

Modélisation et conception d'algorithmes pour la planification automatique du personnel de compagnies aériennes

Carmen Draghici

▶ **To cite this version:**

Carmen Draghici. Modélisation et conception d'algorithmes pour la planification automatique du personnel de compagnies aériennes. Automatique / Robotique. INSA de Toulouse, 2005. Français. NNT : . tel-00011399

HAL Id: tel-00011399
https://theses.hal.science/tel-00011399

Submitted on 17 Jan 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Thèse

Préparée au
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

En vu d'obtention du grade de
Docteur de l'Institut National de Sciences Appliquées de Toulouse

Spécialité : **Systèmes Industriels**

par

Carmen DRAGHICI

MODELISATION ET CONCEPTION D'ALGORITHMES POUR LA PLANIFICATION AUTOMATIQUE DU PERSONNEL DE COMPAGNIES AERIENNES

Soutenue le 29 septembre 2005 devant le jury :

Rapporteurs :	**Jean-Charles BILLAUT** **Christian PRINS**
Examinateurs :	**Simona Iuliana CARAMIHAI** **Jacques ERSCHLER**
Invité :	**Hervé LE ROY**
Directeurs de thèse :	**Jean-Claude HENNET** **Olivier PAILLOT**

A ma grand-mère

T’as vu ? Je suis docteur
maintenant. Pas un qui guérit les
malades, mais docteur quand
même…

Résumé

La planification et la gestion optimale des ressources humaines jouent un rôle important dans la productivité et la compétitivité des entreprises. Dans cette thèse nous nous intéressons à la modélisation et à la résolution de différents problèmes d'optimisation soulevés par la construction de plannings pour les agents qui travaillent dans un contexte aéronautique : la création de vacations, la création de rotation, l'affectation de vacations et de rotations. Pour le problème de construction de vacations, nous proposons une approche de modélisation basée sur le concept de plage horaire et ensuite une méthode heuristique de résolution basée sur l'algorithme FFD (First Fit Decreasing) et sur la génération de colonnes. Le problème de création de rotations est résolu par une méthode de programmation linéaire en variables mixtes. Les problèmes d'affectation de vacations et de rotations sont modélisés comme des problèmes de multi-affectation généralisé. Nous proposons une décomposition temporelle et par qualification et ensuite une transformation du problème d'affectation généralisé en un problème d'affectation simple par relaxation Lagrangienne. Un algorithme ad-hoc est utilisé pour la résolution de chaque problème de base. La plupart des algorithmes élaborés ont été couplés à des bases de données réelles et commercialisés par la société IFR-France.

Mots clés: optimisation de plannings, problèmes, First Fit Decreasing, génération de colonnes, programmation linéaire en variables mixtes, problème de multi-affectation généralisé, relaxation Lagrangienne.

Abstract

Optimal scheduling and management of human resources have an important role in the productivity and competitiveness of an enterprise. In this thesis we are interested in modeling and solving different optimisation problems which rise during the construction of schedules for agents working in an airline context: the shift generation problem, the crew pairing problem, the rostering problem. For the shift generation problem, we propose a modeling approach based on the concept of time slot and then a resolution method based on the First Fit Decreasing algorithm and the column generation technique. The crew pairing problem is solved using a mixed linear programming method. The rostering problem is modeled as a generalized multi-assignment problem. We propose a decomposition in time and by qualification and then a transformation of the generalized assignment problem into a simple assignment problem by Lagrangian relaxation. An ad hoc algorithm is used for the resolution of each basic problem. Most of the proposed algorithms were connected to real databases and commercialised by IFR France.

Keywords: schedules optimisation, First Fit Decreasing, column generation, mixed linear programming, generalized multi-assignment problem, Lagrangian relaxation.

Remerciements

Ce travail de thèse n'aurait pu être réalisés sans mes deux directeurs de thèse, Monsieur Jean-Claude Hennet, directeur de recherche au LAAS-CNRS, et Monsieur Olivier Paillot, ingénieur à IFR France, que je tiens très vivement à remercier. Je suis très reconnaissante pour leur confiance et pour l'autonomie qu'ils m'ont accordée.

Je remercie aussi Monsieur Christian Prince, professeur à L'Université de Technologie de Troyes et Monsieur Jean-Charles Billaut, professeur à L'Université « François Rabelais » de Tours pour l'honneur qu'ils m'ont fait d'accepter de rapporter ce travail et l'intérêt qu'ils y ont porté.

Je tiens également à remercier Monsieur Jacques Erschler, professeur à l'Institut National des Sciences Appliquées de Toulouse et Madame Simona Caramihai, professeur à l'Université « Politehnica » de Bucarest pour le crédit qu'ils ont accordé à mon travail en participant au jury de cette thèse. Un mot particulier pour Madame Caramihai qui a guidé mes premiers pas dans le monde de la recherche à l'Université « Politehnica » de Bucarest et qui m'a donné ensuite l'envie et la motivation d'effectuer cette thèse.

Mes remerciements vont aussi vers Marcel Mongeau, maître de conférences à l'Université « Paul Sabatier » de Toulouse. Notre collaboration sur l'un des chapitres de ce document et aussi ses nombreux conseils m'ont été précieux. Je lui remercie aussi pour son amitié et pour le courage d'avoir partagé avec moi le bureau pendant ma dernière année de thèse.

La motivation et le plaisir quotidien d'effectuer cette thèse n'auraient pu être les mêmes sans mes collègues du LAAS, doctorants ou stagiaires : Yasemin, Manue, Catherine, Steff, Malika, Codruta, Marcos, Sergio, Carmen, Andreea, Tatiana, François, Julien… Merci pour votre aide et votre soutien, mais aussi pour les agréables moments de détente passés ensemble.

Je suis reconnaissante à Hervé Le Roy et à Joan Vich qui ont accepté le défit d'une thèse CIFRE et qui m'ont chaleureusement accueilli au sein d'IFR France et de l'équipe KEOPS.

J'adresse toute ma sympathie à ceux qui chaque matin ont droit à un « Bonjour, KEOPS ! » : Nathalie, Gégé, Sissi, Abdel, Seb, Nino, Pete, Yannick. Ils m'ont conseillé et aidé tout au long de cette thèse.

Je ne pourrais pas passer sous silence Flo, Sandrine et Jean-Marc que j'ai bien harcelé avec mes problèmes administratives où logistiques. Merci pour votre disponibilité et efficacité.

Je remercie de tout cœur à ma famille et à mes amis de Roumanie qui, malgré la distance qui nous sépare, m'ont soutenu et supporté durant ces trois ans.

Enfin, je ne pourrais terminer sans citer celui qui est pour moi le "pisoi cel mai dragut foc".

Table de matières

Introduction……..……5

1. Planification des ressources humaines…….……..…9

1.1 INTRODUCTION 9
1.2 QUELQUES NOTIONS SUR LA PLANIFICATION 9
1.3 LA PLANIFICATION DANS UN CONTEXTE AERONAUTIQUE 11
1.3.1 Personnel au sol 11
1.3.2 Personnel navigant 12
1.4 CONCLUSIONS 14

2. Modèles et techniques de résolution pour les problèmes de planification d'horaires..15

2.1 INTRODUCTION 15
2.2 PROBLEMES CLASSIQUES D'OPTIMISATION COMBINATOIRE 15
2.2.1 Problème d'affectation 15
2.2.2 Problème de bin-packing 16
2.2.3 Problème de recouvrement d'ensemble 17
2.3 TECHNIQUES DE RESOLUTION RETENUES 18
2.3.1 Méthodes de résolution de problèmes de couplage et d'affectation 18
2.3.1.1 Bip Match 18
2.3.1.2 L'algorithme de Busacker et Gowen 20
2.3.1.3 L'algorithme hongrois 20
2.3.1.4 Méthode de relaxation Lagrangienne pour le problème d'affectation généralisé 21
2.3.2 Algorithmes de résolution pour le problème de bin-packing 22
2.3.2.1 Algorithmes on-line 22
2.3.2.2 Algorithmes off-line 23
2.3.3 Génération de colonnes pour la résolution du problème de recouvrement d'ensemble 23
2.3.3.1 Convergence de la méthode 25
2.3.3.2 L'algorithme générateur 26
2.3.3.3 Recherche des solutions entières 26
2.4 CONCLUSION 27

3. Création de vacations pour le personnel au sol……..……29

3.1 INTRODUCTION 29
3.2 CONCEPTS DE BASE : TACHES ET VACATIONS 29
3.2.1 Tâches 30
3.2.2 Vacations 30
3.2.3 Personnel 31
3.3 PRESENTATION DU PROBLEME 31
3.3.1 Contraintes réglementaires 32
3.3.2 Méthodes de résolution dans la littérature 33
3.3.2.1 Modélisation par recouvrement d'ensemble 33
3.3.2.2 Méthodes itératives 35
3.4 MODELISATION PROPOSEE 35
3.4.1 La représentation du temps 35
3.4.2 Plages horaires 36
3.4.3 Tâches, vacations, contraintes 37
3.4.4 Modélisation du problème 41
3.5 METHODE DE RESOLUTION PROPOSEE 42
3.5.1 Génération de vacations 42
3.5.1.1 Construction des patrons de vacations 42
3.5.1.2 Traitement des tâches fixes 42
3.5.1.3 Traitement des tâches mobiles 44
3.5.2 Remplissage de vacations 45

3.6 RESULTATS OBTENUS 45
3.7 CONCLUSIONS 46

4. Création de rotations pour le personnel navigant……..……49
4.1 INTRODUCTION 49
4.2 CONCEPTS DE BASE 49
4.2.1 Rotations *50*
4.2.2 Activités hors-vol *52*
4.2.3 Personnel *52*
4.3 MODELISATION ET RESOLUTION DU PROBLEME 52
4.3.1 Contraintes réglementaires *53*
4.3.2 Méthodes de résolution dans la littérature *53*
4.3.3 Modélisation proposée *56*
4.3.3.1 Construction des PdS 56
4.3.3.2 Construction de rotations à partir des PdS 62
4.3.4 Résultats *64*
4.3.4.1 Construction des PdS 64
4.3.4.2 Construction de rotations à partir des PdS 65
4.3.5 Post traitement des rotations *65*
4.4 PERSPECTIVES : CONSTRUCTION D'ALGORITHMES APPROCHES 68
4.4.1 Construction de PdS *68*
4.4.2 Constructions de rotations *68*
4.5 CONCLUSIONS 69

5. Affectation de vacations/ rotations……..……71
5.1 INTRODUCTION 71
5.2 PRESENTATION DU PROBLEME 71
5.2.1 Contraintes réglementaires *71*
5.3 METHODES DE RESOLUTION DANS LA LITTERATURE 72
5.3.1 Affectation de vacations *72*
5.3.2 Affectation de rotations *75*
5.4 MODELISATION DU PROBLEME 78
5.5 METHODE DE RESOLUTION PROPOSEE 79
5.5.1 Construction de la matrice de coût *80*
5.5.2 Calcul des coûts *81*
5.5.3 L'affectation de réserves pour le personnel navigant *84*
5.5.4 Affectation de séquences *86*
5.6 IMPLEMENTATION ET RESULTATS 86
5.6.1 Validation de la méthode *86*
5.6.2 Implémentation *87*
5.7 CONCLUSION 91

Conclusion……..……93

Bibliographie……..…..……97

Introduction

Le transport aérien connaît une croissance vertigineuse depuis les années 60 car, si en 1945, le trafic aérien comptait 9 millions de passagers par an, il est passé aujourd'hui à 1,8 milliard. Les compagnies aériennes sont nombreuses et ont tendance à se grouper car la concurrence est importante. Rester rentable dans un environnement complexe et concurrentiel est d'autant plus difficile qu'une compagnie est obligée de faire face à d'énormes dépenses pour payer le carburant, le personnel, les taxes d'utilisation des aéroports, l'entretien, la location ou l'achat des aéronefs. Les augmentations de recettes étant très difficiles sur ce marché très concurrentiel, une éventuelle réduction des coûts est essentielle pour assurer la rentabilité d'une compagnie.

La deuxième source de dépense d'une compagnie aérienne après le carburant est constituée par le personnel ; elle représente 15-20% du budget opérationnel. Une gestion efficace des plannings du personnel est donc d'une importance majeure, car une baisse de seulement quelques pourcents de ces coûts entraîne des économies annuelles qui peuvent s'élever à des dizaines de millions de dollars pour des grandes compagnies.

Générer des plannings est un processus difficile, soumis à des règles complexes et parfois contradictoires. De plus, aux contraintes réglementaires se rajoutent des règles sociales, de sécurité, de préférences personnelles...

Dans cette thèse nous nous intéressons à la modélisation et à la résolution de différents problèmes d'optimisation soulevés par la construction de plannings pour les agents qui travaillent dans un contexte aéronautique : le personnel au sol d'un aéroport et le personnel navigant d'une compagnie aérienne. Ce sont des problèmes qui ressemblent aux problèmes d'optimisation combinatoire classiquement étudiés en recherche opérationnelle mais, issus d'applications réelles, ils sont le plus souvent complexes et de grande taille.

Les travaux présentés dans ce manuscrit ont été effectués dans le cadre d'une Convention Industrielle de Formation par la Recherche entre la société IFR France et le groupe *Modélisation, Organisation et Gestion Intégrée des Systèmes d'Activités* du *Laboratoire d'Analyse et d'Architecture des Systèmes* du *Centre National de Recherche Scientifique*.

IFR France conçoit, réalise et commercialise des produits logiciels permettant de maîtriser les différents aspects de la gestion d'une compagnie aérienne, un grand intérêt étant porté à la gestion de personnel. L'objectif principal de ma thèse a été d'étudier, de concevoir et d'implémenter des algorithmes d'aide à l'élaboration des plannings de personnel, en combinant la recherche académique et l'expertise du domaine d'application spécifique.

Le manuscrit est divisé en 5 chapitres. Le premier chapitre présente la problématique de la planification des emplois du temps dans un cadre général. Le problème de création de plannings dans un contexte aéronautique est décrit par la suite. Dans un tel contexte, la charge de travail est constituée des tâches à réaliser en vol et au sol. Selon le type d'activité effectuée, le personnel se divise en personnel au sol et en personnel navigant : le personnel au sol effectue tout le travail au niveau de l'aéroport (l'enregistrement des passagers, l'accueil à l'arrivée et au départ, le service de correspondance et de transit aux aéroports…) ; le personnel

navigant est celui qui constitue les équipages des vols à effectuer : les pilotes, les copilotes, les hôtesses, ...

Compte tenu du fait que la charge de travail est assez importante, les plannings individuels ne sont jamais créés directement à partir des tâches ou des vols. Pour le personnel au sol, les tâches sont groupées dans un premier temps dans ce que l'on appelle des *vacations*. Les vols à effectuer par les navigants sont aussi groupés dans des séquences nommées *rotations*. C'est à partir de ces vacations/ rotations que les plannings sont ensuite construits.

Le deuxième chapitre passe en revue des problèmes d'optimisation combinatoire qui émergent à différents stades de la planification d'horaires : le problème d'affectation, le problème de bin-packing et le problème de recouvrement d'ensemble. Quelques techniques de base retenues pour la résolution de ces problèmes sont ensuite présentées : l'algorithme hongrois, le Bip Match et l'algorithme de Busacker et Gowen pour résoudre le problème d'affectation et quelques algorithmes approchés pour le problème de bin-packing. Quelques notions sur la génération de colonnes sont aussi présentées, car cette méthode est utilisée avec succès pour la résolution de problèmes de très grande taille, problèmes pour lesquels les méthodes classiques de programmation linéaire ne peuvent plus être utilisées.

Le problème de création de vacations avec ses deux versions (construction proprement dite ou remplissage) est présenté dans le troisième chapitre. Dans la littérature, il est traité le plus souvent comme un problème de recouvrement d'ensemble. Les méthodes de résolution proposées sont basées sur la programmation linéaire ou sur des méthodes itératives d'exploration de listes.

Comme ce problème repose sur la satisfaction de nombreuses contraintes temporelles, nous proposons une approche de modélisation basée sur le concept de plage horaire. Le modèle obtenu permet une représentation simple des contraintes, à partir de laquelle nous proposons une méthode heuristique de construction de vacations basée sur la méthode FFD (First Fit Decreasing) et sur la génération de colonnes. L'algorithme que nous proposons pour la construction de vacations peut aussi être utilisé, après quelques modifications, pour le problème de remplissage de vacations prédéfinies.

Le quatrième chapitre traite le problème de création de rotations. La création de rotations est un problème complexe, car, d'un côté, la réglementation à laquelle les rotations sont soumises traite beaucoup de cas particuliers et, d'un autre côté, la combinatoire du problème est considérable. C'est pour cette raison qu'il est habituellement résolu en deux étapes : une première étape, la construction de périodes de service, groupe les vols dans des journées de travail (ou périodes de service) ; la deuxième étape enchaîne les journées de travail, de manière à créer des rotations.

Après avoir présenté quelques méthodes proposées dans la littérature pour la construction de rotations, nous proposons une modélisation du problème de création de périodes de services par programmation linéaire en variables mixtes. L'avantage de cette méthode est qu'elle peut prendre en compte les différents types des périodes de services. Pour le groupement des périodes de services dans des rotations, un modèle linéaire en variables entières est utilisé.

L'utilisation d'un logiciel de programmation linéaire permet la résolution exacte d'une version relâchée de ce problème, obtenue en simplifiant certaines contraintes. Ceci présente l'intérêt majeur de fournir des bornes qui permettent d'évaluer la qualité des différentes méthodes heuristiques proposées pour la recherche de solutions admissibles performantes.

L'analyse du problème en termes de modélisation, de structure et des possibilités de décomposition permet de ramener sa résolution, globalement très complexe à celle d'un ensemble de sous-problèmes plus faciles et bien maîtrisés. Ainsi, la résolution exacte de certaines parties du problème global, a pu être intégrée efficacement dans la méthode globale heuristique de résolution.

Une approche similaire par décomposition et simplification est construite dans le dernier chapitre pour résoudre les problèmes d'affectation du personnel aux vacations et aux rotations.

Le problème d'affectation représente la deuxième étape dans la résolution du problème de création de plannings. Compte tenu de la similarité entre l'affectation des vacations et l'affectation des rotations, les deux cas sont présentés ensemble dans le cinquième chapitre. Après avoir énuméré les contraintes réglementaires communes ou spécifiques à chaque cas, nous analysons quelques modèles et méthodes de résolution proposés dans la littérature.

Ensuite l'approche que nous retenons est exposée. En raison des nombreuses contraintes réglementaires, le problème est modélisé comme un problème de multi-affectation généralisé. Nous proposons une décomposition temporelle et par qualification et ensuite une transformation du problème d'affectation généralisé en problème d'affectation simple par relaxation Lagrangienne. Un algorithme ad-hoc est utilisé pour la résolution de chaque problème de base.

IFR France a développé un produit commercial pour l'affectation de rotations au personnel navigant d'une compagnie aérienne qui utilise notre approche de résolution. Le chapitre V s'achève avec la présentation de ce produit.

Chapitre 1

Planification des ressources humaines

1.1 Introduction

La planification et la gestion optimale des ressources humaines jouent un rôle important dans la productivité et la compétitivité d'une entreprise. Les fonds alloués à la rémunération du personnel sont assez élevés : dans une compagnie aérienne, par exemple, les fonds alloués pour payer le personnel représentent les plus importantes dépenses [Anbil *et al.* 92] après le coût du carburant ; dans un établissement des soins, les coûts salariaux peuvent représenter jusqu'à 70% du budget opérationnel [Weil *et al*. 98]. Une gestion efficace du personnel est donc un facteur économique primordial.

Ce chapitre présente la problématique de la « planification des emplois du temps » dans un cadre général. Le problème de création de plannings dans un contexte aéronautique est décrit, ainsi qu'un passage en revue de quelques méthodes de modélisation et de résolution proposées dans la littérature.

1.2 Quelques notions sur la planification

La planification d'horaires de travail vise, pour un horizon de planification d'un jour à quelques mois, à dimensionner une force de travail et à optimiser l'utilisation de cette ressource, de façon a couvrir un besoin exprimé par une charge de travail prévisionnelle, tout en respectant un ensemble de contraintes précises. Elle aboutit à des programmes définissant les horaires de travail et de repos de la force de travail, en trouvant le meilleur compromis entre les préférences des différents acteurs [Rémi-Robert, 03].

Les contraintes qui viennent d'être mentionnées couvrent ce que l'on appelle *les aspects « JuSTE »* [Chan 02] :

Juridique : la législation en matière de droit du travail (durées de travail et de repos) sur différents horizons de temps (journalier, hebdomadaire, mensuel et annuel)

Social : répartition équitable des tâches entre salariés, entre hommes et femmes, avec respect des disponibilités, préférences individuelles et autres souhaits des salariés; répartition équitable du temps de travail et du repos

Technique : les règlements des différents métiers de l'entreprise (prise en compte des compétences et des niveaux requis)

Economique : respect des besoins de l'entreprise à chaque moment de l'horizon de planification. Cela se présente comme la meilleure adaptation de l'énergie disponible aux

charges à chaque moment de l'horizon. On cherche à ne pas dépenser inutilement cette énergie.

La difficulté de la création d'un planning est moins le processus de création en lui-même que l'obtention d'un planning de bonne qualité. Comment peut-on définir un planning de bonne qualité, alors qu'au sein d'une entreprise, chacun a son propre point de vue? Le planificateur souhaite que le planning soit assez flexible en ce qui concerne la main d'œuvre, pour pouvoir traiter les possibles imprévus. Pour le chef d'entreprise, le planning doit permettre le dimensionnement de la force de travail au plus juste et sa répartition de manière à obtenir le meilleur service au meilleur coût. Pour l'inspection du travail, le planning doit respecter dans les moindres détails la législation du travail et les conventions collectives. Le salarié recherche la satisfaction sociale de son travail, par rapport à ses disponibilités, à ses desiderata ou par rapport à l'équité de traitement. La création des plannings qui prennent en compte toutes ces contraintes, parfois contradictoires, implique à la fois une bonne connaissance du cadre de travail, mais aussi l'utilisation des techniques d'optimisation combinatoire.

Suivant leur spécificité et leur domaine d'application, les plannings portent différents noms : un planning précisant le programme de travail de chaque employé sur un mois s'appelle *tableau de service.*

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
Briand C.	M	M	S	M	N	N	*R*	*R*	N	S	M	S	S	*R*	S
Esquirol P.	M	N	S	N	*R*	M	M	S	M	M	N	*R*	*R*	S	N
Hennet J.-C.	M	S	N	*R*	N	S	S	S	N	*R*	M	S	N	N	N
Huguet M.-J.	N	N	*R*	M	S	M	S	S	N	*R*	N	N	S	M	S
Lopez P.	M	S	N	M	S	N	*R*	M	S	N	*R*	M	S	N	N
Mongeau M.	S	M	S	S	N	*R*	N	N	N	S	S	*R*	M	M	M

16	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*
M	M	S	M	N	N	*R*	*R*	N	S	M	S	S	*R*	S
M	N	S	N	*R*	M	M	S	M	M	N	*R*	*R*	S	N
M	S	N	*R*	N	S	S	S	N	*R*	M	S	N	N	N
N	N	*R*	M	S	M	S	S	N	*R*	N	N	S	M	S
M	S	N	M	S	N	*R*	M	S	N	*R*	M	S	N	N
S	M	S	S	N	*R*	N	N	N	S	S	*R*	M	M	M

(M=matin ; S=soir ; N=nuit ; *R*=repos)

Tableau 1.1. *Exemple de tableau de service*

Si le planning représente les programmes de travail et de repos non nominatifs sur un nombre entier de semaines, il s'agit de *grille de travail*. Si les horaires individuels sont périodiques, les plannings sont *cycliques*, sinon, on parle des plannings *acycliques*.

Semaine	*Lundi*	*Mardi*	*Mercredi*	*Jeudi*	*Vendredi*	*Samedi*	*Dimanche*
1	M	M	S	M	N	*R*	*R*
2	S	N	S	N	*R*	M	M
3	M	S	N	*R*	*R*	S	S
4	N	*R*	*R*	M	S	M	S

(M=matin ; S=soir ; N=nuit ; *R*=repos)

Tableau 1.2. *Exemple de grille de travail cyclique sur 4 semaines*

Même si les premiers travaux datent des années ‘50 (création d’horaires de travail pour les agents d’un poste de péage [Dantzig 54]), l’intérêt concernant le problème de création des plannings de personnel n’a pas diminué. De plus en plus nombreux sont les secteurs visés : l’industrie sidérurgique (industrie « à flux continu »), le secteur bancaire [Jacques 93], les services hospitaliers ([Brusco et Showalter 98], [Weil *et al*. 98], [Bellanti *et a*l. 04]), les centres d’appel ([Keith 79], [Grossman et al. 99]), les aéroports ([Jacquet-Lagrèze *et al*. 98], [Partouche 98]), les compagnies de transport aérien, ferroviaire ([Caprara *et al*. 98b]) et urbain ([Martello & Toth 86]), les commissariats de police et les gendarmeries, les services postaux, la restauration rapide ([Loucks & Jacobs 91]), les caissiers du secteur de la distribution ([Thompson 88]), etc.

Dans [Ernst *et al*. 04] on trouve une revue de littérature sur la planification de personnel en général. Des outils, des modèles et des méthodes utilisés dans différents secteurs sont détaillés.

1.3 La planification dans un contexte aéronautique

Dans un contexte aéronautique, la charge de travail est constituée des tâches à réaliser en vol et au sol, tâches qui sont déterminées la plupart du temps par le programme des vols. Suivant le type du travail à effectuer, le personnel se divise en deux catégories : le personnel au sol et le personnel navigant. Dans la suite de ce chapitre, nous allons présenter les problèmes spécifiques de chacune de ces deux catégories et quelques méthodes de modélisation et de résolution proposées dans la littérature. Dans les chapitres suivants, nous décrirons nos contributions à la résolution de ces problèmes.

1.3.1 Personnel au sol

Le programme des vols d’une compagnie aérienne impose un ensemble de tâches à effectuer au sol : l'enregistrement des passagers, l'accueil à l'arrivée et au départ, le service de correspondance et de transit aux aéroports. Ces tâches sont effectuées par ce que l’on appelle *le personnel au sol*.

Le problème de planification du personnel au sol n’est pas directement formulé à partir des tâches à effectuer ; il y a une étape intermédiaire : les tâches doivent être groupées en *vacations*, et ce sont ces vacations qui sont ensuite affectées au personnel. On appelle « vacation » un ensemble de tâches exécutées consécutivement par la même personne pendant son service journalier.

Suivant le type de contrat du personnel, les vacations doivent être *créées* ou *remplies* avant d’être affectées aux agents. Généralement, le contrat des agents spécifie des bornes inférieures et supérieures pour la durée du travail journalier et mensuel, l’existence des pauses pendant la journée de travail et leurs durées.

Ainsi, chaque type de contrat devient une sorte de « patron » à partir duquel des vacations seront construites et ensuite remplies avec des tâches. Pour plus de clarté, prenons un exemple : soit un agent dont le contrat impose un travail journalier de sept heures au minimum et dix heures au maximum, avec une pause prévue entre 8h00 et 14h00, d’une durée minimum d’une heure et maximum de deux heures. Une vacation qui commence à 6h00 du matin et qui finit à 14h30, avec une pause entre 9h30 et 11h00 respecte les demandes du contrat. Une autre vacation qui commence à 8h00 et finit à 16h00, avec une pause entre 12h15 et 14h00 est aussi une vacation « légale ». Il reste à trouver si les tâches à effectuer ce jour par cet agent peuvent être plus facilement insérées dans la première ou dans la deuxième vacation.

Si, par contre, le personnel travaille en grille, les vacations sont déjà fournies, car chaque case de la grille de travail correspond à une journée de travail, donc à une vacation vide. Les heures de début et de fin étant précisées, il reste seulement à les remplir avec des tâches. Si, par exemple, la grille de travail propre à un certain agent précise que mardi l'agent commence le travail à 7h30 et finit à 15h30, alors la vacation du mardi commence à 7h30 et finit à 15h30 et les tâches que l'agent va effectuer ce jour ne doivent pas dépasser cet intervalle de temps.

Dans la littérature, beaucoup de méthodes de résolution du problème de création des vacations reposent sur des approches par recouvrement d'ensemble. Dans une première étape, tout ou une partie de l'ensemble des vacations candidates sont générées ([Easton & Rossin 91], [Bechold & Brusco 94]). Ensuite, les vacations faisant partie de la solution finale sont sélectionnées selon différentes méthodes : à partir d'une première solution générée par la résolution d'un problème de recouvrement d'ensemble relâché ([Dantzig 54], [Morris & Showalter 83]) ou par une heuristique de construction itérative où les vacations sont sélectionnées les unes après les autres ([McGinnis et al. 78], [Brusco & Johns 96]).

[Segal 74] montre que la création des vacations peut être interprétée comme un problème de flot à coût minimal.

La méthode de résolution que nous proposons pour la création de vacations part d'une modélisation du problème en tant que problème de *bin-packing* : les tâches sont assimilées aux objets et les vacations aux conteneurs. Nous avons choisi cette modélisation car, malgré le fait que le problème de bin-packing est NP-difficile, il existe de nombreux algorithmes qui donnent des solutions approchées. La méthode proposée a en outre l'avantage de pouvoir être utilisée, sans lui apporter beaucoup de modifications, pour la création et pour le remplissage de vacations. Nous allons la présenter en détails dans le chapitre 3.

1.3.2 Personnel navigant

Le personnel navigant d'une compagnie aérienne est celui qui constitue les équipages des vols à effectuer : les pilotes, les copilotes, les hôtesses, ...

La création de plannings pour le personnel navigant est un problème qui attire considérablement l'attention des compagnies aériennes et en même temps des communautés de la recherche opérationnelle. Comme les navigants représentent l'une des ressources les plus coûteuses d'une compagnie, une gestion efficace du personnel s'impose naturellement comme indispensable. Le problème est intéressant aussi du point de vue mathématique, car il fait l'objet de nombreuses modélisations et de nombreuses techniques de résolution, qui relèvent le plus souvent de l'optimisation combinatoire.

La plupart des compagnies aériennes ont divisé la construction du planning en deux sous-problèmes : la création des *rotations* (problème connu aussi dans la littérature sous le nom de « *crew pairing* ») et l'affectation de ces dernières aux navigants (« *crew rostering* »). Une rotation est une séquence d'activités qui commence et qui finit dans la ville de domicile d'un équipage, appelée *base*. Elle peut couvrir un ou plusieurs jours.

Le but du premier sous-problème est de créer un ensemble de rotations à coût minimum qui couvre tous les vols à effectuer par la compagnie dans un certain intervalle temporel. Une fois les rotations créées, elles sont ensuite affectées aux personnels navigants (PN).

Les travaux dans le domaine sont nombreux. En 1969, [Arabeyre et al, 69] fait déjà un tour d'horizon des travaux dédiées à la création de rotations. Dans [Etschmaier et al, 85] et [Rushmeier *et al*, 95] on trouve des revues de littérature plus récentes.

Les dernières recherches formulent le problème de création de rotations comme un problème de partitionnement d'ensemble. Après avoir généré l'ensemble [Yan et al. 02] ou une partie [Chu *et al*. 97] des rotations admissibles, différentes heuristiques sont proposées

pour trouver la solution de coût minimum. Une autre approche consiste à grouper les vols en périodes de service [Desaulniers *et al*. 97a] et, à partir de ces dernières, à créer des rotations. Les références [Crainic & Rousseau 87] et [Lavoie *et al*. 88] proposent des méthodes de résolution basées sur la génération de colonnes.

L'affectation des rotations peut être réalisée suivant différentes approches. L'approche utilisée par la plupart des compagnies nord-américaines ('bidline approach') consiste à créer des plannings anonymes sur l'horizon concerné et à les affecter ensuite aux navigants, par ordre de priorité et aussi suivant leurs préférences. L'avantage de cette approche est que chacun sait exactement si le planning qu'il accepte est en concordance avec ses préférences. Les références [Campbell et al 97] et [Christou et al. 99] proposent des méthodes de constructions des plannings utilisant cette approche. Cette approche a en particulier été appliquée dans les compagnies FedEx et Delta Air Lines. Les plannings sont construits en deux étapes. Dans une première étape, on crée, en nombre maximal, des plannings individuels de très bonne qualité, c'est-à-dire qui respectent les contraintes (dures et souples) dans les moindres détails. Dans la deuxième étape on complète ces plannings à partir des rotations qui n'ont pas pu être affectées auparavant. Les plannings sont ensuite affectés aux navigants par ordre de priorité de ces derniers (qualification et ancienneté). La référence [Campbell *et al*. 97] utilise le recuit simulé pour la première étape et une procédure gloutonne pour la deuxième. L'étude [Christou *et a*l. 99] propose une procédure basée sur des algorithmes génétiques.

La plupart des compagnies européennes créent directement des plannings personnalisés pour chaque navigant. Dans ce type d'approche, le navigant exprime ses préférences avant la création du planning, sans connaître la configuration finale des rotations.

Pour la création de plannings personnalisés, plusieurs méthodes heuristiques ont été proposées, mais les méthodes basées sur le partitionnement d'ensemble («set partitioning») sont de plus en plus utilisées. [Ryan 92] propose une méthode heuristique pour la construction des plannings valides, en utilisant une matrice de contraintes qui facilite la recherche des solutions. Le problème est ensuite résolu par programmation en nombres entiers. [Gamache *et al*. 99] décrit une méthode de résolution basée sur la génération de colonnes. [El Moudani *et al*. 00] propose une méthode d'affectation des rotations utilisant des algorithmes génétiques. Cette méthode prend aussi en compte les éventuels changements de planning.

La programmation par contraintes est aussi utilisée pour résoudre le problème d'affectation de rotations. Cette technique est surtout utilisée dans les cas où le problème a beaucoup de contraintes et où on peut se contenter d'une solution faisable même si elle n'est pas optimale. Souvent, elle est combinée avec des techniques de recherche opérationnelle. Les auteurs [Guerinik *et al*. 95] utilisent la programmation par contraintes comme technique de prétraitement, pour réduire la dimension du problème. Ensuite, il est résolu à l'aide d'une méthode de programmation linéaire complétée par un algorithme de *Branch & Bound*. Une autre alternative est d'utiliser la programmation par contraintes pour générer des colonnes dans un problème de recouvrement d'ensemble *(set covering)* [Caprara *et al*. 98a]. Cette technique facilite la représentation des contraintes les plus complexes lors de la génération de colonnes, tout en garantissant la qualité de la solution, grâce à l'utilisation de la programmation en nombres entiers.

Pour résoudre le problème d'affectation de rotations, nous avons choisi l'approche « européenne » - des plannings personnalisés. Notre problème peut être vu comme un problème d'affectation multiple généralisé. Dans le chapitre 5, nous proposons une décomposition du problème dans un ensemble de problèmes d'affectation simple et un algorithme ad-hoc pour la résolution de chaque sous-problème.

1.4 Conclusions

Dans ce chapitre, nous avons présenté le problème de création des plannings dans différents domaines d'activité, en insistant particulièrement sur le contexte aéronautique.

Le tour d'horizon sur la littérature dans le domaine nous montre que ce problème fait l'objet d'intérêt de la communauté de Recherche Opérationnelle depuis plus d'une trentaine d'années. Plus récemment, des recherches ont été effectuées également dans le domaine de l'Intelligence Artificielle.

De nombreux modèles et méthodes ont été proposés pour la constitution de plannings. Dans le chapitre suivant, nous allons présenter quelques problèmes classiques d'optimisation combinatoire utilisée pour la modélisation des problèmes de création de plannings. Quelques méthodes couramment utilisées pour la résolution de ces problèmes d'optimisation combinatoire sont aussi décrites.

Chapitre 2

Modèles et techniques de résolution pour les problèmes de planification d'horaires

2.1 Introduction

La création de plannings relève de nombreux problèmes d'optimisation. Ces problèmes ressemblent aux problèmes d'optimisation combinatoire classiquement étudiés en recherche opérationnelle, mais présentent également certaines spécificités : issus d'applications réelles, ils sont le plus souvent complexes et de grande taille.

Les algorithmes de programmation linéaire en variables mixtes (continues et discrètes) sont des outils puissants pour la résolution de nombreux problèmes de recherche opérationnelle. Face à des problèmes réels de grande taille, ils perdent rapidement leur efficacité. C'est pourquoi, en pratique, pour des problèmes de planification, on se contente souvent de solutions faisables qui remplissent certains critères de qualité, même si elles ne sont pas optimales. Des méthodes approchées sont alors utilisées à la place des méthodes exactes. L'avantage de ces méthodes est qu'elles présentent un très bon rapport entre la qualité de la solution fournie et le temps de calcul.

Ce chapitre propose un bref passage en revue des problèmes d'optimisation combinatoire qui émergent dans différents stades de la planification d'horaires : le problème d'affectation, le problème de bin-packing et le problème de recouvrement d'ensemble. Nous présentons ensuite les techniques de base retenues pour la suite de nos travaux pour résoudre ces problèmes.

2.2 Problèmes classiques d'optimisation combinatoire

2.2.1 Problème d'affectation

Le problème d'affectation consiste à trouver des liens entre les éléments de deux ensemble distincts, de manière à minimiser un coût et à respecter des contraintes d'unicité de lien. En théorie des graphes, un problème d'affectation simple peut être vu comme un problème de couplage parfait dans un graphe biparti. Un graphe G s'appelle *biparti* si l'on peut diviser ses sommets en deux sous-ensembles et X_2 avec aucune arête de type *(i,j)* avec *i* et *j* dans X_1 ou bien *i* et *j* dans X_2. On appelle *couplage* d'un graphe *G* un sous-ensemble d'arêtes non adjacentes deux à deux.

Problème d'affectation simple

Soit un ensemble de m opérations qui doivent être exécutées par n ressources, $n \geq m$. Chaque couple (opération i, ressource j) (i = 1 à m, j = 1 à n) a un coût associé c_{ij}, qui représente la dépense associée à la réalisation de l'opération i par la ressource j. En supposant que chaque opération doit être exécutée une seule fois et que chaque ressource est utilisée au plus par une seule opération, le problème peut être modélisé de la manière suivante :

$$Min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij} \qquad (2.1)$$

$$\text{sous :} \sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in \{1,..,m\} \qquad (2.2)$$

$$\sum_{i=1}^{m} x_{ij} \leq 1 \qquad \forall j \in \{1,..,n\} \qquad (2.3)$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in \{1,..,m\},\ \forall j \in \{1,..,n\} \qquad (2.4)$$

où x_{ij} est une variable binaire associée à chaque paire (opération i, ressource j) et qui vaut 1 si l'opération i est effectuée par la ressource j et 0 sinon. Si l'on considère que X_1 représente l'ensemble des opérations ($|X_1| = m$) et X_2 l'ensemble des ressources ($|X_2| = n$) du problème (2.1)-(2.4), avec $m \leq n$ alors chaque arête entre un sommet i de X_1 et un sommet j de X_2 signifie que l'opération i peut être effectuée par la ressource j. On associe à chacune de ces arêtes un coût égal à c_{ij}. Comme le coût total d'un couplage est donné par la somme des coûts de chaque arête, le problème d'affectation revient à trouver un couplage de cardinal m et de coût minimal.
Ce modèle correspond à ce que l'on appelle *le problème d'affectation simple*.

Problème d'affectation généralisé

Dans un cadre plus général, où par exemple les ressources sont agrégées, la contrainte (2.3), appelé *contrainte de capacité*, n'est plus suffisante pour décrire l'utilisation des ressources. Dans ce cas, une ressource j est caractérisée par sa capacité S_j et chaque opération i nécessite la quantité de ressource r_{ij} si elle utilise la ressource j. La contrainte (2.3) devient :

$$\sum_{i=1}^{m} r_{ij}^{k} x_{ij} \leq S_j^k \qquad \forall j \in \{1,..,n\}, \forall k \in \{1,...,K\} \qquad (2.5)$$

Dans la formulation du problème, l'indice k traduit l'existence fréquente de plusieurs contraintes de capacité généralement liées à l'utilisation de ressources annexes ou complémentaires par rapport aux ressources de base [Hennet 97]. Le problème d'affectation simple ayant des contraintes de capacité exprimées par (2.5) devient un *problème d'affectation généralisé.*

2.2.2 Problème de bin-packing

Un des problèmes les plus souvent rencontrés dans le domaine de la recherche opérationnelle est le problème de bin-packing. Celui-ci consiste à placer des objets ayant une taille donnée dans des conteneurs de capacité connue. Dans les problèmes les plus basiques, les tailles et les capacités ont une seule dimension (poids, longueur…) et tous les conteneurs ont la même capacité. Dans les problèmes plus complexes, il y a plusieurs types de conteneurs et les objets ont plusieurs dimensions. Souvent, il s'agit de dimensions géométriques ou de

caractéristiques physiques (poids, volume).

Les objectifs les plus fréquents sont soit de remplir au maximum les conteneurs utilisés, sachant que les objets ne doivent pas forcément être tous sélectionnés, soit de placer tous les objets dans un nombre minimal de conteneurs, ou encore d'équilibrer les charges.

[Dyckhoff 90] propose une typologie des problèmes de bin-packing, chaque type étant caractérisé par les attributs suivants :

Dimension. Ce paramètre précise le nombre minimal des variables nécessaires à la description géométrique du modèle : une dimension (1), deux dimensions (2), trois dimensions (3), N dimensions (N).

Type d'assignation. Deux cas sont envisagés : soit tous les objets sont affectés à une sélection de conteneurs (V), soit une sélection d'objets est affectée à tous les conteneurs (B).

Capacité des containeurs. Soit il y un seul conteneur (O) soit il y en a plusieurs avec des capacités identiques (I) ou avec des capacités différentes (D).

Taille des objets. Il peut y avoir peu d'objets de formes différentes (F), plusieurs objets de plusieurs formes différentes (M), plusieurs objets de peu de formes différentes (R), ou des objets de formes identiques (C).

D'une façon classique [Guéret *et al.* 00], un problème élémentaire de bin-packing peut être formulé de la manière suivante :

$$Min \sum_{j=1}^{n} y_j \qquad (2.6)$$

$$\text{sous : } \sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in \{1,...,m\} \qquad (2.7)$$

$$\sum_{i=1}^{m} a_i . x_{ij} \le b_j . y_j \qquad \forall j \in \{1,...,n\} \qquad (2.8)$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in \{1,...,m\},\ \forall j \in \{1,...,n\} \qquad (2.9)$$

$$y_j \in \{0,1\} \qquad \forall j \in \{1,...,n\} \qquad (2.10)$$

ou m est le nombre des objets à placer dans des conteneurs, n est le nombre des conteneurs, a_i donne la taille de l'objet i, b_j donne la capacité du conteneur j, x_{ij} est une variable binaire qui vaut 1 si l'objet i se trouve dans le conteneur j, 0 sinon et y_j est une variable binaire qui vaut 1 si le conteneur j fait partie de la solution, 0 sinon.

Il est bien connu que le problème de bin-packing est un problème NP-difficile [Coffman *et al.* 98]. Cependant, il existe de nombreux algorithmes de résolution approchés efficaces. Les plus performants parmi ces algorithmes seront analysés dans le paragraphe § 2.3.4.

2.2.3 Problème de recouvrement d'ensemble

Un problème de recouvrement d'ensemble (*set covering problem*) peut être modélisé sous la forme suivante :

$$Min \sum_{j=1}^{n} c_j x_j \qquad (2.11)$$

$$\text{sous : } \sum_{j=1}^{n} a_{ij} x_j \ge 1 \qquad \forall i \in \{1,..,m\} \qquad (2.12)$$

$$x_j \in \{0,1\} \qquad \forall j \in \{1,..,n\} \qquad (2.13)$$

Si l'inégalité (2.12) est remplacée par une égalité, on a un problème de partitionnement d'ensemble (*set partitioning problem*). Dans le cas où l'inégalité ($\geq$) change de sens ($\leq$), on a un problème d'emballage (*set packing problem*).

De nombreuses applications peuvent être modélisées comme problèmes de recouvrement d'ensemble. Les problèmes de livraison ou de tournée, les problèmes de planification ou les problèmes de localisation d'entrepôts sont traités comme problèmes de recouvrement d'ensemble dès que l'on veut s'assurer que chaque client est servi par un certain entrepôt, véhicule ou personne. La référence [Balas & Padberg 76] contient une bibliographie importante sur les domaines d'applications du problème de recouvrement d'ensemble.

Les problèmes modélisés en tant que problèmes de recouvrement d'ensemble sont généralement des problèmes de très grande taille et ne peuvent pas être formulés explicitement. Une des techniques les plus utilisées pour la résolution de ce genre de problèmes est la *génération de colonnes*. Nous allons discuter plus en détail de la génération de colonnes dans § 2.3.1.

La génération de colonnes a été utilisée pour la première fois pour la résolution d'un problème de découpe [Gilmore&Gomory 61]. Des problèmes d'affectation et d'ordonnancement de tâches [Savelsbergh 97], [Van Den Akker *et al*. 00], des problèmes de tournée avec contraintes supplémentaires – fenêtres de temps et contraintes de ressources – [Desrosiers *et al.* 84, Desrochers *et al*. 92, Ribeiro & Soumis 94] et des problèmes de construction de plannings dans le domaine des transports aériens [Desaulniers *et al.* 97a], [Barnhart *et al*. 98a] ont aussi été résolus en utilisant cette méthode.

2.3 Techniques de résolution retenues

Dans la paragraphe précédent, nous avons vu quelques modèles de recherche opérationnelle souvent rencontrés pour modéliser des problèmes liés à l'affectation du personnel. Ce paragraphe présente les techniques les plus performantes pour résoudre ces problèmes de recherche opérationnelle.

2.3.1 Méthodes de résolution de problèmes de couplage et d'affectation

Le problème d'affectation simple (2.1)-(2.4) est parmi les problèmes combinatoires les plus faciles à résoudre. En effet, sa matrice de contraintes pour les contraintes (2.2)-(2.3) a la propriété d'unimodularité. Les contraintes binaires (2.4) peuvent donc être remplacées par les contraintes de la relaxation continue suivante sans changer la solution optimale du problème qui est « naturellement » binaire.

$$0 \leq x_{ij} \leq 1 \qquad \forall i \in \{1,..,m\},\ \forall j \in \{1,..,n\} \qquad (2.14)$$

Une conséquence évidente de cette propriété est que des problèmes d'affectation simples de très grande dimension peuvent être résolus directement par des logiciels de programmation linéaire en nombres réels.

Cependant, pour des raisons de souplesse d'utilisation et d'implémentation dans des progiciels commerciaux, nous avons choisi de résoudre les problèmes d'affectation simples par des algorithmes ad-hoc, que nous allons maintenant passer en revue.

2.3.1.1 Bip Match

Le *Bip Match* est un algorithme rapide utilisant les chaînes alternées améliorantes pour trouver un couplage maximal dans un graphe biparti [Prins 94]. Pour un graphe G et un couplage C, une chaîne est *alternée* si elle emprunte alternativement des arêtes dans C et hors

C. Pour un couplage, une chaîne alternée est *améliorante* (ou *augmentante*) si ses deux extrémités sont insaturées par le couplage.

Le fonctionnement de l'algorithme est basé sur les deux suivantes propriétés dues à Berge [Berge 83] :

Propriété 1. Soit un graphe *G*, C un couplage de G et une chaîne alternée améliorante μ pour C. Un transfert sur μ donne un nouveau couplage C' avec |C'| = |C| + 1.[1]

Propriété 2. Un couplage est maximal si est seulement si il n'admet aucune chaîne alternée améliorante.

Le *Bip Match* part d'un couplage initial non vide, obtenu en cherchant les chaînes alternées améliorantes réduites à une arête - on affecte chaque sommet de X à son premier successeur libre, c'est-à-dire n'étant pas déjà extrémité d'une arête du couplage.

A partir de cette affectation initiale, on cherche les chaînes améliorantes, par une exploration en largeur du graphe. Cette recherche vise à améliorer la solution trouvée par l'affectation initiale, puis la solution courante, par l'augmentation de la cardinalité du couplage. On part toujours d'un sommet insaturé de X et on s'arrête à la plus courte chaîne trouvée. On va explorer les sommets alternativement (un sommet de X, un sommet de Y…), de manière que la chaîne ait une arête hors du couplage, puis une arête dans le couplage. L'algorithme s'arrête quand il n'arrive plus à trouver des chaînes améliorantes.

La recherche du couplage maximal est schématisée dans la Figure 2.1 :

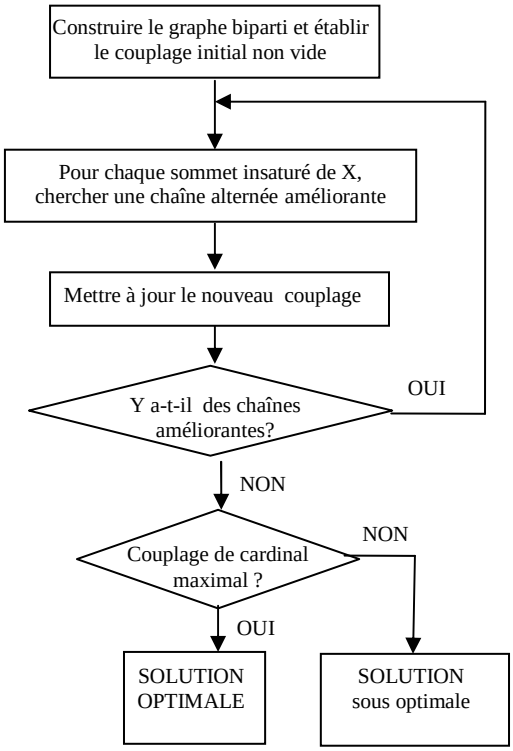


Figure 2.1 *Le Bip Match*

[1] L'opération de *transfert sur μ* consiste à enlever de C les arêtes de μ qui faisaient partie du couplage et à ajouter à C les arêtes de μ qui n'en faisant pas partie. Ceci revient à inverser le statut *dans C / hors C* des arêtes de μ.

Le Bip Match s'inscrit dans la catégorie des méthodes primales. En effet, la recherche des chaînes alternées améliorantes qui augmente la cardinalité du couplage correspond à la construction d'une séquence de solutions admissibles qui converge vers l'optimum. L'avantage d'une méthode primale est d'être très rapide dans le cas de plusieurs solutions équivalentes. Comme le but est d'améliorer la solution initiale, les solutions équivalentes ne sont pas explorées. S'il y a une solution meilleure, elle est retenue, sinon, on arrête l'exploration.

2.3.1.2 L'algorithme de Busacker et Gowen

Un problème d'affectation ou de couplage maximal dans un graphe biparti peut être modélisé comme un problème de flot ; ainsi, la recherche d'une affectation optimale revient à la recherche d'un flot maximal de coût minimal. C'est un problème classique de la théorie des graphes qui cherche à faire passer un débit maximal à travers un réseau à moindre coût.

L'algorithme de Busacker et Gowen [Busacker&Gowen 61] est une méthode simple et efficace pour trouver un tel flot. Son principe de fonctionnement est équivalent à celui du Bip Match : il part d'un flot nul et il l'augmente progressivement, par recherche des chaînes augmentantes.

C'est aussi une méthode primale comme le Bip Match, donc il présente les mêmes avantages que celui-ci. De plus, il prend en compte les coûts d'affectation.

2.3.1.3 L'algorithme hongrois

Un cas particulier du problème d'affectation, souvent traité dans la littérature est celui où le nombre des opérations à effectuer est égal au nombre des ressources utilisées. En effet, tout problème d'affectation simple peut être mis sous cette forme par introduction d'opérations fictives à coût nul pour toutes les ressources dans le cas $m<n$, ou de ressources fictives à coût nul pour toutes les opérations dans le cas $m>n$. Une méthode de résolution du problème d'affectation qui prend en compte cette structure est l'algorithme hongrois.

Développé en 1955 par H.W.Kuhn et nommé ainsi en honneur des mathématiciens J. Egervary et D. König, l'algorithme hongrois [Kuhn 55] est un algorithme dual qui utilise la modélisation sous forme de programme linéaire du problème d'affectation. Toutefois, il peut être vu comme une variante de l'algorithme de Busacker et Gowen, spécialisée pour la structure bipartie du graphe. Du fait de sa grande efficacité sur ce type de problème, c'est l'algorithme de référence en Recherche Opérationnelle pour résoudre le problème d'affectation. Son principe est basé sur le fait que les couplages de poids minimal dans le graphe du problème primal sont exactement les couplages de cardinalité maximale dans le graphe du problème dual. Une schématisation simpliste de l'algorithme hongrois pourrait être la suivante :

Etape 1. On crée la matrice des coûts (c_{ij}) ($i = 1…m, j = 1…n, m = n$) initiale, qui est considérée non négative.

Etape 2. Pour chaque ligne de la matrice, on trouve l'élément minimal et on le soustrait à tous les éléments de cette ligne. S'il reste des colonnes sans coefficients nuls, pour chacune de ces colonnes, on trouve l'élément minimal et on le soustrait à tous les éléments de cette colonne.

Etape 3. On cherche une affectation admissible en vérifiant si, pour chaque ligne et chaque colonne il existe exactement une case marquée. Si oui, l'affectation trouvée est optimale. Sinon, on trace des traits (en nombre minimal) sur les lignes et les colonnes de la matrice, pour couvrir tout les 0 au moins une fois.

Etape 4. On cherche l'élément minimal non marqué. On le soustrait à tous les éléments non marqués et on l'additionne à tous les éléments marqués deux fois (tous les éléments qui se trouvent à l'intersection de deux traits). On retourne à *Etape 3*.

La Figure 2.2 schématise l'algorithme hongrois.

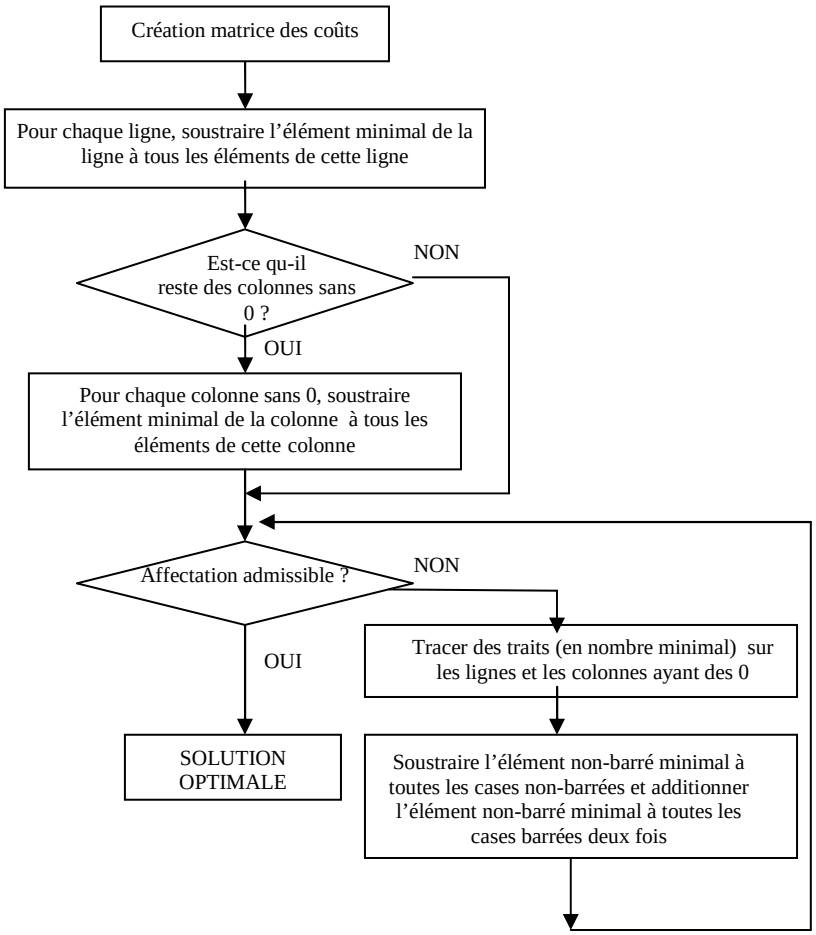


Figure 2.2 *L'algorithme hongrois*

2.3.1.4 Méthode de relaxation Lagrangienne pour le problème d'affectation généralisé

Pour résoudre un problème d'affectation généralisé, on peut chercher à le transformer en un problème d'affectation simple et ensuite résoudre ce problème. Une méthode fréquemment utilisée pour ce genre de transformation est la relaxation Lagrangienne. En dualisant la contrainte (2.5), le critère (2.1) est remplacé par l'expression lagrangienne suivante :

$$L = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{j=1}^{n}\sum_{k=1}^{K} \lambda_{ij}^{k} (\sum_{i=1}^{m} r_{ij}^{k} x_{ij} - S_i^{k}) \qquad (2.15)$$

En relaxant les contraintes de capacité (2.5) et en pondérant leur non-respect, on a à résoudre un problème d'affectation simple, avec les paramètres de coûts c_{ij} remplacés par les paramètres c_{ij}' définis par :

$$c'_{ij} = c_{ij} + \sum_{k=1}^{K} \lambda_{ij}^{k} r_{ij}^{k} \qquad (2.16)$$

Le choix des valeurs numériques, nécessairement non négatives, des variables duales λ_{ij}^{k} est bien sûr très important, car la solution optimale du problème d'affectation généralisé est obtenue pour les valeurs optimales de ces variables duales. Des valeurs candidates pour les paramètres λ_{ij}^{k} peuvent être obtenues en résolvant le problème d'affectation généralisé dans lequel les contraintes binaires sur les variables ont été relaxées. Cela revient à résoudre le problème en remplaçant (2.4) par (2.14):

$$0 \le x_{ij} \le 1 \qquad \forall i \in \{1,..,m\},\ \forall j \in \{1,..,n\}$$

Pour la résolution d'un problème d'affectation simple, les algorithmes décrits ci-dessus peuvent être utilisés.

2.3.2 Algorithmes de résolution pour le problème de *bin-packing*

Il est bien connu que le problème de bin-packing est un problème NP-difficile. Il y a toutefois de nombreux algorithmes de résolution qui donnent des solutions approchées en un temps de réponse polynomial. La qualité de chacun de ces algorithmes est donnée par la garantie de performance dans le pire cas [Coffman *et al.* 98]. Soit *A* un algorithme qui donne une solution approchée et *OPT* un algorithme qui donne la solution optimale. *A(k)* représente le nombre de conteneurs utilisés par l'algorithme A pour caser *k* objets et *OPT(k)* représente le nombre optimal de conteneurs pour caser les mêmes *k* objets. Avec ces notations, la garantie de performance de l'algorithme A, dans le pire cas est définie par :

$$R_A^{\infty} = \lim_{k \to \infty} \frac{A(k)}{OPT(k)} \qquad (2.17)$$

Dans la suite, nous allons présenter quelques algorithmes et leurs performances. Auparavant, nous donnons quelques précisions :

Tous les conteneurs sont vides au début. Chaque conteneur qui n'est pas vide est soit *fermé*, soit *ouvert*. Un conteneur est ouvert si l'on peut encore mettre des objets dedans. Un conteneur fermé n'est plus disponible pour recevoir des objets; une fois fermé, il n'est plus rouvert.

Les objets à mettre dans des conteneurs se trouvent dans une liste L = { o_1, o_2, o_3…}. Cette liste peut être triée suivant la taille des objets, mais la règle de tri dépend de l'algorithme choisi.

Suivant la modalité d'inspection des objets à caser dans des conteneurs, les algorithmes sont divisés en deux catégories : algorithmes *on-line* et algorithmes *off-line*.

2.3.2.1 Algorithmes on-line

Un *algorithme on-line* traite les objets dans leur ordre, sans aucune connaissance de la taille ou du nombre des objets pas encore traités. Les conteneurs sont traités dans l'ordre C_1, C_2, C_3… Comme règle générale, l'algorithme commence par mettre l'objet o_1 dans le conteneur C_1 et ensuite il traite les autres objets dans l'ordre o_2, o_3, o_4… Parmi les algorithmes on-line les plus connus, on trouve : *Next-Fit*, *Worst-Fit*, *First-Fit*, *Best-Fit*.

Next-Fit (NF) est peut être le plus simple des algorithmes de résolution du bin–packing. Une fois le premier objet placé, *NF* met chaque objet restant dans le conteneur du dernier objet placé. Si il n'y a pas de place, il ferme ce conteneur et place l'objet courant dans un conteneur vide. Cet algorithme a une complexité en *O(n)* et une garantie $R_{NF}^{\infty} = 2$. Une analyse complète concernant R_{NF}^{∞} se trouve dans [Johnson 74].

Worst-Fit (WF) place l'objet courant dans le conteneur le moins rempli où l'objet peut rentrer. Si il y a plusieurs conteneurs de ce type, celui avec l'index le plus petit est choisi. S'il n'y a pas de conteneur ouvert dans lequel l'objet courant peut être placé, un conteneur vide est ouvert. Cet algorithme a la même garantie que *NF* [Johnson 74].

First-Fit (FF) place l'objet courant dans le premier conteneur ouvert ayant le plus petit index où l'objet rentre. Si il n'y a pas un tel conteneur, un autre sera ouvert. L'algorithme a une complexité en *O(n log n)*, donc plus élevée que celle du NF, mais une meilleure garantie de performance relative au pire, car $R_{FF}^{\infty} = 17/10$ [Johnson *et al.* 74].

Best-Fit (BF) place l'objet courant dans le conteneur le plus rempli où l'objet rentre. Si il y en a plusieurs équivalents, celui avec le plus petit index est choisi. Si il n'y a pas de conteneur qui puisse recevoir cet objet, un conteneur vide est ouvert. L'algorithme a une complexité O(n *log* n) et une garantie de performance relative au pire $R_{BF}^{\infty} = 17/10$ [Johnson *et al.* 74].

2.3.2.2 Algorithmes off-line

Un *algorithme off-line* permet le tri des objets avant de commencer à les ranger dans les conteneurs. La règle la plus naturelle trie les objets en ordre décroissant de leur taille. En effet, un gros objet est plus facile à insérer lorsque le choix des conteneurs candidats est important. Selon la même logique, il est plus facile de compléter un conteneur déjà rempli avec un petit objet qu'avec un gros.

En combinant cette règle de tri avec les algorithmes qui viennent d'être présentés, on obtient les algorithmes suivants : *Next Fit Decreasing (NFD)*, *First Fit Decreasing (FFD)*, *Best Fit Decreasing (BFD)*. *FFD* et BFD ont la même garantie de performance : $R_{FFD}^{\infty} = R_{BFD}^{\infty} = 11/9$, donc meilleure que dans le cas on-line.

Pendant des années, le *FFD* a été considéré comme le meilleur algorithme d'approximation pour la résolution du problème de bin-packing. Toutefois, des modifications ont été apportées au FFD, en essayant de trouver une meilleure garantie. Yao [Yao 80] a créé l'algorithme *Refined First Fit Decreasing (RFFD)* dont la garantie de performance est $R_{RFFD}^{\infty} \leq 11/9 - 10^{-7}$, mais avec une complexité en $O(n^{10} \log n)$. Ensuite, [Garey & Johnson 85] ont proposé l'algorithme *Modified First Fit Decreasing (MFFD)* qui améliore encore plus le FFD. Sa complexité est la même que celle du FFD, *O(n log n)*, mais la garantie de performance est meilleure : $R_{MFFD}^{\infty} = 71/60$.

Plus de détails concernant les différents algorithmes présentés ci-dessus figurent dans [Coffman *et al*. 98].

2.3.3 Génération de colonnes pour la résolution du problème de recouvrement d'ensemble

Une particularité du problème de recouvrement d'ensemble décrit par (2.11) – (2.13) est que, dans de nombreuses application, il a un très grand nombre de variables, ce qui est équivalent au fait que la matrice (a_{ij}) $(i = 1…m, j = 1…n)$ a un très grand nombre de colonnes

($n>>m$), de telle sorte que toutes ses colonnes ne peuvent pas être explicitées. La question qui se pose alors est de construire une technique capable de résoudre un tel problème, sans tenir compte en même temps de façon explicite de toutes les variables du problème.

Une telle technique est la génération de colonnes. Le principe consiste à ne manipuler qu'un nombre restreint de variables à la fois et à identifier (générer) les variables ou les combinaisons de variables (colonnes) entrantes dans la solution au cours de la résolution, sans les énumérer explicitement.

D'une manière plus générale, on peut réécrire (2.14) – (2.16) sous la forme suivante :

$$(P_1): \begin{cases} Min & cx \\ sous & Ax \geq b \\ & x \geq 0 \end{cases} \qquad (2.18)$$

où c et x sont des vecteurs de dimension n, A est une matrice $m \times n$, et b est un vecteur de dimension m. Même si la matrice A ne peut pas être explicitée, on suppose qu'elle est connue implicitement, c'est-à-dire que ses colonnes correspondent aux éléments d'un ensemble que l'on sait caractériser. On suppose aussi que l'on dispose d'un algorithme « *générateur* » pour générer une colonne de A minimisant une fonction linéaire de la forme :

$$z(A_j) = c_j - \pi A_j \qquad (2.19)$$

ou $\pi = (\pi_1, .., \pi_m)$ est un vecteur ligne de paramètres.

Soit Ω un sous-ensemble des variables du problème (P_1) et A' la sous-matrice de A correspondante ($rang(A') = m$). On note (P_2) la restriction de (P_1) à Ω. En utilisant par exemple l'algorithme du simplexe[2], (P_2) a comme solution optimale :

$$x_B = B^{*-1} b \qquad (2.20)$$

où B^* est une base de A', avec $rang\ (B^*) = m$.

Cette solution est admissible pour P_1, mais pas forcément optimale. Pour l'améliorer, on suit la méthode du simplexe : on cherche une colonne hors base dont le coût réduit est négatif, pour la faire entrer dans la base. C'est ici que l'algorithme générateur intervient. Comme la colonne que l'on cherche ne se trouve pas dans Ω, il faut donc exhiber une ou plusieurs variables non explicitée de P_1 et de générer une nouvelle colonne pour A. On sait que le coût marginal d'une variable hors base est :

$$\bar{c}_j = c_j - \pi A_j \qquad (2.21)$$

où π est le vecteur de multiplicateurs du simplexe associés à B*. A l'aide de l'algorithme générateur, on peut générer la colonne A_r de manière que :

$$\bar{c}_r = \underset{j}{Min}(c_j - \pi A_j) \qquad (2.22)$$

Tant que l'on trouve un c_r négatif, on ajoute x_r à Ω, A_r à A' et on initie une nouvelle itération, en résolvant le nouveau programme linéaire. Lorsque l'algorithme générateur détermine $c_r \geq 0$, l'algorithme s'arrête : toutes les variables hors base ont alors un coût réduit positif ou nul ; la solution courante est donc minimale pour P_1.

D'une façon synthétique, l'approche de résolution par génération de colonnes peut être vue comme un schéma itératif de coopération entre deux types de problèmes plus faciles à résoudre que le problème global :

- *le problème maître*, qui est une restriction à un sous-ensemble de variables du problème P_1 initial et donc, à chaque itération on ne résout que cette restriction ;

[2] Pour une présentation plus détaillé sur le simplexe, le lecteur peut se référer à des ouvrages spécialisés comme [Dantzig 63], [Lasdon 70] ou, plus récemment, [Gondran & Minoux 95].

– *le problème esclave*, résolu à chaque itération par un algorithme générateur qui consiste à générer, si elle existe, une colonne de coût réduit négatif qui, insérée dans la matrice du problème maître restreint courant, permet d'améliorer la solution courante.

La **Figure 2.3**, présente le schéma de coopération entre les deux problèmes (cf. [Mancel 04]) :

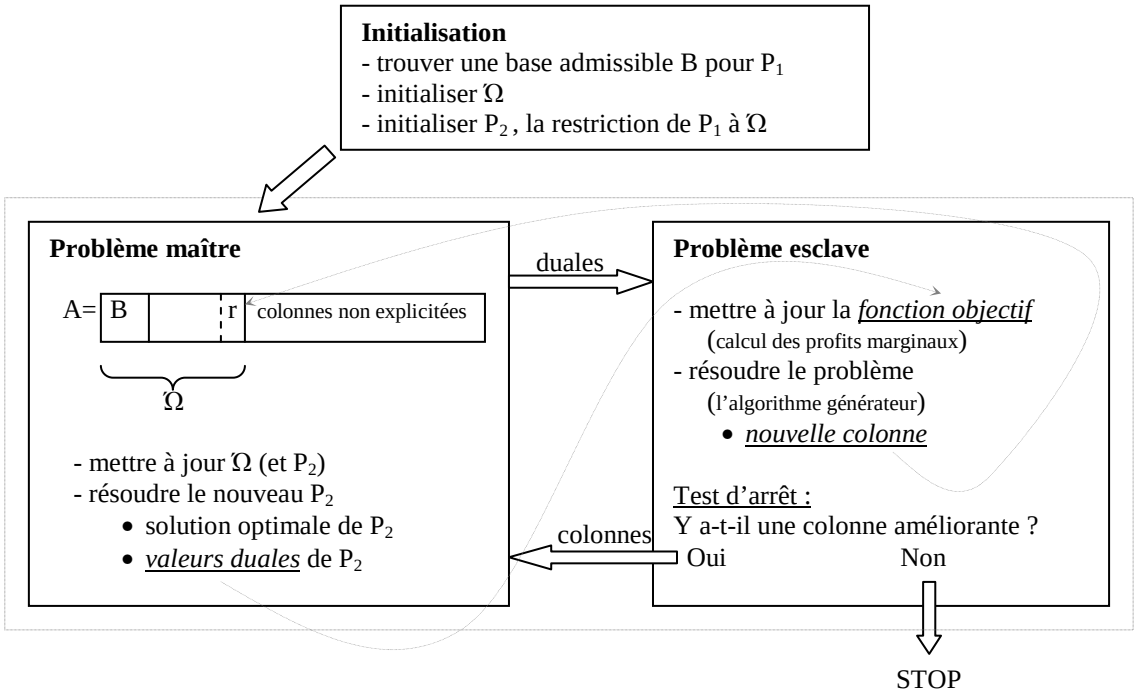


Figure 2.3 *Schéma de fonctionnement de la génération de colonnes*

2.3.3.1 Convergence de la méthode

Comme le simplexe, la génération de colonnes est une méthode convergente et exacte. La vitesse de convergence dépend du nombre d'itérations nécessaires jusqu'à la preuve d'optimalité, autrement dit, dépend de l'efficacité de l'algorithme générateur. Le rôle de cet algorithme est de fournir à chaque itération une nouvelle colonne dont le coût marginal négatif peut améliorer la valeur de la fonction objectif. Même si théoriquement, n'importe quelle colonne de coût négatif peut améliorer la solution courante, il se trouve qu'en pratique un compromis doit être trouvé entre la vitesse de convergence et la complexité de l'algorithme générateur. L'algorithme générateur peut donc résoudre un problème esclave restreint, ou bien faire appel à des méthodes approchées, afin de réduire le temps de calcul. De toute façon, il faut noter que pour sortir du processus itératif il faut utiliser un algorithme exact pour prouver qu'il n'y a aucune colonne de coût réduit négatif.

Une méthode couramment pratiquée pour l'accélération de la génération de colonnes est la génération simultanée de plusieurs colonnes améliorantes à chaque itération [Lasdon 70], [Gamache *et al*. 99]. Ceci permet de réduire le nombre d'itérations nécessaires pour générer les colonnes de la base optimale du problème.

Un autre aspect important lié à la convergence est le fait que les valeurs duales convergent vers l'optimum en oscillant de manière irrégulière. Différents travaux de recherche portent sur les techniques d'accélération de la convergence, consistant à contrôler

au mieux ces valeurs duales. Parmi ces travaux, les méthodes de stabilisation de la procédure de résolution proposées par [Merle et al. 99] puis [Ben Amor 02] sont particulièrement efficaces ; elles se basent sur l'introduction des perturbations dans le problème maître, couplées avec des bornes sur les variables duales. Les résultats reportés dans ces travaux montrent d'importantes réductions du nombre d'itérations et des temps de calcul sur différents problèmes, ainsi que la possibilité de résoudre des problèmes de plus grande taille.

2.3.3.2 L'algorithme générateur

Pour générer de nouvelles colonnes pour le problème maître, une des méthodes les plus souvent utilisées dans la littérature est la recherche de plus court chemin dans un graphe. En effet, une colonne peut souvent être interprétée comme un chemin dans un graphe ou comme un flot dans un réseau.

Les algorithmes générateurs font souvent appel à la théorie des graphes et la programmation dynamique et, plus rarement, à d'autres méthodes comme la programmation linéaire ou la programmation non linéaire (cf. [Lasdon 70]). Ainsi, la génération d'une nouvelle colonne se ramène fréquemment à un problème de plus court chemin dans un graphe avec des contraintes supplémentaires. Les algorithmes générateurs développés pour ce type de problème se basent essentiellement sur un algorithme de programmation dynamique, pseudo-polynomial, présenté dans [Desrochers et Soumis 88, Desrochers *et al*. 92] pour la résolution par génération de colonnes d'un problème de tournée de bus scolaires (avec fenêtres temporelles et contraintes de ressources). Le principe de cet algorithme est d'associer à chaque chemin partiel du graphe une étiquette fournissant une évaluation multi-critère (sur le temps et la consommation des différentes ressources) et d'éliminer des chemins partiels grâce à des règles de dominance. C'est une extension de l'algorithme classique de Bellman-Ford pour la recherche de plus courts chemins dans un graphe [Prins 94].

Des méthodes approchées peuvent également être utilisées pour générer les colonnes du problème maître ; elles se basent généralement sur les mêmes principes mais se contentent de rechercher un chemin réalisable dans le graphe, satisfaisant l'ensemble des contraintes du sous-problème. Cependant, à une itération donnée, le fait de ne pas trouver de colonne améliorante, par un algorithme heuristique, ne prouve pas l'optimalité de la solution courante du problème maître. Pour terminer le processus itératif, on doit donc utiliser un algorithme générateur exact.

2.3.3.3 Recherche des solutions entières

La génération de colonnes appliquée à un problème maître en nombres entiers fournit une solution optimale de sa relaxation linéaire. Généralement, cette solution n'est pas entière, mais elle représente une borne inférieure de la solution entière du problème maître.

Une première approche pour la recherche d'une solution entière consiste à lancer une recherche arborescente classique de type B*ranch & Bound* sur le problème maître restreint à l'ensemble Ω des colonnes explicitées lors du processus de génération de colonnes. C'est une approche heuristique car, même si l'ensemble de colonnes Ω permet de trouver la solution optimale continue du problème maître à la racine de l'arborescence, on n'est pas assuré d'obtenir la solution optimale (ni même une solution réalisable) aux autres noeuds de l'arbre, car des colonnes hors Ω, sont susceptibles d'améliorer la solution (ou de rendre le problème admissible) au cours de la recherche arborescente.

Ainsi, une deuxième approche, appelée B*ranch & Price* [Barnhart *et al.* 98b, Vanderbeck & Wolsey 96, Vanderbeck 00], adapte les principes du *Branch & Bound* au contexte de la génération de colonnes, en permettant d'expliciter de nouvelles colonnes à

chaque noeud de l'arbre, c'est-à-dire en utilisant une procédure d'évaluation des noeuds par la méthode de génération de colonnes. Les techniques de séparation doivent alors permettre de supprimer des solutions fractionnaires, tout en n'introduisant que des contraintes qui puissent être prises en compte aisément par l'algorithme générateur.

2.4 Conclusion

Dans ce chapitre, nous avons présenté quelques modèles et méthodes que nous allons utiliser dans les chapitres suivants pour la modélisation et la résolution du problème de création de plannings. Dans la suite, nous allons voir que des sous-problèmes du problème de création de plannings peuvent être modélisés en tant que problèmes d'affectation ou de bin-packing. Les principes de ces problèmes ont été présentés ainsi que différentes méthodes de résolution : l'algorithme hongrois, le Bip Match et l'algorithme de Busacker et Gowen pour résoudre le problème d'affectation et quelques algorithmes approchés pour le problème de bin-packing.

Quelques notions sur la génération de colonnes ont aussi été présentées, car cette méthode est utilisée avec succès pour la résolution de problèmes de grande taille, problèmes pour lesquels les méthodes classiques de programmation linéaire ne peuvent plus être utilisées. Cette méthode sera utilisée, sous une forme simplifiée, au chapitre 3.

Chapitre 3

Création de vacations pour le personnel au sol

3.1 Introduction

Le programme des vols d'une compagnie aérienne impose un ensemble de tâches à effectuer au sol : des tâches commerciales comme l'enregistrement des passagers, l'accueil à l'arrivée et au départ, le service de correspondance et de transit aux aéroports et des tâches techniques comme le guidage de l'avion sur la piste d'atterrissage, la mise en place des toboggans, les vérifications techniques, le remplissage du réservoir,… C'est le personnel au sol qui effectue ces tâches.

La création de plannings consiste à partager entre les agents toutes les tâches à effectuer durant un intervalle temporel donné, de manière à respecter des contraintes liées à la réglementation du travail, à la prise en compte des compétences du personnel et en même temps à établir un équilibre entre la charge de travail de chaque agent.

Créer directement des plannings personnalisés à partir des tâches à effectuer peut devenir un problème assez difficile à résoudre, vu le nombre des tâches à effectuer et la complexité de la réglementation. C'est pour cette raison qu'habituellement ce problème est décomposé en deux sous-problèmes indépendants (cf. § 1.3.1) : la création de vacations et l'affectation de vacations.

Dans ce chapitre nous allons présenter le problème de création de vacations. Dans la littérature, il est traité le plus souvent comme un problème de recouvrement d'ensemble. Les méthodes de résolution proposées sont basées sur la programmation linéaire ou sont des méthodes itératives qui construisent directement l'ensemble de vacations.

Comme ce problème repose sur la satisfaction de nombreuses contraintes temporelles, nous proposons une approche de modélisation basée sur le concept de plage horaire [Hennet *et al.* 03]. Ce concept permet une représentation générique des tâches et des vacations et une gestion logique des relations temporelles par l'intermédiaire d'une algèbre d'intervalles. A partir de ce modèle, nous proposons une méthode heuristique de construction de vacations basée sur le FFD et la génération de colonnes [Draghici&Hennet 05].

Si les agents travaillent en grille, la création de vacations se transforme en un problème de remplissage de vacations. L'algorithme que nous proposons pour la création de vacations peut être utilisé aussi, avec quelques modifications, pour le problème de remplissage.

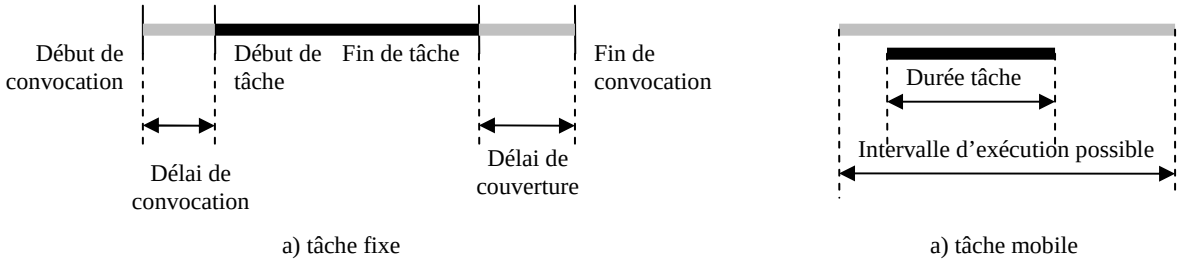
3.2 Concepts de base : tâches et vacations

Avant de décrire le modèle utilisé et la méthode de résolution que nous proposons pour la création de vacations, nous allons présenter avec plus de détails les notions de *tâche* et de

vacation, leurs caractéristiques et les différents types rencontrés. Les principaux attributs du personnel sont aussi exposés.

3.2.1 Tâches

Chaque tâche est caractérisée par un numéro identifiant, une date de début, un horaire et une qualification requise au personnel qui va l'effectuer. On appelle *délai de convocation* l'intervalle de temps qu'une personne doit passer à son poste avant le début de la tâche qu'elle doit effectuer. D'une façon équivalente, le *délai de couverture* est donné par l'intervalle de temps pendant lequel la même personne ne doit pas quitter son poste après la fin de la tâche (Figure 3.1).



a) tâche fixe a) tâche mobile

Figure 3.1 *Représentation d'une tâche*

Selon leur position dans le temps, les tâches se divisent en deux catégories :

- *tâches fixes.* Une telle tâche est, comme son nom le précise, fixée dans le temps. Donc on connaît l'heure de début de convocation, l'heure de début et l'heure de fin de la tâche proprement dite, l'heure de fin de convocation.
- *tâches mobiles*. Une tâche mobile peut être effectuée n'importe quand dans un intervalle de temps donné. Pour ce type de tâche, on connaît le début et la fin de cet intervalle de temps, ainsi que la durée de la tâche. Souvent, pour les tâches mobiles, l'heure de début de la convocation coïncide avec l'heure de début de la tâche et l'heure de fin de convocation avec l'heure de fin de la tâche.

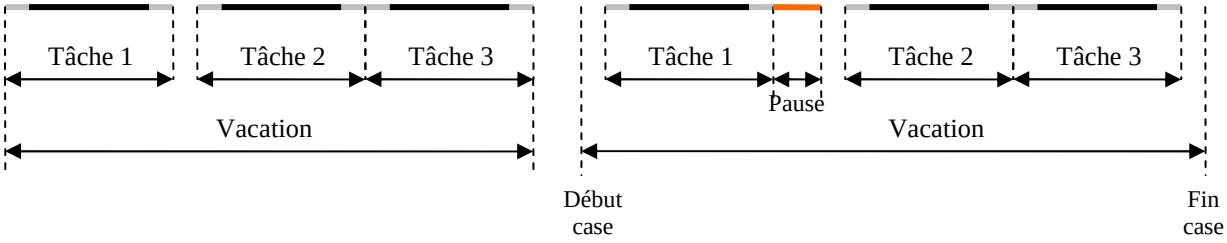
3.2.2 Vacations

Chaque vacation est caractérisée par un numéro identifiant, une date de début, une heure de début et une heure de fin, la qualification requise pour la personne qui doit la réaliser (la même pour toutes les tâches de la vacation) et la liste des tâches comprises. Dans le cas où les vacations sont générées, l'heure de début d'une vacation coïncide avec l'heure de début de convocation de la première tâche comprise dans la vacation et l'heure de fin coïncide avec l'heure de fin de convocation de la dernière tâche de la vacation. Pour le remplissage, les heures de début et de fin d'une vacation sont données par le début et la fin de la case correspondante de la grille de travail. Pour deux tâches successives d'une vacation, la fin du délai de couverture de la première tâche ne correspond pas nécessairement au début de convocation de la suivante : il est supérieur ou égal, mais pas forcément égal.

On distingue deux types de vacations :

- *vacations continues.* Ce sont des vacations où le travail est effectué en une seule fois. L'horaire mentionne l'heure de début de la vacation et l'heure de fin. La durée est déterminée par l'intervalle de temps compris entre le début de la première tâche et la fin de la dernière (Figure 3.2 a).
- *vacations fractionnées.* Ce sont des vacations coupées en deux par une pause non comptée dans le temps de travail. Une pause a les même caractéristiques qu'une tâche

mobile : un intervalle de temps pendant lequel elle doit être effectuée et une durée. (Dans le cas de la création de vacations, seule la durée de la pause est précisée). La durée d'une vacation fractionnée est égale à la durée de la même vacation si elle était continue, moins la durée de la pause (Figure 3.2 b).



a) vacation continue dans le cas '*génération de vacations*' (le début/ la fin de la vacation coïncide avec le début/ la fin de la première / dernière tâche de la vacation)

b) vacation fractionnée dans le cas '*remplissage de vacations*' (le début/ la fin de la vacation coïncide avec le début/ la fin de case correspondante de la grille de travail)

Figure 3.2 Représentation d'une vacation

3.2.3 Personnel

La force de travail est représentée par l'ensemble des salariés, appelés aussi agents, susceptibles de couvrir la charge de travail. Elle est constituée de différentes classes de personnel. Chaque classe est caractérisée par un type de contrat de travail (les mêmes types de contrat qui génèrent les patrons des vacations).

Les caractéristiques de base du personnel sont l'effectif, la disponibilité et les qualifications.

L'effectif. Même si dans une démarche stratégique, on peut supposer le personnel en nombre illimité, afin de le dimensionner, dans une approche plus opérationnelle, l'effectif de chaque classe de personnel est contraint en nombre et en disponibilité. Vu la limitation en nombre des agents, il est possible que la charge de travail ne puisse pas être couverte entièrement. Dans ce cas, des agents en sureffectif (nommés aussi agent *temporaires*) vont effectuer la charge du travail en excès. Toutefois, il faut mentionner que le nombre des agents en sureffectif ne peut pas dépasser un certain pourcentage d'augmentation de l'effectif.

La disponibilité. Pour qu'un agent puisse effectuer une tâche, il doit être disponible pour cette tâche. Il faut voir la disponibilité sous deux angles : d'un côté, la prise en compte des éventuelles activités pré-affectées et des indisponibilités (congé, arrêt maladie,…), et d'un autre côté la prise en compte des desiderata de chaque agent.

Les qualifications. Un agent peut avoir une seule qualification ou peut être multi-qualifié. Il est possible que les qualifications soient hiérarchisées, ce qui veut dire qu'un agent ayant une qualification supérieure peut effectuer le travail d'un agent avec une qualification inférieure, mais pas l'inverse. En tout cas, il faut respecter certaines limites pour ce genre d'affectation, car du point de vue de l'agent, il n'est pas toujours agréable d'effectuer un travail qui demande une qualification inférieure.

3.3 Présentation du problème

Le problème de création de vacations consiste à grouper un ensemble de tâches dans un nombre minimum de vacations, tout un respectant une série de contraintes réglementaires.

Nous avons expliqué au § 3.1 que, suivant les types des contrats du personnel, les vacations doivent être *générées* ou *remplies*. L'objectif du problème change selon le cas abordé.

Dans le premier cas, les vacations doivent être générées en nombre minimal et en respectant une certaine proportion *vacations continues/ vacations fractionnées*, proportion conforme avec l'effectif moyen disponible pour les différents contrats et qualifications.

Si les vacations sont remplies, leur nombre ne peut pas être modifié, car il correspond au nombre des cases de la grille concernée. Dans ce cas, l'objectif du problème est d'équilibrer la charge de travail entre les vacations, de manière à éviter les périodes d'activité trop longues et à ne pas faire travailler les agents sur des périodes trop courtes.

Les contraintes à respecter lors de la création de vacations changent aussi suivant le type de problème (génération ou remplissage). Nous allons présenter quelques contraintes et comment elles sont interprétées selon le cas.

3.3.1 Contraintes réglementaires

Pour que les vacations créées soient valides, de nombreuses contraintes doivent être respectées. Ce sont des contraintes liées aux caractéristiques des tâches et des vacations ou imposées par les différents types de contrats du personnel. Beaucoup de ces contraintes sont spécifiques au contexte du travail donc, pour garder la généralité, nous allons décrire les plus importantes d'entre elles.

(1) *Condition de compatibilité tâche – vacation :* seules des tâches du même jour et de même qualification peuvent faire partie de la même vacation. Comme une vacation représente un travail journalier, c'est évident que l'on ne peut pas grouper au sein d'une même vacation des tâches étalées sur plusieurs jours. On sait qu'une tâche peut être effectuée par une personne seulement si cette personne a la qualification requise. Généralement, les tâches d'un jour ne demandent pas toutes la même qualification et les agents ne sont pas forcément multi-qualifiés. Une vacation étant effectuée entièrement par une seule personne qui a une qualification précise, alors toutes les tâches comprises dans cette vacation doivent avoir la même qualification associée.

(2) *Condition d'inclusion :* toute tâche faisant partie d'une vacation doit être entièrement comprise dans la vacation. La validité de cette condition est vérifiée différemment, selon le problème. Dans le cas de la génération de vacations, l'horaire d'une vacation n'est pas connu à priori ; tout ce que l'on connaît est la durée maximale réglementaire d'une vacation. Donc, pour un ensemble de tâches à grouper dans la même vacation, la condition d'inclusion peut être formulée de la manière suivante : la durée de l'intervalle de temps compris entre le début de la convocation de la première tâche et la fin de la convocation de la dernière tâche doit être inférieure ou égale à la durée maximale d'une vacation. Si les vacations représentent les cases de la grille de travail (le cas du remplissage des vacations), le début et la fin de la vacation sont connus. La condition d'inclusion revient à : une tâche peut faire partie d'une vacation si elle commence après ou en même temps que la vacation et finit avant ou en même temps que la vacation (en l'occurrence, si la convocation de la tâche commence après ou en même temps que la vacation et finit avant ou en même temps que la vacation).

(3) *Condition de pause.* Dans le cas d'une vacation fractionnée, les tâches qui en font partie ne doivent pas écraser la pause et ni la partager en plusieurs parties. Il est possible qu'une tâche chevauche l'intervalle d'exécution possible de la pause ; cela n'est pas considéré comme non respect de la condition si ce qui reste de cet intervalle permet toujours que la pause soit effectuée. Pour plus de clarté, prenons un exemple : soit une vacation à l'intérieur de laquelle, entre 11h00 et 14h00 il faut placer une pause d'une durée de deux heure. On veut affecter à cette vacation une tâche qui

commence à 13h30 et qui finit après 14h00. Comme une pause de deux heures peut toujours être effectuée entre 11h00 et 13h30, cette tâche peut être affectée à la vacation. Par contre, une tâche qui commence à 11h30 et qui finit à 13h45 ne peut pas faire partie de la vacation, car la pause ne peut être accordée ni avant la tâche ni après.

(4) *Condition de non-chevauchement* : en cours de vacation, les tâches ne doivent jamais se chevaucher. Toutefois, dans certains cas le délai de couverture d'une tâche peut chevaucher le délai de convocation de la tâche qui la suit, la condition de non-chevauchement s'appliquant aux tâches proprement dites.

3.3.2 Méthodes de résolution dans la littérature

Les méthodes proposées dans la littérature pour la construction de vacations peuvent être divisée en deux catégories :

- les méthodes qui modélisent le problème comme un problème de recouvrement d'ensemble
- les méthodes itératives qui construisent directement l'ensemble de vacations

3.3.2.1 Modélisation par recouvrement d'ensemble

Le problème de création de vacations peut être modélisé comme un problème de recouvrement d'ensemble.

Un modèle simplifié et non-linéaire pour ce genre de problème est le suivant :

$$\text{Min } z = \sum_{j=1}^{n} c_j x_j \tag{3.1}$$

$$\text{Sous : } \sum_{j=1}^{n} a_{ij} x_j \geq 1 \qquad \forall i \in \{1,...,m\} \tag{3.2}$$

$$\sum_{i=1}^{m} a_{ij} b_i \leq d_j \qquad \forall j \in \{1,...,n\} \tag{3.3}$$

où

n : le nombre de vacations faisables
m : le nombre des tâches à grouper dans des vacations
x_j : une variable binaire qui vaut 1 si la vacation j fait partie de la solution et 0 sinon
c_j : le coût de la vacation j
a_{ij} : une variable binaire qui vaut 1 si la tâche i fait partie de la vacation j et 0 sinon
b_i : la durée de la tâche i
d_j : la durée de la vacation j

Notons que ce modèle ne suffit pas à décrire le problème de création de vacations dans le contexte aéronautique et aéroportuaire car il ne prend pas en compte les contraintes réglementaires (1), (2), (3), (4) ci-dessus. L'utilisation des approches classiques décrites ci-dessous nécessiterait donc de nouveaux tests de faisabilité des vacations candidates.

En général, le problème est résolu de la manière suivante : selon la dimension du problème, toutes ou un sous-ensemble des vacations faisables sont générées. Ayant ainsi fixé les valeurs des paramètres a_{ij} définissant les vacations candidates, une solution courante est calculée soit en résolvant le problème relâché (et linéarisé, car a_{ij} ne sont plus des variables) de recouvrement d'ensemble, soit à l'aide d'une heuristique qui sélectionne les vacations les unes après les autres. La solution finale est trouvée à partir de cette solution initiale, en utilisant différentes procédures itératives.

L'approche de Dantzig

Dantzig [Dantzig 54] propose une méthode de résolution pour un problème de création de vacations pour les agents d'un poste de péage. Toutes les vacations à créer ont une durée de 7 heures et doivent contenir une pause de 30 minutes au milieu. Comme toutes les vacations sont considérées avoir le même coût, l'objectif du problème est dans ce cas de minimiser le nombre des vacations créées. Le critère (3.1) prend donc la forme suivante :

$$\text{Min } z = \sum_{j=1}^{n} x_j \qquad (3.4)$$

Dantzig est le premier à avoir proposé l'utilisation de la solution du programme linéaire relâché dans les réels comme point de départ dans la recherche d'une solution entière. Sur les quelques tests effectués, il remarque que la solution de départ est quand même entière. Toutefois, il propose un algorithme pour arrondir la solution du problème relâché de manière à ce que toutes les contraintes soient respectées et admet que cette solution peut être considérée comme très proche de la solution optimale.

L'approche de Henderson et Berry

Henderson et Berry [Henderson & Berry, 76] proposent une méthode de résolution du problème à partir d'un sous-ensemble de vacations faisables. L'ensemble complet de vacations est défini à l'aide de plusieurs paramètres. Ensuite, un sous-ensemble est généré. Deux méthodes heuristiques sont proposées pour la génération de ce sous-ensemble :

- à chaque itération, on choisit la vacation qui couvre le plus de tâches non encore couvertes par d'autres vacations faisant déjà partie du sous-ensemble
- les vacations du sous-ensemble sont choisies au hasard

Une fois le sous-ensemble généré, il reste à trouver une solution. De nouveau, plusieurs méthodes sont proposées :

- une première méthode calcule la solution du problème relâché de recouvrement d'ensemble formulé pour le sous-ensemble trouvé avant. Ensuite, cette solution est arrondie.
- une autre méthode part de la solution obtenue par la première méthode et essaye de remplacer deux vacations par une troisième qui fait partie du sous-ensemble de départ.
- la dernière méthode proposée utilise la même technique de résolution que la deuxième, mais en partant d'une solution réalisable obtenue au hasard.

Dans cet article, Henderson et Berry ont montré que des sous-ensembles relativement limités en taille par rapport à l'ensemble complet donnent des solutions dont le coût est très proche de celui de l'optimum théorique. De plus ils ont montré qu'au-dessus d'une certaine taille, la méthode qui choisit le sous-ensemble des vacations au hasard est aussi performante que l'autre, la première restant toutefois la meilleure pour des sous-ensembles de petite taille.

En ce qui concerne les méthodes proposées pour le calcul de la solution finale, les auteurs ont constaté que les deux méthodes qui utilisent comme point de départ la solution du problème relâché donnent des résultats bien meilleurs que la méthode qui part d'une solution générée au hasard.

L'approche de Morris et Showalter

Dans leur article [Morris et Showalter 83], les auteurs proposent des approches simples pour différents problèmes de planification d'horaire.

En particulier, ils s'appuient sur une méthode de coupes planes et une procédure de séparation et évaluation.

La contribution des auteurs est à mentionner, car les tests effectués sont nombreux et les courbes de charge variées. Ils ont montré qu'environ 50% des problèmes traités donnent une solution entière dès la première résolution du problème relâché, environs 20% une fois que la borne inférieure du critère à été arrondie supérieurement et seulement 30% nécessitent la séparation et l'évaluation.

3.3.2.2 Méthodes itératives

Les heuristiques de construction itérative reposent sur le calcul d'un indice d'augmentation qui détermine la vacation devant entrer dans la solution.

[Buffa *et al.* 76] proposent l'une des premières heuristiques de construction itérative pour résoudre le problème de construction de vacations. L'approche proposée utilise un indice d'augmentation développé par Luce en 1973 qui tente d'équilibrer le sur-effectif et le sous-effectif.

[McGinnis *et al.* 78] comparent quatre heuristiques de construction itérative pour résoudre le problème de construction de vacations lorsque des vacations continues, de durée unique, doivent couvrir une charge continue sur 7 jours.

3.4 Modélisation proposée

Nous avons vu que les caractéristiques des tâches et des vacations sont liées au temps : heure de début, heure de fin, durée. De plus, les contraintes à respecter lors de la création de vacations sont, pour la plupart, des contraintes temporelles. C'est pourquoi un problème méthodologique clé qui se pose est le choix de la représentation du temps.

3.4.1 La représentation du temps

L'Intelligence Artificielle est un domaine où la modélisation du temps joue un rôle important, car les contraintes temporelles sont omniprésentes dans toute activité et influencent fortement la logique de son déroulement. La plupart de ces activités concernent le monde réel, qui est un monde dynamique. Les faits et les phénomènes qui s'y produisent se produisent dans le temps. La perception et la compréhension humaine du monde réel intègre la notion de temps. Les événements sont liés temporellement, qu'il s'agisse de leur séquencement logique (avant, pendant, après) ou de leur ordonnancement (dates de début et de fin). Le temps apparaît donc comme une entité fondamentale dans les activités humaines aussi bien que dans les phénomènes physiques.

C'est pour cette raison que nous avons porté notre attention sur des travaux concernant la modélisation du temps dans le domaine de l' Intelligence Artificielle.

Les premiers travaux qui considèrent le temps comme une entité indépendante, et non pas comme un paramètre associé à chaque activité, remontent au début des années 80. McDermott définit une théorie du temps qui repose sur des instants, un intervalle temporel étant considéré comme un ensemble d'instants [McDermott 82]. Allen a développé ensuite une logique temporelle basée sur la notion d'intervalle [Allen 84], les instants étant considérés comme des intervalles de temps très courts. Ainsi, 13 relations élémentaires d'ordre entre intervalles ont été proposées (7 directes et 6 inverses). Deux intervalles quelconques I_1 et I_2 sont nécessairement liés par l'une des relations suivantes :

- I_1 est entièrement compris dans I_2 (ou I_2 est entièrement compris dans I_1)
- I_1 et I_2 commencent en même temps, mais I_1 finit avant I_2 (ou I_1et I_2 commencent en même temps, mais I_2 finit avant I_1)

- I_1et I_2 finissent en même temps, mais I_1 commence après I_2 (ou I_1 et I_2 finissent en même temps, mais I_2 commence après I_1)
- I_1 se trouve avant I_2 et il n'y a pas de chevauchement entre les deux (ou I_2 se trouve avant I_1 et il n'y a pas de chevauchement entre les deux)
- I_1 commence avant I_2 et les intervalles se chevauchent (ou I_2 commence avant I_1 et les intervalles se chevauchent)
- I_1 se trouve avant I_2, mais il n'y a pas d'intervalle entre les deux donc, I_1 finit quant I_2 commence (ou I_2 se trouve avant I_1, mais il n'y a pas d'intervalle entre les deux donc, I_2 finit quant I_1 commence)
- I_1et I_2 représente le même intervalle.

Dans ses travaux, Shoham considère que, même si l'approche par intervalles est la plus naturelle, il serait également souhaitable de pouvoir représenter un phénomène instantané. Donc, il définit une autre sémantique du temps, dans laquelle la notion d'instant et celle d'intervalle peuvent être conjointement supportés [Shoham 87].

Le but de ce paragraphe est de donner une simple idée sur la représentation du temps dans la littérature, pour pouvoir focaliser ensuite sur le choix que nous avons fait. Pour une description plus détaillée sur le raisonnement temporel, le lecteur est envoyé à [Vila 94, Vidal 95].

3.4.2 Plages horaires

En partant des travaux présentés dans la section précédente sur la représentation du temps, nous proposons une modélisation de notre problème basée sur la notion de plage horaire [Hennet et al. 03], qui repose principalement sur la notion d'intervalle.

Les plages horaires sont des intervalles de temps dont on peut indiquer l'heure de début, l'heure de fin ainsi que la durée. On définit ici trois types de base :

– les plages continues : une plage continue est le type de plage le plus simple. En effet c'est une plage horaire définie par un début et une fin (Figure 3.3.a). Sa durée est déduite de ces deux attributs :

$$durée = fin - début \qquad (3.5)$$

– les plages fractionnées : une plage fractionnée est un ensemble de plages continues disjointes. Elle est définie par une liste de *m* plages continues (Figure 3.3.b). Nous précisons que cette liste peut être vide (*m* = 0). La durée d'une plage fractionnée est la somme des durées de chaque de ses sous plages :

$$durée = \sum_{i=1}^{m} durée_i = \sum_{i=1}^{m} (fin_i - début_i) \qquad (3.6)$$

– les plages mobiles: une plage mobile est une plage d'une durée déterminée pouvant débuter a diverses heures. Elle est caractérisée par une durée et une plage fractionnée symbolisant les heures de début possible (Figure 3.3.c). Les opérations sur les plages mobiles se ramènent donc à des opérations sur des plages fractionnées.

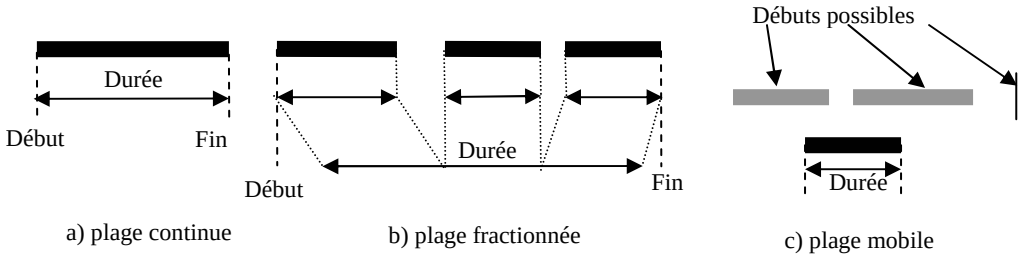


Figure 3.3 *Types de plages horaires*

Définissons maintenant quelques opérations mathématiques sur les plages horaires: addition, soustraction, intersection. On note qu'il s'agit bien de lois de composition internes dans l'ensemble des plages horaires.

L'addition des deux plages horaires. Le résultat est une plage horaire, d'habitude fractionnée, dont l'ensemble des sous plages contient toutes les sous plages des deux termes additionnés (Figure 3.4). S'il y a des plages qui se chevauchent, elles seront réunies.

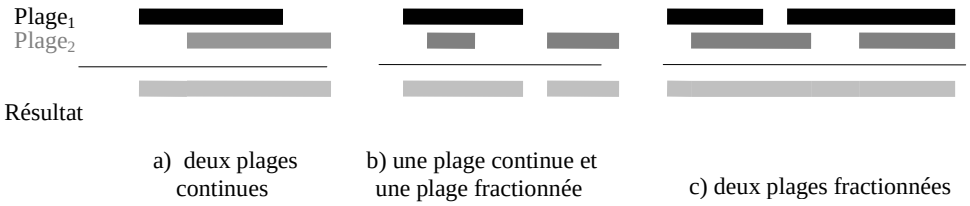


Figure 3.4 *Adition des deux plages horaires*

La soustraction des deux plages horaires. L'opération retire de l'ensemble de sous plages du premier terme la période correspondante aux sous plages du deuxième terme (Figure 3.5).

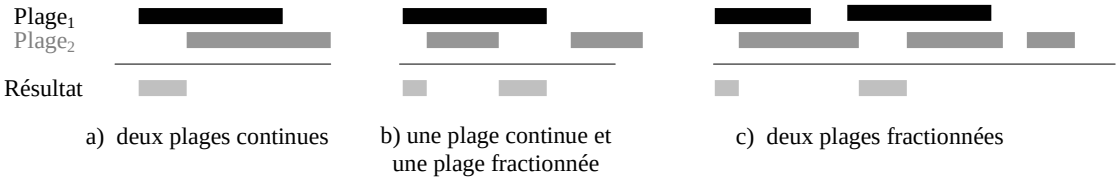


Figure 3.5 *Soustraction des deux plages horaires*

L'intersection des deux plages horaires. La plage résultante contient les sous plages (ou des parties de sous plages) communes aux deux plages horaires (Figure 3.6).

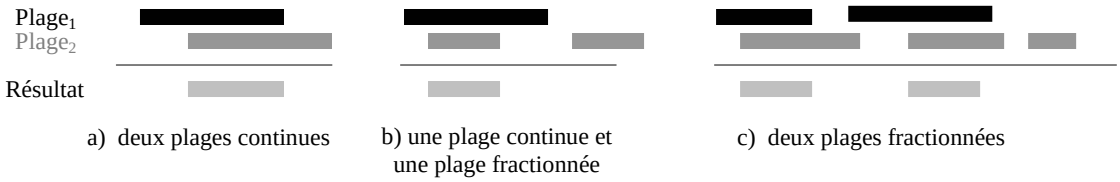


Figure 3.6 *Intersection des deux plages horaires*

3.4.3 Tâches, vacations, contraintes

A l'aide des plages horaires, les tâches, les vacations et les contraintes peuvent être modélisées d'une manière très naturelle :

Une tâche fixe peut être caractérisée par deux plages continues emboîtées : la plage de convocation et la plage de la tâche (Figure 3.7). Une tâche mobile est décrite par une plage mobile. Un ensemble de sous plages donne les dates de début possibles de la tâche.

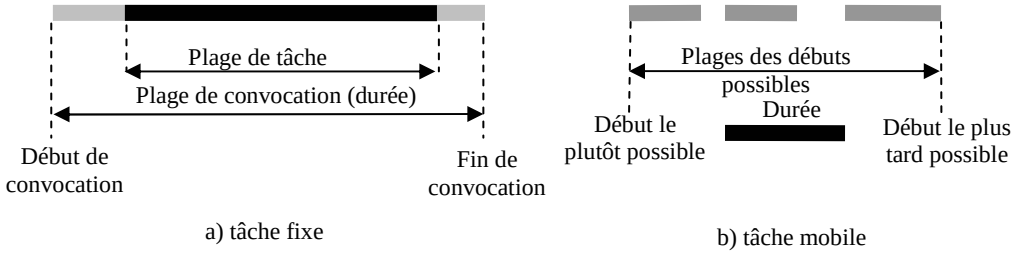


Figure 3.7 *Modélisation des tâches à l'aide des plages horaires*

Une vacation continue est définie par une plage continue qui fixe le début et la fin de la vacation et par l'ensemble de plages qui définissent les tâches de la vacation. Mais l'ensemble des plages qui définissent les tâches de la vacation forment une plage fractionnée. Pour construire une vacation fractionnée dans l'ensemble des plages horaires, il suffit de construire la vacation continue contenant toutes ses tâches et de lui ajouter la plage fractionnée des dates de début possibles de la pause, ainsi que la durée minimum de la pause. Mentionnons que la pause est ainsi représentée comme une plage mobile.

Nous allons maintenant décrire les règles à respecter lors de la création d'un planning, exprimées en termes de plages horaires (les règles (2) – (4) du paragraphe §3.3.1). L'approche par plages horaires nous permet d'exprimer ces contraintes par des relations mathématiques entre les attributs des différentes plages considérées : début, fin, durée.

Soit un ensemble de V vacations et un ensemble de T tâches. Chaque vacation v est caractérisée par une qualification g_v, une date de début z_v, une heure de début a_v et une heure de fin b_v. Si la vacation est fractionnée, on considère en plus I l'ensemble des sous plages des débuts possibles de la pause, m_v^i le début de la sous plage i, n_v^i la fin de la sous plage i et p_v la durée minimum de la pause. De la même manière, pour chaque tâche t, q_t donne la qualification de la tâche, s_t la date de début et c_t et d_t les heures de début et de fin. Si la tâche est mobile, J représente l'ensemble des sous plages des débuts possibles de la tâche, u_t^j et w_t^j le début et la fin de la sous plage j et l_t la durée de la tâche.

Condition d'inclusion. Toute tâche t faisant partie d'une vacation v doit être entièrement comprise dans la vacation. Pour une tâche fixe, les relations mathématiques sont :

$$c_t \geq a_v \qquad (3.7)$$
$$d_t \leq b_v \qquad (3.8)$$

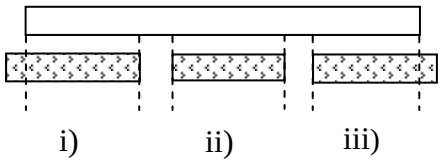
Pour une tâche mobile, la condition d'inclusion de la tâche dans la vacation est un peu différente. Pour au moins un intervalle $[u_t^j, w_t^j]$ de début possible de la tâche, il faut trouver à l'intérieur un instant deb_t tel que :

$$deb_t \geq a_v \qquad (3.7')$$
$$deb_t + l_t \leq b_v \qquad (3.8')$$

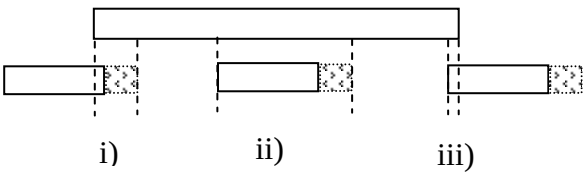
Pour traduire cette condition dans la logique d'intervalles, on peut calculer d'abord l'intersection des deux plages horaires correspondant à la tâche et à la vacation. Pour une tâche fixe, si la durée de la plage résultante est au moins égale à la durée de la tâche, la contrainte est vérifiée (Figure 3.8.a).

Pour une tâche mobile, la plage correspondante est représentée par l'ensemble des sous plages des débuts possibles de la tâche, chaque sous plage «prolongée » à droite par la durée de la tâche. Ainsi, chaque sous plage de débuts possibles devient une sous plage possible de la

tâche. Dans ce cas, l'opération d'intersection est effectuée séquentiellement, sous plage par sous plage. Si, pour une sous plage possible, la durée de la plage résultante est au moins égale à la durée de la tâche, la contrainte est vérifiée. (Figure 3.8.b).



a) vacation continue, tâche fixe
i) la tâche chevauche la vacation
ii) la tâche entre dans la vacation
iii) la tâche chevauche la vacation



b) vacation continue, tâche mobile
i) la tâche peut entrer dans la vacation
ii) la tâche peut entrer dans la vacation
iii) la tâche chevauche la vacation

Figure 3.8 *Toute tâche faisant partie d'une vacation doit être entièrement comprise dans la vacation concernée*

Condition de pause. Dans le cas d'une vacation fractionnée, les tâches qui en font partie ne doivent pas écraser la pause. Pour une tâche fixe, la condition est vérifiée si l'une des deux relations suivantes est vraie pour au moins une des sous plages des débuts possibles de la pause (un intervalle de type $[m_v^i, n_v^i]$):

$$d_t < n_v^i \qquad (3.9)$$

$$m_v^i + p_v \leq c_t \qquad (3.10)$$

Les relations ci-dessus expriment deux contraintes disjonctives, suivant la position de la tâche par rapport à l'intervalle $[m_v^i, n_v^i]$: (3.9) – la pause peut être placée après la tâche, (3.10) – la pause peut être placée avant la tâche.

Pour une tâche mobile, les relations (3.9) et (3.10) peuvent être écrites de manière suivante :

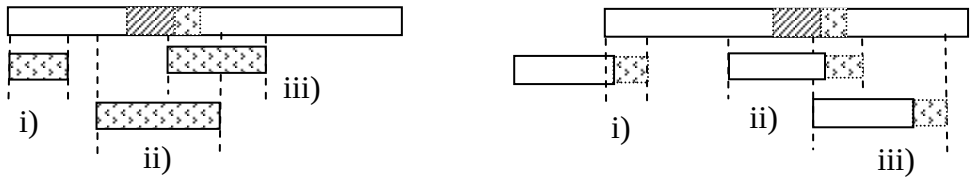
$$m_v^i + p_v < w_t^j \qquad (3.9')$$

$$u_t^j + l_t \leq n_v^i \qquad (3.10')$$

La contrainte est vérifiée si soit (3.9') soit (3.10') est vraie pour au moins un intervalle du type $[m_v^i, n_v^i]$ et au moins un intervalle du type $[u_t^j, w_t^j]$.

Analysons la compatibilité tâche – pause en termes de plages mobiles. Les plages correspondant à une tâche mobile et à la pause de la vacation sont représentées de la même manière qu'auparavant : chaque sous plage des débuts possibles est transformée en sous plage possible de la tâche ou bien de la pause.

On effectue l'opération d'addition séquentiellement, entre paires de plages du type (plage de tâche, sous plage possible de la pause) ou bien (sous plage possible de la tâche, sous plage possible de la pause). Pour une plage résultante fractionnée (deux plages disjointes), la contrainte est respectée. Pour une plage résultante continue, la contrainte est respectée si la durée de cette plage est au moins égale à la somme des durées de la tâche et de la plage. (Figure 3.9).



a) vacation fractionnée, tâche fixe
i) condition de pause respectée
ii) la tâche écrase la pause
iii) condition de pause respectée

b) vacation fractionnée, tâche mobile
i) – iii) condition de pause respectée

Figure 3.9 *En cas de vacation fractionnée, les tâches qui en font partie ne doivent pas écraser la pause*

Condition de non-chevauchement. En cours de vacation, les plages de couverture de deux tâches successives peuvent se chevaucher, sans que les tâches elles-mêmes ne se chevauchent.

Si l'on considère que pour une tâche *t*, c_t et d_t représentent le début et la fin de la plage de convocation et si l'on note $c_t^{'}$ et $d_t^{'}$ le début et la fin de la tâche proprement dite, pour deux tâches fixes *t* et *t'* (*t* avant *t'*), la relation mathématique est la suivante :

$$d_t^{'} < c_{t'}^{'} \qquad (3.11)$$

Si une tâche est mobile (ou bien les deux), on se retrouve dans le contexte de la condition de pause, où la tâche mobile prend la place de la pause et l'intervalle $[c_t, d_t]$ est substitué à $[c_t^{'}, d_t^{'}]$.

En termes de plages horaires, on effectue l'opération d'addition entre les deux plages horaires correspondant aux deux tâches. Une tâche fixe est représentée par la plage de la tâche proprement dite (et non pas par la plage de couverture, comme auparavant). Pour une tâche mobile, la plage de couverture et la plage de la tâche proprement dite sont identiques donc, la représentation d'une tâche mobile reste la même (ensemble de sous plages possibles).

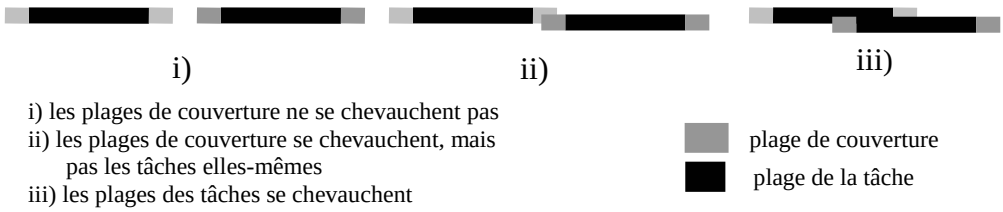


Figure 3.10. *En cours de vacation, les plages de couverture de deux tâches fixes successives peuvent se chevaucher, sans que les tâches elles-mêmes se chevauchent*

La condition de non-chevauchement est respectée si la plage résultante est fractionnée ou si la durée de la plage résultante continue est au moins égale à la durée des deux tâches. La Figure 3.10 présente le cas où les deux tâches sont fixes. Si l'une des tâches est mobile, le contexte est représenté en Figure 3.9.a si l'on substitue la pause par la tâche mobile. En gardant la même substitution, la Figure 3.9.b illustre le cas où les deux tâches sont mobiles.

3.4.4 Modélisation du problème

La création de vacations peut être modélisée comme un problème de bin-packing, où les vacations représentent les conteneurs à remplir et les tâches représentent les objets à mettre dans les conteneurs. Le modèle du problème est celui du chapitre 2, en sachant qu'il faut ajouter les contraintes additionnelles qui ont été présentées dans le paragraphe précédant. Pour résoudre notre problème, nous avons choisi d'utiliser l'algorithme FFD, combiné avec la génération de colonnes [Draghici Hennet 05].
Avant de présenter en détail la méthode de résolution choisie, quelques définitions sont nécessaires.

Rappelons que la durée d_V d'une vacation est donnée par une des deux relations suivantes :

$$d_V = fin_convoc_m - deb_convoc_1 \quad (3.12)$$

$$d_V = h_fin_{case} - h_deb_{case} \quad (3.13)$$

La relation (3.12) exprime la durée d'une vacation dans le cas de la création de vacations où fin_convoc_m représente l'heure de fin de la convocation de la dernière tâche comprise dans la vacation et deb_convoc_1 représente l'heure de début de convocation de la première tâche comprise dans la vacation. La relation (3.13) donne la durée d'une vacation pour le remplissage de vacations, *où* h_deb_{case} et h_fin_{case} expriment l'heure de début et l'heure de fin caractérisant la case correspondante à la vacation *V* dans la grille de travail.

On appelle *durée « rentable » d'une vacation* d_rent_V la somme des durées des tâches comprises dans la vacation.

On appelle *taux de remplissage d'une vacation* tR_V le rapport suivant :

$$tR_V = \frac{d_rent_V}{d_V} \quad (3.14)$$

Si la vacation est vide, le taux de remplissage est égal à zéro.

Une *vacation fractionnée* est *équilibrée* si la plage horaire de la pause est incluse dans la plage continue commençant au début de la vacation et finissant à la fin de la vacation. En bref, il faut qu'il y ait au moins une tâche de chaque coté de la pause.

Le *taux d'équilibre d'une vacation fractionnée* tE_V est calculé à l'aide de l'expression suivante :

$$tE_V = \frac{1 - d_centre_{VP}}{d_V} \quad (3.15)$$

où d_centre_{VP} représente la distance entre le centre de la vacation et le centre de la pause. Si la vacation est continue, le taux d'équilibre est égal au taux de remplissage. Un taux de 100% indique un parfait équilibre tandis qu'un taux de 0% indique un déséquilibre maximum.

Le *taux de contrainte d'une tâche fixe* tC_V dans un ensemble de tâches fixes est égal au nombre de tâches (elle exclue) qui chevauchent la plage horaire de cette tâche.

3.5 Méthode de résolution proposée

3.5.1 Génération de vacations

Dans ce paragraphe, nous allons présenter la méthode de résolution proposée pour la construction de vacations dans le cas où le profil des vacations est donné par les différents types de contrats du personnel.

Les tâches fixes et les tâches mobiles sont traitées séparément. Ce choix est justifié par le fait que, en pratique, les plannings du personnel au sol contiennent beaucoup plus de tâches fixes que de tâches mobiles. Ainsi, notre méthode s'appuie principalement sur la génération de vacations à partir de l'ensemble des tâches fixes. Ensuite, les tâches mobiles sont utilisées pour améliorer les vacations déjà créées. Eventuellement, elles conduisent à créer d'autres vacations.

L'algorithme se déroule en trois étapes :

- construction des *patrons de vacations*
- traitement des tâches fixes
- traitement des tâches mobiles

3.5.1.1 Construction des patrons de vacations

Un *patron de vacation* est un générateur de vacations d'un 'type' donné. Ce type est spécifié par les différents types de contrats du personnel. Ainsi, un patron est créé pour chaque type de contrat. (cf. §1.3.1). Chaque patron est spécifié par : une description des vacations à créer (continue/fractionnée), la durée minimum souhaitable, la durée maximum, un rapport de priorité de ce patron et un nombre de vacations déjà générées avec succès pour ce patron.

Le rapport de force est une forme de proportion. Pour deux patrons *x* et *y*, *x* ayant un rapport de 2 et *y* un rapport de 1, on pourra considérer que pour 3 vacations créées, il y aura 1 créée par *y* et 2 par *x*.

3.5.1.2 Traitement des tâches fixes

Les données d'entrée du problème sont :

La liste des tâches fixes. Cette liste contient toutes les tâches fixes que l'on souhaite placer dans des vacations. Elle est triée par ordre décroissant des taux de contrainte de chaque tâche. Cet ordre correspond à celui qui va de la tâche la plus difficile à placer à la plus facile à placer. Les tâches sont traitées ensuite séquentiellement.

La liste des vacations. Au début, cette liste est vide car aucune vacation n'a été créée. Dans le cas où la liste contient des vacations déjà construites partiellement que l'algorithme doit prendre en compte et améliorer, aucune des tâches de ces vacations ne doit être déclarée dans la liste des tâches.

La liste des patrons de vacations. Elle contient l'ensemble des patrons construits durant l'étape précédente. La liste peut être triée selon le nombre de vacations déjà générées avec succès pour chaque patron. Ce tri aide dans le choix du patron le plus pertinent afin de respecter les proportions.

FFD et génération de colonnes

Le principe général de fonctionnement suit le schéma classique du FFD. La liste des tâches est triée une seule fois au début de l'algorithme et traitée ensuite séquentiellement. La liste des vacations candidates est triée selon l'ordre dans laquelle les vacations ont été *« ouvertes »*. Une vacation est ouverte au moment où la première tâche y est insérée. La tâche courante est placée dans la première vacation de la liste qui peut la contenir.

La génération de colonnes peut être utilisée afin de réduire la dimension de la liste des vacations candidates qui doivent être testées. Une nouvelle colonne correspond à une nouvelle vacation. Si aucune vacation déjà créée ne peut contenir la tâche courante, une nouvelle vacation est créée, selon un des patrons existants. Cette nouvelle vacation peut être continue ou fractionnée, tant qu'elle peut contenir la tâche courante. Toutefois, si la nouvelle vacation créée est fractionnée, l'algorithme cherche une deuxième tâche, afin d'équilibrer la vacation dès sa construction : il place une tâche avant la pause et l'autre après la pause. S'il ne trouve pas de tâche pour équilibrer, la vacation est détruite, car les vacations fractionnées non équilibrées ne sont pas acceptées. Une vacation continue est créée à sa place.

Cette procédure simplifiée de génération de colonnes se révèle suffisante pour la résolution du problème de génération, car une vacation qui vient d'être créée est caractérisée seulement par la tâche qu'elle contient et peut être positionnée dans le temps de manière souple, compatible avec son type. Comme dans [Sarin & Aggarwal 01] le problème maître prend alors la forme d'un problème de recouvrement d'ensemble : pour un ensemble de tâches T, il faut générer un ensemble de vacations $\mathbf{V}$, de manière que :

$$\forall t \in T, \exists \text{ une seule vacation } V \in \mathbf{V} \; ; \; t \in V \qquad (3.16)$$

Ci-dessous, nous présentons l'algorithme pour le traitement des tâches fixes :

Procedure **Gen_*Traite_tâches_fixes***

```
Tant que(Liste_tâches_fixes pas vide)
     Tâche_courante=retirer_tête(Liste_tâches_fixes) ;
     Compteur itérations= Compteur itérations+1 ;
     Si(Compteur itération correspond à Fréquence de tri)
          Trier(Liste vacations) ;
     Fin Si
     Pour Vacation_Courante parcourt (Liste_vacations)
          Si(On peut ajouter Tâche_courante à Vacation_Courante)
               Ajoute(Tâche_courante à Vacation_Courante);
               Aller à Succès_Insertion
          Fin Si
     Fin Pour
     Trier par ordre de priorité (Liste_patrons_vacations);
     Pour Patron_Courant parcourt (Liste_patrons_vacation)
          Nouvelle_vacation = Génère_colonne(Patron_Courant);
          Si(on peut ajouter Tâche_courante à Nouvelle_vacation)
               Ajoute(Tâche_courante à  Nouvelle_vacation);
               Si(Nouvelle_vacation pas équilibrée)
                    Tâche_Equilibre=trouver une tâche dans Liste_tâches
                    telle que (on peut l'ajouter à Nouvelle_vacation) et
                    que( Nouvelle_vacation équilibrée) ;
                    Si(Tâche_Equilibre existe)
                         Ajouter(Tâche Equilibre à Nouvelle_vacation);
                         Compteur_itérations = Compteur itérations + 1 ;
                         Aller à Succès Patron
```

```
                    Fin Si
                    Aller à Echec_Patron
                Fin Si
                Succès_Patron :
                Ajouter(Nouvelle_vacation en fin de Liste_vacations);
                Aller à Succès insertion
            Fin Si
            Echec Patron :
            Détruire(Nouvelle vacation)
            Signaler_échec(Patron_Courant)
        Fin Pour
        Echec Insertion -> Cette tache est écartée du traitement
        Succès insertion :
Fin tant que
```

Amélioration de la solution : tri dynamique de la liste des vacations

L'objectif 'classique' de notre problème (3.16) est de minimiser le cardinal de $\mathbf{V}$, donc le nombre de vacations créées. Cependant, en pratique, celui-ci n'est pas le seul objectif à prendre en compte, car la qualité des vacations créées est essentielle pour évaluer et optimiser la qualité du travail et l'équité entre les agents. C'est pour cette raison que nous avons développé une deuxième version de l'algorithme, qui inclut un tri dynamique de la liste des vacations partiellement remplies. Ce tri permet l'amélioration de la qualité des vacations.

La liste des vacations déjà créées est ordonnée plusieurs fois lors de la construction de nouvelles vacations. Le tri est fait par ordre croissant des taux de remplissage et d'équilibre de chaque vacation. Ainsi, les vacations avec un taux faible de remplissage/ équilibre peuvent être évitées. Un paramètre de qualité détermine la fréquence de tri.

Les deux objectifs du problème sont maintenant de créer des vacations de bonne qualité et d'en créer en nombre minimum. Avec le tri des vacations, la qualité générale des vacations est améliorée, mais leur nombre peut augmenter avec la fréquence de tri. Le compromis entre la qualité des vacations et leur nombre est laissé à l'utilisateur, par le choix de la valeur du paramètre de qualité qui est lié à la fréquence du tri. Ce nouvel algorithme est du type « best fit », donc il a une complexité plus élevée que celle du FFD.

Il faut noter que le tri dynamique de la liste des vacations n'est pas lié au traitement des tâches fixes ; il est également utilisé pendant le traitement des tâches mobiles.

3.5.1.3 Traitement des tâches mobiles

Une fois le placement des tâches fixes terminé, la liste des tâches mobiles est analysée. Le données du problème sont similaires à celles considérées pour les tâches fixes : la liste des tâches mobiles, la liste des vacations, la liste des patrons de vacations. La liste de vacations n'est plus vide comme dans le cas précédant ; elle contient maintenant les vacations créées pour insérer les tâches fixes.

La procédure de traitement est similaire à celle des tâches fixes. Pour pouvoir trier les tâches mobiles de la même façon que les tâches fixes, on associe à chaque tâche l'attribut suivant : *durée de la tâche/amplitude des débuts possibles*. Les tâches sont ordonnées dans la liste en ordre croissant de cet attribut. Si deux tâches ont la même valeur, la tâche qui a la plus longue durée sera placée la première.

Règle d'insertion d'une tâche mobile dans une vacation :

Si une tâche mobile est insérée dans une vacation :

- si elle est ajoutée à une vacation contenant des tâches fixes elle sera fixée
- si elle est ajoutée à un vacation vide elle sera placée telle qu'elle
- si elle est ajoutée à une vacation ne contenant que des tâches mobiles elle sera placée en début ou en fin de la vacation.

Une tâche mobile est fixée si l'on impose son heure de début et de fin. Nous avons choisi de fixer une tâche mobile de manière à ce que son centre coïncide avec le centre de l'intervalle d'exécutions possibles.

3.5.2 Remplissage de vacations

Si le personnel travaille en grille, on ne parle plus de *génération* de vacations, mais de *remplissage (*cf. § 3.1*)*. En effet, les vacations vides sont fournies par les cases de la grille de travail.

La résolution du problème de remplissage de vacations est une adaptation de l'algorithme proposée dans § 3.5.1. Les différences par rapport à la génération de vacations sont :

- *la liste de patrons de vacations* n'existe plus
- *la liste des vacations* n'est plus vide au départ, car elle contient toutes les vacations vides à remplir avec des tâches.

Dans ce contexte, la boucle sur *la liste des patrons de vacations* devient la suivante:

```
Pour Patron_Courant parcourt (Liste_patrons_vacation)
```

décrite dans la procedure **`Gen_Traite_tâches_fixes`** s'effectue maintenant sur *la liste des vacations* et la boucle du début de la procédure sur *la liste des vacations* :

```
Pour Vacation_Courante parcourt (Liste_vacations)
```

est enlevée.

Tri de la liste des vacations

On peut noter que dans le cas du remplissage, le nombre de vacations à remplir est fixé a priori. L'objectif du problème est alors d'obtenir des vacations de bonne qualité. C'est pourquoi la liste des vacations est triée lors du processus de remplissage. N'oublions pas que la liste contient des vacations vides et des vacations partiellement remplies. Les vacations partiellement remplies sont triées facilement, suivant leur taux de remplissage/ équilibre, mais les vacations vides ont toutes ce taux égal à zéro. Elles sont placées en début de liste, dans l'ordre croissant de leur heure de début. Si plusieurs vacations commencent à la même heure, la liste est ordonnée en sélectionnant une de chaque type, afin de favoriser toutes les vacations et non pas seulement les vacations qui commencent le plus tôt. Pour plus de clarté, prenons une exemple : soit un ensemble de sept vacations V_1… V_7 dont les trois premières commencent à 9h00, la quatrième à 10h47, la cinquième et la sixième à 11h16 et la septième à 12h23. L'ordre final de ces vacations dans la liste est la suivante : V_1 (9h00), V_4 (10h47), V_5 (11h16), V_7 (12h23), V_2 (9h00), V_6 (11h16), V_3 (9h00).

3.6 Résultats obtenus

L'algorithme développé pour la génération des vacations a été implémenté en C++ et testé sur un processeur Intel Celeron (930 MHz, 248 Mo de RAM). Il a été testé sur un problème réel de taille moyenne : 450 tâches étalées sur une semaine et correspondant à deux qualifications. Les temps de réponse obtenus (de 2 à 5 secondes) ont été considérés comme

très satisfaisants. En même temps, les paramètres de qualité ont prouvé leur importance en ce qui concerne l'optimisation de l'algorithme. Les résultats sont illustrés dans le Table 1.

Pour le cas du remplissage des vacations, l'adaptation de l'algorithme de création de vacations présentée dans § 3.5.1 a été aussi implémentée et testée pour des données sur une semaine : 226 vacations et 455 tâches correspondant à 2 qualifications. Les résultats sont présentés dans la Table 3.1.

	Création des vacations		**Remplissage des vacations**		
	Temps [secondes]	Vacations crées	Temps [secondes]	Tâches affectées	Vacations restées vides
Algorithme « rapide »	2	259	5	370	9
Charge équilibrée entre vacations	4	261	71	365	3
Vacations de « qualité »	5	260	74	365	3

Table 3.1. Création et remplissage des vacations – résultats.

Dans le cas de la création des vacations, cette dégradation reste limitée: comme on peut le voir dans le Table 1, si l'on préfère un algorithme rapide, sans tenir compte de la qualité des vacations « remplies », le résultat est obtenu dans 2 secondes. Si l'on est intéressé par l'équilibrage de la charge entre vacations ou par la qualité des vacations, le temps de réponse est de 4, respectivement de 5 secondes.

Dans le cas du remplissage des vacations, un résultat sans améliorations est obtenu en quelques secondes et un résultat « de qualité » est fourni en plus d'une minute. Cela s'explique par la dimension de la liste des vacations et par le fait que l'amélioration du résultat est basée sur le réordonnancement de cette liste. Dans le cas de la génération, la liste est vide au début. Elle est remplie au fur et à mesure que les vacations sont créées. Naturellement, le traitement d'une liste vide ou de petite taille prend moins de temps que celui d'une liste de grande taille, comme dans le cas du remplissage des vacations où toutes les vacations sont données dès le début.

Si la première application groupe toutes les tâches dans des vacations, les résultats de la Table 3.1 montrent qu'environ 80 tâches n'ont pas été affectées lors du remplissage des vacations. Ce sont des tâches dont l'horaire ne correspond pas à l'horaire des grilles (donc à l'horaire des vacations). En pratique, elles pourront être effectuées par des employés qui travaillent hors grille.

3.7 Conclusions

Dans ce chapitre, nous avons présenté le problème de création de vacations. Selon les contrats des agents, on distingue deux types de problèmes. Le premier type est celui de la création de vacations par regroupement de tâches. Le deuxième est propre aux environnements où l'activité des agents est programmée selon des grilles de travail ; les vacations vides existent déjà sous la forme des cases de grilles de travail - on parle donc d'un problème de remplissage de vacations.

L'approche de modélisation que nous proposons repose sur le concept de plage horaire. Ce concept permet une représentation générique des tâches et des vacations et une gestion logique des relations temporelles par l'intermédiaire d'une algèbre d'intervalles. Le problème a été formulé comme un problème du type « bin-packing », avec des contraintes additionnelles.

Il a été résolu par une approche heuristique qui combine le FFD avec une procédure simplifiée de génération de colonnes.

Après quelques modifications, le modèle proposé a pu être aussi utilisé avec succès pour la résolution de problèmes de remplissage de vacations prédéfinies.

Chapitre 4

Création de rotations pour le personnel navigant d'une compagnie aérienne

4.1 Introduction

La construction de plannings pour le personnel navigant de compagnies aériennes est un problème difficile, d'une part à cause de sa grande taille et d'autre part à cause de la complexité de la réglementation du travail à prendre en compte. La plupart des compagnies aériennes construisent des plannings en deux étapes considérées comme indépendantes : durant la première étape, appelée *création de rotations*, un ensemble de rotations est construit à partir d'une liste de vols. Ce problème est connu sous le nom anglais de *Crew Pairing*. La deuxième étape est celle d'*affectation de rotations* (*Crew Rostering* en anglais). Elle construit des plannings pour tous les navigants, en leur affectant des rotations, des jours de réserves et de repos règlementaire et en tenant compte des activités pré-affectées. Ce chapitre présente la création de rotations, le problème d'affectation étant traité dans le chapitre 5.

La création de rotations est un problème complexe, car, d'un côté, la réglementation à laquelle les rotations sont soumises traite beaucoup de cas particuliers et, d'un autre côté, la combinatoire du problème est exponentielle. Nous avons vu dans le chapitre 1 que les travaux dédiés à la création de rotations résolvent le problème en deux étapes. Une première étape, la *construction de périodes de service*, groupe les vols dans des journées de travail (ou périodes de service). La deuxième étape enchaîne les journées de travail, de manière à créer des rotations.

Nous commençons le chapitre par l'introduction de quelques notions de base. Ensuite, nous allons décrire quelques méthodes proposées dans la littérature pour la construction de rotations. Nous résolvons le problème de construction de périodes de service par une méthode de programmation linéaire en variables mixtes. L'originalité de cette méthode est qu'elle peut prendre en compte les différents types des périodes de services. Pour le groupement des périodes de services dans des rotations, nous proposons un modèle linéaire en variables entières.

4.2 Concepts de base

Pour éviter d'éventuelles confusions, nous allons introduire quelques définitions. Même si elles sont extraites du *Code de l'Aviation Civile*, et donc propres à la réglementation française, ces définitions introduisent des notions « standard », qui sont utilisées

habituellement dans un contexte aéronautique, indépendamment du pays et de la réglementation concernée:

Temps de vol C'est le temps décompté depuis le moment où l'aéronef commence à se déplacer en vue de gagner l'aire de décollage jusqu'au moment où il s'immobilise à la fin du vol

Période de vol C'est la somme des temps de vol entre deux temps d'arrêts successifs

Temps d'arrêt C'est le temps décompté depuis le moment où l'aéronef s'immobilise à la fin de la dernière étape jusqu'au moment où l'aéronef commence à se déplacer en vue de gagner l'aire de décollage pour effectuer la première étape d'une nouvelle période de vol. (Une étape représente un vol.)

4.2.1 Rotations

Une rotation est constituée d'un ensemble de vols, débutant à la base du navigant et le ramenant à sa base en fin de rotation. La *base* d'un navigant est son aéroport de rattachement.

Les vols à effectuer pendant une rotation peuvent être des vols *en fonction* ou des vols *de mise en place*. Les *vols en fonction* sont des vols qu'un navigant effectue en tant que membre de l'équipage. Les *vols de mise en place* sont des vols que les navigants prennent en tant que passagers. Ce sont des vols qui transfèrent les navigants entre deux aéroports pour qu'ils puissent continuer leur rotation ou qui les ramènent à la base en fin de rotation. Ces vols peuvent être effectués par la compagnie même ou bien par d'autres compagnies. En général, les vols hors-compagnie ne sont choisis que dans les cas où la compagnie n'assure pas certaines liaisons entre différents aéroports.

Périodes de service. Une rotation pouvant couvrir un ou plusieurs jours, les vols qu'elle contient sont groupés dans des journées de travail, appelées *périodes de service* (« duty periods » en anglais).

Chaque période de service (*PdS*) commence avec *un délai de convocation* où *briefing* et finit avec un *débriefing*. Les durées du briefing et du débriefing ont des valeurs prédéfinies et sont spécifiques à chaque compagnie. A la fin de chaque *PdS* il y a obligatoirement une période de repos, proportionnelle au temps de vol effectué durant la *PdS*.

Dans certaines compagnies aériennes, une notion de *type* de *PdS* est introduite. Ainsi, selon l'heure de début et de fin, une *PdS* est *de jour* ou *de nuit*. Une *PdS* est considérée comme une *PdS* de nuit si elle commence ou finit dans l'intervalle $[t_{deb}, t_{fin}]$. Les valeurs de t_{deb} et t_{fin} sont spécifiques à chaque compagnie aérienne et sont exprimées en heure locale de l'aéroport de départ/ arrivée de la *PdS*. Il faut noter que :

- pour établir si une *PdS* est de nuit ou de jour, l'heure prise en compte comme heure de début de la *PdS* est l'heure de début du briefing et non pas l'heure du début du premier vol. De la même manière, l'heure de fin de la *PdS* est spécifiée par l'heure de fin du débriefing. Donc, pour une *PdS* dont le premier vol commence à $h_{deb}^{premier}$, le dernier vol finit à $h_{fin}^{dernier}$, le durée du briefing est D_{brief} et la durée du débriefing de $D_{débrief}$ la procédure pour établir le type de la *PdS* est la suivante :

Procedure ***Calcul_type_PdS***

```
type_PdS = PdS_de_jour //par défaut, toutes les PdS sont des PdS de jour
```

$h_{deb}^{PdS} = h_{deb}^{premier} - D_{brief}$
$h_{fin}^{PdS} = h_{fin}^{dernier} - D_{débrief}$

Si ($h_{deb}^{PdS} \geq t_{deb}$ *ou* $h_{deb}^{PdS} \leq t_{fin}$)
    type_PdS = **PdS_de_nuit**
Fin Si

Si($h_{fin}^{PdS} \geq t_{deb}$ *ou* $h_{fin}^{PdS} \leq t_{fin}$)
    type_PdS = **PdS_de_nuit**
Fin Si

- t_{deb} et t_{fin} étant exprimés en heure locale de chaque aéroport, l'horaire de chaque vol doit être aussi en heure locale de l'aéroport de départ/ arrivée.

Le type d'une PdS est assez important, car il a des influences sur la durée et les limites de temps de vol de la PdS. En général, une PdS de nuit est plus courte qu'une PdS de jour et demande un repos réglementaire d'une plus longue durée.

Les PdS sont soumises à des contraintes réglementaires concernant la durée maximum, la période de vol maximum, la durée du repos réglementaire. Ces règles sont établies par le gouvernement de chaque pays et donnent le cadre général d'application. Ensuite, les conventions civiles de chaque compagnie, complémentaires à la loi, peuvent alléger ou renforcer certaines contraintes.

Equipage. Chaque rotation a une configuration d'équipage qui précise le nombre de navigants de chaque qualification dont la rotation a besoin. La configuration d'un équipage est définie pour chaque vol et dépend du type et du secteur de l'avion qui effectue le vol. Dans un cas idéal, les vols d'une rotation ont tous la même configuration d'équipage, donc l'équipage est le même durant toute la rotation. En pratique, cela ne se produit pas tout le temps, car les rotations peuvent contenir des vols appartenant à plusieurs secteur avion, donc avec des configurations d'équipage différentes. Toutefois, un équipage en configuration minimale doit couvrir toute la rotation, une solution alternative (mise en place ou stand-by) étant trouvée pour les navigants en surplus sur les autres vols. Prenons un exemple simple : soir une rotation avec 3 vols : v_1, v_2 et v_3. Les vols v_1 et v_3 demandent comme équipage un pilote, un copilote et 3 hôtesses. Le vol v_2 a besoin d'un pilote, un copilote et 2 hôtesses. Durant toute la rotation, le pilote, le copilote et deux hôtesses sont en fonction. La troisième hôtesse est considérée en fonction sur v_1 et v_3 et en mise en place sur v_2. Si à la place du vol v_2 on retrouve deux vols $v_{2\text{-}1}$ et v_{2_2} de manière que l'aéroport de départ de $v_{2\text{-}1}$ coïncide avec l'aéroport d'arrivée de v_{2_2}, la solution la moins coûteuse est de laisser au sol, en stand-by, la troisième hôtesse durant ces deux vols.

Identificateur. A chaque rotation on attribue une sorte d'étiquette, appelée identificateur, qui donne des informations sur les vols contenus. Cette étiquette n'est pas unique. Si plusieurs rotations contiennent les mêmes vols, elles auront le même identificateur. Ceci arrive dans le cas des vols répétitifs. Soient deux vols répétitifs v_1 et v_2. Le vol v_1 est effectué chaque lundi durant le mois courant (v_{11} - premier lundi du mois, v_{12} - deuxième lundi du moins, …) et v_2 chaque mardi (v_{21} - premier mardi du mois, v_{22} - deuxième mardi du mois, …). Deux rotations $R_1 = \{v_{11}, v_{12}\}$ et $R_2 = \{v_{21}, v_{22}\}$ auront le même identificateur.

4.2.2 Activités hors-vol

Un navigant effectue aussi des activités au sol, appelées *activités hors-vol*. Un type important d'activité hors-vol est constitué par les *réserves*. Une réserve représente une journée de travail pendant laquelle le navigant n'effectue pas de vol, en restant toutefois disponible pour remplacer un autre navigant, si ce dernier se trouve dans l'impossibilité d'effectuer un vol qui lui a été affecté.

Les jours de réserves attribués à un navigant pendant un mois peuvent être étalés sur tout le mois ou groupés sur plusieurs jours consécutifs, dans ce que l'on appelle un *block-réserve*. La manière dont on affecte les réserves est soumise à des contraintes réglementaires et est la même pour tous les navigants. Dans tous les cas, un certain équilibrage entre les jours de réserves accordés aux navigants s'impose, car rester bloqué au sol pendant un ou plusieurs jours n'est une activité ni agréable ni valorisante.

Parmi les activités hors-vol, on peut mentionner aussi les périodes de formation et d'entraînement, les activités administratives, les visites médicales périodiques. Généralement, ces dernières activités sont programmées à des échéances fixes et ajoutées automatiquement au planning de chaque navigant. Elles ne font l'objet de ce chapitre que dans une petite mesure car, par la suite, nous allons les appeler et les considérer comme *des activités pré-affectées*.

4.2.3 Personnel

Le personnel navigant d'une compagnie aérienne est divisé en deux catégories : le personnel navigant *technique* (pilotes et copilotes) et le personnel navigant *commercial* (chef de cabine et hôtesses/stewards).

Les qualifications de chaque catégorie sont hiérarchisées : un navigant ayant une qualification supérieure est autorisé à effectuer le travail d'un navigant moins qualifié (un pilote peut effectuer un vol en tant que copilote et un chef de cabine peut voler comme hôtesse ou stewart).

Le seul fait d'être pilote ou hôtesse ne suffit pas pour pouvoir effectuer un vol, car la flotte est souvent composée de différents types (ou secteurs) d'avion et chaque secteur demande des qualifications spécifiques. Si, généralement, un navigant qualifié pour un appareil d'un secteur donné est qualifié pour tous les appareils appartenant au même secteur, il n'est pas obligatoirement qualifié pour un autre secteur. En résumé, être qualifié pour un vol signifie avoir la qualification requise par le vol à effectuer sur le secteur d'avion concerné.

Selon leur ancienneté sur différents secteurs d'avion, les navigants peuvent être *juniors* ou *seniors*. L'ancienneté est calculée en fonction du nombre d'heures de vol effectuées sur ce secteur d'avion, donc, en fonction de l'expérience ; un navigant peut être *senior* sur un secteur et *junior* sur un autre. L'ancienneté des navigants est très importante pour la constitution d'un équipage, pour que les moins expérimentés (les juniors) soient toujours accompagnés par les plus expérimentés (les seniors).

4.3 Modélisation et résolution du problème

Le problème de création de rotations consiste à construire un ensemble de rotations qui couvre toutes les vols de la compagnie sur un horizon temporel donné - en général un mois calendaire. La caractéristique de base des rotations est qu'elles doivent être *faisables* par rapport à la réglementation du travail et aux conventions civiles.

L'objectif principal du problème de création de rotations est de générer des rotations qui couvrent de la meilleure façon tous les vols de la compagnie. La qualité des rotations est jugée

le plus souvent selon le nombre des rotations créées, mais il y a aussi d'autres critères comme la durée totale des rotations, le nombre de vols de mise en place utilisées, la rétribution payée par la compagnie aux navigants qui effectuent ces rotations, …

Une compagnie aérienne peut avoir plusieurs bases, donc les vols à couvrir et les rotations à créer sont partagées entre ces bases.

4.3.1 Contraintes réglementaires

Les contraintes à respecter lors de la création de rotations sont nombreuses et spécifiques à chaque pays et chaque compagnie aérienne. Toutefois, il y a des contraintes qui se retrouvent partout, comme celles concernant les limitations sur le temps de vol et de travail ou les repos. Sans perdre de généralité, nous allons en présenter quelques-unes :

(5) *Condition sur le temps de vol :* au sein d'une PdS, le temps de vol ne doit pas dépasser une certaine limite supérieure. Toutefois, il y a des cas spéciaux où des PdS dépassent cette limite. Ces PdS ont besoin de l'autorisation du Ministère du Transport pour être effectuées et s'appellent « périodes de service *dérogatoires* ».

(6) *Condition sur la durée d'une PdS :* la durée d'une PdS ne doit pas dépasser une certaine limite supérieure.

(7) *Condition de repos :* après chaque PdS, une période de repos doit être accordée. La durée du repos dépend de la durée et du temps de vol de la PdS associée. Parfois, une nouvelle PdS peut commencer avant la fin réglementaire du repos. Dans ce cas, la durée du repos qui n'a pas été effectuée sera cumulée à la durée du repos correspondant à la nouvelle PdS.

(8) *Condition sur le temps d'escale* : le temps d'escale entre deux vols consécutifs doit être supérieur à une certaine limite inférieure. La valeur de cette limite est calculée en fonction des durées des opérations comme le débarquement et l'embarquement des passagers, le temps nécessaire au nettoyage et/ ou à l'alimentation en carburant, ... Naturellement, elle dépend du type d'avion.

4.3.2 Méthodes de résolution dans la littérature

La littérature consacrée au problème de création de rotations est très riche et la grande majorité des méthodes de résolution proposées modélisent le problème comme un problème de recouvrement d'ensemble. Une formulation non linéaire de ce problème est présentée ci-dessous :

$$\text{Min} \sum_j c_j x_j \qquad (4.1)$$

$$\text{Sous} \sum_j e_{ij} x_j \geq 1, \qquad \forall i = 1...m \qquad (4.2)$$

où :

m = nombre de vols à grouper dans des rotations

n = nombre de rotations générées

c_j = coût de la rotation j. Le coût d'une rotation peut être calculé en fonction de sa durée et du temps de vol, mais peut représenter aussi le coût payé par la compagnie pour que la rotation soit effectuée. Si l'on souhaite seulement minimiser le nombre de rotations générées, sans tenir compte des coûts, il suffit de mettre $c_j = 1$ dans (4.1).

x_j = variable binaire qui vaut 1 si la rotation j fait partie de la solution et 0 sinon

e_{ij} = variable binaire qui vaut 1 si le vol i est couvert par la rotation j et 0 sinon

Il faut noter que ce modèle prend en compte les possibles vols de mise en place au sein de la compagnie, car (4.2) permet qu'un vol se retrouve en même temps dans plusieurs rotations. Dans une rotation il sera considéré comme vol en fonction, dans les autres comme vol de mise en place. Si le problème est formulé comme un problème de partitionnement d'ensemble (donc (4.2) devient égalité), la possibilité d'intégrer des vols de mise en place est exclue.

Le problème formulé ainsi ((4.1)-(4.2)) est résolu généralement en deux étapes :
a. génération d'une solution initiale
b. amélioration de la solution

Lors de la première étape, un ensemble de rotations faisables est généré. Ayant ainsi fixé les valeurs des paramètres e_{ij} définissant les rotations candidates, une solution courante est calculée en résolvant le problème linéarisé de recouvrement d'ensemble:

$$\text{Min} \sum_{j \in J} c_j x_j \qquad (4.1')$$

$$\text{Sous} \sum_{j : i \in j} x_j \geq 1 \qquad \forall i \in I \qquad (4.2')$$

où :
J : ensemble des rotations générées
I : ensemble des vols à grouper dans des rotations
c_j : coût de la rotation *j*.
x_j : variable binaire qui vaut 1 si la rotation *j* fait partie de la solution et 0 sinon. Chaque colonne *j* a un élément 0 sur la ligne *i* si le vol *i* fait partie de la rotation *j* et 0 sinon.

Pendant la deuxième étape, de nouvelles rotations susceptibles d'améliorer la solution initiale sont construites et rajoutées au modèle linéaire (4.1')-(4.2'). Une nouvelle solution est calculée. Le processus se répète jusqu'à ce que l'amélioration de la solution devienne négligeable.

Une des méthodes les plus utilisées pour la construction de nouvelles rotations (4.1)-(4.2) est la génération de colonnes. Chaque colonne générée correspond à une rotation candidate.

Comme les contraintes réglementaires concernant les PdS sont assez complexes, les rotations ne sont pas construites directement à partir des vols ; il y a une étape intermédiaire : la construction de PdS. Les rotations sont ensuite générées à partir de ces PdS.

L'approche de Rubin

Rubin [Rubin, 73] propose un modèle de partitionnement d'ensemble pour créer des rotations. En ajoutant des limites sur le temps de vol 'disponible' par base, le modèle prend en compte l'aspect multi-base :

$$H_k^L \leq \sum_j h_{kj} x_j \leq H_k^U \qquad (4.3)$$

H_k^L et H_k^U représentent les limites inférieure et supérieure d'heures de vol par personne et par mois disponibles pour la base *k.* Ces limites sont calculées en fonction du temps de vol réglementaire qui peut être effectué par un navigant.
h_{kj} quantifie le nombre d'heures de vols par personne générées par la rotation j pour la base k. Si la rotation ne part pas de k, $h_{kj} = 0$.

Même si (4.2) exclut les vols de mise en place, il y a toutefois la possibilité d'en prendre en compte quelques uns, en rajoutant des lignes supplémentaires à la matrice (e_{ij}). Ces lignes sont prises en compte seulement dans le calcul des coûts des rotations qui les couvrent, mais ne sont pas prises en compte par le problème de partitionnement. L'avantage de cette modélisation est qu'elle intègre les deux catégories de vols de mise en place : compagnie et hors compagnie.

L'amélioration de la solution initiale S (qui est connue dès le début) est obtenue en résolvant plusieurs sous-problèmes de partitionnement d'ensemble de taille réduite : pour un ensemble de rotations $T \subset S$ qui couvre un ensemble de vols, toutes les rotations faisables qui peuvent couvrir ces vols sont générées. S'il existe un ensemble T' avec un coût meilleur que celui de T, T est remplacé par T'.

L'approche de Crainic et Rousseau

[Crainic & Rousseau, 87] proposent une méthode de construction de rotations en deux étapes. La première étape génère des PdS à partir de la liste des vols. La génération des PdS est un simple processus d'enchaînement de vols ; toutefois, l'on cherche à générer des PdS de bonne qualité, en interdisant la construction de certaines PdS. Une fois créées, les PdS ne sont plus retouchées.

La deuxième étape construit, à l'aide d'un processus itératif, des rotations à partir des PdS. :

1. Un ensemble R de rotations de bonne qualité, contenant une seule PdS est construit. Pour les PdS qui ne se retrouvent pas dans ces rotations, des rotations supplémentaires sont générées, même si elles sont de mauvaise qualité. Un compteur *s* est créé et initialisé à 2. Ce compteur donne des informations sur le nombre de PdS que les nouvelles rotations doivent contenir.
2. Le problème continu de recouvrement d'ensemble est résolu pour les rotations de R.
3. R'est généré, ensemble de rotations à maximum *s* PdS, qui pourraient améliorer la solution.
4. Si $R' \neq 0$, alors $R \leftarrow R \cup R'$ et on revient au pas 2. Sinon, on augmente la valeur de *s* d'une unité et on revient au pas 1. L'algorithme s'arrête si la valeur du compteur dépasse une certain seuil S. La valeur de S est choisie en fonction des limites que la plupart des compagnies aériennes imposent pour la durée maximale des rotations.

Une fois toutes les rotations générées, la solution entière est obtenue avec l'heuristique de Salkin [Salkin 75].

L'approche de Lavoie, Minoux et Odier

Les auteurs [Lavoie et al. 88] modélisent le problème de construction de rotations en tant que problème de recouvrement d'ensemble avec beaucoup de colonnes, chaque colonne représentant une rotation valide. On part d'une solution initiale qui est améliorée par une méthode de génération de colonnes. Le modèle construit des rotations pour une seule base.

L'algorithme construit en premier les PdS et ensuite des rotations. Le nombre de vols dans une PdS étant limité, l'énumération de toutes les PdS est possible. Une fois créées, les PdS ne sont plus modifiables.

Une solution initiale est obtenue en résolvant le problème restreint à l'aide de techniques de coupe. Cette solution fournit un ensemble de rotations de bonne qualité.

L'algorithme générateur utilisé pour trouver une nouvelle colonne de coût réduit négatif est équivalent à un problème de plus court chemin dans un graphe où les nœuds sont des PdS et les arêtes des enchaînements valides entre deux PdS consécutives. Ce problème est résolu à l'aide de l'algorithme de Dijkstra. Comme l'algorithme générateur est assez coûteux, à chaque appel tous les plus courts chemins de coût négatif sont gardés.

Généralement, l'algorithme fournit directement la solution entière, même s'il résout la relaxation continue du problème. En cas contraire, des méthodes comme Branch&Bound ou des méthodes de coupe sont utilisées.

4.3.3 Modélisation proposée

Comme nous l'avons vu dans § 4.3.1, les contraintes à prendre en compte lors de la construction de rotations, visent plus les PdS que les rotations : si les PdS d'une rotation et les repos correspondants sont légales, la rotation est légale aussi. C'est pour cette raison que nous avons décidé de garder le schéma 'traditionnel' du problème de création de rotations : génération des PdS, ensuite construction de rotations par groupement de PdS.

4.3.3.1 Construction des PdS

Même si la littérature dédiée au problème de création de rotations est très riche, à notre connaissance, peu d'attention à été accordée à l'étape de construction des PdS. Des travaux minutieux ont été effectués sur la construction de rotations à partir d'une liste de PdS, mais tous ces travaux laissent de côte les algorithmes de génération des PdS ou traitent ce problème d'une manière simpliste.

Etant donné le grand nombre de contraintes réglementaires et leur impact beaucoup plus important sur les PdS que sur les rotations, nous proposons une méthode de création de rotations qui suit le schéma classique en deux étapes – création de PdS et ensuite construction de rotations à partir des PdS – mais qui met l'accent sur la première étape et non pas sur la deuxième. L'originalité de cette méthode réside dans le fait qu'elle prend en compte l'influence que les vols ont sur les PdS, en modifiant leur type – de jour ou de nuit.

A partir de la liste des vols à couvrir sur une période de temps donnée, nous proposons de les grouper entièrement dans des PdS. On connaît le nombre des vols, le nombre maximum de PdS qui peuvent être créées (dans le pire des cas une PdS pour chaque vol) et le nombre maximum de vols qui peuvent être inclus dans une PdS. Pour chaque vol, on connaît l'heure de départ et d'arrivée, la durée du vol, les aéroports de départ et d'arrivée. On connaît aussi la durée maximum d'une PdS de jour et de nuit, le temps de vol maximum compris dans une PdS, la durée du *briefing* et du *débriefing*.

Nous proposons une formalisation du problème de construction des PdS sous forme d'un programme linéaire en variables mixtes. Nous introduisons d'abord les indices et paramètres suivants :

i : numéro de vol, $i=1..I$, où I est le nombre de vols à grouper dans des PdS
j : numéro de PdS, $j=1..J$, ou J est le nombre de *PdS* construites
k : rang d'un vol au sein d'une PdS, $k=1..K$, où K est le nombre maximum des vols dans une PdS

h_d^i : heure de départ du vol_i

h_a^i : heure d'arrivée du vol_i

d_i : durée du vol_i. Il faut noter que $d_i \neq h_a^i - h_d^i$ .Comme une PdS de nuit est déterminée par rapport à l'heure locale de l'aéroport de départ/ arrivée de son premier/ dernier vol, l'horaire de chaque vol est donné en heure locale de l'aéroport de départ/ arrivée.

a_d^i : aéroport de départ du vol_i

a_a^i : aéroport d'arrivée du vol_i

n_d^i : donnée binaire qui vaut 1 si vol_i peut déterminer une PdS de nuit en tant que premier vol de cette PdS

n_f^i : donnée binaire qui vaut 1 si vol_i peut déterminer une PdS de nuit en tant que dernier vol de cette PdS

T_{max} : durée maximum de vol dans une PdS

DJ_{max} : durée maximum d'une PdS de jour (sans compter le *briefing* et le *débriefing*)
DN_{max} : durée maximum d'une PdS de nuit (sans compter le *briefing* et le *débriefing*)
$Dif = DJ_{max} - DN_{max}$ (différence de durée entre un PdS de jour et une PdS de nuit)

D_{brief} = durée du *briefing*
$D_{débrief}$ = durée du *débriefing*

B = la base (Le problème prend en compte une seule base.)

Nous introduisons les variables suivantes :

x_{ijk} : variable binaire qui vaut 1 si le vol_i se trouve dans la PdS_j en position k ; 0 sinon.

$escale_{jk}$: variable continue qui donne la durée de l'escale comprise entre le vol en position k ($\forall k = 1...K-1$) et le vol en position k+1 de la PdS_j. Cette escale existe seulement si il y a un vol sur la position k+1.

$$escale_{jk} = (\sum_{i=1}^{I} x_{ijk+1} h_d^i - \sum_{i=1}^{I} x_{ijk} h_a^i) * \sum_{i=1}^{I} x_{ijk+1} \quad (4.4)$$

$$\Leftrightarrow escale_{jk} = \sum_{i=1}^{I} x_{ijk+1} h_d^i * \sum_{i=1}^{I} x_{ijk+1} - \sum_{i=1}^{I} x_{ijk} h_a^i * \sum_{i=1}^{I} x_{ijk+1}$$

$$\Leftrightarrow escale_{jk} = esc_{jk}^{(1)} - esc_{jk}^{(2)}$$

La forme de cette équation sort du cadre de la programmation linéaire, car c'est une soustraction de deux produits entre une variable continue (le premier facteur de chaque

produit) et une variable binaire $\sum_{i=1}^{I} x_{ijk+1}$. Selon [Bès *et al*. 02][3], chacun de ces deux produits peut être remplacé de la manière suivante :

$$esc_{jk}^{(1)} \geq 0 \quad (4.5)$$

$$esc_{jk}^{(1)} \leq MAX * \sum_{i=1}^{I} x_{ijk+1} \quad (4.6)$$

$$esc_{jk}^{(1)} \leq \sum_{i=1}^{I} x_{ijk+1} h_d^i \quad (4.7)$$

$$esc_{jk}^{(1)} \geq MAX * (\sum_{i=1}^{I} x_{ijk+1} - 1) + \sum_{i=1}^{I} x_{ijk+1} h_d^i \quad (4.8)$$

$$esc_{jk}^{(2)} \geq 0 \quad (4.5')$$

$$esc_{jk}^{(2)} \leq MAX * \sum_{i=1}^{I} x_{ijk+1} \quad (4.6')$$

$$esc_{jk}^{(2)} \leq \sum_{i=1}^{I} x_{ijk} h_a^i \quad (4.7')$$

$$esc_{jk}^{(2)} \geq MAX * (\sum_{i=1}^{I} x_{ijk+1} - 1) + \sum_{i=1}^{I} x_{ijk} h_a^i \quad (4.8')$$

$$\sum_{i=1}^{I} x_{ijk+1} \in \{0,1\} \quad (4.9)$$

$$\sum_{i=1}^{I} x_{ijk+1} h_d^i \leq MAX \quad (4.10)$$

$$\sum_{i=1}^{I} x_{ijk} h_a^i \leq MAX \quad (4.10')$$

MAX fournit la valeur maximale que h_d^i et h_a^i peuvent prendre. Comme les deux paramètres représentent des heures, MAX = 24 heures.

z_j : durée de la PS_j, sans compter le *briefing* et le *débriefing*. Elle est calculée comme somme des durées des vols contenus à laquelle on rajoute la somme des escales :

[3] Un produit d'une variable binaire *b* avec une variable continue *x* peut être remplacée par l'ajout d'une variable (continue) *p* et de 4 contraintes d'inégalité linéaires. Si $m > 0$, alors :

$$\left\{\begin{array}{ll}(x,b,p) \rightarrow & p = b * x \\ & b \in \{0,1\} \\ & x \in [0,m]\end{array}\right\} \Leftrightarrow \left\{\begin{array}{ll}(x,b,p) \rightarrow & p \geq 0 \\ & p \leq m * b \\ & p \leq x \\ & p \geq m * (b-1) + x \\ & b \in \{0,1\} \\ & x \in [0,m]\end{array}\right\}$$

$$z_j = \sum_{k=1}^{K}\sum_{i=1}^{I} x_{ijk} * d_i + \sum_{k=1}^{K-1}(esc_{jk}^{(1)} - esc_{jk}^{(2)}) \quad (4.11)$$

X_{jk} : variable continue qui vaut 1 s'il y a un vol sur la position k de la PS_j ; 0 sinon :

$$X_{jk} = \sum_{i=1}^{I} x_{ijk} \quad (4.12)$$

n_f^{jk} : variable continue qui vaut 1 s'il y a un vol en position k dans la PdS_j et si ce vol peut déterminer une PdS de nuit; 0 sinon :

$$n_f^{jk} = \sum_{i=1}^{I} n_f^i * x_{ijk} \quad (4.13)$$

n_f^j : variable continue qui vaut 1 si le dernier vol de la PdS_j transforme la PdS dans une PdS de nuit ; 0 sinon

$$n_f^j = n_f^{jK} + \sum_{k=1}^{K-1} n_f^{jk} * (1 - X_{jk+1}) \quad (4.14)$$

Encore une fois le produit $\sum_{k=1}^{K-1} n_f^{jk} * (1 - X_{jk+1})$ sort du cadre de la programmation linéaire. En utilisant la transformation de Fortet[4] [Fortet 60], il est remplacé par une variable continue p_{jk} (k = 1… *K-1*), de manière que :

$$p_{jk} = n_f^{jk} * (1 - X_{jk+1}) \quad (4.15)$$
$$p_{jk} \geq 0 \quad (4.16)$$
$$p_{jk} \leq n_f^{jk} \quad (4.17)$$
$$p_{jk} \leq 1 - X_{jk+1} \quad (4.18)$$
$$p_{jk} \geq n_f^{jk} - X_{jk+1} \quad (4.19)$$
$$n_f^{jk} \in \{0,1\} \quad (4.20)$$
$$1 - X_{jk+1} \in \{0,1\} \quad (4.21)$$

A ce moment, on peut écrire :

$$n_f^j = n_f^{jK} + \sum_{k=1}^{K-1} p_{jk} \quad (4.22)$$

[4] Un produit de deux variables binaires b_1 et b_2 peut être remplacée par l'ajout d'une variable (continue) p et de 4 contraintes d'inégalité linéaires:

$$\left\{\begin{array}{ll}(b_1, b_2, p) \rightarrow & p = b_1 * b_2 \\ & b_1 \in \{0,1\} \\ & b_2 \in \{0,1\}\end{array}\right\} \Leftrightarrow \left\{\begin{array}{ll}(b_1, b_2, p) \rightarrow & p \geq 0 \\ & p \leq b_1 \\ & p \leq b_2 \\ & p \geq b_1 + b_2 - 1 \\ & b_1, b_2 \in \{0,1\}\end{array}\right\}$$

p_j : variable qui vaut 1 si PS_j est une PdS de nuit; 0 sinon. Si PS_j commence dans l'intervalle qui définit la période de nuit, alors $\sum_{i=1}^{I} n_d^i x_{ij1} = 1$; si PS_j finit dans l'intervalle qui définit la période de nuit, alors $n_f^j = 1$. On peut déduire la valeur de p_j en fonction de $\sum_{i=1}^{I} n_d^i x_{ij1}$ et de n_f^j. Le tableau suivant montre les valeurs possibles de p_j, calculées en fonction des valeurs de $\sum_{i=1}^{I} n_d^i x_{ij1}$ et de n_f^j :

p_j	$\sum_{i=1}^{I} n_d^i x_{ij1}$	n_f^j
0	0	0
1	1	0
1	0	1
1	1	1

Tableau 4.1 *Calcul des valeurs possibles de* p_j

On peut écrire :

$$p_j = (\sum_{i=1}^{I} n_d^i x_{ij1} \text{ ou } n_f^j) \Leftrightarrow (1- p_j) = (1- \sum_{i=1}^{I} n_d^i x_{ij1})*(1- n_f^j)$$

$$\Leftrightarrow \quad p_j = \sum_{i=1}^{I} n_d^i x_{ij1} + n_f^j - n_f^j * \sum_{i=1}^{I} n_d^i x_{ij1} \tag{4.23}$$

En utilisant la même procédure que pour les produits $(\sum_{i=1}^{I} x_{ijk+1} h_d^i) * \sum_{i=1}^{I} x_{ijk+1}$ et $(\sum_{i=1}^{I} x_{ijk} h_a^i) * \sum_{i=1}^{I} x_{ijk+1}$, on introduit Pr_j (j = 1… J) variable continue, de manière que :

$$\mathrm{Pr}_j = n_f^j * \sum_{i=1}^{I} n_d^i x_{ij1} \tag{4.24}$$

$$\mathrm{Pr}_j \geq 0 \tag{4.25}$$

$$\mathrm{Pr}_j \leq \sum_{i=1}^{I} n_d^i x_{ij1} \tag{4.26}$$

$$\mathrm{Pr}_j \leq n_f^j \tag{4.27}$$

$$\mathrm{Pr}_j \geq \sum_{i=1}^{I} n_d^i x_{ij1} - 1 + n_f^j \tag{4.28}$$

$$n_f^j \in [0,1] \tag{4.29}$$

$$\sum_{i=1}^{I} n_d^i x_{ij1} \in \{0,1\} \tag{4.30}$$

Ainsi, p_j peut être écrit sous la forme suivante :

$$p_j = \sum_{i=1}^{I} n_d^i x_{ij1} + n_f^j - \mathrm{Pr}_j \tag{4.31}$$

Les contraintes du problème s'expriment par les équations suivantes :

$$\sum_{k=1}^{K}\sum_{i=1}^{I} x_{ijk} d_i \le T_{\max} \qquad \forall j = 1...J \qquad (4.32)$$

$$z_j \le DJ_{\max} - p_j * Dif \qquad \forall j = 1...J \qquad (4.33)$$

$$\sum_{j=1}^{J}\sum_{k=1}^{K} x_{ijk} = 1 \qquad \forall i = 1...I \qquad (4.34)$$

$$\sum_{i=1}^{I} x_{ijk} \le 1 \qquad \forall j = 1...J; \forall k = 1...K \qquad (4.35)$$

$$\sum_{i=1}^{I} x_{ijk} \ge \sum_{i=1}^{I} x_{ijk+1} \qquad \forall j = 1...J; \forall k = 1...K-1 \qquad (4.36)$$

$$x_{ijk} + x_{i_1 jk+1} \le 1 \text{ si } (a_d^{i_1} \ne a_a^i \text{ ou } a_a^i = B \text{ ou } h_d^{i_1} < h_a^i)$$
$$\forall i, i_1 = 1...I, i \ne i_1, \forall j = 1...J, \forall k = 1...K-1 \qquad (4.37)$$

$$x_{ijk} = 0 \text{ si } a_d^i = B \qquad \forall i = 1...I; \forall j = 1...J, \forall k = 2...K \qquad (4.38)$$

Les deux premières contraintes assurent la légalité du temps de vol (4.32) et de la durée (4.33) d'une PdS. La contrainte prend en compte le type de PdS (de nuit si $p_j = 1$ et de jour si $p_j = 0$), car la durée réglementaire varie en fonction de celle-ci.

La contrainte (4.34) impose le fait que chaque vol soit couvert une seule fois, ce qui est équivalent au fait que les vols de mise en place compagnie ne sont pas acceptés. Dans le cas où l'on souhaite intégrer dans le modèle la prise en compte des vols de mise en place, l'équation (4.34) peut être remplacée par les contraintes suivantes :

$$\sum_{j=1}^{J}\sum_{k=1}^{K} x_{ijk} \ge 1 \text{ et } \sum_{j=1}^{J}\sum_{k=1}^{K} x_{ijk} \le mep$$

où *mep* représente le nombre maximum de fois qu'un vol peut être couvert : une fois en tant que vol en fonction, *mep-1* fois en tant que vol de mise en place.

La contrainte (4.35) garantit l'existence d'au maximum un vol sur chaque position à l'intérieur d'une rotation.

La contrainte (4.36) assure le fait qu'il n'y a pas de 'trou' à l'intérieur d'une PdS. Une PdS peut contenir $k < K$ vols. Dans ce cas, on veut que les *k* vols se trouvent sur les *k* premières positions de la PdS.

Les contraintes (4.37) et (4.38) traitent l'enchaînement des vols au sein d'une PdS : un vol v_{i_1} peut suivre un vol v_i si l'aéroport de départ du v_{i_1} coïncide avec l'aéroport d'arrivée du v_i et si l'heure de départ du v_{i_1} est supérieure à l'heure d'arrivée du v_i. En même temps, les vols avec départ/ arrivée base sont interdits à l'intérieur d'une PdS (sauf en première/ dernière position).

Les solutions recherchées doivent respecter les contraintes (4.32) – (4.38) et minimiser le temps hors vol (briefing, débriefing, escales) de chaque PdS.

$$Minimiser : \quad \sum_{j=1}^{J}[X_{j1} * (D_{brief} + D_{débrief}) + \sum_{k=1}^{K-1}(esc_{jk}^{(1)} - esc_{jk}^{(2)})] \qquad (4.39)$$

Si, en même temps, l'on désire la minimisation du nombre de PdS construites, le critère peut être modifié ainsi :

$$\textit{Minimiser}: \quad \sum_{j=1}^{J}[X_{j1} * (D_{brief} + D_{débrief}) + \sum_{k=1}^{K-1}(esc_{jk}^{(1)} - esc_{jk}^{(2)})] + \sum_{j=1}^{J} X_{j1} \tag{4.40}$$

4.3.3.2 Construction de rotations à partir des PdS

A la fin de l'étape de construction de PdS, la seule caractéristique de chaque PdS créée est la liste des vols correspondants; le briefing, le débriefing et le repos réglementaire de fin de PdS ne sont pas calculés explicitement. Le briefing et le débriefing peuvent être calculés facilement, car leurs durées sont prédéfinies. La durée du repos dépend parfois de la rotation qui la contient, selon les règles de construction utilisées. Il y a deux modalités d'enchaînement de PdS : une qui permet un *pseudo* chevauchement entre deux PdS consécutives et une autre qui interdit le chevauchement. Dans le premier cas, une PdS a le droit de chevaucher la période du repos réglementaire appartenant à la PdS d'avant. Dans ce cas, son propre repos est prolongé avec la durée du repos d'avant qui n'a pas été effectuée.

Nous utilisons la méthode qui interdit les chevauchements. Dans ce cas, le repos d'une PdS est calculé une seule fois et il ne sera pas modifié. Afin de simplifier l'étape de construction de rotations, les durées correspondantes du briefing, du débriefing et du repos sont calculées avant et '*collées*' à la PdS. Ainsi, chaque PdS sera caractérisée par :

- l'heure de début, qui coïncide avec l'heure du début du briefing
- l'heure de fin, qui coïncide avec l'heure de fin du repos réglementaire
- l'aéroport de départ du premier vol
- l'aéroport d'arrivée du dernier vol

La Figure 4.1 présente l'évolution du profil d'une PdS entre la fin de l'étape de création de PdS et le début de l'étape de construction de rotations.

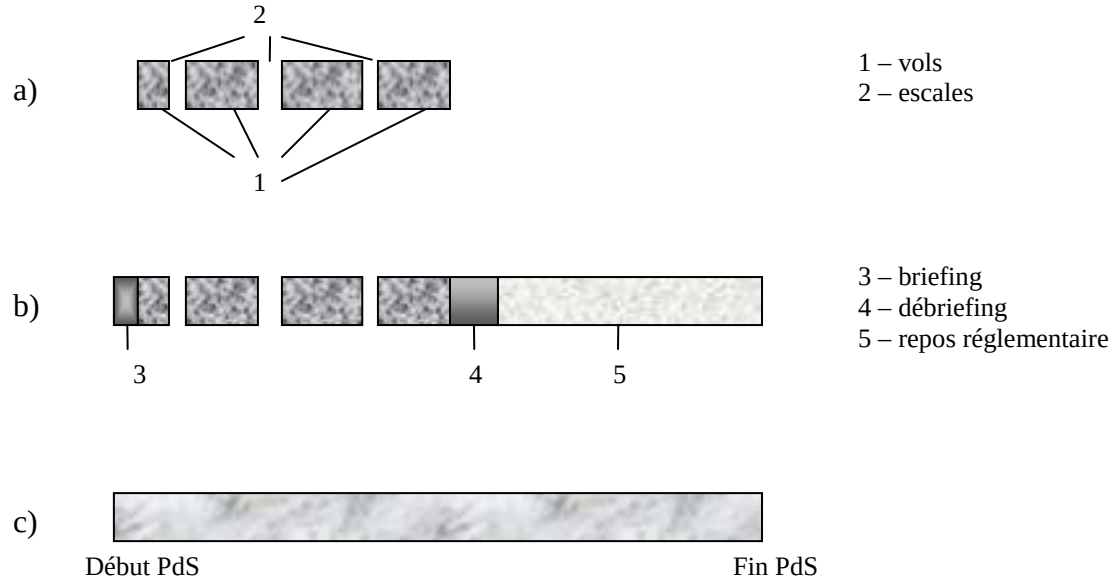


Figure 4.1 *Profil d'une PdS*
***a)** issus de l'étape de création de PdS ; **b)** après le calcul du briefing, du débriefing et du repos ; **c)** au début de l'étape de construction de rotations*

Le problème de constructions de rotations a été modélisé comme un programme linéaire en variables entières. Le modèle peut être vu comme une version simplifiée du problème de création de PdS, car :

- les règles concernant l'enchaînement des vols dans une PdS restent pour la construction de rotations. Dans une PdS, un vol peut suivre un autre seulement si son aéroport de départ coïncide avec l'aéroport d'arrivée du vol d'avant et si les deux vols ne chevauchent pas. De la même manière, deux PdS peuvent être enchaînées s'il y a une PdS qui commence après la fin de l'autre et à l'aéroport de fin de l'autre.
- Toute la réglementation concernant les limites légales du temps de vol, du temps de travail ou le type d'une PdS n'est plus prise en compte, vu que les PdS la respectent déjà.

Ainsi, la construction de rotations peut être modélisée comme un programme linéaire en nombre entiers. On introduit les paramètres suivants :

i : numéro de PdS, *i=1..I,* où *I* est le nombre des PdS à grouper dans des rotations
j : numéro de rotation, *j=1..J,* ou *J* est le nombre des rotations construites
k : rang d'une PdS au sein d'une rotation, *k=1..K,* où *K* est le nombre maximum de PdS dans une rotation. Comme une PdS peut être assimilée à une journée de travail, le nombre de jours consécutifs de travail étant limité par la loi, la valeur de *K* peut être choisie en fonction de cette limite.

On connaît :

h_d^i : heure de début de la PdS_i

h_f^i : heure de fin de la PdS_i

a_d^i : aéroport de départ du premier sol de la PdS_i

a_f^i : aéroport d'arrivée du dernier vol de la PdS_i

Les contraintes à respecter :

$$\sum_{j=1}^{J}\sum_{k=1}^{K} x_{ijk} \le 1 \qquad \forall i \in (1,...,I) \qquad (4.41)$$

$$\sum_{i=1}^{I} x_{ijk} \le 1 \qquad \forall j \in (1,...J),\ \ \forall k \in (1,...,K) \qquad (4.42)$$

$$\sum_{i=1}^{I}\sum_{k=1}^{K} x_{ijk} \le K * y_j \qquad \forall j \in (1,...J) \qquad (4.43)$$

$$x_{ijk} + x_{i_1 jk+1} \le 1 \ \text{ si } \ a_d^{i_1} \ne a_f^i \ et \ h_d^{i_1} < h_f^i \qquad \forall i, i_1 \in (1,...,I), \forall j \in (1,...J), \forall k \in (1,...K-1) \qquad (4.44)$$

$$\sum_{i=1}^{I} x_{ijk+1} \le \sum_{i=1}^{I} x_{ijk} \qquad \forall j \in (1,..,I), \forall k \in (1,..K-1) \qquad (4.45)$$

$$x_{ij1} = 0 \ \text{ si } \ a_d^i \ne B \qquad \forall i \in (1,...,I),\ \ \forall j \in (1,...J) \qquad (4.46)$$

$$x_{i_1 jk} \le \sum_{i=1}^{I} x_{ijk+1} \text{ si } a_f^{i_1} \ne B \qquad \forall i_1 \in (1,...,I), \forall j \in (1,...J), \forall k \in (2,...K-1) \qquad (4.47)$$

x_{ijk} est une variable binaire qui vaut 1 si la PdS_i est en position k dans la rotation ; 0 sinon..

y_j est une variable binaire qui vaut 1 si la rotation j a été créée et 0 sinon.

La contrainte (4.41) assure que chaque PdS soit affectée au plus une seule fois. Réciproquement, la contrainte (4.42) impose que chaque rotation ait au maximum une PdS sur chaque position. La contrainte (4.43) garantit le nombre maximal de PdS dans une rotation. La contrainte (4.44) vérifie les règles d'enchaînement de deux PdS consécutives à l'intérieur d'une rotation. La contrainte (4.45) garantit qu'une rotation avec seulement *p* PdS ($p < K$) a les PdS dans les *p* premières positions. Les contraintes (4.46) et (4.47) assurent le départ de la base au début de la rotation et le retour à la base en fin de rotation.

L'objectif est de minimiser le nombre de rotations créées et le nombre de PdS non comprises dans des rotations. La fonction objectif du modèle est la suivante :

$$\text{Minimiser } \sum_{j=1}^{J} y_j + (1 - \sum_{j=1}^{J}\sum_{k=1}^{K} x_{ijk}) \qquad (4.48)$$

4.3.4 Résultats

Nous avons validé et testé notre modèle en utilisant le logiciel de programmation linéaire Xpress-MP. Les résultats obtenus nous permettent de trouver une solution optimale pour des problèmes réduits à quelques dizaines de vols et sur un horizon de calcul d'une journée.

4.3.4.1 Construction des PdS

Jusqu'à maintenant, notre méthode n'a été testée que sur des données expérimentales et des résultats très satisfaisants ont été obtenus. Le tableau 4.1 présente quelques résultats :

Pb.	Nbr. vols	Nbr.max. PdS (J)	Vols par PdS (K)	Contraintes	Variables	Eléments non zéro	Temps sol.opt. [s]
1	7	3	3	376	84	1380	0.4
2	12	3	4	1566	174	4776	0.1
3	12	5	3	1747	215	5390	78.9
4	15	4	4	3035	280	8852	0.9
5	15	6	3	3045	312	8976	300*
6	19	5	4	6059	430	16570	6.2
7	20	5	4	6930	450	18560	0.9
8	20	8	3	7404	536	20016	555*
9	23	6	4	10583	612	27690	258.5
10	24	6	4	11898	636	30612	1.5

Tableau 4.1. *Création de FdP – résultats expérimentaux*

Chaque ligne donne des informations sur le nombre de vols à grouper, le nombre de PdS à créer, le nombre maximum de vols dans une PdS, le nombre de contraintes, de variables et d'éléments non zéro générés par le modèle, ainsi que le temps de réponse pour la solution

optimale. Dans presque tous les cas testés, la solution optimale a été trouvée. Pour les cases avec un astérisque, le programme a été interrompu avant d'obtenir la solution optimale.

Malgré l'optimalité de la solution obtenue dans presque tous les cas, le modèle présente un inconvénient majeur : le temps de réponse augmente très vite avec la taille du problème.

4.3.4.2 Construction de rotations à partir des PdS

Comme pour la création des PdS, le modèle proposé pour la construction de rotations a été testé sur des données expérimentales. Le tableau 4.2 présente quelques résultats :

Pb.	Nbr. PdS	Nbr.max. rotations	PdS par rotation	Contraintes	Variables	Eléments non zéro	Temps sol.opt. [s]
1	19	5	4	5374	404	14884	0.4
2	20	5	4	5945	425	16255	0.5
3	21	6	4	7881	531	21519	1.1
4	21	7	4	9191	616	25102	1.4
5	21	8	4	1051	701	28685	3
6	22	8	4	11510	734		1
7	23	6	4	9425	581	25445	0.7
8	23	7	4	10992	674	29682	1.1
9	24	6	4	1023	606	27366	0.6
10	24	7	4	11938	703	31086	11.6
11	27	7	4				5610*
12	30	8	4				6228*

Tableau 4.2. *Création de rotations – résultats expérimentaux*

Chaque ligne du tableau donne des information sur le nombre total de PdS, le nombre de rotations à créer, le nombre maximum de PdS dans une rotation, le nombre de contraintes, de variables et d'éléments non zéro générés pas le problème, ainsi que le temps de réponse pour la solution optimale.

Ce modèle présente les mêmes avantages et inconvénients que celui pour la construction de PdS : la solution optimale est obtenue très vite (quelques secondes) pour des jeux de données de petite taille, mais le temps de réponse augmente très vite avec la taille du problème.

4.3.5 Post traitement des rotations

Pour certaines compagnies aériennes, le programme des vols contient en grande majorité des vols répétitifs. Dans ce cas, la construction des rotations peut être faite manuellement, en exploitant au maximum la périodicité des vols. Il faut noter que la liste des rotations obtenues contient beaucoup de rotations ayant le même identificateur. Avec cette structure particulière de rotations, lors de l'étape d'affectation (qui sera présentée en détail dans le chapitre 5), des règles concernant l'équilibrage du temps de travail et du temps de vol, des jours de réserve ou de repos entre les navigants sont plus facilement respectées. Ainsi, des plannings équitables et satisfaisants sont construits.

L'outil que nous proposons essaie d'utiliser les connaissances du planificateur humain. Son expérience lui a prouvé que certaines séquences d'activités contenant des rotations, des

jours de repos ou de réserve facilitent le respect des règles d'équilibrages. Ainsi, avant l'étape d'affectation, nous proposons un post traitement des rotations. Le choix de ce post traitement est laissé à l'utilisateur. S'il désire le groupement des rotations dans des séquences, il fournit le profil des séquences qu'il souhaite construire.

L'utilisateur peut proposer un ou plusieurs type séquences. Un type de séquence est représenté par une liste qui donne des informations sur le type d'activité souhaité et l'ordre d'enchaînement. Une séquence dont le type a la forme suivante

Type_séquence_1 *: ID_1, Réserve, Repos, ID_2, ID_3, Repos*

doit contenir, dans la même ordre : une rotation ayant l'identificateur ID_1, un jour de réserve, un jour de repos, une rotation ayant l'identificateur ID_2, une rotation ayant l'identificateur ID_3 et un jour de repos. Une séquence doit être compacte : des jours sans aucune activité ne sont pas permis à l'intérieur d'une séquence. La **Figure 4.2** présente trois séquences construites selon le type ***Type_séquence_1***. Séquence_1 est une séquence valide, Séquence_2 et Séquence_3 sont des mauvaises séquences, car il y a des trous à l'intérieur. Séquence_2 laisse un jour non couvert entre la première rotation et le jour de réserve. Dans Séquence_3 il y a un jour sans aucune activité entre la deuxième et la troisième rotation.

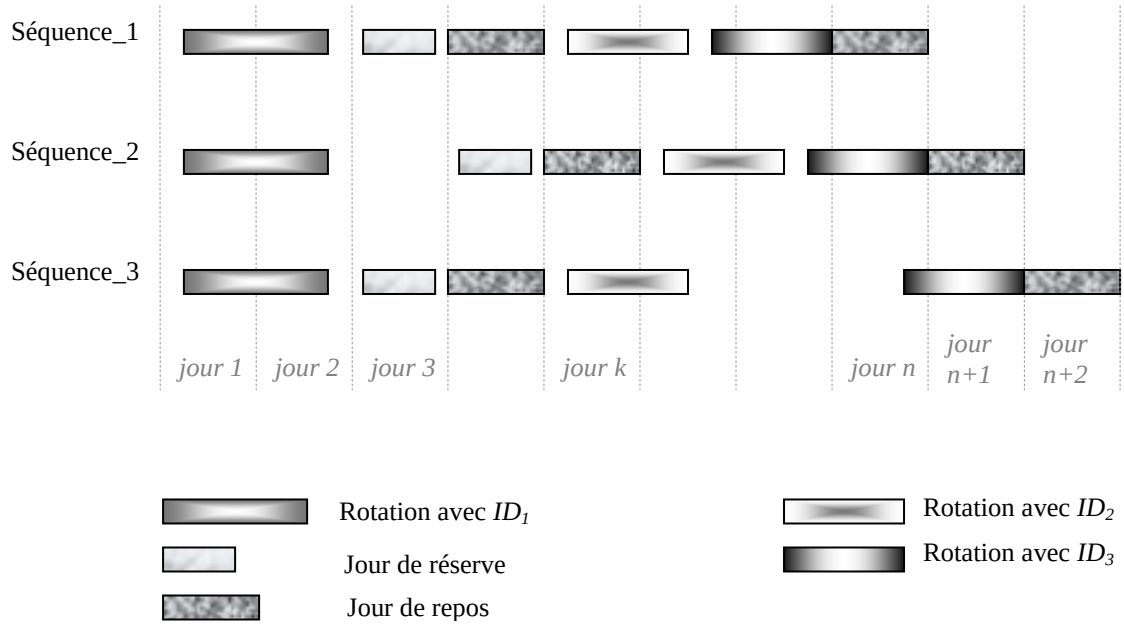


Figure 4.2 *Séquences de rotations*

Pour construire des séquences possibles, nous proposons un algorithme de listes. Afin de faciliter la recherche des rotations dans la liste, elles sont triées en ordre croissant de la date et de l'heure de début. Les types de séquences sont traités séquentiellement ; pour chaque type on cherche à construire le maximum de séquences possibles. Si les rotations se répètent chaque jour, une séquence sera construite pour chaque jour du mois.

Construction d'une séquence. Les éléments d'un type de séquence sont traités un par un et sans changer leur ordre. Si l'élément courant est un jour de réserve ou de repos, un jour de réserve ou de repos est créé et ajouté à la séquence en cours de construction. Si l'élément donne l'identificateur d'une rotation, alors la liste des rotations est parcourue pour trouver la première rotation avec cet identificateur. Dans le cas où l'élément courant n'est pas le premier

élément de la séquence, quelques conditions supplémentaires doivent être respectées. On distingue trois cas différents, selon la nature de l'élément courant :

- l'élément courant est un jour de réserve. On cherche *DateFin* et *HeureFin*, la date et l'heure de fin de la dernière activité insérée dans la séquence en train de construire. On essaie de construire une réserve le jour *DateFin*. L'horaire d'une réserve étant prédéfini, la réserve peut être construite seulement si son heure de début est supérieure à *HeureFin*. Si ce n'est pas le cas, la réserve sera construite pour le jour suivant, *DateFin+1*.
- l'élément courant est un jour de repos. Un jour de repos sera construit le jour *DateFin+1*.
- l'élément courant donne l'identificateur d'une rotation. Dans la liste des rotations, on cherche une rotation ayant l'identificateur demandé, qui commence le jour *DateFin* et après *HeureFin.* S'il n'y a pas de rotation avec ces caractéristiques, une nouvelle recherche est lancée pour le jour *DateFin+1*. Si la nouvelle recherche finit par un échec, la séquence ne peut pas être construite jusqu'à la fin. La partie qui a été construite jusqu'à ce moment est détruite. Le processus essaye de construire une nouvelle séquence du même type qui commence un autre jour.

Si la séquence a été construite avec succès, toutes les rotations comprises sont enlevées de la liste des rotations, car une rotation ne peut pas être insérée dans plusieurs séquences.

Nous présentons ci-dessous la procédure de construction de séquences :

Procedure ***Construction_séquences***

```
Tri(Liste_rotations)
Pour Type_Séquence_Courant parcourt (Liste_Types_Séquences)
   Pour(chaque jour du mois)
     Activité_Courante =Retirer_tête(Liste_activités_Type_Séquence_Courant)
          Si(Activité_Courante = réserve)
             Crée (Nouvelle_séquence)
             Nouvelle_séquence.Commence_avec(jour_de_réserve)
             enConstruction = VRAI
          Fin Si
          Si(Activité_Courante = repos)
             Crée (Nouvelle_séquence)
             Nouvelle_séquence.Commence_avec(jour_de_repos)
             enConstruction = VRAI
          Fin Si
          Si(Activité_Courante=rotation avec ID_k)
             Cherche_dans(Liste_rotations) première rotation ayant ID_k
             Si(trouvé)
                Crée (Nouvelle_séquence)
                Nouvelle_séquence.Commence_avec(rotation_trouvée)
                enConstruction = VRAI
             Fin Si
          Fin Si

          Tant_que   (Liste_activités_Type_Séquence_Courant   pas_vide   et
                          enConstruction)
             Fin_Nouvelle_séquence = Date_fin(Dernière_activité_séquence)
             Si(Activité_Courante = réserve ou repos)
                Crée(réserve ou repos) le jour (Fin_Nouvelle_séquence)
                Si (ChevauchementAvec(Dernière_activité_séquence))
                   Crée(réserve ou repos) le jour (Fin_Nouvelle_séquence+1)
                Fin Si
                Ajoute(réserve ou repos à Nouvelle_séquence)
             Fin Si
```

```
            Si(Activité_Courante=rotation avec IDl)
               Cherche_dans(Liste_rotations) première rotation avec IDl qui
                 commence le jour(Fin_Nouvelle_séquence)
               Si(pas_trouvée ou
                         ChevauchementAvec(Dernière_activité_séquence))
                  Cherche_dans(Liste_rotations) première rotation avec IDl
                    qui commence le jour(Fin_Nouvelle_séquence+1)
               Fin Si
               Si(trouvé)
                  Ajoute(rotation_trouvée à Nouvelle_séquence)
               Fin Si
               Sinon // non trouvé ou chevauchement
                  Détruit(Nouvelle_séquence)
                  enConstruction = FAUX
               Fin Sinon
            Fin Si
         Fin Tant_que

         Si(enConstruction)// Nouvelle_séquence construite avec succès
            Enlève_de(Liste_rotations) toutes les rotation comprises dans
               Nouvelle_séquence
         Fin Si
   Fin Pour
Fin Pour
```

4.4 Perspectives : construction d'algorithmes approchés

Les tests effectués sur les deux modèles proposés ont fourni de très bons résultats : la solution optimale est souvent obtenue en quelques secondes. Par contre, les deux modèles proposés fonctionnent très bien sur des jeux de données de petite taille, mais leur temps de réponse augmente de façon exponentielle avec la taille du problème. Cet aspect nous mène à explorer d'autres modalités de résolution.

4.4.1 Construction de PdS

Pour la construction de PdS, une méthode heuristique basée sur un algorithme de listes pourrait être envisagée. Cet algorithme repose sur le fait que le temps d'escale entre deux vols varie selon que l'équipage effectue le vol suivant sur le même avion ou qu'il change d'avion. Dans le premier cas, la durée du temps d'escale diminue. Comme l'objectif du problème reste de minimiser le temps hors-vol d'une PdS, cette caractéristique du temps d'escale peut être exploitée. Les vols sont groupés en listes selon l'avion qui les effectue (une liste pour tous les vols sur le même avion) et triés en ordre croissant de leur date et heure de départ; ensuite, les listes sont traitées séparément :

Pour chaque liste, on essaie de construire le maximum de PdS entièrement comprises dans la liste. Cette technique de construction réduit a priori la combinatoire du problème.

S'il reste des vols non couverts, d'autres PdS sont construites par exploration des listes des vols.

4.4.2 Constructions de rotations

Comme nous l'avons déjà mentionné, les rotations sont soumises à moins de contraintes que les PdS et, a priori, les rotations construites par enchaînement de PdS légales restent légales, donc les possibilités de combiner des PdS afin de construire des rotations sont infinies. Dans ces conditions, la première voie à explorer pour la construction de rotations reste la génération de colonnes. La combinaison d'un algorithme du type FFD et de la génération de colonne, qui s'est révélée très efficace pour la génération de vacations, est donc aussi prometteuse pour la création de rotations.

4.5 Conclusions

Dans ce chapitre, nous avons présenté le problème de création de rotations. Nous avons vu que c'est un problème complexe, habituellement résolu en deux étapes : construction de périodes de service et groupement de ces deux dernières en rotations. La littérature dans le domaine met l'accent sur la deuxième étape, la construction des périodes de service étant mise un peu dans l'ombre.

Nous proposons une méthode de résolution du problème de création de rotations, également en deux étapes, mais en transférant une grande partie de la complexité du problème sur la construction des périodes de service. Les deux sous-problèmes ont été modélisés comme programmes linéaires en variables mixtes pour la construction des périodes de service et en variables entières pour le groupement dans des rotations.

Enfin, une étape supplémentaire de post traitement des rotations permet de les grouper en séquence, avant l'étape d'affectation.

Il est important de noter que les méthodes proposées ci-dessus pour la création des PdS et des rotations traitent le problème de façon rigoureuse, en cherchant l'optimalité. En contrepartie, ces méthodes ne s'appliquent qu'à de petits jeux de données. En pratique, elles permettent d'évaluer la qualité des algorithmes de résolution approchée. Quelques pistes pour l'élaboration d'algorithmes applicables à des jeux de données réels ont été présentés en complément.

Chapitre 5

Affectation de vacations/ rotations

5.1 Introduction

Le sous-problème d'affectation représente la deuxième étape dans la résolution du problème de création de plannings. Compte tenu de la ressemblance entre l'affectation de vacations et celle de rotations, nous avons décidé de présenter les deux cas ensemble, mais en mettant en évidence les aspects spécifiques de chaque cas.

Ce chapitre commence par une présentation des deux problèmes d'affectation. Après avoir énuméré les contraintes réglementaires communes ou spécifiques à chaque cas, nous analysons quelques modèles et méthodes de résolution proposés dans la littérature.

Ensuite l'approche que nous retenons est exposée. En raison des nombreuses contraintes réglementaires le problème est modélisé comme un problème d'affectation généralisée. Nous proposons une décomposition temporelle et par qualification et ensuite une transformation du problème d'affectation généralisé en problème d'affectation simple par relaxation Lagrangienne. Un algorithme ad-hoc est utilisé pour la résolution de chaque problème de base.

IFR France a développé un produit commercial pour l'affectation de rotations au personnel navigant d'une compagnie aérienne et qui utilise notre approche de résolution. Le chapitre finit par la présentation de ce produit.

5.2 Présentation du problème

L'affectation de vacations consiste à affecter aux agents du personnel au sol (PS), à moindre coût, toutes les vacations à effectuer, de manière à respecter les contraintes liées à la réglementation du travail et aux disponibilités des agents, en prenant en compte leurs préférences.

L'affectation de rotations consiste à affecter au personnel navigant (PN) toutes les rotations à effectuer sur un intervalle temporel donné, de manière à respecter les contraintes liées à la réglementation du travail et aux disponibilités des agents, en prenant en compte leurs préférences. De manière équivalente, il faut trouver des équipages complets pour chaque rotation.

5.2.1 Contraintes réglementaires

Le processus d'affectation doit tenir compte de plusieurs contraintes réglementaires. Ces contraintes représentent l'ensemble des lois et règlements consignés dans le Code du travail

complété par la jurisprudence, les conventions collectives complétant les dispositions légales en matière d'heures supplémentaires et de modulation des horaires. Il faut mentionner que pour un navigant, une journée de travail est représentée par une PdS et le temps de travail est équivalent au temps de vol effectué.

Dans un cadre très général, trois niveaux de contraintes peuvent être distingués :

(1) *Contraintes quotidiennes* : durée minimum et maximum d'une journée de travail, durée maximum du temps de travail. Pour les PS : nombre de pauses avec leurs durées et la fenêtre de temps au cours de laquelle chaque pause peut être prise.
(2) *Contraintes sur n jours glissants* : temps de travail minimum et maximum, temps de repos nécessaire entre deux jours de travail successifs, nombre maximum de jours de travail successifs, temps de repos minimum, positionnement des jours de repos dans la semaine (le week-end ou dans la semaine)
(3) *Contraintes sur une période de plusieurs semaines* : durée exacte du temps de travail effectif moyen sur l'horizon considéré, nombre maximum d'heures supplémentaires permises.

Pour les navigants, il y a aussi des contraintes concernant la qualification sur des vols et la configuration des équipages :

(4) *Contraintes sur la qualification pour un vol*: un navigant peut effectuer un vol s'il a effectué un nombre minimum requis d'heures de vol sur le secteur d'avion correspondant à ce vol dans un intervalle temporel qui précède le vol (*exemple* : un navigant peut effectuer un vol sur un avion de type *A* s'il a effectué au moins *x* heures de vol sur le même type d'avion, lors des *y* derniers mois avant le vol).
(5) *Contraintes entre les membres d'un équipage* : dans le cockpit, si l'un des navigants (pilote ou copilote) est junior, l'autre doit être senior. Dans la cabine, vers certaines destinations, un nombre minimum d'hôtesses doivent parler certaines langues.

Suivant leur importance, les règles à respecter lors de l'affectation peuvent être divisées en deux catégories : règles dures et règles souples. *Les règles dures* doivent être respectées impérativement, car ce sont des règles qui fixent le cadre de travail et la validité des affectations (les règles imposées par la loi). *Les règles souples* contribuent à la qualité de l'affectation (règles économiques, de qualité ou de préférence). Elles peuvent être progressivement relaxées si nécessaire, afin d'accroître le domaine de solutions admissibles.

5.3 Méthodes de résolution dans la littérature

5.3.1 Affectation de vacations

Malgré l'intérêt accordé au problème de création de vacations, peu de travaux ont été dédiés exclusivement au problème d'affectation de ces dernières aux agents. Souvent, cette deuxième étape de la création de plannings est traitée à la suite de la construction des vacations. Pour des raisons d'intégrité, nous avons décidé de présenter entièrement les études qui traitent ensemble la création et l'affectation de vacations.

L'approche d'Alvarez-Valdes

Alvarez-Valdes et ses co-auteurs [Alvarez-Valdes *et al.* 99] proposent une méthode d'affectation de personnel sur un horizon temporel d'une semaine. Le problème est résolu en trois étapes. Dans une première étape, on prend en compte seulement les types des vacations (de nuit, de matin et d'après-midi) et le nombre de jours de repos à accorder par semaine. Des enchaînements admissibles de vacations et jours de repos sur 7 jours sont définis. Le but de cette première étape est de trouver l'ensemble minimal de ces enchaînements qui couvre toutes les vacations de la semaine. Le modèle du problème est le suivant :

Min $\sum_{s \in S} c_s x_s$

Sous : $\sum_{s \in S} a_{ts} x_s \geq r_t, \qquad \forall i \in T$

$x_s \geq b_s$, x_s entier

x_s compatible avec la semaine précédente

où :
S = ensemble des enchaînements admissibles
T = périodes de la semaine
c_s = coût de l'enchaînement *s*
x_s = nombre de fois où l'enchaînement s est pris dans la solution
r_t = nombre de personnels demandés pour la période t
a_{ts} = variable binaire qui vaut 1 si l'enchaînement s couvre la période t et 0 sinon
b_s = borne inférieure pour x_s

Le problème est résolu par une recherche tabou, basée sur la procédure de [Glover & McMillan 86] pour des problèmes de sac à dos multidimensionnel. La méthode suit une oscillation stratégique qui alterne des phases constructives et destructives. Dans une phase constructive, la valeur de x_s est augmentée progressivement et dans une phase destructive elle est diminuée progressivement.

La deuxième étape est un simple problème d'affectation de *n* enchaînements à *n* personnels. Le coût d'affectation est calculé en fonction des objectifs suivants : équilibrage du nombre de week-ends non-travaillés et du nombre d'heures de nuit travaillées, pénalisation pour des séquences trop longues de jours consécutifs travaillés. Le problème est résolu à l'aide de l'algorithme hongrois.

La dernière étape affecte les vacations proprement dites aux personnels, en fonction des enchaînements qui leur ont été attribués lors de la deuxième étape. Cela se fait en deux parties. La première partie affecte les vacations de manière à respecter toutes les contraintes. Chaque personnel reçoit chaque jour la vacation la plus longue admissible ayant le type défini par l'enchaînement correspondant. Deux procédures de correction sont possibles durant cette partie : le remplacement de certaines vacations déjà affectées aux agents et l'échange d'enchaînements entre deux agents.

La deuxième partie cherche à améliorer la solution par rapport à certaines contraintes secondaires, liées au nombre requis d'agents et aux limites de temps de travail. Cela peut se faire en remplaçant certaines vacations par d'autres plus courtes ou en accordant plus de jours de repos.

Cette approche fournit une solution admissible, pas forcément optimale, mais qui donne de bons résultats en pratique.

L'approche de Mason

Dans [Mason *et al.*, 98], on nous propose une méthode d'affectation de vacations pour le personnel du service de douane de l'Aéroport International d'Auckland, Nouvelle Zélande. Les vacations doivent être affectées à des programmes de travail journaliers, intégrés dans des séquences de 9 jours consécutifs (6 jours de travail et 3 jours de repos). Comme il n'y a pas de vacations à cheval sur deux jours, le problème est décomposé d'une façon naturelle en sous-problèmes indépendants qui couvrent un jour, sachant que chaque jour est divisé en périodes de temps de 15 minutes.

Chaque sous-problème a pour objectif de minimiser le coût associé aux agents nécessaires pour effectuer toutes les vacations du jour. Le modèle du problème est le suivant :

$$\text{Min} \sum_{j=1}^{n} c_j x_j$$

$$\text{Sous :} \sum_{j=1}^{m} x_j a_{ij} \geq b_i, \qquad \forall i \in \{1,...,m\}$$

$$x_j \geq 0, \quad x_j \text{ entier}, \qquad \forall j \in \{1,...,n\}$$

où :

a_{ij} = variable binaire qui vaut 1 si la période i est couverte par la vacation j, 0 sinon

x_j = nombre d'agents qui doivent effectuer la vacation j

b_i = nombre minimum de personnel requis pour effectuer la tâche j

c_j = coût de la vacation j, calculé en fonction de la durée de la vacation (nombre de périodes de 15 minutes couvertes)

Chaque problème de ce type est résolu par programmation linéaire en nombres entiers [Ryan 80]. La solution entière est obtenue par un schéma standard de branchement sur les variables.

Les problèmes sont en général de petite taille, donc de très bons temps de réponse sont obtenus (quelques dizaines de secondes).

Ensuite, des plannings individuels couvrant des enchaînements de 9 jours sont construits à l'aide d'une heuristique [Panton 91] qui détermine les jours travaillés et les jours de repos. Cette heuristique génère des plannings faisables de bonne qualité, en assurant en même temps la qualité des plannings plus longs, obtenus par la succession de plusieurs enchaînements consécutifs. Une fois l'enchaînement créé, les vacations sont affectées en ordre décroissant de leur taille.

L'approche de Duffuaa et Al-Sultan

[Duffuaa & Al-Sultan, 99] proposent un modèle stochastique pour l'affectation des vacations au personnel de maintenance des avions dans un aéroport. Les travaux de maintenance des avions se divisent en deux catégories : la première, appelée « maintenance planifiée » inclut la maintenance de routine et préventive ; la deuxième, la « maintenance non-planifiée », inclut les pannes et les urgences.

Si la maintenance planifiée a un comportement déterministe, la maintenance non-planifiée a un caractère stochastique et peut perturber complètement le planning des tâches/ vacations planifiées.

Le modèle proposé contient une partie déterministe et une partie stochastique. La partie déterministe reprend le modèle décrit dans [Roberts & Escudero, 83] qui est un programme linéaire en nombre entières. La partie stochastique combine une approche anticipative et une

approche adaptive, de manière à trouver un compromis entre un planning à long terme et un planning ayant une bonne réactivité face aux urgences de chaque jour.

5.3.2 Affectation de rotations

L'approche de Ryan

Un des articles qui a beaucoup influencé la suite des travaux dans le domaine est celui de Ryan [Ryan 92]. Il est l'un des premiers à avoir modélisé le problème d'affectation de rotations comme un problème de partitionnement d'ensemble. L'idée de base est de générer tous les plannings faisables de chaque navigant. Chaque planning doit respecter la réglementation et les activités pré-affectées. Pour obtenir une solution faisable, un sous-ensemble de ces plannings doit être sélectionné, de manière à affecter un seul planning à chaque navigant et à assurer des équipages complets pour chaque rotation. Pour un problème avec *p* navigants et *t* rotations, le modèle général est le suivant:

$$\begin{aligned} &\text{Min } z = c^T x \\ &\text{Sous : } Ax = b \\ &x_i \in \{0,1\} \end{aligned}$$

avec A une matrice unimodulaire qui a la forme suivante :

$$A = \begin{bmatrix} C_1 & C_2 & C_3 & \dots & C_p \\ L_1 & L_2 & L_3 & \dots & L_p \end{bmatrix}$$

où :

$C_i = e_i e^T$ est une matrice de dimension $p \times n_i$ avec $e_i = [0,0,...1,...,0]$ (1 en position *i*) et $e^T = [1,1,...,1]$.

L_i : matrice ce dimension $t \times n_i$ ou n_i représente le nombre de plannings faisables pour le navigant *i*. Chaque élément l_{jk} vaut 1 si le *k*-ième planning du navigant contient la rotation *j* et 0 sinon.

La matrice A est de dimension $m \times \sum_{i=1}^{p} n_i$, où $m = p + t$.

Les éléments du vecteur *b* ont les valeurs suivantes : b_i = 1 pour $i = 1...p$ et $b_{p+i} = r_i$ pour $i = 1...t$, où r_i représente le nombre de navigants requis pour la i-ième rotation.

Ryan propose une construction par simple énumération des plannings faisables pour chaque navigant. Comme le nombre de plannings pour un seul agent peut atteindre des milliers, des techniques de filtration [Ryan & Falkner, 88] sont utilisées pour limiter leur nombre. Ensuite, le problème est résolu comme un programme linéaire en nombre entiers. La relaxation linéaire obtenue à l'aide de l'algorithme primal du simplexe est améliorée en utilisant une version des stratégies de programmation linéaire élastique utilisées par [Brown & Graves, 75]. La solution entière est obtenue à l'aide d'une technique de *Branch&Bound* qui effectue le branchement sur les contraintes et non pas sur les variables. L'arbre de recherche est exploré en profondeur, en parcourant toujours d'abord la branche dont la valeur de la contrainte est fixée à 1.

L'approche de Dawid

En utilisant le model de Ryan, Dawid [Dawid *et al.* 01] propose une méthode de résolution à l'aide d'une technique de *Branch&Bound*. Cette approche cherche une solution faisable, mais pas forcément optimale. L'avantage de cette méthode est qu'elle arrive à réduire le temps de réponse par rapport à d'autres méthodes proposées dans la littérature pour le même genre de problème.

Le modèle peut être décrit de la manière suivante : soit W un ensemble de plannings individuels générés a priori, M un ensemble de navigants et P un ensemble de rotations, en sachant que chaque rotation *j* a besoin de d_j navigants.

Un planning L est constitué d'un sous-ensemble de rotations qui ne chevauchent pas $j \in P$, assujetties aux contraintes réglementaires. $W_k^M \subseteq W$ représente l'ensemble de plannings faisables associés au navigant *k*. Pour tout $k_1 \neq k_2$, les ensembles $W_{k_1}^M$ et $W_{k_2}^M$ sont disjoints. $W_j^P = \{L | j \in L\}$ représente l'ensemble de tous les plannings *L* contenant la rotation *j*.

Une variable binaire x_i est associée à chaque planning *W*. Si le planning est sélectionné dans la solution, x_i prend la valeur 1, sinon $x_i = 0$. Chaque planning a une utilité u_i. Cette valeur est utilisée pour maximiser l'utilité totale de la solution.

A l'aide de ces notions, Dawid modélise le problème de la manière suivante :

$$\text{Min} \sum_{i=1}^{n} u_i x_i$$

$$\text{Sous :} \sum_{i \in W_k^M} x_i = 1, \qquad \forall k \in \{1,...,m\}$$

$$\sum_{i \in W_j^P} x_i = d_j, \qquad \forall j \in \{1,...,q\}$$

$$x_i \in \{0,1\}, \qquad \forall i \in \{1,...,n\}$$

Par rapport au modèle de Ryan, Dawid intègre dans son modèle la sur-qualification, ce qui veut dire qu'un navigant ayant une qualification supérieure peut effectuer un travail qui demande une qualification inférieure (par exemple, un chef de cabine peut effectuer une rotation en tant que stewart). Cette approche permet la résolution des problèmes ou le nombre de PN ayant une certaine qualification n'est pas suffisant pour couvrir toutes les rotations.

L'algorithme de résolution proposé, qui est une adaptation de l'algorithme de Davis-Putman [Davis & Putman, 96], est une énumération implicite. Même si la recherche d'une solution se fait dans un arbre, cette méthode n'est pas un *Branch&Bound* traditionnel, car il n'y a pas de bornes pour limiter l'espace de recherche d'une solution : c'est une technique de propagation de contraintes qui est utilisée pour réduire le nombre de noeuds à évaluer.

Dans chaque noeud de l'arbre l'état du modèle est défini : $S = \{S_{pos}, S_{neg}, S_{free}\}$ où S_{pos} définit l'ensemble des variables x_i dont la valeur a été fixée à 1, S_{neg} l'ensemble des variables x_i dont la valeur a été fixée à 0 et S_{free} l'ensemble des variables x_i dont la valeur n'a pas encore été fixée. L'état initial du modèle est $S^0 = \{0,0,W\}$. Une solution faisable est obtenue si $S = \{*,*,0\}$ et si toutes les contraintes sont respectées.

Dans chaque noeud de l'arbre de recherche, la variable de branchement est choisie parmi les éléments de S_{free}. Si S_{free} est vide et le modèle est toujours dans un état faisable, alors une solution a été trouvée et la recherche s'arrête. Sinon, l'état courant du modèle est sauvegardé, la valeur de la variable de branchement choisie est fixée à 1 et une propagation de contraintes est effectuée. Si le modèle reste toujours faisable on descend dans l'arbre de recherche et un autre nœud est examiné. Si le modèle devient infaisable après un changement d'état ou suite à un backtracking, l'ancien état doit être restauré et la valeur de la dernière variable choisie est fixée à 0. Si la nouvelle solution est toujours infaisable le nœud courant est enlevé de l'arbre de recherche et un backtracking est effectué. Sinon, la procédure se répète pour une autre variable libre.

L'approche de Gamache

[Gamache *et al.* 99] utilisent la même modélisation que Ryan, mais ils proposent une méthode de résolution basée sur la génération de colonnes.

Une *rotation mère* est une rotation qui doit être affectée à un équipage entier. Chaque rotation mère peut être décomposée en plusieurs *rotations filles*, une rotation fille pour chaque membre de l'équipage. L'objectif du problème est de maximiser la durée des rotations couvertes, donc de minimiser la durée des rotations qui n'ont pas été couvertes.

Le modèle mathématique du problème maître est le suivant :

$$\text{Min} \sum_{p=1}^{n} c_p e_p$$

$$\text{Sous :} \sum_{k=1}^{m} \sum_{r \in R_k} a_{pr} x_r + e_p = b_p, \qquad \forall p \in \{1,...,n\}$$

$$\sum_{r \in R_k} x_r = 1, \qquad \forall k \in \{1,...,m\}$$

$$e_p \geq 0 \qquad \forall p \in \{1,...,n\}$$

$$x_r \in \{0,1\} \qquad r \in \bigcup_{k=1}^{m} R_k$$

ou

R_k : ensemble de plannings faisables pour le navigant *k*

c_p : durée de la rotation *p*

b_p : nombre des rotations filles composant la rotation mère *p*

e_p : nombre de rotation filles appartenant à la rotation mère p et non affectées

a_{pr} : variable binaire qui vaut 1 si la rotation p fait partie du planning r et 0 sinon

x_r : variable binaire qui vaut 1 si le planning r fait partie de la solution et 0 sinon

La relaxation linéaire du problème maître est résolue à l'aide du simplexe pour un ensemble de plannings donnés. Le problème esclave utilise une technique de génération de colonne pour construire de nouveaux plannings de coût réduit négatif.

Dans le problème esclave, un réseau est associé à chaque navigant. Le nœud source et le nœud puits modélisent le début et la fin du mois. Tous les autres noeuds représentent des instants de temps associés au début ou à la fin d'une activité. Chaque arc représente soit une activité (rotation, repos, activité pre-affectée) soit un lien entre deux instants de temps. Un

arc représentant une rotation fait partie du réseau associé à un navigant seulement si le navigant peut l'effectuer.

Chaque chemin trouvé dans un réseau représente un planning valide pour le navigant en question. Le coût associé a chaque chemin représente le coût réduit du planning dans le problème maître et il est calculé à l'aide des variables duales fournies par celui-ci. A chaque résolution d'un problème, tous les plannings de coût réduit négatif sont envoyés vers le problème maître. Cette technique est décrite en détail dans [Desrosiers et al. 95].

La solution entière est obtenue à l'aide d'une recherche partielle de type *Branch&Bound*. L'heuristique utilise une stratégie de recherche en profondeur et s'arrête à la première solution entière trouvée.

5.4 Modélisation du problème

En raison des nombreuses contraintes réglementaires, le problème d'affectation de vacations/ rotations est un problème d'affectation multiple généralisé. Ce problème admet une formulation linéaire en variables mixtes : des variables binaires pour représenter les choix d'affectation et des variables réelles pour représenter les dates et les durées dans les contraintes de réglementation. Il est donc naturel de chercher à le résoudre par les logiciels commerciaux de programmation linéaire en variables mixtes, comme CPLEX ou XPRESS-MP. Cependant, malgré l'efficacité remarquable et sans cesse améliorée de ces logiciels, la résolution globale de ce problème reste inaccessible en raison de l'explosion combinatoire. C'est pour cette raison que nous proposons une décomposition du problème global, ainsi qu'une relaxation du problème d'affectation généralisée en un problème d'affectation simple.

Une décomposition temporelle et par qualification pour l'affectation de vacations

Le problème global étant formulé sur un mois calendaire, nous proposons une décomposition temporelle et par qualification. Chaque sous-problème est relatif à une qualification et couvre un intervalle de temps plus court (typiquement une journée de travail). Les vacations représentant des journées de travail, une décomposition par jours est presque naturelle. Un ensemble de sous-problèmes est ainsi défini pour chaque groupe de vacations qui commencent le même jour. Ainsi, la décomposition par jour n'exclut en rien les périodes de travail nocturnes. Les contraintes du problème global qui couplent les sous-problèmes sont celles liées aux durées de travail et de repos : contraintes sur n jours glissants et contraintes sur une période de plusieurs semaines. Dans la décomposition séquentielle journalière du problème global, ces contraintes vont être réparties sur chaque jour, cette répartition étant mise à jour de façon dynamique à partir du calcul des cumuls de durées de travail et de durées des repos.

Une décomposition temporelle et par qualification pour l'affectation de rotations

Une rotation a une structure un peu plus complexe qu'une vacation, car elle s'étale sur plusieurs jours et est effectuée par un équipage complet et non pas par une seule personne. Même si la décomposition du problème par jours de travail n'est plus réalisable, une décomposition temporelle est toujours possible si les rotations sont groupées suivant leur date de début : un sous-problème est ainsi défini pour chaque groupe de rotations qui commencent le même jour.

Les équipages changent au cours du mois, les navigants tournent entre plusieurs équipages. Etant données aussi les contraintes qui existent au sein d'un équipage, une autre

décomposition sera faite par qualification (pilotes, copilotes, hôtesses…). On appellera « job » le travail élémentaire d'une qualification donnée au sein d'une rotation donnée. Dans le cas où *p* navigants ayant la même qualification sont requis sur une rotation, on définira *p* jobs et on résoudra en séquence *p* sous-problèmes de façon à traiter de proche en proche les contraintes internes liées aux éventuelles incompatibilités réglementaires entre membres du même équipage.

Une troisième décomposition possible peut se faire selon les secteurs d'avion, car les navigants ne sont pas obligatoirement qualifiés sur tous les secteurs d'avion. Cette décomposition sera faite seulement dans les cas où les navigants sont mono qualifiés. En cas de multi qualification, la découpe du problème par secteur n'a pas d'intérêt.

Pour conclure, le problème d'affectation de rotations peut être décomposé dans le temps, par qualification de navigants et, éventuellement, par secteur d'avion: un sous-problème est ainsi défini pour chaque groupe de rotation qui commence le même jour et pour tous les navigants disponibles ayant la même qualification (sur le secteur avion correspondant aux rotations sélectionnées).

Problème d'affectation simple

Le sous-problème d'affectation défini ci-dessus peut être vu comme un problème d'affectation généralisé car, en plus de la contrainte de base qui impose que chaque vacation/ rotation doit être affectée à un agent, il y a des contraintes supplémentaires à respecter. Nous proposons une transformation du problème d'affectation généralisé en problème d'affectation simple par relaxation Lagrangienne : les contraintes sont dualisées et des termes pénalisant leur non respect sont intégrés dans le critère sous forme de coûts. Compte tenu du grand nombre de contraintes, les valeurs des pondérations correspondantes sont établies a priori et non à l'aide d'une méthode itérative. Toutes les contraintes seront ainsi traitées comme des contraintes souples, le non-respect des contraintes strictes étant très fortement pénalisé (méthode du « grand M » en Programmation Linéaire). Au paragraphe § 5.5.2, nous allons détailler le calcul des pondérations.

En résumé, le problème global de multi-affectation généralisé avec beaucoup de variables est décomposé en plusieurs sous-problèmes d'affectation simple. Cette décomposition confère en outre à la technique de résolution, un caractère modulaire facilitant l'interprétation des résultats et la prise en compte de modifications locales. Les contraintes couplantes, généralement d'origine réglementaire, sont transformées en contraintes localisées par des mises à jour temporelles à la fin de chaque période. En outre, le paramétrage des contraintes souples est réalisé grâce à une interface. Le programme fournit ainsi une aide à la décision pour le responsable en charge du planning.

5.5 Méthode de résolution proposée

Pour la résolution du problème, nous proposons une combinaison de deux méthodes classiques de résolution des problèmes d'affectation. Une méthode duale est utilisée comme méthode de base, mais l'étape de décision sur l'admissibilité de la solution courante est effectuée par une méthode primale. Nous avons retenu l'algorithme hongrois comme méthode duale et le Bip Match comme méthode primale. La combinaison choisie essaie de maximiser les avantages des deux méthodes et de minimiser leurs inconvénients.

L'avantage d'une méthode duale est qu'elle fournit toujours la solution optimale, s'il en existe une. Son inconvénient est le fait que la n'est obtenue qu'à la convergence de

l'algorithme (caractéristique de base des méthodes duales, qui sont dites « non-admissibles »). Il n'y a donc pas de solution intermédiaire admissible disponible dans le cas où la convergence est lente. De plus, cette méthode peut prendre beaucoup de temps dans le cas de plusieurs solutions équivalentes, car elles sont toutes explorées.

Les méthodes primales construisent une séquence de solutions admissibles qui converge vers l'optimum. Ceci a pour conséquence que, en mettant des délais temporels pour le déroulement de l'algorithme, si l'algorithme n'a pas trouvé l'optimum à la fin de ce délai, il fournit une solution admissible, mais sous-optimale. On peut imposer qu'une méthode primale améliore strictement la solution courante à chaque itération, ce qui évite l'exploration des solutions équivalentes. Appliqué à des matrices creuses (beaucoup de solutions équivalentes), ce type de méthode est alors très rapide. L'algorithme primal choisi, le Bip Match, s'applique à la recherche d'un couplage maximal de coût nul. Dans ce cas, la solution fournie par le Bip Match n'est donc en général qu'une solution partielle.

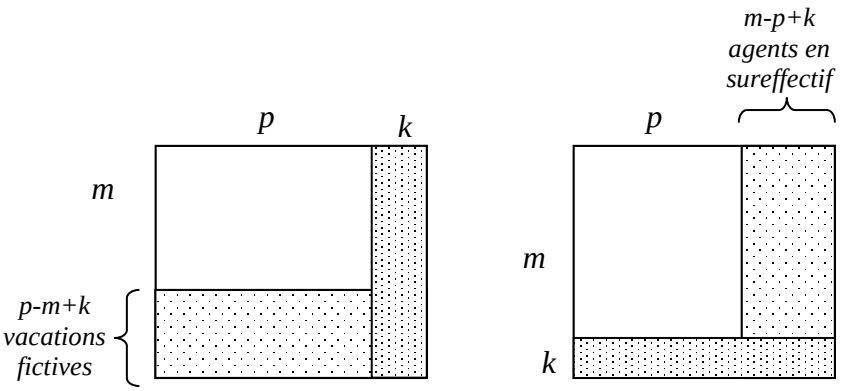
Il faut mentionner que les deux algorithmes choisis ont des complexités polynomiales et très proches l'une de l'autre : l'algorithme hongrois a une complexité en $O(n^3)$ et le Bip Match a une complexité en $O(n^2)$, où *n* représente le nombre des vacations à affecter.

5.5.1 Construction de la matrice de coût

L'algorithme hongrois débute par la construction de la matrice de coût. Les dimensions de la matrice sont fournies par les nombres des vacations/rotations (V/R) et des agents susceptible de les effectuer: une ligne pour chaque V/R et une colonne pour chaque agent. La particularité de cette matrice est qu'elle doit être carrée, se qui signifie que l'algorithme hongrois résout le problème seulement dans le cas où le nombre des V/R est égal au nombre des agents. En pratique, ceci n'arrive presque jamais. Il faut donc trouver une méthode pour ramener notre problème à la forme imposée par l'algorithme standard.

Dimensionnement de la matrice

Soit *m* le nombre des V/R à effectuer et *p* le nombre des agents supposés à les exécuter ($m \neq p$) ; la matrice a donc *m* lignes et *p* colonnes. Pour 'transformer' la matrice dans une matrice carrée, nous introduisons des lignes (si $m<p$) ou des colonnes (si $m>p$) supplémentaires, jusqu'à ce que les nombres de lignes et de colonnes de la matrice soit égal à $n = max(m, p) + k$. Les k lignes ou colonnes supplémentaires donnent des degrés de liberté, pour éviter par exemple des affectations interdites. La valeur de k est établie de manière expérimentale. Nous allons parler plus en détail de la signification des affectations interdites dans §5.2.4.2. Les lignes supplémentaires sont interprétées comme des V/R fictives et les colonnes supplémentaires comme des agents en sureffectif. La matrice finale présente un des profils représentés sur la **Figure 5.1**.



a) ***m*<*p***. On rajoute *p-m+k* lignes supplémentaires.

b) ***p*<*m***. On rajoute *m-p+k* colonnes supplémentaires.

Figure 5.1. La matrice des coûts

5.5.2 Calcul des coûts

Chaque élément de la matrice ($1 \le i \le n, 1 \le j \le n$), représente le coût d'affectation de la V/R *i* à l'agent *j*. Ce coût est calculé en fonction de chaque contrainte à respecter, par dualisation des contraintes. Pour chaque paire (V/R *i*, agent *j*) et chaque contrainte réglementaire un coût pondéré est établi. Le coût total correspondant au couple (vacation *i*, agent *j*) est donné par la somme de coûts de chaque règle à respecter.

Dans notre modèle, la valeur des coûts varie entre 0 et M (où $M \rightarrow +\infty$). Un coût égal à 0 représente le cas idéal d'affectation, un coût égal à M rend l'affectation impossible.

Le coût associé à chaque règle dépend de l'importance accordée à la règle. Les règles dures se traduisent facilement en termes de coûts. Comme ce sont des règles de type «*tout ou rien* », le coût imposé par une telle règle pour l'affectation d'une V/R à un agent est 0 si la règle est respectée, ou M sinon. Les règles souples se traduisent parfois par des objectifs contradictoires. Par conséquent, chacune d'entre elles aura un coût « brut » et une pondération associée à ce coût, comprise dans l'intervalle [0%,100%]. Le coût brut est une fonction des caractéristiques du couple (vacation *i*, agent *j*) pour lesquels la règle est vérifiée. La pondération est introduite dans le modèle de manière interactive, de façon que l'utilisateur puisse choisir la pondération de chaque règle souple, en fonction de l'importance qu'il accorde à chacune d'entre elles. Ainsi, le coût final imposé par une règle souple pour l'affectation (V/R *i*, agent *j*) est le produit entre le coût « brut » de la règle et de la pondération établie par l'utilisateur. Comme les règles souples servent à améliorer la qualité des plannings créés, leurs coûts ne peuvent jamais prendre la valeur M (qui interdirait l'affectation).

Cette manière de calculer les coûts n'est valable que pour $1 \le i \le m, 1 \le j \le p$. Pour les éléments supplémentaires, le coût est calculé différemment. On distingue trois situations différentes :

(1) $1 \le i \le m$ et $p < j \le n$. Cette situation correspond à un couple (vraie V/R, agent en sureffectif).

(2) $m < i \le n$ et $1 \le j \le p$. Cette situation correspond à un couple (V/R fictive, agent réel).

(3) $m < i \le n$ et $p < j \le n$. Cette situation correspond à un couple (vacation fictive, agent en sureffectif).

Chaque élément qui correspond aux points (1) ou (2) aura une valeur stricte positive b<<M, choisie de manière expérimentale. L'explication pour cela est assez simple : si, à un moment donnée, pour une V/R *i* (i=1..m) il n'y a aucun agent qui puisse l'effectuer (c_{ij} = M, j=1…p), on affecte la V/R à un agent temporaire (j > p) dont $coût_{ij}$ = b. De même manière si, pour un agent j il n'y a aucune vraie V/R qui peut lui être affectée, on lui en affecte une « fictive » ($i > m$). Par convention, chaque coût d'affectation $coût_{ij}$ ($i > m$, $p < j \leq p + k$ ou $j > p$ et $m < i \leq m + k$) d'une V/R fictive à un agent en sureffectif est nul.

Pour tout couple (V/R *i*, navigant *j*), le non-respect d'une seule règle dure rend l'affectation impossible. Comme les règles à vérifier sont assez nombreuses et coûteuses en terme de temps de calcul, pour diminuer le temps de réponse, les règles dures seront testées avant les règles souples et chaque fois qu'un règle dure n'est pas respectée par un couple (V/R *i*, navigant *j*), on passe au couple suivant, sans tester les autres règles.

Les fonctions des coûts, en général non linéaires, sont approximées par des fonctions linéaires par morceaux. Les pondérations représentent les paramètres de Kuhn – Tucker. Les valeurs des pondérations sont calculées de manière expérimentale et sans rechercher leurs valeurs optimales, car une valeur qui permet la prise en compte de la règle suffit.

Les pondérations associées à chaque règle dépendent de l'importance accordée à la règle. Les valeurs des paramètres de Kuhn – Tucker associés aux règles dures sont trouvées facilement. Comme ce sont des règles de type «*tout ou rien* », le non respect d'une telle règle rend l'affectation impossible. Il est connu qu'en surévaluant la valeur du paramètre de Kuhn – Tucker correspondant, la règle est prise en compte. Donc, pour chaque règle dure, nous avons décidé de surévaluer cette valeur à M (où $M \rightarrow +\infty$)

Disponibilité. La disponibilité d'un agent étant essentielle pour l'exécution d'une V/R, le test de disponibilité peut être considéré comme une règle dure. En même temps, pour diminuer au minimum le temps de calcul, nos utilisons ce test comme premier test dans le calcul des coûts.

Etablir si un navigant est disponible pour une rotation est un processus un peu plus complexe que la vérification de la disponibilité d'un personnel au sol. La simple vérification du fait que le navigant n'a pas d'autres activités pré-affectées pendant toute la durée de la rotation ne suffit pas. La validité du contrat d'embauche et de la qualification doit être prise en compte aussi. En effet, le contrat d'embauche ou le contrat de qualification peuvent commencer ou finir en cours du mois. Le navigant peut effectuer une rotation seulement si la durée de la rotation est entièrement comprise dans la période de validité du contrat d'embauche et du contrat de qualification.

L'algorithme de calcul des éléments c_{ij} ($1 \leq i \leq m$, $1 \leq j \leq p$) est le suivant :

Procedure ***Matrice_coût***

```
Pour i = 1…m
    Pour j = 1…p

        //étape d'initialisation

          peut_travailler = VRAI

          Si agent_j Est_disponible pour V/R r_i
             c_ij = 0
          Sinon
```

```
                c_ij = M
                peut_travailler = FAUX
            Fin Si

        //c_ij pour les règles dures

        Si (peut_travailler)
            Tant qu'(il y a des règles dures à vérifier et peut_travailler)
                Si régle_courante n'est pas vérifiée
                    c_ij += M
                    peut_travailler = FAUX
                Fin Si
            Fin Tant que
        Fin Si

        //c_ij pour les règles souples

        Si (peut_travailler)
            Pour chaque règle souple
                    c_ij += CoûtRègle*PondérationRègle
            Fin Pour
        Fin Si
    Fin Pour
Fin Pour
```

Introduction des 'cumuls'

Chaque problème d'affectation simple est exécuté a priori indépendamment des autres. L'inconvénient est qu'à l'intérieur du problème il n'y a pas de 'visibilité' sur tout l'horizon temporel concernant le problème global. Donc, si la solution d'un sous problème respecte toutes les contraintes du problème local, il est possible que l'ensemble de toutes les solutions ne respectent pas les contraintes globales. Prenons un exemple simple. Dans le cas de l'affectation de rotations, le problème global *P* est décomposé en *l* sous-problèmes de base. Soit deux problèmes de base, P_i et P_k :

- P_i doit affecter un ensemble de rotations (R_1, …, R_s, … R_m) à un ensemble d'agents (a, b, c, ….). Dans la solution de P_i, l'agent *b* effectue la rotation Rs. Cette rotation commence le jour *j* et s'étend sur 4 jours : $j, j+1, j+2, j+3$.

- P_k affecte un ensemble de rotations (R_1, …, R_t, … R_n) à un ensemble d'agents (b, d, e…). La solution indique que l'agent *b* effectue la rotation R_t qui commence le jour $j+4$ et a une durée de 5 jours : $j+4, j+5, j+6, j+7, j+8$.

Supposons qu'une des règles dures impose au maximum 7 jours de travail consécutifs. L'agent b travaille 4 jours pour effectuer la rotation R_s et 5 jours pour effectuer la rotation R_t. Donc, cette règle est respectée pour chaque sous-problème traité indépendamment. Ce qui n'a pas été pris en compte est le fait que les deux rotations sont consécutives, donc *b* travaille en total 9 jours consécutifs, ce qui n'est plus réglementaire.

Pour éviter ce genre de situation, nous introduisons des cumuls qui gardent l'historique de chaque agent sur tout l'horizon : temps de travail/ de vol déjà effectué, jours consécutifs travaillés. Les cumuls sont mis à jour à la fin de chaque sous-problème. Ainsi, chaque règle sera vérifiée lors de chaque sous-problème par rapport à ces cumuls, et non plus par rapport aux valeurs absolues du problème global.

Soit le même problème P décomposé en *l* sous-problème : $P_1, P_2, ..., P_k, ..., P_l$ et le même agent *x*. La solution de chaque problème P_k affecte une rotation d'une durée d_k à l'agent *x*. La durée maximum du travail sur tout l'horizon est $D_{\max}$. Lors de la résolution du problème P_i, l'agent x a effectué déjà $Cum(x) = \sum_{ind=1}^{k-1} d_{ind}$ heures de travail. Donc on peut lui affecter une rotation *r* d'une durée d_r seulement si $d_r \leq D_{\max} - Cum(x)$.

Algorithme de résolution

Une fois la matrice construite, l'algorithme retient les coûts nuls ou, s'il n'y en a pas, il en crée pour chaque ligne et chaque colonne de la matrice. A ce point, on intègre l'algorithme du Bip Match qui cherche une affectation optimale complète à partir de ces coûts. S'il n'en trouve pas, on commute vers l'algorithme hongrois qui crée de nouveaux coûts nuls pour un nouvel appel du Bip Match. Le processus se répète jusqu'à ce que l'on trouve une solution optimale ou jusqu'à ce que le délai imposé pour trouver le résultat touche à sa fin. Dans le deuxième cas, le Bip Match, qui fournit une solution sous optimale mais partielle, est complété par quelques itérations de l'algorithme de Busacker et Gowen, ce qui assure une solution complète, mais sous-optimale. La Figure 5.2 présente le schéma de résolution proposée.

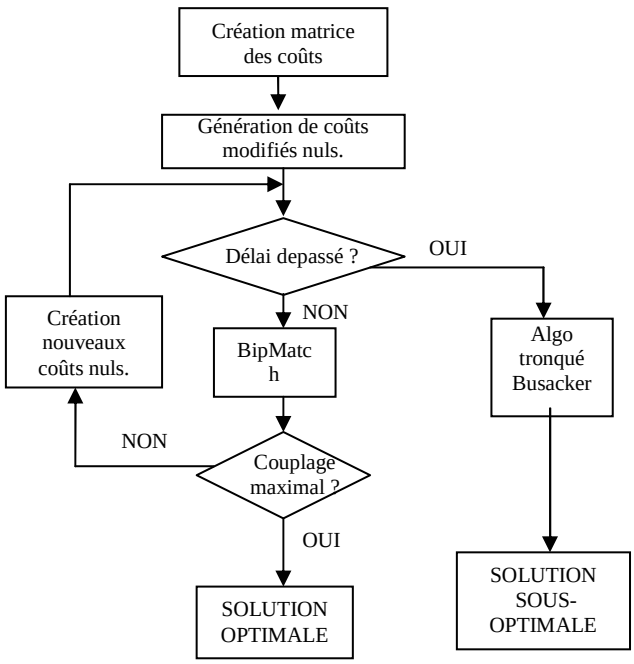


Figure 5.2. *Algorithme d'affectation*

5.5.3 L'affectation de réserves pour le personnel navigant

Le problème d'affectation de rotations prévoit aussi l'affectation d'un nombre prédéfini de réserves par jour et pour chaque qualification. Nous rappelons que les réserves sont des journées de travail à horaire fixe ne contenant a priori aucun vol.

Pour l'affectation de réserves, nous utilisons la même méthode de résolution que pour les rotations, les réserves étant déjà partagées par qualification et par jour.

Une première idée a été d'accorder la même importance aux réserves qu'aux rotations et de les affecter en même temps. Mais, si les rotations doivent être effectuées obligatoirement, les réserves sont effectives seulement si un navigant se retrouve dans l'impossibilité d'effectuer une rotation qui lui à été déjà allouée et 'donne' sa place à la réserve. Le navigant en réserve reste bloqué au sol durant la journée, mais il n'effectue pas obligatoirement une rotation pendant sa réserve.

Ces réflexions nous ont décidé d'affecter les rotations en premier et ensuite les réserves aux navigants qui n'avaient pas reçu des rotations.

On commence par la décomposition temporelle : pour chaque jour du mois, on cherche toutes les rotations qui commencent ce jour et on les met dans *Liste_rotations_du_jour*. On parcourt la liste des rotations pour déterminer les secteurs couverts et les qualificaitons requises par les configuration équipage ; on crée les listes `Liste_secteurs_du_jour` et `Liste_qualifications_du_jour`.

On continue avec la décomposition par secteur et par qualification : pour chaque secteur de la `Liste_secteurs_du_jour` et chaque qualification de la `Liste_qualifications_du_jour` on crée la `Liste_rotations` qui contient toutes les rotations du jour sur le secteur courant ayant besoin d'au moins un navigant avec la qualification courante. Ensuite, on cherche tous les navigants ayant la qualification courante sur le secteur courant ; la `Liste_navigants` est créée.

A ce moment, on peut affecter les rotations de la `Liste_rotations` aux navigants de la `Liste_navigants` en utilisant la méthode proposée dans § 5.2.2.

Une fois les rotations affectées, les réserves peuvent être affectées à leur tour. La `Liste_navigants` est mise à jour, en enlevant de la liste les navigants qui ont été affectés à des rotation. On cherche les réserves à affecter pour la qualification courante et pour le secteur courant ; la `Liste_réserves_du_jour` est créée. Les réserves de la `Liste_réserves_du_jour` sont affectées aux navigants de la `Liste_navigants` de la même manière que les rotations.

Ci-dessous, la procédure ***`Affectation`*** décrit l'affectation de rotations et de réserves :

Procedure ***`Affectation_PN`***

```
Pour (chaque jour_du_mois)
   Crée (Liste_rotations_du_jour)
   Crée (Liste_secteurs_du_jour)
   Crée (Liste_qualifications_du_jour)
   Pour secteur_courant parcourt (Liste_secteurs_du_jour)
      Pour qualification_courante parcourt
                              (Liste_qualifications_du_jour)
         Crée (Liste_rotations)
         Crée (Liste_navigants)
         Affecte les rotations de la Liste_rotations
                     aux navigants de la Liste_navigants
-------------------------------------------------------------------
         Met_à_jour (Liste_navigants)
         Crée (Liste_réserves_du_jour)
         Affecte les réserves de la Liste_réserves_du_jour
                     aux navigants de la Liste_navigants
      Fin Pour
   Fin Pour
Fin Pour
```

Affectation de rotations

Affectation de réserves

5.5.4 Affectation de séquences

Comme nous l'avons vu dans le chapitre 4, à la fin de l'étape de création de rotations il peut y avoir un post traitement des rotations qui consiste en la construction des séquences. Si tel est le cas, le processus d'affectation traite les séquences et non plus les rotations indépendantes. Pour le problème d'affectation, les séquences peuvent être assimilées à des rotations beaucoup plus longues et avec une structure un peu plus complexe, car les séquences permettent l'existence des jours de repos à l'intérieur. Donc, l'algorithme d'affectation a pu être utilisé pour le traitement des séquences sans qu'il ait été nécessaire de lui apporter des modifications importantes.

Parfois, les séquences créées ne couvrent pas l'intégralité des rotations à effectuer durant le mois. Dans ce cas, les séquences sont traitées en premier, les rotations étant affectées une fois que les séquences ont trouvé leurs places dans les plannings des navigants. Cet ordre est naturel et se justifie par la taille plus grande des séquences en comparaison avec celle des rotations. Comme dans les algorithmes off-line de résolution de problèmes de *bin-packing*, les objets sont triés en ordre décroissant de leur taille, car un gros objet est plus facile à insérer au début qu'à la fin et, en même temps, il est plus facile de compléter un conteneur déjà rempli avec un petit objet qu'avec un gros. D'un autre côte, le choix de cet ordre s'explique par l'existence même des séquences : rappelons que leur construction a comme but une meilleure répartition de la charge de travail entre navigants, donc l'amélioration de la qualité des plannings individuels. C'est donc normal de leur accorder plus d'importance.

5.6 Implémentation et résultats

5.6.1 Validation de la méthode

L'algorithme proposé pour la résolution des problèmes d'affectation simple a été implémenté en C++ et testé sur plusieurs jeux de données. Ces jeux de données on été générés aléatoirement, en variant la dimension de la matrice de coûts et la plage des valeurs des coûts. Le tableau 5.1 synthétise les résultats obtenus :

Nb. problèmes testés	4600
Dimension (n) matrice de coûts (nxn)	entre 5 et 50
Valeurs coûts	entre 0 et 1000
% problèmes avec solution sous-optimale	15%
Temps de réponse	1- 2 secondes

Tableau 5.1. *Validation de l'algorithme d'affectation*

Nous avons considéré que les tests effectués sur des matrices de dimension plus grande que 50 par 50 ne sont pas d'intérêt, car dans le cas réel étudié, la dimension de ces matrices dépasse rarement cette limite. En ce qui concerne les bornes des valeurs des coûts, nous avons choisi 0 pour le cas idéal d'affectation ; 1000 représente la valeur choisie pour M (cf. § 5.5.2).

Pour avoir une meilleure vision des résultats obtenus, le problème d'affectation simple a été modélisé en tant que programme linéaire, implémenté à l'aide d'un module de

programmation linéaire commercial (Xpress-MP) et testé sur plusieurs jeux de données. Dans tous les cas, le programme linéaire a fourni la solution optimale en moins d'une seconde. Les mêmes jeux de données ont été utilisés pour tester la méthode que nous proposons.

Le Tableau 5.2 présente les résultats fournis par l'algorithme d'affectation proposée :

	n = 10	**n = 20**	**n = 30**	**n = 40**	**n = 50**
Nb. problèmes résolus	1000	1000	1000	500	500
% sol sous-opt.	11.6	25	22.6	23	36.6
Dev.moyenne sol. opt.	1.2%	1.2%	1.6%	0.8%	0.5
Dev.max sol. opt	11%	11%	11%	9%	5%
Temps réponse moyen[s]	<1	1	2	4	4
Temps réponse max[s]	<1	2	14	116	118

Tableau 5.2 *Résultats pour l'algorithme d'affectation*

On peut observer qu'en fonction de la dimension de la matrice de coûts, le pourcentage de problèmes dont la solution obtenue est sous-optimale augmente. Ce qui est important à préciser est le fait que la déviation par rapport à la solution optimale calculée par le programme linéaire reste limitée – moins de 2%. Il y a des cas où la solution sous-optimale s'écarte à plus de 10 pour cent de la solution optimale , mais ils sont très isolés.

Généralement, le temps moyen de réponse d'un problème varie entre 1 et 2 secondes. Pour des problèmes dont la taille de la matrice de coûts est comprise dans l'intervalle [30, 40], le temps de réponse peut augmenter jusqu'à 10 – 15 secondes. Dans peu de cas et pour des matrices de coûts avec plus de 40 lignes, le temps de réponse peut dépasser 1 minute.

La solution d'un problème résolu par la méthode proposée est un compromis entre le temps de réponse et une solution de qualité (solution optimale). L'algorithme a été conçu pour s'arrêter après un certain nombre d'itérations. Pour la valeur choisie, les tests ont prouvé que la solution sous-optimale obtenue reste de bonne qualité. A priori, cette qualité peut être encore améliorée si le nombre d'itérations augmente.

Notre méthode de résolution a des poins faibles par rapport à la méthode exacte utilisée par la programmation linéaire : temps de réponse plus élevé et solution pas toujours optimale. Pourtant, l' « abandon » de la programmation linéaire est justifié. Nous rappelons que nos travaux ont eu aussi un but industriel : comme la compagnie IFR France développe des produits logiciels commerciaux, un module d'affectation de personnel devait être intégré dans un de ces logiciels. Pour des raisons de coût économique du produit final destiné aux compagnies aériennes, une des contraintes du cahier des charges interdisait l'utilisation des logiciels commerciaux de programmation linéaire. En même temps, les résultats obtenus suite aux tests, ont prouvé que la méthode proposée fournit des résultats de qualité dans un temps de réponse raisonnable.

5.6.2 Implémentation

La compagnie IFR France a développé un produit commercial pour l'affectation automatique des rotations, produit qui intègre la méthode de résolution proposée dans ce chapitre. Il crée des plannings pour tous les navigants d'une compagnie aérienne, en leur affectant des rotations, des jours de réserve et des jours de repos si nécessaire. L'horizon temporel considéré est d'un mois calendaire.

Le module est assez flexible, car il ne déclenche pas automatiquement le processus global d'affectation. L'utilisateur a tout d'abord la possibilité de choisir le mois, la base, les secteurs avion et les qualifications pour lesquels il désire effectuer l'affectation (Figure 5.3).

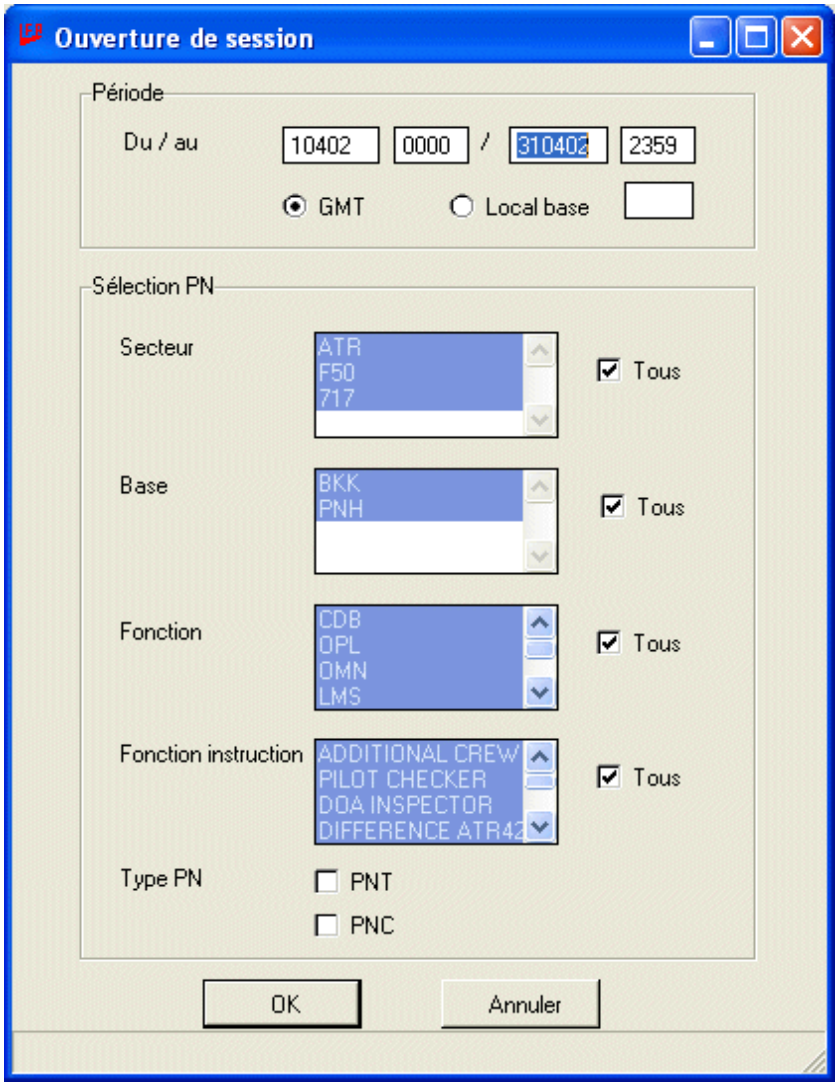


Figure 5.3 *L'utilisateur peut choisir le mois, la base, les secteurs avion et les qualifications pour lesquels il désire effectuer l'affectation*

Ensuite, l'utilisateur choisit des pondérations pour les règles souples (cf. § 5.5.2), en fonction de l'importance qu'il veut accorder à chaque règle (Figure 5.4)
:

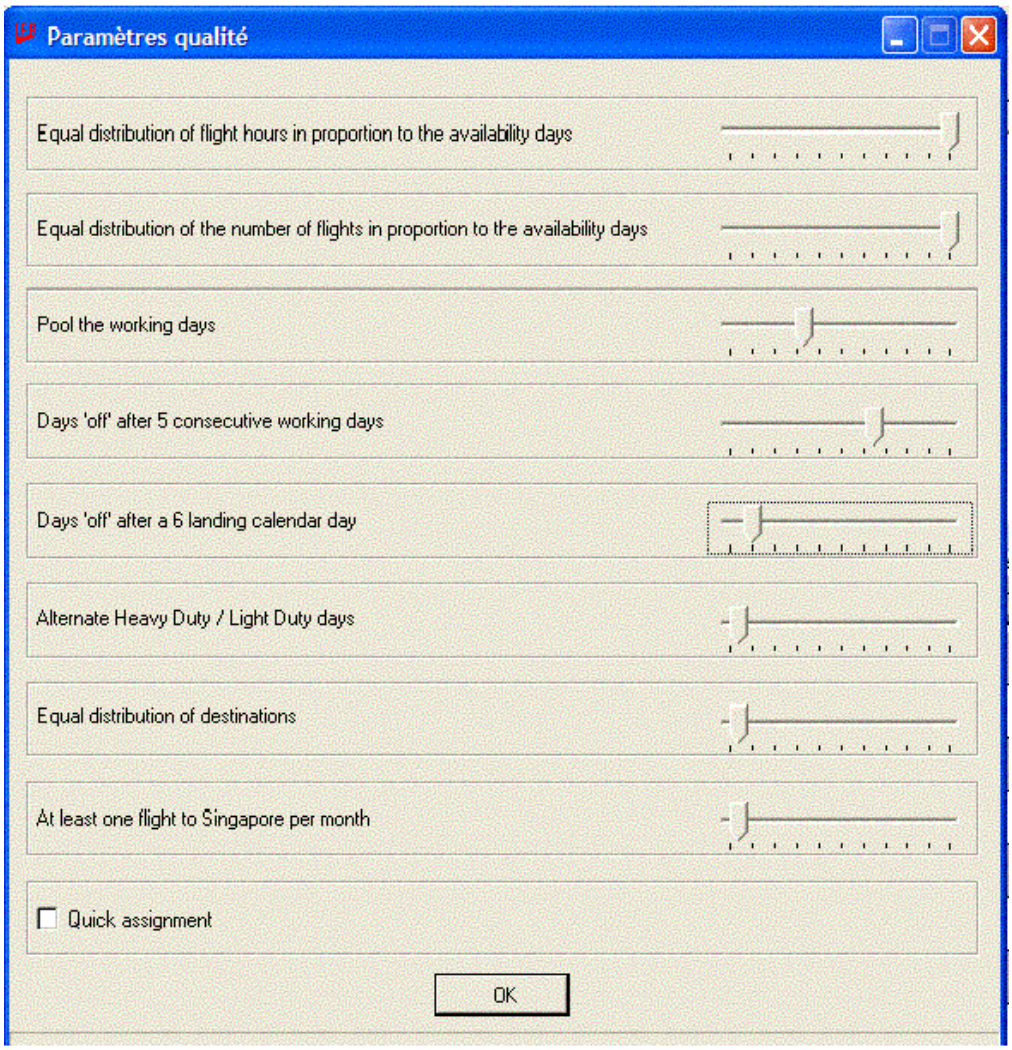


Figure 5.4 *Pondérations associées aux règles souples*

Avant de lancer l'affectation, une visualisation les plannings des navigants concernés est possible. A ce stade, leurs plannings contiennent seulement les activités pré-affectées.

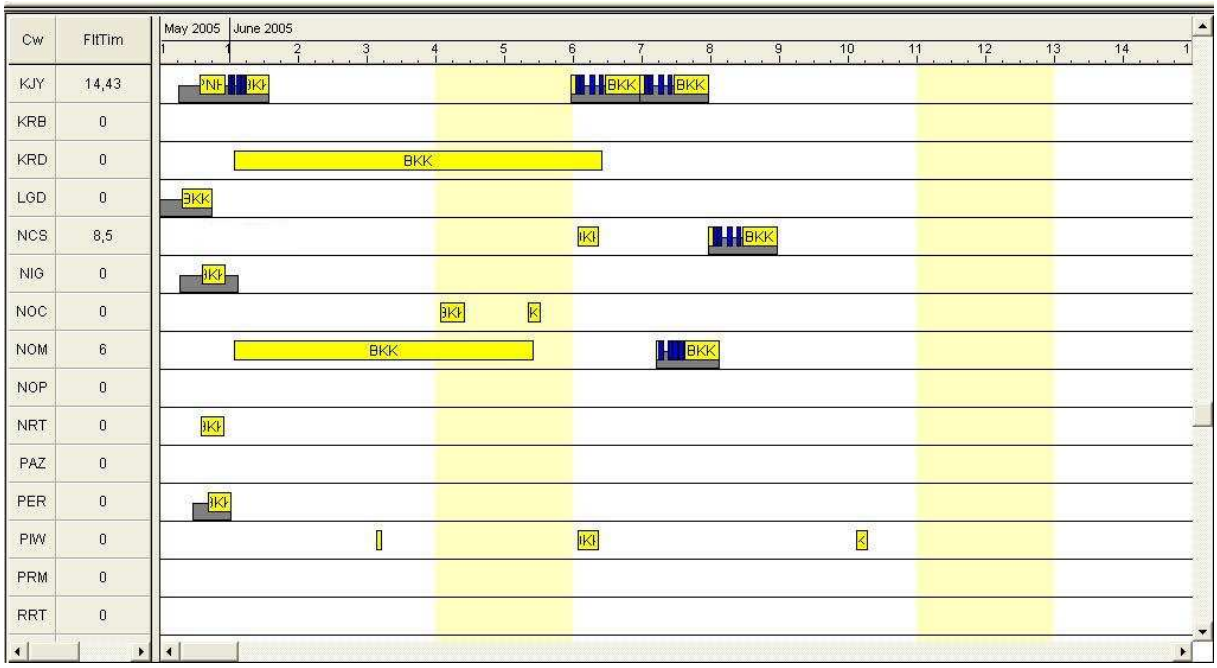


Figure 5.5 *Avant l'affectation, les plannings des navigants contiennent seulement les activités pré-affectées*

Une fois l'affectation effectuée et acceptée par l'utilisateur, les plannings sont mis à jour :

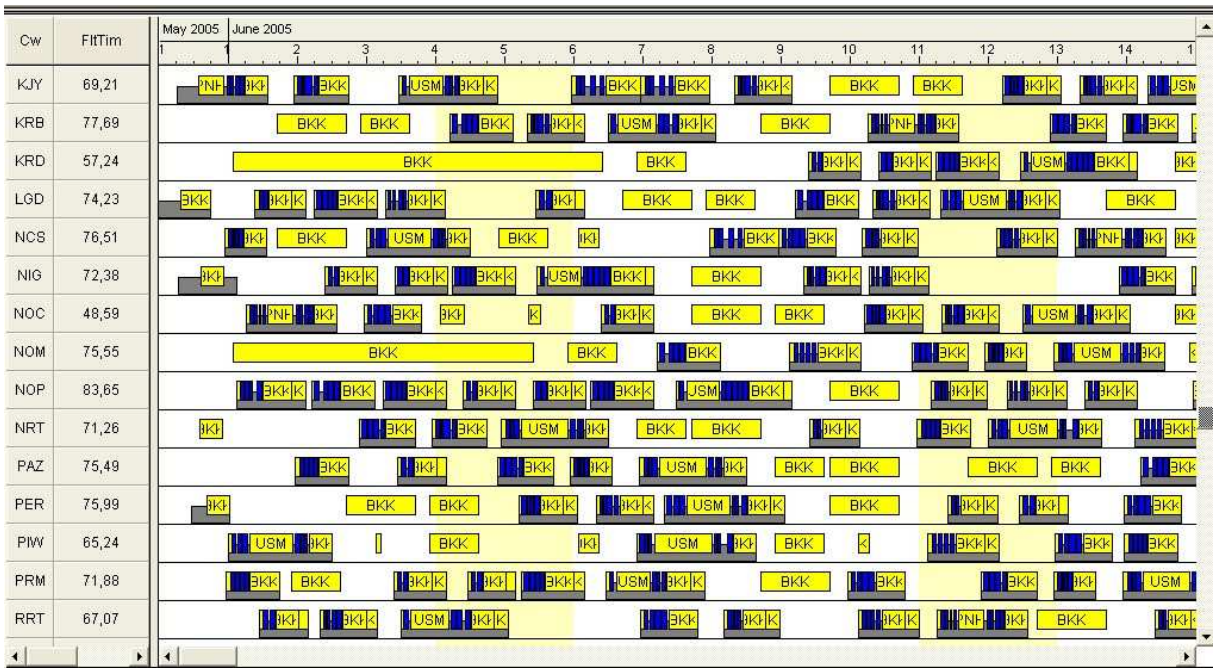


Figure 5.6. *Les plannings des navigants à la fin de l'affectation*

Le module d'affectation a été implémenté en Visual C++. Des tests ont été effectués sur des données réelles, relatifs à une compagnie aérienne de taille moyenne. Pour des raisons de confidentialité, son nom ne sera pas précisé. Tous les tests on été effectués sur un processeur Intel Celeron (930 MHz, 248 Mo de RAM). Quelques informations sur les cas testés sont présentées dans le Tableau 5.3

Nb. rotations	entre 555 et 959
Nb. navigants	entre 71 et 191
Nb. secteurs avions	entre 1 et 3
Nb. sous-problèmes résolus	entre 112 et 206
Temps de réponse	entre 7 et 1500

Tableau 5.3

5.7 Conclusion

Ce chapitre présente la deuxième étape du problème de création de plannings, celui de l'affectation des vacations/ rotations. Nous avons vu que dans la littérature, ce problème est souvent modélisé comme un problème de recouvrement/ partitionnement d'ensemble et résolu à l'aide de la programmation linéaire et des techniques de *Branch&Bound* et/ ou de génération de colonnes.

Nous proposons une approche de modélisation du problème en tant que problème d'affectation généralisé. Après une décomposition temporelle et par qualification, chaque sous-problème résultant est transformé en un problème d'affectation simple.

Un algorithme ad-hoc qui combine deux méthodes classiques de résolution des problèmes d'affectation est utilisé pour résoudre chaque problème d'affectation simple. Bien que les résultats obtenus en résolvant le même problème comme un programme linéaire soient de meilleure qualité, les résultats fournis par notre méthode de résolution restent comparables avec ces derniers.

Le chapitre se conclut par la présentation d'un produit commercial développé par IFR France pour l'affectation de rotations au personnel navigant d'une compagnie aérienne et qui utilise notre approche de résolution.

Conclusion

La création de plannings dans un contexte aéronautique attire l'attention de la communauté de Recherche Opérationnelle depuis plus de quarante ans, car ce problème concentre de nombreux enjeux sur les plans commerciaux et méthodologiques et constitue un domaine d'application privilégié pour l'optimisation combinatoire. En début de ce manuscrit nous présentons les principaux problèmes soulevés par la création de plannings, ainsi que des modèles et des méthodes proposés dans la littérature pour leur résolution.

Nos travaux ont porté sur la construction de plannings pour le personnel travaillant dans un contexte aéronautique : le personnel navigant d'une compagnie aérienne et le personnel au sol d'un aéroport. Problème complexe et de grande taille, la création de plannings est divisée en deux sous-problèmes indépendants : pour le personnel au sol, le premier sous-problème consiste à grouper les tâches dans ce que l'on appelle des *vacations*. Les vols à effectuer par les navigants sont aussi groupés dans des séquences nommées *rotations*. Le deuxième sous problème est commun aux deux types de personnel et consiste à affecter aux agents les vacations/ rotations créées, de manière à construire des plannings personnalisés.

Nous nous sommes intéressés séparément aux problèmes de création de vacation et de rotations. Compte tenu de la ressemblance entre l'affectation de vacations et l'affectation de rotations, les deux cas ont été analysés ensemble, en tenant compte bien sûr des caractéristiques spécifiques de chaque problème.

La création de vacations étant un problème qui repose sur la satisfaction de nombreuses contraintes temporelles, nous proposons une approche de modélisation basée sur le concept de plage horaire. Le modèle obtenu permet une représentation simple des contraintes, à partir de laquelle nous proposons une méthode heuristique de construction de vacations basée sur la méthode FFD (First Fit Decreasing) et sur la génération de colonnes. La solution obtenue avec cet algorithme peut être améliorée si l'on veut que le critère du problème prenne aussi en compte la qualité des vacations construites. Nous analysons aussi le cas où les agents travaillent en grille : le remplissage de vacations. Un des avantages de l'algorithme que nous proposons pour le problème de création est qu'il peut être utilisé, avec quelques modifications, pour le problème de remplissage.

La création de rotations est un problème complexe, car, d'une part, la réglementation à laquelle les rotations sont soumises traite beaucoup de cas particuliers et, d'autre part, la combinatoire du problème est considérable. C'est pour cette raison que nous proposons une résolution en deux étapes : une première étape, la construction de périodes de service (PdS), groupe les vols dans des journées de travail (ou périodes de service) ; la deuxième étape enchaîne les journées de travail, de manière à créer des rotations. Nous proposons une modélisation du problème de création de périodes de services par programmation linéaire en variables mixtes. L'avantage de cette méthode est qu'elle peut prendre en compte les différents types des périodes de services. Pour le groupement des périodes de services dans des rotations, un modèle linéaire en variables entières est utilisé.

Les tests effectués sur les deux modèles proposés ont fourni de très bons résultats sur des jeux de données de petite taille, puisque la solution optimale est obtenue en quelques

secondes mais le temps de réponse augmente exponentiellement avec la taille du problème. Cet aspect nous amène à explorer d'autres modalités de résolution.

Pour la construction de PdS, une méthode heuristique basée sur un algorithme de listes a été envisagée et pourrait maintenant être implémentée. Cet algorithme est basé sur le fait que le temps d'escale entre deux vols dépend du fait que l'équipage effectue le vol suivant sur le même avion ou sur un autre. Dans le premier cas la durée du temps d'escale diminue. Comme l'objectif du problème reste de minimiser le temps hors-vol d'une PdS, cette caractéristique du temps d'escale peut être exploitée. Les vols sont groupés en listes selon l'avion qui les effectue (une liste pour tous les vols sur le même avion) et triés en ordre croissant de leur date et heure de départ; ensuite, les listes sont traitées séparément : pour chaque liste, on essaie de construire le maximum de PdS entièrement comprises dans la liste. Cette technique de construction réduit a priori la combinatoire du problème. S'il reste des vols non couverts, d'autres PdS sont construites par exploration des listes des vols.

En ce qui concerne le problème de construction de rotations à partir des PdS, la première voie à explorer pour rendre plus performante la construction de rotations reste la génération de colonnes.

Pour le problème d'affectation du personnel aux vacations et aux rotations, nous proposons une approche par décomposition et simplification du problème global. En raison des nombreuses contraintes réglementaires, le problème est modélisé comme un problème de multi-affectation généralisé. Nous proposons une décomposition temporelle et par qualification et ensuite une transformation du problème d'affectation généralisé en problème d'affectation simple par relaxation Lagrangienne. Un algorithme ad-hoc est utilisé pour la résolution de chaque problème de base.

La décomposition par qualification permet la prise en compte d'une manière plus flexible des contraintes existant entre les différentes qualifications au sein du même l'équipage. La décomposition temporelle 'divise' l'horizon du problème global en plusieurs intervalles de temps. Pour garder la 'visibilité' sur tout l'horizon, nous introduisons des cumuls qui gardent l'historique de chaque agent sur tout l'horizon : temps de travail/ de vol déjà effectué, jours consécutifs travaillés. Les cumuls sont mis à jour à la fin de chaque sous-problème. Ainsi, chaque règle sera vérifiée lors de chaque sous-problème par rapport à ces cumuls, et non plus par rapport aux valeurs absolues du problème global.

Notre méthode de résolution n'est pas aussi performante que la méthode exacte de programmation linéaire pour la résolution des problèmes d'affectation simple: temps de réponse plus élevé et solution pas toujours optimale. Pourtant, globalement, l' « abandon » de la programmation linéaire est peu pénalisant. Nous rappelons que nos travaux ont eu aussi un but industriel : le module d'affectation de personnel devait être intégré dans un des produits logiciels commerciaux de la compagnie IFR France. Pour des raisons de coût économique du produit final destiné aux compagnies aériennes, une des contraintes du cahier des charges interdisait l'utilisation des logiciels commerciaux de programmation linéaire. En même temps, les résultats obtenus suite aux tests, ont prouvé que la méthode proposée fournit des résultats de qualité dans un temps de réponse raisonnable.

Une des perspectives du travail concernant l'affectation de vacations/ rotations, concerne la prise en compte des desiderata des agents, car la satisfaction des souhaits du personnel a des répercutions importantes sur la qualité du travail. Deux types de desiderata sont distingués : les desiderata individuels et les desiderata concernant les couples travaillant ensemble. Des problèmes d'agrégation de préférences se posent alors et il serait intéressant de réfléchir à la logique d'agrégation à privilégier. Une approche multicritère pourrait être également développée pour le problème d'affectation : au critère classique du problème, de nature économique, l'aspect de satisfaction du personnel peut être ajouté comme deuxième critère.

A plus long terme, nous souhaitons intégrer dans de futures approches l'aspect prévision de la charge de travail, phase amont à la planification, et l'aspect régulation, phase aval du planning. Dans ce deuxième cas, l'analyse des aléas entraînant la modification des plannings pourrait déboucher sur la construction de plannings prévisionnels plus robustes, c'est-à-dire supportant mieux les aléas.

Bibliographie

[Allen 84] J. F. Allen: Towards a General Theory of Action and Time. *Artificial Intelligence*, vol. 23, pages 123-154, 1984

[Alvarez-Valdes *et al.* 99] R. Alvarez-Valdes, E.Crespo, J.M.Tamarit. Labour scheduling at an airport refuelling installation. *Journal of the Operational Research Society*, vol. 50, pages 211-218, 1999

[Anbil *et al.* 92] R.Anbil, R.Tanga, E.L.Johnson: A global approach to crew pairing optimisation. *IBM Systems Journal*, vol. 31, no. 1, pages 71-78, 1992

[Arabeyre *et al.* 69] J.P. Arabeyre, J. Fearnley, C. Steiger, W. Teather. The airline crew scheduling problem: A survey. *Transportation Science*, vol. 3, pages 140-163, 1969

[Balas & Padberg 76] E.Balas, M.W.Padberg. Set Partitioning: A survey. *SIAM Review*, vol. 18, pages 710-760, 1976

[Barnhart et al. 98a] C. Barnhart, N.L. Boland, L.W. Clarke, E.L. Johnson, G.L. Nemhauser, R.G. Shenoi. Flight string models for aircraft fleeting and routing. *Transportation Science*, vol. 32, no. 3, pages 208–220, 1998.

[Barnhart *et al.* 98b] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance: Branch and price : column generation for solving huge integer programs. *Operations Research*, vol. 46, no. 3, pages 316–329, 1998.

[Bechold & Brusco 94] S.E.Bechold, M.J.Brusco: Working set generation methods for labour tour scheduling. *European Journal of Operational Research*, vol. 74, pages 540-551, 1994

[Bellanti *et a*l. 04] F.Bellanti, G.Carello, F.Della Croce, R.Tadei: A greedy-based neighbourhood search approach to a nurse rostering problem. *European Journal of Operational Research*, vol. 153, pages 28-40, 2004

[Ben Amor 02] H. Ben Amor. Stabilisation de l'algorithme de génération de colonnes. Phd thesis, École Polytechnique, Montréal, 2002.

[Berge 83] Graphes, Gauthier-Villars, 1983

[Bès *et al*. 02] C.Bès, J-B.Hiriart-Urruty, M.Mongeau, D.Noll. Communication privée, Université Paul Sabatier, Toulouse, 2002

[Brown & Graves, 75] G.G.Brown, G.W.Graves.Elastic programming: a new approach to large scale mixed integer optimisation. Paper presented at ORSA/TIMS meeting, Las Vegas, 1975

[Brusco & Johns 96] M.J.Brusco, T.R.Johns: A sequential integer programming method for discontinuous labour tour scheduling. *European Journal of Operational Research*, vol. 95, pages 249-261, 1996

[Brusco & Showalter 93] M.J.Brusco , M.J.Showalter: Constraint nurse staffing analysis. *Omega*, vol. 21, no. 1, pages 175-186, 1993

[Buffa *et al.* 76] E.S. Buffa, M.J. Cosgrove, B.J. Luce: An integrated work shift scheduling system. *Decision Sciences*, vol.7, pages 620-630, 1976

[Busacker & Gowen 61] R.G.Busacker, P.J.Gowen. A procedure for determining a family of minimal-cost network flow patterns. *Technical Report 15, Operational Research Office*, Johns Hopkins University, Baltimore, 1961.

[Campbell *et al*. 97] K.W.Campbell, R.B.Durfee, G.S.Hines: FedEx Generates Bid Lines Using Simulated Annealing. *Interfaces*, vol. 27, no. 2, pages 1-16, 1997.

[Caprara *et al*. 98a] A.Caprara, F.Focacci, E.Lamma, P.Mello, M.Milano, P.Toth, D.Vigo: Integrating constraint logic programming and operations research techniques for the crew rostering problem. *Software Practice and Experience*, vol.28, no.1, pages 49-76, 1998

[Caprara *et al*. 98b] A.Caprara, P.Toth, D.Vigo, M.Fischetti : Modeling and solving the crew rostering problem. *Operations Research*, vol.46, no.6, pages 820-830, 1998

[Chan 02] P.Chan. La planification du personnel: acteurs, actions et termes multiples pour une planification opérationnelle des personnes. Thèse de doctorat, Université Joseph Fourier, Grenoble, 2002

[Christou *et al*. 99] I.T.Christou, A.Zakarian, J.-M.Liu, H.Carter: A Two-Phase Genetic Algorithm for Large-Scale Bidline-Generation Problems at Delta Air Lines. *Interfaces* vol. 29, no. 5, pages 51-65, 1999

[Chu *et al*. 97] H.D. Chu, E. Gelman, E. L. Johnson. Solving large scale crew scheduling problems. *European Journal of Operational Research* vol. 97, pages 260-268, 1997

[Coffman *et al*. 98] E.G.Cofmann Jr, G.Galambos, S.Marttello, D.Vigo: "Bin Packing Approximation Algorithms: Combinatorial Analysis" in *Handbook of Combinatorial Optimization*, D.-Z. Du and P. M. Pardalos Eds., Kluwer Academic Publishers, Amsterdam, 1998

[Crainic & Rousseau 87] T.G. Crainic, J. Rousseau. The column generation principle and the airline crew scheduling problem. *INFOR* vol. 25, pages 136-151, 1987

[Dantzig 54] G.B.Dantzig: A comment on Eddie's traffic delay at toll booths. *Operations Research*, vol. 2, pages 339-341, 1954

[Dantzig 63] G.B.Dantzig: Linera Programming and Extensions, Princetown University Press, 1963

[Davis & Putman, 96] M.Davis, H.Putman. A computing procedure for quantification theory. *Journal of the ACM*, vol. 7, pages 201-215, 1960

[Dawid et al. 01] H.Dawid, J.König, C.Strauss. An enhanced rostering model for airline crews. *Conputers & Operations Research* vol. 28, pages 671-688, 2001

[Desaulniers *et al*. 97a] G. Desaulniers, J. Desrosiers, Y. Dumas, S. Marc, B. Rioux, M. M. Solomon. Crew Pairing at Air France. *EJOR* vol. 97, pages 245-259, 1997

[Desaulniers *et al*. 97b] G. Desaulniers, J. Desrosiers, Y. Dumas, M.M. Solomon & F. Soumis. Daily aircraft routing and scheduling. Management Science, vol. 43, no. 6, pages 841–855, 1997.

[Desrochers & Soumis 88] M. Desrochers & F. Soumis. A generalized permanent labelling algorithm for the shortest path problem with time windows. INFOR, vol. 26, no. 3, pages 191–212, 1988.

[Desrochers *et al*. 92] M. Desrochers, J. Desrosiers & M.M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, vol. 40, no. 2, pages 342–354, 1992.

[Desrosiers et al. 84] J. Desrosiers, F. Soumis & M. Desrochers. Routing with time windows by column generation. Networks, vol. 14, pages 545–565, 1984.

[Desrosiers et al. 95] J. Desrosiers, Y. Dumas, M.M.Solomon, F.Soumis. Time constraint routing and scheduling. *Handbooks in Operations Research and Management Science. Network Routing*. vol. 8, pages 35-139, 1995

[Draghici 04] C.Draghici. Construction des plannings pour le personnel navigant de compagnies aériennes, *Rapport LAAS 04425*, 2004

[Draghici *et al.* 04] C.Draghici, J.C.Hennet, O.Paillot. Planification prévisionnelle des personnels navigants, Rapport Convention CIFRE N° 1412707.00 Rapport LAAS N°04005

[Draghici & Hennet, 04] C.Draghici, J.C.Hennet, Une approche par décomposition du problème d'affectation de rotations. *Actes 5ème Conference Francophone de MOdelisation et SIMulation (MOSIM'04)*, 2004, pp.481-486.

[Draghici & Hennet, 05] C.Draghici, J.C.Hennet, Generation of shift schedules - a time slot approach , *Actes IESM'05*, Marakech (Maroc) 2005.

[Duffuaa & Al-Sultan, 99] S.O.Duffuaa, K.S.Al-Sultan. A Stochastic model for scheduling maintenance personnel. *Applied Mathematical Modelling*, vol. 25, pages 385-397, 1999

[Dyckhoff 90] H. Dyckhoff: A typology of cutting and packing problems. *European Journal of Operational Research* vol. 44, pages 145-160, 1990

[Easton & Rossin 91] F.F.Easton, D.F.Rossin: Sufficient working subsets for the tour scheduling problem. *Management Science*, vol. 37, pages 1441-1451, 1991

[El Moudani *et al*.00] W.El Moudani, C.A.Cosenza, M.de Coligny, F.Mora-Camino: A bi-criterion Approach for the Airlines Crew Rostering Problem, rapport LAAS no. 00428, octobre 2000.

[Ernst *et al.* 04] A.T.Ernst, H.Jiang, M.Krishnamoothy, D.Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research* vol. 153, pages 3-27, 2004

[Etschmaier *et al*. 85] M.M. Etschmaier, Dennis F.X. Mathaisel. Airline scheduling: an overview. *Transportation Science* vol. 19, pages 127-138, 1985

[Fortet 60] R.Fortet. Applications de l'algèbre de boole en recherche opérationnelle. *Revue Française de Recherche Opérationnelle*, vol.4, pages 17-26, 1960

[Gamache *et al*.99] M.Gamache, F.Soumis, G.Marquis : A Column Generation Approach for Large-Scale Aircrew Rostering Problems. *Operations Research*, vol. 47, no. 2, pages 247-263, 1999.

[Garey & Johnson 85] M.R.Garey, D.S.Johnson : A 71/60 theorem for bin-packing. *Journal of Complexity*, vol. 1, pages 65-106, 1985

[Gilmore & Gomory, 61] P.C.Gilmore, R.E.Gomory. A Linear Programming Approach to the Cutting Stock Problem. *Operations Research*, vol.9, pages 849-859, 1961

[Glover & McMillann 86] F.Glover, C.McMillann. The general employee scheduling problem: an integration of MS and AI. *Computers & Operations Research*, vol. 13, pages 563-573, 1986

[Gondran & Minoux 95] M.Gondran, M.Minoux: *Graphes et Algorithmes*. Eyrolles, 1995.

[Grossman et al. 99] T. Grossman, D.Samuelson, S.Oh, T.Rohleder : Call Centers. Technical Report, Haskayne School of Business, U. Calgary, 1999

[Guéret *et al.* 00] C. Guéret, C. Prins et M. Sevaux, *Programmation Linéaire.* Eyrolles, 2000

[Guerinik *et al.* 95] N.Guerinik, M. van Canaghem: Solving crew scheduling problems by constraint programming. *Lecture Notes in Computer Science* – Proceedings of the 1st International Conference on Principles and Practice of Constraint Programming, pages 481- 498, 1995

[Henderson & Berry 76] W.B. Henderson, W.L. Berry: Heuristic methods for telephone operator shift scheduling. An experimental analysis. *Management Science*, vol. 22, pages 1372-1380, 1976

[Hennet 97] J.C.Hennet : De l'allocation optimale aux règles d'allocation en temps réel. Concepts et outils pour les systèmes de production, Cépaduès Editions, Coordination : JC.Hennet, 1997, pp.69-85

[Hennet *et al.* 03] J.C. Hennet, C.V. Draghici, F. Py, Génération de Programmes de Travail par un Algorithme de Listes. *Actes ROADEF 2003*, Rapport LAAS N°03494.

[Jacques 93] P.Jacques: Solving a manpower planning problem using constraint programming. *Belgian Journal of Operations Research, Statistics and Computer Science*, vol. 33, no. 4, pages 77-85, 1993

[Jacquet-Lagrèze *et al.* 98] E.Jacquet-Lagrèze, D.Montaut, A. Partouche: The Shift scheduling problem : Different formulations and solution methods. *Foundations of Computing and Decision Sciences*, vol.23, no.4, pages 199-207, 1998

[Johnson 74] D.S.Johnson: Fast algorithms for bin packing. *Journal of Computers and System Sciences*, vol. 8, pages 272-314, 1974

[Johnson *et al.* 74] D.S.Johnson, AZ.Demers, J.D.Ullman, M.R.Garey, R.L.Graham: Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal of Computing*, vol. 3, pages 256-278, 1974

[Keith 79] E.Keith: Operator Scheduling. *AIIE Transactions* vol. 11, pages 37-41, 1979

[Kuhn 55] H.W.Kuhn: The Hungarian Algorithm for the assignment problem. *Naval Research Logistic Quarterly*, vol. 2, pages 83-97, 1955

[Lasdon 70] L.S.Lasdon: Optimisation Theory for Large Systems, Macmillan Series in Operations Research, 1970

[Lavoie *et al.* 88] S. Lavoie, M. Minoux, E. Odier. A new approach for crew pairing problems by column generation with an application to air transportation. *EJOR* vol. 35, pages 45-58, 1988

[Loucks & Jacobs 91] J.S. Loucks, F.R.Jacobs: Tour scheduling and task assignment of a heterogenous work force: A heuristic approach. *Decision Science*, vol. 22, no. 4, pages 719-738, 1991

[Mancel 04] C.Mancel. Modélisation et résolution de problèmes d'optimisation combinatoire issus d'applications spatiales. Thèse de doctorat, Institut National de Sciences Appliquées de Toulouse, 2004

[Martello & Toth 86] S.Martello P.Toth: A heuristic approach to the bus driver scheduling problem. *European Journal of Operational Research*, vol. 24, pages 106-117, 1986

[Mason *et al.*, 98] A.J.Mason, D.M.Ryan, D.M.Panton. Integrated Simulation, Heuristic and Optimisation Approaches to Staff Scheduling. *Operations Research*, vol. 46, no.2, pages 161-175, 1998

[McDermott 82] D.V.McDermott: A Temporal Logic for Reasoning about Processes and Plans. *Cognitive Science*, vol. 6, pages 101-155, 1982

[McGinnis *et al*. 78] L.F.McGinnis, W.D.Culver, R.H.Deane : One- and two-phase heuristics for workforce scheduling. *Computers and Industrial Engineering*, vol. 2, pages 7-15, 1978

[Merle et al. 99] O. Du Merle, D. Villeneuve, J. Desrosiers & P. Hansen. Stabilized column generation. Discrete Mathematics, vol. 194, pages 229–237, 1999.

[Morris & Showalter 83] J.G.Morris, M.J.Showalter: Simple approaches to shift, days-off and tour scheduling problems. *Management Science* vol. 29, no. 8, pages 942-950, 1983

[Panton 91] D. Panton. On the creation of multiple shift continuous operation rosters under general workforce conditions. *Asia Pacific Journal of Operations Research*. vol.8, pages 189-210, 1991

[Partouche 98] A.Partouche : Planification d'horaires de travail. Méthodologie, modélisation et résolution à l'aide de la Programmation Linéaire en Nombres Entiers et de la Programmation par Contraintes. Thèse de doctorat, Université Paris-Dauphine, Paris, 1998

[Prins 94] C.Prins. *Algorithmes de graphes avec programmes en Pascal*. Eyrolles 1994

[Rémi-Robert, 03] A.J. Rémi-Robert : Systèmes interactifs d'aide à l'élaboration de plannings de travail de personnel. Thèse de doctorat, Université Joseph Fourier, Grenoble, 2003

[Ribeiro & Soumis 94] C.C. Ribeiro & F. Soumis. A column generation approach to the multiple-depot vehicle scheduling problem. Operations Research, vol. 42, no. 1, pages 41–52, 1994.

[Roberts & Escudero, 83] S.M.Roberts & L.F.Escudero. Minimum problem-size formulation for the scheduling of plant maintenance personnel. *Journal of Optimisation Theory and Applications*, vol. 39, nro.3, pages 345-362, 1983

[Rubin 73] J.Rubin : A Technique for the Solution of Massive Set Covering Problems, with Application to Airline Crew Scheduling. *Transportation Science*, vol. 7, pages 34-48, 1973

[Rushmeier *et al*, 95] R.Rushmeier, K.Hoffman, M.Padberg. Recent advances in exact optimisation of airline scheduling problems. *Technical Report*, Department of Operations Research and Operations Engineering, George Mason University, 1995.

[Ryan 80] D.M.Ryan. Zip – a zero-one integer programming package for scheduling. Technical Report, *Report C.S.S. 85*, AERE Harwall, Oxfordshire

[Ryan 92] D.M.Ryan: The Solution of Massive Generalised Set Partitioning Problems in Aircrew Rostering. *Journal of Operational Research Society*, vol. 43, no. 5, pages 459-467, 1992.

[Ryan & Falkner, 88] D.M.Ryan, J.K.Falkner. On the integer properties of scheduling set partitioning models. *European Journal of Operational Research* vol. 35, pages 442-456, 1988

[Salkin 75] H.M.Salkin. Integer Programming, Addison-Wesley, 1975

[Sarin & Aggarwal 01] S.C. Sarin, S. Aggarwal. Modeling and Algorithmic development of a staff scheduling problem. *European Journal of Operational Research*, vol. 128, pages 558-569, 2001

[Savelsberg 97] M. Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, vol. 45, no. 6, pages 831–841, 1997.

[Segal 74] M.Segal: The operator-scheduling problem: A network flow approach. *Operations Research*, vol. 22, pages 808-823, 1974

[Shoham 87] Y. Shoham: Temporal logics in AI: Semantical and Ontological Considerations. *Artificial Intelligence*, vol. 33, pages 89-104, 1987

[Thompson 88] G.M. Thompson: A comparison of techniques for scheduling non-homogeneous employees in a service environment subject to non-cyclical demand. Ph.D. Thesis, Florida State University, 1988

[Van Den Akker et al. 00] J.M. Van Den Akker, C.A.J. Hurkens & M.W.P. Savelsbergh. Time-indexed formulations for machine scheduling problems : column generation. *Informs Journal on Computing*, vol. 12, no. 2, pages 111–125, 2000.

[Vanderbeck 00] F. Vanderbeck. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. Operations Research, vol. 48, no. 1, pages 111–128, 2000.

[Vanderbeck & Wolsey 96] F. Vanderbeck & L.A. Wolsey. An exact algorithm for IP column generation. Operations research letters, vol. 19, pages 151–159, 1996.

[Vidal 95] T. Vidal : Le Temps en Planification et en Ordonnancement, Thèse de Doctorat UPS, Toulouse, 1995

[Vila 94] L.Vila: A Survey on Temporal Reasoning in Artificial Intelligence. AI Communications, vol. 7, no. 1, pages 4-28, 1994

[Weil *et al*. 98] G.Weil, K.Heus, P.Chan, P.François. The nurse scheduling problem: a combinatorial problem solved by the combination of the constraint programming and real user's heuristics. Medinfo'98. B Cesnik et al. (Eds.) IOS Press, Amsterdam; pages 508-512, 1998

[Yan *et al*. 02] S. Yan, T.-T. Tung, Y.-P. Tu. Optimal construction of airline individual crew pairings. *Computer & Operations Research* vol. 29, pages 341-363, 2002

[Yao 80] A.C.-C.Yao. New algorithms for bin-packing. *Journal of the ACM*, vol. 27, no. 2, pages 207-227, 1980