Titre: Title:	Agrégation dynamique de contrainte pour la construction de blocs mensuels personnalisés dans un contexte d'équité
Auteur: Author:	Jean-Philippe Nantel
Date:	2009
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Nantel, J.-P. (2009). Agrégation dynamique de contrainte pour la construction de blocs mensuels personnalisés dans un contexte d'équité [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8307/

UNIVERSITÉ DE MONTRÉAL

AGRÉGATION DYNAMIQUE DE CONTRAINTES POUR LA CONSTRUCTION DE BLOCS MENSUELS PERSONNALISÉS DANS UN CONTEXTE D'ÉQUITÉ

JEAN-PHILIPPE NANTEL
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
Avril 2009

Your file Votre référence
ISBN: 978-0-494-49441-7
Our file Notre référence
ISBN: 978-0-494-49441-7

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

AGRÉGATION DYNAMIQUE DE CONTRAINTES POUR LA CONSTRUCTION DE BLOCS MENSUELS PERSONNALISÉS DANS UN CONTEXTE D'ÉQUITÉ

présenté par : NANTEL Jean-Philippe
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. LANGEVIN André, Ph.D., président
M. DESAULNIERS Guy, Ph.D., membre et directeur de recherche
M. HERTZ Alain, Doct. ès sc., membre et codirecteur de recherche
M. DESROSIERS Jacques, Ph.D., membre

À mon père pour ses encouragements,
À ma mère pour son espoir,
je dédie l'intégralité de ce travail comme
témoignage de leur réussite.

À ma famille,
à ma conjointe, à ma nièce
et à tous ceux qui m'ont supporté,
que ce travail soit l'expression de toute ma reconnaissance.

REMERCIEMENTS

Je tiens tout d'abord à remercier André Langevin pour ses précieux conseils, son aide et sans qui ce travail n'aurait pas été possible.

Je veux aussi communiquer mes plus sincères remerciements à mon directeur de recherche Guy Desaulniers pour m'avoir guidé de manière brillante tout au long de ce projet, pour sa compréhension hors du commun de mes idées. J'en fais tout autant pour mon codirecteur de recherche Alain Hertz pour ses idées géniales et ses commentaires constructifs. Je dois en très grande partie le succès de ce travail à mon directeur et à mon codirecteur de recherche. Je tiens aussi à dire merci à Benoît Rochefort et à Issmail Elhallaoui pour leur contribution indispensable à ce travail.

Je veux aussi remercier ma conjointe pour son support moral tout au long de mon travail, sans lequel je n'aurais pu y parvenir.

Je remercie finalement tous les gens de mon entourage qui ont su s'intéresser à mon travail et qui m'ont encouragé à poursuivre ; ils ont contribué grandement à la réalisation de ce projet.

À tous ceux qui ont rendu ce travail possible, je leur communique ma plus sincère reconnaissance et une gratitude infinie.

RÉSUMÉ

Les compagnies de transport aérien font face à des problèmes de logistique difficiles. Une partie de ces problèmes concerne la gestion des ressources humaines et, plus particulièrement, la gestion des horaires (blocs) des pilotes. Le problème de construction des blocs mensuels personnalisés avec équité (CBMPE) consiste à affecter des tâches à des personnes en s'assurant que toutes les tâches soient couvertes et que l'horaire de chacun des employés respecte ses choix personnels et les normes en vigueur. De plus, les horaires doivent être les plus uniformes possible pour les heures de travail payées (crédits) et les jours de congé octroyés. Ce mémoire traite de la résolution du problème CBMPE.

Une méthode de résolution à la fine pointe de la technologie, l'agrégation dynamique de contraintes, a récemment émergé et a permis des gains spectaculaires en temps de calculs pour des problèmes en transport urbain, plus précisément pour la construction des horaires de conducteurs d'autobus. Dans ce mémoire, nous chercherons à appliquer une version heuristique de l'agrégation dynamique de contraintes au problème CBMPE.

Ainsi, pour démarrer la méthode, on a besoin d'une solution initiale respectant toutes les contraintes du problème. Pour que le processus dans son ensemble demeure efficace, très peu de temps doit être consacré à la recherche de cette solution. À cet effet, nous développons une métaheuristique de type recherche taboue auto-ajustée qui a été choisie en raison d'applications précédentes réussies.

Par la suite, on formule le problème CBMPE comme un problème de partitionnement d'ensemble. Il est possible d'associer des variables du problème de partitionnement d'ensemble aux chemins d'un problème de plus court chemin avec contraintes de ressources dans des réseaux acycliques. Puisque la fonction objectif engendre une

non-linéarité au niveau du sous-problème on aura recours à une approximation. On en propose deux types soit d'une part en utilisant une fonction escalier et d'autre part en utilisant une fonction tronquée. Par ailleurs, on utilise habituellement une méthode heuristique de séparation et génération progressive (SGP) (*branch-and-price*) pour résoudre de tels problèmes issus du transport aérien, qui consiste à évaluer par génération de colonnes les bornes inférieures des noeuds de branchement de la méthode de séparation et évaluation progressive (*branch-and-bound*). On propose donc de remplacer la méthode de génération de colonnes par la méthode d'agrégation dynamique de contraintes multi-phases en SGP.

On compare donc les résultats obtenus par des algorithmes heuristiques de génération de colonnes et d'agrégation dynamique de contraintes et on observe que l'agrégation dynamique permet d'importants gains autant du point de vue des temps de calculs que de la qualité des solutions. En effet, en plus d'avoir des solutions ayant un écart-type sur les heures de travail créditées jusqu'à 35% plus petit, on peut réduire les temps de calcul par un facteur trois. La méthode de branchement heuristique explique ces différence dans les qualités de solution. De plus, en comparant les résultats des deux types de formulation on observe que le modèle en escalier permet d'obtenir de meilleurs résultats en terme d'écart-type des crédits et des jours de congés que le modèle avec la fonction tronquée pour des temps de calcul comparables.

ABSTRACT

Civil airlines face huge logistic problems and one of them is the construction of pilot schedules. The personalized bidline scheduling problem with equity (PBSPE) consists in assigning tasks to pilots while making sure that all tasks are covered and that each person's schedule (bidline) respects constraints imposed by safety regulations and collective agreement rules. These bidlines must also respect personal crew preferences and must be as uniform as possible for credited (paid) hours and days off. This master's thesis addresses the PBSPE.

A state-of-the-art solution method named dynamic constraint aggregation recently emerged, allowing spectacular decrease of computational times over column generation for a bus driver scheduling problem. In this thesis, we want to apply a heuristic version of dynamic constraint aggregation to PBSPE.

A feasible solution is needed to start the solution process and a small amount of time must be spent to obtain it in order to have a globally efficient solution process. To do so, we develop a metaheuristic based on a self-tuned tabu search method that has been selected because of earlier successful applications on similar problems.

Thereafter, the PBSPE is formulated as a set-partitioning problem. It is possible to associate variables of this problem to paths of a resource constrained shortest path problem in an acyclic graph. Since the objective function induces non-linearities in the subproblem, an approximative function is used. Two types of approximations are used, one is a step function and the other is a truncated function. Furthemore, a heuristic branch-and-price method is often used to solve such problems from air transport. Such a method proposes to evaluate the lower bounds at the nodes of the branch-and-bound search tree by a column generation method. We then propose to replace column generation by dynamic constraint agregation.

When we compare the results obtained with heuristic column generation and dynamic constraint aggregation algorithms, one can observe that the latter can significantly improve computational times and solution quality. Indeed, the computed solutions had a up to 35% lower standard deviation of the credited hours while the computational times were reduced by a factor of up to three. The heuristice branching method used is responsible for the difference in quality of the computed solutions. Moreover, when we compare the two approximate models, one can see the the model with the step function improves solution quality over the model with a truncated function with a similar computation time.

TABLE DES MATIÈRES

DÉDICACE iv

REMERCIEMENTS v

RÉSUMÉ vi

ABSTRACT viii

TABLE DES MATIÈRES x

LISTE DES TABLEAUX xv

LISTE DES FIGURES xvi

LISTE DES ALGORITHMES xvii

CHAPITRE 1 : INTRODUCTION 1

1.1 : Terminologie 2

1.2 : Vue d'ensemble de la planification des opérations en transport aérien 4

1.2.1 : Planification des vols 4

1.2.2 : Assignation de la flotte 4

1.2.3 : Construction des rotations d'avion 5

1.2.4 : Construction des rotations d'équipage 5

1.2.5 : Construction des horaires mensuels 6

1.2.6 : Gestion des perturbations 7

1.3 : Définition détaillée du problème de construction des blocs mensuels personnalisés avec équité 7

1.3.1 : Contraintes locales 8

1.3.2 : Contraintes globales 9

1.3.3 : Fonction objectif 9

1.4 : Objectifs du mémoire 9

CHAPITRE 2 : REVUE DE LA LITTÉRATURE 11

2.1 : Modes de construction des horaires mensuels 11

2.1.1 : Bidline 11

2.1.2 : Rostering 12

2.1.3 : Preferential Bidding 12

2.2 : Méthodes de résolution 13

2.2.1 : Métaheuristiques 14

2.2.2 : Modèle de partitionnement d'ensemble 15

2.2.3 : Approche par lignes 16

2.2.4 : Approche par colonnes 17

2.2.5 : Approches réseau 19

CHAPITRE 3 : ALGORITHME DE RECHERCHE TABOUE . . . 24

3.1 : Coloration de graphe 24

3.1.1 : Heuristiques 26

3.2 : Algorithme utilisé 30

3.2.1 : Description générale de la méthode 31

3.2.2 : Techniques d'implantation 32

3.2.3 : Approche générale 33

3.2.4 : Espace des solutions 34

3.2.5 : Fonction objectif 35

3.2.6 : Calcul par incrément de la fonction objectif 40

3.2.7 : Voisinages 43

3.2.8 : Solution initiale 46

3.2.9 : Liste taboue 47

CHAPITRE 4 : GÉNÉRATION DE COLONNES ET AGRÉGATION DYNAMIQUE DE CONTRAINTES 48

4.1 : Modélisation du problème 48

4.1.1 : Une approximation du modèle 50

4.2 : Génération de colonnes 54

4.2.1 : Modélisation du sous-problème 55

4.2.2 : Réseau personnalisé 64

4.2.3 : Calcul des bornes pour les niveaux de flexibilité 64

4.2.4 : Résolution des sous-problèmes 66

4.2.5 : Modèle avec la fonction objectif tronquée 68

4.3 : Agrégation dynamique de contraintes 69

4.3.1 : Introduction 69

4.3.2 : Description de la méthode 71

4.3.3 : Application de MPDCA au CBMPE 75

CHAPITRE 5 : EXPÉRIMENTATIONS NUMÉRIQUES 79

5.1 : Instances traitées 79

5.2 : Résultats de l'algorithme tabou 81

5.3 : Résultats de la génération de colonnes 84

5.3.1 : Modèle avec fonction en escalier 84

5.3.2 : Modèle avec fonction tronquée 86

5.4 : Résultats de l'agrégation dynamique 87

5.5 : Conclusion 88

CONCLUSION 89

BIBLIOGRAPHIE 91

LISTE DES TABLEAUX

Tableau 3.1 : Définition des voisinages 45

Tableau 4.1 : Consommation des ressources 63

Tableau 4.2 : Plage de validité des ressources 63

Tableau 5.1 : Caractéristiques des instances 82

Tableau 5.2 : Résultats de la recherche taboue 82

Tableau 5.3 : Qualité de solution pour la recherche taboue 83

Tableau 5.4 : Statistiques du modèle avec fonction escalier 85

Tableau 5.5 : Qualité des solutions pour le modèle avec fonction escalier 86

Tableau 5.6 : Qualité des solutions du modèle avec fonction tronquée . . 86

Tableau 5.7 : Statistiques pour MPDCA 87

Tableau 5.8 : Qualité des solutions de MPDCA 88

LISTE DES FIGURES

Figure 3.1 : Illustration d'un chevauchement 35

Figure 3.2 : Illustration de plusieurs chevauchements 36

Figure 4.1 : Illustration des niveaux de flexibilité 52

Figure 4.2 : Illustration de la fonction objectif 53

Figure 4.3 : Algorithme de séparation et génération progressive 56

Figure 4.4 : Exemple de réseau anonyme utilisé en génération de colonnes 59

Figure 4.5 : Exemple de réseau personnalisé utilisé en génération de colonnes 64

Figure 4.6 : Algorithme MPDCA 73

Figure 4.7 : Exemple de réseau utilisé pour l'agrégation dynamique . . . 78

LISTE DES ALGORITHMES

3.1 : Schéma général d'un algorithme de recherche taboue 32
3.2 : Ajustement dynamique des coefficients de la fonction objectif 40

CHAPITRE 1
INTRODUCTION

D'une part, la conjoncture économique actuelle fait en sorte que beaucoup de pression est exercée sur les entreprises pour qu'elles aient une rentabilité toujours plus grande. Pour les entreprises de certains secteurs d'activités, la masse salariale représente une des plus grandes dépenses et constitue une cible de choix dans un contexte de rationalisation. D'autre part, les horaires des employés doivent souvent respecter des règles très strictes et ce, particulièrement dans le domaine des transports, où les planificateurs sont soumis à des demandes ponctuelles disséminées tout au long d'une période de planification. Aussitôt que le nombre d'employés et de tâches à affecter devient un peu grand, une telle planification sans outil informatique devient ardue, voire même impossible. Un outil permettant de produire automatiquement et rapidement des horaires de personnel respectant les règles applicables et ce, à moindre coût, s'avère très utile et même nécessaire. Un tel outil s'attaque donc à deux problématiques importantes pour les entreprises, soit la simplification de la tâche de planification des horaires et la rationalisation de la masse salariale des employés.

Ce mémoire traite du problème de construction des blocs mensuels personnalisés (CBMPE) pour des compagnies aériennes civiles et l'aspect nouveau est de s'attaquer au problème où les employés ne sont pas considérés comme anonymes, à l'aide d'une approche de résolution relativement nouvelle, l'agrégation dynamique de contraintes.

Pour appliquer l'agrégation dynamique de contraintes pour la construction de blocs mensuels personnalisés, plusieurs étapes sont nécessaires et seront présentées dans ce mémoire. D'autres parties s'ajouteront aussi pour aider un lecteur qui est peu familier avec le contexte. Ainsi, le présent chapitre traite du contexte du problème décrivant la nature exacte du problème résolu et de l'endroit où il se situe dans la chaîne logistique de la planification des opérations en transport aérien. Le

deuxième chapitre présentera une brève revue de la littérature sur le domaine et une présentation du contexte dans lequel s'inscrit l'agrégation dynamique. Le troisième chapitre présente la méthode utilisée pour trouver une solution initiale rapidement. Le chapitre quatre présentera l'application de l'agrégation dynamique et le dernier chapitre présentera les résultats expérimentaux obtenus pour toutes les méthodes de résolution utilisées. On conclura finalement en présentant des avenues intéressantes pour une éventuelle poursuite des travaux.

Ce travail reprend partiellement les travaux de Boubaker (2006) et utilise en grande partie les mêmes données, mais le problème traité ici est plus général que celui de Boubaker (2006) qui en est un cas particulier. Par ailleurs, le chapitre débutera d'abord par une description des termes utilisés tout au long de ce travail, ce qui permettra ensuite de présenter une vue d'ensemble des opérations de planification en transport aérien. Cette partie situera le problème à résoudre par rapport aux autres opérations de planification afin de bien introduire la description détaillée du problème. Viendront ensuite les sections sur les objectifs poursuivis par ce mémoire et la structure de ce dernier pour donner une vue d'ensemble du travail au lecteur.

1.1 Terminologie

La terminologie suivante sera utilisée tout au long de ce mémoire.

- **Crédits de vol** : Il s'agit de la mesure de la quantité de travail pour le personnel travaillant à bord des avions.
- **Station** : Une station est un aéroport donné.
- **Base** : Une base est une station à laquelle est affecté un employé.
- **Segment de vol** : Un segment de vol est un vol sans escale.
- **Connexion** : Période entre deux segments de vol consécutifs dans la même journée, pour un même employé.

- **Repos** : Nuit entre deux segments de vol hors-base, pour un employé.
- **Rotation** (*Pairing*) : Une rotation est composée de segments de vol ainsi que de connexions et peut comprendre des périodes de repos. Une rotation commence à une base et se termine à la même base.
- **Repos post-courrier** : Après chaque rotation, le personnel a droit à une période de repos post-courrier d'une durée typique de 12 heures qui ne peut être écourtée.
- **Jour travaillé** : Une journée travaillée est comptée de minuit à minuit. Par conséquent, aussitôt qu'une rotation ou une tâche préassignée touche à un jour donné, celui-ci est considéré comme travaillé. Par exemple, il est possible qu'une rotation durant 26 heures compte pour trois jours travaillés.
- **Jour de congé** : Un jour de congé est un intervalle de temps de minuit à minuit non touché par une tâche préassignée, par une période de vacances ou par une rotation.
- **Vacances** : Un jour de vacances est une journée non travaillée attribuée à un employé en particulier avant l'optimisation. Une période de vacances ne compte pas comme congé, mais permet de séparer deux séries de jours travaillés consécutifs. Pour chaque jour de vacances, l'employé se voit créditer un certain nombre de crédits et un certain nombre de jours de congé.
- **Horizon de planification** : Période pour laquelle on désire optimiser l'affectation de l'ensemble des tâches chevauchant cet intervalle de temps.
- **Tâche préassignée** (*Carry-in*) : Tâche résultant de l'affectation d'une rotation à une personne lors d'une optimisation précédente pour un horizon de planification antérieur.
- **Flotte** : Une flotte est un ensemble d'appareils du même type possédés par une compagnie de transport aérien. Une compagnie possède typiquement plusieurs flottes d'appareils ; par exemple, une flotte de Boeing 737, une flotte d'Airbus 320, etc.

1.2 Vue d'ensemble de la planification des opérations en transport aérien

L'objectif de cette section est de dresser un portrait sommaire des opérations de planification en transport aérien pour avoir une idée du contexte dans lequel se situe la construction d'horaires de personnel. Tout d'abord, l'industrie du transport aérien faisait face à un problème de logistique impossible à résoudre dans des délais acceptables. Celui-ci a donc été décomposé (Medard et Sawhney, 2007; Desaulniers *et al.*, 1997; Klabjan, 2005; Barnhart *et al.*, 2003; Kohl *et al.*, 2004) en plusieurs problèmes de manière à obtenir rapidement d'excellentes solutions, mais globalement non optimales.

1.2.1 Planification des vols

Cette étape consiste à dresser une liste exhaustive des vols que la compagnie a l'intention d'offrir, principalement en fonction de ses parts de marché et de celles de ses concurrents, des prévisions de la demande et des capacités des différentes infrastructures aéroportuaires, de manière à maximiser son profit. Cette étape peut être considérée comme une planification stratégique et est souvent sous la responsabilité du département de marketing.

1.2.2 Assignation de la flotte

À la fin de l'étape précédente, on obtient une liste de vols à couvrir avec les flottes d'avions disponibles. Il faut donc assigner à chaque vol un type de flotte ou plutôt un type d'avions de manière à maximiser les profits. Il faut aussi prendre en considération certaines restrictions comme l'impossibilité de desservir certains vols avec certains types d'avions. La taille de la flotte doit aussi être prise en considération pour ne

pas affecter simultanément plus d'avions que le nombre disponible. Finalement, on doit s'assurer du respect des contraintes de conservation de flot d'avion à chacune des stations et en tout temps. Si cette vérification n'est pas faite, le problème de construction des rotations d'avion ne sera pas réalisable.

1.2.3 Construction des rotations d'avion

Cette étape consiste à construire des itinéraires pour chacun des avions de chacune des flottes en y incluant les périodes de maintenance nécessaires. Le problème de construction des rotations d'avion est séparable par type de flotte et il n'y a aucun gain possible à résoudre le problème pour toutes les flottes simultanément, si on omet la mobilisation de ressources, qui sont limitées, pour la maintenance. La construction des rotations doit tenir compte des opérations périodiques de maintenance sur les avions qui ne peuvent être retardées et doit évidemment prendre en considération l'affectation de la flotte obtenue au préalable et l'ensemble des chemins doit couvrir tous les vols.

1.2.4 Construction des rotations d'équipage

Il faut, lors de cette étape, construire des rotations qui sont des séquences de vols commençant et se terminant à une même base. Il faut évidemment que tous les vols offerts soient couverts par des rotations. Les rotations d'équipage ne sont pas nécessairement les mêmes que celles des avions puisque les contraintes sont bien différentes et c'est pourquoi les problèmes sont distincts. Les rotations ainsi construites doivent satisfaire les normes et la convention collective en vigueur. Le transporteur ajoute habituellement, lui aussi, des contraintes introduisant des écarts de manière à avancer le début et à repousser la fin d'une tâche pour obtenir un horaire plus robuste. Pour le personnel naviguant, le problème est séparable par type d'avions ou par ensemble

d'avions très similaires et par famille d'avions, car il n'est pas permis pour un employé de changer de type d'avions sans effectuer au préalable une longue formation qui a habituellement une durée plus grande que celle de l'horizon de planification. Les rotations d'équipage ont tendance à suivre les rotations d'avions puisque les temps de connexion sont moins longs, donc moins coûteux, lorsque l'équipage n'a pas besoin de changer d'appareil.

1.2.5 Construction des horaires mensuels

Ce problème vise l'élaboration d'un horaire de travail pour l'équipage de la cabine de pilotage par partitionnement de l'ensemble des rotations, en un nombre de sous-ensembles équivalent au nombre d'employés. L'affectation des rotations doit être faite en respect des normes en vigueur, de la convention collective des employés et des politiques de la compagnie. Ce problème est séparable par flotte d'avions et par base. Ainsi, un des problèmes à résoudre pourrait être l'affectation des rotations pour les employés de la base de Montréal pour les Airbus A320. Il existe principalement trois paradigmes pour la construction des horaires de personnel aérien soit, le *bidline*, le *rostering* et le *preferential bidding*. Le *bidline* consiste en une construction totalement anonyme des horaires et l'affectation est faite ultérieurement principalement en fonction de l'ancienneté des employés. Quant à lui, le *rostering* est un mode de construction personnalisé et tient compte des préférences des individus. En effet, il est possible d'évaluer la satisfaction d'un employé pour un horaire en particulier et il s'agit de maximiser les préférences de tous les individus ou plutôt la somme des préférences de chacun. Par ailleurs, le *preferential bidding* s'apparente beaucoup au *rostering*, mais le système d'attribution des préférences est plus sophistiqué et l'affectation des horaires priorise les employés ayant plus d'ancienneté. Les modes de constructions *bidline* et *rostering* seront décrits plus en détails dans le chapitre portant sur la revue de littérature.

1.2.6 Gestion des perturbations

À l'instar des autres étapes de planification qui s'approchent plus du niveau tactique, hormis la planification des vols qui relève de la planification stratégique, cette étape en est une d'ordre opérationnel. Cette étape vise à « réparer » les planifications déjà faites lorsque surviennent des imprévus tels que des bris d'équipement, des conditions météorologiques peu clémentes ou des indispositions du personnel. C'est à cette étape que les écarts introduits préalablement deviennent utiles et permettent la construction de plans de rechange à moindre coût. Cette étape de planification est donc essentielle pour maintenir les coûts opérationnels relativement bas étant donné que les planifications des étapes précédentes ne sont jamais respectées dans leur intégralité.

1.3 Définition détaillée du problème de construction des blocs mensuels personnalisés avec équité

Comme on l'a décrit brièvement précédemment, le problème de construction des blocs mensuels vise à assigner à chaque employé un certain nombre de rotations d'équipage. L'affectation de ces rotations doit être faite en considérant les tâches préassignées et les vacances. Les horaires des employés sont soumis à des règles locales et globales qui doivent absolument être respectées. Les trois paradigmes de construction des horaires influencent, quant à eux, principalement la fonction à optimiser. Comme on l'a déjà mentionné, le problème est séparable par type d'avions et par base, mais il l'est aussi par type de personnel. Cependant, malgré cette décomposition, la plupart des problèmes issus de cette dernière demeurent de très grande taille, et par conséquent, difficiles à résoudre.

Le problème CBMPE s'incrit dans un contexte de *bidline personnalisé* qui est un paradigme où une partie des employés ont des exigences particulières et où les autres

employés sont traités de façon anonyme. Pour ce problème, les règles auxquelles on a fait référence précédemment sont relativement facilement formulables sous forme de contraintes et on peut les rassembler en deux familles, soit les contraintes locales et globales. Pour terminer la description du problème on doit aussi présenter la fonction à optimiser.

1.3.1 Contraintes locales

Ces contraintes sont appliquées de manière indépendante à chaque horaire ou à chacun des employés et peuvent varier selon la compagnie aérienne. Voici donc les contraintes locales qui seront appliquées à chacun des horaires dans le cadre de ce mémoire.

- **Crédits** : Le nombre de crédits assignés à un employé doit être compris entre une valeur minimale et maximale.
- **Chevauchements** : Deux tâches préassignées, rotations ou vacances ne peuvent être affectées à la même personne si l'une d'elles se déroule en même temps que l'autre. Un repos post-courrier doit être considéré dans l'évaluation des chevauchements pour les rotations et les tâches préassignées.
- **Jours de congé** : Le nombre de jours de congé attribués à un employé doit être supérieur ou égal à une valeur seuil.
- **Jours de travail consécutifs** : Le nombre de jours travaillés consécutifs ne doit jamais dépasser une certaine valeur seuil. Il est important de noter que les vacances et les journées de congé peuvent mettre fin à une série de jours travaillés consécutivement.
- **Personnalisation** : On doit inclure à l'horaire de l'employé la tâche préassignée et la période de vacances qui lui est assignée. Cette contrainte n'est pas présente pour tous les employés.

1.3.2 Contraintes globales

Les contraintes globales régissent l'ensemble des horaires des employés et se regroupent en deux catégories. Voici donc les contraintes globales qui seront appliquées à l'ensemble des horaires dans le cadre de ce mémoire.

- **Partitionnement** : Chaque tâche doit être effectuée par un nombre suffisant d'employés (un seul employé pour le problème traité ici).
- **Disponibilité** : On doit attribuer un horaire à chaque employé c'est-à-dire que les rotations doivent être partitionnées selon le nombre d'employés disponibles.

1.3.3 Fonction objectif

L'objectif du problème est la construction d'un horaire pour chacun des employés de manière à ce que chacune des contraintes soient satisfaites. À cela peut s'ajouter d'autres objectifs complémentaires qui mesurent en général le niveau de satisfaction des employés. La satisfaction de ceux-ci est mesurée différemment selon le paradigme d'évaluation utilisé. Dans le cas du problème traité dans ce mémoire, l'objectif sera de minimiser la somme pondérée des écarts-types du nombre de crédits et du nombre de jours de congé par horaire. Cette manière de procéder s'inscrit donc dans un contexte d'équité qui sera détaillé dans le prochain chapitre.

1.4 Objectifs du mémoire

Comme on l'a mentionné précédemment, même décomposé, le problème d'élaboration des horaires mensuels demeure de très grande taille. Ce dernier doit aussi être résolu dans un contexte d'équité où les employés ne sont pas anonymes, c'est-à-dire que certains ont des particularités et préférences qui nous empêchent de les traiter de façon anonyme. L'objectif du mémoire est donc de développer une méthode plus

rapide que celles existantes pour résoudre ce problème. Pour y parvenir, on aura recours à l'agrégation dynamique de contraintes. Cette méthode a déjà fait ses preuves pour un problème similaire notamment dans les travaux de Boubaker (2006), mais elle requiert une solution initiale réalisable. Une métaheuristique de type recherche taboue semble appropriée pour trouver une telle solution et sera développée. Par conséquent, ce mémoire contribuera à :

- Développer une métaheuristique de type recherche taboue pour le problème de construction de blocs mensuels personnalisés dans un contexte d'équité pour le personnel de cabine de pilotage de compagnies aériennes.
- Développer un modèle mathématique approprié à ce problème.
- Appliquer l'agrégation dynamique de contraintes à ce modèle.
- Comparer les performances de la génération de colonnes par rapport à l'agrégation dynamique de contraintes.

CHAPITRE 2
REVUE DE LA LITTÉRATURE

Ce chapitre est consacré à une brève revue de la littérature sur la construction d'horaires de personnel pour les compagnies de transport aérien. La première partie sera dédiée à la description des modes de construction des horaires applicables au problème traité par ce mémoire. Ensuite seront présentées plusieurs méthodes de résolution qu'il est possible d'utiliser pour résoudre le problème. Par ailleurs, beaucoup de travaux sont consacrés au problème de construction des rotations d'équipage, mais ceux sur la construction des blocs mensuels sont un peu plus rares ; le sont encore plus ceux traitant du mode de construction *bidline*. Cependant, puisque les problèmes de construction des rotations d'équipage et celui de construction des blocs mensuels peuvent se formuler sous des formes ayant certaines similitudes, il est possible d'utiliser des méthodes de résolution « propres » au problème de construction des rotations pour résoudre le problème de construction des blocs.

2.1 Modes de construction des horaires mensuels

Comme on l'a déjà mentionné, il existe trois principales tendances pour construire les horaires mensuels et en voici une description un peu plus détaillée.

2.1.1 Bidline

Ce mode de construction des horaires implique deux phases de résolution. La première consiste à construire des horaires anonymes qui couvrent toutes les rotations et la seconde consiste à affecter les horaires plus tôt construits aux individus. Lors de

la deuxième phase, les employés attribuent des cotes (*bids*) aux horaires qu'ils préfèrent et ces horaires sont ensuite affectés selon la priorité qu'ont les individus, cette dernière étant souvent basée sur l'ancienneté. Le *bidline* peut donc être considéré comme le plus simple des trois modes puisqu'il ne tient pas compte des préférences de chacun des individus lors de la résolution du problème de construction des blocs mensuels, ce qui le simplifie considérablement. Ce mode a déjà été utilisé par plusieurs compagnies, notamment American Airlines (Russell, 1989) et British Airways (Wilson, 1981).

2.1.2 Rostering

D'une part, le *rostering* fait appel à la notion de personnalisation des horaires. Contrairement au *bidline*, le *rostering* ne nécessite qu'une seule phase. Celle-ci prend donc en considération les préférences et besoins des individus ainsi que les préassignations lors de la résolution du problème. D'autre part, les individus émettent une liste de préférences servant à établir un pointage à chaque horaire, et l'objectif poursuivi par ce mode de construction est la maximisation de la satisfaction globale des employés. Ce dernier est principalement associé aux transporteurs européens comme Air France (Giaferri *et al.*, 1982; Gontier, 1985; Gamache et Soumis, 1998), Alitalia (Nicoletti, 1975; Marchettini, 1980; Sarra, 1988; Federici et Paschina, 1990), Lufthansa (Glanert, 1984), SwissAir (Tingley, 1979), mais a déjà été utilisé aussi par Air New-Zealand (Ryan, 1992) et EL-Al Israel Airline (Mayer, 1980).

2.1.3 Preferential Bidding

Le *preferential bidding* s'apparente beaucoup au *rostering*, mais l'objectif diffère un peu. En effet, les préférences des employés ayant plus d'ancienneté sont favorisées au détriment de ceux en ayant moins tout en s'assurant qu'il est possible de couvrir le

reste des rotations avec les employés encore disponibles. De plus, la définition de préférence est habituellement plus sophisitiquée dans le *preferential bidding*, c'est-à-dire qu'un grand nombre de caractéristiques d'un horaire peuvent être considérées pour établir l'affinité qu'a un employé pour un horaire en particulier. Ce mode de construction d'horaires est surtout associé aux compagnies aériennes nord-américaines et a déjà été utilisé par Canadian Pacific Airlines (Byrne, 1988), Midwest Express Airlines, Air Canada (Gamache *et al.*, 1998), mais aussi par Qantas (Moore *et al.*, 1978). De plus, une amélioration du modèle implanté chez Air Canada est parue et propose une métaheuristique en support à une méthode de génération de colonnes (Gamache *et al.*, 2007)

2.2 Méthodes de résolution

Le problème à résoudre dans le cadre de ce mémoire s'apparente au *rostering* sauf que l'on considère que les employés n'ont pas de préférences et par conséquent, la fonction à optimiser est différente. En effet, il s'agit plutôt de minimiser l'écart-type des crédits de vol et l'écart-type des jours de congé, ce qui a pour effet d'uniformiser les horaires sur le plan de la quantité de travail et de la quantité de congés. Cependant, le problème traité peut aussi s'apparenter au *bidline* puisque plusieurs employés sont toujours considérés comme étant anonymes et en conséquence, l'attribution de certains horaires doit être faite a posteriori. Certains utilisent parfois le terme *bidline personnalisé* pour décrire ce mode de construction.

Dans la section qui suit, plusieurs méthodes de résolution sont présentées, mais toutes ne concernent pas le *bidline personnalisé*. Effectivement, plusieurs approches sont destinées au problème de construction des rotations d'équipage (crew pairing), et même au problème de construction de tournées de véhicules et d'horaires de personnel (vehicle and crew scheduling problem - VCSP) dans le cas de l'agrégation dynamique de contraintes. De plus, certaines approches sont traitées de manière plus générale

sans se restreindre à une application particulière d'un auteur. Cependant, tous les éléments présentés sont potentiellement utilisables pour le problème de construction des blocs mensuels dans un contexte de *bidline personnalisé* et donnent une très bonne idée des recherches et des progrès effectués pour de tels problèmes.

Selon Gopalakrishnan et Johnson (2005), les méthodes de résolution peuvent être regroupées selon trois approches : l'approche par lignes, l'approche par colonnes et l'approche réseau. La section qui suit présente donc quelques méthodes de résolution pour ce problème regroupées selon leur type d'approche. Toutefois, certaines d'entre elles ne peuvent être apparentées à aucune des précédentes approches, notamment les métaheuristiques dont la description de quelques-unes suit.

2.2.1 Métaheuristiques

Tout d'abord, on remarque quelques utilisations de métaheuristiques pour la résolution de problèmes de construction des blocs mensuels. D'une part, Campbell *et al.* (1997) proposent une méthode de recuit simulé pour résoudre un problème multi-objectifs de minimisation des crédits non-couverts (*open time*), de minimisation des horaires générés, c'est-à-dire du nombre d'employés nécessaire et de maximisation de la qualité de vie induite par les horaires générés (*purity constraints*). Une phase de réoptimisation basée sur une heuristique gloutonne est utilisée pour redistribuer les crédits non-couverts à de nouveaux horaires en relaxant quelque peu les contraintes liées à la qualité de vie et ce, en ne modifiant pas les horaires déjà générés. Cette approche a été implantée par FedEx pour des fins d'analyse des règles régissant le travail des pilotes lors de la négociation de la convention collective de ces derniers.

D'autre part, Christou *et al.* (1999) proposent un algorithme génétique en deux phases pour résoudre un problème multi-objectifs similaire au précédent. La méthode construit donc une multitude de bons horaires selon la qualité de vie des employés à l'aide d'un mécanisme de construction de chemin dans un arbre où les noeuds

représentent des rotations. Par la suite, un algorithme génétique où les individus initiaux sont générés au hasard est utilisé pour assigner les crédits non-couverts à des horaires valides. Les individus sont évalués relativement à la population et évoluent à l'aide de croisements à un élément, d'inversions et de mutations. Les enfants en résultant sont conservés si leur valeur est bonne et remplacés par leurs parents sinon. Cependant, la plupart des individus correspondent à des solutions non-réalisables et la méthode comprend une procédure de réparation plutôt que de pénaliser les irréalisabilités qui sont trop nombreuses. L'algorithme s'arrête lorsqu'un individu correspondant à une solution de très bonne qualité est obtenu. Cette méthode a été implantée chez Delta Airlines et a permis des économies monétaires substantielles tout en maintenant le même niveau de qualité des horaires générés.

2.2.2 Modèle de partitionnement d'ensemble

Toutes les méthodes qui suivent jusqu'à la fin du chapitre sont basées sur une formulation de partitionnement d'ensemble. Le modèle général suivant tient compte des contraintes globales de partitionnement et de disponibilité. Les contraintes locales sont implicitement incluses dans le modèle par la définition de Ω_m. On utilisera les notations suivantes pour définir ce programme linéaire en variables binaires :

- Ω_m : Ensemble des horaires admissibles pour l'employé m ;
- k : Nombre d'employés ;
- v : Nombre de rotations à affecter ;
- c_j : Coût de l'horaire j ;
- a_{ij} : Paramètre binaire $= 1$ si l'horaire j couvre la rotation i, $= 0$ sinon ;
- b_i : Nombre d'employés requis pour effectuer la rotation i ;
- x_j^m : Variable binaire $= 1$ si l'horaire j est choisi pour l'employé m, $= 0$ sinon.

Le problème de construction des blocs mensuels peut se formuler ainsi :

$$\text{Minimiser} : \sum_{m=1}^{k} \sum_{j \in \Omega_m} c_j x_j^m \tag{2.1}$$

Sujet aux contraintes :

$$\sum_{m=1}^{k} \sum_{j \in \Omega_m} a_{ij} x_j^m = \quad b_i \qquad \forall i \in \{1, 2, ..., v\} \tag{2.2}$$

$$\sum_{j \in \Omega_m} x_j^m = \quad 1 \qquad \forall m \in \{1, 2, ..., k\} \tag{2.3}$$

$$x_j^m \in \{0, 1\} \qquad \forall m \in \{1, 2, ..., k\}, \forall j \in \Omega_m. \tag{2.4}$$

L'objectif (2.1) vise la minimisation des coûts des horaires choisis tandis que le bloc de contraintes (2.2) s'assure que chaque rotation soit couverte par un nombre suffisant de personnes. Le bloc de contraintes (2.3) s'assure qu'une personne ait la charge de travail correspondant à un horaire. Ces dernières contraintes combinées aux suivantes (2.4) stipulent que chaque personne doit être affectée à un seul horaire.

Cette modélisation met en évidence que l'on se retrouve vis-à-vis un problème très difficile, principalement parce qu'il est en nombres entiers et que le nombre de colonnes ou de variables x_j^m est énorme : $|\Omega_m| > 10^8$ pour un problème de taille moyenne (Barnhart *et al.*, 2003). Ceci explique donc pourquoi beaucoup de recherches ont été faites dans le but de résoudre plus efficacement ce problème et on présentera ici les principaux développements.

2.2.3 Approche par lignes

Tout d'abord, étant donné la grande taille des problèmes de construction des blocs mensuels et de construction des rotations d'équipage, il y a une vingtaine d'années, les techniques de résolution à la fine pointe de la technologie étaient des méthodes heuristiques basées sur l'approche par lignes. En effet, une des premières méthodes de résolution appliquée avec succès sur un problème de construction des rotations d'équipage consistait, à partir d'un modèle de partitionnement d'ensemble ayant une solution réalisable, à prendre un sous-ensemble de petite taille des rotations du problème. On génère ensuite toutes ou la plupart de celles qu'il est possible d'obtenir

en réarrangeant les segments de vols couverts par ces rotations. Plus précisément, on décortique l'ensemble des rotations choisies pour en obtenir tous les segments de vols couverts et on génère toutes les rotations possibles avec ces segments de vols. On choisit ensuite l'ensemble de rotations nouvellement généré de moindre de coût qui couvre tous les segments de vols et si la somme des coûts de ce nouvel ensemble est inférieur à l'original, on remplace celui-ci par le nouveau dans le problème de partitionnement d'ensemble. Puisque le nombre de rotations choisies est relativement petit, le problème obtenu l'est aussi et la relaxation linéaire d'un petit problème de partitionnement d'ensemble a tendance à être presque ou déjà entière. Il ne reste donc que très peu de travail à faire en séparation et évaluation progressive, ce qui se traduit par une résolution très rapide et qui est la principale raison du succès de cette approche. Cependant, il est difficile d'établir des critères se traduisant par un choix efficace des rotations à réarranger et le choix au hasard de ces dernières est la stratégie la plus efficace. Cette approche a été implantée chez American Airlines (Gershkoff, 1989).

Par ailleurs, plusieurs autres travaux utilisent cette approche ; mentionnons notamment Housos et Elmroth (1997); Anbil *et al.* (1991); Baker *et al.* (1985); Graves *et al.* (1993). Les travaux précédemment cités utilisent tous une méthode semblable à celle de Gershkoff (1989) qui vient d'être décrite et qui présente bien l'approche par lignes. L'avantage de cette approche est que l'on obtient une solution réalisable à chaque itération de l'algorithme, mais l'optimalité globale n'est aucunement garantie.

2.2.4 Approche par colonnes

Cette approche peut s'apparenter à l'approche par lignes sauf que lors de la génération de colonnes, toutes les lignes sont prises en considération simultanément. Cette approche peut converger vers l'optimalité ou en être très près, mais la solution optimale de la relaxation linéaire d'un problème de partitionnement d'ensemble de moyenne ou grande taille est généralement hautement fractionnaire. Ainsi, l'obtention d'une solution entière nécessite beaucoup de travail et des méthodes efficaces.

Génération a priori des colonnes

La génération a priori des colonnes est une méthode consistant principalement en deux étapes, soit une de génération des colonnes et une de résolution du problème de partitionnement d'ensemble peuplé des colonnes générées a priori. Cette méthode a notamment été utilisée par Jarrah et Diamond (1997) dans un contexte de *bidline* où il fallait minimiser les crédits non couverts. Les auteurs ont utilisé des heuristiques gloutonnes pour générer des colonnes en y incluant des règles de décision pour écarter le plus tôt possible des solutions partielles menant inévitablement à des horaires non réalisables. En effet, le générateur de colonnes peut être modélisé comme un arbre dont les noeuds correspondent à des rotations. On commence d'abord avec une rotation en début de période comme racine en ajoutant des rotations pouvant être faites subséquemment en respect des contraintes. Les branches de l'arbre sont coupées à l'aide de règles de décisions heuristiques comme une borne supérieure sur le nombre minimal de crédits atteignable par cette branche et comme des dominances sur les états ; entre autres, un noeud en domine un autre si son nombre de crédits accumulés est plus grand et que son potentiel de violation de contraintes est plus faible, c'est-à-dire qu'il risque moins de les violer. L'avantage de cette méthode est qu'il est possible pour des personnes comme les planificateurs d'horaires d'interagir avec le générateur de colonnes et de modifier manuellement les colonnes à l'aide d'interfaces d'utilisateurs avant de lancer l'optimisation. Ainsi, la méthode donne plus de flexibilité à l'utilisateur et permet de mesurer rapidement l'impact de certaines décisions puisque le problème de partitionnement d'ensemble n'est pas trop long à résoudre. Cependant, pour que cette méthode soit considérée comme exacte, toutes les colonnes ou horaires possibles doivent être générés, ce qui peut s'avérer long, rendant par le fait même le problème de partitionnement d'ensemble très gros et, par conséquent, très long à résoudre.

Programmation par contraintes

Cette approche utilise la programmation par contraintes pour résoudre le sous-problème plutôt que la programmation dynamique, plus communément utilisée. Cette approche est tout de même basée sur la recherche d'un plus court chemin dans un graphe pour trouver des colonnes de coût réduit négatif, mais cet aspect est transformé sous forme de contraintes (Fahle *et al.*, 2002). Cette méthode de résolution a l'avantage d'être plus flexible que la programmation dynamique dans la formulation du sous-problème et permet de tenir compte des règles complexes du transport aérien plus facilement. En effet, il est relativement simple de considérer des préassignations à l'aide de cette approche. Finalement, la propagation de contraintes a définitivement un potentiel d'accélération élevé pour ce problème de satisfaction de contraintes. Les travaux de Fahle *et al.* (2002) démontrent que cette approche est prometteuse quoique pas tout à fait compétitive avec des méthodes à la fine pointe de la technologie comme l'agrégation dynamique de contraintes. Il s'agit cependant d'un bel exemple d'intégration de la programmation par contraintes avec la recherche opérationelle, mais qui demande à être amélioré.

2.2.5 Approches réseau

Ce type d'approche est étroitement lié à celui par colonnes et s'en distingue par la manière de générer les colonnes. Effectivement, les problèmes impliquant le déplacement de véhicules ou d'appareils peuvent se modéliser sur des graphes orientés dont les sommets correspondent à des emplacements et les arcs à des déplacements. On peut aussi avoir recours à des graphes ou à des réseaux espace-temps pour modéliser des problèmes comme la construction des rotations d'avions et de construction des rotations d'équipage. De plus, dans un problème de construction des blocs mensuels, une solution peut être représentée sous forme de diagramme de Gantt qui est aussi un ensemble de chemins dans un réseau temporel. Un des principaux avantages d'une

telle modélisation est qu'il est relativement facile de considérer des coûts hautement non linéaires qui sont particulièrement présents dans les problèmes de planification des opérations en transport aérien.

Génération de colonnes

La génération de colonnes est une méthode très bien adaptée aux problèmes de construction d'horaires et de rotations puisque ces derniers sont naturellement formulables avec des modèles de partitionnement d'ensemble et que tous les aspects non linéaires se retrouvent dans le sous-problème. Cependant, cette approche a besoin d'une méthode performante pour résoudre ce sous-problème et c'est justement avec l'introduction d'un algorithme de programmation dynamique spécialisé que Desaulniers *et al.* (1997) ont eu du succès chez Air France pour le problème de construction des rotations d'équipage. En effet, il est possible de formuler le sous-problème comme un problème de plus court chemin avec contraintes de ressources, connu NP-difficile, mais la programmation dynamique se comporte de manière pseudo-polynomiale si les fonctions de prolongation de coût et de ressources sont non-décroissante.

Par ailleurs, les travaux de Weir et Johnson (2004) présentent une utilisation de la génération de colonnes pour résoudre un problème de *bidline*. La méthode que ces auteurs présentent est composée de trois phases distinctes résolvant chacune un plus petit problème plutôt que de s'attaquer directement au problème en entier. L'objectif poursuivi par ces travaux est de construire des horaires mensuels apportant une grande qualité de vie aux employés en respectant le plus possible une régularité dans la disposition des tâches ayant pour effet de diminuer la fatigue. La première phase de leur approche tente de construire des patrons d'horaires avec des rotations non-datées de manière à couvrir toutes les rotations du problème. Les patrons sont construits par énumération implicite et une fois que leur nombre est suffisant, un problème en nombres entiers mixte est résolu avec ces patrons. La deuxième phase tente de placer des rotations datées dans ces patrons d'horaires et se résout de manière semblable au

problème de la première phase. La dernière phase trouve une solution de la meilleure qualité possible pour toutes les rotations n'ayant pas pu être placées dans les patrons de la première phase.

La génération de colonnes consiste donc en une alternance entre la résolution du problème maître restreint (PMR) et du ou des sous-problèmes (SP). On résout une relaxation linéaire du problème de partitionnement d'ensemble qui nous donne les valeurs des variables duales associées à chacune des contraintes. Or, la plupart des contraintes dans le problème de construction des blocs mensuels sont associées à la couverture des tâches et ainsi, une valeur élevée d'une telle variable duale peut être interprétée comme une « mauvaise » couverture de la tâche, c'est-à-dire qu'il y a un potentiel d'amélioration de l'objectif en la couvrant autrement. Lors de la résolution du ou des sous-problèmes, les chemins passant par cette tâche auront tendance à avoir un coût réduit plus négatif, par conséquent, un potentiel d'amélioration de l'objectif élevé. Les valeurs des variables duales sont donc intégrées aux réseaux pour aider à trouver des chemins de coût réduit négatif et de tels chemins, qui correspondent à des horaires ou à des colonnes, sont ajoutés au problème maître dont la relaxation linéaire sera à nouveau résolue. Ce processus se poursuit jusqu'à ce qu'aucune colonne de coût réduit négatif ne soit générée, ce qui garantit l'optimalité pour un problème en nombres réels. Pour un problème en nombres entiers tel que le nôtre, on peut ensuite utiliser une méthode séparation et génération progressive (SGP) (*branch-and-price*) pour trouver une solution entière .

On peut avoir recours à l'ajout de coupes au problème pour resserrer la relaxation linéaire du domaine réalisable du problème de partitionnement d'ensemble de manière à accélérer la résolution du problème en nombres entiers à l'aide de SEP. On donne le nom de *branch-and-cut* à cette approche. Un tel solveur a été développé par Hoffman et Padberg (1993) et est capable de résoudre de gros problèmes de partitionnement d'ensemble (1000 contraintes, 1 million de variables) jusqu'à l'optimalité. Le *branch-and-cut* n'est cependant pas une méthode basée à l'origine sur la génération

de colonnes, mais il existe des applications qui utilisent des coupes conjointement avec la génération de colonnes comme Gamache *et al.* (1998).

Par ailleurs, pour obtenir une solution optimale par génération de colonnes, on doit avoir recours à nouveau à la génération de colonnes à l'intérieur des noeuds de branchement de la SEP, ce que l'on appelle SGP. À chaque nouveau noeud de branchement de l'arbre, on résout la relaxation linéaire du problème maître restreint (PMR) et on cherche à générer de nouvelles colonnes qui seront ajoutées au PMR. Desaulniers *et al.* (1998); Barnhart *et al.* (1998); Klabjan (2005) présentent une revue exhaustive de ce type de méthodes avec, entre autres, un exemple d'application sur la construction de rotations d'équipage.

La génération de colonnes a cependant ses limites dont la difficulté à résoudre optimalement des problèmes de très grande taille. En effet, lorque le nombre de rotations est très grand, le nombre de contraintes dans le problème maître l'est lui aussi, ce qui cause un phénomène de dégénérescence faisant exploser le temps de calcul en raison des nombreux pivots inutiles de l'algorithme du simplexe utilisé pour résoudre le PMR. Même si plusieurs astuces peuvent être utilisées pour repousser les limites implicites du nombre de contraintes ou de rotations, la dégénérescence a poussé les chercheurs à modifier l'approche.

Agrégation dynamique de contraintes

L'agrégation dynamique de contraintes propose une solution à la dégénérescence et permet des gains en termes de performance. Cette approche repose sur le principe qu'il est possible de combiner plusieurs tâches en une seule si ces tâches ont tendance à être couvertes une à la suite de l'autre dans un même horaire dans une solution optimale. On peut ainsi réduire considérablement la taille du problème maître, mais la bonne partition ou le bon regroupement de tâches est inconnu au début de la résolution. C'est pourquoi le processus doit être dynamique, c'est-à-dire que la partition

doit évoluer au fil des itérations. Les idées de base de la méthode ont été présentées dans Villeneuve (1999), mais le développement et l'implantation sont présentés dans El Hallaoui *et al.* (2005, 2008b,a). De plus, ces travaux proposent une méthode plus appropriée pour obtenir les valeurs des variables duales désagrégées rendant ainsi l'agrégation dynamique de contraintes vraiment efficace.

L'agrégation dynamique de contraintes consiste en une série d'agrégations et de désagrégations de la partition. Ainsi, un chemin incompatible avec une partition Q peut devenir compatible en désagrégeant Q. Cette méthode s'adapte donc au fil des itérations à tout chemin ayant un coût réduit négatif, c'est-à-dire ayant le potentiel de faire diminuer l'objectif. Or, il a été prouvé par El Hallaoui *et al.* (2005) que l'algorithme converge au même titre que la génération de colonnes.

Les travaux de Boubaker (2006) (Boubaker *et al.*, 2008) portent aussi sur l'agrégation dynamique de contraintes et sur un problème de construction des horaires mensuels dans un contexte de *bidline*. Le présent mémoire est la suite de Boubaker (2006) et reprend quelques-unes des idées proposées en raison des résultats impressionnants obtenus comme la réduction du temps de calcul d'un facteur de 61 pour les plus grosses instances avec l'application de l'agrégation dynamique par rapport à la génération de colonnes.

CHAPITRE 3
ALGORITHME DE RECHERCHE TABOUE

Ce chapitre présente la première étape du processus de résolution, soit la construction d'une solution initiale de bonne qualité. Une telle solution est en effet nécessaire à l'application de l'agrégation dynamique de contraintes comme méthode de résolution. Puisque cette étape ne constitue qu'une petite partie de toute l'approche, peu de temps de calcul doit y être consacrée. Conséquemment, l'algorithme mis en place devra être très rapide, ce qui implique que la fonction objectif devra être évaluée très rapidement et les mouvements vers des solutions voisines devront être trouvés tout aussi rapidement. De plus, il est important que la solution trouvée soit de bonne qualité pour que la partition initiale, la manière d'agréger les tâches, de l'agrégation dynamique de contraintes soit la moins différente possible de celle de la solution optimale.

Par ailleurs, le problème de construction d'horaires de personnel traité ici peut être formulé comme un problème de partitionnement d'ensemble et une formulation analogue de coloration de graphe sera utilisée dans cette partie pour résoudre le problème.

3.1 Coloration de graphe

On peut modéliser notre problème comme un problème de coloration de graphe en définissant l'ensemble des sommets V comme l'ensemble des rotations, tâches préassignées et vacances. On cherche une couleur, une personne pour effectuer la tâche, par sommet, seulement dans le cas des rotations puisque les tâches préassignées et les vacances sont déjà affectées à des personnes. L'ensemble des arêtes E est composé

des paires de sommets ou tâches qu'il est impossible pour un employé d'effectuer simultanément.

On obtient ainsi un graphe $G = (V, E)$ dont une petite partie des sommets est déjà colorée, et on cherche à colorer ceux restant avec un nombre fixe de couleurs, soit le nombre de personnes disponibles. On se retrouve donc avec un problème de k-coloration (k-Col) où dans notre cas, k est le nombre d'employés. On peut alors construire l'horaire d'une personne en cherchant dans le graphe tous les sommets ayant sa couleur, la couleur qui lui correspond. On sait empiriquement que le nombre minimal de couleurs nécessaires pour colorer G, le nombre chromatique χ, est inférieur au nombre d'employés sans toutefois connaître le premier. Trouver ce nombre est un problème qui est connu comme étant NP-difficile. En effet, le premier problème ayant été démontré comme appartenant à l'ensemble NP-complet est SAT-CNF qui est un problème de satisfaisabilité sous une forme normale conjonctive (Cook, 1971) et il est possible de lui faire subir une série de transformations pour qu'il soit équivalent à k-Col sous sa forme décisionnelle, démontrant ainsi que ce dernier est lui aussi NP-complet dans le cas général. Cependant, le graphe $G = (V, E)$ en est un d'intervalles (Hajòs, 1957; Kolen *et al.*, 2007) et on sait que les graphes d'intervalles sont des graphes triangulés qui admettent un schéma d'élimination parfait (Gupta *et al.*, 1979; Kolen et Lenstra, 1995), c'est-à-dire qu'il existe un ordre statique dans lequel on peut colorer séquentiellement les sommets en leur affectant la plus petite couleur non-utilisée par ses voisins pour obtenir une coloration avec χ couleurs. Déterminer un tel schéma est un problème facile dans le cas des graphes d'intervalles, mais il a été démontré que deux préaffectations suffisaient à rendre le problème NP-difficile (Biró *et al.*, 1992). Ceci rend donc cette propriété inutilisable dans notre cas. De plus, à chaque sommet on associe un poids correspondant au nombre de crédits de la rotation, de la tâche préassignée ou de la période de vacances et la somme des poids des sommets ayant la même couleur doit être comprise entre deux valeurs. D'autres contraintes s'ajoutent aussi, mais sont difficiles à modéliser en termes de graphes. Ces contraintes, le nombre minimal de jours de congé et le nombre maximal de jours

travaillés consécutifs, sont évaluées différemment. On ajoute donc des contraintes supplémentaires à un problème déjà connu comme étant difficile, le rendant ainsi encore plus difficile.

Toutefois, certains algorithmes exacts existent et trouvent des solutions pour de petites instances (à ce jour, environ 100 sommets) (Herrmann et Hertz, 2002), mais la taille du problème à traiter est beaucoup plus grande et l'utilisation d'un algorithme exact pour trouver une partition initiale n'est donc pas envisageable.

3.1.1 Heuristiques

Puisque les méthodes exactes ne peuvent être utilisées, on doit se replier sur des heuristiques pour trouver une solution initiale au problème, mais ces dernières ne sont pas toutes équivalentes en termes de performance et de qualité de solution. Cette partie sera consacrée à un bref survol de quelques-unes des méthodes qu'il est possible d'utiliser pour colorer des graphes.

Algorithmes gloutons

Le principe général de ces heuristiques est de prendre à chaque étape ou itération la meilleure décision possible selon un ou plusieurs critères sans remettre en question les décisions prises antérieurement. Une des méthodes très utilisées est l'heuristique de saturation (DSATUR) (Brélaz, 1979) qui consiste à colorer, à chaque itération, le sommet ayant le plus grand nombre de voisins déjà colorés. Il existe aussi l'heuristique RLF (*Recursive Largest First*) (Leighton, 1979) qui affecte chaque couleur séquentiellement en ordre croissant au plus grand nombre de sommets possible. Ainsi, l'algorithme commence par dresser une liste de tous les sommets non colorés de laquelle on choisit le sommet ayant le plus grand nombre de voisins non colorés et on lui affecte la couleur courante. Ce sommet et tous ceux qui lui sont adjacents sont retirés

de la liste et on colore ainsi d'autres sommets jusqu'à ce que la liste soit vide. On passe ensuite à la couleur suivante et ainsi de suite jusqu'à ce que tous les sommets soient colorés.

Il existe aussi des algorithmes gloutons qui, à l'instar des deux algorithmes précédents, utilisent un ordre statique pour la coloration des sommets. Ces derniers sont ainsi classés selon un ordre déterminé à l'avance, puis colorés séquentiellement dans cet ordre en affectant la plus petite couleur non utilisée par ses voisins. On peut classer les sommets selon des ordres différents le plus simple étant au hasard (*Random*), mais il est aussi possible de le faire selon un ordre non-croissant des degrés (nombre d'arêtes incidentes) des sommets (*Largest First*) (Johri et Matula, 1982). Un autre ordre (*Smallest Last*) (Johri et Matula, 1982) possible est tel que le i^e sommet de la liste a le plus petit degré dans le graphe induit par les sommets $1, ..., i$ de cette liste.

La plus performante de ces méthodes est sans contredit RLF, suivie de près par DSATUR, tandis que *Smallest Last* et *Largest First* ne font qu'un peu mieux que *Random* (Hertz, 2003). La complexité de RLF est en $O(|V|^3)$ tandis que les autres ont une complexité de $O(|V|^2)$.

Métaheuristiques

Les métaheuristiques sont, en général, des méthodes comprenant une procédure maîtresse guidant des heuristiques spécifiques au problème dans le but d'obtenir des solutions de bonne qualité. On peut les différencier, entre autres, selon la manière dont elles manipulent les solutions, soit une seule à la fois ou par population. Les premières, en raison de l'unicité de la solution à chaque itération, sont souvent appelées méthodes à recherche locale ou méthodes de trajectoire. Voici un bref aperçu de quelques méthodes existantes qui pourraient être utilisées.

Méthodes à recherche locale Les deux méthodes de recherche locale les plus connues sont probablement le recuit simulé (Kirkpatrick *et al.*, 1983) et la recherche

taboue (Glover, 1986). Le fondement du recuit simulé repose sur une probabilité, principalement décroissante au fil des itérations, d'accepter des mouvements qui détériorent la solution. Quant à eux, les mouvements qui l'améliorent sont systématiquement acceptés. Malgré tous ses avantages, cette méthode a recours très fréquemment au hasard et ce, même pour le choix d'une solution voisine. Ainsi, si un optimum global est très près de la solution courante, il est possible que l'algorithme ne choisisse pas une solution voisine de cet optimum alors que c'était le meilleur choix. Du moins, cette méthode dispose d'un théorème (Aarts *et al.*, 1997) prouvant que les probabilités de trouver un optimum augmentent au fil des itérations sous certaines conditions. Ce théorème garantit même l'optimalité : il existe un $L \in \mathbb{R}$ tel que $\lim_{k\to\infty} P(k) = 1$ si $\sum_{k=1}^{\infty} e^{\frac{L}{T_k}} = \infty$ où T_k est la température, un paramètre, à la k^e itération et $P(k)$ la probabilité de trouver un optimum à la k^e itération. Toutefois, ce critère sur la température fait en sorte que la convergence est excessivement lente, rendant par conséquent ce théorème inutilisable en pratique.

Par ailleurs, la recherche taboue a l'avantage d'examiner chacune des solutions d'un voisinage pour en choisir la meilleure (*Best Improvement*) empêchant ainsi de passer à côté d'un optimum global. Cependant, il est aussi possible de ne pas passer en revue l'ensemble des solutions voisines en prenant la première qui améliore la solution courante (*First Improvement*). Cette dernière manière de procéder a l'avantage d'être beaucoup plus rapide lorsque la taille des voisinages est relativement grande et nécessite donc moins d'efforts dans la construction de voisinages plus petits, mais on retrouve ainsi un des défauts du recuit simulé, soit la possibilité de passer à côté d'un optimum global. Certes, on peut contrôler le cyclage avec une liste taboue, qui est un liste de mouvements interdits, tandis qu'avec le recuit simulé, on se fie plutôt au hasard. Il y a cependant un risque que la liste taboue, si on y met trop « d'intelligence », empêche d'aller visiter des solutions intéressantes ou qu'elle ne contraigne pas suffisamment le cyclage dans certains cas, ce qui implique que les choix à faire dans l'utilisation de la liste taboue sont importants et même cruciaux pour que cette méthode fonctionne bien.

Dans le même ordre d'idées, il existe d'autres méthodes de recherche locale un peu plus récentes qui émergent en popularité telles que la recherche à voisinages variables (*variable neighbourhood search* - VNS) (Mladenović et Hansen, 1997) qui, contrairement aux méthodes précédentes, utilise plus d'une structure de voisinage. Cette méthode permet donc de diversifier l'exploration de l'espace des solutions et ainsi de mettre en évidence plus de régions intéressantes conduisant à une méthode plus robuste que la recherche taboue ou que le recuit simulé.

Méthodes basées sur les populations Les deux principales méthodes basées sur des populations de solutions sont les algorithmes génétiques (Davis, 1991) et la recherche dispersée (*scatter search*) (Glover, 1977). L'idée principale des algorithmes génétiques consiste à faire évoluer un ensemble de solutions souvent avec des méthodes de recherche locale et de synthétiser l'information contenue dans l'ensemble de ces solutions en combinant les éléments intéressants des solutions prometteuses. Les algorithmes basés sur la recherche dispersée, quant à eux, proposent plutôt de réparer avec une procédure spécifique et d'améliorer à l'aide de méthodes de trajectoires des solutions obtenues à partir de combinaisons linéraires d'un ensemble de solutions de référence. À l'itération suivante, on choisit un ensemble de solutions parmi celles nouvellement obtenues et parmi celles de références qui deviendront le nouvel ensemble de solutions de référence à partir duquel on peut répéter le processus.

Les méthodes basées sur les populations fonctionnent en général très bien pour trouver des régions intéressantes de l'espace des solutions et un peu moins bien que les méthodes de recherche locale pour explorer intensivement une région. Un algorithme de recherche locale guidé par une méthode basée sur les populations fonctionne habituellement très bien puisqu'il complémente les forces des deux approches. Cependant, la mise en place d'une telle approche est un peu plus complexe et un peu plus lourde en ressource machine même si, après un certain temps de calcul, cette approche dépasse généralement les méthodes de recherche locale en termes de qualité de solution.

3.2 Algorithme utilisé

La recherche taboue a été retenue comme méthode de résolution principalement en raison du succès obtenu par Tabucol (Hertz et de Werra, 1987) pour les problèmes de coloration. Cette méthode a l'avantage d'être relativement simple à implanter et de donner des résultats de qualité satisfaisante très rapidement, ce qui est exactement la finalité recherchée. On évite ainsi la lourdeur de VNS et des algorithmes basés sur les populations et le hasard du recuit simulé.

Par ailleurs, il a été choisi d'utiliser une structure de voisinage inspirée par la recherche à voisinage variable pour la génération des solutions voisines afin d'accroître l'efficacité de l'algorithme dans le but d'obtenir rapidement des régions intéressantes de l'espace des solutions ou plutôt des mouvements intéressants issus de ces régions.

Cette section est consacrée à la description de l'algorithme implanté en faisant le plus possible abstraction du langage et des structures de données utilisées. On y trouvera les principaux ingrédients composant la méthode choisie. On commencera donc par décrire brièvement le contexte général d'un algorithme de recherche taboue suivi de la description de l'espace des solutions considéré, de l'objectif évalué, du calcul par incrément et de la structure des voisinages. Ensuite sera présentée la manière d'obtenir une solution initiale pour démarrer l'algorithme et, finalement, la description de la manière de gérer la liste taboue.

Nous utiliserons les notations suivantes tout au long de ce chapitre :

- $G(V, E)$: Graphe ayant V comme ensemble de sommets et E comme ensemble d'arêtes ;
- k : Nombre d'employés ;
- v : Sommet du graphe tel que $v \in V$;
- c : Couleur d'un sommet du graphe ; correspond à un employé ;
- $C_i(v)$: Couleur attribuée au sommet v à l'itération i ;

- $M(v, c, c')$: Changement de couleur du sommet v de la couleur c à la couleur c' ; $M(v, c, c') = M_a(v, c') \oplus M_r(v, c)$;
- $M_r(v, c)$: Mouvement partiel : « décoloration » du sommet v de sa couleur c ;
- $M_a(v, c)$: Mouvement partiel : coloration du sommet non-coloré v avec la couleur c ;
- S : Espace de solutions ;
- s : Solution $s \in S$;
- s^* : Meilleure solution rencontrée ;
- s^i : Solution $s^i \in S$ à la i^e itération ;
- s^i_{part} : Solution $s^i_{part} \notin S$ à la i^e itération issue de l'application d'un mouvement partiel ;
- $s^i \xrightarrow{M(v,c,c')} s^{i+1}$: Application du changement de couleur $M(v, c, c')$ à s^i pour obtenir s^{i+1} ;
- T : Liste taboue ;
- $N(s)$: Voisinage autour de s ;
- $MinCr$: Nombre minimal de crédits à octroyer à un employé ;
- $MaxCr$: Nombre maximal de crédits à octroyer à un employé ;
- $NbCr^c$: Nombre de crédits dans l'horaire de l'employé c ;
- $MinCg$: Nombre minimal de jours de congé à octroyer à un employé ;
- $NbCg^c$: Nombre de jours de congé dans l'horaire de l'employé c ;
- $MaxJC$: Nombre maximal de jours de travail consécutifs ;
- μ_{Cr} : Moyenne du nombre de crédits par employé ;
- μ_{Cg} : Moyenne du nombre de jours de congé par employé ;
- σ_{Cr} : Écart-type du nombre de crédits ;
- σ_{Cg} : Écart-type du nombre du jours de congé.

3.2.1 Description générale de la méthode

La communauté scientifique s'accorde généralement pour affirmer que l'on doit la recherche taboue à Glover (1986). Il s'agit d'une méthode itérative qui, selon

l'approche utilisée, prend le premier voisin qui améliore la solution ou le meilleur voisin. Une liste de mouvements interdits, la liste taboue, permet à l'algorithme de ne pas systématiquement retourner sur ses pas et ainsi de se sortir d'un minimum local. Il est aussi possible de mémoriser des ensembles de solutions en interdisant de revisiter une solution déjà évaluée, mais cette approche bien que plus efficace pour empêcher le cyclage, présente des inconvénients majeurs : la mémorisation de solutions entières occupe beaucoup de place en mémoire et la vérification de l'appartenance de la solution vers laquelle on veut se déplacer à une telle liste de solutions est coûteuse en temps de calcul.

On peut résumer ce qui vient d'être énoncé de manière très succinte grâce à l'algorithme 3.1 tiré de Hertz (2004). Tout d'abord, si on définit $f(s)$ comme la valeur de l'objectif pour la solution s, on peut définir $N^T(s)$ comme suit pour l'algorithme :

$$N^T(s) = \{s' \in N(s) \mid (s' \notin T) \text{ ou } (f(s') < f(s^*))\}$$

Algorithme 3.1 Schéma général d'un algorithme de recherche taboue

1: Choisir une solution $s \in S$, poser $T = \emptyset$ et $s^* := s$
2: **tant qu'** aucun des critères d'arrêt n'est pas satisfait **effectuer**
3: Déterminer une solution s' qui minimise $f(s')$ dans $N^T(s)$
4: **si** $f(s') < f(s)$ **alors**
5: Poser $s^* := s'$
6: Poser $s := s'$ et mettre à jour T

3.2.2 Techniques d'implantation

Voici un ensemble de recommandations formulées originalement sous forme de maximes dans la thèse de doctorat de Zufferey (2002) qui donnent les grandes lignes des qualités que l'on devrait retrouver dans une métaheuristique basée sur une méthode de trajectoire ou de recherche locale.

Tout d'abord, la notion de solution voisine est très importante pour que l'algorithme soit efficace. Tel que mentionné par Zufferey (2002), une solution voisine s'

ne doit pas contenir les mêmes défauts que la solution courante s, c'est-à-dire que le mouvement vers une solution voisine doit régler au moins un des défauts de la solution s, quitte à en créer plusieurs autres dans s'.

Ensuite, la solution optimale devrait être accessible à partir d'une séquence de solutions voisines l'une de l'autre, qu'elles soient réalisables ou non. En effet, dans certains problèmes comme le nôtre, il est difficile d'obtenir une solution réalisable et on ne doit surtout pas se limiter à ces dernières si on veut avoir une chance de tomber sur une solution optimale. Le fait de ne pas se limiter à ces dernières peut aussi faciliter grandement les déplacements dans l'espace des solutions. On peut donc mesurer à l'aide de l'objectif la « distance » à la réalisabilité pour guider l'algorithme en plus de l'objectif réel.

Dans le même ordre d'idées, afin de bien guider la recherche, une solution voisine ne doit pas être trop différente de son homologue original. Ceci facilite en plus la mise en oeuvre du calcul incrémental qui accélère grandement l'évaluation des solutions. Cette dernière tactique est expressément pertinente dans le cas où on veut évaluer un très grand nombre de voisins comme c'est le cas ici.

3.2.3 Approche générale

L'approche retenue pour résoudre le problème se décompose en plusieurs étapes et le but de cette section est de mettre en évidence l'interaction entre les étapes et le cadre général dans lequel s'inscrivent ces étapes.

La première étape consiste à trouver une solution initiale faisant partie de l'espace des solutions de l'algorithme de recherche taboue pour pouvoir démarrer ce dernier. L'algorithme démarre ensuite la phase I qui consiste à trouver une solution réalisable et s'arrête à la première trouvée. La sortie de cette étape est donc une solution réalisable qui pourrait être directement utilisée pour démarrer l'agrégation dynamique de

contraintes. Cependant, comme elle n'est pas nécessairement de très grande qualité, il est facile de l'améliorer. C'est exactement le rôle de la phase II qui cherche à minimiser la somme pondérée des écarts-types des crédits et des jours de congé tout en évitant les violations de contraintes. Puisqu'il est très difficile d'évaluer quels seraient les coefficients qui équilibreraient toutes les composantes de la fonction objectif, cette phase prévoit des mécanismes d'ajustement des coefficients de manière à parcourir l'espace des solutions en ayant une certaine proportion de solutions réalisables sans toutefois se restreindre à celles-ci.

3.2.4 Espace des solutions

Puisqu'il s'agit en première partie d'un problème de réalisabilité, il aurait été étrange de ne pas permettre les solutions non-réalisables. En effet, toute solution dont tous les sommets sont colorés est donc considérée comme valide. De plus, les sommets correspondants à des tâches préassignées ou à des périodes de vacances doivent obligatoirement avoir la couleur de l'employé affecté à cette tâche ou période de vacances. Une solution ne respectant pas cette dernière condition ne fera pas partie de l'espace des solutions. Toutefois, une solution impliquant qu'un employé aie plusieurs rotations pendant sa période de vacances est considérée comme valide, même si elle viole plusieurs contraintes comptabilisées dans l'objectif (voir §3.2.5) puisqu'il sera possible de déplacer les rotations. La couleur des sommets correspondant aux préaffectations ne change donc jamais. Cette manière de procéder a l'avantage de permettre, à partir de n'importe quelle solution, par une succession de changements de couleurs, d'obtenir une solution optimale même si les chances que l'algorithme en trouve une en un temps raisonnable sont minces. La dernière affirmation a introduit indirectement la notion de solution voisine par l'entremise des changements de couleur, ce qui permet de définir précisément une solution voisine s' comme une solution s à laquelle un sommet a été changé de couleur, c'est-à-dire qu'une rotation est enlevée de l'horaire d'un employé, que l'on nommera horaire source, pour être ajoutée à celui d'un autre employé, que l'on nommera horaire de destination.

3.2.5 Fonction objectif

Comme mentionné au début de ce chapitre, pour que l'algorithme soit performant, la fonction objectif doit être évaluée très rapidement et la meilleure manière d'y parvenir passe par l'utilisation du calcul incrémental qui sera présenté dans la prochaine section. Cependant, les incréments à appliquer à la fonction objectif doivent être calculés intelligemment pour que l'algorithme soit vraiment performant. Par conséquent, certaines composantes de la fonction objectif ne sont pas propices au calcul par incrément. Cette section et la suivante sont donc consacrées à la description de l'objectif évalué et à la façon de l'évaluer.

La fonction objectif compte six composantes dont quatre pour mesurer la « distance » à la réalisabilité et les deux dernières servent à discriminer les « bonnes » des « mauvaises » solutions réalisables. Même si l'affirmation suivante peut paraître redondante, il s'avère important de mentionner que toute contrainte dont on ne mesure pas la « distance » à la réalisabilité ne peut être violée, notamment les préaffectations et la couverture de toutes les tâches ou vacances.

Chevauchements

On considère que deux tâches i et j se chevauchent (voir figure 3.1), étant donné que i débute avant ou en même temps que j, si le début de la tâche j est plus tôt que la fin de la tâche i. Le terme de l'objectif comptabilisant les chevauchements est noté F_1. Lorsque plusieurs tâches se chevauchent, on compte les chevauchements en

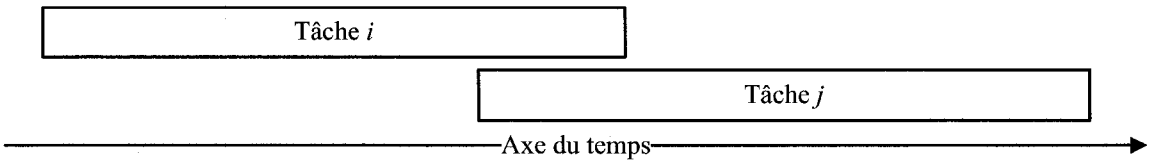

Figure 3.1 – Illustration d'un chevauchement

prenant les tâches deux à deux. Ainsi dans l'exemple de la figure 3.2, on compte 5 chevauchements lorsque les tâches sont prises deux à deux.

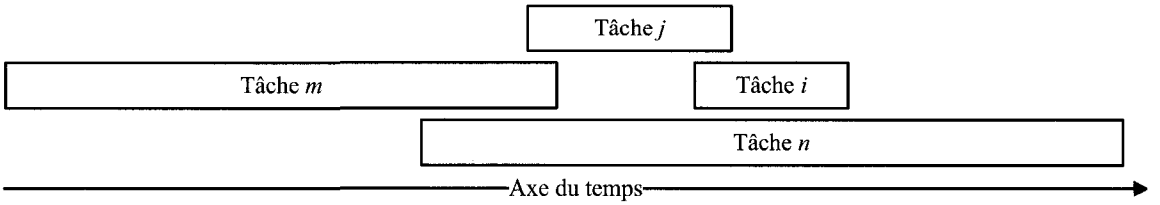

Figure 3.2 – Illustration de plusieurs chevauchements

Crédits

On associe un certain nombre de crédits, qui est l'unité de mesure du travail, à chaque tâche et la somme des crédits des tâches assignées à un employé doit être comprise entre une borne inférieure et une borne supérieure. Chaque crédit en dehors de l'intervalle délimité par les bornes inférieures et supérieures est comptabilisé dans la fonction objectif. Si on note k le nombre d'employés, le calcul de la composante crédits F_2 se fait de la manière suivante :

$$F_2^c = \begin{cases} MinCr - NbCr^c & \text{si} \quad NbCr^c < MinCr \\ NbCr^c - MaxCr & \text{si} \quad NbCr^c > MaxCr \\ 0 & \text{sinon} \end{cases}$$

et

$$F_2 = \sum_{c=1}^{k} F_2^c .$$

Jours de congé

L'horaire d'un employé doit comprendre un nombre minimal de jours de congé et chaque jour de congé en deçà du minimum permis est comptabilisé dans la composante jours de congé F_3 de la fonction objectif. Le calcul de cette composante se fait

de la manière suivante :

$$F_3^c = \begin{cases} MinCg - NbCg^c & \text{si} \quad MinCg > NbCg^c \\ 0 & \text{sinon} \end{cases}$$

et

$$F_3 = \sum_{c=1}^{k} F_3^c \, .$$

Jours de travail consécutifs

Un employé ne peut pas travailler plus d'un certain nombre de jours de manière consécutive. Conséquemment, chaque jour d'une série de jours travaillés consécutifs en excès est comptabilisé dans la composante jours consécutifs F_4 de la fonction objectif. Soit j_{max}^c le nombre de séries de jours travaillés consécutifs pour l'employé c. Si on note L_j^c le nombre de jours dans la série j pour l'employé c, $j \in [1, ..., j_{max}^c]$, le calcul de cette composante se fait de la manière suivante :

$$F_4^{cj} = \begin{cases} L_j^c - MaxJC & \text{si} \quad L_j^c > MaxJC \\ 0 & \text{sinon} \end{cases}$$

et

$$F_4 = \sum_{c=1}^{k} \sum_{j=1}^{j_{max}^c} F_4^{cj} \, .$$

Écarts-types

Le calcul des écarts-types est utilisé dans la phase d'uniformisation et sert à construire des horaires les moins différents possibles les uns des autres en terme de jours de congés et de crédits. On a plutôt recours à des estimateurs que l'on calcule avec μ_{Cr} pour la composante F_5 de l'objectif et μ_{Cg} pour la composante F_6, mais μ_{Cg} n'est pas constant. La composante des jours de congé F_6 sera donc réévaluée à chaque fois que μ_{Cg} change. On négligera cependant cette variation lors de l'évaluation des

mouvements, c'est-à-dire qu'étant donné l'évaluation d'un mouvement qui fait changer la moyenne, la nouvelle moyenne ne sera pas calculée et on fera le calcul avec la valeur courante. Le calcul des deux composantes se fait de la manière suivante :

$$\begin{aligned} F_5^c &= (NbCr^c - \mu_{Cr})^2 \\ F_6^c &= (NbCg^c - \mu_{Cg})^2 \\ F_5 &= \sum_{c=1}^{k} F_5^c \\ F_6 &= \sum_{c=1}^{k} F_6^c \,. \end{aligned}$$

On utilise donc la somme du carré des écarts dans l'objectif plutôt que l'écart-type principalement parce que les deux mesurent la même quantité et que le calcul de la somme du carré des écarts est plus rapide. On peut retrouver les écarts-types de la manière suivante à tout moment au cours de l'algorithme si nécessaire :

$$\begin{aligned} \sigma_{Cr} &= \sqrt{\tfrac{F_5}{k}} \\ \sigma_{Cg} &= \sqrt{\tfrac{F_6}{k}} \,. \end{aligned}$$

Fonctions complètes

Lors de la phase de recherche d'une solution réalisable, la fonction objectif est la suivante :

$$F = \sum_{j=1}^{4} \alpha_j F_j$$

où les coefficients α_j sont fixés au début de l'algorithme à des valeurs déterminées empiriquement qui permettent à la phase I de l'algorithme de converger rapidement. Lors de la phase d'uniformisation, on utilise plutôt cette fonction :

$$F = \sum_{j=1}^{6} \alpha_j F_j \,.$$

Les coefficients sont ajustés dynamiquement lors de cette phase de résolution de manière à obtenir un équilibre entre toutes les composantes de la fonction objectif qui ont des ordres de grandeur bien différents (Alabas-Uslu, 2007; Gamache *et al.*, 2007). En effet, à chaque B itérations, l'algorithme observe le nombre de fois où la solution était réalisable au cours de ces B dernières itérations et on peut ensuite augmenter les coefficients des estimateurs d'écarts-types si la solution était trop souvent réalisable et ceux des quatre autres composantes si elle ne l'était pas suffisamment. On peut aussi choisir de ne rien faire si on juge que les coefficients sont équilibrés. Cependant, l'algorithme ne peut ajuster les coefficients de la variance des crédits et de la variance des jours de congé l'un par rapport à l'autre puisqu'aucun point de repère ne peut être utilisé pour évaluer objectivement une solution. Il revient donc à l'utilisateur d'estimer les deux coefficients ou plutôt la différence d'ordre de grandeur. Typiquement, le coefficient α_6 doit être au moins 100 fois plus grand que α_5 et peut aller jusqu'à 50000 fois plus grand si on désire prioriser des horaires équitables du point de vue des jours de congé. L'algorithme 3.2 décrit mieux ce qui a été implanté. Soit $\beta \in [0, 1]$ la proportion de solutions réalisables visée à chaque tranche de B itérations, $\Delta \in [0, 1]$ et $\Delta < \beta$ la tolérance que l'on se laisse sur β et $\delta > 1$ le coefficient de modification des α_j. On introduit une tolérance autour de β pour éviter les modifications inutiles aux coefficients, ce qui implique de nombreux calculs, donc plus de temps. On définit ainsi un intervalle acceptable de proportions de solutions réalisables qui implique que lorsque la proportion de solution réalisable est à l'intérieur de cet intervalle, on considère que l'algorithme se comporte correctement. Les expérimentations ont démontré que les valeurs suivantes des paramètres conduisaient à de meilleurs résultats : $B = 500$, $\beta = \frac{3}{10}$, $\Delta = \frac{1}{10}$ et $\delta = \frac{3}{2}$. Si on note Nb_{real} le nombre de solutions réalisables au cours des B dernières itérations, $i_{courante}$ le numéro de l'itération courante et F_j^i la valeur de la composante de l'objectif associée au coefficient α_j à l'itération i, on obtient l'algorithme suivant.

Algorithme 3.2 Ajustement dynamique des coefficients de la fonction objectif

si $i_{courante}$ modulo $B = 0$ **alors**
  si $Nb_{real} < (\beta - \Delta)B$ **alors**
    pour $j = 1$ jusqu'à 4 **effectuer**
      pour $i = i_{courante} - B$ jusqu'à $i_{courante}$ **effectuer**
        si $F_j^i > 0$ **alors**
          Poser la variable $verification := vrai$
      si $verification == vrai$ **alors**
        Poser $\alpha_j = \delta\alpha_j$
      sinon
        Poser $\alpha_j = \frac{\alpha_j}{\delta}$
  si $Nb_{real} > (\beta + \Delta)B$ **alors**
    Poser $\alpha_5 = \delta\alpha_5$ et $\alpha_6 = \delta\alpha_6$
  si $(\beta - \Delta)B < Nb_{real} < (\beta + \Delta)B$ **alors**
    si un des six α_j est supérieur à 10^6 **alors**
      Poser $\alpha_j = \frac{\alpha_j}{\delta} \quad \forall j \in [1, ..., 6]$

3.2.6 Calcul par incrément de la fonction objectif

Le calcul par incrément a l'avantage de permettre très rapidement l'obtention de la valeur d'une solution voisine. Ceci permet d'évaluer un très grand nombre de voisins dans un délai plus que respectable. C'est cette approche qui, lorsque combinée à une étude astucieuse des mouvements, permet d'obtenir un algorithme rapide. Le calcul par incrément nécessite, pour être mis en place, une ou plusieurs structures de données capable de contenir, pour une solution s^i, l'écart par rapport à toutes les solutions voisines s^{i+1}. Cependant, ici, la notion de voisin n'est pas prise dans le sens de $N(s^i)$, mais plutôt dans le sens de toute solution s^{i+1} qu'il est possible d'obtenir en changeant une rotation d'horaire dans la solution s^i. Si on note $F_j(s^i)$ la valeur non-pondérée de l'objectif pour la composante j et la solution s^i, $\Delta_j^i(v, c, c')$ l'incrément à appliquer à $F_j(s^i)$ à la i^e itération pour obtenir $F_j(s^{i+1})$ étant donné $s^i \xrightarrow{M(v,c,c')} s^{i+1}$, on obtient la relation suivante : $F_j(s^{i+1}) = F_j(s^i) + \Delta_j^i(v, c, c')$

Seules trois composantes, les trois plus lourdes à évaluer, font appel au calcul incrémental, soit les chevauchements, les jours de congé et les jours de travail consécutifs. On expliquera d'abord comment on évalue les chevauchements et les deux

autres seront alors plus simples à décrire puisque l'idée sous-jacente peut y être apparentée. Pour clarifier les explications qui suivent, il s'avère plus pratique de définir $\Delta_j^i(v,c,c')$ comme : $\Delta_j^i(v,c,c') = \Delta a_j^i(v,c') + \Delta r_j^i(v,c)$ où $\Delta a_j^i(v,c')$ est l'impact sur la composante j de l'objectif de l'ajout de la rotation correspondant à v à l'horaire destination correspondant à c' et $\Delta r_j^i(v,c)$ est l'impact sur la composante j de l'objectif du retrait de la rotation correspondant à v à l'horaire source correspondant à c.

Lors du mouvement d'une rotation v d'un horaire source c à un horaire de destination c' à l'itération i, la grande majorité des Δa_j^{i+1} restent inchangés. Plus précisément, étant donné $s^i \xrightarrow{M(v,c,c')} s^{i+1}$ on a que $\Delta a_j^i(v',c'')) = \Delta a_j^{i+1}(v',c'')\ \forall\ (c'' \neq c)$ et $\forall\ (c'' \neq c')$. Il est donc possible d'épargner beaucoup de temps de calcul en ne mettant à jour que les incréments susceptibles de changer lors d'un mouvement. Il est aussi possible que dans certains cas qui seront présentés plus loin :

$$\Delta a_j^i(v',c'') = \Delta a_j^{i+1}(v',c'') \text{ pour } c'' = c \text{ ou } c'' = c'.$$

D'une part, on décrira brièvement le fonctionnement des procédures permettant de mettre à jour les Δa_j^i en prenant pour acquis qu'ils ont déjà été initialisés c'est-à-dire que tous les Δa_j^0 sont connus. En effet, l'initialisation peut être faite par une procédure inefficace et l'impact sur la rapidité de l'algorithme serait minimal puisque cette procédure n'est exécutée qu'une seule fois. Par conséquent, la procédure d'initialisation ne sera pas présentée. Par ailleurs, la solution initiale peut contenir, pour certains horaires, des rotations chevauchant des tâches préassignées ou des périodes de vacances. L'initialisation des Δr_j^0 et la mise à jour des Δr_j^x doit prendre cet aspect en considération, mais on l'ignorera dans l'initialisation des $\Delta a_j^0(v,c)$ et la mise à jour des $\Delta a_j^i(v,c)$ puisque le mouvement $M_a(v,c)$ n'est pas autorisé si v chevauche une tâche préassignée ou une période de vacances dans l'horaire correspondant à c. Il est pertinent de rappeller que v ne peut être qu'une rotation.

D'autre part, l'initialisation des Δr_j^0 ne sera pas présentée pour les mêmes raisons que dans le cas de Δa_j^0. La mise à jour des Δr_j^i ne sera pas présentée non plus parce

que la procédure n'est pas très sophistiquée, donc peu intéressante. En effet, étant donné que le nombre z de rotations ayant la couleur c, par exemple, est généralement bas, c'est-à-dire que $z \ll |V|$ ou $z \approx \frac{|V|}{k}$, il n'est pas coûteux de réévaluer complètement les $\Delta r_j^{i+1}(v', c'') \; \forall \; v' \mid C_{i+1}(v') = c$ ou $C_{i+1}(v') = c'$.

L'algorithme pour la mise à jour des $\Delta a_1^x(M_a(v_m, c_n))$ se base sur le principe qu'il existe deux cas possibles pour que deux tâches se chevauchent. Étant donné $s^i \xrightarrow{M(v,c,c')} s^{i+1}$, on cherchera à mettre à jour : $\Delta a_1^{i+1}(v', c'')$ pour $c'' = c$ ou $c'' = c'$. En effet, les rotations v' qui chevauchent une autre tâche v sont :

1. Début de $v \leq$ début de $v' <$ fin de v
2. Début de $v' <$ début de $v <$ fin de v'

On trouve ainsi très rapidement toutes les rotations v' telles que :

$$\Delta a_1^i(v', c'') \neq \Delta a_1^{i+1}(v', c'').$$

L'algorithme nécessite deux listes des rotations ; l'une triée en ordre croissant du moment de début de la rotation et l'autre triée en ordre croissant de l'instant de fin de la rotation. On peut ainsi parcourir ces listes pour trouver rapidement toutes les rotations v' qui chevauchent la rotation déplacée v.

Par ailleurs, pour $\Delta a_3^{i+1}(v', c'')$ et $\Delta a_4^{i+1}(v', c'')$, il est possible de trouver un intervalle de temps $[t_t, t_u]$, étant donné $s^i \xrightarrow{M(v,c,c')} s^{i+1}$, tel que toute rotation v' chevauchant cet intervalle a de très fortes chances de voir son impact $\Delta a_j^i(v', c'')$ pour $c'' = c$ ou $c'' = c'$ changer. Puisque l'algorithme précédent trouve efficacement les chevauchements, on s'en est inspiré pour la mise à jour des $\Delta a_3^{i+1}(v', c'')$ et $\Delta a_4^{i+1}(v', c'')$. Reste à savoir comment définir $[t_t, t_u]$; ce dernier ne doit être ni trop grand, pour éviter les pertes de temps, ni trop petit, pour évaluer correctement les Δ. On définit t_t comme le début de la première journée de congé antérieure au début de la rotation ajoutée ou retirée v_j. De manière analogue, on définit t_u comme la fin du premier jour de congé postérieur à la fin de v_j. Ainsi, une rotation v' qui ne chevauche pas

$[t_t, t_u]$ ne sera pas influencée par v puisque $[t_t, t_u]$ peut être interprété comme la zone modifiée d'un horaire et, par conséquent, toute rotation v' en dehors de cet intervalle aura le même impact sur l'horaire qu'avant le retrait ou l'ajout de v. On peut donc utiliser la même technique que celle pour évaluer les chevauchements et lorsque l'on trouve une rotation v qui chevauche l'intervalle $[t_t, t_u]$, on réévalue l'impact de l'ajout de v à l'horaire modifié.

3.2.7 Voisinages

D'une part, étant donné l'utilisation du calcul incrémental, la forme de voisinage la plus appropriée dans notre contexte est l'utilisation de mouvements élémentaires (*1-move*), c'est-à-dire le changement de couleur d'un sommet pour passer à une solution voisine. Cette manière de procéder est identique à ce qui est utilisé dans Tabucol (Hertz et de Werra, 1987) qui a déjà fait ses preuves. D'autre part, comme on l'a mentionné précédemment, l'algorithme utilise une structure de voisinage inspirée de la recherche à voisinages variables. Il faut noter cependant que la seule idée reprise de VNS est la présence de plusieurs voisinages. Par contre, si on observe un peu plus la structure proposée plus loin, on constate que l'on a plutôt un seul voisinage décomposé en plusieurs sous-ensembles dont l'intersection peut être non-vide. Il est toutefois pratique de faire référence à plusieurs voisinages puisque cette multiplicité met en évidence la fonction de chacun. On compte donc sept voisinages ou sous-ensembles et chacun d'eux a pour fonction d'améliorer au moins une composante de la fonction objectif.

Par ailleurs, puisque les intersections des voisinages ne sont pas nécessairement vides, il est possible d'évaluer plusieurs fois le même mouvement, ce qui représente une perte de temps. Pour pallier ce problème, il suffit de se doter d'une structure de données capable de contenir à quelle itération on a évalué l'ajout d'une tâche v à l'horaire correspondant à c et de vérifier avant d'évaluer un mouvement s'il a déjà été évalué à l'itération courante.

Tout d'abord, pour la partie qui suit, on revient à la terminologie initiale c'est-à-dire que l'on utilisera une couleur c pour faire référence à un horaire et un sommet v pour une rotation. En effet, le déplacement d'une rotation d'un horaire à un autre est plus intuitif et facile à comprendre que le changement de couleur d'un sommet quoique les deux représentations soient tout à fait équivalentes. Voici donc en quoi consiste chacun des voisinages utilisés :

1. Ce voisinage a pour but de déplacer une rotation v d'un horaire source c dans lequel on retrouve au moins un chevauchement et où v est impliqué dans un chevauchement dans c, vers un horaire de destination c' quelconque. Ce voisinage règle donc les problèmes de chevauchements.
2. Ce voisinage a pour but de déplacer une rotation v d'un horaire source c comportant trop de crédits vers un horaire de destination c' en ayant moins que c. Ce voisinage règle donc les problèmes dûs aux crédits en excès.
3. Ce voisinage a pour but de déplacer une rotation v d'un horaire source c n'ayant pas suffisamment de jours de congé et où le retrait de v de c libère au moins une journée travaillée, vers un horaire de destination c' en ayant plus que c. Ce voisinage s'adresse aux problèmes de jours de congé insuffisants.
4. Ce voisinage a pour but de déplacer une rotation v d'un horaire source c ayant trop de jours travaillés consécutivement vers un horaire de destination c' en ayant moins que c et ce, en autant que le retrait de v à c diminue le nombre de jours travaillés consécutifs. Ce voisinage vise donc les problèmes dûs au nombre de jours consécutifs en excès.
5. Ce voisinage a pour but de déplacer une rotation v d'un horaire source c ayant plus de crédits que le minimum requis vers un horaire de destination c' en ayant moins que le minimum requis. Ce voisinage vise donc les problèmes dûs au nombre de crédits insuffisants.
6. Ce voisinage a pour but de déplacer une rotation v d'un horaire source c ayant plus de crédits que la moyenne du nombre de crédits vers un horaire de destination c' ayant moins de crédits que c. Ce voisinage vise donc à uniformiser les horaires du point de vue du nombre de crédits.

7. Ce voisinage a pour but de déplacer une rotation v d'un horaire source c ayant moins de jours de congé que la moyenne du nombre de jours de congé vers un horaire de destination c' ayant plus de jours de congé que c. Ce voisinage vise donc à uniformiser les horaires du point de vue du nombre de jours de congé.

Pour un mouvement recherché $M(v, c, c')$, le tableau 3.1 résume les critères permettant de définir les sept voisinages. La colonne No. indique le voisinage, la suivante indique la composante de l'objectif concernée et les autres les critères pour trouver des mouvements satisfaisants.

Tableau 3.1 – Définition des voisinages

No.	Composante	Critère sur c	Critère sur c'	Critère sur v
1	1	$F_1^c > 0$	-	$\Delta r_1^i(v, c) < 0$
2	2	$NbCr^c > MaxCr$	$NbCr^{c'} \leq NbCr^c$	-
3	3	$F_3^c > 0$	$NbCg^{c'} \geq NbCg^c$	$\Delta r_3^i(v, c) > 0$
4	4	$F_4^c > 0$	$F_4^{c'} < F_4^c$	$\Delta r_4^i(v, c) < 0$
5	2	$NbCr^c > MinCr$	$NbCr^{c'} < MinCr$	-
6	5	$NbCr^c > \mu_{Cr}$	$NbCr^{c'} < NbCr^c$	-
7	6	$NbCg^c < \mu_{Cg}$	$NbCg^{c'} > NbCg^c$	-

L'utilisation d'un voisinage en particulier dépend du contexte ou de l'état dans lequel se trouve l'algorithme. En effet, lors de la première phase de l'algorithme, seuls les cinq premiers voisinages sont utilisés tandis que, dans la deuxième phase, les sept le sont. Dans cette dernière phase, lorsque l'on cherche à uniformiser et que la solution est réalisable, on choisit au hasard entre les deux derniers voisinages. Lorsque la solution n'est pas réalisable, on peut se permettre avec une probabilité d'environ 25% de choisir un mouvement issu des deux derniers voisinages dans le but de diversifier la recherche, mais sinon on utilise les cinq premiers pour redevenir réalisable. D'un autre côté, lors de la première phase, les voisinages sont utilisés séquentiellement et on ne change que lorsque celui dans lequel on cherche un mouvement est vide. On utilise le dernier voisinage ayant généré un mouvement comme point de départ pour en générer un autre. Dans la deuxième phase, on choisit au hasard le voisinage à utiliser.

3.2.8 Solution initiale

L'objectif de l'implantation de l'algorithme de recherche taboue est de fournir une partition ou une solution initiale pour démarrer l'agrégation dynamique de contraintes, mais le premier a lui aussi besoin d'une solution initiale pour débuter. La qualité de cette dernière n'a pas une grande importance, car même avec une solution de très mauvaise qualité, c'est-à-dire une solution qui viole un très grand nombre de contraintes, l'algorithme converge très rapidement.

Si le problème ne comportait pas de contraintes difficiles à modéliser en termes de graphe comme le nombre minimum de jours de congé, l'heuristique RLF aurait été un très bon choix étant donné les bons résultats qu'elle obtient dans un tel contexte dans un temps raisonnable. Il aurait été encore plus judicieux d'utiliser un algorithme de coloration séquentiel exact, qui est polynomial pour les graphes d'intervalles, mais les préaffectations rendent le problème NP-difficile. Par ailleurs, en utilisant RLF sur un problème relaxé de certaines contraintes qui brisent la structure du graphe comme le nombre minimal et maximal de crédits, le nombre minimal de jours de congé et le nombre maximal de jours travaillés consécutifs, il n'est pas garanti que l'on obtienne une bonne solution en tenant compte a posteriori de ces contraintes. Il en va de même pour toutes les autres heuristiques présentées précédemment. Étant donné la nature du problème à résoudre, la qualité hasardeuse des solutions de toutes les heuristiques présentées en début de chapitre porte à croire qu'il ne s'agit pas d'une bonne approche.

La solution initiale a donc été générée en attribuant une couleur au hasard $C(v)$ à chacun des sommets $v \in V$ où $C(v) \in [1, k]$ est entier et v est une rotation. Mentionnons que cette manière de procéder est différente de Random précédemment présentée qui attribuait au hasard l'ordre dans lequel on colore les sommets plutôt que la couleur directement. L'approche utilisée a l'avantage d'être la méthode sans biais la plus rapide puisqu'aucune vérification ou validation n'est nécessaire. Ceci

implique donc qu'il est possible qu'une ou plusieurs rotations chevauchent une tâche préassignée. Ce genre de violation de contrainte devrait disparaître rapidement de la solution puisque les mouvements définis précédemment permettent de régler un tel conflit sans en créer de nouveau.

3.2.9 Liste taboue

Une liste taboue a été utilisée pour empêcher l'algorithme de stagner dans des minimums locaux. Des mouvements ont donc été interdits pendant un certain nombre d'itérations I. Plus précisément, étant donné le changement de couleur du sommet $v : C_i(v) \rightarrow C_{i+1}(v)$ où $C_i(v) \neq C_{i+1}(v)$, on interdira de donner à nouveau la couleur $C_{i+1}(v)$ au sommet v pendant I itérations. Le nombre I est fixé dynamiquement au cours de l'algorithme selon la fonction $I = \sqrt{|N(s)|}$. En utilisant cette taille on se prémunit contre le cyclage adéquatement sans toutefois créer trop de restrictions, ce qui aurait pour effet d'augmenter l'objectif (Hertz *et al.*, 1995). Cette manière de procéder n'est pas vraiment restrictive puisqu'elle permet de changer à nouveau la couleur du sommet v venant d'être coloré en autant que cette couleur ne soit pas taboue pour ce sommet. Le rôle de la liste taboue est donc uniquement de prévenir le cyclage et ne prend pas en considération l'intensification de la recherche autour d'une solution nouvellement obtenue.

CHAPITRE 4
GÉNÉRATION DE COLONNES ET AGRÉGATION DYNAMIQUE DE CONTRAINTES

Ce chapitre est dédié à la description des deux méthodes heuristiques basées sur la programmation mathématique qui sont utilisées pour résoudre le problème, soit la génération de colonnes et l'agrégation dynamique de contraintes. Or, ces deux méthodes reposent sur une décomposition de Dantzig et Wolfe (1960) du problème et ainsi, partagent une formulation qui ne varie pas beaucoup d'une méthode à l'autre. La première section de ce chapitre décrira tout d'abord la modélisation du problème et décrira ensuite la méthode de génération de colonnes. Quant à elle, la dernière section présentera l'application de l'agrégation dynamique de contraintes.

4.1 Modélisation du problème

On présente ici un modèle de partitionnement d'ensemble utilisé pour résoudre le problème CBMPE. La formulation introduite ici est semblable à celle qui a été présentée précédemment dans le chapitre de revue de littérature, mais les particularités du problème de construction des blocs mensuels personnalisés y sont explicitées. D'une part, ce modèle est mieux adapté à la génération de colonnes et à l'agrégation dynamique étant donné son nombre réduit de contraintes par rapport au premier. En effet, les contraintes de couverture des tâches sont réduites aux rotations seulement et les contraintes sur l'unicité de l'horaire de chacun des employés considérés comme anonymes sont d'emblée agrégées. D'autre part, cette formulation réflète mieux, ou plutôt exactement, le problème à résoudre. Ainsi, l'objectif, même s'il pouvait être intégré implicitement au modèle précédent, est énoncé de manière explicite dans la

formulation qui suit. Voici donc la notation utilisée pour formuler le problème suivie de la dite formulation.

- E : Ensemble des employés ;
- E^p : Sous-ensemble d'employés composé de ceux ayant une ou plusieurs tâches qui leur sont préaffectées : $E^p \subseteq E$;
- E^a : Sous-ensemble d'employés composé de ceux n'ayant aucune tâche préaffectée : $E^a \subseteq E,\ E^a \cup E^p = E$; tous les éléments de cet ensemble sont considérés identiques ;
- $\bar{E}$: Ensemble d'employés composé des employés de E^p avec un employé supplémentaire que l'on notera *anonyme* ;
- Ω_e : Ensemble des horaires admissibles pour tout employé $e \in \bar{E}$.
- V : Ensemble des rotations à affecter ;
- a_{vh} : Paramètre qui vaut un si l'horaire h couvre la rotation v et zéro sinon ;
- x_h^e : Variable qui vaut un si l'horaire h est choisi pour l'employé e et zéro sinon ;
- $NbCr_h$: Nombre de crédits dans l'horaire h ;
- $NbCg_h$: Nombre de jours de congé dans l'horaire h ;
- μ_{cr} : Moyenne du nombre de crédits par employé ;
- μ_{cg} : Moyenne du nombre de jours de congé par employé ;
- α_{cr} : Poids pour les crédits ;
- α_{cg} : Poids pour les jours de congé.

Le problème de construction des blocs mensuels peut se formuler comme :

Minimiser

$$\sum_{e \in \bar{E}} \sum_{h \in \Omega_e} \left(\alpha_{cr} \frac{(NbCr_h - \mu_{cr})^2}{|E|} + \alpha_{cg} \frac{(NbCg_h - \mu_{cg})^2}{|E|} \right) x_h^e \tag{4.1}$$

Sujet aux contraintes :

$$\sum_{e\in\bar{E}}\sum_{h\in\Omega_e} a_{vh}x_h^e = 1 \qquad \forall v \in V \tag{4.2}$$

$$\sum_{h\in\Omega_e} x_h^e = |E^a| \qquad e = anonyme \tag{4.3}$$

$$\sum_{h\in\Omega_e} x_h^e = 1 \qquad \forall e \in E^p \tag{4.4}$$

$$x_h^e \in \{0,1\} \qquad \forall e \in \bar{E} \text{ et } \forall h \in \Omega_e. \tag{4.5}$$

On cherche donc à minimiser une somme pondérée de la variance des crédits et des jours de congé (4.1) en s'assurant que toutes les tâches sont couvertes (4.2). Les contraintes (4.3) nous assurent que le nombre d'horaires affectés à des employés considérés comme anonymes est égal au nombre de ces derniers, tandis que les contraintes (4.4) imposent l'octroi d'un horaire à chacun des employés considérés comme identifiables. Finalement, les contraintes (4.5) imposent l'intégrité sur les variables d'horaire.

Ce modèle comporte quelques difficultés majeures empêchant l'utilisation de la génération de colonnes comme méthode de résolution. Il est cependant possible de contourner ces difficultés en ayant recours à une approximation du modèle qui permet de conserver le comportement général de ce dernier en dépit de quelques petits désavantages. Ainsi, les quelques paragraphes qui suivent décriront les difficultés du modèle ainsi que la nature des approximations faites.

4.1.1 Une approximation du modèle

D'abord il est facile de calculer μ_{cr} si on définit Cr_v comme le nombre de crédits d'une tâche v et V' comme l'ensemble des rotations, tâches préassignée et périodes de vacances. On trouve ainsi facilement que $\mu_{cr} = \sum_{v\in V'} Cr_v/|E|$. D'une part, contrairement à la valeur du terme μ_{cr} qui est connue, la valeur du terme μ_{cg} ne se calcule pas facilement. En effet, une des principales difficultés de ce modèle réside dans le

fait que μ_{cg} est en fait une fonction : $\mu_{cg} = \sum_{e \in \bar{E}} \sum_{v \in \Omega_e} NbCg_v x_v^e / |E|$ où x_v^e sont les valeurs des variables dans la solution optimale qui ne sont pas connues a priori. On peut outrepasser cette difficulté en approchant μ_{cg} à l'aide de la solution retenue de l'algorithme de recherche taboue dont le temps de calcul ne représente en moyenne qu'une petite partie du temps total consacré à résoudre le problème.

D'autre part, puisque l'on cherche à résoudre le problème par génération de colonnes et qu'avec cette méthode, l'une des manières les plus efficaces de résoudre le sous-problème est de le formuler comme un problème de plus court chemin avec ressources et de le résoudre par programmation dynamique, on est limité dans la modélisation du sous-problème. En effet, les deux composantes de l'objectif demeurent non linéaires au niveau du sous-problème en plus de ne pas être non-décroissantes. On ne peut donc pas utiliser la fonction objectif telle que formulée en (4.1) et on aura plutôt recours à une approximation. Or, plusieurs manières d'approcher un tel objectif pourraient être utilisées pour résoudre le problème, mais on se concentrera sur deux d'entre elles.

Premier modèle : fonction escalier

Il est d'abord possible de remplacer la fonction objectif par une fonction en escalier avec Z paliers. On peut définir cette fonction d'abord à l'aide de M^z blocs (marches) où $z = 1, \ldots, Z$ et où Z est ce que l'on appellera le nombre de niveaux de flexibilité :

$$M^z = \left\{ (NbCr, NbCg) |\ NbCr \in [\mu_{cr} - l_{cr}^z, \mu_{cr} + u_{cr}^z],\ NbCg \in \left[\mu_{cg} - l_{cg}^z, \mu_{cg} + u_{cg}^z\right] \right\}$$

Les valeurs des paramètres définissant la largeur des paliers sont croissantes, c'est-à-dire que $l_{cr}^z < l_{cr}^{z+1}$, $u_{cr}^z < u_{cr}^{z+1}$, $l_{cg}^z < l_{cg}^{z+1}$ et $u_{cg}^z < u_{cg}^{z+1}$, $\forall z \in \{1, \ldots, Z-1\}$. Pour obtenir une fonction, on construit Z domaines tels que $D^1 = M^1$ et $D^z = M^z \setminus M^{z-1}$, $\forall z \in \{2, \ldots, Z\}$. Pour obtenir la pénalité P^z à imposer à un horaire on utilise la fonction suivante :

$$P^z = \left(\alpha_{cr} \frac{\left(\beta_{cr}^z - \mu_{cr}\right)^2}{|E|} + \alpha_{cg} \frac{\left(\beta_{cg}^z - \mu_{cg}\right)^2}{|E|} \right) \tag{4.6}$$

où $\beta^z_{cr} = \max\{l^z_{cr}, u^z_{cr}\}$ et $\beta^z_{cg} = \max\{l^z_{cg}, u^z_{cg}\}$. Ainsi, la fonction objectif (4.1) est remplacée par :

$$\sum_{e \in \bar{E}} \sum_{h \in \Omega_e} P^{z(h)} x^e_h \tag{4.7}$$

où $z(h)$ est le bloc tel que $(NbCr_h, NbCg_h) \in D^{z(h)}$.

Cette façon d'approximer le problème rend donc les coûts constants sur les arcs du réseau utilisé pour modéliser le sous-problème, ce qui permet l'utilisation de la programmation dynamique comme méthode de résolution. La figure 4.1 illustre de manière générale la construction des blocs M^z pour les crédits ou les jours de congé. On y voit entre autres qu'il n'est pas imposé que les blocs aient tous la même largeur et aussi que les blocs sont imbriqués les uns dans les autres. En effet, les blocs ont été construits dans le modèle tels que le montre la figure 4.1, c'est-à-dire que l'on a construit 5 régions d'égale largeur, d'une part, entre la borne inférieure et la moyenne et d'autre part, entre la moyenne et la borne supérieure. Comme le sous-entend l'affirmation précédente, les régions doivent être construites de manière à ce que $\mu_{cr} - l^5_{cr} = MinCr$, $\mu_{cr} + u^5_{cr} = MaxCr$, $\mu_{cg} - l^5_{cg} = MinCg$ et $\mu_{cg} + u^5_{cg} = MaxCg$ où $MinCr$ est la borne inférieure sur les crédits, $MaxCr$ est la borne supérieure sur les crédits, $MinCg$ est la borne inférieure sur les jours de congé, $MaxCg$ est la borne supérieure sur les jours de congé et où toutes ces bornes sont imposées par la convention collective, sauf dans le cas de $MaxCg$ qui est déduite de cette dernière.

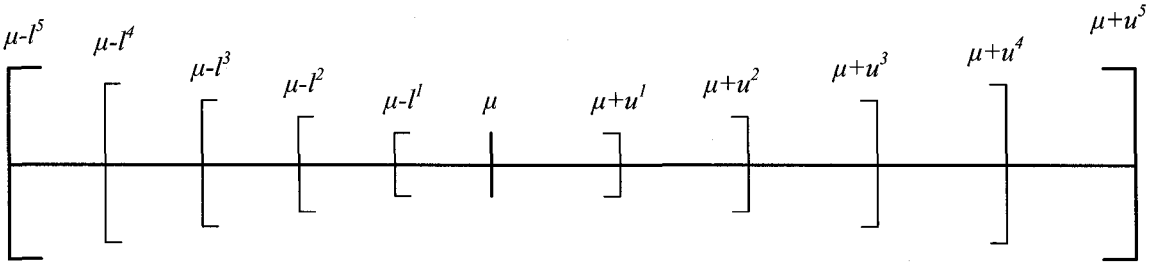

Figure 4.1 – Illustration des niveaux de flexibilité

Par ailleurs, la figure 4.2 illustre bien la définition des domaines D^z et leur pénalité

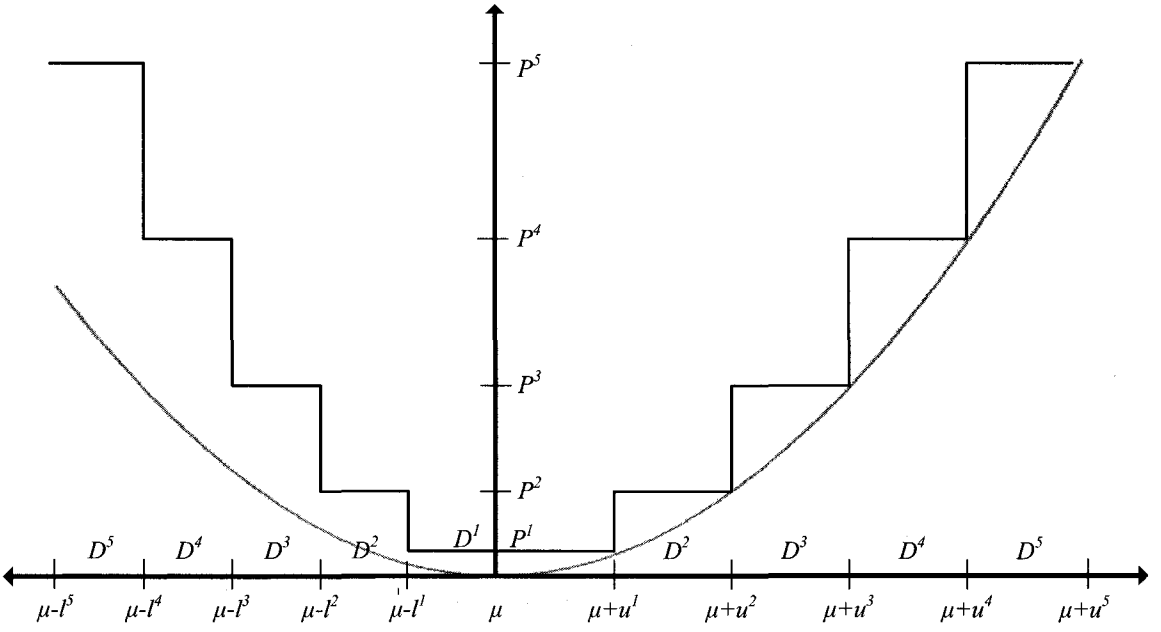

Figure 4.2 – Illustration de la fonction objectif

P^z correspondante. On voit que la fonction en escalier (4.6) a tendance à suivre le comportement de la fonction quadratique (4.1), ce qui est un des aspects souhaités de l'approximation. Cependant, la partie gauche de la fonction en escalier s'écarte un peu de la fonction quadratique en raison de l'introduction d'une fonction $\max\{x\}$ pour la détermination des β^z.

Deuxième modèle : fonction objectif tronquée

Dans un autre ordre d'idées, il est possible de remplacer la fonction objectif (4.1) par une fonction de type $\max^2\{0, x\}$ de manière à ce que cette dernière soit non-décroissante par rapport à $NbCr_h$ et $NbCg_h$. Cette fonction engendrera toujours des coûts évoluant de manière non linéaire au niveau du sous-problème, mais ils seront plus faciles à traiter. On abordera les raisons qui font en sorte que la fonction (4.8) est plus facile à traiter que la fonction (4.1) dans la section sur la résolution du sous-problème. À l'instar de la précédente approximation, la présente ne nécessite pas la

construction d'éléments au préalable et on peut directement introduire la fonction qui remplacera (4.1) :

$$\sum_{e\in\bar{E}}\sum_{h\in\Omega_e}\left(\alpha_{cr}\frac{\max^2\{0,(NbCr_h-\mu_{cr})\}}{|E|}+\alpha_{cg}\frac{\max^2\{0,(NbCg_h-\mu_{cg})\}}{|E|}\right)x_h^e. \quad (4.8)$$

Cette approximation de (4.1) pénalisera donc fortement les horaires comportant trop de crédits ou de jours de congé par rapport à la moyenne, mais aucunement ceux n'en ayant pas suffisamment, toujours par rapport à la moyenne. Cependant, puisque les excès de crédits ou de jours de congé sont fortement pénalisés, les horaires générés devraient comporter un nombre de crédits et de jours de congé légèrement au dessus de la moyenne ou en dessous de cette dernière. Les horaires dont ces quantités sont clairement en dessous de la moyenne ne devraient pas apparaître trop souvent puisque les horaires dont les excès de crédits et de jours de congé, qui compenseront ce manque, auront un coût élevé.

4.2 Génération de colonnes

La génération de colonnes est une méthode itérative qui alterne entre deux étapes soit la résolution du problème maître restreint (PMR) et la résolution du sous-problème (SP). Le PMR est celui formulé par (4.1)-(4.5) dans lequel tous les horaires possibles ne sont pas connus et dont les contraintes d'intégrité (4.5) sont relaxées. On résout le PMR par la méthode du simplexe primal de manière à obtenir des solutions primale et duale. À partir des valeurs des variables duales, il est possible de générer des variables ou colonnes dont on peut évaluer le coût réduit à l'aide d'une modélisation réseau du sous-problème intégrant les variables duales du PMR (voir §4.2.1). S'il n'existe pas de colonnes ayant un coût réduit négatif, l'optimalité de la relaxation linéaire du problème maître est atteinte. Sinon on intègre les colonnes de coût réduit négatif au PMR et on démarre une nouvelle itération avec la résolution du nouveau PMR. Pour la plus grosse instance, en raison du nombre excessivement

élevé de pivots nécessaire à l'obtention d'une solution par l'algorithme du simplexe, on a plutôt opté pour l'utilisation de la méthode des points intérieurs pour résoudre le PMR à chaque itération. Cette dernière a permis des gains en temps de calcul importants pour cette instance.

Pour trouver une solution entière, on utilisera une méthode de séparation et génération progressive (*Branch-and-Price*) dont le fonctionnement est montré succintement dans la figure 4.3. Il s'agit d'une méthode de séparation et évaluation progressive dans laquelle les bornes inférieures sur la valeur de l'objectif sont calculées par génération de colonnes. Cependant, étant donné la très grande taille des instance traitées, on aura recours à une version heuristique de l'approche. En effet, on fixe des variables x_h^e à un si leur valeur dépasse un certain seuil. Cette stratégie accélère la résolution en raison d'un temps d'exploration de l'arbre de branchement considérablement réduit. Il y a cependant un mince risque qu'en procédant ainsi il ne soit plus possible, après avoir fixé plusieurs variables, d'obtenir une solution réalisable. Il est possible de limiter cet effet en choisissant un seuil relativement élevé. De plus, étant donné la grande taille des instances et la multitude de manières possibles de couvrir les tâches, ce phénomène ne s'est jamais produit au cours de tous les essais. On utilise aussi une autre stratégie de branchement heuristique qui branche plutôt sur les séquences de tâches plutôt que sur les valeurs des variables. En effet, on analyse la solution courante et on cherche des paires de tâches consécutives qui ont tendance à être fréquemment utilisées. Ainsi, on imposera plusieurs séquences de tâches simultanément résultant en un seul noeud fils. Toute colonne ne respectant pas l'imposition de cette séquence de tâche sera écartée. De plus, puisque l'on utilise plus d'une méthode de branchement, des mécanismes de pondérations sont mis en place afin de choisir l'une ou l'autre des méthodes à une itération donnée.

4.2.1 Modélisation du sous-problème

Comme on l'a déjà mentionné, on a recours à une méthode de génération de colonnes pour résoudre le problème de construction des horaires du personnel navigant.

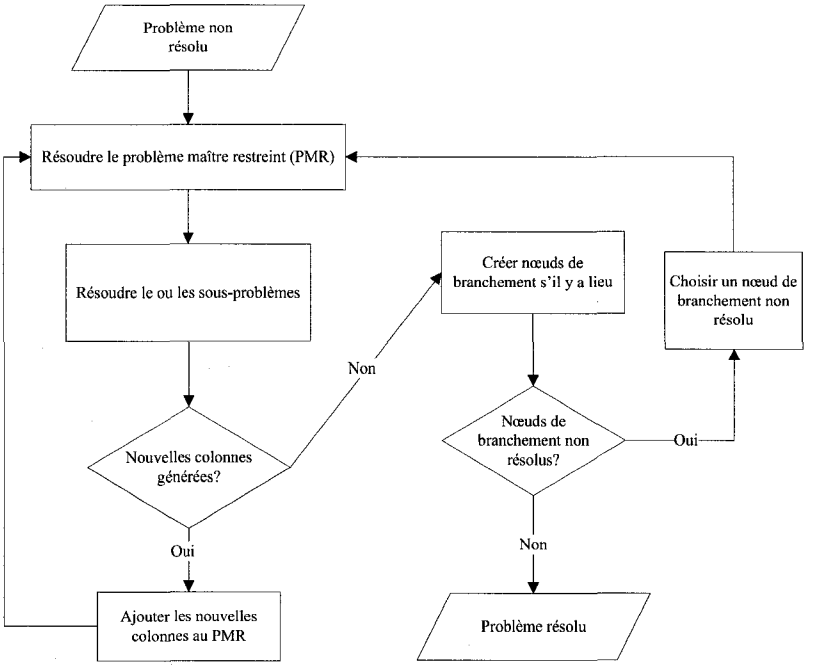

Figure 4.3 – Algorithme de séparation et génération progressive

Pour ce faire, on a besoin d'un problème maître que l'on vient de décrire et d'un sous-problème que l'on s'efforcera de décrire ici. Comme on l'a mentionné dans le chapitre sur la revue de la littérature, il est très souvent possible de formuler le sous-problème comme un problème de plus court chemin avec contraintes de ressources dans un graphe temporel. C'est avec une telle modélisation que l'on abordera la génération d'horaires mensuels valides. En effet, les contraintes locales ne sont considérées que lors de la résolution du ou des sous-problèmes et la plupart de ces dernières sont modélisées avec des ressources. On utilisera le premier modèle d'approximation à l'aide de la fonction en escalier pour décrire le sous-problème.

Présentation du modèle

Jusqu'ici, on a toujours fait référence au sous-problème comme étant un élément unique alors qu'en fait, pour le problème CBMPE, on en a plusieurs. De plus, on

aura toujours un réseau par sous-problème, un sous-problème en particulier étant modélisé par un réseau. Il y aura donc un réseau ou graphe par employé identifiable, c'est-à-dire un employé ayant au moins une tâche préassignée ou une période de vacances, et il existe un réseau pour tous les autres employés. Plus précisément, on aura $\left|\bar{E}\right|$ réseaux ou sous-problèmes. Chacun de ces réseaux sera dupliqué plusieurs fois et légèrement modifié en fonction du nombre de niveaux de flexibilité Z choisi ($Z = 5$ dans notre cas). On obtient ainsi un grand nombre de réseaux, chacun étant capable de générer des horaires réalisables. On peut modéliser un problème de plus court chemin de manière plus formelle en définissant d'abord l'indice $z \in Z$ comme le niveau de flexibilité du réseau, e comme l'indice identifiant l'employé et prenant une valeur particulière, *anonyme*, dans le cas d'un réseau anonyme.

Semblablement au précédent chapitre, on a $G_e^z = (V_e^z, A_e^z)$ qui est le graphe composé de l'ensemble des sommets V_e^z et de l'ensemble des arcs A_e^z. On aura un petit sous-ensemble d'arcs $I_e^z \subset A_e^z$ où chaque arc de ce sous-ensemble correspond à une tâche préassignée ou une période de vacances pour l'employé e et qui sont des tâches imposées. Ce sous-ensemble sera vide dans le cas où $e = anonyme$. On définit s_e^z comme la source et t_e^z comme le puits où $\{s_e^z, t_e^z\} \in V_e^z$ pour chacun des réseaux, ces deux derniers étant des sommets particuliers. On introduit aussi R comme étant l'ensemble des ressources utilisées et t_{ij}^r la consommation de la ressource $r \in R$ sur l'arc $(i,j) \in A_e^z$. Ceci permet d'introduire $[a_i^r, b_i^r]$ comme étant la plage de validité de la ressource r au sommet i. Le coût réduit d'un arc (i,j) sera noté c_{ij} et la manière de le calculer sera présentée plus loin. Du côté des variables, soit x_{ij} le flot sur l'arc $(i,j) \in A_e^z$ et T_i^r la variable qui donne la valeur de la ressource $r \in R$ au sommet $i \in V_e^z$. Le problème de plus court chemin avec contraintes de ressources peut se formuler comme suit.

Minimiser :

$$\sum_{(i,j) \in A_e^z} c_{ij} x_{ij} \tag{4.9}$$

Sujet aux contraintes :

$$\sum_{j:(i,j)\in A_e^z} x_{ij} - \sum_{j:(i,j)\in A_e^z} x_{ji} = 0 \qquad \forall i \in V_e^z \setminus \{s_e^z, t_e^z\} \tag{4.10}$$

$$\sum_{j:(i,j)\in A_e^z} x_{ij} = 1 \qquad i = s_e^z \tag{4.11}$$

$$\sum_{j:(i,j)\in A_e^z} x_{ji} = 1 \qquad i = t_e^z \tag{4.12}$$

$$x_{ij}\left(T_i^r + t_{ij}^r - T_j^r\right) \leq 0 \qquad \forall (i,j) \in A_e^z, \forall r \in R \tag{4.13}$$

$$a_i^r \leq T_i^r \leq b_i^r \qquad \forall i \in V_e^z, \forall r \in R \tag{4.14}$$

$$x_{ij} = 1 \qquad \forall (i,j) \in I_e^z \tag{4.15}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A_e^z \tag{4.16}$$

On cherche donc le chemin le moins coûteux ou le plus court chemin entre la source et le puits (4.9). Les contraintes (4.10)-(4.12) et (4.16) assurent la conservation de l'unique unité de flot dans le réseau tandis que le bloc de contraintes (4.15) garantit que les tâches préassignées et les vacances sont effectuées. On verra (§4.2.2) que les contraintes (4.15) sont très faciles à traiter. Les contraintes (4.13) nous assurent que la consommation des ressources sur les arcs est juste. Quant à elles, les contraintes (4.14) nous assurent que les ressources se situent dans les plages de valeurs permises.

Description du graphe

Puisqu'on a un graphe temporel tel que certains arcs corespondent à des tâches, il est facile de représenter l'horaire mensuel d'un employé à l'aide d'un chemin dans ce même graphe. On voit donc qu'il y a un partage d'information du problème maître vers le sous-problème et que la notion d'horaire et de chemin peut être équivalente. Pour décrire ce partage, on décrira d'abord les noeuds constituant les graphes G_e^z, ce qui permettra de décrire les arcs et il sera finalement possible de décrire les coûts c_{ij}. La figure 4.3 montre comment s'agencent les noeuds et les arcs dont la description suit.

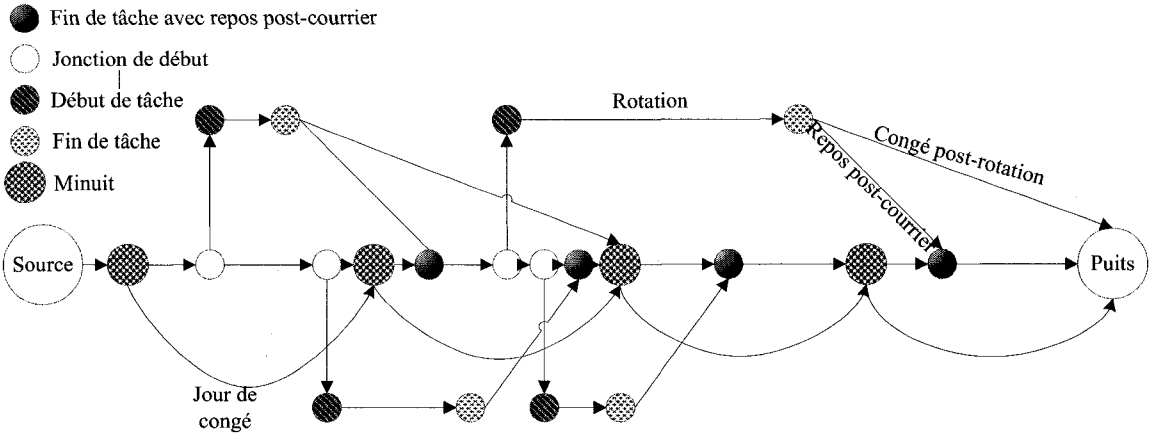

Figure 4.4 – Exemple de réseau anonyme utilisé en génération de colonnes

On peut regrouper les noeuds selon leur type au nombre de sept :

- Source : Origine du graphe orienté ou point de départ de tous les chemins.
- Minuit : Noeud sur l'axe du temps représentant le début et la fin d'une journée.
- Jonction de début : Noeud sur l'axe du temps associé avec un événement dans la période correspondant au début d'une rotation, une période de vacances ou une tâche préassignée. Un tel noeud existe pour chacun de ces débuts de tâche, sauf si cette tâche débute à minuit, auquel cas le noeud minuit est plutôt utilisé.
- Début de tâche : Noeud associé au début d'une rotation, une période de vacances ou une tâche préassignée. On retrouve un noeud de ce type par tâche.
- Fin de tâche : Noeud associé à la fin d'une rotation, une période de vacances ou une tâche préassignée. On retrouve un noeud de ce type par tâche.
- Fin de tâche avec repos post-courrier : Noeud sur l'axe du temps correspondant à l'instant à partir duquel on considère qu'une tâche est terminée et qu'un employé peut en commencer une autre. On compte un noeud de ce type par rotation ou par tâche préassignée.
- Puits : Fin du graphe orienté ou destination de tous les chemins dans ce graphe. Ce noeud pour des raisons pratiques correspond au dernier noeud minuit, ce que l'on expliquera dans la section sur la résolution des sous-problèmes.

On termine la description de ce graphe avec les arcs qui relient les noeuds :

- Attente : Ces arcs effectuent la liaison entre les divers événements. Ces arcs relient ensemble tous les noeuds de l'axe du temps comme le noeud source, les noeuds minuit, les noeuds de jonction de début, les noeuds de fin de tâche avec repos post-courrier. L'ensemble de ces arcs constituent en fait l'axe du temps.
- Congé : Arc d'un minuit au minuit suivant représentant un jour de congé.
- Rotation : Arc d'un noeud début de tâche à un noeud fin de tâche représentant la réalisation de la tâche.
- Repos post-courrier : Arc d'un noeud fin de tâche à un noeud fin de tâche avec repos post-courrier représentant la réalisation d'un repos post-courrier. Chaque rotation et chaque tâche préassignée dispose d'un tel arc.
- Congé post-rotation : Arc d'un noeud fin de tâche au noeud minuit suivant le premier minuit croisé par l'arc de repos post-courrier. Si l'arc de repos post-courrier ne croise pas de noeud minuit, l'arc fin de tâche avec congé n'existe pas.
- Départ : Arc entre la source et le premier minuit ou entre la source et la première tâche préassignée. On ne retrouve qu'un seul arc de ce type par réseau.

Pour définir les coûts des arcs (i, j), on doit d'abord introduire les variables duales du problème maître. Soit $\sigma^{anonyme}$ la variable duale associée à la contrainte (4.3), $\sigma^e, \forall e \in E^p$, les variables duales associées aux contraintes (4.4) et $\pi^v, \forall v \in V$, les variables duales associées aux contraintes (4.2). Ceci nous permet d'introduire la notion de coût réduit d'un horaire ou d'une colonne. Étant donné un horaire h convenant à un employé $e \in \bar{E}$ et un niveau de flexibilité z, le coût réduit $\bar{c}_h^e$ de la variable associée à cet horaire se calcule comme :

$$\bar{c}_h^e = P^z - \sigma^e - \sum_{v \in V} a_{vh}\pi^v. \tag{4.17}$$

On peut interpréter le coût réduit d'un horaire, s'il est négatif, comme une mesure du potentiel de diminution de la valeur de l'objectif si on introduisait cet horaire dans le PMR, étant donné la base actuelle. Après avoir à nouveau résolu le PMR, on devrait

observer que la colonne tout juste introduite est en base et on peut à nouveau utiliser les variables duales pour générer des horaires intéressants. On peut ainsi itérer un nombre fini de fois pour atteindre l'optimalité de la relaxation linéaire du problème maître. Par conséquent, lorsque l'on génère des horaires avec un sous-problème, on doit chercher à minimiser son coût réduit. Pour ce faire, on peut utiliser la fonction objectif (4.9) pour qu'elle soit équivalente à (4.17) en plaçant convenablement P^z et les variables duales de (4.17) sur les arcs (i,j) de A_e^z. Plus précisément, on cherchera à définir c_{ij}, $\forall\ (i,j) \in A_e^z$. Donc, pour le réseau d'un employé $e \in \bar{E}$, pour un niveau de flexibilité z, tous les arcs auront un coût nul à l'exception des arcs suivants dont le coût sera :

- Arc de départ : $c_{oj} = P^z - \sigma^e$, $\forall\ (o,j) \in A_e^z$
- Tâche : Si on définit I'^z_{ve} comme l'ensemble des arcs représentant une rotation v dans le graphe G_e^z, alors $c_{ij} = -\pi^v$, $\forall (i,j) \in I'^z_{ve}, \forall v\ \in V$.

En faisant la somme des coûts de tous les arcs d'un chemin comme le mentionne (4.9), on obtient bel et bien la même formulation qu'en (4.17) où les coefficients a_{vh} sont unitaires, ce qui est juste puisque les tâches sont couvertes une et une seule fois.

Utilisation des ressources

Dans un autre ordre d'idées, on utilise les contraintes (4.13)-(4.14) pour modéliser toutes les contraintes locales hormis les chevauchements, c'est-à-dire les contraintes sur le nombre maximal et minimal de crédits, la contrainte sur le nombre minimal de jours de congé et la contrainte sur le nombre maximal de jours travaillés consécutifs. On a utilisé quatre ressources pour les contraintes (4.13)-(4.14) soit une pour les crédits, une pour les jours de congé, une pour les jours travaillés consécutifs et une pour corriger une légère lacune du graphe. Voici donc une brève description de ressources utilisées.

- NbCrMax : Il s'agit de la ressource qui mesure le nombre maximal de crédits lors d'un horizon de planification. Cette ressource est nulle au départ et croît jusqu'au noeud puits.

- NbCrMin : Cette ressource est utilisée pour que la contrainte sur le nombre minimal de crédits soit respectée. En effet, on ne peut utiliser la ressource NbCrMax en raison du bloc de contraintes (4.13) (Gamache et Soumis, 1998; Irnich et Desaulniers, 2005). Cette ressource prend la valeur du nombre minimal de crédits à octroyer au noeud source et décroît quand une tâche est effectuée.
- NbCgMax : Cette ressource mesure le nombre maximal de jours de congé à allouer. Bien qu'il n'y ait pas de contrainte sur le nombre maximal de jours de congé, on a besoin de cette ressource pour évaluer l'objectif.
- NbCgMin : Pour les mêmes raisons que pour NbCrMin, on a besoin de cette ressource pour s'assurer que la contrainte sur le nombre minimal de jours de congé est respectée.
- NbJCMax : Il s'agit de la ressource qui mesure le nombre de jours travaillés consécutivement. On remet cette ressource à zéro chaque fois que NbCgMax augmente d'une unité.
- AjustC : Puisqu'il existe habituellement deux chemins distincts entre un minuit et le suivant, il est possible que les ressources NbCgMax et NbCgMin n'évoluent pas correctement. En effet, seul le chemin direct via l'arc minuit à minuit comptabilisera une journée de congé, tandis que le chemin empruntant les arcs d'attente ne le fera pas. Or lorsqu'un horaire dépasse la moyenne des jours de congé, il devient avantageux d'utiliser le deuxième chemin, ce qui devrait être interdit. On peut donc mettre cette ressource binaire à un en sortant d'un noeud minuit et la remettre à zéro en effectuant une tâche de manière à ce qu'il soit possible de passer la fenêtre du minuit suivant qui exige que la ressource soit à zéro. Ainsi, il devient impossible de franchir un noeud minuit si on arrive d'un arc d'attente et que l'on n'a pas effectué une tâche depuis le dernier départ d'un noeud minuit.

On peut maintenant décrire la consommation des ressources $t^r_{ij}, \forall (i,j) \in A^z_e, \forall r \in R$ tel que le fait le tableau 4.1.

Tableau 4.1 – Consommation des ressources

Arcs	Ressource					
	NbCr Max	NbCr Min	NbCg Max	NbCg Min	NbJC Max	AjustC
Attente	0	0	0	0	+1 si départ de minuit, 0 sinon	+1 si départ de minuit, 0 sinon
Congé	0	0	+1	-1	Remise à 0	0
Rotation v	$+NbCr_v$	$-NbCr_v$	0	0	+1 $\forall$ minuit croisé	-1
Repos post-courrier	0	0	0	0	+1 si croise un minuit, 0 sinon	+1 si croise un minuit, 0 sinon
Congé post-rotation	0	0	1	-1	Remise à 0	0
Départ	0	0	0	0	0	0

Quant à lui, le tableau 4.2 montre les bornes a_i^r et b_i^r des fenêtres des ressources pour les ressources NbCrMin, NbCgMin et AjustC. Celles des autres ressources ne changent pas selon le type de noeud du graphe. Ces bornes sont $[0, Max_r^z]$ pour toutes les autres ressources et tous les noeuds ; Max_r^z est la borne supérieure pour la ressource r et le niveau de flexibilité z. De manière analogue, Min_r^z est la borne inférieure pour la ressource r et le niveau de flexibilité z.

Tableau 4.2 – Plage de validité des ressources

Noeud	Ressource		
	NbCrMin	NbCgMin	AjustC
Source	$[MinCr^z, MinCr^z]$	$[MinCg^z, MinCg^z]$	$[0, 0]$
Minuit	$[0, MinCr^z]$	$[0, MinCg^z]$	$[0, 0]$
Jonction de début Début tâche Fin post-courrier	$[0, MinCr^z]$	$[0, MinCg^z]$	$[0, 1]$
Fin tâche	$[0, MinCr^z]$	$[0, MinCg^z]$	$[0, 0]$
Puits	$[0, 0]$	$[0, 0]$	$[0, 0]$

4.2.2 Réseau personnalisé

Il est possible de se départir des contraintes (4.15) en effectuant certaines transformation au graphe $G_e^z = (V_e^z, A_e^z)$. En effet, puisque pour certains employés $e \in E^p$, les tâches préassignées et les périodes de vacances sont imposées, tout chemin dans le graphe doit absolument passer par les arcs représentant ces dernières. La figure 4.5 donne un exemple de réseau incluant les contraintes (4.15).

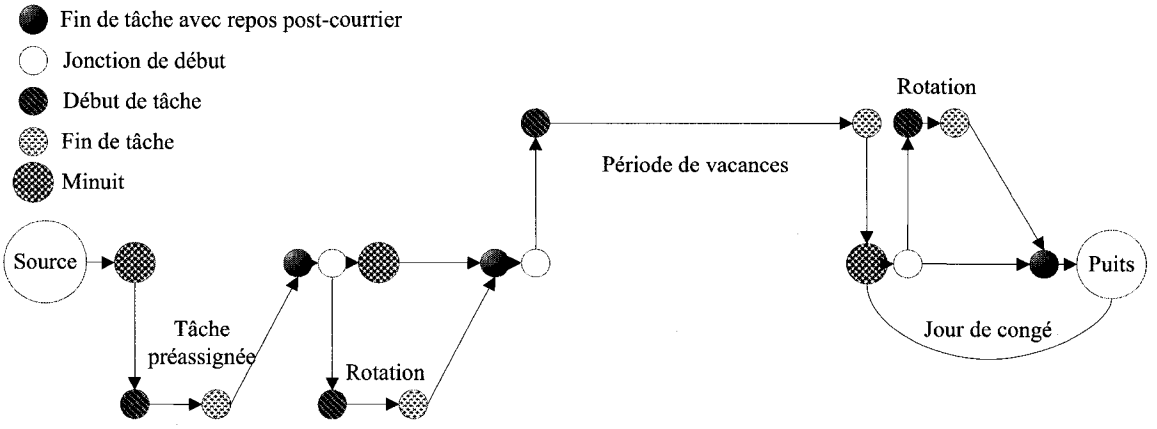

Figure 4.5 – Exemple de réseau personnalisé utilisé en génération de colonnes

4.2.3 Calcul des bornes pour les niveaux de flexibilité

Tout d'abord, il est important de définir la manière de discrétiser le domaine des ressources utilisées pour évaluer l'objectif. Ainsi, on a choisi de découper chacun des deux intervalles entre la moyenne et la borne supérieure absolue ainsi qu'entre la moyenne et la borne inférieure absolue en z parties préférablement égales. Rappellons d'abord que $MaxCr$, $MinCr$, $MaxCg$ et $MinCg$ sont les bornes inférieures et supérieures pour les jours de congé et les crédits et les bornes utilisées pour $z = Z$. Rappellons aussi que les incréments pour $z = Z$ sont définis par les précédentes bornes. Le calcul des incréments pour $z \in \{1, 2, ..., Z-1\}$ est

$$u_{cr}^{z} = \left\lfloor z\left(\frac{MaxCr - \mu_{cr}}{Z}\right)\right\rfloor \qquad \forall z \in \{1, 2, ..., Z-1\} \tag{4.18}$$

$$l_{cr}^{z} = \left\lfloor z\left(\frac{\mu_{cr} - MinCr}{Z}\right)\right\rfloor \qquad \forall z \in \{1, 2, ..., Z-1\} \tag{4.19}$$

$$u_{cg}^{z} = \left\lfloor z\left(\frac{MaxCg - \mu_{cg}}{Z}\right)\right\rfloor \qquad \forall z \in \{1, 2, ..., Z-1\} \tag{4.20}$$

$$l_{cg}^{z} = \left\lfloor z\left(\frac{\mu_{cg} - MinCg}{Z}\right)\right\rfloor \qquad \forall z \in \{1, 2, ..., Z-1\}. \tag{4.21}$$

Il reste à choisir une valeur Z de manière à ce que les réseaux issus du niveau de flexibilité le plus serré génèrent des horaires de très bonne qualité du point de vue de la qualité de vie des employés. En posant $Z = 5$ on obtient un nombre acceptable de sous-problèmes et les sous-problèmes les plus serrés permettent un écart-type d'un jour pour les congés et d'environ 120 minutes pour les crédits. Ainsi, n'importe quel horaire issu des sous-problèmes les moins flexibles sera de très bonne qualité. Une solution idéale ne contiendra que des horaires issus de tels sous-problèmes.

Quant à eux les poids α_{cr} et α_{cg} des composantes de l'objectif sont tels que l'on donne la même importance à la composante crédits qu'à la composante jours de congé de l'objectif en plus d'éviter la manipulation de trop grands nombres. Si on définit P_{cr}^{max} comme la valeur maximale que peut prendre la composante crédits de la pénalité d'un horaire, P_{cg}^{max} de manière analogue pour les jours de congé et que l'on exige que $P_{cr}^{max} = P_{cg}^{max}$, les deux composantes seront équilibrées. Ainsi le calcul des coefficients pour pondérer la pénalité d'un horaire se fait de la manière suivante :

$$\alpha_{cr} = \frac{P_{cr}^{max}}{\max\left\{\left(MaxCr^{Z} - \mu_{cr}\right)^{2}, \left(\mu_{cr} - MinCr^{Z}\right)^{2}\right\}} \tag{4.22}$$

$$\alpha_{cg} = \frac{P_{cg}^{max}}{\max\left\{\left(MaxCg^{Z} - \mu_{cg}\right)^{2}, \left(\mu_{cg} - MinCg^{Z}\right)^{2}\right\}}. \tag{4.23}$$

4.2.4 Résolution des sous-problèmes

La méthode la plus efficace pour réoudre le problème de plus court chemin avec contraintes additionelles de ressource est un algorithme de programmation dynamique (Irnich et Desaulniers, 2005). Tout d'abord, on utilise une étiquette pour définir les caractéristiques d'un chemin partiel, nous informant, entres autres, sur les ressources consommées et sur son coût. L'algorithme cherche donc à propager des étiquettes dans le réseau du noeud source jusqu'au noeud puits selon les arcs du réseau. Il est possible de comparer des étiquettes entre elles de manière à en réduire le nombre. Une étiquette a en dominera une autre b au noeud i si $T_i^r(a) \leq T_i^r(b),\ \forall r \in R$, et si $c_i(a) \leq c_i(b)$ où $c_i(l)$ est le coût du chemin partiel correspondant à l'étiquette l et $T_i^r(l)$ est la consommation de la ressource r au noeud i de l'étiquette l. De telles règles permettent d'éliminer un grand nombre d'étiquettes, mais quand le nombre de ressources est grand comme dans notre cas, ce n'est pas toujours suffisant.

En effet, pour accélérer la résolution des sous-problèmes, il est préférable de restreindre les règles de dominance à certaines ressources uniquement, ce qui implique que plusieurs étiquettes qui seraient normalement conservées dans une approche exacte seront rejetées. Ceci économise beaucoup de temps dans la résolution des sous-problèmes puisque plusieurs étiquettes et leurs successeurs ne seront pas propagées. Plusieurs chemins ne seront donc jamais générés en raison de ce type de dominance. Par contre, puisque l'on utilise deux ressources pour mesurer les crédits de vol et que les deux évoluent dans des sens opposés, c'est-à-dire qu'une augmente en se dirigeant vers la source tandis que l'autre diminue, il n'est pas intéressant de chercher à dominer sur ces deux ressources simultanément. En effet, une des deux ressources empêche toujours de dominer une étiquette par une autre sauf dans le cas où deux étiquettes ont le même nombre de crédits et de jours de congé.

Pour que les règles de dominance demeurent valides, c'est-à-dire qu'elles nous garantissent une solution optimale s'il en existe une, la fonction de prolongation des

coûts sur les arcs doit être non-décroissante. En effet, il est impossible de décider si une étiquette a ayant un coût et des consommation de ressources moindre que l'étiquette b sera meilleure que b s'il est possible que le coût de l'étiquette b devienne inférieur à celui de a après que les deux aient été propagées à un noeud successeur. En effet, il est impossible de prévoir, par exemple, qu'un chemin partiel plus coûteux deviendra le moins cher. C'est ce qui explique que l'on ait recours à des approximations de la fonction objectif (4.1) étant donné que le coût pouvait décroître par rapport à NbCrMax et NbCgMax.

Choix des sous-problèmes

Habituellement, lorsque le nombre de sous-problèmes est élevé, il est préférable de ne pas tous les résoudre à chaque itération. Soit une liste permanente complète de tous les sous-problèmes, on résout séquentiellement chaque sous-problème séquentiellement selon l'ordre dans lequel il se trouve dans la liste. Chaque sous-problème résolu est retiré de la liste et placé temporairement dans une autre. Une fois un critère d'arrêt atteint, on redéfinit un ordre au hasard dans la liste auxiliaire et on ajoute les sous-problèmes de cette dernière liste à la liste originale dans l'ordre tout juste déterminé. Ceci permet de ne pas favoriser un sous-problème plutôt qu'un autre. Le critère d'arrêt habituellement utilisé est de résoudre un nombre prédéterminé de réseaux ayant généré au moins une colonne de coût réduit négatif. Ceci implique donc que sur une liste de 100 sous-problèmes, étant donné que 40 d'entre eux ne permettent pas de générer des colonnes de coût réduit négatif, le fait d'exiger 60 succès revient à résoudre tous les sous-problèmes.

On peut remarquer rapidement que les réseaux étiquetés comme anonymes génèrent des horaires convenant à plusieurs personnes et, par conséquent, que ces réseaux ont le potentiel de satisfaire un plus grand nombre de contraintes. Il est donc intéressant de prioriser la résolution de ces derniers. On peut ainsi replacer toujours au début les réseaux étiquettés comme anonymes dans la liste permanente.

Il est cependant possible de faire mieux en résolvant tous les sous-problèmes correspondant au niveau de flexibilité le plus serré pour favoriser la génération d'horaires de qualité. Or on sait grâce aux travaux de Boubaker (2006) et à quelques essais numériques que les instances admettent des solutions dont les horaires sont issus des sous-problèmes des deux niveaux les plus serrés. On cherche donc à résoudre à chaque itérations tous les sous-problèmes associés à ces niveaux de flexibilité. Pour assurer la réalisabilité, on ne se limitera pas qu'à ces derniers et on choisit de résoudre le tiers de ceux restant. Cependant le critère d'arrêt détermine le nombre de succès à obtenir. On sait que lors des premières itérations, la plupart des sous-problèmes admettent des colonnes de coût réduit négatif, mais le nombre de ces sous-problèmes tend à décroître au fil des itérations. La politique choisie implique donc que le nombre de sous-problèmes résolus tend à croître au fil des itérations.

Par ailleurs, lorque l'on impose l'utilisation d'un horaire et que ce dernier n'est pas considéré comme anonyme, il n'est plus nécessaire de continuer à résoudre les sous-problèmes associés à l'employé qui vient de voir son horaire imposé, puisqu'aucune nouvelle colonne en provenance de ces réseaux ne sera jamais plus utilisée.

4.2.5 Modèle avec la fonction objectif tronquée

Ce modèle est, d'un certain point de vue, le plus simple des deux. En effet, le graphe de ce modèle est dérivé du sous-problème le plus flexible du modèle avec la fonction escalier duquel on a enlevé les coûts associés aux pénalités P^z. Les réseaux ne sont, par conséquent, plus dupliqués Z fois. Le calcul des coûts est donc pris en charge par le solveur qui a alors besoin de certains paramètres pour ce faire. Effectivement, il faut transmettre à ce dernier la valeur des moyennes du nombre de crédits et du nombre de jours de congé. De plus, puisque les pénalités ne sont plus incluses comme des constantes sur des arcs, on doit modifier l'objectif pour considérer les nouvelles pénalités. Ainsi, pour un employé $e \in \bar{E}$ et une étiquette k existant au noeud puit

t_e, le coût c^k du chemin associé à l'étiquette k sera :

$$c^k = \sum_{(i,j)\in A_e^z} c_{ij} x_{ij}^k + \gamma_{t_e}^k. \tag{4.24}$$

Le terme nouvellement introduit γ_i^k est la valeur de la pénalité du chemin correspondant à l'étiquette k au noeud i, tandis que les variables x_{ij}^k nous indiquent si l'arc (i,j) est utilisé dans le chemin correspondant à l'étiquette k. En prenant la valeur de ce terme au noeud puits t_e, on obtient la pénalité totale du chemin. La pénalité γ_i^k est calculée selon la fonction :

$$\gamma_{t_e}^k = \alpha_{cr} \left(\max\left\{0, T_{t_e}^{NbCrMax} - \mu_{cr}\right\}\right)^2 + \alpha_{cg} \left(\max\left\{0, T_{t_e}^{NbCgMax} - \mu_{cg}\right\}\right)^2. \tag{4.25}$$

Dans le même ordre d'idées, lors de la propagation d'une étiquette au noeud suivant, le terme plus haut doit être recalculé à nouveau en tenant compte de la consommation des ressources NbCrMax et NbCgMax sur l'arc utilisé pour propager.

4.3 Agrégation dynamique de contraintes

Cette section porte sur l'agrégation dynamique de contraintes d'un problème de partitionnement d'ensemble dans un contexte de génération de colonnes. On présentera d'abord un léger survol de la méthode en portant particulièrement attention au contexte de son émergence pour ensuite présenter la méthode et quelques notions théoriques qui s'y rattachent. On pourra finalement présenter l'application de l'agrégation dynamique de contraintes étant donné l'application de la génération de colonnes au même problème.

4.3.1 Introduction

Cette méthode existe principalement parce que la génération de colonnes peut mettre beaucoup de temps à résoudre des problèmes de très grande taille. En effet,

une méthode de génération de colonnes n'arrivera pas à résoudre en temps raisonnable un problème de quelques milliers de contraintes avec des colonnes comportant plus d'une dizaine d'éléments non-nuls. Les instances résolues dans ce mémoire comportent heureusement en moyenne moins de six éléments non-nuls par colonne, ce qui permet de repousser un peu la limite du nombre de contraintes, mais comme en témoignent les résultats du prochain chapitre, lorsque la taille de l'instance grandit, le temps de calcul devient rapidement trop long. En effet, le temps de résolution peut aller jusqu'à un peu plus d'une dizaine d'heures pour un horizon de planification d'un mois. Or cette situation est loin d'être idéale et une amélioration s'impose.

La principale raison qui explique une telle explosion du temps de calcul en fonction de la taille du problème est le phénomène de dégénérescence. L'algorithme du simplexe utilisé pour résoudre le PMR peut alors pivoter un très grand nombre de fois sans changer de point extrême. En effet, dans un problème de partitionnement d'ensemble de grande taille, c'est-à-dire comportant un grand nombre de contraintes, le nombre moyen de bases par point extrême est excessivement grand. C'est lorsque ce nombre est élevé que la dégénérescence est susceptible de se manifester parce qu'il est possible de représenter un même point d'un très grand nombre de manières. La méthode du simplexe peut alors faire plusieurs pivots et changer plusieurs fois de base, alors que toutes ces bases représentent le même point. Pour limiter la portée du phénomène, on peut diminuer le nombre de contraintes et ainsi diminuer le nombre de bases par point extrême et le risque d'effectuer des pivots dégénérés. L'agrégation dynamique de contraintes est donc une technique qui vise à accélérer la résolution d'une itération de génération de colonnes sans trop en augmenter le nombre et ce, en réduisant le nombre de contraintes grâce à l'agrégation de ces dernières. En effet, pivoter dans un problème où les contraintes sont agrégées est beaucoup plus rapide que dans un problème où elles ne le sont pas.

Par ailleurs, l'agrégation dynamique accélère aussi la résolution en nombres entiers du problème car elle nécessite moins de noeuds de branchement en séparation

et génération progressive (*branch-and-price*). En effet, lorsque les contraintes sont agrégées, le nombre de variables fractionnaires a tendance à être moins élevé, donnant lieu à un arbre de branchement beaucoup plus petit. Il existe habituellement une multitude de solutions semblables dans le PMR du noeud zéro et l'agrégation dynamique a tendance à en choisir une comportant peu de variables fractionnaires.

4.3.2 Description de la méthode

L'agrégation dynamique de contraintes de partitionnement d'ensemble telle qu'utilisée dans le cadre de ce mémoire est une méthode basée sur la génération de colonnes comme on en a décrit les principes de base lors des sections précédentes. Soit un problème maître (PM) comme celui décrit par (4.1)-(4.5). On cherchera à agréger plusieurs contraintes selon une partition Q de toutes les contraintes de manière à obtenir un PM agrégé (PMA). Cette partition nous indique quelles tâches ou contraintes seront effectuées par une même personne et par conséquent, peuvent être remplacées par une seule tâche ou contrainte. La partition Q est telle que les regroupements (clusters) sont mutuellement disjoints et que ces derniers couvrent l'ensemble des tâches. On peut résoudre le PMA de manière à obtenir les valeurs des variables duales agrégées desquelles on peut obtenir les valeurs des variables duales désagrégées. On expliquera un peu plus loin comment y parvenir. Ces valeurs peuvent être utilisées, comme en génération de colonnes, pour résoudre le ou les sous-problèmes. On obtient ainsi de nouvelles colonnes qui ne sont pas nécessairement compatibles avec la partition Q, c'est-à-dire qu'une colonne nouvellement générée peut couvrir seulement une partie des tâches d'un regroupement, ce qui nous empêche de l'inclure au PMA. Une colonne est donc compatible par rapport à une partition Q si le chemin correspondant à la colonne (voir §4.2.1) passe par toutes les tâches du regroupement ou par aucune et ce, pour tous les regroupements. Donc, comme toutes les colonnes ne sont pas nécessairement compatibles, il faut choisir s'il faut désagréger les regroupements de tâches du PMA qui rendraient compatibles les colonnes incompatibles

que l'on veut inclure ou continuer la résolution en n'incluant que les colonnes compatibles sans désagréger et conserver les colonnes incompatibles dans une pile pour un usage ultérieur. Le critère utilisé pour décider s'il faut désagréger est le suivant :

$$r = \frac{\min_{p \in P''_Q} \bar{c}_p}{\min_{p \in \bar{P}'_Q} \bar{c}_p} < \lambda \text{ et } \min_{p \in \bar{P}'_Q} \bar{c}_p < 0,$$

où P''_Q est l'ensemble des colonnes compatibles générées, $\bar{P}'_Q$ est l'ensemble des colonnes incompatibles générées, $\bar{c}_p$ est le coût réduit d'une colonne p et λ est un paramètre. Les deux relations doivent donc être vérifiées pour qu'il y ait désagrégation. Une colonne incompatible ayant un fort potentiel de faire diminuer l'objectif par rapport aux colonnes compatibles déclenchera donc une désagrégation de la partition. Ainsi, le paramètre λ peut prendre n'importe quelle valeur positive, mais les meilleurs résultats sont obtenus lorsqu'il est compris entre zéro et un, c'est-à-dire que l'on n'exige pas que $\min_{p \in \bar{P}'_Q} \bar{c}_p$ soit beaucoup plus petit que $\min_{p \in P''_Q} \bar{c}_p$.

Il existe plusieurs variantes de l'agrégation dynamique de contraintes, toutes issues de l'amélioration de la méthode initiale. La méthode initiale, DCA (Dynamic Constraint Agregation) (El Hallaoui *et al.*, 2005), est celle qui a été décrite jusqu'à présent et qui a servi de fondement aux suivantes. Toutefois, il faut introduire la notion de degré d'incompatibilité d'une colonne par rapport à une partition Q pour pouvoir introduire MPDCA (Multi-Phase DCA) (El Hallaoui *et al.*, 2008b) qui est la première amélioration de DCA. Or, le degré d'incompatibilité est une mesure de la différence des regroupements des tâches d'une colonne par rapport aux regroupements de la partition Q. Plus cette mesure est élevée, plus de regroupements de Q devront être scindés pour que la colonne soit compatible avec la nouvelle partition. Une définition plus formelle et rigoureuse sera fournie plus loin dans la section sur les sous-problèmes utilisés. MPDCA propose donc de contrôler les colonnes à inclure au PMA en fonction de leur degré d'incompatibilité p. Plus précisément, plusieurs phases sont effectuées (figure 4.6) et une phase k permet d'inclure toute colonne ayant un degré d'incompatibilité $p \leq k$. On augmente k itérativement jusqu'à $|W| - 1$ pour

obtenir une solution optimale, mais on peut décider de se limiter aux quelques premières phases, ce qui est généralement une technique d'accélération efficace rendant la méthode cependant heuristique. En n'acceptant que les colonnes peu incompatibles au début, on s'assure de ne pas trop ralentir la résolution du PMA. En effet, en introduisant une colonne dont le degré d'incompatibilité est p, la partition Q comportera au maximum p regroupements ou contraintes additionnelles. Cette approche est cependant sensible à la qualité de la partition initiale, ce qui signifie que si on dispose d'une manière efficace pour trouver une bonne partition initiale, la résolution du problème sera rapide. Pour des fins de concision, la figure 4.6 montre le fonctionnement de l'algorithme MPDCA pour résoudre la relaxation linéaire d'un problème, mais on peut aussi l'intégrer à un schéma de séparation et évaluation progressive comme on l'a vu en génération de colonnes.

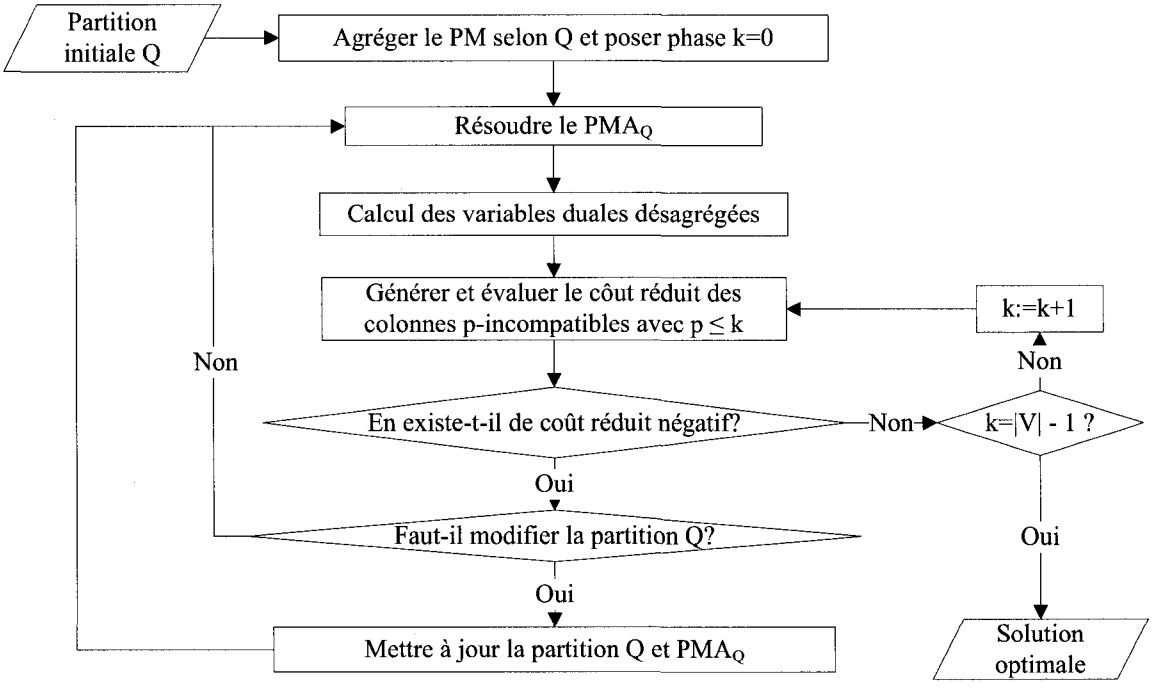

Figure 4.6 – Algorithme MPDCA

Dans un autre ordre d'idées, les deux méthodes présentées convenaient mieux à des problèmes où le temps de résolution du problème maître était important et

ne garantissait pas vraiment d'économies de temps au niveau des sous-problèmes. BDCA (Bi-Dynamic Constraint Agragation) (El Hallaoui *et al.*, 2008a) répond à ce besoin en agrégeant les tâches au niveau du sous-problème aussi d'où l'appellation de Bi-DCA parce qu'il y a agrégation aux deux niveaux. C'est un paramètre β qui régit l'agrégation des tâches au niveau du sous-problème de la manière suivante : on construit un sous-ensemble $\mathcal{Q} \subseteq Q$ des $\beta |Q|$ regroupements dont les variables duales associées aux contraintes (4.2) correspondantes sont les plus négatives et on enlève tout arc du réseau pouvant mener à un chemin incompatible par rapport à un regroupement de $\mathcal{Q}$. Les chemins générés par le réseau ainsi réduit peuvent être fortement incompatibles par rapport à $Q \setminus \mathcal{Q}$, mais si la taille de $\mathcal{Q}$ est assez grande, la partition n'a pas à être désagrégée beaucoup pour pouvoir y inclure les nouvelles colonnes. Pour que l'approche demeure une méthode exacte, si lors de la résolution d'un ou plusieurs problèmes de plus court chemin à une itération donnée on ne trouve pas de chemin(s) de coût (coût réduit) négatif dans le ou les réseaux agrégés, le ou les réseaux seront complètement désagrégées et résolus à nouveau. Un des avantages de BDCA est que seules les valeurs des variables duales associées à $Q \setminus \mathcal{Q}$ ont besoin d'être calculées en plus d'avoir à résoudre un réseau plus petit.

Désagrégation des variables duales

On a présenté plusieurs fois jusqu'à maintenant la problématique de la désagrégation des variables duales et c'est assurément la technique proposée pour résoudre le problème qui assure le succès de DCA (El Hallaoui *et al.*, 2005). Pour obtenir les valeurs des variables duales désagrégées α à partir des variables duales agrégées $\hat{\alpha}$, on doit résoudre un système d'équations et d'inéquations de très grande taille. En effet, si on définit W_l comme un regroupement de tâches, L comme l'ensemble des regroupements tel que $Q = \{W_l : l \in L\}$, $\bar{P}'_Q$ comme l'ensemble des colonnes incompatibles générées et $\bar{P}^*_Q$ comme un sous-ensemble de $\bar{P}'_Q$ pour lequel on veut que le bloc de contraintes (4.27) soit satisfait, on peut définir le problème de satisfaction de

contraintes comme :

$$\sum_{w \in W_l} \alpha_w = \hat{\alpha}_l, \quad \forall l \in L \tag{4.26}$$

$$\sum_{w \in W_l} a_{wp} \alpha_w \leq c_p, \quad \forall p \in \bar{P}_Q^* \subseteq \bar{P}_Q'. \tag{4.27}$$

Les contraintes (4.26) nous assurent que la somme des variables duales désagrégées d'un regroupement équivaut aux variables duales agrégées pour les contraintes que l'on cherche à désagréger et que les coûts réduits du PMRA courant soient positifs ou nuls. Les contraintes (4.27) nous assurent que les coûts réduits obtenus pour les colonnes ou variables incompatibles choisies soient non-négatifs. Résoudre ce problème directement peut s'avérer très coûteux en temps de calcul étant donné sa grande taille. Il est toutefois possible d'appliquer le changement de variable :

$$\pi_l^h = \sum_{j=1}^{h} \alpha_l^j, \quad \forall h \in \{1, \ldots, |W_l|\}, l \in L, \tag{4.28}$$

qui permet de formuler (4.26)-(4.27) comme un problème de plus court chemin si on se limite à certaines colonnes pour peupler $\bar{P}_Q^*$. Le lecteur est invité à consulter El Hallaoui *et al.* (2005) pour connaître le type de colonnes à inclure à $\bar{P}_Q^*$ et la manière de construire le graphe duquel on peut trouver les plus courts chemins.

4.3.3 Application de MPDCA au CBMPE

Mentionnons d'emblée que l'on utilise toujours un cadre de SGP et que l'arbre d'énumération, quoique plus petit en raison du nombre moindre de variables fractionnaires, est toujours de grande taille. On a donc encore recours à des règles de branchement heuristiques soit les mêmes que celles utilisées en génération de colonnes. On a cependant ajouté une autre règle accélérant la résolution. Puisque l'on sait que la valeur de la relaxation linéaire à la dernière itération du noeud zéro de l'arbre d'énumération est bien meilleure qu'en génération de colonnes, on cherchera

à en profiter. La borne inférieure du noeud père sera donc utilisée comme borne inférieure approximative pour arrêter la résolution d'un noeud de branchement, même s'il est possible qu'il existe encore des colonnes de coût réduit négatif à générer à ce noeud.

Par ailleurs, la principale différence dans la modélisation d'un problème pour l'application de MPDCA par rapport à la génération de colonnes est dans la formulation du sous-problème. En effet, il faut modifier légèrement cette dernière en y ajoutant quelques noeuds et arcs pour inclure des chemins correspondant à la partition initiale déduite de la solution de l'algorithme tabou. Tout le reste demeure identique principalement parce que la formulation pour la génération de colonnes était prévue en conséquence d'une utilisation ultérieure des mêmes réseaux pour l'application de l'agrégation dynamique. L'idée derrière l'ajout des noeuds et arcs est qu'il doit être possible pour l'algorithme de programmation dynamique des sous-problèmes de trouver un chemin correspondant à l'horaire d'un employé d'après la solution initiale sans utiliser les arcs d'attente. Il existera donc un chemin partant de la source visitant consécutivement tous les arcs correspondant aux tâches d'un horaire de manière à ce qu'aucun arc du réseau courant autres que ceux correspondant à la réalisation de la tâche ne soit utilisé. Il faut donc ajouter des arcs pour que de tels chemins existent. Cependant, pour s'assurer que la consommation des ressources soit juste, on a besoin de deux arcs entre deux tâches d'un même horaire, d'où l'introduction de noeuds supplémentaires. En effet, il est possible qu'il y ait des journées de congé entre deux tâches, ce qui implique qu'il faille remettre à zéro la ressource comptant le nombre maximal de jours travaillés consécutifs. Cependant, il faut aussi ajouter une unité à cette ressource pour comptabiliser correctement le premier jour travaillé de la rotation, mais il est impossible d'à la fois remettre à zéro et consommer une unité de la même ressource sur un seul arc. Les noeuds introduits sont identiques à n'importe quel noeud sur l'axe du temps, hormis les minuits. Quant à eux, les arcs (voir les arcs en gras de la figure 4.7) valent la peine que l'on s'y attarde ; on en retrouve quatre types :

- Arc source à début première tâche : Cet arc n'existe que pour les employés anonymes et les employés identifiables n'ayant pas de tâches préassignées en début d'horizon. En effet, un arc existe déjà entre la source et la première tâche dans le réseau pour la génération de colonnes pour les employés ayant une tâche préassignée en début d'horizon. On consommera le nombre de jours de congé équivalent au nombre d'arcs de minuit à minuit qu'il aurait fallu emprunter pour se rendre au noeud de début de tâche à partir de la source. On ajoute aussi un jour travaillé consécutif et il existe un arc de ce type par employé.
- Arc fin de tâche à noeud intermédiaire : C'est sur cet arc que seront consommés les jours de congé correspondant à ce qui aurait été consommé en passant par les arcs auparavant présents dans le réseau. Plus précisément, sera consommé le nombre de jours de congé qu'aurait un chemin entre la fin de la tâche et le début de la suivante. Si le nombre de jours de congé est strictement positif, la ressource comptabilisant les jours travaillés consécutifs sera remise à zéro.
- Arc noeud intermédiaire à début de tâche : Cet arc a pour unique but de comptabiliser un jour travaillé consécutif si la fin de la première tâche est au moins un jour plus tôt que le début de la tâche suivante. On ne pouvait pas à la fois ajouter une unité consommée à cette ressource et la remettre à zéro et c'est ce qui explique qu'on utilise deux arcs plutôt qu'un seul.
- Arc fin dernière tâche à puits : La consommation des jours de congé sera analogue à ce qui est fait pour les autres types d'arcs. Il existe un arc de ce type par employé.

Le réseau équivalent pour un employé identifiable ne sera pas présenté puisqu'il est relativement facile de se l'imaginer à partir du graphe présenté à la section précédente. Par ailleurs, c'est à l'aide du graphe de la figure 4.7 qu'il est le plus facile d'expliquer clairement la notion de degré d'incompatibilité. On peut voir cette mesure comme une ressource non-décroissante sur le réseau. Tout arc permettant, à partir d'un chemin initial, d'aller vers l'axe du temps consommera une unité de cette ressource. De manière analogue, tout arc partant d'un noeud sur l'axe du temps autre que la source et se rendant à un noeud sur le chemin initial comptabilisera une unité de cette

ressource. Dans le graphe, les arcs consommant la ressource degré d'incompatibilité sont marqués d'un I°. Par exemple, pour un chemin initial, il en coûtera quatre unités de cette ressource pour couvrir une rotation appartenant à un autre chemin initial s'il y a des tâches avant et après la rotation dans les deux chemins. Il est facile de s'imaginer qu'un tel chemin est non désirable puisqu'il modifie considérablement la partition Q.

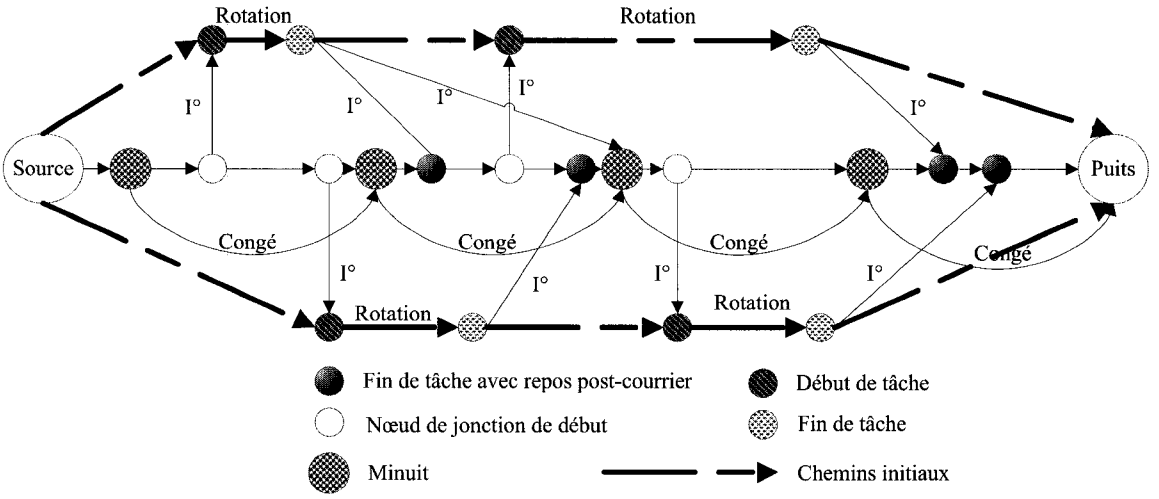

Figure 4.7 – Exemple de réseau utilisé pour l'agrégation dynamique

CHAPITRE 5
EXPÉRIMENTATIONS NUMÉRIQUES

Ce chapitre est consacré aux résultats des expérimentations sur les méthodes présentées. Il convient tout d'abord de présenter les instances utilisées pour les tests et la manière de les obtenir. Par la suite seront présentés les résultats des approches utilisées en débutant d'abord par la méthode de recherche taboue en tant que méthode de construction d'une solution initiale. On présentera ensuite les résultats obtenus par génération de colonnes et ceux obtenus avec la méthode d'agrégation dynamique de contraintes. Les essais ont tous été réalisés à l'aide de *Grid Engine* du GERAD (Groupe d'Études et de Recherche en Analyse des Décisions) sur les machines *Gauss* (4x Dual Core AMD Opteron 275 - 2193,82 MHz, 8Gb RAM , linux 64 bit).

5.1 Instances traitées

Les données utilisées pour tester les méthodes de résolution proviennent d'un client de la compagnie Kronos Inc. et un seul jeu de données de 2924 rotations sur un horizon d'un mois a été utilisé. Pour effectuer des tests sur des instances de taille différente, des sous-ensembles de 291, 506, 700, 974, 1187, 1507 et 2165 rotations prises au hasard ont été construits. Toutes ces données ont été reprises des travaux de Boubaker (2006) et sont identiques. Quant à elles, les tâches préassignées et les périodes de vacances, qui ne faisaient pas partie des données de Boubaker (2006), ont été générées aléatoirement selon certains paramètres que l'on s'efforcera de décrire.

Tout d'abord, à partir d'un jeu de données comportant seulement des rotations, il est relativement simple d'obtenir une bonne approximation du nombre de tâches préassignées qu'une instance réelle contiendrait. Il suffit de faire une moyenne du

nombre de rotations en cours sur plusieurs minuits au milieu de l'horizon de planification. Ainsi si la fin de l'horizon de planification était devancée, la fin des tâches coupées deviendrait des tâches préassignées pour l'horizon suivant. Par ailleurs, les tâches préassignées débutent toutes, par définition, au temps zéro et se termine en moyenne $\mu_d/2$ jours plus tard selon une distribution uniforme où μ_d est la durée moyenne d'une rotation.

Par ailleurs, la génération de périodes de vacances dépend de plusieurs hypothèses. On suppose que les vacances peuvent durer 3, 7 ou 14 jours et que ces durées sont représentées équitablement. D'une part, les périodes de vacances ou les congés débutent et se terminent à minuit et peuvent avoir lieu à n'importe quel moment dans l'horizon de planification. D'autre part, on a estimé, d'une manière arbitraire, que le quart des employés prenant des vacances aurait aussi une tâche préassignée dans leur horaire. Du côté du nombre de périodes de vacances à générer, on suppose d'abord que les employés ont droit à trois semaines de vacances par an. Si on note $|E|$ le nombre d'employés, le calcul du nombre de jours de vacances à générer pour un mois est : $\frac{|E|*3*7}{12}$. En supposant que l'on doit générer autant de blocs de 3, 7 ou 14 jours, le nombre de blocs d'une seule des durées est : $\left\lfloor \frac{|E|*3*7}{(3+7+14)*12} \right\rfloor$.

Finalement, puisque l'objectif consiste à minimiser la somme pondérée de l'écart-type des crédits et des jours de congé, les employés ayant une période de vacances de deux semaines sont désavantagés par rapport aux employés n'en ayant pas puisqu'ils n'ont que la moitié de la période pour accumuler les crédits et les jours de congé. Pour palier ce problème, il suffit de créditer des jours de congé et des crédits pour les périodes de vacances. On estime à 4500 la moyenne des crédits à attribuer à un employé pour un horizon de planification d'un mois, ce qui correspond approximativement à 150 crédits par jour. Donc 150 crédits seront crédités pour chaque jour de vacances. De manière semblable, les employés ont droit à au moins 10 jours de congé par mois. On crédite, par conséquent, un tiers de jour de congé par jour de vacances et on arrondit à l'entier supérieur pour que le nombre total de jours de

congé crédités soit entier et non nul. Par ailleurs, pour les tâches préassignées, on ne crédite pas de jours de congé, mais seulement des crédits. Étant donné la durée déjà établie de ces tâches, on accorde un crédit pour 6,65 minutes de ces dernières. Ce nombre correspond au rapport moyen entre le nombre de crédits et la durée d'une rotation pour l'instance complète.

D'autre part, puisque l'on ajoute des tâches au problème, il faut aussi ajuster le nombre d'employés pour ne pas affecter la réalisabilité de ce dernier. On recalcule donc le nombre d'employés disponibles en faisant la somme des crédits de toute tâche, incluant les périodes de vacances, et on divise ce nombre par 4500 qui est le nombre moyen de crédits par employé souhaité. On arrondit le résultat à l'entier supérieur et c'est pourquoi on obtient un nombre de crédits toujours légèrement inférieur à 4500.

Le tableau 5.1 résume les caractéristiques des instances telles que le nombre de rotations n, le nombre de tâches préassignées $Nb_{carry-in}$, le nombre de périodes de vacances $Nb_{vacances}$, le nombre d'employés Nb_{emp}, le nombre d'employés étant considérés comme anonymes $Nb_{anonymes}$, le nombre de contraintes dans le problème maître en génération de colonnes Nb_{CtrPM}, le nombre moyen de crédits par employé μ_{Cr} et, finalement, une approximation du nombre moyen de jours de congé par employé faite à l'aide de l'algorithme de recherche taboue pour solution initiale. C'est cette dernière valeur, arrondie à l'entier le plus près, qui sert à construire les modèles en génération de colonnes et en agrégation dynamique de contraintes. On identifiera une instance par son nombre de rotations n dans les tableaux subséquents.

5.2 Résultats de l'algorithme tabou

Pour ces essais, on a permis à l'algorithme de s'exécuter pendant des temps arbitraires comparables aux temps de Boubaker (2006). Le tableau 5.2 indique pour chaque instance : le nombre d'itérations nécessaires à l'obtention d'une solution réalisable $NbIter_{ini}$, le temps en seconde pour y parvenir T_{ini}, le nombre d'itérations

Tableau 5.1 – Caractéristiques des instances

n	$Nb_{carry-in}$	$Nb_{vacances}$	Nb_{emp}	$Nb_{anonymes}$	Nb_{CtrPM}	μ_{Cr}	μ_{Cg}
291	17	12	61	35	318	4466,11	13,8033
506	24	24	102	60	549	4489,93	14,1765
700	36	32	147	87	761	4479,84	14,1837
974	48	44	206	125	1056	4488,83	14,1117
1187	63	52	250	148	1290	4495,14	14,1280
1507	76	64	316	192	1632	4497,81	14,0570
2165	112	92	456	275	2347	4492,03	14,0636
2924	153	124	617	371	3171	4492,96	14,0827

Tableau 5.2 – Résultats de la recherche taboue

n	$NbIter_{ini}$	T_{ini}	$NbIter_{sol}$	$NbIter_{tot}$	T_{tot}	$IterParSec$
291	487	0,02	71558	161430	26	6208,85
506	640	0,04	74364	90589	35	2588,26
700	858	0,07	24714	42251	42	1005,98
974	1281	0,16	7986	23173	62	373,76
1187	1550	0,23	12819	21867	79	276,80
1507	1976	0,35	15592	21101	90	234,46
2165	2795	0,83	34973	34991	164	213,36
2924	3785	1,45	89923	93051	226	411,73

requis pour trouver la meilleure solution $NbIter_{sol}$, le nombre total d'itérations effectuées $NbIter_{tot}$, le temps total d'exécution de l'algorithme T_{tot} et, finalement, le nombre moyen d'itérations effectuées par seconde $IterParSec$. On remarque que le nombre d'itérations nécessaires à l'obtention d'une solution initiale est sensiblement le même que le nombre de rotations, ce qui nous indique que l'algorithme cible bien les violations de contraintes et n'effectue pas beaucoup de mouvements inutiles. Tout ceci permet d'obtenir une première solution réalisable très rapidement. Cependant, de manière générale, le nombre moyen d'itérations par seconde tend à décroître selon la taille de l'instance, indiquant que l'algorithme est sensible à cette taille. Ce comportement est dû au fait que lors de la phase d'uniformisation des horaires, le nombre de minimums locaux est élevé, rendant souvent nécessaire l'exploration complète des voisinages propres à la phase d'uniformisation dont la taille croît exponentiellement par rapport à celle de l'instance. La stratégie de la première amélioration est impuis-

sante face à un nombre élevé de minimums locaux, à l'instar de Boubaker (2006) qui se limitait à un sous-ensemble de taille constante de son voisinage, rendant l'effort de calcul moins dépendant de la taille de l'instance. On remarque une légère anomalie quant au nombre d'itérations par seconde obtenu pour la dernière instance. Il serait possible que ce résultat soit dû au fait que l'algorithme est beaucoup plus rapide lorsque la solution est non réalisable en raison de la taille restreinte des voisinages lors de cette phase.

Le tableau 5.3 présente les caractéristiques des solutions obtenues avec l'algorithme de recherche taboue pour chaque instance : l'écart-type des crédits pour la première solution réalisable trouvée σ_{cr}^{solini}, l'écart-type des jours de congé pour la première solution réalisable σ_{Cg}^{solini}, le nombre de crédits de l'horaire qui en contient le moins dans la meilleure solution Min_{Cr}, le nombre de crédits de l'horaire qui en contient le plus dans la meilleure solution Max_{Cr}, l'écart-type des crédits dans la meilleure solution σ_{Cr}, le nombre de jours de congé dans l'horaire qui en contient le moins pour la meilleure solution Min_{Cg}, le nombre de jours de congé de l'horaire qui en contient le plus dans la meilleure solution Max_{Cg} et, finalement, l'écart-type des jours de congé de la meilleure solution σ_{Cg}

Tableau 5.3 – Qualité de solution pour la recherche taboue

n	σ_{Cr}^{solini}	σ_{Cg}^{solini}	Min_{Cr}	Max_{Cr}	σ_{Cr}	Min_{Cg}	Max_{Cg}	σ_{Cg}
291	353,070	1,97947	4170	4847	150,966	12	16	1,05302
506	339,900	1,87173	4100	5078	164,799	12	17	1,07020
700	349,604	1,96922	4128	5021	160,470	11	17	1,14298
974	339,741	1,99672	4106	5036	187,449	11	18	1,17917
1187	371,415	1,94165	4133	5026	180,751	10	17	1,18980
1507	362,674	2,04069	4156	5063	172,702	10	17	1,02032
2165	341,269	2,07200	4058	5087	189,694	10	17	1,17462
2924	340,233	2,02967	4033	5097	206,432	10	17	1,16596

D'une part, peu de statistiques sont fournies sur les premières solutions réalisables étant donné la piètre qualité de ces dernières. En effet, avec des écarts-types approximativement cinq fois plus élevés que les solutions de la génération de colonnes pour

les crédits et presque trois fois dans le cas des jours de congé, il est évident que ces solutions ne sont pas directement utilisables. D'autre part, la qualité des meilleures solutions trouvées est considérablement meilleure, voire même excellente compte tenu du temps consacré à la résolution. Ces solutions pourraient être une alternative intéressantes aux solutions obtenues par génération de colonnes dans le cas où le temps disponible à la résolution est très limité. Cependant, la qualité tend à diminuer en fonction de l'augmentation de la taille de l'instance. Toutefois, d'après le tableau 5.2, on remarque que les meilleures solutions des plus grandes instances ont été obenues près des dernières itérations, ce qui nous laisse espérer qu'en augmentant le temps résolution, on obtiendrait de meilleures solutions. Les résultats obtenus ici sont significativement meilleurs que ceux obtenus par Boubaker (2006) en termes de qualité de solution pour une durée comparable.

5.3 Résultats de la génération de colonnes

Les résultats qui suivent ont été obtenus à l'aide de la version 4.5 de GENCOL, une librairie de fonctions en langage C conçue à l'origine par des chercheurs du GERAD et permettant d'implanter la méthode de génération de colonnes décrite en §4.2. L'unité de mesure du temps est la seconde et les résultats des deux modèles vus au chapitre précédent seront présentés dans deux sections différentes.

5.3.1 Modèle avec fonction en escalier

Le tableau 5.4 présente des statistiques sur la résolution par GENCOL du modèle. Parmi ces statistiques, on retrouve pour chaque instance : le temps total nécessaire à la résolution T_{Total}, le temps passé à résoudre le problème maître T_{PM}, le temps passé à résoudre les sous-problèmes T_{SP}, le nombre de sous-problèmes Nb_{SP}, le nombre de noeuds de branchement dans l'arbre d'énumération en SGP Nb_{noeuds}, le nombre de

Tableau 5.4 – Statistiques du modèle avec fonction escalier

n	T_{Total}	T_{PM}	T_{SP}	Nb_{SP}	Nb_{noeuds}	Nb_{frac}	Nb_{Iter}
291	59,9	7,0	50,4	135	61	317	363
506	338,6	78,8	254,1	215	78	508	568
700	971,8	256,1	703,8	305	101	643	695
974	1981,4	677,2	1258,3	410	137	824	716
1187	4428,1	1428,4	2902,2	515	164	1094	863
1507	6919,5	3241,4	3529,3	625	208	1434	933
2165	19546,8	8829,4	9889,8	910	321	1980	1206
2924	39031,3	26675,6	12122,2	1235	289	2455	1149

variables fractionnaires à la dernière itération du noeud zéro Nb_{frac} et, finalement, le nombre d'itérations effectuées.

On remarque que les temps augmentent rapidement avec la taille de l'instance et que ce n'est que pour la dernière instance que le temps consacré au PM devient plus grande que celui consacré au SP. C'est donc pour cette instance que l'on croit que l'application de l'agrégation dynamique aura le plus grand impact. Étonnamment, moins de noeuds de branchement ont été nécessaires pour la plus grosse instance que celle où n=2165, alors que de manière générale, plus la taille est grande, plus le nombre de noeuds l'est. On peut expliquer ce résultat par le fait que les règles de branchement sont heuristiques et que pour n=2924, de bonnes décisions ont été prises. On remarque aussi le nombre élevé de variable fractionnaires, ce qui permet d'affirmer que la quantité de travail à effectuer en SGP est importante.

Le tableau 5.5 présente des statistiques sur les solutions obtenues. Ces statistiques sont les mêmes que celles présentées dans les 6 dernières colonnes du tableau 5.3.

On voit dans le tableau 5.5 que les solutions que l'on obtient sont définitivement meilleures que celles obtenues par la méthode taboue (tableau 5.3). De plus, en observant les résultats des mesures de dispersion fournies que les horaires des solutions finales sont tous issus des sous-problèmes associés aux deux niveaux de flexibilité les plus serrés. Cependant, on observe que la qualité des solutions tend à diminuer avec la taille de l'instance, ce qui n'est pas un comportement souhaitable.

Tableau 5.5 – Qualité des solutions pour le modèle avec fonction escalier

n	Min_{Cr}	Max_{Cr}	σ_{Cr}	Min_{Cg}	Max_{Cg}	σ_{Cg}
291	4359	4589	74,467	13	16	0,826
506	4289	4704	80,158	13	17	0,846
700	4251	4710	92,791	13	16	0,927
974	4259	4701	81,968	12	16	0,846
1187	4259	4734	99,022	12	17	0,922
1507	4259	4734	95,596	12	16	0,891
2165	4256	4732	100,892	12	17	0,966
2924	4255	4735	111,183	12	17	1,050

Tableau 5.6 – Qualité des solutions du modèle avec fonction tronquée

n	Min_{Cr}	Max_{Cr}	σ_{Cr}	Min_{Cg}	Max_{Cg}	σ_{Cg}
291	3936	4671	147,317	11	16	1,086
506	3927	4726	152,362	11	18	1,125
700	3949	4731	156,685	11	17	1,186
974	3926	4818	188,148	10	18	1,291
1187	3911	4846	186,335	10	18	1,510
1507	3910	4804	183,914	10	18	1,469
2165	3916	4799	195,590	10	19	1,505
2924	3912	4905	175,145	10	18	1,330

5.3.2 Modèle avec fonction tronquée

Pour ce modèle, nous présentons d'abord le tableau 5.6 qui rapporte les résultats concernant la qualité des solutions.

Les solutions obtenues contiennent toutes le défaut évoqué lors du précédent chapitre à savoir que le nombre de crédits et le nombre de jours de congé de l'horaire en contenant le moins s'approchent beaucoup des bornes inférieures correspondantes puisque ce cas est fortement sous-pénalisé. Ce modèle donne donc des solutions de qualité moindre par rapport au modèle avec la fonction escalier pour des temps de résolution très semblables. Les statistiques détaillées sur les temps de résolution ne seront donc pas présentées. Cette approche est largement supplantée par l'algorithme de recherche taboue si on considère les temps de calcul.

Tableau 5.7 – Statistiques pour MPDCA

n	T_{Total}	T_{PM}	T_{SP}	Nb_{SP}	Nb_{noeuds}	Nb_{frac}	Nb_{Iter}
291	48,8	1,0	46,0	135	33	220	143
506	176,7	1,9	169,0	215	41	283	203
700	346,0	4,0	329,8	305	49	357	224
974	537,8	7,0	512,7	410	36	499	185
1187	1221,1	19,6	1164,5	515	71	610	315
1507	1974,5	26,0	1882,6	625	60	717	289
2165	5148,9	71,8	4937,5	910	47	953	257
2924	10941,7	161,3	10460,6	1235	66	1314	346

5.4 Résultats de l'agrégation dynamique

On présente dans le tableau 5.7 les statistiques de résolution des instances avec l'agrégation dynamique (MPDCA).

On voit rapidement que les temps de calcul sont moins importants qu'en génération de colonnes et que cette baisse est plus marquée au niveau du PM. De plus, un nombre beaucoup moins élevé de noeuds de branchement a été nécessaire principalement en raison du nombre moins élevé de variables fractionnaires. On voit aussi que le temps consacré à la résolution des sous-problèmes est considérable par rapport, à celui requis pour résoudre le PM alors que le nombre de SP est le même qu'en génération de colonnes. En effet, MPDCA permet de réduire le temps total de résolution par rapport à la génération de colonnes en moyenne par un facteur de trois alors que ce facteur est plutôt près de 87 pour le PM, mais de seulement de 1,8 pour le SP. De plus, avec MPDCA, le nombre de noeuds de branchement résolus est en moyenne 3,3 fois moins élevé alors que le nombre de variables fractionnaires l'est 1,8 fois moins. Cependant, alors que l'on s'attendait à une augmentation du nombre d'itérations, on constate plutôt une diminution. On explique ce comportement par le fait qu'un nombre considérablement moins important de noeuds de branchement sont résolus en raison du nombre moins élevé de variables fractionnaires.

Le tableau 5.8 présente les résultats permettant de quantifier la qualité des solu-

Tableau 5.8 – Qualité des solutions de MPDCA

n	Min_{Cr}	Max_{Cr}	σ_{Cr}	Min_{Cg}	Max_{Cg}	σ_{Cg}
291	4357	4591	65,750	13	16	0,878
506	4372	4610	68,422	13	16	0,722
700	4363	4602	71,389	13	16	0,771
974	4371	4727	72,339	13	16	0,729
1187	4378	4616	72,514	13	16	0,742
1507	4378	4628	66,890	13	16	0,738
2165	4374	4624	68,215	13	16	0,766
2924	4374	4614	71,404	13	16	0,784

tions obtenues avec MPDCA. Les colonnes sont identiques à celles du tableau 5.3, 5.5 et 5.6.

On voit dans le tableau 5.8 que les solutions obenues avec l'agrégation dynamqiue sont nettement meilleures que celles obtenues avec la génération de colonnes. Les écarts-types des crédits y sont considérablement réduits par rapport aux solutions obtenues en génération de colonnes, tandis que les écarts-types des jours de congé le sont aussi, mais de façon moins spectaculaire.

5.5 Conclusion

Les résultats présentés démontrent clairement que l'application de l'agrégation dynamique a un impact important en ce qui a trait à l'amélioration, à la fois des temps de calcul et de la qualité des solutions. Cependant, tandis que l'on réussit à couper les temps de calcul d'un facteur d'environ deux à trois, les travaux de Boubaker (2006) présentent plutôt une application où les temps sont coupés par un facteur de 60. Bien que, dans le cas présent, le temps consacré au PM est diminué d'un facteur moyen d'environ 87, le temps consacré au SP n'est diminué en moyenne que d'un facteur de 1,8.

CONCLUSION

Dans le cadre de ce mémoire, nous avons cherché à résoudre le problème de construction des horaires personnalisés de personnel navigant avec équité. Ce problème en est un de très grande taille et est long à résoudre avec les approches traditionnellement utilisées comme la séparation et génération progressive (SGP). Nous avons donc proposé de remplacer la génération de colonnes utilisée en SGP par l'agrégation dynamique de contraintes, une méthode à la fine pointe de la technologie. Cette méthode nécessitant une solution initiale réalisable pour le problème nous avons développé une métaheuristique de type recherche taboue pour en trouver une. Les résultats obtenus montrent que l'approche permet de résoudre les temps de calculs en moyenne par un facteur 3 tout en permettant de réduire les écarts-type des crédits en moyenne par un facteur de 1,32 et les écarts-type des jours de congés en moyenne par un facteur de 1,19

Ce mémoire a repris en partie les travaux de Boubaker (2006) sur la version anonyme du problème. Bien que les problèmes soient similaires, l'implantation des mêmes approches de résolution aux deux différents problèmes est considérablement différente. L'application avec succès de l'agrégation dynamique de contraintes à un problème décomposé selon Dantzig et Wolfe (1960) comportant un nombre très élevé de sous-problèmes constitue une première. La métaheuristique et les mécanismes de contrôle en génération de colonnes utilisés ont révélé des améliorations considérables par rapport aux travaux de Boubaker (2006). Les résultats obtenus à l'aide de l'agrégation dynamique de contraintes sont cependant moins spectaculaires. Ces derniers étaient, par contre, prévisibles étant donné que l'agrégation dynamique est une méthode réduisant essentiellement les temps de résolution du problème maître et que le problème à résoudre dans le cas présent nécessite énormément d'efforts dans la résolution des sous-problèmes.

En observant les résultats obtenus à l'aide de MPDCA, on constate que le temps consacré à résoudre les sous-problèmes devient disproportionné par rapport au temps consacré à résoudre le problème maître. C'est donc au niveau du sous-problème que se trouve le potentiel d'amélioration de l'application. L'application de BDCA est donc une avenue intéressante et la suite logique à la poursuite des travaux. Cette dernière méthode nous laisse entrevoir d'importants gains pour les temps de calculs des sous-problèmes.

BIBLIOGRAPHIE

AARTS, E., KORST, J. et VAN LAARHOVEN, P. (1997). Simulated annealing. *Dans* AARTS, E. et VAN LAARHOVEN, P., éditeurs : *Local search in combinatorial optimization*, pages 91–120. John Wiley & Sons, Inc., New York, NY.

ALABAS-USLU, C. (2007). A self-tuning heuristic for a multi-objective vehicle routing problem. *Journal of the Operational Research Society*, 59, 988–996.

ANBIL, R., GELMAN, E., PATTY, B. et TANGA, R. (1991). Recent advances in crew-pairing optimization at American Airlines. *Interfaces*, 21, 62–74.

BAKER, E., BODIN, L. et FISHER, M. (1985). The development of a heuristic set covering based system for aircrew scheduling. *Transportation Policy Decision Making*, 3, 95–110.

BARNHART, C., COHN, A., JOHNSON, E., KLABJAN, D., NEMHAUSER, G. et VANCE, P. (2003). Airline crew scheduling. *Dans* HALL, R., éditeur : *Handbook of Transportation Science*, pages 517–560. Kluwer Scientific Publishers, Norwell, MA.

BARNHART, C., JOHNSON, E., NEMHAUSER, G., SAVELBERGH, M. et VANCE, P. (1998). Branch-and-price : Column generation for solving huge integer programs. *Operations Research*, 43, 491–499.

BIRÓ, M., HUJTER, M. et TUZA, Z. (1992). Precoloring extension. i : Interval graphs. *Discrete Mathematics*, 100(1-3), 267–279.

BOUBAKER, K. (2006). Recherche taboue et agrégation dynamique de contraintes pour la construction d'horaires mensuels d'équipages aériens dans un contexte d'équité. Mémoire de Maîtrise, École Polytechnique de Montréal.

BOUBAKER, K., DESAULNIERS, G. et ELHALLAOUI, I. (2008). Bidline scheduling with equity by heuristic dynamic constraint aggregation. Rapport technique G-2008-43, Les Cahiers du GERAD, Montréal.

BRÉLAZ, D. (1979). New methods to color the vertices of a graph. *Communications of the ACM*, 22(4), 251–256.

BYRNE, J. (1988). A preferential bidding system for technical aircrew. *Dans AGIFORS Symposium Proceedings*, volume 28, pages 87–99.

CAMPBELL, K., DUFREE, R. et HINES, G. (1997). Fedex generates bid lines using simulated annealing. *Interfaces*, 27(2), 1–16.

CHRISTOU, I., ZAKARIAN, A., LIU, J.-M. et CARTER, H. (1999). A two-phase genetic algorithm for large-scale bidline-generation problems at Delta Airlines. *Interfaces*, 29(5), 51–65.

COOK, S. (1971). The complexity of theorem-proving procedures. *Dans STOC '71 : Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, New York, NY.

DANTZIG, G. et WOLFE, P. (1960). The decomposition algorithm for linear programming. *Operations Research*, 8, 101–111.

DAVIS, L. (1991). *Handbook of genetic algorithms*. Van Nostrand Reinhold, New York, NY.

DESAULNIERS, G., DESROSIERS, J., DUMAS, Y., MARC, S., RIOUX, B., SOLOMON, M. et SOUMIS, F. (1997). Crew pairing at Air France. *European Journal of Operational Research*, 97(2), 245–259.

DESAULNIERS, G., DESROSIERS, J., GAMACHE, M. et SOUMIS, F. (1998). Crew scheduling in air transportation. *Dans* CRAINIC, T. et LAPORTE, G., éditeurs : *Fleet Management and Logistics*, pages 169–185. Kluwer Academic Publishers, Boston, MA.

EL HALLAOUI, I., DESAULNIERS, G., METRANE, A. et SOUMIS, F. (2008a). Bidynamic constraint aggregation and subproblem reduction. *Computers & Operations Research*, 35, 1713–1724.

EL HALLAOUI, I., METRANE, A., SOUMIS, F. et DESAULNIERS, G. (2008b). Multi-phase dynamic constraint aggregation for set partitioning type problems. *Mathematical Programming A. DOI : 10.1007/s10107-008-0254-5.*

EL HALLAOUI, I., VILLENEUVE, D., DESAULNIERS, G. et SOUMIS, F. (2005). Dynamic aggregation of set-partitioning constraints in column generation. *Operations Research*, 53(4), 632–645.

FAHLE, T., JUNKER, U., KARISCH, S., KOHL, N., SELLMANN, M. et VAABEN, B. (2002). Constraint programming based column generation for crew assignment. *Journal of Heuristics*, 8(23), 59–81.

FEDERICI, F. et PASCHINA, D. (1990). Automated rostering model. *Dans AGIFORS Symposium Proceedings*, volume 12, pages 19–47.

GAMACHE, M., HERTZ, A. et OUELLET, J. (2007). A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding. *Computers and Operations Research*, 34(8), 2384–2395.

GAMACHE, M. et SOUMIS, F. (1998). A method for optimally solving the rostering problem. *Dans* YU, G., éditeur : *Operations Research in the Airline Industry*, pages 124–157. Kluwer, Norwell, MA.

GAMACHE, M., SOUMIS, F., DESROSIERS, J., VILLENEUVE, D. et GÉLINAS, . (1998). The preferential bidding system at Air Canada. *Transportation Science*, 32, 246–255.

GERSHKOFF, I. (1989). Optimizing flight crew schedules. *Interfaces*, 19(4), 29–43.

GIAFERRI, C., HAMON, J. et LENGLINE, J. (1982). Automatic monthly assignment of medium-haul cabin crew. *Dans AGIFORS Symposium Proceedings*, volume 22, pages 69–95.

GLANERT, W. (1984). A timetable approach to the assignment of pilots to rotations. *Dans AGIFORS Symposium Proceedings*, volume 24, pages 369–391.

GLOVER, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1), 156–166.

GLOVER, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5), 533–549.

GONTIER, T. (1985). Longhaul cabin crew assignment. *Dans AGIFORS Symposium Proceedings*, volume 25, pages 44–66.

GOPALAKRISHNAN, B. et JOHNSON, E. (2005). Airline crew scheduling : State-of-the-art. *Annals of Operations Research*, 140, 305–337.

GRAVES, G., MCBRIDE, R., GERSHKOFF, I., ANDERSON, D. et MAHIDHARA, D. (1993). Flight crew scheduling. *Management Science*, 39(6), 736–745.

GUPTA, U., LEE, D. et LEUNG, J. (1979). An optimal solution for the channel-assignment problem. *IEEE Transactions on Computers*, 28(11), 807–810.

HAJÒS, G. (1957). Über eine art von graphen. *Internationale Mathematische Nachrichten*, 11, 65.

HERRMANN, F. et HERTZ, A. (2002). Finding the chromatic number by means of critical graphs. *ACM Journal of Experimental Algorithms*, 7(10), 1–9.

HERTZ, A. (2003). Application des métaheuristiques à la coloration des sommets d'un graphe. *Dans* PIRLOT, M. et TEGHEM, J., éditeurs : *Résolution de problèmes de RO par les métaheuristiques*, pages 21–48. Hermes Science Publication, Paris.

HERTZ, A. (2004). Métaheuristiques. Notes du cours MTH 6311 - Optimisation Combinatoire, École Polytechnique de Montréal.

HERTZ, A. et de WERRA, D. (1987). Using tabu search techniques for graph coloring. *Computing*, 39(4), 345–351.

HERTZ, A., TAILLARD, E. et de WERRA, D. (1995). A tutorial on tabu search. *Dans Proc. of Giornate di Lavoro AIRO'95 (Enterprise Systems : Management of Technological and Organizational Changes)*, pages 13–24, Italy.

HOFFMAN, K. et PADBERG, M. (1993). Solving airline crew-scheduling problems by branch-and-cut. *Management Science*, 39, 657–682.

HOUSOS, E. et ELMROTH, T. (1997). Automatic optimization of subproblems in scheduling airline crews. *Interfaces*, 27(5), 68–77.

IRNICH, S. et DESAULNIERS, G. (2005). Shortest path problems with resource constraints. *Dans* DESAULNIERS, G., DESROSIERS, J. et SOLOMON, M., éditeurs : *Column Generation*, pages 33–65. Springer, New York, NY, USA.

JARRAH, A. et DIAMOND, J. (1997). The problem of generating crew bidlines. *Interfaces*, 27(4), 49–64.

JOHRI, A. et MATULA, D. (1982). Probabilistic bounds and heuristic algorithms for coloring large random graphs. Rapport technique, Department of Computer Science and Engineering, Southern Methodist University, Dallas, Texas.

KIRKPATRICK, S., GELATT, C. D. et VECCHI, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598), 671–680.

KLABJAN, D. (2005). Large-scale models in the airline industry. *Dans* DESAULNIERS, G., DESROSIERS, J. et SOLOMON, M., éditeurs : *Column Generation*, pages 163–195. Springer, New York, NY, USA.

KOHL, N., LARSEN, A., LARSEN, J., ROSS, A. et TIOURINE, S. (2004). Airline disruption management - perspectives, experiences and outlook. Rapport technique, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Lyngby.

KOLEN, A. et LENSTRA, J. (1995). Combinatorics in operations research. *Dans* GRAHAM, R. L., GRÖTSCHEL, M. et LOVÁSZ, L., éditeurs : *Handbook of combinatorics (vol. 2)*, pages 1875–1910. MIT Press, Cambridge, MA, USA.

KOLEN, A., LENSTRA, J., PAPADIMITRIOU, C. et SPIEKSMA, F. (2007). Interval scheduling : a survey. *Naval Research Logistics*, 54(5), 530–543.

LEIGHTON, F. (1979). A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards*, 84, 489–506.

MARCHETTINI, F. (1980). Automatic monthly crew rostering procedure. *Dans AGIFORS Symposium Proceedings*, volume 20, pages 23–59.

MAYER, M. (1980). Monthly computerized crew assignment. *Dans AGIFORS Symposium Proceedings*, volume 20, pages 93–124.

MEDARD, C. et SAWHNEY, N. (2007). Airline crew scheduling from planning to operations. *European Journal of Operational Research*, 127(3), 1013–1027.

MLADENOVIĆ, N. et HANSEN, P. (1997). Variable neighborhood search. *Computers and Operations Research*, 24(11), 1097–1100.

MOORE, R., EVANS, J. et NGO, H. (1978). Computerized tailored blocking. *Dans AGIFORS Symposium Proceedings*, volume 18, pages 343–361.

NICOLETTI, B. (1975). Automatic crew rostering. *Transportation Science*, 9, 33–42.

RUSSELL, D. (1989). Development of an automated crew bid generation system. *Interfaces*, 19, 44–51.

RYAN, D. (1992). The solution of massive generalized set partitioning problems in air crew rostering. *Operations Research*, 45, 649–661.

SARRA, D. (1988). The automatic assignment model. *Dans AGIFORS Symposium Proceedings*, volume 28, pages 23–37.

TINGLEY, G. (1979). Still another method for the monthly aircrew assignment problem. *Dans AGIFORS Symposium Proceedings*, volume 19, pages 143–203.

VILLENEUVE, D. (1999). *Logiciel de génération de colonnes.* Thèse de doctorat, École Poytechnique de Montréal, Canada.

WEIR, J. et JOHNSON, E. (2004). A three-phase approach to solving the bidline problem. *Annals of Operations Research*, 127, 283–308.

WILSON, B. (1981). BA's Regular O.R. crew planning model for the 1980's. *Dans AGIFORS Symposium Proceedings*, volume 21, pages 257–270.

ZUFFEREY, N. (2002). *Heuristiques pour les problèmes de coloration des sommets d'un graphe et d'affectation de fréquences avec polarités.* Thèse de doctorat, École Polytechnique Fédérale de Lausanne, Suisse.