	0	0	0
Solution BB	-1 105 236,9	-661 675,9	-750 622,9

pour les instances exécutées avec la version statique et avec la version qui a la plus petite valeur de k. Toutefois, aucune solution n'a été retrouvée pour les autres versions.
L'analyse suivante donnera plus d'importance à l'étude de la non-faisabilité de l'instance CS_m_0_m. On n'arrive pas à trouver ni une solution réalisable entière ni une solution réalisable fractionnaire. En analysant davantage l'indice de non-faisabilité ainsi que les tâches non couvertes, on constate que 11 rotations et un employé sont non couverts. En comparant les rotations de l'employé dans la solution initiale (résolu avec la version originale du produit) avec les rotations non couvertes, on trouve que les rotations ne sont pas identiques, cela dit que le cluster (ou les clusters) qui devrait correspondre à l'employé en question a (ont) été affecté(s) à un autre employé.
La première vérification qu'on va effectuer pour s'assurer qu'il n'y a aucun problème avec l'intégration de la méthode d'exploration de voisinage dans le processus global de résolution est de construire une instance réduite en considérant seulement les rotations de l'employé non réalisable en question et en résolvant l'instance avec la méthode d'exploration de voisinage (l'instance réduite a été construite à partir de la solution de l'instance résolue avec la version originale du produit en considérant seulement la solution de l'employé en question). Le tableau 5.7 décrit l'instance réduite en question :

Tableau 5.7 Description de l'instance réduite.

Instance	Nombre de nœuds	Nombre d'arcs	Nombre d'arcs de type rotations
CS_red_prob_1	153	260	12

La résolution de cette instance réduite produit une solution et donc le chemin a été généré pour l'employé en question. À partir de ce résultat, on suppose que la raison pour laquelle l'instance CS_m_0_m ne produit pas de solution réalisable est causée par l'élimination de chemin par la dominance. Le réseau englobant toutes les tâches est un réseau de grande taille. Des millions d'étiquettes sont construites au cours de la résolution du PPCCCR et les règles de dominance utilisées sont des règles heuristiques. On nomme cette supposition l'hypothèse 2.

5.4.2 Vérification de l'hypothèse 2

Le moyen le plus efficace pour vérifier l'hypothèse 2 est d'analyser l'exécution de la solution obtenue avec la version originale du produit. Lors de cette résolution, on extrait tous les chemins qui ont été générés pour l'employé en question afin de déterminer à quel niveau le chemin qui fait partie de la solution finale a été généré.
On remarque que le chemin n'a pas été généré dans le problème-cadre (rappelons que le problème-cadre représente la première phase de résolution. Cette phase résout un problème linéaire pour s'assurer de la faisabilité du problème).
En effet, la première apparition du chemin en question était dans la phase satisfaction (phase de branchement) au niveau du nœud de branchement numéro 7.
Le chemin a été généré, car le réseau utilisé dans le nœud de branchement 7 contient moins de tâches (et donc moins de possibilités) par rapport au réseau initial. En effet, le solveur adopte deux méthodes de branchement, la méthode *Cfix* et la méthode *Tsplit*. La première méthode fixe l'horaire complet d'un employé (ou de plusieurs employés) si le score de l'horaire est suffisamment élevé, tandis que la deuxième fixe quelques tâches qui ont un score élevé, de plusieurs employés. (Le score est calculé à partir du flot par tâche.)
Les deux méthodes de branchement fixent des tâches et donc éliminent un nombre important de nœuds et d'arcs du réseau original ce qui conduit à un réseau relativement plus petit et donc un effet de dominance moins agressive. Par conséquent, le chemin en question se génère après plusieurs étapes de branchement.

5.5 Étude de la dominance

5.5.1 Effets indésirables de l'heuristique de dominance existante dans la version originale du produit

Pour l'instance CS_m_0_m exécutée avec la méthode d'exploration de voisinage, l'employé étudié n'a pas de solution dans le problème-cadre, car d'une part, le chemin cible est dominé et d'autre part, les autres chemins possibles pour cet employé (qui ont été générés lors de la résolution avec la version originale du produit) ont été éliminés avec la MEV, car ils ne sont pas à une distance zéro par rapport à l'ensemble initial de clusters.
L'objectif est de déterminer à quel niveau, le label correspondant au chemin cible a été dominé. Pour ce faire, il faut être capable de suivre l'ensemble des labels générés. Le réseau est de grande taille et des millions de labels pourraient être générés. Il est donc pratiquement impossible de suivre tous les labels générés. Pour remédier à ce problème, on va utiliser la méthode suivante pour suivre les labels intéressants à l'étude : on suppose que la succession

des rotations extraites du chemin cible donne le cluster suivant : $\{T_1, T_2, T_3, T_4, T_5\}$.
On affiche les labels qui couvrent au moins une tâche du cluster. Les groupes de labels trouvés sont :

- Groupe avec seulement une tâche du cluster : (T_1)
- Groupe avec deux tâches (T_1, T_2)
- Groupe avec deux tâches (T_1, T_3)
- Groupe avec trois tâches :(T_1, T_2, T_3)
- Groupe avec quatre tâches : (T_1, T_2, T_3, T_5)

On remarque qu'aucun label ne contient les deux rotations T_3 et T_4 et en particulier la rotation T_4. Cela veut dire qu'aucun label n'atteint cette rotation. La non-génération du chemin qui contient la succession $\{T_1, T_2, T_3, T_4, T_5\}$ est alors causée par l'aspect heuristique de la dominance.
Plusieurs explications sont possibles. On trouve entre autres :

- La dominance est faite sur un sous-ensemble de ressources à chaque nœud. En effet, il existe 33 ressources au total et on domine seulement sur un maximum de 7. Les ressources utilisées dans la dominance diffèrent d'un modèle de génération de colonnes à l'autre. La résolution utilise 4 modèles de génération de colonnes avec des paramètres affectant la dominance qui varient d'un modèle à l'autre.
- Si un ensemble de labels qui ont les mêmes valeurs du sous-ensemble de ressources sur lequel la dominance est effectuée, alors, un seul label est gardé d'une manière aléatoire et les autres labels sont éliminés.
- L'application des filtres : un filtre permet de garder un nombre limité de labels après chaque dominance. Un oracle est défini pour calculer pour chaque label, une valeur correspondante après chaque dominance. C'est une fonction qui fait une somme pondérée de la qualité des valeurs des ressources. Par exemple, pour la ressource comptant le nombre de crédits, la valeur est faible si le nombre de crédits est différent de ce que l'on attend pour la date du mois à laquelle le chemin représenté par le label est rendu. C'est un calcul heuristique basé sur l'expertise des analystes.

  Les labels qui ont les plus petites valeurs seront éliminés afin de garder un nombre de labels cohérent avec la taille de la mémoire allouée. Théoriquement, ce processus peut avoir un effet néfaste sur la qualité de la solution. Cependant, au niveau pratique, l'utilisation de ce processus est importante, voire primordiale, car le nombre de labels générés est très élevé, ce qui rend la résolution du problème impossible avec les machines utilisées et dans des temps pratiques.

La figure 5.1 illustre la raison principale qui accentue l'effet néfaste de l'application heuristique de la dominance. Par exemple, pour le nœud n de la figure 5.1, on remarque un

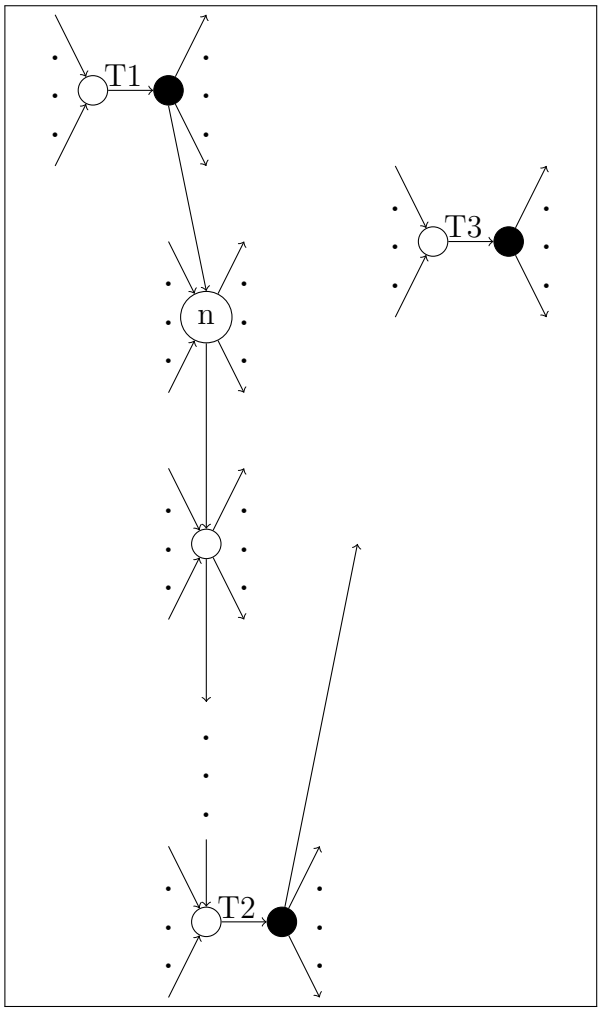

Figure 5.1 Illustration d'une partie du réseau

très grand nombre d'arcs incidents à n. Or, l'heuristique de la dominance est appliquée dans le nœud n après le processus de prolongation des labels sur les arcs incidents à ce nœud. Par conséquent, un grand nombre de labels qui pourraient être utiles pour la solution finale sera éliminé sans l'existence d'une raison valide. Voici quelques exemples des situations rencontrées :

— Il est difficile de générer un chemin qui contient trois rotations successives $(T1, T2, T3)$ tel que la première est située en haut du réseau (c.à.d. s'il n'y a pas ou il y a peu d'arcs de suivi descendants entre les arcs qui représentent les jours non travaillés et la rotation), la deuxième se situe en bas du réseau (c.à.d. s'il y a un nombre important d'arcs de suivi descendants entre les arcs qui représentent les jours non travaillés et la rotation) et la troisième se situe en haut du réseau (voir Figure 5.1), car pour qu'un label traverse verticalement le réseau, il doit passer par un grand nombre d'arcs et donc, il va passer par plusieurs nœuds. Or, on sait que l'heuristique de la dominance

est appliquée à chaque nœud. Donc, l'aspect heuristique de la dominance est plus agressif dans ce type de situation, car le chemin peut s'éliminer à n'importe quel nœud (après application de la dominance) même si ce dernier est capable de produire une meilleure solution s' il n'est pas éliminé .

— À cause de la structure actuelle du réseau, le nombre de chemins possibles entre deux arcs de type rotation est très élevé. En particulier, les arcs qui représentent les jours de congés et les arcs qui représentent les jours libres (voir réseau figure 3.1), alors on ne peut en aucun cas garder tous les labels possibles qui sont pareto-optimaux, car ce nombre est combinatoire et donc très élevé. Or, la mémoire actuelle de la machine ne peut pas supporter tous ces labels. Donc, on procède à une élimination heuristique afin de garder un nombre de labels cohérent.

— Après l'application de la dominance, on évalue le nombre de labels gardés. Si on constate qu'il n'y a pas assez de mémoire pour prolonger la totalité de labels, on élimine aléatoirement un nombre important pour ne pas dépasser la taille de la mémoire accordée.

5.5.2 Solutions proposées

Plusieurs solutions ont été proposées pour remédier aux problèmes causés par la dominance. Ces solutions ont deux objectifs : conserver un nombre plus élevé de labels et ajouter des critères pour garder les labels les plus intéressants.

On a ajouté plusieurs ressources afin de définir une mesure qui peut différencier d'une manière plus précise entre les labels. L'ajout des ressources dans la dominance a produit un temps plus élevé, car le temps du processus de dominance croit rapidement avec le nombre de ressources utilisées. En outre, les ressources qu'on a utilisées n'ont pas pu améliorer l'aspect aléatoire de la dominance. Un exemple des ressources utilisées est la ressource *MEVRESDOM1* qui a pour but de ne pas dominer les labels qui proviennent des différents clusters entre eux (on permet seulement à la dominance d'être appliquée entre les labels du même cluster). L'objectif de cette ressource est de garder le maximum de labels diversifiés qui proviennent de clusters différents.

En plus de l'ajout des ressources, on a modifié les paramètres pour pouvoir garder un nombre plus élevé de labels ou de dominer sur plus de 7 ressources (qui est le maximum actuel). Cependant la solution obtenue avec ces modifications était toujours non réalisable ou sa génération prend un temps très élevé.

5.5.3 Évaluation de la version originale du produit

L'objectif de cette section est de comprendre si c'est la MEV qui n'est pas compatible avec la version originale du produit ou si les problèmes rencontrés avec la MEV pourraient apparaitre également dans la version originale.
Pour ce faire, on utilise des tests sur la version originale du produit en changeant légèrement quelques paramètres ou en effectuant une petite perturbation sur les données.
Voici une partie des tests effectués :

- Le problème étudié est caractérisé par un intervalle de crédit. Cet intervalle représente les valeurs de crédit minimum et maximum qu'un membre d'équipage peut effectuer. Pour l'instance traitée, l'intervalle de crédit est entre 127 et 135. On va modifier cet intervalle afin de prendre un intervalle plus large qui englobe l'intervalle de départ. L'intervalle choisi est entre 110 et 140. L'instance qui correspond à ce changement est nommée FA_cred_eval.
- La deuxième modification effectuée est la relaxation de la contrainte qui spécifie un nombre maximum de jours de repos. L'instance qui correspond à ce test est nommée FA_doff_eval.
- La troisième modification correspond à la réduction du problème. En effet, on va éliminer 5 membres d'équipage ainsi que les taches qui ont obtenu lors de l'exécution avec les données originales. Cette instance est nommée FA_dataEdit_eval.

Tableau 5.8 Résultats suite à la perturbation de la version originale

Nom de l'instance	FA_original	FA_cred_eval	FA_doff_eval	FA_dataEdit_eval
μ	0	21.84	0	0
Temps total (s)	16822.3	-	13532.14	-
Solution BB (s)	-3962758.41	-	-3411394.82	-
Satisfaction (%)	76.43	-	69,27	-

Même si on a relaxé le problème original en relaxant quelques contraintes, ou en réduisant la taille du problème en fixant les horaires de quelques employés, on remarque que la résolution se détériore.
On obtient une solution non réalisable avec FA_cred_eval, une solution réalisable réelle, mais avec le problème qui ne converge pas vers une solution entière pour FA_dataEdit_eval. En outre, on obtient une solution qui converge dans un temps plus petit, mais avec une détérioration de la qualité de la solution avec FA_doff_eval.
Les mauvais résultats sont causés par l'algorithme de la dominance existant. Dans le premier cas, en élargissant la fenêtre du crédit, l'ensemble de labels qui peuvent être générés et plus

grand et donc une dominance plus drastique et mois précise. La même situation se présente pour le deuxième résultat.
Pour le troisième cas, en résolvant un problème réduit, les tâches qu'on a enlevées du problème sont alors enlevées du sous-problème de chaque employé. Donc, la résolution du PPCCCR est perturbée et la dominance change de comportement.
En partageant les résultats obtenus avec les analystes d'AD OPT, on m'a informé que la version du produit dédiée à cette compagnie est toujours en amélioration et ils doivent effectuer du *tuning* des paramètres quasiment sur chaque jeu de données traité afin de trouver une solution réalisable entière pour le mois en question.

CHAPITRE 6 CONCLUSION ET RECOMMANDATIONS

Le travail présenté dans ce document portait sur l'analyse et l'amélioration d'un produit dédié à la résolution du PCHMME d'une compagnie donnée en exploitant le processus séquentiel de résolution des catégories des membres de cabines, les similarités entre la catégorie des chefs et la catégorie FA, ainsi que le temps de résolution relativement court de la catégorie des chefs par rapport à la catégorie FA.
Dans ce chapitre, on présente une synthèse des travaux réalisés afin de réduire le temps de résolution élevé de la catégorie FA. Ensuite, on discutera les limitations de la solution proposée. Finalement on proposera des améliorations qui pourraient améliorer davantage la solution proposée.

6.1 Synthèse des travaux

La toute première étape qu'on a effectuée est l'analyse du problème et des données d'entrée. On a effectué une étude statistique qui compare les caractéristiques de la catégorie des chefs avec les caractéristiques de la catégorie des FA. On a également étudié les règles imposées sur chaque catégorie. Cette étude était basée principalement sur l'analyse des fichiers d'entrées et de sorties de chaque catégorie. Des outils ont été développés en Python pour extraire les statistiques nécessaires.
La deuxième étape était la conception et l'implémentation des outils de construction d'un ensemble initial de clusters. On a implémenté plusieurs programmes informatiques en Python et en langage C qui font appel aux librairies de CPLEX pour tester les différentes méthodes de constructions proposées.
La troisième étape concernait la conception de l'algorithme de la méthode d'exploration de voisinage et la proposition d'une approche pour intégrer la MEV dans le processus global déjà existant.
La quatrième étape du projet était l'implémentation de la MEV dans le code du produit en utilisant le même langage utilisé (la compagnie utilise le langage C au niveau de l'optimiseur) et en respectant l'architecture adoptée. On a également implémenté les différentes variantes de la MEV qu'on a proposée.
Au niveau des tests, on a d'abord utilisé les données originales du produit, par la suite on a utilisé des données modifiées. Ces données ont été produites suite à un raffinement de données qu'on a effectué.
Les tests effectués n'ont pas reflété les résultats attendus. Alors, on a effectué une étude

diagnostique pour comprendre les raisons qui ont causé ce problème. On a posé des hypothèses pour expliquer les raisons pour lesquelles on n'obtient pas de bons résultats. Par la suite on a vérifié la validité de chacune de ces hypothèses. Après plusieurs vérifications, on a constaté que la raison principale qui nuit à la MEV apparait dans l'aspect heuristique de la dominance. La majorité des chemins (colonnes) ciblés par la MEV sont éliminés d'une manière quasi aléatoire à cause de l'heuristique de la dominance existante dans la version originale du produit.
Dans ces conditions, il n'est pas possible de tester à sa juste valeur la méthode MEV. La version originale du produit devra être stabilisée pour produire de bonnes solutions de façon régulière. Par la suite, il pourra être possible de l'accélérer avec la méthode MEV. L'évaluation dans de bonnes conditions reste à faire.

6.2 Limitations de la solution proposée

La limitation principale de la MEV est la dominance heuristique adoptée au niveau du produit original de la compagnie. En effet, l'intégration de la MEV avec la version existante du produit ne peut pas donner de bons résultats.
La deuxième limitation est liée à la méthode d'exploration de voisinage. En effet, cette dernière peut rejeter des chemins qui pourraient contribuer à la solution globale, car ils ne font pas partie du voisinage défini. Cependant, ces chemins ont plus de chance d'être générés dans des itérations futures. L'amélioration qu'on peut obtenir au niveau du temps de résolution dépend alors de la qualité de l'ensemble initial de clusters. En effet, si les colonnes de bonne qualité ne figurent pas dans des voisinages proches de l'ensemble initial de clusters, on sera donc obligé d'effectuer plus d'itérations et donc plus de temps de résolution.

6.3 Travaux futurs

Les premiers travaux devraient porter sur la version originale du produit. Une première idée pour réduire l'élimination aléatoire de labels serait de modifier le modèle pour réduire le nombre de labels générés pour chaque suite de rotations. Le modèle actuel a une ressource qui compte les jours de repos et une qui compte les jours libres. La ressource sur les jours de repos doit satisfaire des contraintes de minimum et de maximum. La présence simultanée de ces deux contraintes empêche l'élimination par dominance et demande de garder des labels pour tous les nombres de jours de repos. De plus, la présence de la ressource qui compte les jours libres demande de conserver des labels pour les couples (nombre de jours de repos, nombre de jours libres). Comme le nombre de labels devient trop grand le logiciel fait des choix aléatoires pour réduire ce nombre.

Un meilleur modèle serait de supprimer la ressource sur les jours libres et la contrainte de borne supérieure sur les jours de repos. La contrainte sur le nombre minimum de jours de repos est suffisante pour obtenir une bonne solution. Il suffira, après la résolution, de transformer des jours de repos en jours libres pour satisfaire la contrainte de maximum. De plus, cette transformation pourra être faite en donnant de bonnes propriétés à la solution. Par exemple, grouper les repos en blocs de 2-3 jours et bien répartir ces blocs durant le mois. Ce modèle serait plus facile à résoudre car un seul label serait conservé pour chaque suite de rotations. Avec seulement la contrainte de minimum sur les jours de repos, la seule solution pareto-optimale est celle qui a le plus de jours de repos. Il faudrait aussi revoir les autres éléments de la modélisation pour voir s'il n'y a pas d'autres améliorations possibles.

RÉFÉRENCES

[1] "Revenue of commercial airlines worldwide," Statistica, 2019.

[2] "Airline industry passenger traffic globally," Statistica, 2019.

[3] G. Desaulniers, J. Desrosiers et M. M. Solomon, *Column generation*. Springer Science & Business Media, 2006, vol. 5.

[4] C. Barnhart, P. Belobaba et A. Odoni, "Applications of operations research in the air transport industry," *Transportation Science*, vol. 37, p. 368–391, 11 2003.

[5] F. Marchettini, "Automatic monthly cabin crew rostering procedure," 1980.

[6] W. Glanert, "A "timetable" approach to the assignment of pilots to rotations," 1984.

[7] B. Nicoletti, "Automatic crew rostering," *Transportation Science*, vol. 9, n°. 1, p. 33–42, 1975. [En ligne]. Disponible : https://doi.org/10.1287/trsc.9.1.33

[8] J. Buhr, "Four methods for monthly crew assignment - a comparison of efficiency," *AGIFORS Symp. Proc*, vol. 18, p. 403–430, 01 1978.

[9] G. Tingley, "Still another solution method for the monthly aircrew assignment problem," *AGIFORS Symp. Proc*, vol. 19, p. 143–203, 1979.

[10] D. Sarra, "The automatic assignment model," *AGIFORS Symp. Proc*, vol. 28, p. 23–37, 1988.

[11] H. J. e. L. J. Giaferri, C., "Automatic monthly assignment of medium-haul cabin crew," *AGIFORS Symp. Proc*, vol. 22, p. 69–95, 1982.

[12] P. R. Day et D. M. Ryan, "Flight attendant rostering for short-haul airline operations," *Operations Research*, vol. 45, n°. 5, p. 649–661, 1997. [En ligne]. Disponible : https://doi.org/10.1287/opre.45.5.649

[13] M. Gamache et F. Soumis, "A method for optimally solving the rostering problem," dans *Operations Research in the Airline Industry*. Springer, 1998, p. 124–157.

[14] M. Gamache *et al.*, "A column generation approach for large scale aircrew rostering problems," *Operations Research*, vol. 47, p. 247–262, 04 1999.

[15] W. Moudani, C. Cosenza et M. Coligny, "A bi-criterion approach for the airlines crew rostering problem," vol. 1993, 03 2001, p. 486–500.

[16] T. Fahle *et al.*, "Constraint programming based column generation for crew assignment," *Journal of Heuristics*, vol. 8, 01 2002.

[17] M. Sellmann *et al.*, "Crew assignment via constraint programming : Integrating column generation and heuristic tree search," *Annals OR*, vol. 115, p. 207–225, 09 2002.

[18] B. Maenhout et M. Vanhoucke, "A hybrid scatter search heuristic for personalized crew rostering in the airline industry," *European Journal of Operational Research*, vol. 206, nº. 1, p. 155–167, 2010.

[19] G. B. Dantzig et P. Wolfe, "Decomposition principle for linear programs," *Operations research*, vol. 8, nº. 1, p. 101–111, 1960.

[20] F. Vanderbeck, *Implementing Mixed Integer Column Generation*, 03 2006, p. 331–358.

[21] M. Savelsbergh, "A branch-and-price algorithm for the generalized assignment problem," *Operations Research*, vol. 45, p. 831–841, 12 1997.

[22] I. Elhallaoui *et al.*, "Dynamic aggregation of set-partitioning constraints in column generation," *Operations Research*, vol. 53, nº. 4, p. 632–645, 2005.

[23] ——, "Multi-phase dynamic constraint aggregation for set partitioning type problems," *Mathematical Programming*, vol. 123, p. 345–370, 06 2010.

[24] ——, "Bi-dynamic constraint aggregation and subproblem reduction," *Computers and OR*, vol. 35, p. 1713–1724, 05 2008.