Titre: Title:	Horaires mensuels en transport aérien avec équité
Auteur: Author:	Laurence Rioux-Fiset
Date:	2016
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Rioux-Fiset, L. (2016). Horaires mensuels en transport aérien avec équité [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/2217/

UNIVERSITÉ DE MONTRÉAL

HORAIRES MENSUELS EN TRANSPORT AÉRIEN AVEC ÉQUITÉ

LAURENCE RIOUX-FISET
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
AOÛT 2016

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

HORAIRES MENSUELS EN TRANSPORT AÉRIEN AVEC ÉQUITÉ

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

M. EL HALLAOUI Issmaïl, Ph. D., président
M. SOUMIS François, Ph. D., membre et directeur de recherche
M. DESAULNIERS Guy, Ph. D., membre et codirecteur de recherche
M. VILLENEUVE Daniel, Ph. D., membre

REMERCIEMENTS

Tout d'abord, j'aimerais remercier mon directeur de recherche, François Soumis, qui a cru en moi dès mes études de 1er cycle. Il m'a permis de m'initier à la recherche opérationnelle en m'offrant l'opportunité de réaliser un stage le temps d'un été au GERAD durant mon baccalauréat. Cet avant-goût m'a convaincu encore plus de mon intérêt pour le domaine et c'est avec plaisir que j'ai pu entreprendre un an plus tard ma maîtrise sous sa supervision. Un grand merci aussi à mon codirecteur Guy Desaulniers qui a toujours su trouver du temps pour répondre à mes interrogations malgré son horaire chargé.

Merci à Benoit Rochefort qui a pris le temps de m'assister dans l'aspect technique de mon projet. Merci également à Regis Sahore de AD OPT qui a eu la patience de m'initier au logiciel sur lequel j'ai travaillé et avec qui j'ai discuté de mes idées pour faire avancer le projet.

Je remercie Issmaïl El Hallaoui d'avoir accepté la présidence de mon jury ainsi que Daniel Villeneuve qui a accepté d'en être membre.

J'aimerais chaleureusement remercier toutes les personnes que j'ai côtoyées au GERAD et avec qui j'ai eu énormément de plaisir à échanger sur une panoplie de sujets. Tous ces moments de détente à se raconter nos week-ends, à jouer au *baby-foot* et à se lancer quelques blagues ont été les bienvenus ! Un merci particulier à ceux qui ont partagé le même local que moi : Cherif Sellal, Moncef Ilies Nasri, Clément Altman et Paul Javal.

Je souhaite également remercier ma mère, Dianne Rioux, mon père, Michel Fiset, ainsi que le reste de ma famille pour m'avoir supporté dans tous les projets que j'ai entrepris et dans toutes les épreuves que j'ai traversées. Un sincère merci à mes grands-parents, Simone et Roger, grâce à qui j'ai pu mener à terme toutes ces années d'études sans le moindre souci financier. Merci à mon colocataire, Guillaume Gagnon, pour sa patience et pour m'avoir permis de travailler à la maison dans un environnement calme et chaleureux. Finalement, un gros merci à Émilie Veillette pour son écoute, ses conseils et son appui inconditionnel.

RÉSUMÉ

Une compagnie aérienne doit créer pour ses pilotes et agents de bord des horaires qui minimisent ses dépenses, mais qui respectent aussi toute la règlementation en vigueur dans le transport aérien ainsi que les conventions collectives des employés. Quelques compagnies considèrent aussi la satisfaction des employés. Dans le cas qui nous intéresse, la satisfaction est liée à l'équité des horaires élaborés en terme du nombre d'heures travaillées.

Un logiciel de planification développé chez AD OPT, division spécialisée dans le transport aérien de la compagnie *Kronos*, crée des horaires mensuels avec cet objectif d'équité. Tout d'abord, un programme linéaire permet de définir pour chaque employé une cible de temps à travailler selon son historique de travail et l'ensemble des tâches à couvrir durant le mois. Ensuite, un programme en nombres entiers est résolu permettant de créer les horaires mensuels en tentant de respecter au mieux ces cibles de temps.

Comme les horaires sont mensuels, ce logiciel doit être utilisé à chaque mois pour créer de nouveaux horaires. Il est donc entendu qu'un résultat doit être obtenu dans des délais assez brefs. Or, sur la machine du GERAD qui sera utilisée pour ce mémoire, le logiciel prend 32 jours en temps CPU avant de fournir une première solution entière et cette solution n'est pas nécessairement proche de l'optimalité. Ce temps est beaucoup plus court chez AD OPT, car ils utilisent des machines multi-processeurs. Il reste quand même trop long et doit être réduit.

Ce projet de maîtrise a donc pour but de comprendre les raisons qui rendent le logiciel si lent à fournir des solutions entières et d'établir des stratégies qui permettront d'obtenir des résultats au moins aussi bons, mais en un laps de temps beaucoup plus court.

Tout d'abord, nous modifions le rôle que prend l'une des ressources dans les règles de dominance lors de la génération de colonnes. Cette modification permet d'éliminer beaucoup plus de chemins partiels et d'accélérer la recherche de plus courts chemins dans les sous-problèmes.

Ensuite, nous changeons la manière de pénaliser les jours de congé excédentaires donnés aux employés. Nous proposons une pénalisation individuelle pour chaque occurrence directement au niveau des sous-problèmes à la place de considérer le nombre total d'occurrences de l'ensemble des employés dans le problème maître et de pénaliser seulement s'il y a dépassement d'un certain seuil.

Après, nous proposons une toute nouvelle stratégie de branchement. Pour les 20 premiers branchements, nous fixons uniquement des colonnes respectant certains critères de qualité.

Les critères sont initialement très sévères pour ensuite s'adoucir à chaque fois que trop peu de colonnes sont fixées. Pour les branchements ultérieurs, nous alternons entre la fixation de tâches à des employés et la fixation de colonnes en se basant uniquement sur le flot.

Finalement, nous améliorons le calcul du nombre minimal de jours de congé que chaque employé doit avoir dans son bloc. De cette façon, la pénalisation des jours de congé excédentaires est plus réaliste et on ne paie plus pour des congés inévitables.

Tous ces changements ont permis à chaque fois de gagner en temps de résolution. Au final, on se retrouve maintenant avec une première solution entière dont la valeur a été réduite de 71.72% par rapport à celle de la solution de référence et qu'on a obtenue en 1.33 jours au lieu de 32.06, ce qui représente donc 95.85% moins de temps.

Les améliorations mentionnées ont été implémentées chez AD OPT. Les responsables ont été satisfaits autant par le temps de résolution nettement diminué que par la qualité des nouvelles solutions. Des facteurs additionnels pour considérer les préférences des employés pourront éventuellement être ajoutés, option qui auparavant n'était pas envisageable étant donné la difficulté qu'on avait à traiter l'équité seulement.

ABSTRACT

Airlines have to provide schedules for pilots and flight attendants that minimize crew costs but also respect the regulations governing the airline industry and collective agreements. In addition, some airlines aim to consider the satisfaction of their employees. In our case, satisfaction is based on equity between employee schedules in terms of hours worked.

AD OPT, a company that provides crew planning solutions for the airline industry, has developed software that creates monthly schedules with equity constraints. First, a linear program assigns a target for time worked to each employee, considering his/her work history and all the flights that need to be covered during the month. Then an integer program produces the monthly schedules, trying to respect these targets as well as possible.

The software is used on a monthly basis to create new schedules. It is understood that results must be obtained in a relatively short time. However, the software takes 32 days in CPU time on a GERAD machine before providing an initial integer solution and this solution is not necessarily close to optimality. This time is much shorter at AD OPT that uses multi-thread computing. Nevertheless, it remains too long and must be reduced.

This master's thesis examines why the software is so slow in providing integer solutions and describes new strategies to obtain equally good, if not better, solutions, in a much shorter time.

First, we modify the role played by one of the resources in the dominance rules when generating new columns. This change eliminates many more partial paths and accelerates the search for a shortest path from the source node to the sink node in each subproblem.

Secondly, we change the way that surplus days off, given to employees, are penalized. Instead of considering all occurrences across the entire body of staff and only penalizing if a threshold is exceeded, we propose a penalty for each occurrence directly in the subproblems.

Third, we propose a new branching strategy. For the first 20 branching nodes, we fix columns that meet certain quality criteria. The criteria are initially very severe then soften whenever too few columns are fixed. For subsequent branching nodes, we alternate between fixing tasks to employees and fixing columns based only on flows in subproblems.

Finally, we improve the calculation of the minimum number of days off that each employee may have in his/her schedule. In this way, the penalization of surplus days off is more realistic and we no longer pay for unavoidable days off.

All these changes helped to decrease the amount of time needed by the software to provide

solutions. We now have a first integer solution whose value is reduced by 71.72% and which was obtained in 1.33 days instead of 32.06. This represents a 95.85% reduction in time.

The mentioned improvements have been implemented at AD OPT. The software's managers were satisfied both by the decreased computation time and the quality of the new solutions. Additional constraints, considering employees' preferences, may eventually be added, an option that was previously not possible given the difficulty of dealing with equity.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

La planification optimale d'horaires de personnel joue un rôle crucial pour plusieurs entreprises. En effet, elle permet d'utiliser les employés le plus efficacement possible pour les tâches à effectuer et ainsi d'abaisser les coûts de l'entreprise reliés à la gestion de la main-d'oeuvre. C'est particulièrement important dans le domaine du transport aérien où les compagnies aériennes gèrent un nombre considérable de pilotes et d'agents de bord. Il faut créer des horaires pour chaque employé permettant de couvrir l'ensemble des liaisons offertes tout en respectant la réglementation en vigueur et les conventions collectives. Il est estimé que la main d'oeuvre représente le deuxième coût le plus important après le carburant, occupant un total de 23.4% des dépenses (Belobaba et al., 2015). Dans une industrie extrêmement compétitive, il devient alors essentiel de minimiser ce coût pour être en mesure d'améliorer la rentabilité.

1.1 Contexte du projet

AD OPT, la division de la compagnie *Kronos Canadian Systems Inc.* spécialisée dans la planification d'horaires en transport aérien, construit présentement les horaires mensuels de l'ensemble des pilotes et du personnel naviguant pour de nombreux clients. Plusieurs logiciels sont développés en fonction du mode d'attribution des horaires aux employés. Pour le client qui nous intéresse, le logiciel permet de maximiser la satisfaction globale du personnel.

Selon les spécifications précisées par le client, le logiciel peut être adapté pour considérer de nouvelles exigences. Il a ainsi été demandé que la satisfaction globale du personnel soit mesurée en fonction de l'équité au niveau du nombre d'heures travaillées par chaque employé pour le mois étudié. Plus précisément, les deux principaux objectifs sont les suivants :

1. Attribuer équitablement les tâches du mois pour que les employés aient le plus possible la même moyenne de temps travaillé par jour au travail.
2. Équilibrer l'usage annuel des employés.

Les tâches sont des rotations et des *stand-by*. Une rotation est une séquence de vols et de repos débutant et se terminant à un même aéroport et un *stand-by* est une journée en réserve pour pallier une éventuelle absence. Une même rotation peut nécessiter plusieurs agents de bord pour s'occuper des différentes responsabilités.

Afin de répondre à cette demande, un programme linéaire a été conçu pour assigner une cible

de temps de travail à faire durant le mois à chaque employé qui tient en compte les heures de vols à couvrir, la disponibilité des employés et leur historique de travail. On veut que les horaires soient ensuite construits en respectant au mieux ces cibles.

Le logiciel présentement en place est détaillé dans les trois points suivants. On y décrit les coûts, le processus d'optimisation ainsi que les stratégies de résolution adoptées.

1.1.1 Coûts

On cherche à trouver une solution en nombres entiers de coût minimal. Les coûts proviennent de trois sources différentes :

- **Pénalités pour la non-couverture** : Les tâches pour lesquelles il a été impossible d'assigner un employé sont dites *non couvertes*. On pénalise une tâche non couverte selon le niveau d'importance accordé à la tâche. Par exemple, les rotations non couvertes sont très fortement pénalisées tandis que les *stand-by* à assurer de la maison le sont beaucoup moins.
- **Pénalités sur la qualité** : Ces pénalités assurent qu'on respecte le plus possible les exigences précisées par le client sur la qualité des horaires. Par exemple, on pénalise si un employé a trop de journées de congé, s'il passe trop souvent d'un quart de travail matinal à un quart de soirée, s'il a des journées de travail vides, etc. Au total, on compte 13 pénalités différentes.
- **Pénalités sur le balancement (équité)** : Pour chaque employé, l'écart entre le temps de travail octroyé et celui ciblé est pénalisé par la fonction suivante :

$$PenalBal(Block,\ Target,\ NbWorkedDays) = \left\lfloor \frac{|Block - Target|}{NbWorkedDays} \right\rfloor^2 \qquad (1.1)$$

  où *Block* est le temps de travail en minutes de l'horaire donné à l'employé, *Target* est le temps ciblé en minutes et *NbWorkedDays* est le nombre de jours de travail de l'employé. Un jour de travail est un jour pour lequel l'employé a au moins une tâche à son horaire. Les autres jours sont des journées en vacances ou en congé.

Une solution optimale sera donc une solution en nombres entiers de valeur minimale pour la somme totale de ces trois sources de coûts.

Les pénalités pour la non-couverture ainsi que les pénalités sur la qualité sont détaillées dans l'annexe A.

1.1.2 Processus d'optimisation

Le processus d'optimisation se déroule en trois phases consécutives :

1. **Minimisation de la non-couverture** : Pour cette première phase, les pénalités de qualité et de balancement sont inactives et les contraintes d'intégrité sont relaxées. L'objectif du programme est uniquement de minimiser les coûts reliés à la non-couverture de tâches. Le nombre total de tâches non couvertes dans la solution obtenue est passé à la phase suivante comme un nombre d'occurrences permis avant d'appliquer la pénalisation. Seules les tâches non couvertes excédant ce seuil seront pénalisées.
2. **Minimisation des pénalités sur la qualité** : Pour cette deuxième phase, on active maintenant les pénalités de qualité et on utilise les nouveaux seuils de non-couverture de la phase précédente. Les contraintes d'intégrité sont relaxées. L'objectif du programme est de minimiser les coûts reliés aux pénalités de qualité et de non-couverture uniquement. Le nombre d'occurrences de chacune des pénalités de qualité dans la solution obtenue est transmis à la phase suivante comme un nombre d'occurrences autorisées avant d'appliquer la pénalisation correspondante. Par exemple, si 100 occurrences sont obtenues pour une certaine pénalité de qualité, on commencera à pénaliser dans la phase 3 à partir de la 101^{e} occurrence seulement.
3. **Balancement** : Pour cette troisième phase, on résout tout d'abord un programme linéaire permettant d'attribuer une valeur cible de temps de travail pour chaque employé. Ensuite, un programme en nombres entiers est résolu avec les seuils obtenus des deux phases précédentes. C'est ici que sont activées les pénalités de balancement. La solution optimale finalement obtenue fournira les horaires finaux des employés d'équité maximale avec des tâches non assignées et des violations sur la qualité contenues dans la mesure du possible sous les seuils calculés dans les deux phases précédentes.

En résumé, dans les deux premières phases, on ne cherche qu'à avoir un ordre de grandeur des contraintes à imposer dans la phase finale afin de ne pas pénaliser des éléments inévitables. Comme les sauts d'intégrité des programmes de ces deux phases ne sont pas grands, les relaxations linéaires fournissent de bonnes approximations du nombre d'occurrences auquel s'attendre pour chacune des pénalités. Le programme principal considérant l'équité est résolu dans la troisième et dernière phase et se conclut par l'obtention des horaires mensuels.

1.1.3 Stratégies utilisées

Lors des trois phases d'optimisation, on utilise la méthode de génération de colonnes pour résoudre les programmes linéaires (phases 1 et 2) et le programme linéaire en nombres entiers (phase 3). Pour chaque itération de la génération de colonnes, le problème maître est résolu avec l'algorithme *barrier* (points intérieurs) de *CPLEX*. Ensuite, de nouvelles colonnes sont générées par l'intermédiaire de sous-problèmes considérant les variables duales obtenues du problème maître. On a un sous-problème par employé, représenté par un réseau dans lequel les divers chemins de la source au puits décrivent tous les blocs mensuels possibles pour l'employé correspondant. Le plus court chemin représente donc l'horaire de plus petit coût réduit. Plusieurs horaires de coûts réduits négatifs peuvent être générés pour un même employé à chaque itération.

Pour parcourir l'arbre de branchement, on utilise une stratégie d'exploration en profondeur. Dans le cas où on a atteint une feuille de l'arbre, c'est la stratégie meilleur d'abord qui est utilisée pour choisir le prochain noeud.

À chaque noeud visité de l'arbre, on recourt à 5 configurations des valeurs des paramètres pour la résolution des sous-problèmes. Celles-ci diffèrent par les valeurs attribuées à quelques paramètres et par les ressources sur lesquelles on s'appuie pour la dominance. Les deux premières configurations, HAFEAS et HBKFEAS, permettent de se rendre à une solution réalisable et les trois suivantes, HA, HB et HBK, améliorent la solution. Ces trois configurations d'amélioration sont toujours utilisées dans le même ordre : on commence par la plus approximative et rapide pour finir par la plus précise. La dominance se base sur beaucoup plus de ressources lorsqu'on passe à la configuration suivante.

À la fin de chaque noeud, on procède au branchement ou on élague le noeud s'il devient impossible d'améliorer la solution courante en empruntant celui-ci. Pour brancher, la méthode Tsplit est utilisée. Trois noeuds fils sont créés, le premier pour lequel 100 tâches au maximum sont imposées à des employés, le deuxième pour lequel une seule tâche est imposée à un employé et le troisième pour lequel cette même tâche est interdite à l'employé. Pour imposer ou intedire une tâche, on modifie légèrement les sous-problèmes. Ainsi, imposer une tâche à un employé nécessite le retrait de l'arc correspondant à cette tâche des réseaux de tous les autres employés tandis que l'interdire nécessite le retrait de l'arc du réseau de l'employé concerné seulement. Le branchement ne crée pas une partition disjointe du domaine de solution, mais ce dernier est entièrement couvert. On attribue un score entre 0 et 1 à chaque affectation tâche/employé. Pour le premier noeud fils, si plus de 100 de ces affectations ont un score en haut de 0.51, on choisit les 100 meilleures. Autrement, on choisit toutes celles qui ont un score supérieur à 0.51. Pour le deuxième noeud fils, on impose l'affectation de meilleur score

seulement et pour le troisième noeud fils, on interdit cette même affectation.

La résolution se conclut lorsque le premier scénario parmi les suivants se produit : 200 noeuds ont été explorés depuis qu'une première bonne solution a été trouvée, 20 feuilles de l'arbre ont été atteintes depuis qu'une solution a été trouvée (pas nécessairement qualifiée de bonne), la solution qui vient d'être trouvée est assez proche de la solution relaxée ou il ne reste plus de noeuds à explorer. Une solution est qualifiée de *bonne* si la différence relative par rapport à la solution relaxée est inférieure à 10%. Une solution est assez proche de la solution relaxée si la différence relative est inférieure à .001% fois le nombre de noeuds explorés. Beaucoup de noeuds sont élagués au cours du processus. Si la différence relative entre la solution entière courante et la borne inférieure d'un noeud est en deçà de 3%, on élague ce noeud.

On travaille donc avec une résolution qui est heuristique plutôt qu'exacte. En effet, on veut accélérer le processus et obtenir de bonnes solutions rapidement au lieu d'obtenir la meilleure sans tenir compte du temps de calcul.

1.2 Problématique

Chaque employé est lié à une base, soit un aéroport, de laquelle toutes ses rotations doivent débuter et se terminer. Le problème d'horaires est séparable par base, par famille d'avions et par poste (pilote, copilote et agent de bord). Le logiciel actuel cause problème durant les mois les plus achalandés pour les bases du client auxquelles sont affectés un très grand nombre d'employés. De juin à août, beaucoup de liaisons sont offertes pour répondre à la forte demande estivale. Certaines bases dans les villes où le traffic est le plus important peuvent nécessiter jusqu'à 1000 agents de bord pour couvrir les rotations assurées par les avions d'une même famille. Dans de telles conditions, le processus d'optimisation prend énormément de temps avant de fournir une toute première solution entière. Même si cette dernière n'est pas nécessairement proche de l'optimalité, on doit malgré tout s'en contenter puisque des horaires doivent être remis au client.

Les horaires étant mensuels, le logiciel doit rouler chaque mois pour créer de nouveaux horaires. Dans un tel contexte, il est essentiel d'être en mesure de fournir les horaires dans un délai raisonnable. En effet, un problème dans la solution obtenue ou des changements d'employés ou de vols dans le mois suivant pourraient contraindre à relancer une optimisation. De plus, une nouvelle demande du client pour que soient considérées les préférences des employés nécessitera des ajouts au logiciel actuel et le temps de résolution s'en trouvera donc allongé.

En bref, le temps de résolution est beaucoup trop important et doit absolument être réduit. Aussi, trouver une solution entière plus rapidement permettrait potentiellement d'en trouver

plusieurs au lieu d'une seule. On aurait la possibilité de continuer le processus d'optimisation pour trouver d'autres solutions entières de meilleur coût au lieu de se satisfaire du premier résultat.

1.3 Données étudiées

AD OPT a fourni au GERAD les données de 6 mois différents de l'année 2015. Pour ces mois, on veut construire les horaires mensuels des agents de bord de la plus grande base du client en assignant les rotations assurées par une famille d'avions d'une capacité d'environ 150 passagers. Les problèmes à résoudre sont de taille considérable, le nombre total de contraintes variant entre 15028 et 19388 (affectation et couverture). Le nombre de tâches préassignées et à assigner ainsi que le nombre d'employés à considérer sont détaillés dans le tableau 1.1.

Tableau 1.1 Description des 6 ensembles de données

	Avril 2015		**Juin 2015**		**Juillet 2015**	
	983 employés		1022 employés		1066 employés	
Tâches	Nb	Temps	Nb	Temps	Nb	Temps
Préassignées	8146	1521h08	11836	1693h12	10701	1295h50
À assigner	15039	69520h11	17537	73239h17	18322	79798h27
Rotations	9039	61750h05	9884	66698h05	10414	72890h30
Stand-by	6000	7770h06	7653	6541h12	7908	6907h57
TOTAL	23185	71041h19	29373	74932h29	29023	81094h17

	Août 2015		**Septembre 2015**		**Octobre 2015**	
	1030 employés		1008 employés		996 employés	
Tâches	Nb	Temps	Nb	Temps	Nb	Temps
Préassignées	9461	1346h35	8400	1288h40	8756	2048h25
À assigner	13998	80249h23	17698	74958h19	17478	70918h00
Rotations	10555	73363h50	9877	67986h10	9375	63835h45
Stand-by	3443	6885h33	7821	6972h09	8103	7082h15
TOTAL	23459	81595h58	26098	76246h59	26234	72966h25

Les données d'avril 2015 sont celles qui ont été fournies au tout début du projet et avec lesquelles la majeure partie des tests a été effectuée. Les 5 autres mois n'ont été mis à la disposition du GERAD qu'à la toute fin du projet afin de confirmer la validité des résultats. C'est donc le mois d'avril qui est utilisé pour illustrer la problématique et définir les objectifs. Les autres données ne serviront qu'à appuyer le bon fonctionnement des améliorations proposées.

Les deux premières phase du processus de résolution s'exécutent dans des temps raisonnables

puisque les contraintes d'intégrité sont relaxées. C'est la denière phase qui est problématique, soit celle du balancement, résolue en nombres entiers. Une exécution de référence tournée sur une machine au GERAD avec tous les paramètres identiques à ceux utilisés chez AD OPT donne les temps indiqués dans le tableau 1.2.

Tableau 1.2 Exécution de référence - Avril 2015

Phase	temps CPU
Phase 1 : Pénalités non-couverture	1h19
Phase 2 : Pénalités qualité	34h23
Phase 3 : Balancement *	769h24

* Phase non complétée

La phase 3 a été arrêtée manuellement lors de l'obtention de la première solution entière au bout d'un peu plus de 32 jours de travail en temps CPU. Au-delà de ce temps, il aurait été excessif de continuer la résolution. La solution entière trouvée est décrite dans le tableau 1.3. On y indique le coût total des 3 différents types de pénalités. Elle servira de référence dans la suite du projet.

Tableau 1.3 Solution de référence - Avril 2015

Valeur :	76121
Pénalités non-couverture :	120
Pénalités qualité :	19290
Pénalités balancement :	56711

Chez AD OPT, le temps réel écoulé avant d'obtenir une solution est de l'ordre de 10 jours. L'utilisation de machines dotées de plusieurs processeurs permet de paralléliser le traitement et donc d'accélérer la résolution.

1.4 Objectifs de recherche

On vise tout d'abord à identifier précisément les éléments qui rendent la résolution de la phase de balancement aussi longue. On voudra ensuite proposer des méthodes qui corrigeront les points faibles trouvés et qui permettront ainsi d'obtenir des solutions entières beaucoup plus rapidement.

La solution de référence ayant pris un peu plus de 32 jours de calcul, on cherche à obtenir une solution entière en moins de 3 jours, soit un temps CPU réduit par un facteur 10. Cette solution devra être au moins aussi bonne en terme de valeur que celle de référence, donc inférieure ou égale à 76121.

On testera les améliorations proposées avec chacun des 5 autres mois de données pour s'assurer que les gains en temps de résolution ne soient pas attibuables au hasard et qu'elles pourront réellement servir chez AD OPT dans les futures mises à jour de leur logiciel.

1.5 Plan du mémoire

Une revue de littérature sera présentée au chapitre 2 dans le but de donner un aperçu des différentes techniques développées jusqu'à présent pour construire les horaires mensuels des pilotes et agents de bord en transport aérien. Après, un diagnostic des causes de ralentissement dans la phase de balancement sera détaillé au chapitre 3. Au chapitre 4, on décrira les améliorations proposées afin d'accélérer le logiciel pour ensuite traiter de leur implémentation chez AD OPT au chapitre 5. Finalement, on conclura au chapitre 6 en soulevant les points forts et les points faibles du projet et en mentionnant de nouvelles avenues de recherche.

CHAPITRE 2 REVUE DE LITTÉRATURE

Le problème de planification d'horaires d'équipages dans l'industrie du transport aérien a été un sujet de recherche particulièrement étudié au cours des 50 dernières années. Étant donné la complexité du problème, il est souvent divisé en deux. On construit tout d'abord les rotations pour ensuite construire les blocs où on assigne ces rotations aux pilotes et agents de bord. Dans la plupart des cas, ces blocs sont mensuels. Les rotations sont celles qui permettent de couvrir l'ensemble des vols de la compagnie à coût minimum et qui respectent toute la réglementation régissant le transport aérien. Les blocs mensuels peuvent ensuite être construits selon divers objectifs, mais doivent dans tous les cas considérer les contraintes découlant des conventions collectives des employés.

Trois approches différentes sont possibles pour la construction de blocs mensuels : les blocs anonymes, les blocs personnalisés et les blocs personnalisés avec considération de la séniorité. Les blocs anonymes (*bidlines*) sont des horaires mensuels construits sans être assignés à un employé en particulier. L'assignation s'effectue après coup au moyen d'enchères que font les employés sur les horaires proposés. Les blocs personnalisés (*rostering*) visent à construire des horaires qui s'adaptent le plus possible aux préférences des employés. On maximise la satisfaction du personnel tout en s'assurant que les rotations soient couvertes et que les objectifs de la compagnie soient atteints. Finalement, les blocs personnalisés avec considération de la séniorité (*preferential bidding*) ont le même objectif, mais allouent une priorité aux préférences des employés avec le plus de séniorité.

Plusieurs articles et chapitres de livre résument l'état des recherches pour chacune de ces trois approches, notamment Desaulniers et al. (1998), Barnhart et al. (2003), Gopalakrishnan et Johnson (2005), Klabjan (2005) et plus récemment Kasirzadeh et al. (2015). Pour cette revue de littérature, on se concentrera sur la fabrication des blocs mensuels. On révisera tout d'abord l'état de l'art pour chacune des trois approches présentées ci-haut et on concluera en présentant ce qui a été fait jusqu'à présent pour considérer l'équité.

2.1 Construction de blocs mensuels

2.1.1 Blocs anonymes

Dans l'article de Beasley et Cao (1996), on présente un algorithme utilisant la relaxation lagrangienne d'un programme en nombres entiers pour assigner N tâches à K employés. Les multiplicateurs de cette relaxation sont mis à jour avec la méthode du sous-gradient. Une

borne inférieure est ainsi calculée. Un arbre de branchement intégrant cette borne inférieure permet d'atteindre l'optimalité. Les auteurs ont testé leur algorithme avec un problème de 500 tâches à assigner à 208 employés. Celui-ci a pu être résolu à l'optimalité en 1 heure 23 minutes.

Jarrah et Diamond (1997) ont réalisé un système de planification semi-automatique pour une compagnie aérienne américaine. Les planificateurs décident tout d'abord d'un ensemble de paramètres de qualité et des colonnes s'accordant à ces paramètres sont ensuite générées par le système. Un modèle de partionnement d'ensemble ayant pour variables ces colonnes maximise finalement la couverture des rotations en utilisant un minimum de blocs. Les planificateurs analysent les résultats et sélectionnent les blocs qu'ils veulent conserver. Le processus est répété avec de nouveaux paramètres jusqu'à obtention d'une solution complète. Ces travaux ont permis de diminuer significativement le temps requis pour générer des blocs et d'améliorer la qualité de ces derniers par rapport à ce qui était obtenu auparavant avec le système alors en place qui demandait beaucoup plus de décisions et de travail de la part des planificateurs. Pour la couverture, un maximum de 3.51% du temps total des rotations n'a pu être assigné dans les tests effectués.

Campbell et al. (1997) ont développé pour la compagnie de transport international FedEx un générateur de blocs mensuels afin de déterminer rapidement les impacts de nouvelles règles de travail dans le cadre de négociations avec l'association de leurs pilotes. On désire minimiser le nombre de blocs requis et les rotations non couvertes et maximiser la pureté des blocs. La pureté est synonyme de régularité dans le bloc. Plus il y a de variations, moins la pureté est grande. Par exemple, un bloc dont toutes les rotations sont en avant-midi et dont toutes les destinations sont domestiques aura une très grande pureté comparativement à un bloc variant entre des destinations domestiques et internationales en avant-midi et en après-midi. La métaheuristique de recuit simulé est utilisée. On débute par une solution possible et de petits ajustements sont apportés de manière incrémentale. Une fois le plus grand nombre possible de bons blocs obtenus, les rotations encore non couvertes sont distribuées par un algorithme glouton. Cette nouvelle méthode a permis à FedEx de construire leurs blocs mensuels en un maximum de 10 heures pour leur plus grosse base.

Christou et al. (1999) ont développé pour Delta Air Lines un algorithme génétique se déroulant en deux phases pour remplacer le processus semi-automatique qui était jusqu'alors utilisé. L'objectif est de maximiser la qualité et la valeur moyenne des blocs mensuels. La valeur d'un bloc est son temps crédité total. En maximisant ce temps, on favorise une bonne paie pour les employés qui ont un bloc tout en réduisant pour la compagnie le nombre total d'employés à utiliser. La qualité d'un horaire est mesurée par sa pureté. La première phase

construit donc le plus grand nombre possible de blocs d'excellente qualité. La deuxième phase complète ces blocs avec les rotations qui ne sont pas encore couvertes en visant la plus grande valeur possible. Cette nouvelle approche a permis de construire l'ensemble des blocs mensuels domestiques chez Delta en moins de 2 jours, tâche qui requérait entre 4 et 6 jours précédemment. La régularité s'est elle aussi grandement améliorée. Initialement, entre 4% et 19% des blocs étaient constitués de rotations de durée similaire et débutant le même jour de la semaine. En utilisant le nouvel algorithme, entre 27% et 41% des blocs ont maintenant cette caractéristique. Quant au nombre total de blocs, il a légèrement diminué ce qui permet donc d'utiliser moins d'employés.

Weir et Johnson (2004) ont proposé une méthode divisant la construction des blocs anonymes en 3 phases. Le but est de tenir compte de la qualité de vie des employés en visant une certaine régularité dans leur horaire et leur permettre ainsi de conserver des habitudes de sommeil. La première phase cherche à développer des patrons avec l'ensemble des rotations en résolvant un programme linéaire mixte. La phase 2 utilise ces patrons pour construire des horaires réalisables en tentant de couvrir dans la mesure du possible l'ensemble des rotations. Un modèle de partitionnement d'ensemble est utilisé. Finalement, la phase 3 assigne les rotations restantes si nécessaire. Les tests effectués ont permis d'obtenir des horaires demandant aux employés d'ajuster une fois au maximum leur cycle de travail.

2.1.2 Blocs personnalisés

Au sein des compagnies aériennes, les préférences sont très souvent considérées à l'aide d'un système d'enchères. Un employé peut donc indiquer ses préférences en attribuant un certain poids à des éléments précis tels que des rotations, des week-ends de congé, des périodes de repos, des départs matinaux, des destinations, etc. Il peut aussi y avoir des éléments caractérisant l'horaire global, tels que le nombre d'heures travaillées plus grand ou plus petit qu'une certaine valeur, un nombre total de jours de congé, etc. Ces éléments pour lesquels il y a enchère sont propres à chaque compagnie, mais le principe demeure le même. Par exemple, 75 éléments sont utilisés chez Air Canada (Gamache et al., 1998). La valeur d'un horaire peut alors être déterminée par les poids qu'a attribués l'employé aux divers éléments qui le composent.

Day et Ryan (1997) décrivent l'approche qu'ils ont développée pour les vols domestiques de la compagnie Air New Zealand. Les blocs sont construits en trois étapes. Tout d'abord, les jours de congé sont attribués aux employés afin que chacun en ait le même nombre. Tous les agencements de jours de congé pour la période étudiée sont générés et un modèle de partionnement les assigne sous la contrainte qu'il y ait un certain nombre d'employés au travail

à chaque jour. Ensuite, les rotations sont affectées selon les congés déterminés précédemment. Des blocs légaux sont générés pour chaque employé et un modèle de partitionnement d'ensemble est résolu de telle sorte que les blocs choisis respectent toute la réglementation. Finalement, une analyse de la solution est faite pour apporter des améliorations au niveau de l'équité entre les employés afin que les charges de travail les plus demandantes soit également réparties. Cette approche a été implémentée chez Air New Zealand en 1993 et a permis d'obtenir des blocs de plus grande qualité en beaucoup moins de temps, et ce, avec un seul employé affecté à la construction des horaires au lieu de 4 précédemment.

Gamache et Soumis (1998) ont présenté la première méthode exacte pour le problème de blocs personnalisés. La génération de colonnes est utilisée pour résoudre la relaxation linéaire d'un modèle de partitionnement d'ensemble. Pour chaque employé, les colonnes sont obtenues en solutionnant un problème de plus court chemin dans un réseau. Les noeuds sont associés aux rotations et les arcs définissent les passages possibles entre les rotations et les congés. Un algorithme de *branch-and-bound* intégrant cette génération de colonnes mène à une solution entière. Pour diminuer le temps de résolution, une stratégie de génération de colonnes disjointes est proposée, colonnes qui ne contiennent pas de tâches en commun. Une alternative est aussi présentée permettant de considérer les demandes spéciales des employés durant l'optimisation au lieu de le faire traditionnellement avant la construction des blocs.

Cette méthode n'est pas adaptée pour les très gros problèmes avec beaucoup d'employés et de rotations. Des stratégies de contrôle dans la génération de colonnes ont donc été développées par Gamache et al. (1999) qui ont permis l'obtention de solutions entières de grande qualité dans des temps très raisonnables. Au final, des tests effectués par les chercheurs ont permis de vérifier que ces stratégies permettent de diminuer le temps de résolution par un facteur 1000. La méthode incluant ces stratégies de contrôle a été implémentée chez Air France. Une comparaison avec le programme CADET auparavant utilisé a permis de constater que le nouvel optimiseur est en mesure d'assigner toutes les tâches en utilisant 6.2% moins d'employés.

El Moudani et al. (2001) ont proposé une approche heuristique bi-critère qu'ils ont testée pour des problèmes de taille moyenne. L'objectif est d'obtenir des blocs minimisant les coûts d'opération pour la compagnie aérienne tout en atteignant un certain niveau global de satisfaction chez les employés. Divisée en deux étapes, l'approche produit tout d'abord un ensemble de blocs procurant un haut degré de satisfaction pour ensuite passer à un processus d'optimisation s'appuyant sur un algorithme génétique qui permet de réduire les coûts.

Dawid et al. (2001) traitent d'un algorithme qu'ils ont développé, nommé SWIFTROSTER. Cet algorithme a comme particularité de faire appel à des stratégies basées sur les spécifités du problème de blocs personnalisés dans le but de fortement réduire les temps d'exécution pour

le traitement de grandes instances. SWIFTROSTER se base sur une énumération implicite récursive faisant appel à la propagation. Celui-ci a été testé pour une compagnie aérienne européenne de taille moyenne et les résultats ont démontré une amélioration considérable au niveau du temps de calcul pour fournir de premières solutions réalisables comparativement aux systèmes alors en place. Pour 779 agents de bord et 1711 rotations, l'algorithme a fourni des blocs mensuels en une dizaine de minutes.

Fahle et al. (2002) utilisent la génération de colonnes basée sur la programmation par contraintes. Habituellement, le sous-problème est un problème de plus court chemin avec contraintes de ressources résolu par programmation dynamique. Les auteurs proposent une formulation du sous-problème comme un problème de satisfaction de contraintes et utilisent un algorithme de programmation par contraintes pour le résoudre. Il y a une ou plusieurs contraintes pour chaque règlement de la compagnie aérienne et une contrainte encapsulant un algorithme de plus court chemin. Le problème maître est quant à lui un programme en nombres entiers. Les auteurs affirment qu'une telle formulation pour le sous-problème permet de considérer des règlements plus complexes qui pourraient normalement limiter l'usage de la génération de colonnes usuelle.

Sellmann et al. (2002) ont approfondi le travail de Fahle et al. (2002) en proposant une heuristique d'arbre de recherche basée sur la programmation par contraintes. Chaque variable correspond à une rotation à assigner dont le domaine est l'ensemble des employés pouvant l'assurer. Dans l'arbre de recherche, chaque niveau est associé à l'assignation d'une rotation. Chaque noeud feuille représente donc une solution faisable décrite par le chemin de la racine à la feuille. Les auteurs montrent ensuite comment il est possible de combiner cette heuristique avec l'approche de génération de colonnes basée sur la programmation par contraintes de Fahle et al. (2002) de façon à créer un algorithme hybride qui permet de surmonter les faiblesses des deux méthodes utilisées seules. Une implémentation prototype a été développée, mais aucun produit pouvant rivaliser avec les codes développés dans l'industrie n'est disponible, ce qui a empêché toute comparaison avec les méthodes alors en place.

Kohl et Karisch (2004) offrent une revue de littérature sur le sujet des blocs personnalisés. Ils présentent ensuite en détail les algorithmes et méthodes employés dans le système *Carmen Crew Rostering* au développement duquel ils ont collaboré, utilisé entre autres par British Airways, KLM et Scandinavian Airlines. Ils montrent ainsi comment un problème de construction de blocs est résolu dans l'industrie en mettant en évidence les considérations pratiques. Les trois grands composants du système sont abordés : le générateur, l'optimiseur et l'évaluateur des règlements.

Maenhout et Vanhoucke (2010) ont proposé l'utilisation d'une méta-heuristique évolution-

niste dans laquelle les solutions sont combinées pour en générer de meilleures. On considère la qualité de la solution comme une moyenne pondérée des coûts d'opérations et du niveau de respect des préférences. Des ensembles de solutions initiales sont tout d'abord générés et, ensuite, des paires de ces solutions de référence sont combinées selon une stratégie de sélection. Une solution ainsi créée conserve ou améliore les meilleures caractéristiques des deux solutions parents. L'utilisation de la méta-heuristique chez Brussels Airlines pour le mois de juin 2006 a permis de réaliser des économies de €22500 comparativement à la solution manuelle qui avait été construite. La charge de travail a été plus équitablement répartie chez les pilotes de 40% selon les indicateurs d'équité utilisés. L'algorithme était aussi en mesure de beaucoup mieux satisfaire les demandes des pilotes.

Kasirzadeh et al. (2015) se sont penchés sur la construction de blocs personnalisés pour pilotes considérant les activités préassignées et deux types de préférences : les vacances et les vols. Un modèle de partionnement d'ensemble renforcé pour ces préférences spécifiques est présenté. L'objectif est de couvrir l'ensemble des rotations une seule fois avec un coût d'opération minimum tout en satisfaisant un nombre minimal de préférences des deux types. Pour résoudre le modèle, les auteurs recourent à la génération de colonnes intégrée dans un algorithme de *branch-and-bound*. Il y a un sous-problème de génération de colonnes par pilote défini sur un réseau espace-temps acyclique dans lequel on recherche le plus court chemin sous contraintes de ressources. Les branchements utilisés dans le *branch-and-bound* consistent à fixer des colonnes et des inter-tâches. Les tests ont donné de bons résultats. En moyenne, 29.09% des vols assignés correspondent aux préférences des pilotes et 13.76% des pilotes ont les vacances qu'ils avaient demandées dans leur horaire.

2.1.3 Blocs personnalisés avec considération de la séniorité

L'article de Gamache et al. (1998) traite de la construction séquentielle d'horaires mensuels personnalisés chez Air Canada considérant les préférences selon la séniorité. Pour ce faire, un problème résiduel est résolu pour chaque employé, du plus ancien au plus récent. Ainsi, un programme en nombre entiers donne le score maximal que l'employé peut avoir considérant les rotations et employés restants. Ce programme est résolu à l'aide de la génération de colonnes intégrée dans un algorithme de *branch-and-bound*. La solution optimale fournit le bloc de meilleur score possible pour l'employé tout en assurant une solution réalisable pour les autres. Son horaire est alors fixé et on passe à l'employé suivant en enlevant au problème les rotations nouvellement couvertes. Les résultats montrent que les plus grands problèmes ont été résolus entre 1 et 8 heures, le plus long ayant 82 pilotes et 602 rotations. Un système basé sur cette méthode est en place chez Air Canada depuis 1995.

Il peut arriver qu'il devienne impossible pour les employés moins anciens de couvrir les rotations résiduelles. Dans un tel cas, un retour en arrière (backtracking) est fait pour revenir sur les mauvaises décisions prises antérieurement. Ces retours en arrière sont souvent coûteux en terme de temps. Pour les éviter, Gamache et al. (2007) ont conçu un test permettant de vérifier la faisabilité des blocs pour les employés qui restent considérant les rotations encore à couvrir. Ce problème est modélisé par une coloration de graphe. Une coloration du graphe proposé existe si et seulement si une solution faisable existe. Un algorithme de recherche tabou est présenté pour résoudre ce problème. Cette amélioration a permis de diminuer grandement le nombre de retours en arrière.

Achour et al. (2007) proposent une toute première méthode exacte. Tout comme il a été présenté par Gamache et al. (1998), la résolution est séquentielle et des programmes en nombres entiers sont résolus pour chaque employé en ordre d'ancienneté, du plus ancien au plus jeune. La différence est au niveau de la fixation des blocs. On peut en fixer aucun ou plusieurs à chaque résolution d'un programme. Le bloc d'un employé est fixé seulement lorsqu'un unique horaire de score maximal peut donner lieu aux scores optimaux pour les employés moins anciens. On considère donc pour un employé tous les blocs procurant le score maximal et on élimine progressivement ceux qui n'engendreront pas les scores optimaux chez les plus jeunes. Aussi, des rotations peuvent être identifiées comme essentielles à un employé senior pour que celui-ci atteigne son score maximal. On les écarte alors des réseaux des autres employés moins anciens et on élimine toutes les colonnes les contenant. Cette approche a été testée avec des données d'Air Canada et a permis d'améliorer significativement les horaires de leurs pilotes grâce à une plus grande qualité dans les solutions.

2.2 Traitement de l'équité

Boubaker et al. (2010) adressent pour la toute première fois le problème de planification d'horaires mensuels anonymes avec équité. L'équité est considérée par rapport au nombre de jours de congé et au nombre d'heures payées qui doivent le plus possible être les mêmes d'un horaire à l'autre. Pour ce faire, un modèle de partitionnement d'ensemble est proposé dont la fonction objectif cherche à minimiser la somme des variances dans le nombre d'heures payées et de jours de congé par horaire. Un poids permet d'accorder plus d'importance à la distribution des congés ou, inversement, à la distribution des heures payées selon la priorité de la compagnie aérienne. Deux méthodes sont présentées pour résoudre un tel modèle : une heuristique de *branch-and-price* explorant une seule branche de l'arbre de branchement et une heuristique d'agrégation dynamique des contraintes qui, combinée à la première, permet d'améliorer le temps de résolution. Les auteurs ont réussi à produire une solution en un peu

moins d'une heure à l'aide de la deuxième méthode pour un problème de 2924 rotations à assigner à 564 pilotes.

Cette approche est la seule présentée dont le but principal est l'équité. On considère l'équité dans plusieurs articles cités précédemment, mais toujours comme un objectif secondaire.

Par exemple, Day et Ryan (1997) font un prétraitement avant la construction des blocs de manière à calculer le minimum et le maximum d'occurrences de chacune des tâches les moins désirables (rotations de plus de 10 heures, rotations de nuit, être sur appel, etc) qu'on peut donner à un employé. Ainsi, il y a une certaine équité dans la répartition de ces tâches. On s'assure aussi que le nombre total d'heures travaillées ne dépasse pas une limite.

Maenhout et Vanhoucke (2010) ont comme second objectif après la minimisation des coûts de préserver une équité parmi l'ensemble des employés. Pour les éléments qu'on souhaite être répartis de manière équitable, on pénalise la déviation par rapport à la valeur moyenne des contraintes de consommation de ressources associées. Par exemple, pour Brussels Airlines, on assure l'équité au niveau des éléments suivants : le temps en vol, le temps de travail total, le nombre d'escales et le temps passé à l'extérieur de la base.

CHAPITRE 3 DIAGNOSTIC

Dans ce chapitre, on abordera dans la première section les outils qui ont été développés au fil du projet afin de mieux analyser le journal (*log file*) de l'activité interne du logiciel.

Les deux grandes sections suivantes traiteront en détail des composants du logiciel qui ont été identifiés comme prenant trop de temps dans le processus d'optimisation. Ce diagnostic se base sur l'analyse de la résolution originale du mois d'avril 2015. Dans chacune de ces sections, on commencera tout d'abord par expliquer en détail le fonctionnement actuel du composant pour ensuite décrire les éléments qui le rendent problématique. Finalement, on mentionnera brièvement la direction de recherche qui devra être prise pour corriger la situation et favoriser un meilleur temps de résolution.

3.1 Outils d'analyse développés

Cinq programmes en Python ont été écrits qui permettent d'afficher des informations sur la résolution ou de générer divers fichiers texte de statistiques. Ils seront décrits successivement.

sol_analysis.py

Entrée : journal d'activités *bh.log*
Sortie : affichage à l'écran

Ce programme affiche les détails de toutes les solutions entières trouvées lors de la résolution. Ces détails sont présentés dans le même ordre qui sera utilisé lorsque seront exposés les résultats dans le chapitre 4. On retrouve entre autres le temps CPU requis pour arriver à la solution, la valeur de cette dernière et la contribution de chacune des trois différentes pénalités présentées dans l'introduction au coût total.

L'avantage d'un tel programme est qu'on peut l'utiliser pour avoir très rapidement accès aux solutions obtenues lors des tests effectués ainsi qu'à toutes les informations qui permettent de juger de la qualité de ces solutions. Auparavant, il fallait fouiller le journal d'activités et consulter séparément tous les rapports signalant une nouvelle solution entière. En plus de prendre beaucoup de temps, on pouvait mal comparer l'ensemble des solutions, car il fallait naviguer de rapport en rapport. Maintenant, toutes les solutions sont résumées en une seule ligne chacune et sont imprimées l'une à la suite de l'autre.

Le programme lance des avertissements si des anomalies se produisent. Une alerte est donc

émise lorsque la somme des pénalités n'égale pas la valeur de la solution et lorsqu'à un rapport de nouvelle solution l'itération associée n'est pas trouvée. De cette façon, on s'assure que tous les détails de la solution sont cohérents et qu'on affiche le bon temps de résolution correspondant étant donné qu'il est indiqué dans les lignes des itérations seulement.

Aussi, il arrive très souvent dans le journal que plusieurs rapports de solution soient imprimés à un même noeud pour signaler des solutions de même valeur ou avec d'infimes améliorations. Dans le but de ne pas alourdir inutilement l'affichage du programme, on ne considère pour un même noeud que le dernier rapport produit, soit celui avec la solution de plus petite valeur.

read_bal.py

Entrée : journal d'activités *bh.log*
Sortie : *read_bal.txt*

Comme l'un des principaux buts du processus d'optimisation est de créer des horaires équitables, il a été jugé pertinent d'écrire un programme qui permet d'étudier plus attentivement l'atteinte de cet objectif. Un fichier texte est ainsi produit qui, pour chaque employé, indique le temps de travail ciblé, le temps de travail obtenu dans la solution finale, le nombre de jours de travail et le coût de la pénalité de balancement calculé à partir de ces trois dernières données.

Ces informations n'étaient pas toutes disponibles initialement dans le journal d'activités. Il a donc été ajouté au code des impressions qui permettent d'y afficher tout juste après leur calcul les cibles de temps de travail de chaque employé. À la toute fin de la résolution sont aussi ajoutés leur temps de travail finalement obtenu de même que leur nombre de journées de travail.

Un bref coup d'oeil au fichier texte généré donne la possibilité d'évaluer le respect des cibles et de repérer les employés pour qui le temps de travail de l'horaire final est très éloigné de celui espéré.

txt_maker.py

Entrées : journal d'activités *bh.log*, liste des tâches du mois *act.in*
Sorties : 22 fichiers texte détaillés dans l'annexe B

Ce programme génère tous les fichiers texte de statistiques. Les données sont disposées dans la plupart de ces fichiers de manière à pouvoir aisément tracer les courbes qu'on désire étudier en joignant successivement les points (x, y), avec x les valeurs prises dans la première colonne

et y celles prises dans l'une des colonnes suivantes.

Toutes les figures présentées dans ce document ont été réalisées à l'aide de *Wolfram Mathematica 10*. Un document modèle *Mathematica* a été élaboré pour dessiner l'ensemble des courbes décrites dans les fichiers statistiques. On peut donc avoir accès rapidement à de très nombreuses représentations graphiques, ce qui donne le moyen d'analyser facilement des aspects précis du processus de résolution pour n'importe quelle exécution.

employee.py

Entrée : journal d'activités *bh.log*
Sortie : *emp#.txt*

Ce programme permet de connaître l'ensemble des tâches qui ont été assignées à un employé depuis le début de la résolution. On précise au programme le numéro de l'employé qui nous intéresse ainsi que le numéro du noeud auquel on désire avoir un aperçu des assignations courantes.

Le résultat est un fichier au numéro de l'employé dans lequel sont imprimées toutes ses tâches préassignées et toutes les tâches qui lui ont été fixées par le biais de branchements du début de la résolution jusqu'au noeud précisé. De cette manière, on peut connaître à un moment précis du processus de résolution les disponibilités restantes de l'employé et avoir une idée claire de ce à quoi ressemble son horaire afin d'analyser l'efficacité des branchements effectués.

Il est aussi possible d'écrire $LAST$ au lieu d'un numéro de noeud pour avoir un aperçu de l'horaire mensuel final de l'employé.

Avant de créer ce programme, quelques ajouts ont été faits au code du logiciel pour obtenir plusieurs informations manquantes. Ainsi, on imprime maintenant dans le journal d'activités les tâches préassignées de chaque employé avant que commence la résolution. Aussi, les méthodes de branchement utilisées initialement et celles qui seront potentiellement utilisées dans la suite du projet affichent à présent le détail de toutes les tâches qu'elles permettent de fixer.

costs_stats.py

Entrées : journal d'activités *bh.log*, compte-rendu *SppMpProb.txt*
Sortie : affichage à l'écran

Ce programme fournit un aperçu rapide de la meilleure solution entière trouvée. On y retrouve la contribution des trois sources de coût détaillées dans l'introduction ainsi que des

statistiques sur l'équité et sur une pénalité de qualité qu'on étudiera particulièrement dans la suite de ce document, DOinSurplus.

Plus précisément, on retrouve pour l'équité la valeur moyenne de la pénalité de balancement d'un employé. Pour la pénalité de qualité, on retrouve son nombre total d'occurrences, le nombre moyen d'occurrences par employé ainsi que le maximum d'occurrences pour un employé.

3.2 Branchements

3.2.1 Fonctionnement de la stratégie actuelle

Plusieurs méthodes de branchement sont implémentées dans le logiciel. Quelques-unes ont été développées chez AD OPT au fil des ans afin de mieux répondre aux besoins spécifiques des différentes applications.

Les méthodes compétitionnent les unes contre les autres à la fin de chaque noeud visité dans l'arbre de branchement qui n'est pas élagué. Pour ce faire, chaque méthode mth se voit attribuer un score initial $s_{init}^{mth} \in [0, 1]$ selon les valeurs prises par les différentes variables du problème. Ensuite, chacun de ces scores est reporté sur un nouvel intervalle à l'aide de paramètres définis par l'utilisateur pour chaque méthode, $ScoreMin^{mth}$ et $ScoreMax^{mth}$, qui sont respectivement les bornes inférieure et supérieure du nouvel intervalle. On obtient donc pour une méthode mth son score final :

$$s_{fin}^{mth} = s_{init}^{mth} \times (ScoreMax^{mth} - ScoreMin^{mth}) + ScoreMin^{mth} \tag{3.1}$$

Par exemple, l'utilisateur peut choisir que $ScoreMin^{mth} = 20$ et $ScoreMax^{mth} = 30$. Ainsi, l'intervalle reporté pour la méthode mth est $[20, 30]$. Un score initial s_{init}^{mth} de 0.5 donnera donc un score final s_{fin}^{mth} de 25. De cette façon, on peut favoriser certaines méthodes en établissant un ordre de préférence. Celle avec le meilleur score final sera finalement choisie.

Pour le processus d'optimisation étudié, c'est Tsplit qui est l'unique méthode de branchement utilisée. Plusieurs autres sont en compétition avec elle, mais la borne inférieure très grande du score final de Tsplit ainsi que son utilisation toujours possible empêchent toute autre méthode d'être considérée. Tsplit signifie *task splitting*, donc « division des tâches » en français. Son but est d'imposer des tâches à des employés précis en modifiant légèrement les sous-problèmes. On donne un score $s_{task,emp}$ à l'imposition de la tâche $task$ à l'employé emp en sommant les valeurs prises dans le problème maître par les variables associées à l'employé qui couvrent cette tâche.

Les 4 paramètres énumérés ci-dessous contrôlent la sélection des affectations tâche/employé qui seront imposées par Tsplit. La valeur actuelle de chacun de ces paramètres est indiquée entre parenthèses :

- **TsplitSelectMax** (100) : Détermine le nombre maximal d'affectations imposées.
- **TsplitSelectThreshold** (0.51) : Détermine le seuil de sélection des affectations. Celles de score $s_{task,emp}$ plus petit que ce seuil ne sont pas considérées.
- **TsplitSelectComplementSumMax** (40.0) : Le complément du score d'une affectation est $1 - s_{task,emp}$. Ce paramètre détermine la somme maximale des compléments des affectations retenues.
- **TsplitScoreAvg** (0) : Détermine si le score initial de Tsplit s_{init}^{Tsplit} est la moyenne des scores $s_{task,emp}$ des affectations finalement retenues (1) ou bien le score minimal parmi ces affectations (0).

Tout d'abord, Tsplit élimine toutes les affectations tâche/employé telles que $s_{task,emp} < 0.51$, donc en dessous du seuil de sélection. Dans le cas où aucune affectation n'a un score $s_{task,emp} \geq 0.51$, Tsplit obtient un score final $s_{fin}^{Tsplit} = -1$ et une autre méthode de branchement sera utilisée. Ensuite, on élimine les affectations avec les scores $s_{task,emp}$ les plus faibles pour ne conserver que les 100 meilleures. Si on a déjà 100 affectations ou moins, cette étape est ignorée. Finalement, les affectations avec les scores $s_{task,emp}$ les plus faibles sont éliminées si nécessaire de façon à ce que la somme des compléments $1 - s_{task,emp}$ des affectations restantes soit inférieure ou égale à 40. On donne comme score initial s_{init}^{Tsplit} à Tsplit le score $s_{task,emp}$ le plus petit parmi les affectations finalement sélectionnées pour être imposées.

Trois noeuds fils sont créés. Le premier se voit imposer toutes les affectations sélectionnées indiquées ci-haut. On n'impose pour le deuxième qu'une seule affectation, soit celle avec le meilleur score $s_{task,emp}$. Le troisième noeud se voit interdire cette même affectation. Le parcours en profondeur de l'arbre de branchement privilégie le premier noeud créé.

3.2.2 Éléments problématiques

Dans la phase de balancement, la solution de la relaxation linéaire au noeud 0 pour le mois d'avril 2015 contient un peu plus de 5000 variables fractionnaires, donc des variables avec un flot $\in]0, 1[$. Considérant que la plupart des noeuds prennent du temps à être résolus, il faut donc être agressif dans la stratégie de branchement de manière à pouvoir se rendre assez rapidement vers une solution entière. C'est ce qui est tenté avec la recherche en profondeur combinée à Tsplit. On se dirige toujours vers le noeud fils où 100 tâches ou presque viennent d'être fixées avec un seuil de sélection très bas (0.51), donc peu restrictif. Ainsi, il est espéré que la solution du problème relaxé soit toujours un peu moins fractionnaire au fur et à mesure

que l'on descend dans l'arbre.

Le point faible de cette méthode est qu'elle rend le problème extrêmement complexe. Plus on est profond dans l'arbre de branchement, plus le nombre de tâches imposées aux employés est important. Les rotations du client ont la caractéristique d'être très courtes. Elles consistent en grande majorité d'un aller-retour qui prend moins d'une demi-journée à faire. Les plus longues rotations sont d'une journée et demi tout au plus. Ainsi, à un stade avancé de la résolution, on se retrouve avec énormément de courtes tâches fixées et seulement de petits intervalles épars de disponibilité chez les employés. Le solveur doit essayer d'y assigner les tâches non imposées par Tsplit tout en considérant les cibles de temps de travail de chaque employé et les pénalités sur la qualité.

Sur la figure 3.1, on peut voir les différents types de tâches qui ont été imposés par Tsplit des noeuds 0 à 200 pour 6 employés pris au hasard. Les rotations (RT), les *stand-by* (SB), les jours de congé (CG) et les tâches au sol (GND) sont fixés et se retrouvent aussi dans la solution entière finale. Les blocs vides en gris sont les disponibilités restantes.

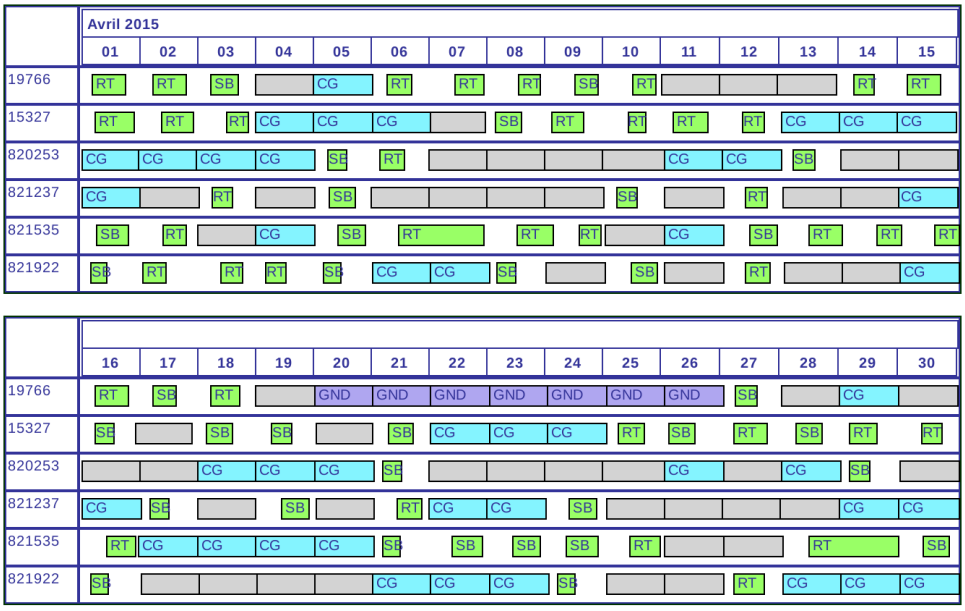

Figure 3.1 Éléments fixés par Tsplit et disponibilités restantes de 6 employés au noeud 200

Cette complexification du problème se traduit par des difficultés à obtenir une solution faisable pour la relaxation linéaire lors de la visite d'un nouveau noeud de l'arbre. Comme mentionné dans l'introduction, 5 configurations de paramètres sont utilisées dans un ordre précis à chaque noeud pour la résolution des sous-problèmes : les deux premières pour se rendre à une solution faisable et les trois suivantes pour l'améliorer. Normalement, beaucoup de temps est passé sur les configurations HA, HB et HBK puisque le solveur travaille

à améliorer la solution en considérant à chaque fois les branchements nouvellement effectués au noeud précédent. Mais à partir d'un certain moment, la majeure partie du temps passé sur un noeud est dédiée aux configurations de faisabilité, donc à essayer de se rendre à une solution faisable.

Comme on peut voir sur la figure 3.2, on passe initialement beaucoup de temps sur les configurations HA et HB. Vers le noeud 200, le temps cumulatif d'utilisation de la configuration HBKFEAS grimpe en flèche contrairement à celui des configurations d'amélioration qui devient de moins en moins important. HBKFEAS est la deuxième configuration de faisabilité, celle utilisée tant et aussi longtemps qu'on a pas une solution faisable.

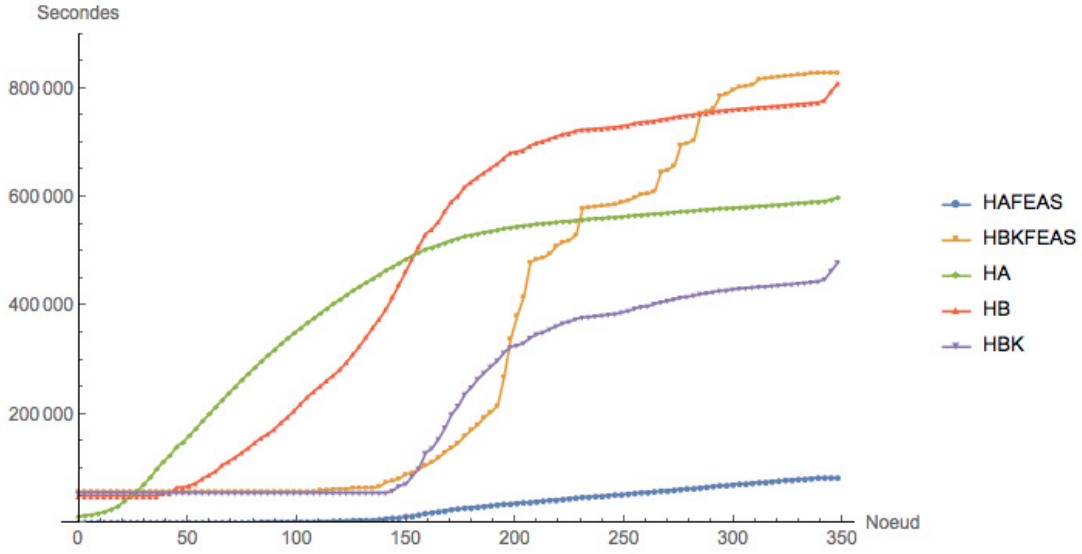

Figure 3.2 Temps cumulatif d'utilisation des 5 configurations de paramètres

À ce même noeud, Tsplit a déjà fixé un peu plus de 18000 tâches et congés à des employés. Comme le présente la figure 3.3, le nombre cumulatif de tâches fixées progresse alors nettement moins vite.

Les petites pointes de la figure 3.3 s'expliquent elles aussi par le problème de faisabilité. À partir du noeud 200, alors qu'on a encore environ 1000 variables fractionnaires, on se bute presque toujours à une relaxation linéaire infaisable et ce, malgré énormément de temps passé sur les configurations de faisabilité. Ainsi, à un branchement, 100 tâches sont fixées au premier noeud fils. On résout ensuite ce noeud pour en arriver presqu'à chaque fois à une relaxation infaisable. On remonte donc au noeud parent pour descendre vers le deuxième noeud fils, soit celui où seule l'affectation de meilleur score $s_{task,emp}$ est imposée. Ce scénario se répète fréquemment, ce qui fait qu'on avance à coup d'une seule fixation au final pour deux noeuds visités.

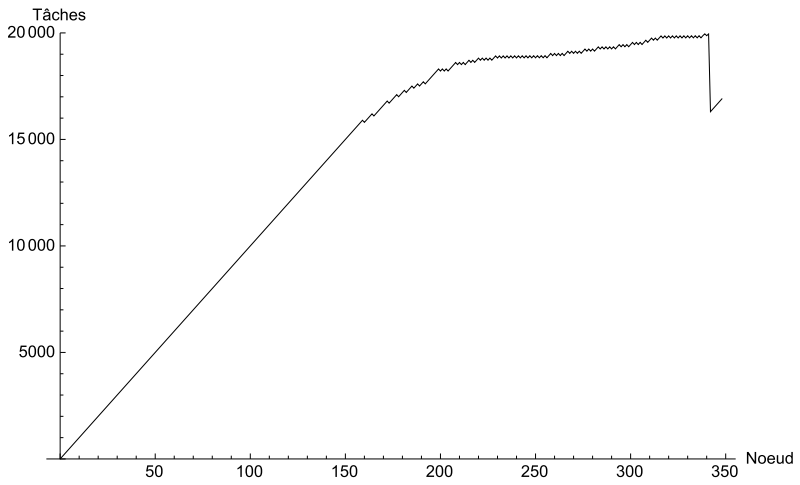

Figure 3.3 Nombre cumulatif de tâches fixées avec Tsplit

Un essai avec les données du même mois, mais en éliminant le 2/3 des tâches et des employés a aussi montré que le problème devient extrêmement difficile même s'il est largement diminué. Comme le montre la figure 3.4, le temps passé à chaque noeud visité est assez constant jusqu'au moment où beaucoup de tâches ont été fixées. On prend alors de plus en plus de temps à chaque nouveau noeud visité avant de se rendre enfin à une solution faisable ou de conclure que la relaxation n'a pas de solution. Avec un problème plus gros, ce temps qui s'étire prend des proportions beaucoup plus importantes.

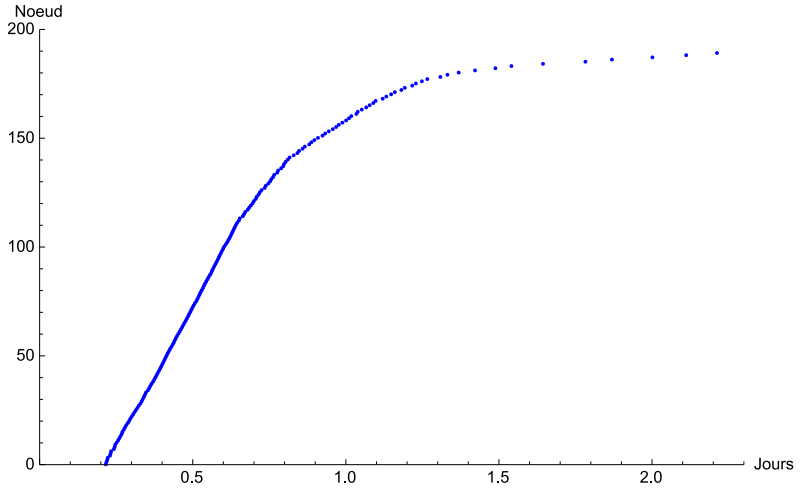

Figure 3.4 Résolution des noeuds dans le temps avec le problème réduit de 2/3

3.2.3 Direction de recherche

Il sera nécessaire d'établir une nouvelle stratégie de branchement. Celle-ci devra elle aussi être agressive étant donné le nombre excessivement important de variables fractionnaires au noeud 0, sans pour autant complexifier le problème comme le fait Tsplit. Les différentes méthodes implémentées devront donc être étudiées et celles qui seront choisies pourront être modifiées de façon à les adapter aux caractéristiques du problème actuel.

3.3 Pénalité de qualité DOinSurplus

3.3.1 Fonctionnement de la pénalité

La pénalité DOinSurplus, pour « jours de congé en surplus », est utilisée dans le but d'éviter de donner des jours de congé en trop. Avant de commencer la phase 2, on évalue pour chaque employé le nombre de jours de congé qu'il devrait avoir au minimum selon ses disponibilités et les règles de travail de son poste. On vise à respecter exactement ce minimum et les journées de congé en surplus sont les occurrences de congé au-delà de cette valeur. Le coût unitaire de DOinSurplus est l'un des plus grands parmi l'ensemble des pénalités de qualité, soit 1000. C'est donc dire qu'on accorde une grande importance à ne pas donner des congés plus que nécessaire.

Tel qu'expliqué dans l'introduction, le nombre d'occurrences obtenues de cette pénalité de qualité dans la phase 2 est transmis à la phase 3 comme un seuil permis avant de commencer à pénaliser au coût de 1000. Une limite supérieure stricte est également fixée dont la valeur est le seuil + 20%. Ainsi, 500 occurrences obtenues à la phase 2 fera en sorte qu'à la phase 3, 525 occurrences coûteront 25000 tandis que 500 occurrences ou moins n'engendreront aucun coût. La limite supérieure stricte serait de 600.

Lors de la génération de colonnes, on recherche dans un réseau le plus court chemin d'un noeud source à un noeud puits. Pour le problème qu'on traite, les colonnes générées sont des horaires mensuels et un ensemble R de ressources permet de modéliser certaines contraintes. Trois sous-ensembles disjoints partitionnent R selon le rôle joué dans la dominance, soient R_{ine}, R_{ega} et R_{auc}. R_{ine} regroupe les ressources comparées avec une inégalité, R_{ega} celles comparées avec une égalité et R_{auc} celles qui ont comme unique tâche de nous informer sur certaines données des chemins.

Pour un noeud i du réseau, on associe à chaque chemin partiel k une étiquette $\boldsymbol{E_i^k}$ définie par le coût C_i^k du chemin et 3 vecteurs représentant la quantité accumulée de chaque ressource au noeud i, un pour chacun des 3 sous-ensembles de R :

$$\boldsymbol{E_i^k} = (C_i^k, \boldsymbol{T_{i,ine}^k}, \boldsymbol{T_{i,ega}^k}, \boldsymbol{T_{i,auc}^k}) \tag{3.2}$$

Les étiquettes sont comparées afin d'éliminer certains chemins partiels, ce qui permet d'éviter de considérer l'ensemble des possibilités et d'accélérer le processus. En comparant deux étiquettes $\boldsymbol{E_i^u}$ et $\boldsymbol{E_i^v}$ au noeud i, on peut éliminer le chemin partiel v si les 3 critères suivants sont respectés :

1. $C_i^u \leq C_i^v$
2. $T_{i,ine}^{u,r} \leq T_{i,ine}^{v,r}\ \forall r \in R_{ine}$
3. $T_{i,ega}^{u,r} = T_{i,ega}^{v,r}\ \forall r \in R_{ega}$

Le choix du rôle d'une ressource dans la dominance ainsi que de possibles changements d'échelle pour les valeurs de certaines ressources font en sorte que cette construction de plus courts chemins est heuristique. On peut ainsi s'assurer d'un bon compromis entre le temps de génération et la qualité des colonnes produites.

La ressource utilisée pour compter le nombre de jours de congé est DO COUNT. Elle est comparée à égalité, ce qui fait en sorte qu'on peut comparer deux étiquettes à des fins de dominance seulement si leur nombre cumulatif de journées de congé est le même. La valeur de cette ressource est discrète et varie généralement entre 0 et 20.

La pénalisation des jours excédentaires de congé est globale, c'est-à-dire qu'elle dépend de la somme totale des occurrences de chaque employé. Elle est donc considérée au niveau du problème maître et non directement lors de la génération de colonnes.

3.3.2 Éléments problématiques

Un très grand nombre d'occurrences

Le seuil transmis par la phase 2 est de 1488. Ceci nous indique donc avant même le début de la phase finale que le nombre d'occurrences de la pénalité sera extrêmement élevé. En effet, tel qu'expliqué précédemment dans l'introduction, ce seuil est le nombre de jours de congé excédentaires obtenus dans la relaxation linéaire du problème ne considérant pas l'équité. En conséquence, rajouter les contraintes d'intégrité et d'équité risque d'augmenter inévitablement le nombre d'occurrences au-delà de ce seuil.

C'est en effet le comportement remarqué sur la figure 3.5. Jamais le solveur n'est en mesure de descendre le total de jours de congé excédentaires en deça de cette limite au cours de la résolution. Après les 50 premiers noeuds, une certaine stabilité s'installe puisque le nombre

d'occurrences pénalisées est de 2 pendant un long moment. Toutefois, à partir du noeud 200, cette stabilité fait place à un oscillement considérable. Le solveur peine à maintenir le nombre d'occurrences, ce dernier variant entre 4 et 55 au dessus du seuil.

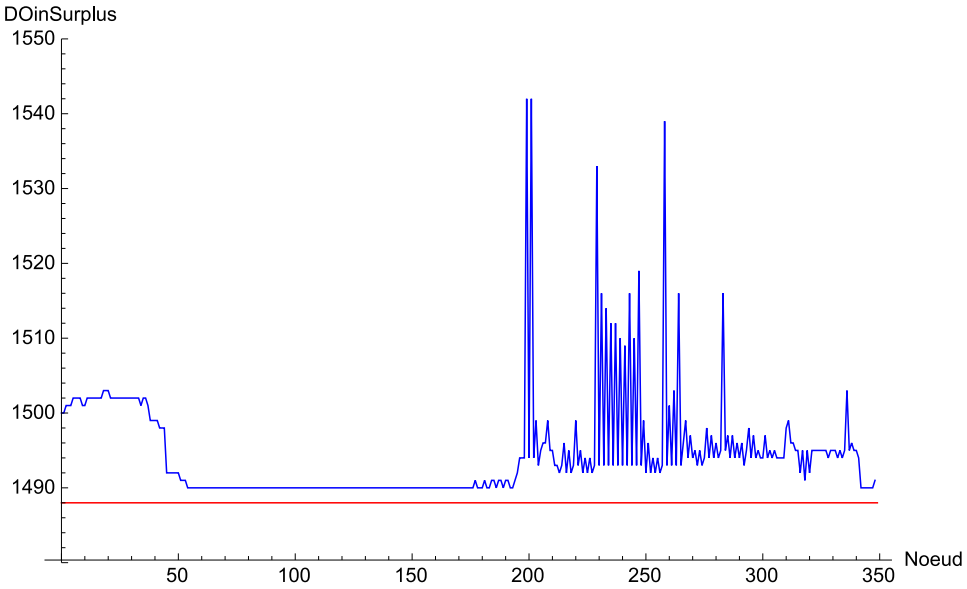

Figure 3.5 Nombre de jours de congé excédentaires à la fin de chaque noeud

Cette oscillation est étroitement liée à la complexification du problème occasionnée par la méthode de branchement Tsplit. En effet, l'instabilité commence au moment même où l'on rencontre les premières difficultés de faisabilité liées aux branchements effectués depuis le début de la résolution, soit au noeud 200. Comme les employés se retrouvent avec de petits intervalles épars de disponibilité où Tsplit n'a encore fixé aucune tâche, le solveur peine à remplir ces espaces par des rotations tout en gérant la contrainte d'équité et le nombre de congés excédentaires.

Un examen des coûts révèle que les pénalités de balancement varient à l'opposé de la pénalité DOinSurplus. Tout d'abord, le solveur assigne l'ensemble des rotations restantes dans les quelques disponibilités des employés en ajoutant le moins possible de jours de congé au-delà du minimum requis de chaque employé. On obtient une solution qui a un nombre de jours de congé en surplus baissé au plus près du seuil, mais avec des pénalités de balancement élevées. Ensuite, en réaction à ces pénalités de balancement coûteuses, le solveur se concentre cette fois-ci à rendre les horaires plus équitables en assignant les rotations restantes de manière à ce que les employés obtiennent un temps de travail très proche de celui visé. Il se soucie donc moins de ne pas dépasser le nombre minimal de congés à donner. On obtient dans ce cas une solution de bien meilleure équité, mais avec un nombre beaucoup plus grand de congés

excédentaires. Le comportement se répète encore, d'où les oscillations dans le nombre de congés en surplus.

En comparant le coût de cette pénalité au coût total de l'ensemble des pénalités de qualité tout au long du processus (fig. 3.6), on remarque la très grande similarité des 2 courbes. C'est donc essentiellement DOinSurplus qui dicte le coût relié à la qualité des horaires. Seuls les noeuds 315 à 350 font exception puisque c'est BlankDay pénalisant les journées de travail vides qui fait alors fluctuer le coût total des pénalités de qualité.

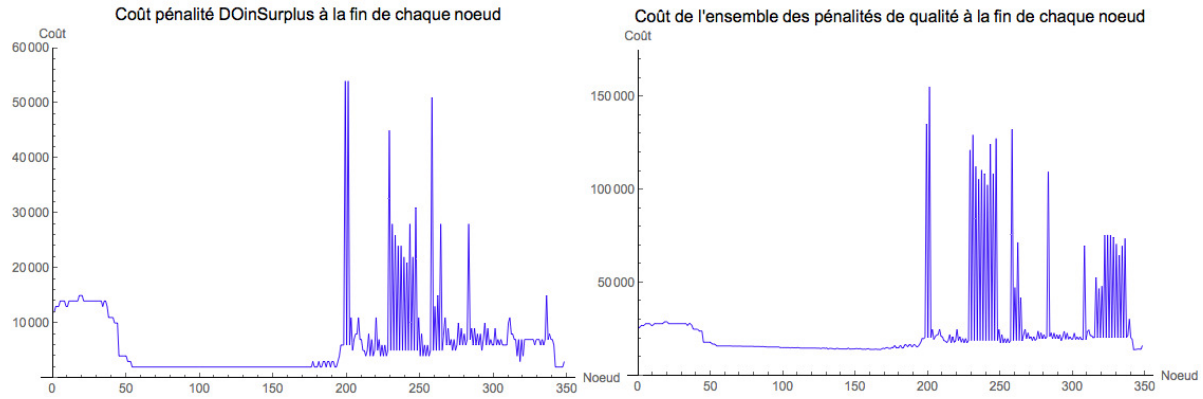

Figure 3.6 Coût de la pénalité DOinSurplus vs Coût de l'ensemble des pénalités

La distribution finale des jours de congé en surplus présentée à la figure 3.7 nous confirme que le nombre total d'occurrences dans la solution finale est excessivement grand. Seulement 257 employés, soit un peu plus du quart de l'effectif, ont obtenu le nombre minimal de jours de congé. Les autres se retrouvent avec un nombre de jours de congé excédentaires variant entre 1 et 9. Qu'il y ait autant d'employés avec 2 occurrences ou plus nous indique que les minimums de jours de congé calculés dans la phase précédente sont possiblement sous-estimés.

D'ailleurs, l'employé avec 9 occurrences a toujours obtenu ce nombre de jours de congé en trop dans toutes les exécutions lancées, même avec divers changements de paramètres. Cela nous démontre donc qu'une cible plus réaliste pour celui-ci serait son minimum actuel additionné de 9 unités. Un tel écart n'est pas acceptable et contribue inutilement à ajouter des coûts au problème.

Génération de colonnes très lente

Le temps de résolution pour le problème maître et les sous-problèmes à chaque noeud est montré à la figure 3.8. On constate que beaucoup de temps est passé sur la résolution du problème maître durant la seconde moitié du processus d'optimisation. Ceci est expliqué par

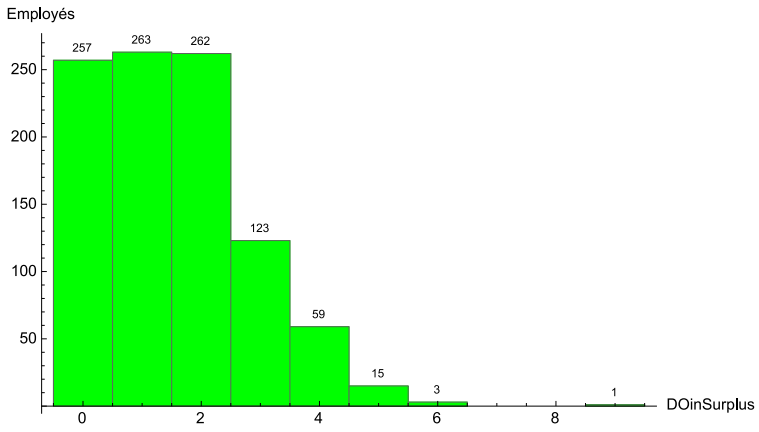

Figure 3.7 Distribution finale du nombre de jours de congé excédentaires par employé

les problèmes de faisabilité occasionnés par Tsplit. À partir du noeud 200, la majeure partie du temps de résolution est passée à essayer d'obtenir une solution faisable. Le problème maître accumule alors de plus en plus de variables et il prend conséquemment plus de temps à être résolu par l'algorithme de points intérieurs.

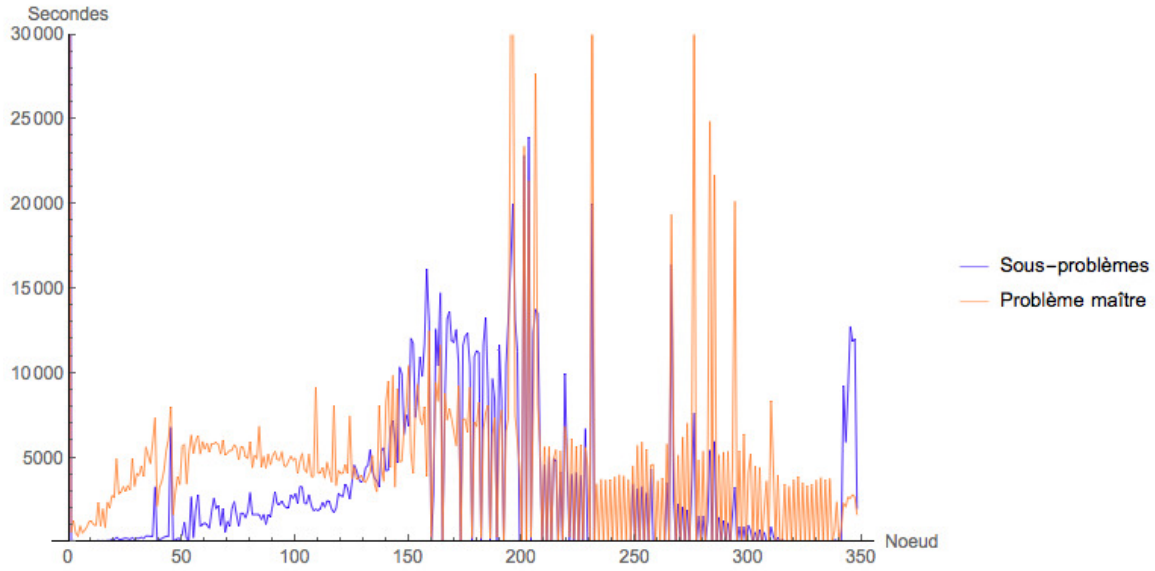

Figure 3.8 Temps de résolution pour le problème maître et les sous-problèmes à chaque noeud

La résolution des sous-problèmes prend elle aussi anormalement beaucoup de temps tout au long du processus. Il y a énormément d'itérations de génération de colonnes à chaque noeud comme le témoigne la figure 3.9. La dominance n'est donc pas efficace, trop d'étiquettes étant conservées lors de la recherche de plus courts chemins dans les sous-problèmes afin de générer de nouvelles colonnes. Les ressources à examiner sont celles du sous-ensemble R_{ega}

puisqu'elles limitent énormément le nombre d'éliminations possibles. En effet, l'un des trois critères pour éliminer une étiquette exige lors de la comparaison avec une autre étiquette que toutes les ressources de R_{ega} soient de valeurs égales. Si les ressources de ce sous-ensemble sont susceptibles de prendre des valeurs différentes d'une étiquette à l'autre, il devient alors très difficile de respecter ce critère.

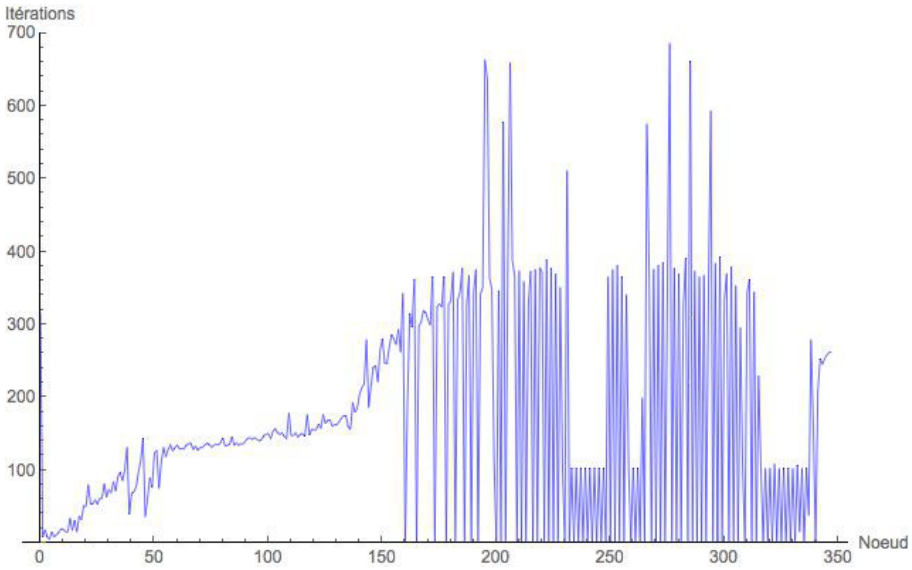

Figure 3.9 Nombre d'itérations à chaque noeud

En examinant les 4 ressources de R_{ega}, seule DO COUNT, qui compte les jours de congé, semble causer problème. Les autres sont peu utilisées ou bien prennent 3 valeurs différentes au maximum. Tel que mentionné précédemment, DO COUNT prend des valeurs entre 0 et 20. Sa présence dans R_{ega} suppose que le nombre de jours de congé augmente similairement d'un horaire à l'autre au fur et à mesure qu'on avance dans le mois. Le nombre de jours de congé serait alors très semblable pour tous les chemins partiels. Ce n'est pas ce qui semble se produire puisque la génération est si longue. Quelques contraintes encadrent l'attribution des jours de congé, mais pas assez pour qu'ils soient disposés de la même manière dans les possibilités d'horaire d'un employé.

On pourrait être porté à croire qu'une comparaison à égalité est nécessaire pour cette ressource étant donné qu'une comparaison avec une inégalité avantagerait les chemins partiels avec plus de congés ou bien ceux avec moins de congés selon le sens de l'inégalité. On éliminerait alors les possibilités d'avoir moins de congés ou bien beaucoup de congés en début de bloc. Cette éventualité devra être examinée s'il est choisi de faire passer DO COUNT de R_{ega} à R_{ine}.

3.3.3 Direction de recherche

Le fonctionnement de la pénalité DOinSurplus serait entièrement à revoir. Plus précisément, il faudrait réviser son coût ou l'utilisation du seuil de manière à éviter une oscillation aussi importante dans le nombre d'occurrences. Idéalement, le nombre de jours de congé excédentaires demeurerait tout de même identique ou n'augmenterait que très peu. Aussi, on vise à retravailler la stratégie de dominance lors de la génération de colonnes en changeant la façon de gérer la ressource associée à la pénalité, soit DO COUNT. Finalement, il serait important de retravailler le procédé qui détermine le nombre minimal de journées de congé de chaque employé afin qu'on puisse obtenir des estimations plus justes.

CHAPITRE 4 AMÉLIORATIONS PROPOSÉES

Les améliorations qui ont été apportées tout au long du projet seront présentées ici en ordre chronologique. Ce n'est pas l'ordre logique qui aurait été préférable face au diagnostic établi au chapitre 3, mais plutôt un ordre qui est venu naturellement selon l'apprentissage progressif du fonctionnement du code à la base du produit d'optimisation étudié.

La solution de référence est pour le mois d'avril 2015. C'est le seul jeu de données pour lequel on a laissé le logiciel tourner jusqu'au bout. Les autres mois n'ont donné aucune solution après une semaine d'exécution et le logiciel a été stoppé manuellement à chaque fois. On commencera donc à présenter les résultats pour ces mois à partir de la première amélioration.

Les solutions seront présentées dans un tableau comme celui ci-dessous (4.1). On a dans l'ordre le noeud où la solution a été trouvée (N), le temps CPU de résolution pour obtenir cette solution (Temps), la valeur de la solution (Val), la valeur de la relaxation linéaire (Relax), le nombre de jours de congé excédentaires (DIS), le coût des pénalités de qualité (Qual), le coût des pénalités de non-couverture (N-C), le coût des pénalités de balancement (Bal) et, finalement, le score. Le coût des pénalités de qualité est séparé en deux colonnes. La première est le coût de la pénalité DOinSurplus et la deuxième est le coût pour le reste des pénalités de qualité. Le score est la valeur de la solution moins le coût de la pénalité DOinSurplus.

Tableau 4.1 Solution de référence

Avril 2015

N	**Temps**	**Val**	**Relax**	**DIS**	**Qual**		**N-C**	**Bal**	**Score**
341	32j 01h 24m	76121	63906	1494	6000	13290	120	56711	70121

Cette façon de faire permet de séparer l'analyse des coûts de la solution et l'analyse de la pénalité DOinSurplus. Comme la façon de pénaliser a été changée, il devenait difficile de comparer les nouvelles solutions avec les anciennes. De cette manière, on a la possibilité de comparer de façon indépendante les coûts des solutions par l'intermédiaire du *score* et les occurrences de la pénalité DOinSurplus avec la colonne *DIS*.

Dans chaque tableau de résultats, on présentera les deux premières solutions entières trouvées s'il y en a au moins deux. On rapportera également surlignées en gris les solutions entières trouvées avant l'amélioration présentée, ce qui permettra d'expliciter l'effet de cette dernière en facilitant la comparaison.

Pour conclure ce chapitre, on examinera si l'objectif du projet a été atteint en comparant la solution de référence avec la solution obtenue en bénéficiant de l'ensemble des améliorations proposées.

4.1 Modification des règles de dominance et ajustement de paramètres

4.1.1 Dominance modifiée

On a vu précédemment que la ressource DO COUNT $\in R_{ega}$ qui compte les jours de congé ralentit énormémement le processus de génération de colonnes. Pour qu'une étiquette soit éliminée, un des critères exige lors de la comparaison avec une autre étiquette que leur nombre de journées de congé soit identique. Ceci limite grandement le nombre d'étiquettes éliminées par dominance. Il faudrait idéalement trouver un moyen de favoriser les éliminations en faisant passer DO COUNT de R_{ega} à R_{ine} ou en effectuant un changement d'échelle permettant de regrouper des valeurs proches, par exemple en prenant l'entier inférieur de la division par 2.

Le changement d'échelle n'est pas une alternative assez agressive qui pourrait permettre un gain important en temps de résolution. Regrouper certaines valeurs n'augmenterait que légèrement le nombre d'éliminations possibles. Il a donc été décidé de faire passer la ressource DO COUNT de R_{ega} à R_{ine} et de modifier la manière dont est mise à jour sa valeur à chaque noeud. La nouvelle fonction de mise à jour est la suivante :

$$DoCountMAJ(actuel,\ cible) = -1 \times MIN\{actuel,\ cible\} \tag{4.1}$$

Lorsqu'un chemin partiel est prolongé et qu'on arrive à un nouveau noeud dans le réseau, la ressource DO COUNT prend comme valeur le négatif du minimum entre le nombre de jours de congé donnés du noeud source au nouveau noeud le long du chemin partiel étudié et le nombre ciblé, cette cible étant le minimum de jours de congé qui doit être donné à l'employé. La ressource commence donc à 0 et diminue à coup d'un congé jusqu'à atteindre le négatif de la cible.

Cette façon de faire favorise les étiquettes avec le plus grand nombre de jours de congé et, finalement, celles dont la cible est déjà atteinte. En effet, on éliminera un chemin partiel si, en le comparant à un autre chemin, son coût est plus grand, chacune de ses ressources dans R_{ine} dont DO COUNT sont de valeurs plus grandes ou égales et chacune de ses ressources dans R_{ega} sont de mêmes valeurs. La seule différence avec la dominance utilisée précédemment est qu'il est maintenant possible lorsqu'on compare deux étiquettes d'en éliminer une même si elles ne partagent pas le même nombre de jours de congé. Il n'y a plus aucune restriction

d'égalité quant à cette ressource et les valeurs plus grandes de congés dominent les plus faibles puisqu'on travaille avec des valeurs négatives.

Il faudra toutefois vérifier si cette nouvelle façon de gérer la ressource DO COUNT aura un impact sur la répartition des jours de congé dans les horaires construits. Comme les chemins partiels avec un plus grand nombre de jours de congé sont favorisés, il serait possible que les horaires aient tendance à avoir beaucoup de leurs jours de congé en début de mois.

4.1.2 Paramètres ajustés

Beaucoup de temps est passé sur chaque noeud visité de l'arbre lors de la résolution, souvent sans amélioration significative du coût de la solution courante. Il peut même se passer 4-5 itérations sans que la solution ne change. Puisqu'à chaque itération de nouvelles colonnes sont générées, le problème maître se retrouve avec de plus en plus de variables et prend toujours un peu plus de temps à être résolu. Les paramètres sont donc à revoir de façon à ne pas traîner inutilement en longueur à un noeud pour des gains très maigres sur la solution.

Voici donc les paramètres dont la valeur a été modifiée :

1. **SppModNetMaxFail :** Ce paramètre stoppe une itération lorsque le nombre précisé de sous-problèmes échouent à générer des colonnes de coût réduit négatif. Il devient actif dès le moment où un sous-problème a été en mesure d'en générer au moins une.
   Modification : $10 \rightarrow NULL$
   Explication : Pour beaucoup d'itérations, on se retrouvait avec un succès au bout d'une dizaine de sous-problèmes, ce qui activait le paramètre et arrêtait donc momentanément l'itération courante. Dans ces nombreux cas, très peu de nouvelles colonnes étaient générées, soit environ 5 au maximum. Ajouter si peu de colonnes au problème maître améliorait rarement ou très peu la solution et on se retrouvait alors avec un problème presque identique pour l'itération suivante. C'est en partie pourquoi on pouvait avoir énormément d'itérations à un même noeud sans pourtant améliorer de beaucoup la solution. Annuler ce paramètre permettra de donner une chance à tous les sous-problèmes de générer des colonnes, même si les premiers échouent. On évitera ainsi de stagner pendant plusieurs itérations successives en se laissant l'opportunité d'améliorer significativement la solution d'une itération à l'autre.

2. **SppModItrMaxFeas :** Nombre maximal d'itérations qu'on peut passer sur une configuration d'amélioration.
   Modification : $100 \rightarrow 50$ (configurations HA et HB), $60 \rightarrow 20$ (configuration HBK)

Explication : Il arrivait pour beaucoup de noeuds qu'on passe par les trois configurations d'amélioration HA, HB et HBK, ce qui fait un total de $100 + 100 + 60$ itérations. Passer par autant d'itérations s'avérait souvent inutile puisqu'au final la solution ne s'améliorait que très peu. Il aurait été préférable dans ces cas de conclure ce noeud tout simplement et de procéder au branchement. Le nombre maximal d'itérations permis sur une configuration a donc été diminué de moitié ou plus. Le paramètre *SppModNetMaxFail* annulé devrait diminuer le nombre d'itérations nécessaires et pour les autres cas, ces limites permettront de ne pas perdre trop de temps à un même noeud.

3. **SppModLogCostDecMinRatio :** Diminution relative minimale de la solution qui doit s'être produite au cours des 10 dernières itérations. Une plus petite diminution provoque un arrêt de la configuration courante pour passer à la suivante.
   Modification : $0 \rightarrow 0.01$
   Explication : On s'assurera qu'au minimum une diminution relative de 1% de la solution s'est produite sur les 10 dernières itérations, faute de quoi on passera à la prochaine configuration. Dans la même veine que pour la redéfinition des paramètres ci-haut, ceci permettra de ne pas s'attarder sur une configuration si elle ne permet plus d'améliorer efficacement la solution courante.

L'ajustement du paramètre *SppModNetMaxFail* pourrait potentiellement augmenter de beaucoup le temps requis pour résoudre un noeud de l'arbre de branchement. Pour le mois d'avril 2015, on travaille avec 983 employés, ce qui signifie que jusqu'à 983 sous-problèmes peuvent maintenant être résolus à une même itération sans qu'aucun critère d'arrêt ne puisse mettre fin à l'itération si le taux d'échec à générer de nouvelles colonnes est élevé.

On compte sur la modification de la dominance pour accélérer énormément la résolution des sous-problèmes et compenser pour la possibilité qu'on puisse en avoir plus à résoudre. On prévoit aussi que le nombre d'itérations à un même noeud diminuera de beaucoup. En effet, la solution devrait s'améliorer significativement d'une itération à l'autre du fait qu'on donnera la chance à tous les sous-problèmes de générer de nouvelles colonnes et qu'on fournira donc plus de nouvelles variables au problème maître. Si ce n'est pas le cas, les deux autres paramètres nouvellement ajustés assureront de ne pas perdre inutilement trop de temps à ce noeud. Ces deux éléments devraient faire en sorte qu'au final, les noeuds soient résolus plus rapidement. Ce sera à surveiller dans les résultats.

4.1.3 Résultats

Comme le démontrent les résultats du tableau 4.2, les gains en terme de temps de résolution sont énormes pour le mois d'avril 2015. Les changements ont permis de réduire de 27.68 jours le temps avant d'obtenir une première solution entière, ce qui représente plus précisément 86.34% moins de temps. Pour les autres mois, aucune solution entière n'avait initialement été trouvée après une semaine d'exécution avec la première version du logiciel. On est maintenant en mesure d'en obtenir une dans des délais beaucoup plus raisonnables, soit entre 1j 08h et 5j 22h.

Tableau 4.2 Changements dominance et paramètres

Avril 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
341	32j 01h 24m	76121	63906	1494	6000	13290	120	56711	70121
199	4j 09h 05m	73708	14627	1488	0	14170	6760	52778	73708
201	4j 09h 06m	55465	14627	1487	0	3810	6730	44925	55465

Juin 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
215	5j 21h 49m	43762	39950	345	0	160	21430	22172	43762

Juillet 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
206	2j 20h 14m	32064	25877	595	1000	450	23070	7544	31064
221	3j 04h 42m	27600	25808	591	0	50	20870	6680	27600

Août 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
209	5j 01h 44m	68969	28425	3000	2000	2140	1010	63819	66969
256	5j 09h 55m	64814	27942	3000	2000	3620	1010	58184	62814

Septembre 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
194	1j 19h 03m	247339	246552	432	43000	5210	23530	175599	204339
198	1j 19h 11m	245884	245494	432	43000	5200	22540	175144	202884

Octobre 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
197	1j 08h 49m	101340	100158	253	22000	290	18610	60440	79340
206	1j 09h 14m	100191	98677	253	22000	170	18610	59411	78191

Pour ce qui est de la qualité des solutions, on constate une certaine similarité avec un léger avantage pour les nouvelles solutions. En regardant la première solution obtenue pour avril 2015, on peut voir que le nombre d'occurrences de jours de congé excédentaires est très

légèrement diminué et que la pénalité de balancement a baissé de 6.94%. Par contre, quelques *stand-by* n'ont pas été couverts, ce qui occasionne une pénalité de non-couverture plus grande. Quant aux pénalités de qualité autres que DOinSurplus, une petite augmentation de 6.62% s'est produite.

Toutefois, en attendant une minute de plus, on obtient déjà une deuxième solution entière qui, elle, a une pénalité de balancement 20.78% plus petite que la solution de référence et des pénalités de qualité diminuées de 71.33%. On peut donc très bien attendre cette deuxième solution entière et se retrouver avec une qualité et une équité grandement améliorées. Un délai aussi court est probablement attribuable au hasard, mais généralement la deuxième solution entière trouvée dans tous les tests effectués permet de corriger les valeurs anormalement grandes prises par certaines pénalités dans la première solution entière obtenue.

Ce gain majeur en temps de résolution s'explique en grande partie par le changement dans la stratégie de dominance. Pour avril 2015, on a passé en moyenne 618.97 secondes par noeud de branchement sur la résolution des sous-problèmes comparativement à 3489.81 secondes précédemment.

Comme expliqué un peu plus haut, la nouvelle fonction de mise à jour pour la ressource DO COUNT et son passage de R_{ega} à R_{ine} font en sorte que les valeurs plus grandes de congés dominent les plus faibles. Cette façon de faire laisse porter à croire que les jours de congé auront tendance à se retrouver en grande partie en début de mois. En effet, avec cette nouvelle dominance, on conserve les chemins partiels ayant le plus grand nombre de jours de congé ou ayant déjà atteint leur cible. Ainsi, la dominance permettra d'éliminer beaucoup de chemins partiels avec un petit nombre de jours de congé, c'est-à-dire les possibilités d'horaire avec peu de congés donnés depuis le début du mois.

En regardant les résultats, ce n'est pourtant pas la tendance que l'on retrouve dans les horaires obtenus. La nouvelle dominance pousse à donner les journées de congé le plus tôt possible, mais d'un autre côté, dès qu'on a atteint la cible, toutes les occurrences excédentaires sont pénalisées par la pénalité DOinSurplus. Plusieurs règles régissent la répartition des congés à l'intérieur d'un mois. Si le nombre ciblé de jours de congé est entièrement donné en début de bloc, les règles feront en sorte que d'autres congés devront obligatoirement être donnés dans la suite de l'horaire qui seront tous des occurrences de jours de congé excédentaires fortement pénalisées. Ces horaires avec tous les congés au début coûteront alors très chers. Il y a donc un équilibre qui se crée et les congés sont bien répartis.

L'ajustement des paramètres a diminué considérablement le nombre d'itérations à chaque noeud, comme on peut le voir sur la figure 4.1.

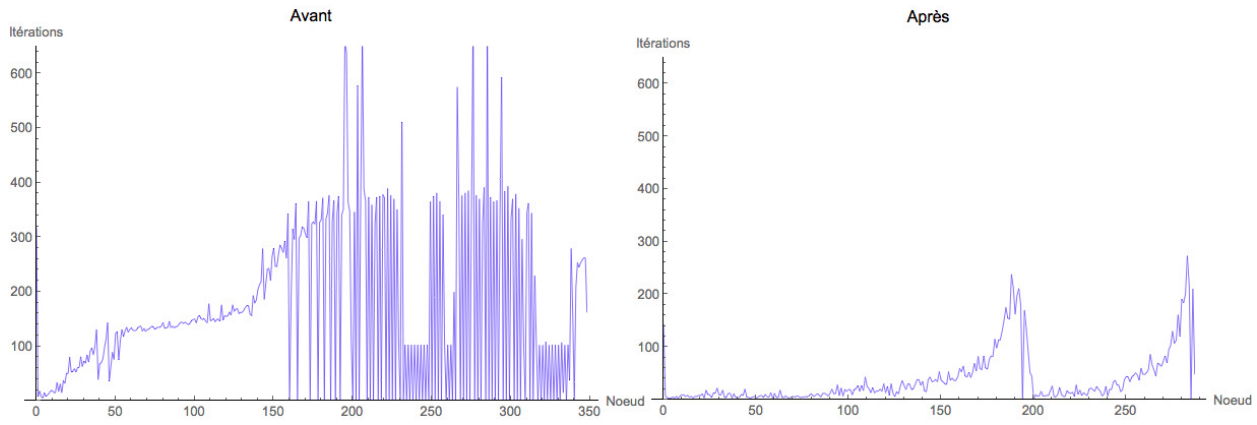

Figure 4.1 Nombre d'itérations à chaque noeud - Avril 2015

Imposer une amélioration relative de 1% et diminuer de la moitié ou plus le nombre maximal d'itérations possibles pour chaque configuration d'amélioration permet donc d'être beaucoup plus efficace dans la résolution en interdisant toute longueur étirant le temps de calcul pour très peu de bénéfices.

On pourrait penser de prime abord que ces nouvelles règles auraient un impact négatif sur la qualité de la solution. En effet, couper dans le nombre d'itérations et exiger une certaine amélioration pourrait retirer la chance au solveur de se rendre aux meilleures solutions possibles. Cette nouvelle sévérité est compensée par le fait que pour chaque itération, on résout maintenant chacun des sous-problèmes tandis qu'on s'arrêtait précédemment dès qu'on totalisait 10 insuccès. On a ainsi beaucoup plus de chances de générer de nouvelles colonnes et d'améliorer la solution. Avec l'ancienne stratégie de dominance, résoudre tous les sous-problèmes aurait été impensable, car cela aurait demandé beaucoup trop de temps. En raison du passage de DO COUNT de R_{ega} à R_{ine}, la génération de colonnes s'est énormément accélérée. C'est pourquoi on peut maintenant se permettre de résoudre tous les sous-problèmes.

En regardant le graphique *après* de la figure 4.1, on voit que le problème de faisabilité provoqué par Tsplit persiste toujours. On trouve des solutions entières aux noeuds 199 et 201. Le nombre d'itérations est de plus en plus élevé en s'approchant de ces noeuds puisque beaucoup de tâches ont alors été fixées et que le problème devient très complexe. On passe en conséquence énormémement de temps sur les configurations de faisabilité. C'est normal étant donné que la stratégie de branchement n'a pas encore été modifiée.

La figure 4.2 nous montre qu'on passe en moyenne plus de temps sur la résolution des sous-problèmes par itération. C'est ce à quoi il fallait s'attendre, car on résout maintenant tous les sous-problèmes. Malgré tout, ce temps moyen par itération n'a pas considérablement

augmenté grâce à la nouvelle stratégie de dominance. De plus, le nombre d'itérations par noeud étant maintenant considérablement plus petit, on s'en retrouve peu affecté au final.

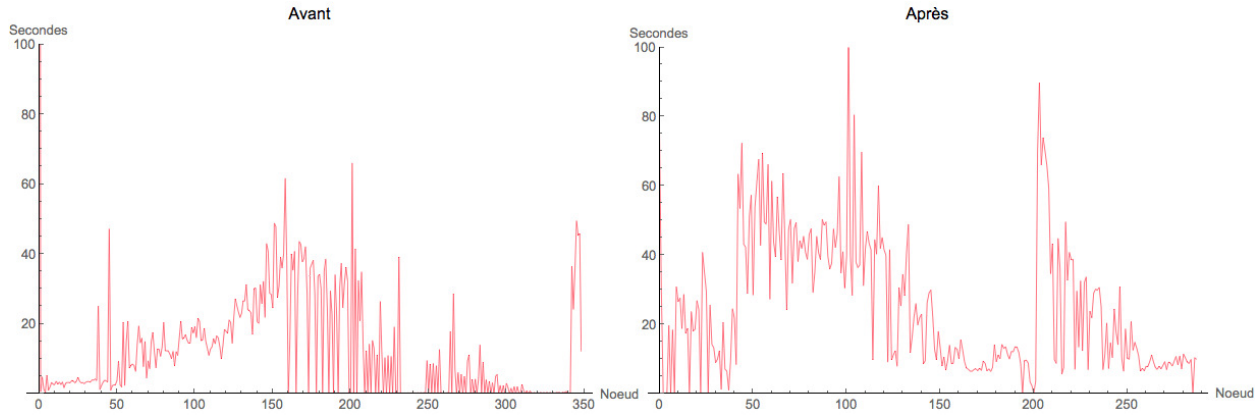

Figure 4.2 Temps moyen passé sur la résolution des sous-problèmes par itération à chaque noeud - Avril 2015

4.2 Pénalisation individuelle de DOinSurplus

4.2.1 Nouvelle façon de pénaliser les jours de congé excédentaires

Auparavant, la pénalité de qualité DOinSurplus était considérée au niveau du problème maître. Dans la phase 2, la relaxation linéaire du problème est résolue sans considération de l'équité. Le nombre de jours de congé excédentaires obtenus dans la solution fractionnaire était alors transmis à la phase 3 comme un nombre d'occurrences tolérées. Seulement les occurrences au-delà de ce seuil étaient pénalisées au coût de 1000 chacune.

On a vu dans le diagnostic établi au chapitre précédent que la complexification du problème occasionnée par Tsplit contribue au comportement oscillatoire de la pénalité. L'utilisation d'un seuil de tolérance avec un si grand nombre d'occurrences pour cette pénalité est elle aussi susceptible d'y contribuer. Il peut se produire de grandes variations dans les coûts qui sont difficiles à gérer pour le problème maître. Par exemple, si le seuil est fixé à 1500, augmenter le nombre d'occurrences de 1490 à 1497 n'engendre aucune variation du coût tandis qu'un passage de 1497 à 1507 occurrences fait soudainement grimper le coût de 7000.

Quand le nombre total d'occurrences de jours de congé excédentaires est inférieur à 1500, les congés ne sont pas pénalisés dans les sous-problèmes et des colonnes avec beaucoup de journées de congé peuvent alors être générées. Le problème maître est ensuite réoptimisé et le coût des autres pénalités diminue tandis que le nombre de jours de congé augmente.

On a donc une pénalisation coûteuse des jours de congé excédentaires puisque le nombre total d'occurrences dépasse alors 1500. En réaction à cette augmentation, les congés sont fortement pénalisés dans les sous-problèmes. On génère alors des colonnes avec moins de jours de congé à n'importe quel prix. On réoptimise le problème maître et le coût des autres pénalités augmente tandis que le nombre de jours de congé diminue. Cette oscillation peut se poursuivre longtemps avant de se stabiliser.

La tâche serait potentiellement simplifiée si cette pénalité était considérée directement lors de la génération des colonnes.

La solution proposée pour faciliter la gestion de cette pénalité est d'abolir le seuil d'occurrences tolérées et de pénaliser tout simplement chaque occurrence. De cette façon, on considère maintenant le coût des jours de congé excédentaires au niveau des sous-problèmes. Les colonnes transmises au problème maître ont alors comme coût la pénalité de balancement pour l'employé concerné plus la pénalité pour chaque jour de congé donné au-delà du minimum visé. Cette technique a pour avantage de permettre une génération de colonnes plus soucieuse du respect du nombre ciblé de jours de congé et évitera ainsi l'oscillation qu'on retrouvait précédemment avec le nombre total d'occurrences.

Les premiers tests de cette nouvelle méthode ont vite montré qu'un nouveau coût unitaire est nécessaire. Pénaliser chaque occurrence au coût de 1000 avec un nombre de jours de congé excédentaires variant entre 1400 et 1500 pour avril 2015 fait en sorte que les solutions obtenues ont une valeur dont plus de 90% est attribuable à la pénalité DOinSurplus. On se retrouve certes avec un nombre d'occurrences diminué, mais en contrepartie le coût total des pénalités de balancement est extrêmement élevé comparativement à ce qui était obtenu précédemment. Le solveur se concentre uniquement à baisser le nombre de jours de congé excédentaires du fait que la valeur de la solution dépend presque exclusivement de cette pénalité. Comme l'un des objectifs secondaires du logiciel après la couverture des rotations est de fournir des horaires équitables, ce ne sont pas des solutions désirables.

Plusieurs tests pour optimiser ce paramètre en utilisant un coût unitaire variant entre 25 et 500 ont permis d'établir qu'un coût de 50 est le meilleur compromis. Une valeur plus grande a tendance à occasionner trop de bruit à l'équité tandis qu'avec une valeur plus basse on obtient généralement beaucoup trop d'occurrences de journées de congé excédentaires. À 50, on observe une meilleure équité dans les solutions en plus d'une légère diminution dans le nombre de jours de congé en surplus par rapport à ce qu'on avait avant d'utiliser la nouvelle façon de pénaliser.

4.2.2 Résultats

L'élément qui ressort le plus des résultats présentés dans le tableau 4.3 est la meilleure équité des horaires. En comparant la première solution obtenue avant l'implémentation de la nouvelle pénalisation proposée avec celle obtenue après, le balancement a diminué en moyenne de 36.86% pour les 5 mois où il y a eu amélioration. Seul le mois de juillet se retrouve avec une augmentation, qui est de 36.20%.

Pour ce qui est du temps de résolution, on remarque que la première solution entière est obtenue beaucoup plus rapidement qu'avant pour les mois d'avril, juin et août avec une diminution moyenne de 29.34%. Ces mois étaient initialement ceux nécessitant le plus de temps avant qu'une toute première solution entière soit obtenue. Pour le mois de juillet, le temps est sensiblement le même qu'avant avec un écart négligeable de 30 minutes.

Les deux derniers mois, septembre et octobre, qui étaient les plus rapides à résoudre ont vu leur temps d'obtention d'une première solution entière substantiellement augmenter. Ce qui est intéressant pour ces deux instances est que le nombre de jours de congé excédentaires a beaucoup augmenté et que l'équité s'est améliorée drastiquement. En effet, pour septembre, le nombre de jours de congé excédentaires a augmenté de 212.04% et la pénalité de balancement a diminué de 31.86%. Pour octobre, le nombre de jours de congé excédentaires a augmenté de 19.37% et la pénalité de balancement a diminué de 83.58%.

Ceci démontre que pour ces deux mois, le seuil d'occurrences allouées transmis par la phase 2 était largement sous-évalué. On peut voir qu'avec l'ancienne méthode de pénalisation, il y avait 43 occurrences au dessus du seuil pour septembre et 22 pour octobre, au coût de 1000 chacune. Le solveur se préoccupait donc très peu de l'équité des horaires afin d'être en mesure de se rapprocher le plus possible d'un seuil inatteignable.

Abolir ce seuil et pénaliser chaque occurrence permet d'obtenir un meilleur équilibre entre équité et congés excédentaires. Pour septembre et octobre, la résolution a pris légèrement plus de temps puisque cette fois-ci le solveur se souciait un peu plus de l'équité. Pour les autres mois, on gagne en temps et en équité et on se retrouve même avec moins de jours de congé excédentaires qu'auparavant.

Concernant les autres sources de coût, les pénalités de qualité autres que DOinSurplus ont un coût total inférieur ou égal pour tous les mois étudiés. Du côté des pénalités de non-couverture, le coût est extrêmement similaire. Quelques premières solutions ont une rotation non couverte (juin et septembre), mais la situation se corrige toujours une fois la deuxième solution obtenue.

La figure 4.3 nous montre que le comportement de la pénalité se rapproche maintenant un peu

Tableau 4.3 Changement pénalisation DOinSurplus

Avril 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
199	4j 09h 05m	73708	14627	1488	0	14170	6760	52778	73708
201	4j 09h 06m	55465	14627	1487	0	3810	6730	44925	55465
200	2j 11h 03m	117700	87586	1324	66200	2570	6520	42410	51500
228	3j 14h 12m	97290	87429	1290	64500	330	4200	28260	32790

Juin 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
215	5j 21h 49m	43762	39950	345	0	160	21430	22172	43762
202	4j 23h 54m	260013	55234	327	16350	500	222190	20973	243663
245	5j 22h 32m	57883	54850	320	16000	160	21070	20653	41883

Juillet 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
206	2j 20h 14m	32064	25877	595	1000	450	23070	7544	31064
221	3j 04h 42m	27600	25808	591	0	50	20870	6680	27600
210	2j 20h 44m	56625	53788	518	25900	260	20190	10275	30725
251	3j 05h 26m	54827	53202	513	25650	30	20230	8917	29177

Août 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
209	5j 01h 44m	68969	28425	3000	2000	2140	1010	63819	66969
256	5j 09h 55m	64814	27942	3000	2000	3620	1010	58184	62814
203	3j 14h 43m	186175	181832	2996	149800	500	0	35875	36375
207	3j 14h 54m	185811	181832	2997	149850	200	1000	34761	35961

Septembre 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
194	1j 19h 03m	247339	246552	432	43000	5210	23530	175599	204339
198	1j 19h 11m	245884	245494	432	43000	5200	22540	175144	202884
197	2j 05h 52m	321121	217523	1348	67400	450	133620	119651	253721
204	2j 05h 58m	217413	217330	1343	67150	10	31690	118563	150263

Octobre 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
197	1j 08h 49m	101340	100158	253	22000	290	18610	60440	79340
206	1j 09h 14m	100191	98677	253	22000	170	18610	59411	78191
198	2j 04h 52m	41346	38132	302	15100	250	16070	9926	26246
249	2j 14h 39m	39663	37660	297	14850	10	16020	8783	24813

plus à celui escompté lors d'une minimisation. Avant, le nombre d'occurrences oscillait sans cesse d'un noeud à l'autre et le solveur travaillait très fort pour atteindre le seuil apparaissant en rouge sur le graphique *avant*. Maintenant, on commence avec beaucoup d'occurrences et on en diminue le nombre progressivement au fur et à mesure qu'on avance dans la résolution. On a donc une pénalisation plus stable qu'auparavant, ce qui aide grandement à la résolution du problème.

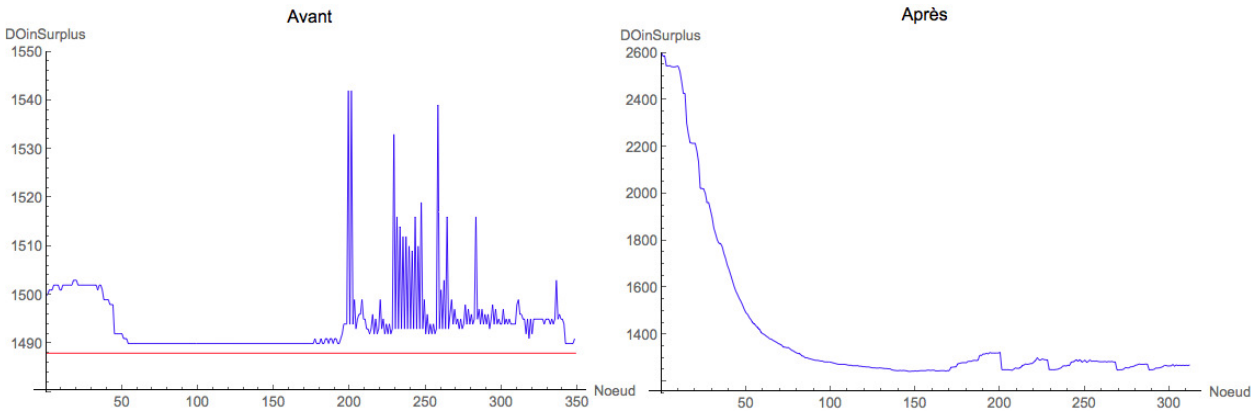

Figure 4.3 Nombre de jours de congé excédentaires à la fin de chaque noeud - Avril 2015

4.3 Méthode de branchement

4.3.1 Potentiel de Cfix

La méthode de branchement présentement utilisée, Tsplit, complexifie le problème et provoque des difficultés au niveau de la faisabilité lorsqu'un grand nombre de tâches a été fixé. Une nouvelle stratégie doit donc être adoptée.

Comme le nombre de variables fractionnaires est d'environ 5000 dans la solution du noeud 0 pour le mois d'avril 2015, le grand défi de cette nouvelle stratégie est de fixer beaucoup de variables par noeud pour faire descendre rapidement ce nombre tout en évitant de rendre le problème difficile comme le fait Tsplit. On veut pouvoir se rendre dans les meilleurs délais vers une solution entière. De plus, il faut prendre de bonnes décisions de branchement pour que les gains potentiels en temps de résolution ne soient pas obtenus au détriment de la qualité des solutions.

Après étude des différentes méthodes de branchement implémentées dans le logiciel, c'est la méthode *column fixing* (Cfix) qui s'est imposée comme la meilleure alternative. Cfix fixe des variables du problème maître, donc fixe des colonnes à 1. Chaque colonne se voit attribuer

un score $s_{col} \in [0, 1]$ correspondant à la valeur que prend la variable associée dans la solution de la relaxation linéaire. Un sous-groupe des variables est ainsi fixé selon différents critères se basant sur ces scores.

Parmi les solutions proposées pour les branchements, Cfix est en mesure de réduire très rapidement le nombre de variables fractionnaires. Son plus grand avantage est que, contrairement à Tsplit, Cfix simplifie le problème au lieu de le complexifier. En effet, lorsqu'une colonne est fixée, un employé se voit attribuer son horaire mensuel complet et cet employé peut alors être écarté du problème pour la suite de la résolution. On se retrouve ainsi avec de moins en moins d'employés au fur et à mesure que des colonnes sont fixées.

Le comportement de Cfix est dicté par les quelques paramètres ci-dessous. Seuls ceux qui seront utilisés sont présentés, les autres étant inactifs. La valeur donnée à ces paramètres pour les tests préliminaires est indiquée entre parenthèses.

- **CfixSelectMax** (50) : Détermine le nombre maximal de colonnes sélectionnées.
- **CfixSelectThreshold** (0.85) : Détermine le seuil de sélection des colonnes. Les colonnes de score s_{col} plus petit que ce seuil ne sont pas considérées.
- **CfixSelectInteger** (1) : Indique si oui (1) ou non (0) il est permis de sélectionner des colonnes de flot déjà égal à 1.
- **CfixScoreAvg** (0) : Détermine si le score initial de Cfix s_{init}^{Cfix} est la moyenne des scores s_{col} des colonnes retenues (1) ou bien le score minimal parmi ces colonnes (0).

Un premier test a été lancé pour avoir un aperçu du fonctionnement de Cfix. Le seuil de sélection a été fixé à 0.85 tel que proposent Kasirzadeh et al. (2015) dans leur article. On permet la sélection de colonnes de flot 1 de façon à ce que celles-ci soient définitivement fixées. Le paramètre CfixScoreAvg est à 0 afin d'être cohérent avec la manière dont est déterminé le score de Tsplit. Pour ce qui est du nombre maximal de colonnes sélectionnées, il a été mis à 50 de façon à ne pas restreindre le nombre total de fixations si on a beaucoup de bonnes colonnes en terme de flot. Le score initial s_{init}^{Cfix} de Cfix est reporté entre 10 et 20 et celui de Tsplit s_{init}^{Tsplit} entre 0 et 10 afin que Cfix soit toujours la méthode utilisée lorsque possible, à défaut de devoir utiliser Tsplit. Les paramètres de Tsplit restent les mêmes que présentés au chapitre 3.

La figure 4.4 permet de constater qu'énormément de colonnes sont fixées dans les premiers branchements, du noeud 0 au noeud 20. Une fois passés ces noeuds, le nombre de fixations progresse plus lentement. Les colonnes avec un flot important se font plus rares. Les solutions trouvées aux différents noeuds étant très fractionnaires, chaque employé se retrouve avec $4-5$ colonnes dont la somme des flots est égale à 1. Il arrive donc très peu souvent que l'une d'entre elles ait un flot plus grand que 0.85. Il y a alors une alternance entre Tsplit et Cfix qui se

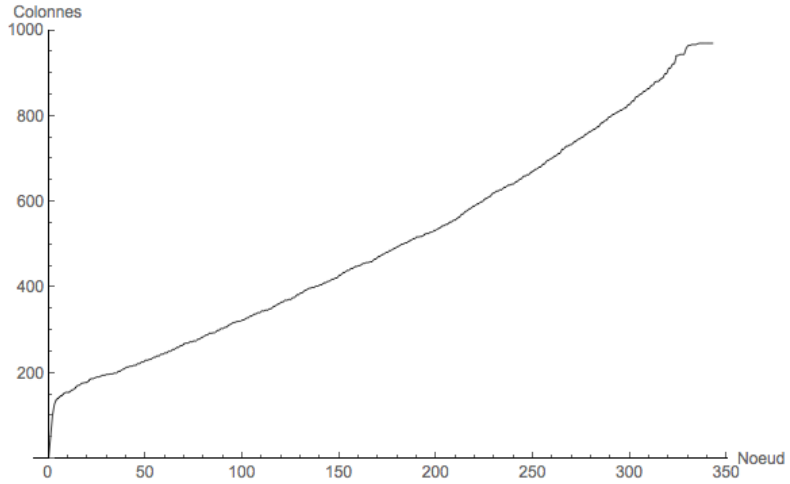

Figure 4.4 Nombre cumulatif de colonnes fixées avec Cfix pour chaque noeud - Avril 2015

produit.

Un comportement récurrent dans les branchements est observé. À la fin d'un noeud, aucune colonne n'est candidate à être fixée puisqu'aucune n'a un flot supérieur à 0.85. C'est donc Tsplit qui est utilisé et 100 tâches ou moins sont fixées. À la fin du noeud visité suivant, entre 1 et 5 colonnes sont candidates avec un flot égal à 1 ou très proche. Elles sont donc fixées. En se penchant sur le détail de ces colonnes, on peut voir qu'elles sont celles d'employés pour lesquels des tâches viennent tout juste d'être fixées par Tsplit au branchement précédent. Il y a donc utilisation de Tsplit à un noeud et au suivant on fixe les colonnes contenant le plus de ces tâches fixées.

Puisque beaucoup de colonnes sont fixées dans les 20 premiers branchements, la valeur des solutions entières trouvées dépend étroitement de la qualité de ces colonnes. Le seul critère pour les fixations étant le flot, une grande partie d'entre elles sont fixées malgré leur grand coût. Ainsi, une fois passés les 20 premiers noeuds où sont fixées un très grand nombre de colonnes, on se retrouve avec un problème réduit, mais des colonnes coûteuses sont alors imposées aux solutions. On obtient d'ailleurs une première solution entière dont le coût de la pénalité de balancement est au-dessus de 70000, ce qui est énorme en comparant avec ce qu'on avait avant. On peut voir le détail de cette solution dans le tableau 4.4 avec en gris les deux premières solutions trouvées précédemment.

Cfix est bel et bien une bonne méthode de branchement vers laquelle on peut se tourner puisque le temps de résolution s'est grandement réduit. Toutefois, deux éléments sont à surveiller. Tout d'abord, l'alternance entre Tsplit et Cfix expliquée ci-haut fait en sorte qu'on utilise deux branchements consécutifs pour fixer les mêmes éléments. Puisque les colonnes

Tableau 4.4 Première solution entière avec Cfix 0.85

Avril 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
200	2j 11h 03m	117700	87586	1324	66200	2570	6520	42410	51500
228	3j 14h 12m	97290	87429	1290	64500	330	4200	28260	32790
382	1j 21h 04m	163567	146707	1415	70750	10700	4530	77587	92817

fixées sont celles les plus étroitement liées aux tâches fixées au noeud précédent, ce branchement sert un peu à confirmer les fixations déjà faites par Tsplit. On a tout de même besoin de fixer ces colonnes pour réduire le problème. La stratégie devra donc permettre d'éviter de passer deux branchements consécutifs sur les mêmes éléments. Ensuite, il faudra assurer une certaine qualité sur les colonnes fixées durant les 20 premiers noeuds. Le coût total d'une colonne étant déterminé par la pénalité sur l'équité et nouvellement par les journées de congé excédentaires depuis le passage de cette pénalité du problème maître aux sous-problèmes, un contrôle sur ces pénalités serait à considérer en plus de celui sur le flot.

4.3.2 Nouvelle stratégie

Après maints essais gravitant autour de l'idée de contrôler la qualité des premières colonnes fixées, la stratégie suivante s'est avérée la plus efficace et a donc été adoptée.

Les colonnes considérées par Cfix doivent respecter les règles suivantes :

Noeuds 1 à 10 : $\left\lfloor \frac{|Block-Target|}{NbWorkedDays} \right\rfloor \leq EcartLimite \qquad DIS \leq \left\lfloor \frac{EvaluationPh2}{NbEmployes} \right\rfloor$

Noeuds 11 à 20 : $\left\lfloor \frac{|Block-Target|}{NbWorkedDays} \right\rfloor \leq EcartLimite \qquad DIS \leq \left\lceil \frac{EvaluationPh2}{NbEmployes} \right\rceil$

Noeuds 21 à 22 : aucune restriction

Noeuds 23 à ... : Cfix utilisable à chaque 6 noeuds seulement

Les variables et constantes sont décrites dans le tableau 4.5.

Les paramètres de Cfix prennent les valeurs indiquées dans le tableau 4.6.

Pour les 20 premiers noeuds, on tente d'aller fixer le maximum de colonnes de bonne qualité pour réduire le problème le plus possible. Deux éléments sont contrôlés en plus du flot : l'équité et les congés excédentaires. On entame les premiers branchements en étant très exigeant et on le devient un peu moins à chaque fois qu'il n'y a plus assez de colonnes fixées.

Au tout début, la racine carrée de la pénalité de balancement doit être inférieure ou égale à un écart limite de 0. Dès que moins de 5 colonnes sont fixées à un branchement, on incrémente

Tableau 4.5 Variables et constantes - Nouvelle stratégie

—	$\left\lfloor \frac{\lvert Block-Target \rvert}{NbWorkedDays} \right\rfloor$	:	Racine carrée de la pénalité de balancement de la colonne.
—	$EcartLimite$	:	Écart limite du balancement. Commence à 0, augmente de 1 unité lorsque moins de 5 colonnes ont été fixées à un branchement.
—	DIS	:	Nombre de jours de congé excédentaires de la colonne.
—	$EvaluationPh2$	:	Nombre total d'occurrences de jours de congé excédentaires obtenues dans la relaxation linéaire de la phase 2.
—	$NbEmployes$	:	Nombre total d'employés.

Tableau 4.6 Paramètres de Cfix - Nouvelle stratégie

—	CfixSelectMax	:	50
—	CfixSelectThreshold	:	0.60 (Noeuds 1 à 20) 0.85 (Noeuds 21 à ...)
—	CfixSelectInteger	:	1
—	CfixScoreAvg	:	0

cet écart limite de 1 unité dans le but d'être plus efficace sur le nombre de fixations. On est alors allé chercher presque toutes les colonnes respectant ce degré de sévérité, d'où la nécessité d'un adoucissement du contrôle. Cette façon de faire permet aussi à la méthode de s'adapter selon le développement de la résolution. On ne s'attarde pas inutilement à être trop sévère si cela ne nous permet pas de fixer beaucoup de colonnes.

Concernant les jours de congé excédentaires, on utilise le nombre total d'occurrences obtenues dans la relaxation linéaire résolue à la phase 2. C'est une estimation du nombre auquel on devrait s'attendre. Cette estimation est légèrement sous-estimée puisqu'en phase 3 on résout en nombres entiers et que la pénalité de balancement est ajoutée. Plus de contraintes devraient donner lieu à plus d'occurrences. Pour diminuer le plus possible les occurrences de cette pénalité, on exige que les colonnes fixées aux 10 premiers noeuds aient un nombre de jours de congé excédentaires inférieur ou égal au plancher de la moyenne d'occurrences par employé calculée avec l'estimation. Après ces 10 noeuds, on augmente la limite d'une unité, donc au plafond de cette même moyenne, afin d'élargir le nombre de colonnes candidates possibles.

Tout au long de ce contrôle sur la qualité, le seuil pour le flot des colonnes est à 0.60. On se laisse ainsi une plus grande marge de manoeuvre puisque le seuil usuel de 0.85 aurait empêché d'en considérer beaucoup. On a maintenant d'autres contrôles que celui sur le flot ce qui justifie une tolérance un peu plus grande pour ce paramètre.

Après 20 noeuds, on cesse tout contrôle sur la qualité et Cfix reprend son comportement normal avec une sélection standard des colonnes se basant uniquement sur le flot. Le seuil est donc remis à 0.85. Pour les deux branchements suivants, on laisse la chance aux colonnes qui n'auront pas passé les contrôles de qualité tout au long des 20 premiers branchements d'être fixées si leur flot est supérieur à 0.85. Si ces colonnes ont toujours un flot aussi important après 20 branchements, c'est qu'elles ont un grand potentiel de faire partie d'une solution entière à venir. On laisse deux branchements pour effectuer ces fixations puisque c'est le nombre requis pour cette tâche qui a été observé pour tous les 6 mois étudiés. Beaucoup de colonnes étaient fixés aux deux premiers noeuds suivant la fin du contrôle sur la qualité et on passait ensuite à l'utilisation de Tsplit au troisième noeud, faute de colonnes candidates.

Pour les branchements subséquents, on permet à Cfix d'être utilisé à chaque 6 noeuds seulement. Pendant 5 noeuds, Tsplit fixe des tâches. Au 6^{e}, Cfix vient réduire le problème en fixant toutes les colonnes pour lesquelles beaucoup de tâches ont été fixées par Tsplit ainsi que des colonnes ayant tout simplement un grand flot au dessus de 0.85. Cette technique permet de fixer une vingtaine de colonnes à chaque fois et donc de réduire le problème d'une vingtaine d'employés à chaque 6 noeuds. On évite ainsi l'alternance Tsplit/Cfix qu'on avait initialement et qui répétait essentiellement les mêmes branchements sur deux noeuds consécutifs.

4.3.3 Résultats

Les deux premières solutions entières avant et après l'implémentation de cette nouvelle stratégie de branchement figurent pour chaque mois dans le tableau 4.7. L'effet le plus manifeste est certainement le temps de résolution fortement réduit. On a obtenu une première solution entière plus rapidement pour tous les mois de données. La diminution moyenne du temps est de 37.2%, la plus grande étant de 47.19% pour le mois de juin.

Pour ce qui est de la qualité, en comparant la première solution entière obtenue avant l'utilisation de cette nouvelle stratégie et celle obtenue avec, on peut faire les constats suivants.

Tout d'abord, l'équité s'est améliorée pour trois des six mois, soit avril, juillet et août avec des pénalités de balancement de coût total plus petit en moyenne de 21.66%. Quant à juin et septembre, ce coût a légèrement augmenté, de 3.37% et 10.16% respectivement. Pour octobre, l'augmentation du coût des pénalités de balancement est de 84.92%, mais la deuxième solution entière répare cette anomalie en offrant plutôt une diminution du coût de 25.33%, solution qui est de plus obtenue plus rapidement que la première résultant de l'utilisation exclusive de Tsplit.

Ensuite, le nombre de journées de congé en surplus a en général augmenté, mais de très peu.

Tableau 4.7 Changement de la stratégie de branchement

Avril 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
200	2j 11h 03m	117700	87586	1324	66200	2570	6520	42410	51500
228	3j 14h 12m	97290	87429	1290	64500	330	4200	28260	32790
201	1j 08h 33m	91636	86792	1336	66800	270	3590	20976	24836
220	1j 12h 58m	88128	86767	1325	66250	50	3390	18438	21878

Juin 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
202	4j 23h 54m	260013	55234	327	16350	500	222190	20973	243663
245	5j 22h 32m	57883	54850	320	16000	160	21070	20653	41883
175	2j 15h 19m	72780	58248	365	18250	10050	22800	21680	54530
184	2j 15h 44m	61758	58248	367	18350	60	21720	21628	43408

Juillet 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
210	2j 20h 44m	56625	53788	518	25900	260	20190	10275	30725
251	3j 05h 26m	54827	53202	513	25650	30	20230	8917	29177
194	1j 13h 52m	57321	55063	548	27400	190	20670	9061	29921
219	1j 18h 56m	56147	55020	543	27150	310	20440	8247	28997

Août 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
203	3j 14h 43m	186175	181832	2996	149800	500	0	35875	36375
207	3j 14h 54m	185811	181832	2997	149850	200	1000	34761	35961
227	2j 14h 12m	186207	183215	2997	149850	420	1000	34937	36357
234	2j 14h 53m	183912	183087	2997	149850	150	100	33812	34062

Septembre 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
197	2j 05h 52m	321121	217523	1348	67400	450	133620	119651	253721
204	2j 05h 58m	217413	217330	1343	67150	10	31690	118563	150263
221	1j 14h 49m	241761	237862	1295	64750	12970	32230	131811	177011
231	1j 15h 56m	239226	237862	1299	64950	12770	31280	130226	174276

Octobre 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
198	2j 04h 52m	41346	38132	302	15100	250	16070	9926	26246
249	2j 14h 39m	39663	37660	297	14850	10	16020	8783	24813
184	1j 13h 00m	54655	36749	323	16150	910	19240	18355	38505
228	2j 00h 51m	40082	36577	309	15450	100	17120	7412	24632

Il y a effectivement pour cinq des six mois en moyenne 5.06% plus d'occurrences. Seul le mois de septembre a vu diminuer son nombre d'occurrences, de 3.93%.

Finalement, concernant les autres sources de coût, les résultats sont assez équivalents. Seuls deux éléments sont à noter. Premièrement, aucune des premières solutions obtenues avec la nouvelle stratégie n'a donné lieu à de grandes pénalités de non-couverture puisque toute rotation a pu être couverte à chaque fois. Ceci n'avait pas été possible dans les résultats précédents pour les mois de juin et de septembre. Deuxièmement, un coût beaucoup plus important des pénalités de qualité autres que DOinSurplus est observé pour juin et septembre. Toutefois, la situation se corrige pour juin avec la deuxième solution entière obtenue seulement 25 minutes après la première.

Somme toute, les gains importants en terme de temps de résolution ne sont pas contrebalancés par une qualité amoindrie des solutions. On a certes quelques occurrences de jours de congé excédentaires en plus, mais en contrepartie, l'équité des horaires s'est améliorée dans 4 cas sur 6.

Le tableau 4.8 permet d'étudier les fixations de colonnes effectuées lors des 20 premiers branchements et des 2 branchements subséquents. Le nombre de jours de congé excédentaires permis commence à 1 étant donné que, pour avril 2015, $\left\lfloor \frac{EvaluationPh2}{NbEmployes} \right\rfloor = \left\lfloor \frac{1488}{983} \right\rfloor = 1$.

Dès que le critère *EcartLimite* bornant la pénalité de balancement limite trop le nombre de colonnes fixées, il augmente de 1 unité ce qui provoque à chaque fois un regain dans le nombre de fixations au branchement suivant avec un minimum de 11. La méthode s'adapte donc très bien au déroulement de la résolution et permet de ne pas s'attarder sur des critères trop sévères qui ne permettent plus de fixer 5 colonnes ou plus à un branchement.

Le contrôle sur la qualité se conclut après 20 branchements et on retrouve à ce moment le contrôle standard se basant uniquement sur le flot. Le seuil est ainsi haussé à 0.85. Aux branchements 21 et 22 sont fixées les colonnes qui ont souvent été rejetées par le contrôle sur la qualité qui était jusqu'alors exercé, mais dont le flot est tout de même demeuré considérable. 28 des 45 colonnes fixées à ces branchements ont été rejetées 10 fois ou plus. Autrement dit, si le seul critère avait été un seuil de 0.85 sur le flot, ces 28 colonnes auraient eu 10 occasions ou plus d'être fixées. Parmi ces 28 colonnes, 17 d'entre elles ont été rejetées à chacun des 20 premiers branchements.

Au total, 1572 rejets de colonnes de flot supérieur ou égal à 0.85 se sont produits. La solution entière finale est composée de 897 colonnes qui n'ont jamais été rejetées et de 86 qui ont été rejetées au moins une fois. Ces 86 colonnes ont totalisé 747 rejets ce qui veut donc dire que 825 des 1572 rejets, soit 52.48%, ont servi à écarter définitivement des colonnes de trop

Tableau 4.8 Comportement de Cfix pour les 22 premiers branchements - Avril 2015

Branch.	**Jours de congé exc. permis**	**EcartLimite**	**Fixations**	$\mathbf{s_{init}^{Cfix}}$
1	1	0	11	0.6106
2	1	0	2	0.6802
3	1	1	14	0.6129
4	1	1	1	0.6065
5	1	2	11	0.6081
6	1	2	2	0.6032
7	1	3	14	0.6066
8	1	3	3	0.6154
9	1	4	12	0.6063
10	1	4	7	0.6009
11	2	4	27	0.6038
12	2	4	12	0.6044
13	2	4	10	0.6028
14	2	4	16	0.6027
15	2	4	17	0.6024
16	2	4	5	0.6006
17	2	4	8	0.6023
18	2	4	6	0.6009
19	2	4	4	0.6070
20	2	5	18	0.6009
21	*NULL*	*NULL*	44	0.8548
22	*NULL*	*NULL*	1	1.0000

mauvaise qualité qui auraient autrement été fixées par un Cfix standard.

L'observation du temps cumulatif d'utilisation des configurations sur la figure 4.5 atteste des bénéfices de la nouvelle stratégie du côté de la faisabilité.

Pour avril, la première solution entière est trouvée au noeud 201. On peut voir que, pour se rendre jusqu'à ce noeud, les deux configurations de faisabilité HAFEAS et HBKFEAS ont été les deux configurations les moins utilisées parmi les cinq. On commence à les utiliser un peu plus souvent à partir du noeud 190, mais on arrive à chaque fois à trouver une solution faisable puisque les configurations d'amélioration sont autant sollicitées à ce moment. En effet, on ne se bute à aucune relaxation linéaire infaisable.

La nouvelle stratégie de branchement permet donc d'éliminer le problème de faisabilité auparavant occasionné par l'utilisation exclusive de Tsplit. On recourt toujours à cette méthode de branchement, mais l'utilisation de Cfix à chaque 6 noeuds vient compléter le travail de

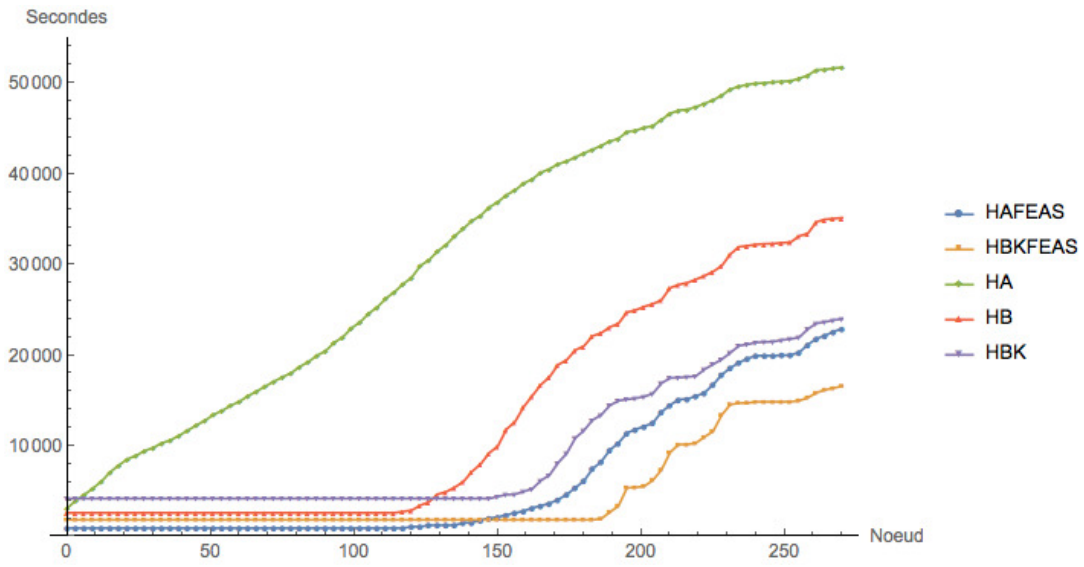

Figure 4.5 Nouveau temps cumulatif d'utilisation des 5 configurations - Avril 2015

Tsplit en achevant et en fixant définitivement les horaires ayant le plus de tâches fixées. On évite ainsi au solveur de devoir assigner les tâches restantes aux petites fenêtres de travail encore disponibles chez les employés.

La figure 4.6 expose le portrait global de la fixation des colonnes au fil de la résolution. On retrouve toujours le grand nombre de fixations aux 20 premiers noeuds. Il est un peu plus modéré que celui présenté lors du test préliminaire de Cfix (fig. 4.4), mais le contrôle exercé sur la qualité permet de réduire le problème tout en imposant des colonnes peu coûteuses aux solutions à venir, ce qui explique les biens meilleurs résultats qu'on a pour l'équité. Les pénalités de balancement sont en effet passées d'un coût de 77587 à un coût de 20976.

La forme d'escalier retrouvée dans la courbe témoigne de l'utilisation de Cfix à chaque 6 noeuds seulement lorsqu'on commence à alterner entre Tsplit et Cfix.

Finalement, la figure 4.7 confirme le très bon rendement de la nouvelle stratégie pour faire tomber rapidement le nombre de variables fractionnaires. Ce dernier diminue légèrement plus vite au fur et à mesure qu'on se rapproche du 0 et donc d'une solution entière.

4.4 Calcul des cibles de journées de congé

4.4.1 Révision du nombre minimal de jours de congé

Le nombre de jours de congé excédentaires est anormalement élevé comme nous avons pu voir dans tous les résultats jusqu'à présent. En particulier pour les mois d'avril, d'août et de

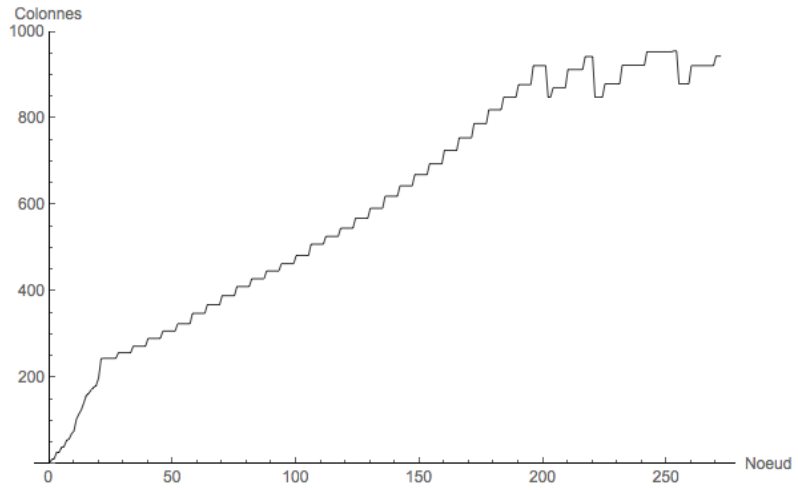

Figure 4.6 Nouveau nombre cumulatif de colonnes fixées avec Cfix pour chaque noeud - Avril 2015

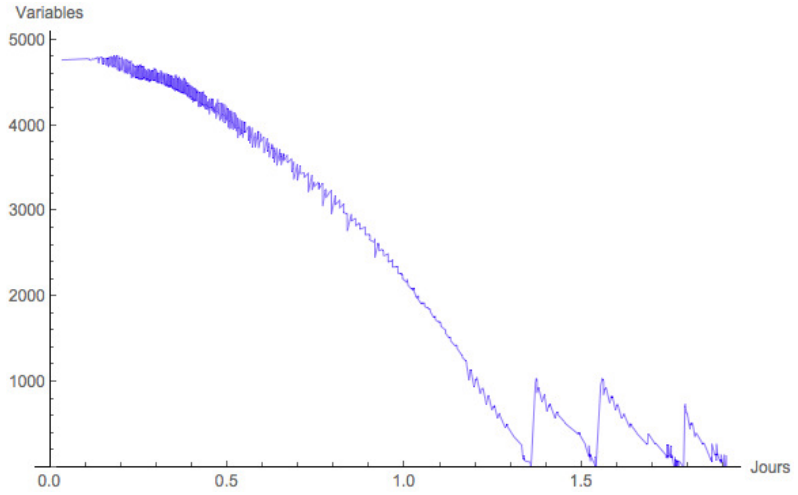

Figure 4.7 Nombre de variables fractionnaires de chaque solution par rapport au temps - Avril 2015

septembre où on a en moyenne 1.29, 2.91 et 1.33 jours de congé excédentaires par employé respectivement. Les autres mois ont un nombre moyen de jours de congé excédentaires par employé un peu plus raisonnable, tous en-deçà de 0.5.

Chaque employé se voit attribuer un nombre minimal de jours de congé qu'il devrait avoir dans son bloc mensuel. Cette valeur est utilisée comme cible. Tout congé supplémentaire est à éviter, donc pénalisé. Si le nombre de jours de congé excédentaires est si élevé pour certains mois, alors les cibles sont peut-être en cause. En effet, des cibles sous-évaluées donneront assurément lieu à beaucoup de congés en surplus.

Le procédé pour calculer les cibles a donc été révisé pour s'assurer que ces dernières soient les minimums les plus réalistes possibles. Dans un tel cas, les jours de congé excédentaires seraient réellement des congés qu'il aurait été possible d'éviter avec un horaire mieux configuré. Le nombre d'occurrences pour cette pénalité pourrait tout de même demeurer élevé, mais au moins on s'assurera que chacun de ces congés est bien un surplus et non un congé inévitable.

Initialement, le minimum de jours de congé était calculé de façon très simple. On utilisait une seule des règles régissant l'attribution des jours de congé qui en impose 8 à l'intérieur d'une période de 28 jours. Étant donné que les mois sont de 30 ou 31 jours, on additionnait à ces 8 jours de congé obligatoires un congé additionnel. Plus précisément, on appliquait la formule suivante :

$$minDO(nbJoursEffectifs) = \left\lfloor 0.5 + \frac{8 * nbJoursEffectifs}{28} \right\rfloor \tag{4.2}$$

où *minDO* est le minimum de jours de congé et *nbJoursEffectifs* est le nombre de jours où l'employé n'est pas en vacances. Tout employé disponible sur l'ensemble du mois avait donc une cible de 9 jours de congé.

Or, plusieurs autres règles d'attribution font en sorte que, pour certains employés, cette valeur minimale de 9 ne pourra jamais être atteinte peu importe l'horaire mensuel donné. Il faudrait idéalement calculer la cible d'un employé en considérant absolument toutes les règles d'attribution dans le but de s'assurer que le minimum représente bel et bien la plus grande borne inférieure du nombre de jours de congé que l'employé peut avoir dans un bloc réalisable.

Une nouvelle étape a donc été ajoutée au processus d'optimisation, nommée « phase des congés », dédiée au calcul des cibles. Pour chacun des employés est calculé le meilleur bloc mensuel qu'on peut lui donner en terme de congés selon ses disponibilités. Le chemin de plus faible coût est recherché dans un réseau dont les arcs représentent des jours de congé ou de travail. Seuls les arcs de congé ont un coût non nul. Le réseau est construit de manière à considérer les disponibilités de l'employé et des ressources veillent à ce que toutes les règles d'attribution de congés soient respectées. Le nombre de jours de congé obtenus dans le chemin de plus faible coût est alors utilisé par la suite comme nouvelle cible du nombre minimal de jours de congé. Malgré le grand nombre d'employés, les problèmes de plus court chemin sont résolus extrêmement rapidement ce qui ajoute un temps négligeable de calcul.

Après quelques tests, une brève révision des plus courts chemins de quelques employés a permis d'identifier un élément problématique causant une sous-évaluation du nombre minimal de jours de congé. Pour le client étudié, il y a possibilité de donner des jours vides de travail

dans le bloc des employés. Ces jours vides sont ensuite comblés par le client selon ses besoins une fois le jour venu. Ils sont fortement pénalisés (5000) dans le programme en nombres entiers, car le client désire qu'ils soient donnés en dernier recours seulement. Dans les réseaux pour la détermination du nombre minimal de jours de congé, les arcs de journées vides étaient de coût nul et les plus courts chemins trouvés les empruntaient fréquemment. On évitait de cette façon d'utiliser des arcs de congé de coût positif en passant par des jours vides qui, contrairement aux rotations, contribuent beaucoup moins au temps de travail. Étant donné que des règles d'attribution des congés se basent sur le temps travaillé depuis le dernier congé, les chemins étaient irréalistes. La situation a été corrigée en ajoutant le coût de la pénalité d'un jour vide sur les arcs correspondants, coût 5 fois plus grand que celui des arcs de congé. On évitera donc dans la mesure du possible ces jours vides.

4.4.2 Résultats

Ce n'est pas du côté du temps de résolution que cette amélioration a eu un impact. Un coup d'oeil au tableau 4.9 nous montre que le temps pour obtenir la première solution entière est très semblable à ce qu'on avait avant. Trois des six mois ont vu ce temps augmenter, la plus importante augmentation étant de 1.34% pour le mois d'août. Les trois autres ont vu ce temps diminuer, la plus importante diminution étant de 4.81% pour le mois de septembre.

Les plus grands changements se trouvent plutôt dans l'équité des horaires et dans le nombre de jours de congé excédentaires.

Le nombre de journées de congé en surplus a drastiquement chuté pour tous les mois. La diminution moyenne est de 51.52%. La plus importante est de 82.86% pour le mois de septembre et la plus petite est de 13.25% au mois d'août. L'impact sur la moyenne de jours de congé excédentaires par employé est présenté dans le tableau 4.10. On a maintenant pour tous les mois à l'exception d'août une moyenne en-deça de 1 jour de congé excédentaire par employé.

La figure 4.8 permet de comparer la distribution des employés selon leur nombre de jours de congé en surplus dans la solution entière finale avant et après l'amélioration du calcul du nombre minimal de jours de congé. À présent, 80.37% des employés ont une journée de congé excédentaire ou moins comparativement à 56.26% avant. Plus de la moitié n'en ont aucun (53.00%).

La diminution majeure dans le nombre d'occurrences de cette pénalité ainsi que la distribution des employés selon leur nombre de journées de congé en surplus qui se concentre maintenant surtout à 0 ou 1 occurrence nous confirment que le minimum de jours de congé est calculé

Tableau 4.9 Révision des cibles du nombre minimal de jours de congé

Avril 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
201	1j 08h 33m	91636	86792	1336	66800	270	3590	20976	24836
220	1j 12h 58m	88128	86767	1325	66250	50	3390	18438	21878
196	1j 07h 59m	56981	36576	743	37150	1990	1560	16281	19831
250	1j 20h 22m	46223	36576	742	37100	350	1520	7253	9123

Juin 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
175	2j 15h 19m	72780	58248	365	18250	10050	22800	21680	54530
184	2j 15h 44m	61758	58248	367	18350	60	21720	21628	43408
203	2j 16h 27m	37499	34095	153	7650	500	20350	8999	29849
247	2j 20h 35m	35721	33789	146	7300	280	20190	7951	28421

Juillet 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
194	1j 13h 52m	57321	55063	548	27400	190	20670	9061	29921
219	1j 18h 56m	56147	55020	543	27150	310	20440	8247	28997
203	1j 14h 17m	149996	36918	268	13400	2560	121890	12146	136596
208	1j 14h 42m	48569	36918	267	13350	2310	21980	10929	35219

Août 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
227	2j 14h 12m	186207	183215	2997	149850	420	1000	34937	36357
234	2j 14h 53m	183912	183087	2997	149850	150	100	33812	34062
232	2j 15h 02m	150134	134676	2600	130000	330	30	19774	20134
280	2j 18h 05m	147802	133938	2597	129850	40	0	17912	17952

Septembre 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
221	1j 14h 49m	241761	237862	1295	64750	12970	32230	131811	177011
231	1j 15h 56m	239226	237862	1299	64950	12770	31280	130226	174276
195	1j 12h 57m	55519	39517	222	11100	500	24320	19599	44419
251	1j 17h 30m	50241	38934	215	10750	0	23270	16221	39491

Octobre 2015

N	Temps	Val	Relax	DIS	Qual		N-C	Bal	Score
184	1j 13h 00m	54655	36749	323	16150	910	19240	18355	38505
228	2j 00h 51m	40082	36577	309	15450	100	17120	7412	24632
192	1j 12h 26m	237709	23347	131	6550	5140	218010	8009	231159
214	1j 15h 08m	24781	23086	115	5750	0	15880	3151	19031

Tableau 4.10 Moyenne de jours de congé excédentaires par employé

Mois	**Avant**	**Après**
Avril 2015	1.36	0.76
Juin 2015	0.36	0.15
Juillet 2015	0.51	0.25
Août 2015	2.91	2.53
Septembre 2015	1.28	0.22
Octobre 2015	0.32	0.13

avec une plus grande justesse. Puisqu'on considère toutes les règles d'attribution des congés, on évite de payer pour énormémement de congés inévitables qui étaient évalués comme étant en excédant auparavant. Le minimum se rapproche plus du nombre de jours de congé que le solveur n'aura autre choix que de donner.

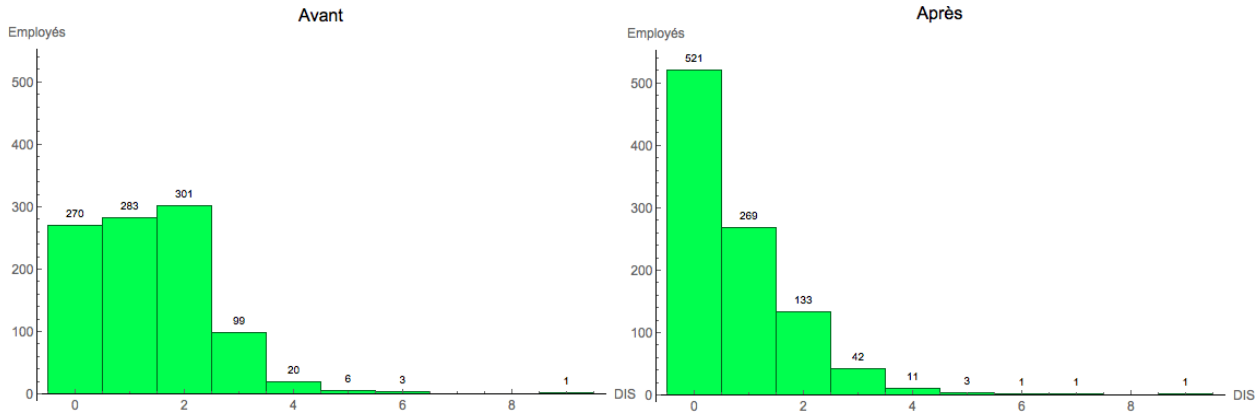

Figure 4.8 Distribution des employés selon leur nombre de jours de congé excédentaires - Avril 2015

Le plus gros avantage de cette baisse dans le nombre d'occurrences est que le solveur se concentre beaucoup plus sur l'équité des horaires, donc à diminuer les pénalités de balancement. En effet, la pénalité des jours de congé excédentaires accaparait précédemment une bonne partie de la valeur de la première solution entière trouvée, soit en moyenne 47.10%. La diminution moyenne de 51.52% du nombre d'occurrences implique que le coût de cette pénalité a diminué aussi du même pourcentage. Les pénalités de balancement reprennent donc une plus grande place dans la valeur de la solution, d'où l'effort du solveur pour les diminuer un peu plus.

Toujours en considérant la première solution entière trouvée, la valeur des pénalités de balancement a diminué pour 5 des 6 mois avec une moyenne de 53.15%. La diminution la plus

importante s'est produite en septembre avec 85.13%. Seul le mois de juillet a vu cette valeur augmenter, de 34.05%. Elle était toutefois déjà très basse, donc une petite augmentation s'est traduite en grand pourcentage.

Pour ce qui est des pénalités de non-couverture et des pénalités de qualité autres que celle des journées de congé excédentaires, on dénote très peu de différences avec les résultats précédents. La couverture des tâches s'est en général légèrement améliorée puisque la pénalité correspondante est maintenant de plus petite valeur. Seuls juillet et octobre font exception, car la première solution entière ne permet pas de couvrir 1 rotation et 2 rotations respectivement. La deuxième solution entière trouvée règle ce problème dans les deux cas.

Les autres pénalités de qualité sont somme toute semblables. On dénote par contre une diminution notable pour septembre (12970 à 500) et une petite augmentation pour juillet (190 à 2560).

En bref, la pénalisation des jours de congé excédentaires est devenue plus réaliste. On évalue beaucoup mieux le nombre minimal de jours de congé requis pour un employé dans son bloc de manière à ce que soient respectées toutes les règles en place. En conséquence, on ne paie que pour les journées de congé réellement en surplus et le solveur s'attarde donc davantage à l'équité des horaires.

4.5 Atteinte de l'objectif du projet

Le tableau 4.11 permet la comparaison entre la solution de référence et la première solution entière obtenue en utilisant l'ensemble des améliorations présentées dans ce chapitre.

Tableau 4.11 Solution de référence et solution avec améliorations - Avril 2015

N	**Temps**	**Val**	**Relax**	**DIS**	**Qual**		**N-C**	**Bal**	**Score**
341	32j 01h 24m	76121	63906	1494	6000	13290	120	56711	70121
196	1j 07h 59m	56981	36576	743	37150	1990	1560	16281	19831

Le temps CPU de calcul pour se rendre à la première solution entière est réduit de 95.84%. Il a diminué par un facteur 24. C'est largement supérieur au facteur 10 visé initialement.

Un si grand gain sur le temps ne s'est pas fait au détriment de la qualité de la solution. Au contraire, celle-ci s'est largement améliorée. Comme la pénalisation des jours de congé en surplus a changé, on ne peut comparer la valeur des solutions. En comparant plutôt le score, soit la valeur de la solution moins les coûts engendrés par les journées de congé excédentaires, on dénote une réduction de 71.72%.

Les horaires sont maintenant plus équitables, car les pénalités de balancement ont diminué de 71.29%. Pour ce qui est des pénalités de qualité autres que celle des jours de congé en surplus, elles ont baissé de 85.03%. Seules les pénalités de non-courveture ont augmenté en passant de 120 à 1560 puisque quelques *stand-by* n'ont pu être assignés.

Quant aux jours de congé excédentaires, le nombre d'occurrences est réduit de 50.27%. Cette forte réduction est presqu'entièrement attribuable au calcul plus juste du nombre minimal de jours de congé des employés qui permet d'éviter de considérer des congés inévitables comme étant excédentaires. La nouvelle façon de pénaliser les journées de congé en surplus a aussi contribué à diminuer le nombre d'occurrences, mais ses principaux apports sont d'avoir éliminé les importantes oscillations dans le nombre de jours de congé excédentaires d'un noeud à l'autre lors de la résolution et d'avoir introduit un meilleur équilibre entre équité et congés excédentaires.

Comme les exécutions des autres mois avec le logiciel initial ont été stoppées manuellement sans aucune solution entière obtenue, on ne peut avoir de pourcentages précis sur la diminution du temps de résolution et l'amélioration de la qualité pour ces données. Toutefois, comme aucun résultat n'avait été produit au bout d'une semaine, on peut affirmer que le temps de résolution a diminué d'au moins 61.64% pour juin, 77.21% pour juillet, 62.48% pour août, 78.01% pour septembre et 78.31% pour octobre. Concernant la qualité, il n'y a pas de solutions initiales auxquelles se référer pour comparer. Par contre, le score des solutions a descendu au fil des améliorations apportées comme nous laissent voir les résultats présentés au cours de ce chapitre. Même chose pour le nombre de jours de congé excédentaires qui a grandement diminué.

En bref, pour le mois de référence d'avril 2015, les améliorations qui ont été proposées permettent l'obtention d'une première solution entière dans un délai extrêmement raccourci, et ce, sans compromis sur la qualité étant donné qu'elle est meilleure que ce qu'on avait avant. Seules les pénalités de non-couverture ont légèrement augmenté, mais cette augmentation est largement compensée par les gains sur toutes les autres pénalités et sur le temps.

L'objectif d'obtenir une première solution entière de qualité égale ou meilleure en un temps réduit par un facteur d'au moins 10 est donc atteint.

CHAPITRE 5 EXPÉRIMENTATION EN CONTEXTE INDUSTRIEL

Dans ce chapitre, on abordera l'implémentation des améliorations chez AD OPT. Après présentation des résultats auprès des personnes responsables, un accès à leurs bureaux a été fourni afin que les améliorations proposées soient testées sur leurs machines. Ces dernières sont plus puissantes que celles du GERAD et ont plusieurs processeurs disponibles.

Les deux premières améliorations ainsi que la dernière avaient toutes déjà été implémentées chez AD OPT au cours du projet par leurs développeurs. Plus précisément, le changement dans la stratégie de dominance et l'ajustement de paramètres ont été adoptés en octobre 2015. Comme le démontrent les résultats du tableau 4.2, les gains en temps de résolution étaient énormes et ces améliorations ont rapidement été mises en place pour que les exécutions futures puissent en profiter. Ensuite, la nouvelle pénalisation des jours de congé excédentaires et la révision du procédé pour calculer le nombre minimal de jours de congé de chaque employé ont été introduits dans la mise à jour du logiciel de février 2016. La seule différence avec ce qui a été présenté dans les sections correspondantes du chapitre 4 est le coût unitaire de la pénalité des jours de congé excédentaires qui n'a pas été modifié. Il a été laissé à 1000 plutôt que diminué à 50.

Seule la nouvelle stratégie de branchement n'avait pas encore été considérée jusqu'alors. Étant donné qu'elle représente un changement important de la méthode Tsplit utilisée habituellement, on a préféré la soumettre à de nombreux essais et la perfectionner pour la rendre la plus performante possible avant de la proposer chez AD OPT.

Deux éléments devaient donc être testés durant cette période d'accès à leurs machines : la nouvelle stratégie de branchement et la modification du coût unitaire des jours de congé excédentaires.

Il a été demandé de généraliser certains paramètres de la stratégie de branchement détaillée dans la section 4.3.2 pour qu'elle soit utilisable avec n'importe quelles données. On présentera donc tout d'abord la généralisation adoptée. Ensuite, on montrera les résultats des tests effectués sur les machines de AD OPT pour finir avec une conclusion sur l'expérience.

5.1 Généralisation de la stratégie de branchement

Deux paramètres en particulier se devaient d'être généralisés afin que la stratégie puisse s'adapter le mieux possible aux différentes données rencontrées au cours d'une année de planification d'horaires pour le client étudié.

Le premier est α, soit le noeud auquel on incrémente d'une unité la limite du nombre de jours de congé excédentaires permis pour qu'une colonne soit considérée lors du contrôle sur la qualité exercé aux premiers branchements. Puisque les critères sur la qualité sont utilisés aux 20 premiers branchements, il avait initialement été décidé que l'augmentation d'une unité se ferait au 10e branchement, au milieu du processus.

Le deuxième paramètre, β, est le nombre de branchements pour lesquels on exerce ce contrôle sur la qualité, qui est actuellement fixé à 20. Des tests effectués avec diverses valeurs avaient permis de déterminer que c'était la valeur optimale pour les données étudiées au GERAD.

La stratégie de branchement généralisée est décrite dans les prochaines lignes.

Les colonnes candidates à être fixées par Cfix doivent respecter les règles suivantes :

N	**1**	**à**	$\boldsymbol{\alpha}$	:	$\left\lfloor \frac{\lvert Block-Target\rvert}{NbWorkedDays} \right\rfloor \leq EcartLimite$	$DIS \leq \left\lfloor \frac{EvaluationPh2}{NbEmployes} \right\rfloor$
N	$\boldsymbol{\alpha + 1}$	**à**	$\boldsymbol{\beta}$	:	$\left\lfloor \frac{\lvert Block-Target\rvert}{NbWorkedDays} \right\rfloor \leq EcartLimite$	$DIS \leq \left\lceil \frac{EvaluationPh2}{NbEmployes} \right\rceil$
N	$\boldsymbol{\beta + 1}$	**à**	$\boldsymbol{\beta + 2}$	:	aucune restriction.	
N	$\boldsymbol{\beta + 3}$	**à**	**...**	:	Cfix utilisable à chaque 6 noeuds seulement. Dans les autres cas, utilisation de Tsplit.	

EcartLimite commence à 0 et augmente de 1 unité lorsque moins de 5 colonnes ont été fixées à un branchement. On obtient α et β par les calculs suivants :

$$\beta = \left\lfloor \frac{NbEmployes}{50} \right\rfloor \tag{5.1}$$

$$\alpha = \left\lfloor \beta \times \left(\left\lceil \frac{EvaluationPh2}{NbEmployes} \right\rceil - \frac{EvaluationPh2}{NbEmployes} \right) \right\rfloor \tag{5.2}$$

Les paramètres de Cfix prennent les valeurs indiquées dans le tableau 5.1.

Tableau 5.1 Paramètres de Cfix - Généralisation de la stratégie

—	CfixSelectMax	:	50
—	CfixSelectThreshold	:	0.60 (Noeuds 1 à β) 0.85 (Noeuds $\beta + 1$ à ...)
—	CfixSelectInteger	:	1
—	CfixScoreAvg	:	0

Pour les mois étudiés au GERAD, on travaillait toujours avec environ 1000 employés. Comme

les tests ont montré que 20 était le nombre optimal de branchements alloués aux fixations de colonnes sous contrôle de qualité, ce ratio de 20 branchements pour 1000 employés est utilisé pour le calcul de β.

Concernant α, on utilise $EvaluationPh2$, le nombre total de jours de congé excédentaires obtenus dans la relaxation linéaire résolue à la phase 2, qui est une bonne estimation du nombre auquel on devrait s'attendre dans la solution entière. Limiter le nombre d'occurrences permises dans une colonne par le plancher de la moyenne de jours de congé excédentaires par employé calculée avec l'estimation et ensuite par son plafond reste une bonne approche. Toutefois, on aimerait considérer un peu mieux cette moyenne. Par exemple, si elle est de 1.21 occurrences par employé, il serait logique de permettre 1 occurrence un peu plus longtemps que 2 lors des β noeuds où est effectué le contrôle. En effet, 1.21 est beaucoup plus proche de 1 que de 2.

On se sert donc de la partie fractionnaire de cette moyenne pour calculer α. On multiplie β par l'écart entre la moyenne et le plus petit entier supérieur à cette moyenne. De cette façon, le nombre de branchements auxquels on utilisera le plancher et ensuite le plafond comme limite sera proportionnel à l'écart de la moyenne avec ces valeurs.

Par exemple, pour un mois avec 1000 employés et pour lequel on obtient 1350 journées de congé en surplus dans la solution de la relaxation linéaire de la phase 2, on obtient $\beta = 20$ et $\alpha = 13$. La fixation de colonnes avec contrôle sur la qualité se ferait donc sur les 20 premiers branchements pour lesquels on limiterait le nombre de jours de congé excédentaires des colonnes considérées à 1 pour les 13 premiers branchements et ensuite à 2 pour les 7 branchements suivants.

5.2 Tests

Les tests ont été effectués avec le mois d'avril 2016 dont la description des tâches à couvrir est donnée dans le tableau 5.2.

5.2.1 Résultats

Une exécution de référence a été lancée avec le nombre de processeurs utilisés habituellement chez AD OPT, soit 4 pour les sous-problèmes et 4 aussi pour le problème maître. Il a été étudié qu'au-delà de ces valeurs, on n'accélère plus la résolution du problème avec les méthodes et paramètres en place. Les résultats sont présentés dans le tableau 5.3. La colonne *Temps* est pour le temps réel écoulé et non plus pour le temps CPU. Plusieurs processeurs sont maintenant utilisés et on veut les solutions entières le plus rapidement possible en temps

Tableau 5.2 Description des données

	Avril 2016	
	875 employés	
Tâches	Nb	Temps
Préassignées	8198	7702h19
À assigner	15292	61394h43
Rotations	9289	59696h25
Stand-by	6003	1698h18
TOTAL	23490	69097h02

réel.

Ensuite, une exécution a été lancée avec la nouvelle stratégie de branchement. On a aussi abaissé le coût unitaire de la pénalité des jours de congé excédentaires à 100, ce qui représente une diminution par un facteur 10. On utilise toujours 4 processeurs pour les sous-problèmes et 4 pour le problème maître. Les deux premières solutions entières obtenues sont décrites dans le tableau 5.4.

Tableau 5.3 Solutions de référence - AD OPT

N	**Temps**	**Val**	**Relax**	**DIS**	**Qual**		**N-C**	**Bal**	**Score**
176	1j 17h 46m	175435	174802	122	122000	830	15390	37215	53435
179	1j 17h 47m	174263	174493	122	122000	730	15390	36143	52263

Tableau 5.4 Solutions avec nouvelle stratégie de branchement - AD OPT

N	**Temps**	**Val**	**Relax**	**DIS**	**Qual**		**N-C**	**Bal**	**Score**
161	2j 00h 25m	155724	37214	169	16900	5050	117720	16054	138824
173	2j 00h 37m	49241	37214	168	16800	10	18520	13911	32441

La nouvelle stratégie n'a pas donné lieu à un meilleur temps de résolution contrairement à ce qu'on avait au GERAD. La première solution est obtenue après un temps plus long de 15.90%.

On a 38.52% plus de jours de congé excédentaires, mais les pénalités de balancement ont diminué de 56.86%. Ceci est expliqué par le coût unitaire des jours de congé excédentaires qui est passé de 1000 à 100. Ces résultats nous démontrent donc qu'une diminution de ce coût permet un plus grand soucis du solveur pour l'équité des horaires.

Les pénalités de non-couverture sont initialement très grandes à cause d'une rotation non couverte, mais la deuxième solution obtenue 12 minutes plus tard réussit à la couvrir.

L'utilisation de la stratégie de branchement proposée implique un plus grand nombre de colonnes générées qu'avec Tsplit. On compare les deux méthodes à la figure 5.1.

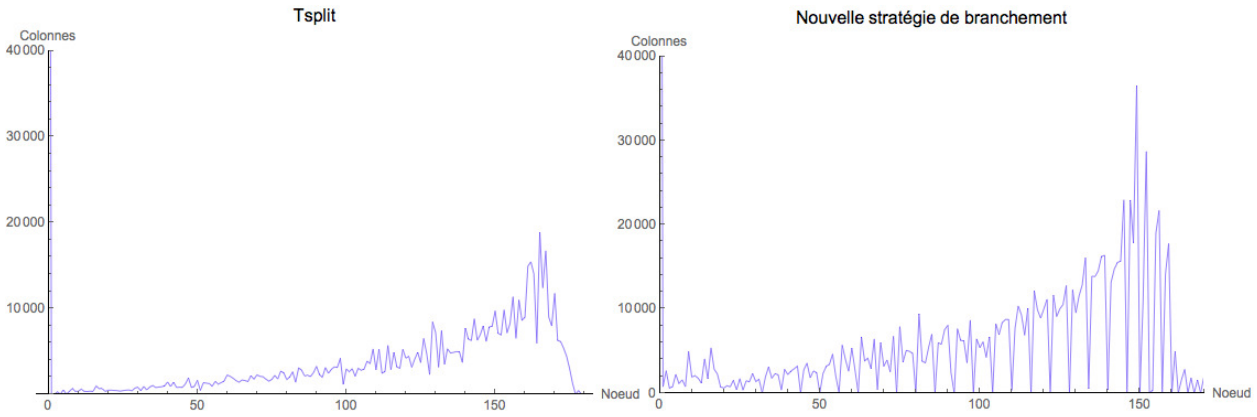

Figure 5.1 Nombre de colonnes générées à chaque noeud

Comme on peut voir sur la figure 5.2, on a toujours pour Tsplit le problème de faisabilité. En effet, le temps passé sur le problème maître devient très important tout juste avant l'obtention de la solution entière. Toutefois, l'utilisation de 4 processeurs pour le problème maître divise presque par 4 le temps nécessaire pour le résoudre et atténue de beaucoup la perte de temps causée par la complexification du problème de Tsplit.

Pour la nouvelle stratégie, le temps de résolution est majoritairement passé sur les sous-problèmes.

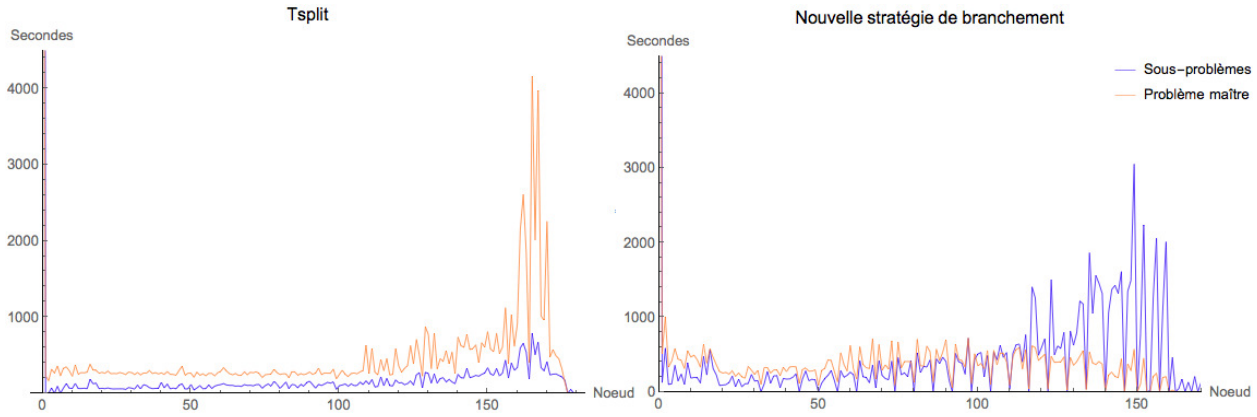

Figure 5.2 Temps de résolution pour le problème maître et les sous-problèmes à chaque noeud

5.2.2 Explications des résultats

La version du logiciel utilisée chez AD OPT est plus récente que celle qui était utilisée au GERAD. Quelques modifications ont été apportées qui ont pu jouer un rôle dans la moins bonne performance de la nouvelle stratégie de branchement.

En examinant les ressources de R_{ega} et R_{ine}, on remarque qu'une ressource a été ajoutée à R_{ega}. Cette ressource, REST_CONS_DO, compte le nombre de journées de congé consécutives lors d'une période de repos. Une situation problématique survenait quelques fois pour certains employés qui empêchait qu'un horaire leur soit construit. Cette situation ainsi que les différentes configurations d'horaires possibles pour y répondre sont présentées dans le tableau 5.5

Tableau 5.5 Problème de construction d'horaires

Situation rencontrée :						
Rotation	→	Jour libre	→	Jour libre	→	Rotation
Possibilité 1 :						
Rotation	→	Congé	→	Congé	→	Rotation
Possibilité 2 :						
Rotation	→	Congé	→	Jour vide	→	Rotation
Possibilité 3 :						
Rotation	→	Jour vide	→	Congé	→	Rotation
Possibilité 4 :						
Rotation	→	Jour vide	→	Jour vide	→	Rotation

Il arrive qu'entre deux rotations les seules possibilités soient de mettre des jours de congé ou des jours vides afin de respecter les règles régissant l'attribution des congés en considération de l'effort de travail. Étant donné que la ressource DO COUNT est élément de R_{ine} et que sa formule de mise à jour travaille avec le négatif du nombre de jours de congé donnés jusqu'alors, la solution 1 domine et les solutions 2, 3 et 4 sont éliminées puisqu'elles possèdent moins de jours de congé.

Or, une règle stipule que le repos minimal lors de deux journées de congé consécutives est de 62 heures. S'il y a moins de 62 heures dans la situation évoquée ci-haut entre la fin de la première rotation et le début de la deuxième, la solution 1 devient illégale. Puisque les autres possibilités ont été éliminées par dominance, aucune option ne s'offre à l'employé pour être légal et le solveur déduit qu'il est impossible de lui construire un horaire.

C'est pourquoi la ressource REST_CONS_DO a été ajoutée à R_{ega}. Elle force le solveur à conserver les 3 dernières possibilités, ce qui permettra de trouver un horaire légal à l'employé.

Or, une ressource en plus dans R_{ega} implique plus de chemins partiels conservés à chaque

noeud et donc une génération de colonnes plus lente. Puisque cette nouvelle ressource ne prend que 3 valeurs différentes, l'impact sur le temps de génération est modeste. Mais comme la nouvelle stratégie de branchement implique beaucoup plus de colonnes générées que c'est le cas avec Tsplit, même un impact modeste se fait ressentir et la stratégie est désavantagée.

5.2.3 Corrections

L'utilisation de plusieurs processeurs pour la résolution du problème maître a grandement diminué l'impact sur le temps de résolution qu'avait initialement la complexification du problème occasionnée par Tsplit. La nouvelle stratégie de branchement pourrait bénéficier de la même façon du parallélisme dans le traitement du problème. Comme beaucoup plus de temps est passé sur la résolution des sous-problèmes, il serait logique de leur allouer plus de processeurs.

On utilisera donc 4 processeurs pour le problème maître, mais on augmentera à 8 le nombre de processeurs pour les sous-problèmes, soit le nombre total de processeurs de la machine utilisée.

Pour diminuer le nombre d'itérations de génération de colonnes qui n'améliorent que très peu la solution courante, un paramètre non mentionné précédemment a été ajusté, soit *SppModLogItr*. Ce paramètre est lié à *SppModLogCostDecMinRatio* qui a été décrit et révisé dans la section 4.1. En effet, il détermine à combien d'itérations remonter pour évaluer si la diminution relative minimale *SppModLogCostDecMinRatio* de la solution s'est produite.

Pour chaque itération, on prend la valeur courante de la solution $CostNewest$ et on prend la valeur $CostOldest$ de la solution *SppModLogItr* itérations plus tôt pour calculer si l'inégalité suivante est respectée, faute de quoi on passe à la prochaine configuration de paramètres :

$$\frac{(CostOldest - CostNewest)}{|CostOldest|} \geq SppModLogCostDecMinRatio \tag{5.3}$$

SppModLogItr était initialement à 10 pour les 3 configurations d'amélioration HA, HB et HBK. Quant à *SppModLogCostDecMinRatio*, il a été fixé à 0.01 dans la section 4.1. Pour ces 3 configurations, il devait donc avoir une diminution relative minimale de 1% sur les 10 dernières itérations.

SppModLogItr a été réduit de moitié à 5 pour HA et HB et à 8 pour la dernière et la plus précise des configurations, HBK.

Beaucoup de temps était perdu à générer un grand nombre de colonnes pour de très petits gains sur la solution en utilisant une même configuration. Il fallait passer par un minimum

de 10 itérations avant que le critère d'arrêt *SppModLogCostDecMinRatio* signale le problème. En réduisant de moitié le nombre d'itérations requis pour évaluer l'amélioration minimale de la solution, on réalise 2 fois plus vite le problème et on perd donc beaucoup moins de temps.

Le paramètre a été diminué à seulement 8 pour HBK pour s'assurer de ne pas trop être exigeant envers la configuration la plus précise. En effet, on ne veut pas trop contraindre la performance du solveur de manière à ce qu'il puisse améliorer la solution même si cela demande occasionnellement un peu plus d'itérations.

Les résultats obtenus suite à ces quelques ajustements sont présentés dans le tableau 5.6.

Tableau 5.6 Solutions avec nouvelle stratégie de branchement après corrections - AD OPT

N	**Temps**	**Val**	**Relax**	**DIS**	**Qual**		**N-C**	**Bal**	**Score**
154	1j 12h 09m	146341	34499	159	15900	1350	118510	10581	130441
167	1j 13h 03m	49640	34499	160	16000	6100	18630	8910	33640

La première solution entière a été obtenue dans un temps réel 13.45% plus court que lors de l'exécution de référence. Toutefois, une rotation n'a pas été couverte ce qui engendre des coûts importants de la pénalité de non-couverture.

La deuxième solution entière couvre cette rotation et est tout de même obtenue dans un temps 11.29% plus court que la première solution de l'exécution de référence. Son score est plus petit de 37.05%, en grande partie à cause de la pénalité de balancement qui a diminué de 75.35%. Quant aux jours de congé excédentaires, on a 31.15% plus d'occurrences.

Ces résultats confirment l'impact que peut avoir le coût unitaire des journées de congé en surplus. Puisqu'on travaille avec beaucoup d'occurrences de cette pénalité, une diminution significative de ce coût, ici d'un facteur 10, permet d'avoir des pénalités de balancement beaucoup plus petites et donc une meilleure équité des horaires. Par contre, on se retrouve avec plus de jours de congé excédentaires.

5.3 Conclusions sur l'expérience

La nouvelle stratégie de branchement aura, après quelques ajustements, permis d'obtenir des solutions entières dans un délai plus court qu'avec Tsplit. On a certes dû augmenter le nombre de processeurs pour le traitement des sous-problèmes, mais comme plusieurs machines sont allouées à chaque mois pour la construction des blocs mensuels du client, cela ne devrait pas causer problème. Autrement, à chaque coup d'oeil jeté à l'exécution avec 8 processeurs accordés aux sous-problèmes, jamais plus de 6 d'entre eux étaient utilisés. On pourrait donc

éventuellement abaisser ce nombre à 6 sans affecter la vitesse de la résolution si une étude plus approfondie de diverses exécutions permettait de certifier qu'il n'y a jamais plus de 6 processeurs qui peuvent être utilisés pour le traitement des sous-problèmes.

Pour ce qui est du coût unitaire des jours de congé en surplus, l'option de baisser ce coût pourra être considérée chez AD OPT à la lumière des résultats présentés. On peut donc diminuer le coût unitaire au profit d'horaires beaucoup plus équitables avec pour effet d'avoir plus de jours de congé excédentaires. L'autre alternative est de le maintenir à 1000 pour pousser le solveur à diminuer le plus possible le nombre d'occurrences au détriment de l'équité des horaires.

Il a finalement été demandé de tester la stratégie de branchement avec un ensemble de bases différentes du client afin de vérifier s'il est envisageable de remplacer définitivement la méthode Tsplit. Les bases pour lesquelles les données ont été fournies sont très petites avec un maximum de 150 employés. Les exécutions originales utilisant Tsplit comme méthode de branchement se concluent toutes en moins de 5 heures, et ce, avec un seul processeur utilisé.

Pour toutes ces bases, l'utilisation de la nouvelle stratégie de branchement a donné des résultats dans des délais plus longs que lorsque Tsplit est utilisé. Ces résultats étaient prévisibles puisqu'un nombre aussi bas d'employés permet d'éviter la complexification du problème qu'occasionne Tsplit lorsque utilisé pour de grandes bases. Dans de telles conditions, on ne rencontre jamais le problème de faisabilité et la résolution est constante jusqu'à l'obtention de solutions entières. Il n'y a donc pas place à l'amélioration du côté des branchements lorsqu'on travaille avec aussi peu d'employés.

Avec la nouvelle stratégie utilisant Cfix, beaucoup plus de colonnes sont générées, ce qui prend un temps important. Pour les grandes bases, ce temps plus important requis pour résoudre des sous-problèmes était très inférieur au temps passé sur les configurations de faisabilité avec Tsplit. Puisqu'on perd pour les petites bases le problème de faisabilité, Tsplit retrouve l'avantage sur le temps de résolution total avant une solution entière.

La stratégie de branchement a été développée exclusivement pour répondre aux difficultés rencontrées dans les problèmes avec beaucoup d'employés lorsque Tsplit est utilisé. Dans cette perspective, l'utilisation de la stratégie avec de plus petits problèmes n'a pas été considérée. Ainsi, dans de futurs travaux, il serait intéressant de déterminer le nombre d'employés à partir duquel il devient avantageux d'utiliser la nouvelle stratégie au lieu de Tsplit. On pourrait alors l'activer lorsque le problème étudié contient assez d'employés pour bénéficier de l'avantage qu'elle procure sur le temps de résolution et utiliser Tsplit dans tous les autres cas.

Néanmoins, 3 des 4 améliorations proposées dans ce projet ont été officiellement adoptées chez

AD OPT. Pour ce qui est de la stratégie de branchement, elle a démontré une plus grande efficacité que Tsplit pour les bases avec beaucoup d'employés et pourra donc éventuellement être utilisée afin d'accélérer la résolution des plus gros problèmes.

CHAPITRE 6 CONCLUSION

L'objectif de ce projet de maîtrise était de comprendre les raisons expliquant le très grand intervalle de temps requis avant d'obtenir une première solution entière lors de la construction de blocs mensuels avec équité pour une base avec environ 1000 employés et de proposer des améliorations pour réduire significativement ce temps.

On présentera dans cette conclusion une synthèse des travaux qui ont été réalisés pour atteindre cet objectif. On exposera ensuite les limitations de la solution qui a été proposée pour finalement mentionner des améliorations futures qui pourront être apportées.

6.1 Synthèse des travaux

Des outils d'analyse ont tout d'abord été développés afin d'étudier rapidement les exécutions lancées du logiciel et de repérer facilement à l'aide de graphiques les anomalies dans le processus de résolution. Grâce à ces outils, un diagnostic a été rendu au chapitre 3 détaillant les éléments problématiques, soient la pénalité de qualité des jours de congé excédentaires et la méthode de branchement Tsplit. Plus précisément, pour la pénalité des jours de congé excédentaires, le rôle de la ressource comptant les journées de congé dans la stratégie de dominance ainsi que la pénalisation uniquement des occurrences dépassant un seuil nuisent aux bonnes performances du solveur. Quant à la méthode de branchement, les fixations de nombreuses tâches au fil de la résolution complexifient progressivement le problème, ce qui fait en sorte qu'énormément de temps est passé à tenter d'obtenir une solution faisable à certains noeuds.

Des améliorations ont ensuite été proposées au chapitre 4 afin de corriger les éléments problématiques énoncés dans le diagnostic.

Tout d'abord, la ressource comptant les jours de congé lors de la génération de colonnes n'est plus comparée à égalité, mais avec une inégalité. Ainsi, lors de la comparaison de chemins partiels, on augmente le nombre d'éliminations possibles ce qui fait en sorte que beaucoup moins d'étiquettes sont conservées. La génération de colonnes en est donc accélérée.

Ensuite, pour la pénalisation des jours de congé excédentaires, on a éliminé l'utilisation du seuil de pénalisation pour simplement pénaliser chaque occurrence, directement dans les sous-problèmes.

Après, nous avons proposé une stratégie de branchement utilisant Cfix. Pour les 20 premiers branchements, on fixe des colonnes qui ont une pénalité de balancement et des occurrences de

congés excédentaires en deça de certaines limites. Ces limites augmentent progressivement au fil de ces 20 branchements pour ne pas trop contraindre le nombre de colonnes fixées. Une fois passés ces 20 premiers branchements, on alterne entre Cfix et Tsplit en se basant uniquement sur le flot pour déterminer les éléments à fixer.

Finalement, l'évaluation du nombre minimal de jours de congé de chacun des employés a été complètement revue. On considère maintenant toutes les règles qui régissent l'attribution des congés de manière à ce que le minimum obtenu soit le plus réaliste possible, donc atteignable pour certaines configurations d'horaire. On évite ainsi de pénaliser des congés alors qu'ils sont inévitables.

Au final, on a réussi à obtenir une première solution entière dans un temps de résolution réduit d'un facteur 24. Cette solution a 50.27% moins de journées de congé en surplus et sa valeur est plus petite de 71.72% de ce qu'on avait initialement.

6.2 Limitations de la solution proposée

Tous les travaux proposés dans ce projet portent sur un logiciel spécifique de chez AD OPT développé pour un client particulier. Les résultats obtenus ne sont donc pas applicables pour la construction des blocs des autres clients. En effet, les améliorations qui ont été présentées se concentrent sur une pénalité de qualité très spécifique, les jours de congé excédentaires, qui ne revient pas nécessairement pour d'autres clients.

Seule l'idée derrière la stratégie de branchement pourrait potentiellement être réutilisée pour d'autres projets. En effet, on pourrait fixer des colonnes selon d'autres critères que le flot et introduire des critères de qualité qui deviennent de moins en moins stricts au fur et à mesure qu'on avance dans la résolution. Par exemple, dans le cas où l'on voudrait considérer des préférences précises des employés, on pourrait fixer seulement des colonnes contenant un nombre de préférences respectées plus grand qu'une certaine limite. Lorsqu'il n'y a plus assez de fixations, on pourrait diminuer cette limite ou bien éventuellement passer à une alternance entre Tsplit et Cfix se basant uniquement sur le flot.

Aussi, les améliorations apportées au logiciel ont été étudiées pour une seule base, soit la plus grande du client. À chaque mois de données, on avait toujours environ 1000 employés. Pour de petites bases avec très peu d'employés, on a vu que la nouvelle stratégie de branchement perd son avantage sur Tsplit. Il n'a pas été étudié dans quelles conditions exactes la stratégie peut être plus performante que Tsplit lorsqu'on travaille avec d'autres bases.

Pour ce qui est des autres améliorations, soit la nouvelle façon de pénaliser les jours de congé excédentaires, le changement dans la stratégie de dominance ainsi que la révision de

l'évaluation du nombre minimal de jours de congé, elles ont toutes été officiellement adoptées dans le logiciel chez AD OPT et sont donc valables pour n'importe quelle base du client.

Les outils de diagnostic qui ont été développés dans le cadre de ce projet pourraient être utilisés pour analyser le processus de résolution des problèmes des autres clients afin d'identifier les faiblesses et de trouver les corrections à apporter. Ce mémoire peut être vu comme une preuve du concept qu'une analyse systématique du processus de résolution permet de l'améliorer fortement.

6.3 Améliorations futures

Le premier élément qui demanderait des recherches plus poussées est la stratégie de branchement. On voudrait être en mesure de préciser toutes les conditions pour lesquelles il devient avantageux de l'utiliser au lieu de Tsplit. De cette manière, on pourrait créer un nouveau paramètre autorisant ou non l'utilisation de la stratégie selon les données du problème étudié. Il faudrait donc faire des tests avec diverses bases comprenant un nombre différent d'employés et d'heures de vol à couvrir pour ensuite analyser dans quelles circonstances la stratégie de branchement a le mieux performé.

Il serait aussi intéressant d'étudier les autres méthodes de branchement implémentées dans le logiciel. On pourrait potentiellement trouver une meilleure méthode à intégrer à la stratégie présentée plutôt que de continuer à utiliser Tsplit en alternance avec Cfix une fois la fixation de colonnes avec contrôle sur la qualité terminée.

Le deuxième élément sur lequel il faudrait se pencher est une révision des coûts des pénalités de qualité. Puisqu'on pénalise maintenant chaque occurrence de jour de congé excédentaire et que AD OPT utilise toujours un coût unitaire de 1000 pour cette pénalité, le coût total des journées de congé en surplus prend une place extrêmement importante dans la valeur de la solution. Il faudrait donc évaluer de nouveau l'importance qu'on veut accorder à chacune des pénalités de qualité et faire les ajustements nécessaires afin de retrouver des proportions semblables dans la valeur de la solution à ce qu'on avait avant de changer la pénalisation des jours de congé excédentaires.

RÉFÉRENCES

H. Achour, M. Gamache, F. Soumis, et G. Desaulniers, “An exact solution approach for the preferential bidding system problem in the airline industry”, *Transportation Science*, vol. 41, no. 3, pp. 354 – 365, 2007. DOI : 10.1287/trsc.1060.0172

C. Barnhart, A. M. Cohn, E. L. Johnson, D. Klabjan, G. L. Nemhauser, et P. H. Vance, “Airline crew scheduling”, dans *Handbook of Transportation Science*, R. W. Hall, éd. Boston, MA : Springer US, 2003, pp. 517–560.

J. Beasley et B. Cao, “A tree search algorithm for the crew scheduling problem”, *European Journal of Operational Research*, vol. 94, no. 3, pp. 517 – 526, 1996. DOI : 10.1016/0377-2217(95)00093-3

P. Belobaba, A. Odoni, et C. Barnhart, *The Global Airline Industry (2nd Edition)*. Wiley, 2015.

K. Boubaker, G. Desaulniers, et I. Elhallaoui, “Bidline scheduling with equity by heuristic dynamic constraint aggregation”, *Transportation Research Part B : Methodological*, vol. 44, no. 1, pp. 50 – 61, 2010. DOI : 10.1016/j.trb.2009.06.003

K. W. Campbell, R. B. Durfee, et G. S. Hines, “Fedex generates bid lines using simulated annealing”, *Interfaces*, vol. 27, no. 2, pp. 1–16, 1997. DOI : 10.1287/inte.27.2.1

I. T. Christou, A. Zakarian, J.-M. Liu, et H. Carter, “A two-phase genetic algorithm for large-scale bidline-generation problems at Delta air lines”, *Interfaces*, vol. 29, no. 5, pp. 51–65, 1999. DOI : 10.1287/inte.29.5.51

H. Dawid, J. Konig, et C. Strauss, “An enhanced rostering model for airline crews”, *Computers & Operations Research*, vol. 28, no. 7, pp. 671 – 688, 2001. DOI : 10.1016/S0305-0548(00)00002-2

P. R. Day et D. M. Ryan, “Flight attendant rostering for short-haul airline operations”, *Operations Research*, vol. 45, no. 5, pp. 649 – 61, 1997. DOI : 10.1287/opre.45.5.649

G. Desaulniers, J. Desrosiers, M. Gamache, et F. Soumis, “Crew scheduling in air transportation”, dans *Fleet Management and Logistics*, T. G. Crainic et G. Laporte, éds. Boston, MA : Springer US, 1998, pp. 169–185.

W. El Moudani, C. Cosenza, M. de Coligny, et F. Mora-Camino, “A bi-criterion approach for the airlines crew rostering problem”, Berlin, Germany, 2001, pp. 486 – 500.

T. Fahle, U. Junker, S. Karisch, N. Kohl, M. Sellmann, et B. Vaaben, “Constraint programming based column generation for crew assignment”, *Journal of Heuristics*, vol. 8, no. 1, pp. 59 – 81, 2002. DOI : 10.1023/A:1013613701606

M. Gamache, F. Soumis, D. Villeneuve, J. Desrosiers, et E. Gelinas, “The preferential bidding system at Air Canada”, *Transportation Science*, vol. 32, no. 3, pp. 246–255, 1998. DOI : 10.1287/trsc.32.3.246

M. Gamache, F. Soumis, G. Marquis, et J. Desrosiers, “A column generation approach for large-scale aircrew rostering problems”, *Operations Research*, vol. 47, no. 2, pp. 247 – 62, 1999. DOI : 10.1287/opre.47.2.247

M. Gamache, A. Hertz, et J. Ouellet, “A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding”, *Computers & Operations Research*, vol. 34, no. 8, pp. 2384 – 2395, 2007. DOI : 10.1016/j.cor.2005.09.010

M. Gamache et F. Soumis, “A method for optimally solving the rostering problem”, dans *Operations Research in the Airline Industry*, G. Yu, éd. Boston, MA : Springer US, 1998, pp. 124–157.

B. Gopalakrishnan et E. L. Johnson, “Airline crew scheduling : State-of-the-art”, *Annals of Operations Research*, vol. 140, no. 1, pp. 305–337, 2005. DOI : 10.1007/s10479-005-3975-3

A. I. Z. Jarrah et J. T. Diamond, “The problem of generating crew bidlines”, *Interfaces*, vol. 27, no. 4, pp. 49–64, 1997. DOI : 10.1287/inte.27.4.49

A. Kasirzadeh, M. Saddoune, et F. Soumis, “Airline crew scheduling : models, algorithms, and data sets”, *EURO Journal on Transportation and Logistics*, pp. 1–27, 2015. DOI : 10.1007/s13676-015-0080-x

D. Klabjan, “Large-scale models in the airline industry”, dans *Column Generation*, G. Desaulniers, J. Desrosiers, et M. M. Solomon, éds. Boston, MA : Springer US, 2005, pp. 163–195.

N. Kohl et S. E. Karisch, “Airline crew rostering : Problem types, modeling, and optimization”, *Annals of Operations Research*, vol. 127, no. 1, pp. 223–257, 2004. DOI : 10.1023/B:ANOR.0000019091.54417.ca

B. Maenhout et M. Vanhoucke, “A hybrid scatter search heuristic for personalized crew rostering in the airline industry”, *European Journal of Operational Research*, vol. 206, no. 1, pp. 155 – 67, 2010. DOI : 10.1016/j.ejor.2010.01.040

M. Sellmann, K. Zervoudakis, P. Stamatopoulos, et T. Fahle, “Crew assignment via constraint programming : integrating column generation and heuristic tree search”, *Annals of Operations Research*, vol. 115, pp. 207–225, 2002. DOI : 10.1023/A:1021105422248

J. D. Weir et E. L. Johnson, “A three-phase approach to solving the bidline problem”, *Annals of Operations Research*, vol. 127, no. 1, pp. 283–308, 2004. DOI : 10.1023/B:ANOR.0000019093.93633.bc

ANNEXE A DESCRIPTION DES PÉNALITÉS

Pénalités pour la non-couverture

Une tâche non couverte est pénalisée selon son niveau d'importance. Les 4 niveaux sont les suivants :

- **Tâche de niveau 0** : Les rotations.
  Coût : 100000
- **Tâche de niveau 1** : Les *stand-by* à assurer de l'aéroport.
  Coût : 1000
- **Tâche de niveau 2** : Les *stand-by* à assurer de la maison.
  Coût : 100
- **Tâche de niveau 3** : Les *stand-by* de courte durée.
  Coût : 10

Pénalités sur la qualité

Les 13 pénalités de qualité sont les suivantes :

- **SingleDO** : Jours de congé individuels, c'est-à-dire un congé d'une seule journée précédé et suivi par du travail.
  Coût : 200
- **DOinSurplus** : Jours de congé en surplus, soit des jours de congé donnés au-delà du minimum requis pour l'employé pendant le mois.
  Coût : 1000
- **MaxTransitionPerMonth** : Dépassement du nombre maximum de transitions permis durant le mois. Une transition est le passage vers un autre type de quart de travail.
  Coût : 1re occurrence : 250, 2e occurrence : 550, 3e occurrence et plus : 800
- **EarlyToLateTransition** : Transition d'un quart de travail du matin à un quart de travail de soir.
  Coût : 10
- **LateToEarlyTransition** : Transition d'un quart de travail du soir à un quart de travail du matin.
  Coût : 12500
- **ElongFirstDay** : Tâche de plus longue durée effectuée au premier jour d'une séquence

de travail.
Coût : 100

— **ElongLastDay** : Tâche de plus longue durée effectuée au dernier jour d'une séquence de travail.
Coût : 150

— **LateOffFirstDay** : Quart de travail de soir passé hors-service au premier jour d'une séquence de travail.
Coût : 50

— **BlankDay** : Journée de travail vide à être remplie ultérieurement par la compagnie aérienne selon les besoins.
Coût : 5000

— **EarlyInLateBlock** : Tâche matinale présente dans un quart de travail de soir.
Coût : 1000

— **EarlyPerWorkDay** : Nombre moyen de tâches matinales par jour de travail dans l'horaire mensuel d'un employé.
Coût : 20000

— **PrgRestrictionMonth** : Restrictions non respectées au niveau des rotations assignées à un employé dans le mois étudié.
Coût : 10

— **PrgRestrictionWorkBlock** : Restrictions non respectées au niveau des rotations assignées à un employé dans une séquence de travail.
Coût : 10

ANNEXE B FICHIERS D'ANALYSE

Les fichiers texte suivants sont produits par le programme *txt_maker.py* à partir du journal (log file) de l'activité interne du processus de résolution.

Distribution de certaines données dans la solution entière finale

- **distr_bal** : Racine carrée de la pénalité de balancement pour chaque employé.
- **distr_do_sur** : Nombre de jours de congé excédentaires de chaque employé.
- **distr_blocks** : Temps de travail assigné à chaque employé.
- **distr_targets** : Temps de travail ciblé de chaque employé.

Comportement de la résolution au fil des noeuds visités dans l'arbre

- **node_col** : Nombre de colonnes générées à chaque noeud.
- **node_ite** : Nombre d'itérations de génération de colonnes à chaque noeud.
- **node_moy_ite** : Moyenne du temps passé sur une itération de génération de colonnes pour chaque noeud.
- **sol_bnd** : Valeur de la relaxation linéaire du problème et de la borne inférieure du noeud respectivement à chaque noeud.
- **node_frac** : Nombre de variables fractionnaires dans la solution finale de chaque noeud.
- **node_col_dist** : Distribution des colonnes selon leur valeur prise dans la solution finale de chaque noeud (]0, .1[- [.1, .2[- [.2, .3[- [.3, .4[- [.4, .5[- [.5, .6[- [.6, .7[- [.7, .8[- [.8, .9[- [.9, 1[- 1).
- **node_costs** : Coût des pénalités de non-couverture, des pénalités de qualité et des pénalités de balancement respectivement dans la solution finale de chaque noeud.
- **node_do_sur** : Nombre total de jours de congé excédentaires dans la solution finale de chaque noeud.

Informations sur les branchements effectués

- **sbb_cumul** : Pour chaque noeud, nombre cumulatif de colonnes fixées par Cfix, de tâches de type TRP, GND et DO respectivement se retrouvant dans une colonne fixée

par Cfix, de tâches fixées par Tsplit non incluses dans une colonne fixée, de tâches de type TRP, GND et DO respectivement fixées par Tsplit non incluses dans une colonne fixée.

— **sbb_method** : Utilisation des méthodes Tsplit et Cfix respectivement à chaque branchement (nombre de fixations et score).
— **sbb_staff** : Pour chaque noeud, nombre d'employés pour lesquels une colonne correspondante n'a toujours pas été fixée par Cfix.
— **sbb_target** : Pour chaque noeud, nombre cumulatif de colonnes fixées avec un temps de travail supérieur à celui initialement ciblé et de colonnes fixées avec un temps inférieur, moyenne cumulative en minutes de l'écart au temps initialement ciblé pour les colonnes fixées avec un temps de travail supérieur et pour les colonnes fixées avec un temps inférieur, moyenne cumulative globale.
— **sbb_do_surp** : Pour chaque noeud, nombre cumulatif de jours de congé excédentaires de l'ensemble des colonnes fixées par Cfix et moyenne cumulative de jours de congé excédentaires par colonne fixée.
— **staff_tasks** : Pour chaque noeud, nombre cumulatif de tâches de type TRP, GND et DO fixées par l'intermédiaire de Tsplit ou Cfix pour chaque employé.

Détails sur le temps de résolution

— **time_node** : Pour chaque noeud, temps écoulé (en jours) depuis le début de la résolution.
— **time_frac** : Pour chaque solution trouvée, temps de résolution (en jours) et nombre de variables fractionnaires.
— **time_sp_mp** : Temps passé sur la résolution des sous-problèmes, sur la résolution du problème maître et au total à chaque noeud.
— **mod_cumul** : Temps cumulatif d'utilisation des modèles HAFEAS, HBKFEAS, HA, HB, HBK respectivement à chaque noeud.