PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Mise à jour des horaires de personnel travaillant sur des quarts
Auteur: Author:	Camille Froger
Date:	2015
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Froger, C. (2015). Mise à jour des horaires de personnel travaillant sur des quarts [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1752/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/1752/
Directeurs de recherche: Advisors:	François Soumis, Guy Desaulniers, & Beyime Tachefine
Programme: Program:	Mathématiques appliquées

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

MISE À JOUR DES HORAIRES DE PERSONNEL TRAVAILLANT SUR DES QUARTS

CAMILLE FROGER
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)
MAI 2015

© Camille Froger, 2015.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

MISE À JOUR DES HORAIRES DE PERSONNEL TRAVAILLANT SUR DES QUARTS

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

M. LE DIGABEL Sébastien, Ph. D., président
M. SOUMIS François, Ph. D., membre et directeur de recherche
M. DESAULNIERS Guy, Ph. D., membre et codirecteur de recherche
M. TACHEFINE Beyime, Ph. D., membre et codirecteur de recherche
Mme REKIK Monia, Ph. D., membre

REMERCIEMENTS

Je tiens tout d'abord à remercier mes directeurs de recherche : François Soumis, Guy Desaulniers et Beyime Tachefine. François Soumis a accepté de m'accueillir au GERAD sous sa direction et m'a permis de travailler sur ce projet. Sans lui, je ne serais certainement pas là où je suis actuellement. Guy Desaulniers m'a beaucoup épaulée et encouragée tout au long de ce projet que ce soit pour me donner des idées ou pour les relectures. Enfin, Beyime Tachefine a apporté une aide régulière avec un point de vue plus pratique sur le projet.

Je souhaite remercier Sébastien Le Digabel qui a accepté de présider mon jury ainsi que Monia Rekik qui a accepté d'en faire partie.

Merci à François Lessard pour sa patience et sa dévotion pendant l'ensemble du projet. Il a résolu avec moi de manière rigoureuse l'ensemble des problèmes qui se sont présentés sur notre chemin et ce fut un réel plaisir de travailler avec lui. Merci également à Simon Proulx et Antoine Legrain d'avoir pris le temps de m'expliquer leurs travaux sur la génération de scénarios de demande et de m'avoir ainsi fait gagner un temps précieux.

Je souhaite également remercier toutes les personnes du GERAD qui ont partagé mon quotidien pendant ces deux années passées à leurs côtés pour les moments de détente bienvenus qu'ils m'ont apportés. Je remercie notamment mes camarades de bureau, Marilène Cherkesly, Charles Gauvin, Atoosa Kasirzadeh, Hélène Michon-Lacaze, Romain Montagné, Lê Nguyên Hoang et Frédéric Quesnel mais aussi Claire Bernard, Sébastien Debia, Antoine Legrain, Thibault Lehouiller, Jérémy Omer, Samuel Rosat et Bastien Talgorn.

Je souhaite remercier mes amis de Montréal et en particulier Hélène Michon-Lacaze, Geoffroy Mouret, Arthur Ruellan et Julien Keutchayan sans qui ces deux années auraient été bien ternes. Je remercie également l'ensemble de la troupe de PolyThéâtre qui m'a permis d'exprimer mes émotions. Merci à Nadia Arfaoui et Faiza Maskhouni d'avoir été de magnifiques colocataires et à Sophie O'Shaugnessey pour tous les moments de complicité partagés.

Je remercie aussi l'ensemble de ma famille de m'avoir accompagnée dans ce projet et particulièrement Francine Paumier pour sa relecture attentive. Je remercie enfin et surtout Emmanuel Bach qui m'a soutenue de manière inconditionnelle.

RÉSUMÉ

L'objectif d'une entreprise est de minimiser les dépenses liées au personnel tout en respectant les règlementations et conventions collectives et en offrant un service de qualité qui répond à la demande de la clientèle. Ainsi, la génération d'horaires de personnel prend une grande place dans les problèmes traités en recherche opérationnelle. Il est cependant difficile de prévoir cette demande de manière précise longtemps à l'avance et il est possible que l'horaire généré un mois plus tôt avec une certaine prévision de demande ne soit plus valable pour la demande observée le jour même. Le fait que la demande réelle ne sera pas égale à la demande prévue peut parfois être anticipé quelques jours à l'avance et il faut alors mettre à jour l'horaire pour répondre au mieux à cette demande.

Ce projet de maîtrise répond à ce problème de mise à jour de l'horaire de personnel travaillant sur des quarts. Nous nous plaçons dans un contexte de vente au détail où l'horaire est non continu. Nous proposons un modèle explicite en nombres entiers qui permet une résolution exacte du problème.

Dans un premier temps, nous générons différents scénarios de demande qui nous permettront de tester nos propositions. Ces scénarios sont générés avec introduction de paramètres aléatoires afin d'avoir un nombre représentatif d'instances de tests. Nous définissons ensuite différentes transformations possibles pour les quarts. Il est possible d'autoriser plus ou moins de transformations et nous faisons donc différentes propositions avec un nombre différent de quarts proposés afin de tester laquelle sera la plus satisfaisante. Enfin, nous définissons une structure de coût adaptée à la ré-optimisation qui pénalise la modification de l'horaire

L'analyse des solutions obtenues pour ces différentes propositions nous permet d'en isoler deux particulièrement intéressantes à la fois de par leur coût et de par leur temps de résolution. Il est cependant clair que l'utilisateur doit faire un arbitrage entre le coût de la solution obtenue et le nombre de modifications que subit l'horaire. Il y a donc une grosse part de choix de la part de l'utilisateur que nous ne pouvons pas effectuer.

ABSTRACT

Personnel shift scheduling is a great operations research problem. The user is willing to minimize its workforce costs while answering to customers' demand and being sure to enforce the regulations and collective agreements. However, the schedule being planned one month ahead, the real demand is difficult to forecast precisely. It is thus possible that the schedule, while responding to the forecast demand, doesn't respond to the real demand. If one can predict few days earlier that the real demand won't be the same than the forecast one, rescheduling is possible.

This master's thesis is responding to this shift rescheduling problem. We deal with a retail environment that is to say non-continuous work. This is an integer program model which permits an exact resolution of the problem.

The first step of the thesis is to generate representative demand scenarios. Some random factors are introduced in this generation. This allows to have enough scenarios to claim that our results are representative. The second step is to define the different shift transformations' possibilities. Then, each transformation can be used or not in the shift proposal. We thus enumerate different propositions, each having different characteristics. We also redefine the cost structure so that it is adapted to rescheduling and penalizes schedule's modifications.

Analysing the different solutions derived from the different propositions puts forward two of these propositions whose cost and computational time are satisfactory. However, the main conclusion is that the user have to make an arbitration between the cost of the final solution and the number of schedule modifications and choices of parameters are very important decisions to make.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

La problématique de planification d'horaires de personnel se retrouve dans de nombreuses entreprises. Que ce soit dans le domaine de la vente au détail, du transport (aérien, ferroviaire, transport en commun dans les villes), de la santé (hôpitaux, ambulances) ou de la production industrielle, les entreprises font face à la gestion d'un grand nombre d'employés avec des qualifications variées. Il faut alors produire un horaire adapté qui répond à la demande et qui respecte les contraintes imposées par la législation et les conventions collectives. Cette problématique représente un enjeu d'autant plus important pour les entreprises que les dépenses liées à la masse salariale représentent souvent la majeure partie de leur coût de fonctionnement. Dans la suite du document, nous prendrons la vente au détail comme exemple pour illustrer nos propos.

1.1 Contexte de l'étude

La planification d'horaires de personnel est une tâche ardue. En effet, l'entreprise cherche à minimiser le coût de fonctionnement lié à la main d'œuvre tout en prenant en compte les éléments suivants :

- **La satisfaction de la demande.** Face à une clientèle, s'il y a trop peu d'employés, ils ne pourront pas répondre à la demande en un temps raisonnable et les clients seront mécontents. L'entreprise risque alors de perdre sa clientèle à court terme si les personnes ne souhaitent pas attendre d'être servies et s'en vont sans rien acheter mais aussi à moyen et long terme si les clients mécontents décident de changer de fournisseur. Si il y a trop d'employés, l'entreprise aurait pu en mobiliser moins tout en gardant sa qualité de service. Dans les deux cas, l'entreprise perd de l'argent. Le nombre de clients à servir pouvant se ramener au nombre d'employés nécessaires pour le service, la demande sera exprimée en nombre d'employés par la suite.
- **La qualification des employés.** Dans un contexte où plusieurs activités sont effectuées, chaque employé est qualifié ou non pour chacune des activités. Une activité ne peut pas être assignée à un employé qui n'est pas qualifié pour l'effectuer.
- **Les contraintes de travail.** Les lois en vigueur ainsi que les conventions collectives signées avec les syndicats imposent différentes contraintes liées à l'horaire des employés. Parmi ces contraintes, nous retrouvons de manière courante : durée minimale et maximale de quart, temps de repos minimal entre deux quarts, temps de travail maximal dans la semaine, nombre de jours de repos minimal dans la semaine.

— **Les préférences des employés.** Dans la limite du possible, les entreprises cherchent aussi à satisfaire les préférences de leurs employés. Par exemple, si un employé A préfère travailler le matin et un employé B le soir, à qualifications égales, il serait dommage de faire travailler B le matin et A le soir. La satisfaction de ces préférences permet une meilleure ambiance dans l'entreprise ainsi qu'une meilleure productivité et est donc souhaitable.

Face à un grand nombre d'employés, il est impossible pour un gestionnaire de générer un horaire manuellement en respectant l'ensemble de ces contraintes et il faut donc faire appel à un logiciel. La volonté de minimisation de coûts oriente vers un logiciel d'optimisation. De nombreuses recherches sont ainsi effectuées en recherche opérationnelle pour la génération d'horaires de personnel performants.

Générer un horaire de personnel suppose également d'être capable d'anticiper la demande à satisfaire. Une demande s'exprime généralement, dans le cas du commerce de détail, en nombre d'employés nécessaires par période de 15 minutes. Cette demande peut dépendre d'énormément de critères non maîtrisés. La météo, l'organisation d'une vente ou un évènement important diffusé à la télévision en sont des exemples mais il en existe beaucoup d'autres. Si la prévision de la demande est mal estimée, la demande réelle ne sera pas satisfaite par l'horaire généré et les coûts pour l'entreprise seront plus élevés que ce qu'ils auraient pu être.

Parfois, le gestionnaire peut se rendre compte plus tard que la demande réelle sera différente de celle prévue au moment de générer l'horaire. Par exemple, une mauvaise météo non prévue en plein été peut diminuer la demande de la journée considérée. De même, si un ou plusieurs employés sont absents de manière imprévue, il faut les remplacer, ce qui peut aussi être considéré comme une augmentation de la demande à satisfaire. Il faut alors modifier l'horaire prévu à moindre coût tout en respectant l'ensemble des contraintes citées plus haut. Une nouvelle fois, l'utilisation d'un logiciel pourra aider grandement le gestionnaire.

1.2 Motivations

Plusieurs entreprises proposent actuellement des logiciels de gestion d'horaires de personnel travaillant sur des quarts. Les logiciels font également la mise à jour d'horaires déjà générés lors d'une modification de la demande. Cependant, sur une période donnée fixe, cette mise à jour est effectuée de manière séquentielle. Par exemple, si la modification de la demande provient d'absences multiples de salariés, la mise à jour est gérée salarié par salarié et n'est donc pas optimale. Le problème de mise à jour globale n'a, à notre connaissance, jamais été

traité. Ce projet est donc motivé par la volonté de traiter ce problème dans le but d'améliorer les logiciels existants. Nous utiliserons donc un logiciel déjà en service comme base de notre étude.

1.3 Environnement de l'étude

Il y a de nombreuses variantes dans un problème d'horaires de personnel. Il convient donc de préciser l'environnement dans lequel on se place.

On peut considérer un environnement **non continu** où on travaille à la journée avec des heures fixes de fermeture (vente au détail, conducteurs de métro) ou un environnement **continu** où le travail se fait 24h/24 (hôpitaux, usines). Comme nous souhaitons avant tout étudier le cas de la vente au détail, nous nous plaçons dans le cas non continu. Dans la majeure partie des cas, la période de mise à jour s'étendra alors sur une semaine car c'est souvent la période maximale sur laquelle il y des restrictions (temps de travail maximal notamment). Cependant, notre travail pourrait s'étendre à des contextes non continu et se généraliser à d'autres cas que celui de la vente au détail (contrôleurs aériens ou centres d'appel par exemple).

Dans un même contexte, il y a souvent plusieurs activités à effectuer. Par exemple, dans un magasin, il faut entre autres des employés à la caisse, au service à la clientèle et à l'inventaire. Dans un même quart, un employé peut donc effectuer plusieurs activités. Pour simplifier le problème, on peut considérer que les employés ne font qu'une activité par quart. Dans notre cas, nous utiliserons cette simplification.

Dans un problème d'horaire de personnel, il y a également la question des pauses. En effet, de manière générale, le placement des pauses à l'intérieur d'un quart est flexible. L'exploitation de ces flexibilités permet d'obtenir de meilleures solutions mais amène une plus grande difficulté de gestion. De nombreux travaux sur les horaires de personnel réfléchissent donc à la manière d'inclure le placement des pauses dans la génération de l'horaire. Pour l'instant, les pauses sont encore placées dans une deuxième étape et nous ne nous préoccuperons pas du placement des pauses dans la ré-optimisation de l'horaire. Pour aller plus loin, il faudrait aussi prendre en compte cet aspect dans la mise à jour.

Dans un environnement de vente au détail, les entreprises sont souvent divisées en plusieurs départements où les tâches sont différentes. Chaque employé est alors assigné à un département précis. Cependant, au moment de la génération d'horaires, il se peut que des transferts d'employés entre départements soient permis. Cela permet une plus grande flexibilité et donc de meilleures solutions. Dans notre cas de mise à jour de l'horaire, nous nous occupons d'un

unique département et ne permettons pas les transferts. La présence de quarts anonymes, comme expliqué ci-dessous, permet toutefois de ne pas interdire totalement les transferts mais de les considérer dans un second temps. Cela ne fera cependant pas partie de notre problématique.

Lors de la confection de l'horaire, il est rare de pouvoir couvrir de manière exacte en tout temps la demande et il se peut que sur certaines périodes courtes, il y ait trop (resp. trop peu) d'employés par rapport à la demande. On dit alors qu'on est en situation de **sur-couverture** (resp. **sous-couverture**). Dans un programme d'optimisation, on peut soit interdire ce genre de situation soit le pénaliser pour que ça arrive le moins possible. Dans notre cas, nous choisissons d'interdire la sous-couverture et de pénaliser la sur-couverture.

Le logiciel utilisé attribue à chaque employé un horaire et renvoie également une liste de quarts anonymes. Les quarts anonymes sont des quarts qui ne sont attribués à aucun employé mais qui sont pris en compte comme des quarts travaillés pour répondre à la demande. Le gestionnaire ou un autre logiciel d'aide à la décision trouve ensuite qui pourra travailler pendant ces quarts. On peut ainsi faire appel à des travailleurs à temps partiel embauchés pour l'occasion ou à des employés d'un autre département qui connait une demande moins forte et qui a donc des employés à disposition. Le fait de produire des quarts anonymes permet la présence d'une solution réalisable dans tous les cas même si la sous-couverture est interdite.

1.4 Problématique et plan du mémoire

Concernant le problème, on a un fichier de données qui représente un horaire initial et qui comprend les informations suivantes sur le problème :

- La période d'étude du problème de durée typique d'une semaine.
- Les informations sur les activités à savoir la valeur de la demande initiale pour chacune d'elles sur la semaine étudiée.
- Les informations sur les employés qui comprennent l'horaire planifié pour la semaine étudiée mais aussi les disponibilités, les qualifications et les contraintes de planification (durée minimale et maximale des quarts et durée de repos minimale entre deux quarts).
- Les informations sur les quarts anonymes c'est-à-dire la liste des quarts planifiés pour la semaine et les contraintes de durée minimale et maximale pour ces quarts anonymes.

Nous avons affaire à une perturbation de la demande pour certaines activités et nous devons mettre à jour l'horaire pour répondre le mieux possible à cette nouvelle demande. Nous souhaitons avoir le moins de modifications possibles dans l'horaire et un temps de résolution

faible. Nous procédons donc à une ré-optimisation qui prend en compte l'horaire initial et qui cherche à le modifier le moins possible.

Une idée pour répondre à ce problème est d'utiliser une méthode de résolution exacte. À partir des quarts déjà planifiés dans l'horaire initial, nous énumérons des quarts transformés selon des règles de modifications acceptables et les proposons comme variables dans un modèle de programmation en nombres entiers afin d'en retirer les plus judicieux.

Afin de pouvoir tester les méthodes proposées, il faut évidemment avoir accès à des scénarios de demande modifiée. Ces scénarios doivent être construits et c'est également un travail compris dans ce projet de maîtrise.

Ainsi, après une revue de littérature dans le chapitre 2 permettant de faire le point sur les travaux déjà effectués autour des horaires de personnel et de leur mise à jour ainsi que sur la construction de scénarios de demande, nous présenterons dans le chapitre 3 le modèle de ré-optimisation ainsi que la méthode de résolution utilisée. Dans le chapitre 4, nous expliquerons les différentes propositions que nous avons faites au niveau des quarts générés pour tester notre modèle. Enfin, nous analyserons les résultats de ces tests dans le chapitre 5 avant de conclure au chapitre 6.

CHAPITRE 2 REVUE DE LITTÉRATURE

Dans ce chapitre, nous présentons une revue de littérature des travaux déjà effectués sur le problème. Nous commençons par développer les recherches effectuées sur la génération d'horaires de personnel en général (section 2.1), puis nous nous intéressons plus particulièrement à la mise à jour de ces horaires (section 2.2). Enfin, dans la section 2.3, nous décrivons les différentes propositions de la littérature pour la génération de scénarios de demande.

2.1 Génération des horaires de personnel

La génération d'horaires de personnel est traitée en recherche opérationnelle depuis les années 50. En effet, c'est Dantzig (1954) qui l'évoqua la première fois. Il répondait à un article d'Edie (1954) qui voulait optimiser le fonctionnement des péages de New York et notamment le délai d'attente des véhicules qui s'y présentaient. Dans son article, Edie proposait une méthode heuristique. Dantzig, lui, a proposé de résoudre un programme linéaire en nombres entiers qui correspond en fait à un problème de recouvrement. Le problème de recouvrement a l'avantage d'être facile à formuler et de réduire considérablement le nombre de contraintes puisque celles-ci sont prises en compte lors de l'énumération des quarts. Il a cependant l'inconvénient majeur de contenir un nombre de variables énorme dès que l'on veut amener un peu de flexibilité au problème. Cela fonctionne ici car Dantzig est confronté à un cas facile où il n'y a qu'une activité à effectuer (caissier au péage) et il ne prend pas en compte la possibilité de placer des pauses de manière flexible à l'intérieur du quart.

Par la suite, de nombreuses recherches ont été menées pour améliorer la proposition de Dantzig (1954) et permettre plus de flexibilité au modèle. Le problème du placement des pauses et le cas multi-activités ont notamment fait l'objet de beaucoup de travaux. On distingue également les articles traitant de problèmes continus de ceux traitant de problèmes non continus. Ernst et al. (2004) propose une division du problème en 6 étapes de résolution et fait une revue de littérature des travaux traitant de chacune de ces étapes. Il propose aussi une liste de travaux pour différents domaines d'applications. Plus récemment, Van den Bergh et al. (2013) a proposé une revue de littérature en divisant les travaux en fonction des caractéristiques des problèmes résolus.

Concernant le placement des pauses, les chercheurs ont dans un premier temps cherché à le rendre plus flexible. Ainsi, Segal (1974) a proposé une approche en deux parties utilisant des modèles de flot pour résoudre le problème appliqué aux centres d'appels. Il travaille dans

un contexte non continu mono-activité. La première partie consiste à déterminer un premier horaire avec les pauses fixées au milieu du quart. Dans un deuxième temps, Segal autorise le déplacement des pauses en gardant fixes les horaires de début et de fin du quart. Ceci n'est évidemment pas optimal mais permet d'arriver à une meilleure solution que Dantzig (1954).

Bechtold et Jacobs (1990) ont formulé un modèle implicite qui donne un résultat exact dans un environnement mono-activité non continu. Ils permettent cependant de placer uniquement une pause de durée fixe à l'intérieur d'un quart. Cela n'est pas toujours réaliste. Il n'est en effet pas rare de devoir avoir plusieurs pauses à l'intérieur d'un même quart par exemple. Aykin (1996) résout ce problème en permettant plusieurs pauses qui doivent chacune être placée dans une certaine fenêtre de temps. Il étend en fait le modèle de Dantzig (1954). On est toujours dans un environnement non continu et mono-activité. Rekik et al. (2010) proposent des extensions de ces deux modèles qui permettent de garder les avantages de chacun d'eux. Ils ajoutent également le concept de pauses décomposables qui permet d'effectuer une pause en deux temps. Par exemple, si il faut effectuer une pause de 30 minutes au bout de 4 heures de travail, on peut effectuer deux pauses de 15 minutes réparties parmi les 4 heures considérées.

Concernant l'environnement multi-activités, Lequy et al. (2012) distinguent deux catégories pour le travail des employés : le travail facilement interruptible appelé activité et le travail non interruptible appelé tâche. Tenir la caisse dans un supermarché ou répondre au téléphone dans un centre d'appel sont ainsi des exemples d'activités alors qu'effectuer un vol en tant que membre d'équipage d'une compagnie aérienne ou faire une opération chirurgicale en tant que médecin sont des tâches. Dans le commerce au détail, le travail est principalement interruptible. De plus, on ne traite pas les deux catégories de la même manière notamment au niveau des contraintes à prendre en compte. Nous ne nous intéresserons donc dans un premier temps qu'au travail interruptible.

Concernant cette catégorie de travail, les recherches sont relativement récentes. Elles ont commencé avec les travaux de Vatri (2001) et d'Omari (2002) qui appliquaient l'affectation d'activités au domaine du contrôle aérien. Ce domaine ressemble en plusieurs points au commerce de détail puisque les activités y sont souvent facilement interruptibles. Omari (2002) propose ainsi une heuristique utilisant un réseau multi-flots avec variables de ressources et contraintes supplémentaires pour placer les activités à l'intérieur de quarts déjà construits et dans lesquels les pauses sont déjà fixées. Vatri (2001) étend le modèle d'Omari (2002) afin d'effectuer dans une même étape la création des quarts et le placement des activités et des pauses à l'intérieur de ceux-ci.

Bouchard (2008) propose quant à lui une méthode exacte de programmation linéaire en

nombre entiers dont le modèle est basé sur un problème de flot dans un réseau. Sa méthode abandonne cependant l'idée d'agréger la génération des quarts et l'affectation d'activités. Le problème qu'il résout est donc celui d'Omari (2002) à savoir l'affectation d'activités dans des quarts déjà créés.

Demassey et al. (2005) proposent un modèle explicite dans le cas multi-activité où les quarts ne sont pas fixés à l'avance et où le problème des pauses doit aussi être géré. Ils génèrent les horaires réalisables avec une méthode de génération de colonnes. Afin de ne générer que les horaires dont le coût réduit est négatif, une contrainte appelée *cost-regular global constraint* est introduite. De leur côté, Côté et al. (2007) utilisent la contrainte *regular* habituellement utilisée en programmation par contraintes et l'intègrent à un modèle de programmation en nombres entiers (MIP). Ils comparent ensuite les résultats entre un MIP classique et le MIP avec intégration de la contrainte *regular* supplémentaire et obtiennent de bons résultats pour cette dernière méthode.

Quimper et Rousseau (2010) utilisent les langages formels pour résoudre ce problème. Cela leur permet d'obtenir des graphes dans lesquels la recherche à grand voisinage est facilement utilisable. Côté et al. (2012) proposent quant à eux un algorithme branch-and-price. Ils utilisent les grammaires non contextuelles pour modéliser plus facilement les contraintes et ainsi réussissent à résoudre des problèmes de grande taille. Côté et al. (2011) proposent un modèle implicite. Une nouvelle fois, ils utilisent les grammaires non contextuelles pour modéliser les contraintes et obtiennent de très bons résultats. Enfin, très récemment, Dahmen et Rekik (2015) ont proposé de résoudre le modèle multi-activité sur plusieurs jours ce qui n'était pas le cas jusqu'à maintenant. Elles proposent une heuristique hybride combinant la recherche tabou et la méthode de séparation et évaluation.

2.2 Mise à jour des horaires

Concernant la mise à jour des horaires, à ma connaissance, aucune recherche n'a été effectuée sur le sujet dans le domaine exact dans lequel nous travaillons à savoir commerce au détail ou assimilé. Cependant, de nombreux travaux sont effectués dans le domaine des horaires d'infirmiers.

Ainsi, Moz et Pato (2003) traitent le problème dans le cas d'absence de personnel. Ils proposent un modèle en nombres entiers basé sur un problème de flot avec contraintes supplémentaires. Cependant, les tests informatiques ne sont effectués qu'avec une heuristique car le temps de résolution du modèle exact serait trop long. L'objectif lors de la mise à jour est de minimiser le nombre de changements imposés aux infirmières. Ici, les quarts sont fixés et

seul l'assignement des infirmières aux quarts change lors de la mise à jour. Il n'y a pas de possibilité d'allongement ou de réduction des quarts déjà fixés.

Le domaine infirmier diffère cependant du domaine du commerce de détail en plusieurs points. Premièrement, c'est un environnement continu où le travail doit être effectué 24h/24. De plus, le coût économique des solutions n'est pas toujours l'objectif à minimiser. Dans un tel domaine, il faut aussi et surtout faire attention à la qualité du travail fourni. Le temps supplémentaire par exemple est rarement admis car les infirmiers risqueraient d'être trop fatigués et de faire des erreurs professionnelles. On privilégie donc les solutions de bonne « qualité » aux solutions les moins coûteuses. L'objectif est ici très différent du domaine du commerce au détail où on peut se permettre beaucoup plus de flexibilité au niveau de l'horaire. Nous souhaitons ainsi orienter notre travail vers la modification des quarts planifiés et ne souhaitons donc pas fixer ces quarts comme proposé par Moz et Pato (2003).

2.3 Génération des scénarios de demande

Déjà en 1954, Edie (1954) se posait des questions sur l'évaluation de la demande afin de pouvoir optimiser. Il est évident qu'il ne sert à rien d'optimiser un horaire de personnel si on ne connait pas la demande à satisfaire. De notre côté, nous avons besoin d'avoir accès à différents scénarios de demande afin de pouvoir tester notre méthode. Ce besoin se retrouve également si on cherche à résoudre le problème stochastique d'horaires de personnel.

L'un des problèmes majeurs pour l'accès à des scénarios de demande est le manque de données. Ainsi, dans un magasin, il est rare que le gérant note le jour même la demande réelle et qu'on puisse ainsi observer la différence entre la demande prévue initialement et la demande réelle. De plus, afin d'avoir des résultats significatifs, nous avons besoin de beaucoup plus d'un scénario sur une même période. Ainsi, nous devons générer des scénarios de demande de manière artificielle mais qui soient tout de même réalistes. Plusieurs études ont donc déjà été effectuées pour générer des scénarios de demande satisfaisants.

Legrain (2011) propose un modèle où la demande peut être perturbée à plusieurs échelles. Quatre durées de perturbations sont définies : une journée, quatre heures, une heure et 15 minutes. L'ensemble de ces perturbations peuvent se superposer les unes aux autres. Chaque perturbation suit une loi log-normale dont les paramètres peuvent être calculés à partir des données historiques. Comme on introduit une loi aléatoire et une fois les paramètres de cette loi déterminés, on peut générer autant de scénarios qu'on le souhaite.

Proulx (2014) s'est lui procuré les données de demande prévue et de demande réelle sur 28 jours. Il produit ensuite les scénarios de demande à partir de ces données. Cependant, les

corrélations interviennent ici sur une journée et pas une semaine alors que nous souhaitons pouvoir introduire des corrélations sur plusieurs jours. De plus, le fait d'avoir besoin de données historiques pour utiliser cette méthode ne nous satisfait pas car nous n'y avons pas accès.

Ainsi, nous nous inspirerons plus tard de la méthode de Legrain (2011) qui est plus adaptable dans notre cas.

CHAPITRE 3 MODÈLE ET MÉTHODE DE RÉSOLUTION

Dans ce chapitre, nous présentons le modèle utilisé pour résoudre notre problème ainsi que la méthode utilisée pour le résoudre. Nous commençons donc par énoncer le modèle (section 3.1) puis nous expliquons en détail la manière dont nous générons les quarts utilisés dans le modèle (section 3.2) ainsi que les pénalités de modification qui leur sont associées (section 3.3). Enfin, nous décrivons la méthode de résolution du modèle (section 3.4).

3.1 Modèle

3.1.1 Idée générale du modèle

Nous proposons un modèle de programmation en nombres entiers qui vise à minimiser les coûts de main d'œuvre pour l'entreprise tout en satisfaisant la demande et les différentes contraintes de travail des employés. Ce modèle est un modèle explicite où chaque quart susceptible d'être utilisé est énuméré. Ainsi, à partir de l'horaire initialement prévu, nous générons un certain nombre de quarts transformés que nous proposons pour la ré-optimisation. La manière de générer ces quarts est expliquée dans la section 3.2.

Nous pouvons également choisir de fixer une partie de l'horaire. On a alors un certains nombre de **quarts fixés** qui ne peuvent pas être modifiés mais doivent être pris en compte pour le calcul du coût, de la couverture de la demande et pour certaines contraintes de travail. Les quarts fixés peuvent être de deux natures :

— **Horaire non modifiable.** Il est possible que la modification de la demande soit annoncée en milieu de semaine alors que le début de la semaine a déjà été travaillé. Évidemment, nous ne pouvons alors plus modifier l'horaire du début de la semaine et celui-ci est considéré comme fixé.

— **Choix pour la ré-optimisation.** Nous pouvons choisir pour accélérer la résolution du problème de fixer une partie de l'horaire car nous savons, après analyse, que cette partie ne sera pas modifiée dans la solution optimale.

On discrétise le temps en périodes d'une durée de 15 minutes. Il y a donc 96 périodes par jour. Les durées seront par la suite exprimées en nombre de périodes. Aucune durée considérée ne peut donc être inférieure à 15 minutes. La demande est également exprimée selon cette discrétisation : pour chaque période de 15 minutes, on a une valeur de demande constante.

Nous expliquons ci-dessous la structure de coûts utilisée pour le modèle ainsi que la manière

dont nous gérons les contraintes.

Structure de coûts et modélisation des fonctions linéaires par morceaux

Deux types de coûts sont à prendre en compte :

- **Les coûts pour l'optimisation**. Ce sont les coûts déjà utilisés par le logiciel d'optimisation déjà en service. Ils comprennent les coûts de main d'œuvre mais aussi les pénalités de quarts anonymes et de sur-couverture. Ces coûts ne sont pas forcément les coûts réels payés par l'entreprise mais correspondent à une structure pénalisant les situations que nous ne souhaitons pas retrouver dans l'horaire. Le détail de cette structure de coûts est présenté en annexe A.
- **Les coûts pour la ré-optimisation**. Nous souhaitons modifier l'horaire initial le moins possible. Nous ajoutons donc à chaque quart un coût qui pénalise la modification que subit l'horaire si le quart en question est choisi lors de la ré-optimisation. La méthode de définition de ces coûts est détaillée à la section 3.3.

On a une fonction linéaire par morceaux pour les coûts de main d'œuvre, de sur-couverture et de quarts anonymes. Explicitons la manière dont nous modélisons ces coûts à l'aide d'un cas générique.

On a un certain nombre d'objets $N \in \mathbb{N}$. Le coût de ces objets est croissant en fonction de leur nombre. Ainsi, nous avons un tableau de coûts c de taille L tel que le premier objet coûte c_1, le deuxième c_2, le $k^{ème}$ c_k. Si N est supérieur à L, l'ensemble des objets au-dessus de ce nombre coûtent chacun c_L. Ainsi, le coût des N objets est égal à :

$$C(N) = \sum_{k=1}^{min(N,L-1)} c_k + max(0, N+1-L)c_L \tag{3.1}$$

Si on veut minimiser ce coût on peut introduire des variables n_k pour $1 \leq k < L$ qui vaudront 1 si $N \geq k$ et 0 sinon. On introduit également la variable n_L qui vaut $max(0, N+1-L)$. On pose alors le programme en nombre entiers :

$$\begin{aligned} \min\ z(n) &= \sum_{1 \leq k \leq L} n_k c_k \\ \text{sujet à :}\ N &= \sum_{1 \leq k \leq L} n_k \\ & n_k \in \{0,1\}\ \forall k \in [1, L[\\ & n_L \in \mathbb{N} \end{aligned} \tag{3.2}$$

Proposition 3.1.1 *La solution optimale* $(\tilde{n}_k)_{k\in[1,L]}$ *de ce programme en nombre entiers est égale à :*

$$\tilde{n}_k = \begin{cases} 1 & si\ N \geq k\ et\ k < L \\ 0 & si\ N < k < L \\ max(0, N+1-L) & si\ k = L \end{cases}$$

et la valeur de la solution est $C(N)$*.*

Preuve Soit une solution réalisable n_k, $k \in [1, L]$.

— Si il existe $k_1 < k_2 < L$ tels que $n_{k_1} = 0$ et $n_{k_2} = 1$, la solution n'_k définie par :

$$n'_k = \begin{cases} n_k & \text{pour} \quad k \neq k_1 \text{ et } k \neq k_2 \\ 1 & \text{pour} \quad k = k_1 \\ 0 & \text{pour} \quad k = k_2 \end{cases}$$

est réalisable et a un coût moins élevé donc la solution n_k n'est pas optimale.

— On montre de la même manière qu'on ne peut pas avoir $k_1 < L$ tel que $n_{k_1} = 0$ et $n_L > max(0, N+1-L)$.

Ainsi :

— Si $N < L$, il existe nécessairement k^* tel que la solution optimale vérifie $\tilde{n}_k = 1$ pour $k \leq k^*$ et 0 sinon. Comme $N = \sum_{1\leq k\leq L} n_k$ on a nécessairement $k^* = N$.

— Si $N \geq L$, on a nécessairement $\tilde{n}_k = 1$ pour $k < L$ et comme $N = \sum_{1\leq k\leq L} n_k$, on obtient $\tilde{n}_L = N + 1 - L > 0$.

On obtient donc bien la solution optimale voulue. La valeur de la solution optimale est alors :

$$\begin{aligned} z(\tilde{n}) &= \sum_{k=1}^{L} n_k c_k \\ &= \begin{cases} \sum_{k=1}^{N} c_k & \text{si} \quad N < L \\ \sum_{k=1}^{L-1} c_k + (N+1-L)c_L & \text{si} \quad N \geq L \end{cases} \\ &= \sum_{k=1}^{min(N,L-1)} c_k + max(0, N+1-L)c_L \\ &= C(N) \end{aligned}$$

Ainsi, le programme 3.2 est équivalent à minimiser $C(N)$. Nous utiliserons cette modélisation par la suite.

Gestion des contraintes

En dehors de la couverture de la demande, nous prenons en compte différentes contraintes de travail :

1. **Qualifications des employés**. On peut assigner un quart à un employé seulement si celui-ci est qualifié pour l'activité effectuée pendant ce quart.
2. **Disponibilité des employés**. La disponibilité de chaque employé est précisée pour chacune des périodes. Nous devons la respecter.
3. **Durée de quart minimale et maximale**. Pour chaque employé et chaque jour, des durées de travail minimale et maximale sont spécifiées.
4. **Délai minimum d'avertissement**. Un certain délai est requis pour informer l'employé que son horaire sera modifié. Le délai minimum d'avertissement n'est pas précisé dans les données fournies et on le choisit nul par défaut.
5. **Temps de repos minimum entre les quarts**. Entre deux quarts, un employé doit se reposer pendant une certaine durée minimale. Cette durée peut être différente pour chaque employé et pour chaque jour.

Les contraintes 1 , 2, 3 et 4 seront internalisées lors de la génération des quarts : nous ne générons pas de quarts qui pourraient mettre en défaut l'une de ces quatre contraintes.

La cinquième contrainte de temps de repos minimal doit par contre être présente dans les contraintes du problème en nombres entiers que nous formulons (ce sont les contraintes (3.6)). Lorsqu'un des quarts concernés est un quart fixé, nous pouvons internaliser cette contrainte en ne générant pas de quart dont le début ou la fin est trop proche du quart fixé. Cela nous permet de ne pas vérifier la contrainte de temps de repos minimal lorsqu'un quart fixé est impliqué.

3.1.2 Modèle mathématique

Le modèle mathématique que nous proposons fait appel à la notation suivante :

Ensembles :

- A : ensemble des activités.
- E : ensemble des employés.
- H : ensemble des jours étudiés.
- P : ensemble des périodes étudiées.
- P^M : ensemble des périodes où l'horaire est modifiable.
- J_e^F : ensemble des quarts fixés pour l'employé $e \in E$.

— J^{FY} : ensemble des quarts anonymes fixés.
— J_e^P : ensemble des quarts proposés pour l'employé $e \in E$.
— J^{PY} : ensemble des quarts anonymes proposés pour la ré-optimisation.
— J^P : ensemble des quarts proposés. On a donc $J^P = (\bigcup_{e \in E} J_e^P) \cup J^{PY}$.
— J : ensemble des quarts du problème (fixés et proposés) : $J = (\bigcup_{e \in E} J_e^F) \cup J^{FY} \cup J^P$

Paramètres :

— d'_{ap} : demande modifiée pour la période $p \in P$ et l'activité $a \in A$
— t_j : durée en périodes du quart $j \in J$
— r_{ep} : nombre de périodes minimum de repos pour l'employé $e \in E$ qui vient de travailler un quart $j \in J_e^P$ s'étant terminé à la période $p \in P^M$.
— A_{jap} : vaut 1 si le quart $j \in J$ couvre l'activité $a \in A$ pendant la période $p \in P$, 0 sinon.
— S_{jh} : vaut 1 si le quart $j \in J$ commence au jour $h \in H$, 0 sinon.
— S_{jp} : vaut 1 si le quart $j \in J$ commence à la période $p \in P$ (p incluse), 0 sinon.
— E_{jp} : vaut 1 si le quart $j \in J$ se termine à la période $p \in P$ (p incluse), 0 sinon.
— c_j : pénalité de modification associée au quart $j \in J_e^P$ proposé pour l'employé $e \in E$.
— $c_e^R h$: coût de repos pour l'employé $e \in E$ le jour $h \in H$.
— c^S : tableau des coûts de main d'œuvre. On notera c_k^S la valeur à l'indice k du tableau.
— l^S : taille du tableau c^S.
— c^O : tableau des coûts de sur-couverture. On notera c_k^O la valeur à l'indice k du tableau.
— l^O : taille du tableau c^O.
— c^Y : tableau des coûts de quarts anonymes. On notera c_k^Y la valeur à l'indice k du tableau.
— l^Y : taille du tableau c^Y.

Variables :

— X_{ej} : vaut 1 si on attribue le quart $j \in J_e^P$ à l'employé $e \in E$, 0 sinon.
— R_{eh} : vaut 1 si l'employé $e \in E$ se repose le jour $h \in H$, 0 sinon.
— X_j : vaut 1 si on choisit le quart anonyme $j \in J^{PY}$ pour le nouvel horaire, 0 sinon.
— T_{ek} ($k \in [1, l^S[$) : vaut le nombre de périodes travaillées par l'employé $e \in E$ dans la semaine étudiée entre $32(k-1)$ et $32k-1$.
— T_{el^S} vaut le nombre de périodes travaillées au-dessus de $32(l^S - 1)$ pour l'employé $e \in E$ dans la semaine étudiée.
— O_{apk} ($k \in [1, l^O[$) : vaut 1 si on a au moins k quarts en sur-couverture pour l'activité $a \in A$ à la période $p \in P$, 0 sinon.
— O_{apl^O} : vaut le nombre de quarts en sur-couverture au-dessus de l^O pour l'activité

$a \in A$ à la période $p \in P$.

— Y_{pk} ($k \in [1, l^Y[$) : vaut 1 si on a au moins k quarts anonymes couvrant la période $p \in P$, 0 sinon.

— Y_{pl^Y} : vaut le nombre de quarts anonymes couvrant la période $p \in P$ au-dessus de l^Y.

Le nombre de variables du modèle est environ égal à $|J^P|$ car l^S, l^O, l^Y et $|H||E|$ sont négligeables devant $|J^P|$. Nous influons ainsi directement sur le nombre de variables et donc sur le temps de résolution en proposant plus ou moins de quarts transformés. Dans notre cas, le nombre de variables varie entre 10000 et 90000 en fonction des propositions comme nous pouvons le voir dans le tableau 4.5.

On écrit alors le modèle de programmation linéaire en nombres entiers :

$$\min \sum_{e\in E}\sum_{k\leq l_e} c_k^S T_{ek} + \sum_{p\in P}(\sum_{a\in A}\sum_{k\leq l^O} c_k^O O_{apk} + \sum_{k\leq l^Y} c_k^Y Y_{pk}) + \sum_{e\in E}(\sum_{j\in J_e^P} c_j X_{ej} + \sum_{h\in H} c_{he}^R R_{eh}) \tag{3.3}$$

$$\text{sujet à : } \sum_{e\in E}(\sum_{j\in J_e^P} A_{jap}X_{je} + \sum_{j\in J_e^F} A_{jap}) + \sum_{j\in J^{PY}} A_{jap}X_j + \sum_{j\in J^{FY}} A_{jap} - \sum_{k\leq l^O} O_{apk} = d'_{ap}$$
$$\forall a \in A,\ \forall p \in P \tag{3.4}$$

$$\sum_{a\in A}(\sum_{j\in J_e^P} S_{jh}X_{je} + \sum_{j\in J_e^F} S_{jh}) + R_{eh} = 1\ \forall h \in H,\ \forall e \in E \tag{3.5}$$

$$\sum_{j\in J_e^P} E_{jp}X_{ej} + \sum_{j\in J_e^P} S_{jp'}X_{ej} \leq 1\ \forall e \in E,\ \forall p \in P^M,\ \forall p < p^{'} \leq p + r_{ep} \tag{3.6}$$

$$\sum_{j\in J_e^P} t_j X_{ej} + \sum_{j\in J_e^F} t_j = \sum_{k\leq l^S} T_{ek}\ \forall e \in E \tag{3.7}$$

$$\sum_{a\in A}(\sum_{j\in J^{PY}} A_{jap}X_j + \sum_{j\in J^{FY}} A_{jap}) = \sum_{k\leq l^Y} Y_{pk}\ \forall p \in P \tag{3.8}$$

$$T_{ek} \leq 32\ \forall k \in [1, l^S[,\ \forall e \in E \tag{3.9}$$

$$X_{je}, X_j \in \{0,1\}\ \forall e \in E,\ \forall j \in J \tag{3.10}$$

$$R_{eh} \in \{0,1\}\ \forall e \in E,\ \forall h \in H \tag{3.11}$$

$$T_{ek} \in \mathbb{N}\ \forall k \in [1, l^S],\ \forall e \in E \tag{3.12}$$

$$O_{apk} \in \{0,1\}\ \forall k \in [1, l^O[,\ O_{apl^O} \in \mathbb{N},\ \forall a \in A,\ \forall p \in P \tag{3.13}$$

$$Y_{pk} \in \{0,1\}\ \forall k \in [1, l^Y[,\ Y_{pl^Y} \in \mathbb{N},\ \forall a \in A,\ \forall p \in P \tag{3.14}$$

La fonction objectif (3.3) représente les coût tels que décrits dans la section 3.1.1. Les deux premiers termes représentent les coûts déjà présents lors de l'optimisation alors que le dernier représente les pénalités associées à la modification de l'horaire.

Les contraintes (3.4) sont les contraintes de couverture de la demande. Nous interdisons la sous-couverture mais autorisons la sur-couverture. Les contraintes (3.5) assurent qu'un repos ou un quart est assigné à chaque employé chaque jour. Les inégalités (3.6) représentent les contraintes de temps de repos minimal qui doit être respecté entre les quarts travaillés. Les contraintes (3.7) et (3.8) sont utiles pour modéliser les fonctions de coûts de la même manière que dans le système (3.2). Elles permettent de décomposer le temps de travail des employés et le nombre de quarts anonymes utilisés pour chaque période. Les variables décomposant la sur-couverture sont directement incluses dans la contrainte (3.4). Enfin, les six derniers groupes de contraintes sont des contraintes d'integrité ou de borne des variables.

Le nombre de contraintes est directement relié aux données du problème : nombre d'employés, nombre d'activités et nombre de périodes étudiées. Nous ne pouvons donc pas influer sur ce nombre.

3.2 Génération de quarts transformés

Nous souhaitons utiliser l'horaire initialement généré pour créer des quarts utilisables lors de la mise à jour. Les quarts que nous proposons dans le modèle sont donc des transformations des quarts initiaux que nous appelons **quarts transformés** ou **quarts proposés**. Dans cette section, nous expliquons la manière de générer les quarts que nous proposons ensuite dans le modèle. Nous dressons une liste de transformations possibles pour les quarts et nous énumérons également les possibilités de proposition de nouveaux quarts pour chaque employé.

3.2.1 Possibilités de transformations

Chaque entreprise ayant ses propres règles et conventions collectives, il est important que le choix des transformations possibles soit le plus large et le plus modulable possible. Afin de pouvoir tester notre méthode, il est important de définir un certain nombre de transformations que nous pourrons utiliser. Nous proposons les trois types de transformations simples suivantes :

1. Modifier l'activité du quart (*jobMod*).
2. Raccourcir le quart d'un côté (*red*).
3. Allonger le quart d'un côté (*ext*).

Nous pouvons aussi penser à combiner deux types de transformations :

— Combiner 1. avec 2. ou 3. On obtient alors un quart allongé ou raccourci d'un côté et dont l'activité est modifiée (*JobAndHour* abrégé en *J&H*).

— Combiner 2. et 3 à condition que les transformations interviennent de deux côtés différents du quarts. On obtient alors un quart décalé. La longueur d'allongement d'un côté n'est tout de même pas nécessairement égale à la longueur de raccourcissement de l'autre côté (*shift*).

Nous ne nous autorisons pas ici à allonger ou raccourcir un quart des deux côtés. Il serait possible de le rajouter si l'entreprise concernée l'autorisait dans ses conventions.

Enfin, il serait possible de combiner les trois transformations pour obtenir un quart décalé et dont l'activité est modifiée. Nous n'utiliserons cependant jamais cette possibilité dans nos propositions.

3.2.2 Propositions de nouveaux quarts

Nous voulons aussi rendre possible l'échange de quarts entre les employés ou le travail un jour initialement prévu pour être un jour de repos. Il y a donc la possibilité de création de nouveaux quarts pour les employés c'est-à-dire la proposition de quarts non directement issus de l'horaire initial de l'employé ainsi que la possibilité de création de quarts anonymes. Nous proposons ainsi :

— La mise à disposition de chaque quart planifié initialement en tant que quart anonyme et pour les autres employés.
— La mise à disposition des sections supprimées lors du raccourcissement en tant que quarts anonymes et pour les autres employés.

3.2.3 Solution réalisable

Il est possible que malgré la proposition de l'ensemble de ces quarts, la solution ne soit pas réalisable sans autorisation de la sous-couverture. Nous pouvons détecter ce problème avant optimisation par une analyse simple des quarts proposés et nous générons dans ce cas des quarts anonymes supplémentaires couvrant les périodes et les activités problématiques. Pour générer ces quarts, nous effectuons les étapes suivantes activité par activité :

1. Calculer pour chaque période le nombre de quarts couvrant la période considérée nécessaires pour rendre la solution réalisable. On obtient un tableau dont les indices correspondent aux périodes.
2. Parcourir le tableau. Dès que l'on trouve une période $p1$ où la valeur est strictement positive, faire :

(a) Chercher le nombre de périodes consécutives pour lesquelles cette valeur est strictement positive. On trouve $p2$ la dernière période considérée.

(b) Sachant qu'un quart anonyme doit avoir une durée comprise entre $minOSLength$ et $maxOSLength$, construire un ou plusieurs quarts couvrant les périodes $p \in [p1, p2]$.

Dans notre cas, la construction des quarts (étape 2.(b)) se fait de la manière suivante :

1. Si $p2 - p1 < minOSLength$, on construit un quart anonyme commençant à $p1$ et de durée $minOSLength$.
2. Si $minOSLength \leq p2 - p1 \leq maxOSLength$, on construit un quart anonyme commençant à $p1$ et de durée $p2 - p1$.
3. Sinon $p2 - p1 > maxOSLength$. On cherche à couvrir l'intervalle $[p1, p2]$ en couvrant le moins possible de périodes en dehors de cet intervalle car cela augmenterait le coût des quarts anonymes dans la solution de manière inutile. Nous posons pour cela les divisions euclidiennes :
   — $p2 - p1 = N_m minOSLength + r_m$ avec $0 \leq r_m < minOSLength$.
   — $p2 - p1 = N_M maxOSLength + r_M$ avec $0 \leq r_M < maxOSLength$.
   On a alors $N_m \geq N_M$ et il faut nécessairement N_m+1 quarts de longueur $minOSLength$ ou $N_M + 1$ quarts de longueur $maxOSLength$ pour couvrir la totalité de l'intervalle.

   — Si $N_m > N_M$, on a :

$$\begin{aligned} r_m &= N_M maxOSLength + r_M - N_m minOSLength \\ &< (N_M + 1) maxOSLength - N_m minOSLength \\ &= (N_M + 1)(maxOSLength - minOSLength) - (N_m - N_M - 1) minOSLength \\ &\leq (N_M + 1)(maxOSLength - minOSLength) \\ &\leq N_m(maxOSLength - minOSLength) \end{aligned}$$

   On peut donc diviser r_m en N_m parties de longueur l_k ($k \in [1, N_m]$) inférieure à $(maxOSLength - minOSLength)$, cette longueur pouvant être nulle. Ainsi, on crée N_m quarts consécutifs de longueur $minOSLength + l_k$. La somme des longueurs de ces quarts vaut bien $p2 - p1$ et les quarts couvrent donc exactement l'intervalle $[p1, p2]$.
   — Si $N_m = N_M$, quelle que soit la longueur des quarts, il en faudra au moins $N_m+1 = N_M+1$ pour couvrir l'intervalle. La longueur cumulée minimum sera atteinte pour

$N_m + 1$ quarts de longueur $minOSLength$ et ces quarts couvrent l'intervalle. Nous créons donc de tels quarts.

3.2.4 Contraintes et paramètres

Bien entendu, il faut respecter les quatre contraintes (1-4) présentées à la section 3.1.1 que nous souhaitons internaliser. Ainsi, si un quart généré ne remplit pas l'une de ces quatre contraintes, nous ne le ferons pas entrer dans la liste des quarts proposés pour l'employé. Notons que le délai minimum d'avertissement présenté dans la contrainte 4. peut prendre des valeurs différentes en fonction du type de transformation, si l'utilisateur le souhaite.

Plusieurs paramètres interviennent ensuite pour pouvoir définir complètement ces transformations. Pour l'extension (resp. la réduction) d'un quart, nous devons définir :

— **Les côtés qui peuvent être allongés (resp. réduits)**. On peut permettre l'allongement (resp. la réduction) du quart seulement d'un côté (au début ou à la fin) ou des deux côtés. L'attribut *side* associé à ce paramètre peut donc valoir *both*, *start* ou *end*.
— **Le temps minimum et maximum** que nous pouvons ajouter (resp. retirer) au quart. Nous devons définir une fenêtre $[minAdd, maxAdd]$ (resp. $[minCut, maxCut]$) représentant les limites de durée que nous pouvons ajouter (resp. retirer) aux quarts.
— **La résolution**. La résolution *res* est telle que nous augmentons (resp. réduisons) les quarts que d'une durée d respectant la formule $d = minAdd + Kres,\ K \in \mathbb{N}$ (resp. $d = minCut + Kres,\ K \in \mathbb{N}$). Ainsi, la différence entre deux durées de transformations proposées est toujours un multiple de res.

Pour la modification d'activité d'un quart, il n'y a pas de paramètre spécifique à préciser.

Les valeurs des paramètres par défaut sont données dans le tableau 3.1.

Tableau 3.1 Paramètres par défaut des transformations.

Type	Nom	Valeur
ext	*side*	*both*
	minAdd (min)	0
	maxAdd (min)	120
	res (min)	15
red	*side*	*both*
	minCut (min)	0
	maxCut (min)	120
	res (min)	15

3.3 Pénalités de modification

Comme nous souhaitons modifier l'horaire initial le moins possible, nous introduisons un coût supplémentaire qui pénalisera la modification. Celui-ci doit être cohérent avec la structure de coûts d'optimisation (annexe A) en fonction des priorités de l'entreprise. L'entreprise doit par exemple répondre aux questions :

- Combien de périodes de sur-couverture est-on prêt à laisser plutôt que de modifier l'horaire ?
- Préfère-t-on garder l'équité de temps de travail entre les employés et modifier l'horaire de deux d'entre eux ou modifier l'horaire d'un seul mais perdre cette équité ?

En fonction des réponses à ces questions, les pénalités seront plus ou moins élevées.

Pour l'employé, le fait que le quart ait été transformé est un inconvénient en soi. Cependant, la longueur de la modification a aussi une influence sur la perception de l'employé. En effet, ce n'est pas le même inconvénient de devoir rester un quart d'heure de plus que prévu ou deux heures. Nous choisissons donc d'avoir deux types de pénalités :

- **Une pénalité fixe (*fix*)** appliquée si le quart est transformé.
- **Une pénalité variable (*var*)** par période appliquée à chaque période effectivement modifiée. Ainsi, un quart décalé d'une heure aura une pénalité variable de $8 * var$ car il y a 4 périodes affectées par la transformation au début du quart et 4 périodes à la fin du quart.

Le fait de proposer des quarts pour un employé sur une journée n'implique pas qu'un de ces quarts sera choisi lors de la ré-optimisation. On doit donc aussi associer un coût au fait d'imposer un repos à un employé qui devait normalement travailler (*rest*). Ainsi, on définit pour chaque jour un coût de repos égal à $fix + t_j var$ où t_j est la durée du quart planifié dans l'horaire de l'employé le jour considéré. Si aucun quart n'est planifié pour l'employé ce jour là, le coût de repos est évidemment nul.

Si l'utilisateur le souhaite, il est possible de complexifier ce système de coût. On peut par exemple instaurer une préférence parmi les transformations. On pourrait alors mettre des pénalités fixes différentes en fonction de la nature de la transformation.

D'autres adaptations de ces pénalités pourraient être imaginées. La seule condition est que la pénalité associée à chaque quart puisse être calculée au moment de la génération du quart et ne dépende donc pas des autres quarts choisis lors de la ré-optimisation. Ainsi, nous pourrions vouloir assurer une équité entre les employés en ne modifiant pas l'horaire de l'un trois jours de suite alors qu'un autre ne subirait aucune modification. Nous ne pouvons pas

assurer ce genre d'équité avec ce modèle de coût. Nous pouvons cependant assurer cette équité d'une semaine sur l'autre : si un employé a subi des modifications de son horaire la semaine précédente, nous pouvons associer des pénalités plus fortes à ses quarts.

Nous devons choisir des valeurs par défaut pour nos propres tests. Nous souhaitons tester nos propositions dans un cadre où les modifications seront préférées à la sur-couverture et aux quarts anonymes. Nous devons donc utiliser des pénalités plus basses que les pénalités de sur-couverture et de quarts anonymes. Par défaut, nous choisissons :

$$fix = 200\$$$
$$var = 75\$/période$$

3.4 Méthode de résolution

Le modèle (3.3)-(3.14) est un modèle de programmation linéaire en nombres entiers de grande taille et nous choisissons d'utiliser un logiciel commercial pour le résoudre. Le solveur choisi est FICO Xpress version 7.4. L'arbre de branchement et les coupes sont gérés par FICO Xpress. Le branchement interne de FICO Xpress fonctionne de la manière suivante : à chaque étape de l'arbre de branchement,

- Trouver la variable X_{ej} la plus proche de 0,5.
- Faire un branchement à partir de cette variable : une branche fixe $X_{ej} = 0$, l'autre fixe $X_{ej} = 1$.

Dans notre cas, avec un nombre limité de quarts proposés, l'arbre de branchement est parcouru en entier. Lors de la génération de l'horaire initial, au contraire, le processus s'arrête lorsque la différence relative entre la borne inférieure et la valeur de la meilleure solution entière trouvée est inférieure à 3%.

Comme les quarts sont déjà tous générés a priori, nous n'utilisons pas la génération de colonnes pourtant utilisée lors de la production de l'horaire initial afin de générer des quarts adaptés au profil de la demande.

CHAPITRE 4 DIFFÉRENTES INSTANCES

Nous avons accès à deux jeux de données. L'un comprend 49 employés, l'autre 95. Chacun de ces jeux donne la valeur de la demande initialement prévue sur une durée d'une semaine. Nous présentons ici les résultats pour le jeu de données de 49 employés. Les tests ont aussi été effectués pour le jeu de données de 95 employés et nous permettent de confirmer que les résultats obtenus sont valables pour plusieurs jeux de données et non pas pour un en particulier.

Afin de valider notre modèle, nous proposons de générer différentes instances de tests. Une instance est la combinaison de deux choses :

- **Un scénario** de demande modifiée. Les scénarios diffèrent par les jours où la demande est modifiée et l'intensité des modifications pour chaque période.
- **Une proposition** de quarts. Les propositions diffèrent par le nombre de quarts proposés et la nature de ces quarts.

Ayant uniquement deux jeux de données sur une seule semaine, il nous est impossible d'avoir accès à un nombre de scénarios de demande conséquent. Nous générons donc des scénarios de demande avec la méthode décrite dans la section 4.1). Nous décrirons ensuite les différentes propositions pour les quarts dans la section 4.2. Enfin, dans la section 4.3, nous présentons les caractéristiques des propositions pour les quarts en fonction des différents scénarios de demande.

4.1 Génération de scénarios de demande modifiée

Afin que nos instances soient représentatives, nous souhaitons générer plusieurs scénarios de demande pour un même jeu de données initial. Un scénario est la composition de :

- **Un patron** décrivant la forme globale que prend la modification de la demande : jours et activités de modification ainsi que jours où l'horaire est modifiable.
- **Une intensité** qui représente la tendance (à la hausse ou à la baisse) et l'importance de la perturbation.
- **Un numéro** pour différencier les scénarios qui sont du même patron et de la même intensité.

On génère 5 scénarios différents pour chaque couple patron-intensité. Il y 4 patrons différents et 4 intensités différentes. On génère donc 80 scénarios de demande différents. Par la suite, si nous faisons référence à un scénario particulier, nous lui donnerons le nom patron-intensité-numéro. Par exemple, le scénario 2-10%-5 est le $5^{ème}$ scénario pour le patron 2 d'intensité

10%.

Nous présentons par la suite les caractéristiques des quatre patrons puis la méthode de génération utilisée et enfin la démarche pour vérifier la cohérence de cette méthode de manière qualitative et quantitative.

4.1.1 Différents patrons

Les différents patrons que nous proposons d'étudier sont les suivants :

- **Patron 1.** On est dimanche, l'horaire pour la semaine suivante a déjà été généré et publié. Une vente (ou une mauvaise météo) est annoncée pour le milieu de la semaine suivante (mercredi et jeudi). Il y aura une modification de la demande pour ces deux jours et la demande va ensuite se reporter à la fin de la semaine : il y aura une augmentation (resp. diminution) de la demande le mercredi et le jeudi mais les clients ne viendront donc pas (resp. viendront) vendredi et samedi et il y aura alors une diminution (resp. augmentation) de la demande par rapport à ce qui a été prévu. L'ensemble des activités voient leur demande modifiée. Pour la mise à jour, on peut modifier l'ensemble de l'horaire de la semaine (du lundi au dimanche).
- **Patron 2.** Il y a une perturbation sur une journée (jeudi) et on est la veille. Seules quelques activités ont leur demande modifiée. Pour la mise à jour, on peut modifier l'horaire de l'ensemble des jours restants (du jeudi au dimanche).
- **Patron 3.** On est dans la même situation que le patron 2 mais toutes les activités voient leur demande modifiée.
- **Patron 4.** Il y a une perturbation sur une journée (dimanche) mais le reste de la semaine a déjà été travaillé. On est la veille et on peut donc modifier l'horaire du dimanche uniquement. Seules quelques activités ont leur demande modifiée.

Avec ces quatre patrons, nous représentons un grand panel de situations possibles avec des modifications de demande plus ou moins importantes et des souplesses pour modifier l'horaire plus ou moins larges.

Pour l'ensemble de ces patrons, nous ajoutons un paramètre d'intensité (int) de la modification qui peut valoir -20%, -10%, 10% ou 20%. L'intensité de modification représente la tendance globale de modification, on devra alors avoir :

$$\sum_{a \in A_{mod}} \sum_{p \in P_{mod}} d'_{ap} = arr\left((1+int)\sum_{a \in A} \sum_{p \in P_{mod}} d_{ap}\right) \tag{4.1}$$

— $arr(x)$ donne l'arrondi à l'entier le plus proche de la valeur x. Si la valeur décimale de x est $0,5$, on arrondi à l'entier supérieur.
— int est la valeur de l'intensité considérée,
— A_{mod} est l'ensemble des activités à modifier,
— P_{mod} est l'ensemble des périodes sur lesquelles la modification de demande est effectuée,
— d_{ap} est la valeur de la demande initiale pour l'activité a à la période p,
— d'_{ap} est la valeur de la demande modifiée pour l'activité a à la période p

Avoir plusieurs intensités permet d'avoir plus de diversité dans nos instances et ainsi des résultats plus représentatifs.

4.1.2 Méthode de génération des scénarios

Présentons maintenant en détails la manière dont nous générons ces scénarios. Notre algorithme de génération se divise en deux étapes. Lors de la première étape, nous modifions la demande sans se soucier des contraintes à respecter. C'est pendant la deuxième étape que nous adaptons cette demande modifiée afin de la rendre entière et de respecter l'équation (4.1).

Modification de la demande

Nous nous inspirons du travail de Legrain (2011) pour la modification de la demande. Legrain (2011) propose d'introduire quatre durées de perturbations de la demande : 1 journée, 4 heures, 1 heure et 15 minutes. De notre côté, nous ne souhaitons pas trop modifier le profil de la demande. Nous utiliserons donc uniquement les perturbations sur une journée et sur 4 heures.

Comme nous souhaitons multiplier la demande par $(1 + int)$ en moyenne, nous commençons donc par multiplier l'ensemble des valeurs de demande pour toutes les activités et toutes les périodes à modifier par cette valeur. Cela représente la perturbation sur une journée (ou plus dans le cas du patron 1). Cette perturbation est donc fixe.

Nous souhaitons également introduire un paramètre aléatoire pour pouvoir générer plusieurs patrons différents avec la même intensité de modification. Nous introduisons alors des perturbations aléatoires de durée 4 heures. Ainsi, pour chaque période, on tire une variable aléatoire selon la loi de Bernoulli de moyenne 0,05 qui indique si la demande des quatre prochaines heures sera modifiée. Si la variable tirée vaut 1, on modifie les 16 prochaines périodes avec une intensité donnée par une autre variable aléatoire suivant une loi log-normale de moyenne

1 et de variance $0,05$. En moyenne, la variation de la demande reste donc proportionnelle à $(1 + int)$.

Cependant, le fait d'introduire ce paramètre aléatoire peut nous éloigner du respect de l'équation (4.1). De plus, les valeurs de d'_{ap} ne sont pas forcément entières. Il faut donc adapter cette demande modifiée pour respecter ces deux contraintes. Nous essayons de plus, lors de cette adaptation et dans la limite du possible, que le profil de la demande modifiée ne s'éloigne pas trop de celui de la demande initiale.

Adaptation de la demande modifiée

Nous commençons par rendre la demande entière en arrondissant chaque valeur à l'entier le plus proche : $d'_{ap} \leftarrow arr(d'_{ap})\ \forall a \in A,\ \forall p \in P_{mod}$.

Nous posons alors :

$$D = arr\left((1 + int)\sum_{a\in A}\sum_{p\in P_{mod}} d_{ap}\right)$$
$$D' = \sum_{a\in A}\sum_{p\in P_{mod}} d'_{ap}$$

où D et D' correspondent aux deux membres de l'équation (4.1). Comparons leur valeur :

- Si $D = D'$, il n'y a pas d'adaptation à faire car l'équation (4.1) est vérifiée et la demande modifiée est entière. On s'arrête.
- Si $D < D'$, trop de valeurs de d' ont été arrondies à l'entier supérieur. Nous dressons donc la liste des valeurs qui ont été arrondies à l'entier supérieur que nous trions par ordre croissant de valeur décimale. En cas d'égalité des valeurs décimales, les d'_{ap} sont triées par ordre croissant de valeur entière. Nous diminuons alors les d'_{ap} de 1 de la manière suivante :

  Nous commençons par prendre le premier élément d'_{ap} de la liste en le retirant de cette liste et on fait $d'_{ap} \leftarrow d'_{ap} - 1$. Si les voisins $d'_{a(p+1)}$ et $d'_{a(p-1)}$ sont dans la liste, nous les considérons en regardant si leur adaptation permettrait de s'approcher du profil de la demande initiale à savoir si $d'_{a(p+1)} - d'_{a(p)}$ serait alors plus proche de $d_{a(p+1)} - d_{a(p)}$ (resp. $d'_{a(p-1)} - d'_{a(p)}$ plus proche de $d_{a(p-1)} - d_{a(p)}$). Si c'est le cas, on adapte le voisin, on l'enlève de la liste et on regarde de nouveaux ses propres voisins à travers une fonction récursive.

  Une fois que la fonction récursive est bloquée, les voisins ne sont pas dans la liste ou leur adaptation ne servirait pas à s'approcher du profil de la demande initiale, on

prend de nouveau le premier élément de la liste.
Bien entendu, à chaque adaptation, on met à jour $D^{'} \leftarrow D^{'} - 1$. Si $D = D^{'}$, alors on arrête l'algorithme.

— Si $D > D^{'}$, trop de valeurs de $d^{'}$ ont été arrondies à l'entier inférieur. Nous dressons de la même manière la liste des valeurs qui ont été arrondies à l'entier inférieur que nous trions par ordre décroissant de valeur décimale. En cas d'égalité des valeurs décimales, les $d^{'}_{ap}$ sont triées par ordre décroissant de valeur entière. Nous augmentons alors les $d^{'}_{ap}$ de 1 avec la même méthode que celle décrite dans le point précédent.

Si à la fin de cette étape, nous avons toujours D différent de $D^{'}$, nous effectuons une deuxième étape d'adaptation de la demande en essayant d'aller dans le sens d'une demande la plus lisse possible. Moins la demande varie et plus on la considère lisse.

Ainsi, pour chaque période, nous attribuons un certain nombre de points qui indiquent si l'adaptation de la demande à cette période irait dans le sens du lissage. Ainsi, si $D < D^{'}$, on somme les deux valeurs suivantes :

— $d^{'}_{ap} - d_{ap}$ qui représente la différence entre la demande modifiée et la demande initiale.
— $(d^{'}_{ap} - d^{'}_{a(p+1)}) + (d^{'}_{ap} - d^{'}_{a(p-1)})$ qui donne la différence entre la valeur de la demande modifiée à la période considérée et les valeurs aux périodes voisines.

Si $D > D^{'}$, on prend l'opposé de cette valeur.

Ainsi, les périodes qui ont le plus grand nombre de points sont celles dont l'adaptation irait le plus dans le sens d'une demande plus lisse et plus proche du profil de la demande initiale. On adapte les valeurs dans l'ordre de la liste tout en mettant à jour le nombre de points à chaque adaptation. On n'enlève pas la valeur de la liste au moment où on l'adapte et on arrivera alors forcément au bout d'un moment à $D = D^{'}$.

On respecte alors l'équation (4.1) et la demande modifiée est entière.

4.1.3 Vérification de la cohérence

Nous souhaitons vérifier que la demande modifiée ainsi obtenue est cohérente c'est-à-dire que le profil de la demande modifiée n'est pas complètement différent de celui de la demande initiale.

Comparaison qualitative

Nous pouvons bien évidemment comparer qualitativement les profils sur un graphique. Un exemple est présenté dans la figure 4.1. On présente ici la demande et la demande modifiée pour une activité sur la journée de jeudi dans le cadre du patron 2 avec l'intensité 20%.

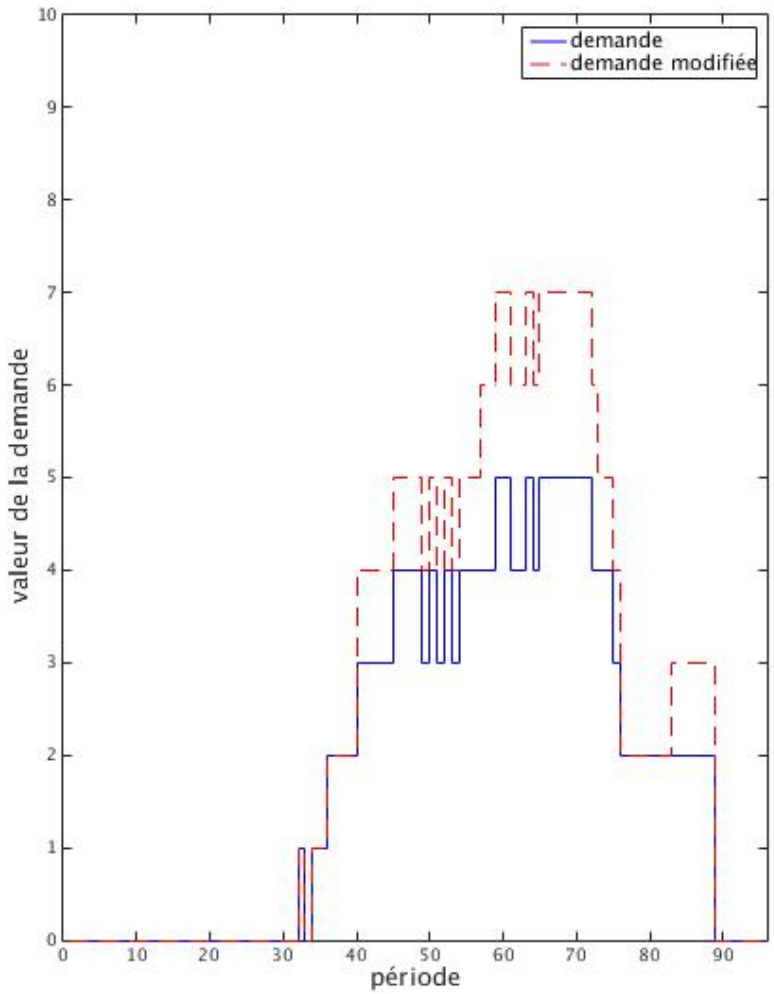

Figure 4.1 Demande et demande modifiée (patron 2, intensité 20%).

On voit ici que le profil de la demande initiale est respecté par la demande modifiée. On constate aussi que les paramètres aléatoires ont introduit tout de même une différence puisque la demande initiale est constante sur les dernières périodes (de 75 à 89) contrairement à la demande modifiée.

Comparaison quantitative

Il est néanmoins difficile et fastidieux de comparer graphique après graphique les demandes pour les 80 scénarios générés. Nous souhaitons donc pouvoir quantifier le rapprochement entre les deux profils afin de pouvoir les comparer plus facilement. Nous calculons alors un indice de lissage I_{da} par activité avec la méthode de calcul suivante :

$$I_{da} = \frac{\sum_{p \in P_{mod}} |d_{ap} - d_{a(p+1)}|}{\sum_{p \in P_{mod}} d_{ap}} \tag{4.2}$$

On peut ensuite calculer l'indice de lissage global I_d en faisant une moyenne de ces indices sur toutes les activités :

$$I_d = \frac{\sum_{a \in A} I_{da}}{|A|} \tag{4.3}$$

On calcule cet indice pour la demande initiale (I_d) et pour l'ensemble des 80 scénarios de demandes modifiées dont on fait la moyenne ($I_{d'}$). On obtient :

$$I_d = 0,1425$$
$$I_{d'} = 0,1431$$

Ces deux indices sont très proches et cette analyse quantitative nous permet donc d'affirmer que notre méthode de génération de demande modifiée est cohérente avec la demande initiale.

4.2 Propositions pour les quarts

Nous souhaitons également produire des instances avec des quarts proposés et fixés de différentes natures afin de voir quelle possibilité donne de meilleurs résultats. Nous énumérons ici les différentes propositions. Comme nous l'avons vu dans le chapitre 3, nous pouvons jouer sur plusieurs paramètres dans la génération des quarts :

- **Les quarts fixés.** Nous pouvons décider de fixer ou non un certain nombre de quarts de l'horaire initial.
- **Les transformations utilisées.** Nous pouvons autoriser ou interdire certains types ou combinaisons de types de transformations des quarts présentées dans la section 3.2.1.
- **Valeur des pénalités.** Les pénalités de modification sont des paramètres libres dont nous pouvons décider la valeur.

— **Utilisation du générateur de quarts du logiciel.** Si nous le souhaitons, nous pouvons décider de ne pas générer les quarts nous-mêmes mais de laisser le générateur de quarts du logiciel le faire. On peut décider d'utiliser cette option pour les quarts anonymes, pour les quarts des employés ou pour les deux.

L'ensemble des propositions et leurs caractéristiques sont résumées dans le tableau 4.1.

On précise dans le tableau :

— Quelles transformations sont autorisées :
  — « Nb » indique le nombre de transformations autorisées par quart (1 ou 2).
  — « Selec. » indique si certains types particuliers de transformations ont été interdits.
  — « Jours mod. » (resp. « Act. mod. ») est coché si seuls les quarts couvrant les jours où la demande est modifiée (resp. couvrant les activités dont la demande est modifiée) peuvent être transformés.
  — « Toutes » (resp. « Aucune ») indique que toute (resp. aucune) transformation est autorisée.
— La valeur des pénalités : par défaut (-), nulles ou multipliées par 5 par rapport aux valeurs par défaut.
— Les activités dont la demande est modifiée : scénarios par défaut (-) ou seulement deux activités perturbées.
— Si on génère des quarts avec le générateur de quarts du logiciel pour les employés (Emp.) ou pour la banque de quarts anonymes (QA).

Tableau 4.1 Résumé des propositions.

Propositions			Transformations autorisées							Pénalités				Act. mod.		Gén. quarts	
			Nb		Selec.	Jours mod.	Act. mod.	Toutes	Aucune	-	Nulles	*5	Fixe double	-	Seul. 2	Emp.	QA
			1	2													
exact								x			x			x		x	x
fixed									x					x			x
*trans*1			x							x				x			
*trans*2	-	-		x						x				x			
		genOS		x						x				x			x
		noPen		x							x			x			
		morePen		x								x		x			
		doublePen		x									x	x			
		dayOfMod		x		x				x				x			
		modJobs		x			x			x				x			
	lessReplaced			x	x						x			x			
*trans*2	*lessJobs*	-		x						x					x		
		modJobs		x			x			x					x		

4.2.1 Les extrêmes

Afin d'avoir une référence de comparaison, nous commençons par tester les choix extrêmes dont le coût de la solution sera respectivement le plus élevé et le plus bas.

Ré-optimisation exacte (*exact*)

La proposition qui aura le coût de solution le plus faible est la ré-optimisation exacte. Elle consiste à utiliser le logiciel pour générer l'horaire adapté à la demande modifiée sans se soucier de l'horaire initial. La génération des quarts pour les employés et des quarts anonymes est faite par le générateur de quarts du logiciel.

On aura ainsi accès à la solution optimale pour la demande considérée qui est la solution de coût le plus faible possible. Cependant, comme l'horaire initial n'est pas pris en compte lors de la ré-optimisation, nous nous attendons à avoir énormément de modifications pour les quarts. De plus, comme nous utilisons le logiciel pour l'ensemble de la ré-optimisation, le temps de résolution sera vraisemblablement de l'ordre de celui de l'optimisation initiale ($360s$) ce qui est trop long dans une perspective de mise à jour. Cette proposition peut être assimilée à l'extrême où toutes les transformations sont permises et où les pénalités de modification sont nulles.

Solution initiale (*fixed*)

La proposition de coût le plus élevé que nous pourrons atteindre est celle où l'ensemble des quarts des employés est fixé. L'horaire n'est ici pas modifié : nous gardons la solution initiale. Des sur-couvertures et sous-couvertures sont alors engendrées par cette non-modification de l'horaire. Comme la sous-couverture est interdite, nous optimisons en permettant uniquement la génération de quarts anonymes par le logiciel. La sur-couverture et les quarts anonymes ainsi créés pour compenser la non-modification impliqueront un coût de la solution élevé.

Cette proposition est l'extrême inverse de celle présentée précédemment. C'est l'équivalent d'une proposition où les pénalités de modification sont infinies ou d'une proposition avec aucune transformation autorisée.

Décrivons maintenant des propositions intermédiaires.

4.2.2 Transformations autorisées

Nous pouvons autoriser plus ou moins de transformations. Cela mènera à un nombre de quarts proposés différents et donc à des solutions de qualités différentes mais aussi sûrement

à des temps de résolution différents.

Types de transformations

La première proposition est de n'utiliser qu'un seul type de transformation à la fois c'est-à-dire d'interdire les combinaisons de transformations. Nous autorisons ici aussi les nouveaux quarts tels que décrits dans la section 3.2.2. Ce choix est nommée ***trans1***.

La deuxième proposition est d'autoriser la combinaison d'au maximum deux types de transformations. Nous gardons également l'autorisation des nouveaux quarts. On nomme ce choix ***trans2***.

Puisque les quarts proposés dans *trans1* le sont aussi dans *trans2*, la solution optimale de *trans1* sera une solution réalisable pour *trans2*. Nous aurons donc forcément :

$$\mathbb{C}(trans1) \geq \mathbb{C}(trans2)$$

où $\mathbb{C}(x)$ représente le coût de la solution optimale pour une instance avec la proposition x.

Cependant, comme plus de quarts sont proposés dans $trans2$, il est très probable que l'on ait $\mathbb{T}(trans1) \leq \mathbb{T}(trans2)$ où $\mathbb{T}(x)$ est le temps de résolution d'une instance avec la proposition x.

En étudiant le nombre de quarts proposés pour le patron 1 pour chaque type de transformations (détails dans le tableau 4.5 de la section 4.3.2), nous constatons que la majorité des quarts proposés, soit entre 62% (pour *trans1*) et 74% (pour *trans2*), sont de type *replaced* c'est-à-dire que ce sont des quarts ajoutés tel que décrit dans la section 3.2.2. Cependant, peu de ces quarts sont effectivement utilisés dans la solution optimale. Nous aimerions donc diminuer le nombre de quarts de type *replaced* pour que le temps de résolution diminue tout en gardant la même qualité de solution. Nous pouvons analyser plus précisément la nature des quarts *replaced* proposés et effectivement utilisés dans la solution optimale de *trans2*.

Les quarts *replaced* sont admissibles pour l'employé mais ne suivent pas les règles de transformation a priori autorisées puisque ce sont des quarts ajoutés. Ils peuvent être de différents types :

- Ils peuvent correspondre à la combinaison des trois types de transformations avec ou sans respect des bornes $[minAdd, maxAdd]$ et $[minCut, maxCut]$.
- Ils peuvent correspondre à la combinaison d'au plus deux types de transformations mais sans respect des bornes $[minAdd, maxAdd]$ et $[minCut, maxCut]$.
- Ils peuvent également correspondre à une combinaison interdite à savoir la réduction

ou l'allongement des deux côtés du quart avec changement ou non de l'activité.

Sans considérer le fait que l'activité a été changée ou non et sans considérer la longueur d'allongement ou de réduction, nous distinguons donc 8 types de transformations pour les quarts *replaced* : *ext&none*, *red&none*, *none&ext*, *none&red*, *ext&ext*, *red&ext*, *ext&red* et *red&red*. Chacun de ces types est la combinaison de deux types de transformations simples appliqués pour le premier au début du quart et le second à la fin du quart, *none* correspondant à aucun changement. Le tableau 4.2 donne le nombre de quarts correspondant à chacun de ces types pour *trans2* dans le cas du patron 1. Ces nombres sont plus faibles dans le cas de *trans1* et pour les autres patrons mais les proportions restent les mêmes.

Tableau 4.2 Type des quarts *replaced* proposés (*trans2, patron 1*).

Type	activité non modifiée	activité modifiée	**total**	**total (%)**
ext&none	0	0	**0**	**0**
red&none	74	271	**345**	**0,5**
none&ext	3	18	**21**	**0,0**
none&red	29	252	**281**	**0,4**
ext&ext	209	746	**955**	**1,5**
red&ext	3252	24092	**27344**	**42,1**
ext&red	3072	20960	**24032**	**37,0**
red&red	1993	10024	**12017**	**18,5**
total	**8632**	**56363**	**64995**	**100,0**

On constate que le nombre de quarts dont l'activité est modifiée est beaucoup plus élevé que les quarts dont l'activité n'est pas modifiée. De plus, les types principalement présents sont *red&ext*, *ext&red* et *red&red*. Analysons les types utilisés dans la solution optimale de *trans*2 dans le tableau 4.3. On fait ici une moyenne sur les 80 scénarios de demande.

On constate que le le nombre de quarts utilisés de type *ext&red*, *red&ext* ou *red&red* n'est pas négligeable et on remarque qu'un nombre important de ces quarts voient leur activité également modifiée. Parmi les transformations uniques, seules *none&red* n'est pas négligeable. Nous proposons donc de tester une instance où seuls les quarts *replaced* de type *none&red* sont gardés (***lessReplaced***). Nous souhaitons analyser à quel point le coût de la solution va augmenter avec cette proposition.

Par la suite, nous présentons des propositions faisant varier d'autres paramètres. Ces propositions peuvent être combinés avec *trans*1, *trans*2 ou *lessReplaced* indifféremment. Afin de pouvoir comparer les résultats de manière efficace, nous choisissons *trans*2 comme proposition par défaut avec laquelle nous combinerons l'ensemble des propositions qui vont suivre.

Tableau 4.3 Type des quarts *replaced* utilisés (% moyen).

Type	**total**	dont activité modifiée
ext&none	**0,0**	-
red&none	**0,2**	100
none&ext	**0,0**	-
none&red	**6,6**	87,5
ext&ext	**4,0**	71,4
red&ext	**28,0**	98,6
ext&red	**36,5**	86,2
red&red	**18,5**	83,3
total	**100**	**34,3**

Jours de transformations

Comme présenté à la section 4.1, en fonction des patrons, la demande est modifiée certains jours et pas d'autres et la modification de l'horaire peut être autorisée certains jours où la demande est restée fixe. Par exemple, pour les patrons 2 et 3, la demande est modifiée uniquement le jeudi mais l'horaire peut être mis à jour du jeudi au dimanche.

Par défaut, nous générons des quarts transformés sur l'ensemble des jours où la modification de l'horaire est possible. Cependant, il peut être intéressant d'étudier la solution lorsqu'on ne génère des quarts que les jours où la demande est modifiée (***trans2_dayOfMod***) car il est probable que la plupart des quarts transformés choisis se concentrent sur ces journées. Les quarts travaillés en dehors de ces journées sont alors fixés. Nous aurons donc nécessairement :

$$\mathbb{C}(trans2\_dayOfMod) \geq \mathbb{C}(trans2)$$

Nous espérons avoir $\mathbb{T}(trans2\_dayOfMod) \leq \mathbb{T}(trans2)$. Avec cette proposition, nous espérons garder la même qualité de solution que $trans2$ mais diminuer le temps de calcul de la solution optimale.

Activités concernées par les transformations

Par défaut, nous transformons tous les quarts même ceux dont l'activité ne voit pas sa demande perturbée. Nous pourrions transformer seulement les quarts dont l'activité a sa demande modifiée (***trans2_modJobs***). En effet nous pouvons imaginer que les modifications de l'horaire se font principalement sur des activités dont la demande est modifiée. En

diminuant ainsi le nombre de quarts proposés, nous espérons diminuer le temps de résolution du problème tout en gardant la même qualité de solution.

Génération de quarts anonymes

Lors de la génération des quarts, nous proposons aussi un certain nombre de quarts anonymes tel que présenté aux sections 3.2.2 et 3.2.3. Par défaut, nous nous contentons de proposer ces quarts.

Cependant, nous souhaitons vérifier si les quarts anonymes proposés suffisent ou si nous pourrions atteindre une meilleure solution en en proposant plus. Nous proposons donc de résoudre le problème en utilisant le générateur de quarts anonymes du logiciel déjà en service (***trans2_genOS***).

4.2.3 Valeur des pénalités de modification

Comme expliqué à la section 3.3, c'est à l'utilisateur de définir la valeur des pénalités de modifications. Nous avons défini des pénalités par défaut en fonction de différents critères. Cependant, il semble intéressant d'étudier les solutions optimales pour d'autres valeurs des pénalités afin d'analyser l'impact de leur variation sur la solution. Nous faisons donc trois propositions jouant sur la valeur des pénalités :

Pas de pénalité

La première proposition est de choisir les pénalités nulles (***trans2_noPen***). Cela revient à ne pas pénaliser la modification de l'horaire tout en maîtrisant les transformations autorisées. On aura :

$$\mathbb{C}(exact) \leq \mathbb{C}(trans2\_noPen) \leq \mathbb{C}(trans2)$$

Si le coût de la solution de $trans2$ est éloigné du coût de la solution $exact$, l'analyse du coût de $trans2\_noPen$ nous permettra de comprendre d'où vient cette différence de coût. En effet, si $\mathbb{C}(trans2\_noPen)$ est plus proche de $\mathbb{C}(exact)$ que de $\mathbb{C}(trans2)$, cela signifie que les pénalités de modification appliquées pour $trans2$ dissuadent les modifications et donc empêchent d'arriver à une meilleure solution. Si c'est l'inverse, cela signifie que les quarts proposés pour la ré-optimisation dans $trans2$ ne permettent pas d'atteindre une solution de qualité proche de la solution $exact$.

Pénalités plus élevées

On peut également étudier la proposition où on choisit des pénalités plus élevées ***trans2_morePen***. Nous choisissons ici de multiplier par 5 les pénalités par défaut. Nous avons donc alors une pénalité fixe de 1000$ et une pénalité variable de 375$/*période*. Cette proposition nous permettra d'avoir une nouvelle indication sur l'effet persuasif des pénalités.

De plus, il existe plusieurs symétries dans le problème. Il est possible que de faibles pénalités induisent une exploration lente de l'arbre de branchement. Nous souhaitons donc regarder si en augmentant les pénalités, le temps de résolution va diminuer.

Doubles pénalités pour les combinaisons

La pénalité fixe par défaut pénalise le fait de transformer un quart sans se soucier du type de transformation. Cela veut dire qu'un quart réduit d'une heure est pénalisé exactement à la même hauteur qu'un quart réduit d'une heure et dont l'activité change. Cela n'est pas toujours réaliste puisqu'on préfère a priori avoir le moins de modifications même au sein d'un quart déjà modifié.

Nous proposons donc ici de doubler la pénalité fixe si on a une combinaison de deux types de transformations ou si le quart est de type *replaced* (***trans2_doublePen***).

4.2.4 Nombre d'activités dont la demande est modifiée

Nous avons proposés des scénarios où beaucoup d'activités voient leur demande modifiée. Nous souhaitons tout de même faire un test avec seulement une ou deux activités modifiées. Nous gardons donc les patrons avec les jours de modifications définis mais en ne perturbant la demande que de deux activités (***lessJobs***). Les patrons 2 et 3 sont alors les mêmes.

Cette proposition est un peu à part car elle change les scénarios de demande et pas les quarts proposés. La valeur de la demande modifiée étant alors différente, nous ne pourrons pas comparer cette proposition directement aux autres. Cependant, cela nous permettra notamment d'analyser la différence de temps de calcul en fonction de l'intensité de la perturbation : est-ce qu'une perturbation moins importante amène un temps de résolution plus faible ?

Nous testerons cette proposition avec *trans2* et *trans2_modJobs*.

4.3 Données

Nous présentons ici les caractéristiques du jeu de données initial et celles des différentes propositions que nous venons de faire.

4.3.1 Solution initiale

On désigne par ***init*** la solution initiale. Les caractéristiques de la solution initiale pour le jeu de données à 49 employés et 7 activités sont présentées dans la tableau 4.4.

Tableau 4.4 Caractéristiques de la solution initiale.

Nombre de quarts	employés	184
	quarts anonymes	0
Sur-couverture (% du temps travaillé)		2,0
Coût ($/h)		506,87

La solution initiale ne comprend donc aucun quart anonyme.

4.3.2 Informations sur les propositions

Le tableau 4.5 présente le nombre de quarts proposés pour le patron 1 pour chacune des propositions énumérées ci-dessus ainsi que le nombre de quarts par type de transformation. QA désigne le nombre de quarts anonymes.

Tableau 4.5 Nombre de quarts proposés (patron 1).

Propositions			Nature de la transformation							tot.	QA
			none	*red*	*ext*	*jobMod*	*shift.*	*J&H.*	*repl.*		
exact			-	-	-	-	-	-	-	-	-
fixed			0	0	0	0	0	0	0	0	-
*trans*1			184	1315	2124	1062	0	0	7720	12405	117
*trans*2	-	-	184	1315	2124	1062	1654	16438	64995	87772	117
		genOS									-
		noPen									117
		morePen									
		doublePen									
		dayOfMod	112	819	1286	648	989	10006	35429	49289	74
		modJobs	184	1138,35	1867,8	776,7	1442,6	12062,45	53652,0	71123,9	104,75
lessReplaced			184	1315	2124	1062	797	16438	281	22201	117
*trans*2	*lessJobs*	-	184	1315	2124	1062	1654	16438	64995	87772	117
		modJobs	184,0	507,5	769,8	264,0	365,5	4152,3	21145,4	27388,5	50,1

On constate une forte différence du nombre de quarts proposés en fonction des propositions ce qui est normal et attendu. Comme nous l'avions déjà signalé, les quarts de type *replaced* prennent une forte place dans les quarts proposés.

On constate que le nombre de quarts proposés est multiplié par 7 entre les propositions *trans*1 et *trans*2 ce qui fait une augmentation considérable du nombre de variables pour le modèle.

Le fait de ne proposer des quarts que les jours où la demande est modifiée (*dayOfMod*) diminue tout de même presque par deux le nombre de quarts proposés ce qui semble logique puisque, pour le patron 1, 4 des 7 jours voient leur demande modifiée.

Pour *modJobs*, il faut préciser que même si toutes les activités peuvent avoir leur demande modifiée dans le patron 1, ce n'est pas toujours le cas. Ainsi, nous avons fait une moyenne sur les 20 scénarios. Si toutes les activités étaient modifiées pour tous les scénarios, le nombre de quarts proposés pour *trans2_modJobs* serait le même que le nombre de quarts proposés pour *trans2*. De manière évidente, si on a moins d'activités dont la demande est modifiée (*trans2_lessJobs*) et qu'on ne propose des quarts que pour les activités perturbées, le nombre de quarts proposés diminue considérablement.

Le nombre de quarts anonymes est plutôt faible car la longueur minimale de quart anonyme $minOSLength$ est égale à 240 dans notre jeu de données. Or, comme les morceaux de quarts raccourcis ont une longueur inférieure à $maxCut = 120$, aucun de ces morceaux n'est proposé comme quart anonyme. Les quarts anonymes proposés sont donc uniquement les quarts initialement planifiés pour les employés et dont la longueur est comprise entre $minOSLength$ et $maxOSLength$.

En fonction du patron, le nombre de quarts proposés est bien évidemment différent. Dans le tableau 4.6, nous présentons le nombre de quarts proposés pour chaque patron dans le cas de la proposition *trans2*.

Tableau 4.6 Nombre de quarts proposés (*trans2*).

Patron	Quarts proposés et nature de la transformation								QA
	none	*red*	*ext*	*jobMod*	*shift.*	*J&H*	*repl.*	*tot.*	
Patron 1	184	716	2124	1062	797	16032	66857	87772	117
Patron 2	112	857	1301	648	948	10289	40997	55152	74
Patron 3									
Patron 4	26	194	314	150	223	2472	13456	16835	16

La différence entre les patrons tient évidemment du fait que les jours où l'horaire est modifiable sont moins nombreux pour le patron 4 par rapport aux patrons 2 et 3 et encore moins nombreux pour le patron 1.

Nous avons donc grâce à ces tableaux, un ordre de grandeur du nombre de quarts proposés pour chaque instance.

CHAPITRE 5 ANALYSE DES RÉSULTATS

Dans ce chapitre, nous présentons et analysons les résultats de nos tests. Plusieurs indicateurs sont à observer :

- Le temps de résolution du problème par l'optimiseur (section 5.1)
- Le coût de la solution avec et sans prise en compte des pénalités (section 5.2)
- Le nombre de quarts modifiés et les caractéristiques de ces quarts telles que le type de transformations subies ou les jours concernés (section 5.3)
- Le taux de sur-couverture et de quarts anonymes (sous-couverture) dans la solution (section 5.4)

Après l'observation de ces indicateurs, nous conclurons de manière partielle dans la section 5.5. Nous effectuons nos calculs sur un système d'exploitation Windows 8.1 (2013) 64 bits avec un processeur Intel(R) Core(TM) CPU @ 3.40 GHz. Le code est en Java.

5.1 Temps de calcul

Nous commençons par observer le temps de résolution réel pour chaque instance. Nous donnons dans le tableau 5.1 la moyenne de ce temps pour chaque patron et chaque proposition ainsi que la moyenne pour les 80 scénarios de demande pour chaque proposition.

Tableau 5.1 Temps de résolution (s).

Propositions			Pat. 1	Pat. 2	Pat. 3	Pat. 4	**Moyenne**
init			-	-	-	-	**359, 9**
exact			408, 3	131, 4	130, 6	18, 1	**172, 1**
fixed			6, 1	5, 7	5, 9	5, 6	**5, 8**
*trans*1			3, 7	1, 7	1, 7	0, 7	**2, 0**
*trans*2	-	-	24, 7	12, 3	12, 3	3, 2	**13, 1**
		genOS	66, 6	31, 6	31, 9	11, 6	**35, 4**
		noPen	51, 9	20, 9	20, 6	3, 8	**24, 3**
		morePen	20, 2	11, 5	11, 6	3, 0	**11, 6**
		doublePen	23, 4	12, 2	11, 8	3, 1	**12, 6**
		dayOfMod	14, 1	2, 8	2, 8	3, 1	**5, 7**
		modJobs	17, 6	3, 4	7, 0	0, 9	**7, 2**
lessReplaced			8, 5	4, 2	4, 2	0, 9	**4, 5**
*trans*2	*lessJobs*	-	23, 0	12, 3	12, 0	3, 1	**12, 6**
		modJobs	3, 3	1, 2	1, 4	0, 5	**1, 6**

On observe que le temps de résolution pour la proposition *trans2* est en moyenne 6 fois plus

élevé que pour l'instance *trans1*.

L'utilisation de *dayOfMod* et *modJobs* permet de réduire le temps de calcul de manière significative. De manière attendue, c'est d'autant plus le cas que le nombre de quarts proposés est réduit lors de leur utilisation. Ainsi, l'effet de *dayOfMod* est particulièrement frappant pour les patrons 2 et 3 où le nombre de jours où la demande est modifiée par rapport au nombre de jours où l'horaire est modifiable est le plus faible (1/4 contre 3/7 pour le patron 1 et 1/1 pour le patron 4). Pour *modJobs*, la diminution du temps de résolution est plus élevée pour les patrons où seules certaines activités sont perturbées (patrons 2 et 4). De même, la diminution de temps pour *trans2_lessJobs_modJobs* par rapport à $trans2\_lessJobs$ est vraiment plus importante que pour *trans2_modJobs* par rapport à *trans2*.

Nous pouvons constater en observant le temps de résolution pour les instances *trans2_lessJobs* et en comparant les temps entre les patrons 2 et 3 que le fait d'avoir moins d'activités modifiées n'influe pas beaucoup sur le temps de résolution. De même, contrairement à nos attentes, *trans2_morePen* n'a pas un temps de résolution vraiment plus faible par rapport à *trans2*. Le fait d'augmenter les pénalités n'a donc pas permis de diminuer de manière assez significative le temps d'exploration de l'arbre de branchement.

Nous remarquons cependant que le temps de résolution lorsque les pénalités sont nulles (*trans2_noPen*) augmente fortement puisqu'il est presque multiplié par deux. On a sûrement ici des symétries qui allongent le parcours dans l'arbre. En dehors de la pénalisation des modifications, il n'est donc pas négligeable de mettre des pénalités pour différencier facilement les solutions que l'on accepte de celles qu'on ne souhaite pas avoir.

Cependant, nous pouvons constater que le temps est vraiment différent en fonction des patrons. Ainsi, les instances du patron 1 ont un temps 2 fois plus élevé que celles des patrons 2 et 3 qui elles-mêmes ont un temps de résolution 4 fois plus élevé que le patron 4 en moyenne. Cela est sûrement dû au nombre de quarts proposés qui diminue de plus en plus en fonction des patrons plutôt qu'à la taille de la perturbation car celle-ci est de même taille pour le patron 2 ou pour le patron 4.

Nous proposons d'étudier le rapport entre le temps de résolution et la taille de la perturbation ainsi qu'entre le temps de résolution et le nombre de quarts proposés.

Pour étudier le rapport entre le temps de résolution et la taille de la perturbation, nous choisissons des propositions où tous les autres paramètres sont égaux à savoir : même nombre de quarts proposés et mêmes valeurs de pénalités de modifications. Nous utilisons donc les instances de *trans2* et de *trans2_lessJobs*, patrons 2 et 3. L'importance de la perturbation

est représentée par la donnée :

$$P = \sum_{a \in A} \sum_{p \in P} |d_{ap} - d'_{ap}|$$

Nous mettons en abscisse l'importance de la perturbation (P) et en ordonnée le temps de résolution associé. Nous obtenons le nuage de points de la figure 5.1. La courbe rouge est la droite de régression linéaire pour ces valeurs.

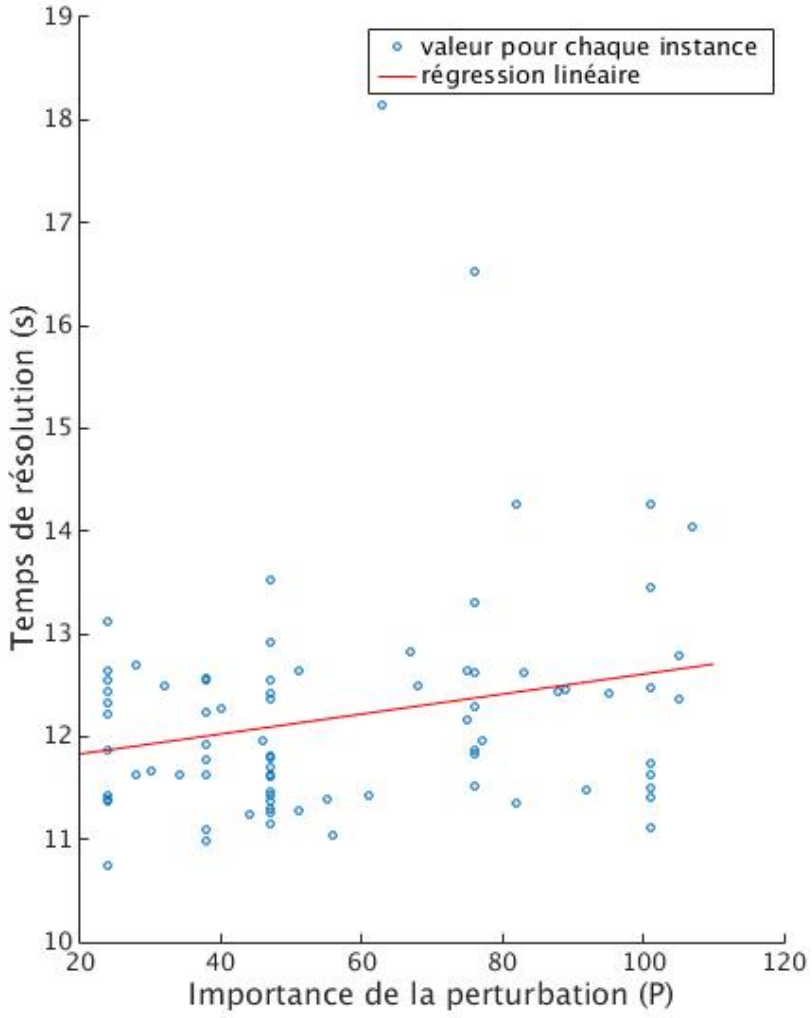

Figure 5.1 Rapport temps de résolution/importance de la perturbation.

On constate qu'il y a une très faible corrélation entre le temps de résolution et l'importance de la perturbation. Le facteur de corrélation est de 0.16. Ainsi, le fait d'avoir une perturbation

de plus ou moins grande intensité n'a qu'une très faible influence sur le temps de résolution.

On compare aussi le nombre de quarts proposés (nombre de variables dans le modèle) avec le temps de résolution du problème. Pour cela, on compare les propositions avec mêmes valeurs de pénalités de modification et même perturbation de la demande à savoir *trans1*, *trans2*, *trans2_dayOfMod*, *trans2_modJobs* et *lessReplaced*. On obtient le nuage de points de la figure 5.2.

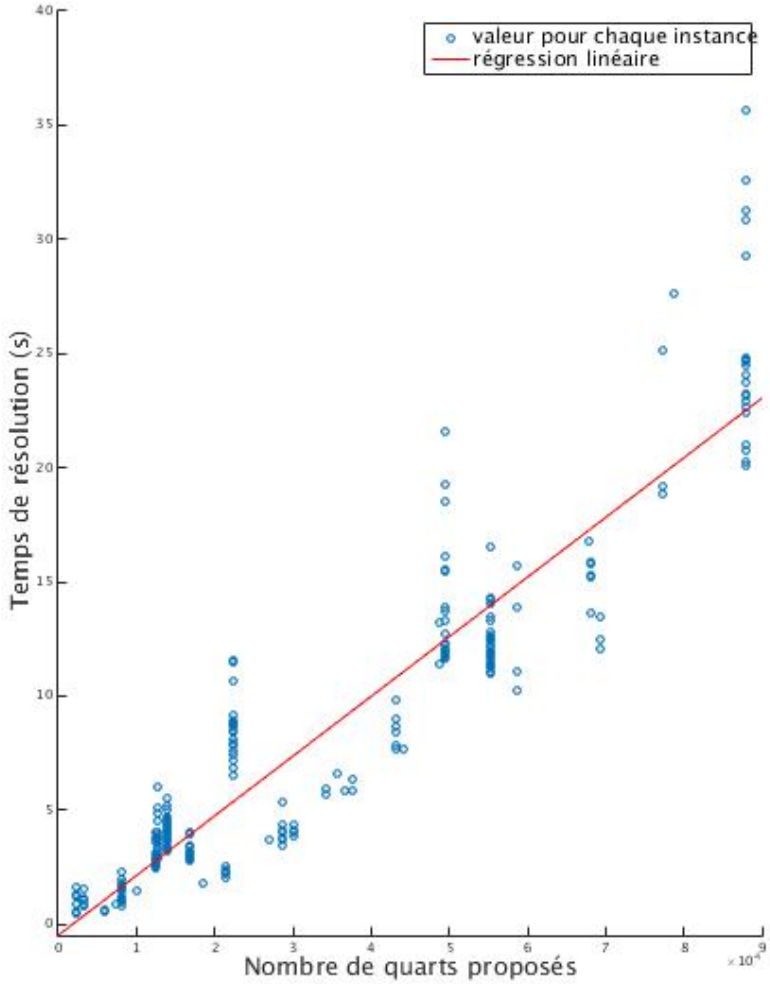

Figure 5.2 Rapport temps de résolution/nombre de quarts proposés.

On constate qu'il y a un rapport clair entre le nombre de quarts proposés et le temps de résolution. Le facteur de corrélation est ici de 0.95 ce qui est très élevé. Ainsi, plus on propose de quarts et donc plus il y a de variables dans le modèle, plus le problème met du temps à

être résolu. Cela prouve la nécessité de travailler sur la bonne sélection des quarts que nous proposons en amont afin de pouvoir diminuer le temps de résolution tout en gardant une solution de qualité.

5.2 Coûts

Il est vraisemblable que le coût des solutions des instances avec un scénario d'intensité 20% sera bien plus élevé que celui des solutions des instances avec un scénario d'intensité -20%. Ces coûts ne sont pas comparables en l'état. Nous choisissons donc de diviser le coût de la solution par le nombre d'heures travaillées. Cela permet une comparaison efficace. On regarde ici le coût sans les pénalités de modification (seulement le coût de la main d'œuvre, de la sur-couverture et des quarts anonymes) et le coût avec les pénalités de modification fixées à 200$ de pénalité fixe et 75$/*période* de pénalité variable. Cela permet de voir quelle partie du coût est dû aux modifications de l'horaire. Les coûts des différentes propositions sont présentés dans le tableau 5.2.

Tableau 5.2 Coûts ($/$h$).

Propositions			sans pénalités					avec pénalités
			Pat. 1	Pat. 2	Pat. 3	Pat. 4	**Moyenne**	**Moyenne**
init			-	-	-	-	**506, 87**	**-**
exact			544, 46	518, 99	519, 77	515, 0	**524, 56**	**726, 15**
fixed			1363, 86	669, 20	765, 92	765, 92	**863, 58**	**863, 58**
*trans*1			784, 61	542, 18	578, 34	555, 66	**615, 20**	**638, 07**
*trans*2	-	-	684, 95	520, 41	549, 28	535, 45	**572, 52**	**596, 85**
		genOS	665, 74	519, 19	545, 14	534, 92	**566, 23**	**590, 12**
		noPen	676, 84	519, 10	545, 84	535, 09	**569, 22**	**624, 81**
		morePen	724, 45	484, 09	519, 75	481, 12	**590, 20**	**609, 64**
		doublePen	685, 79	520, 14	548, 93	535, 55	**572, 60**	**596, 78**
		dayOfMod	686, 89	530, 24	559, 19	535, 45	**577, 95**	**601, 33**
		modJobs	695, 73	541, 45	552, 25	557, 39	**586, 71**	**609, 36**
lessReplaced			738, 83	525, 32	561, 37	550, 48	**594, 0**	**614, 21**
*trans*2	*lessJobs*	-	533, 75	509, 76	509, 68	515, 74	**517, 23**	**533, 88**
		modJobs	652, 67	550, 39	550, 87	538, 73	**573, 17**	**582, 56**

On observe que le fait de ré-optimiser plutôt que de garder l'horaire initial en ajoutant uniquement des quarts anonymes (*fixed*) permet d'améliorer significativement le coût de la solution (ici, on gagne de 220$ à 300$ par heure en fonction de la proposition choisie). Quand on refait l'optimisation de zéro en permettant la génération de tous les quarts possibles (*exact*), les coûts associés à l'optimisation sont effectivement les plus faibles possibles pour une même demande (*lessJobs* excepté). Cependant, lorsqu'on prend en compte les pénalités de modification, le coût devient l'un des plus élevé. Cela vient du fait que nous ne nous

soucions pas de l'horaire initial et qu'il n'y a pas de pénalité de modification lors de la ré-optimisation. Le nombre de modifications est donc très élevé.

Nous constatons que le fait d'utiliser plus de quarts (*trans2*) fait tout de même gagner une somme non négligeable puisque le coût de la solution *trans2* est 7% plus faible que le coût de la solution *trans1*. Ainsi, l'ajout de possibilités de transformations permet d'atteindre une meilleure solution.

La génération de quarts anonymes par le logiciel (*genOS*) n'améliore pas la solution de manière significative (diminution du coût de 1%). Cela signifie que nos propositions de quarts anonymes sont suffisantes. De plus, comme nous l'avons vu à la section précédente, le temps de résolution de cette proposition est vraiment plus élevé. Nous ne retiendrons pas cette proposition.

Le fait de ne mettre aucune pénalité (*noPen*) diminue à peine le coût de la solution. Les pénalités données (200$ fixes et $5\$/min$) sont donc trop faibles pour dissuader de faire des modifications qui pourraient permettre de diminuer le coût de la solution. On constate cependant que le coût avec prise en compte des pénalités de modification est vraiment élevé pour *noPen*. Cela indique que de nombreuses modifications sont faites. Nous le vérifierons dans la section 5.3.1 et expliquerons que cela est dû aux nombreuses symétries du problème. On observe également que le coût de la solution lorsque les pénalités sont plus élevées (*morePen*) est $3,5\%$ plus élevé que *trans2*. Cela signifie que les pénalités de modification ont réellement dissuadé de transformer certains quarts. Des sur-couvertures ont été laissées ou des quarts anonymes ont été utilisés à la place de potentielles modifications. Enfin, le coût de *trans2_doublePen* est le même que *trans2*. L'augmentation de la pénalité fixe pour certaines transformation n'a donc pas eu d'influence sur le coût.

Nous observons enfin que la proposition *modJobs* ne donne pas de solution de qualité notamment dans le cas *lessJobs* où un nombre très faible d'activités a été modifié. On constate que le coût de la solution augmente ici de 11%. La différence est moins visible pour *trans2* car plus d'activités sont perturbées et donc l'influence de la suppression des quarts couvrant les activités non perturbées est faible. Nous ne pouvons donc pas retenir cette proposition.

La proposition *dayOfMod* où on ne permet la transformation des quarts que les jours où la demande est modifiée parait quant à elle efficace. En effet le coût de la solution est augmenté de seulement $0,8\%$. De plus, nous avons précédemment vu que le temps de résolution associé était vraiment satisfaisant. Cette proposition est à retenir.

Enfin, *lessReplaced* a un coût $3,4\%$ plus élevé que *trans2*. Le fait d'enlever certains types de quarts *replaced* qui étaient utilisés dans la solution optimale de *trans2* a donc nettement

influé sur la qualité de cette solution.

Nous souhaitons pouvoir comparer l'ensemble des propositions en prenant en compte à la fois le temps de résolution et la qualité de la solution obtenue. Nous étudions donc le rapport temps/coût pour l'ensemble des propositions. Les résultats sont présentés sur la figure 5.3. On trace aussi la courbe convexe reliant les points ayant le meilleur rapport coût-temps. Nous n'incluons pas les propositions *exact* et *fixed* du graphique car elles sont très éloignées des autres en temps (pour *exact*) ou en coût (pour *fixed*). *trans2_doublePen* étant quasiment confondu avec *trans2*, il n'apparait pas sur le graphique. Bien entendu, nous n'incluons pas les deux propositions *trans2_lessJobs* qui sont non comparables directement avec les autres puisque la valeur de la demande modifiée est différente.

Nous constatons que les deux propositions *trans2_morePen* et *trans2_modJobs* sont dominées. Cela confirme le fait qu'elles sont moins intéressantes que les autres propositions.

La courbe est d'abord décroissante de manière très abrupte de *trans2_genOS* à *trans2_dayOfMod* puis prend une pente plus douce jusqu'à *trans1*. Ainsi, comme nous l'avons avancé, *trans2_dayOfMod* paraît une proposition intéressante à la fois en temps et en coût. On peut gagner un peu en coût mais en perdant beaucoup en temps en utilisant *trans2*, *trans2_doublePen*, *trans2_noPen* ou *trans2_genOS* et on peut gagner un peu en temps mais en perdant beaucoup en coût avec *lessReplaced* ou *trans1*.

Bien entendu, le coût et le temps ne sont pas les seuls critères de choix et il faut aussi prendre en compte le nombre de quarts modifiés si on veut pouvoir choisir la proposition qui convient le mieux.

5.3 Transformations des quarts

Nous étudions la nature de la modification de l'horaire sous deux angles : le nombre de transformations totales subies puis la nature exacte des transformations choisies.

5.3.1 Nombre de modifications

Nous souhaitons analyser le nombre de transformations des quarts des employés en réussissant à comparer l'ensemble des propositions alors qu'elles ont un nombre de quarts proposés très différents. On calcule le ratio :

$$\frac{quarts\,transformés}{quarts\,transformables}$$

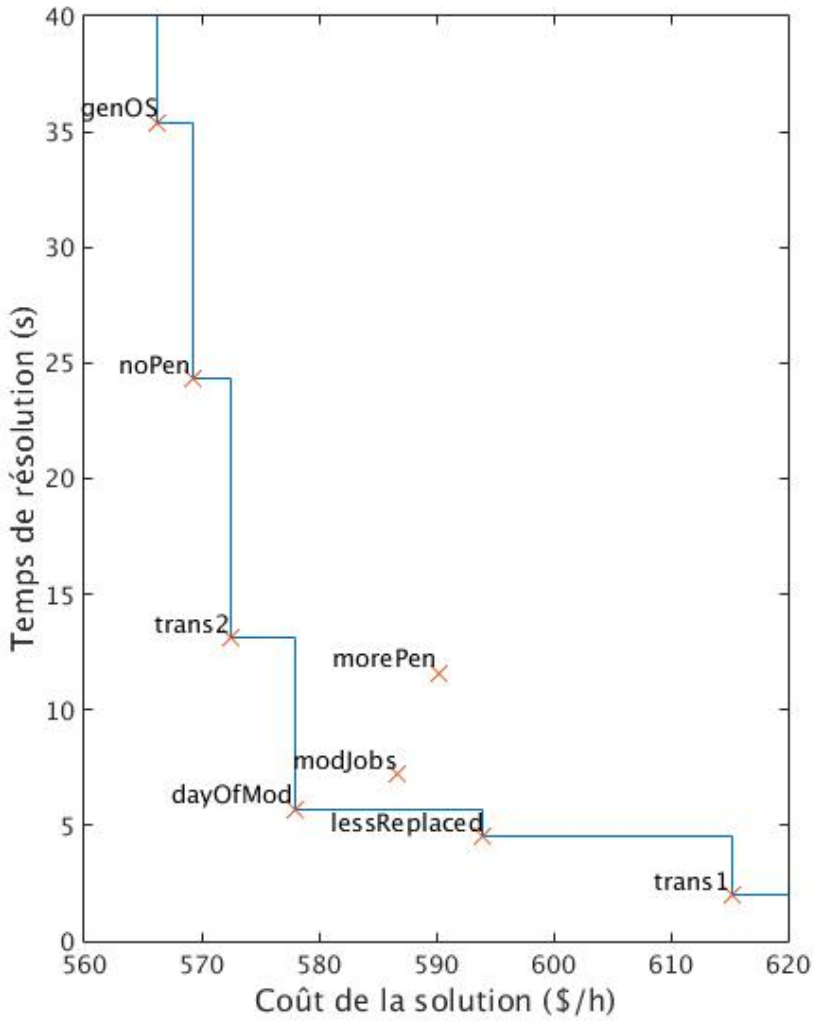

Figure 5.3 Comparaison de la qualité des différentes propositions.

Les quarts transformables sont les quarts non fixés dans l'horaire lors de la ré-optimisation. Nous présentons les résultats dans le tableau 5.3.

Comme avancé dans l'analyse des coûts, nous constatons que le nombre de modifications pour l'instance *exact* est très élevé par rapport aux autres instances. Cela est dû au fait qu'on permet toute modification sans pénalités. Le nombre de modifications pour *trans2_noPen* est aussi très élevé par rapport à *trans2*. Nous avons cependant pu constater lors de la comparaison des coûts que le coût sans pénalités n'était pas si différent. Une analyse plus poussée nous permet de voir que ce taux de modifications est en fait dû au grand nombre de symétries dans le problème. De nombreux employés voient en effet leurs quarts échangés sans

Tableau 5.3 Nombre de quarts modifiés (%).

Propositions			Pat. 1	Pat. 2	Pat. 3	Pat. 4	**Moyenne**
init			-	-	-	-	**-**
exact			78, 8	69, 4	70, 2	82, 3	**75, 2**
fixed			0	0	0	0	**0**
*trans*1			31, 8	16, 5	18, 5	42, 1	**27, 2**
*trans*2	-	-	34, 2	17, 9	20, 1	48, 1	**30, 1**
		genOS	33, 8	17, 4	19, 8	47, 9	**29, 7**
		noPen	54, 3	49, 5	50, 4	64, 6	**54, 7**
		morePen	29, 0	15, 1	17, 6	39, 8	**25, 4**
		doublePen	33, 7	17, 5	20, 2	46, 9	**29, 6**
		dayOfMod	33, 6	13, 5	15, 6	48, 1	**27, 7**
		modJobs	33, 2	14, 2	19, 0	35, 8	**25, 5**
lessReplaced			33, 0	17, 5	19, 2	43, 7	**28, 3**
*trans*2	*lessJobs*	-	24, 3	15, 6	15, 0	36, 5	**22, 9**
		modJobs	14, 0	7, 3	8, 2	20, 2	**12, 4**

aucun effet réel sur la couverture de la demande ou le coût (sans les pénalités de modification). Prenons l'exemple du scénario 4-20%-4 pour laquelle le coût de la solution est exactement le même entre *trans2* et *trans2_noPen* mais où le taux de quarts modifiés est 57, 7% pour *trans2* et 65, 4% pour *trans2_noPen*. On a deux employés (A et B) dont l'horaire du dimanche (jour modifié) est :

Employé A :

— Horaire initial : Activité 7 de 12h30 à 17h15.
— Horaire *trans2* : Activité 7 de 10h45 à 17h15.
— Horaire *trans2_noPen* : Activité 10 de 11h00 à 17h15.

Employé B :

— Horaire initial : Activité 10 de 11h00 à 17h15.
— Horaire *trans2* : Activité 10 de 11h00 à 17h15.
— Horaire *trans2_noPen* : Activité 7 de 10h45 à 17h15.

Les quarts choisis sont exactement échangés entre *trans2* et *trans2_noPen*. De plus, *trans2_noPen* implique une modification d'horaire pour l'employé B alors qu'il n'en avait pas avec *trans2* et une modification plus lourde (et donc plus pénalisée) pour l'employé A qui subit un changement d'activité et d'heure avec *trans2_noPen* alors qu'il change uniquement d'heure avec *trans2*. Cela prouve l'utilité de mettre des pénalités de modifications, même très faibles.

Nous pouvons observer que le nombre de modifications diminue avec l'augmentation des pénalités (*doublePen* et *morePen*). Cela semble logique puisque le fait de pénaliser plus fortement les modifications ne peut que dissuader d'utiliser des quarts transformés. La proposition *trans2_doublePen* a un nombre de quarts modifiés tout de même très proche de *trans2* alors que le nombre de transformations observées pour *trans2_morePen* est de 5 points moins important par rapport à *trans2* ce qui est une différence importante.

On constate que le nombre de modifications est un peu plus élevé pour *trans2* que pour *trans1*. Même avec les mêmes pénalités, le fait d'avoir plus de quarts proposés augmente donc de manière sensible le nombre de modifications. Comme nous l'avons déjà indiqué, les pénalités que nous avons choisies par défaut sont trop faibles pour dissuader réellement les modifications.

En comparant *trans2_lessJobs* et *trans2_lessJobs_modJobs*, on constate que le nombre de modifications est nettement plus faible pour le deuxième cas. Cela montre que la diminution du nombre de quarts a aussi diminué les possibilités de modifications effectivement utilisées. Cela confirme ce que nous avions avancé dans la section précédente à savoir que *modJobs* n'est pas une proposition intéressante car elle supprime trop de quarts effectivement utilisés dans la solution optimale. On peut observer le même phénomène si nous comparons *trans2* et *trans2_modJobs* pour les patrons 2 et 4.

On constate que la proposition *trans2_dayOfMod* voit aussi un nombre de modifications fortement diminué par rapport à *trans2* principalement pour les patrons 2 et 3 qui sont les patrons pour lesquels la diminution du nombre de quarts proposés est la plus forte. Cependant, si on regarde la différence de coûts pour ces deux patrons dans le tableau 5.2, on constate que le coût n'augmente que de $1,8\%$ ce qui reste acceptable.

On observe enfin que le nombre de modifications pour le patron 4 est plus élevé que pour les autres patrons. Nous pouvons supposer que cela vient du fait que le nombre de quarts modifiables est beaucoup plus faible pour le patron 4. De plus, nous pouvons supposer que les modifications de quarts se font principalement les jours de modification de la demande. Ainsi, le ratio est forcément plus faible pour les patrons 2 et 3 par rapport aux patrons 1 et 4. Cette hypothèse peut être vérifiée en ne considérant que les jours où la demande est modifiée. On obtient alors les ratios suivants pour le nombre de quarts modifiés pour la proposition *trans2* :

- Patron 1 : 55,2%
- Patron 2 : 53,9%
- Patron 3 : 62,5%
- Patron 4 : 48,1%

On constate ici que les ratios sont tous assez proches. Le patron 4 est devenu le patron avec le ratio le moins élevé.

Étudions de plus près le nombre de modifications effectuées hors des jours de perturbation de la demande. Dans le tableau 5.4, nous présentons le taux de quarts qui sont modifiés les jours de non modification de la demande par rapport au nombre total de quarts modifiés.

Tableau 5.4 Nombre de quarts modifiés les jours non modifiés (% des quarts modifiés).

Propositions			Pat. 1	Pat. 2	Pat. 3	Pat. 4	**Moyenne**
init			-	-	-	-	**-**
exact			36,9	69,9	68,7	0	**43,9**
fixed			0	0	0	0	**0**
*trans*1			1,8	28,3	24,9	0	**13,7**
*trans*2	-	-	1,6	26,7	23,5	0	**13,0**
		genOS	1,6	27,2	23,8	0	**13,2**
		noPen	18,2	62,7	61,8	0	**35,7**
		morePen	1,9	31,0	26,3	0	**14,8**
		doublePen	1,7	27,3	23,4	0	**13,1**
		dayOfMod	0	0	0	0	**0**
		modJobs	1,7	29,5	22,4	0	**13,4**
lessReplaced			1,7	27,4	24,4	0	**13,4**
*trans*2	*lessJobs*	-	2,4	30,2	32,9	0	**16,4**
		modJobs	4,1	44,8	40,9	0	**22,5**

Évidemment, pour le patron 4, aucun quart n'est modifié les jours de non modification de la demande puisque le dimanche, jour de modification est le seul jour où les quarts sont modifiables. De même, les propositions *fixed* et *trans2_dayOfMod* ont toutes les deux des ratios nul car aucun quart n'est modifiable les jours où la demande est perturbée dans leurs cas.

Exact et *trans2_noPen* se distinguent des autres propositions de par leurs ratios nettement plus élevés. Cela s'explique une nouvelle fois par les symétries présentes dans le problème et la non-pénalisation de la modification dans ces deux propositions.

Les autres propositions ont toutes des ratios très proches : 1,7% pour le patron 1 environ, 28% pour le patron 2 et 23% pour le patron 3. Si nous regardons de plus près les quarts modifiés, nous nous rendons compte que pour toutes les propositions, il y a exactement :

— 1 quart réduit d'une période pour le patron 1.
— 5 quarts tous réduits d'une période pour les patrons 2 et 3.

Les ratios sont alors plus ou moins élevés en fonction du nombre total de quarts modifiés. Le

fait que ce soit des modifications peu importantes explique le faible impact de la suppression de ces quarts sur le coût de *trans2_dayOfMod*.

5.3.2 Types de transformations

Il peut également être intéressant d'étudier quels types de transformations sont majoritairement utilisés dans la solution optimale. Nous présentons les taux dans le tableau 5.5.

Tableau 5.5 Type des transformations pour les quarts modifiés (%).

Propositions			Nature de la transformation							tot.
			red	*ext*	*jobMod*	*shift.*	*J&H.*	*repl.*	*rest*	
init			-	-	-	-	-	-	-	-
exact			2,6	2,6	1,2	2,3	5,3	67,2	8,8	100
fixed			-	-	-	-	-	-	-	-
*trans*1			36,7	33,6	9,3	-	-	9,0	9,6	100
*trans*2	-	-	31,4	22,9	2,8	0	25,7	8,5	8,6	100
		genOS	31,4	23,6	2,7	0	25,4	8,3	8,6	100
		noPen	28,3	23,9	3,4	0,0	28,9	12,9	2,5	100
		morePen	33,6	23,1	2,9	0	22,0	5,1	13,3	100
		doublePen	31,6	25,6	3,2	0	21,3	9,0	9,3	100
		dayOfMod	21,4	25,9	3,1	0	29,5	9,8	10,3	100
		modJobs	33,3	22,9	2,9	0	23,3	7,6	10,1	100
lessReplaced			33,3	22,9	3,1	0	29,4	1,1	10,3	100
*trans*2	*lessJobs*	-	41,1	22,3	1,6	0,0	22,9	6,4	5,6	100
		modJobs	55,0	14,9	2,2	0	15,3	3,7	8,9	100

La proposition *exact* a une forte prédominance de quarts de type *replaced*. Une nouvelle fois, ceci est dû à la non prise en compte de l'horaire initial lors de la ré-optimisation pour cette proposition.

Pour les autres propositions, on constate que les trois types de transformations majoritaires sont la réduction (*red*), l'allongement (*ext*) et la combinaison activité-heure (*J&H*).

On pourrait expliquer le fort taux de *J&H* par le fait que la structure de coûts tel que nous l'avons imaginée ne pénalise pas le fait de choisir une combinaison de transformations plutôt qu'une transformation unique. Ainsi, si on souhaite allonger ou raccourcir le quart, le fait de changer ou non d'activité n'a pas d'influence sur le coût du quart. Si il existe des symétries dans le problème, il est possible que certaines activités soient modifiées alors que ce n'est pas nécessaire. Cependant, nous pouvons constater en regardant les valeurs pour *doublePen* que cette explication ne peut pas être satisfaisante. En effet, pour *doublePen*, le taux de *J&H* diminue un peu au profit notamment de *ext* et aussi un peu de *jobMod* mais ce taux reste très élevé.

5.4 Sur-couverture et sous-couverture

On souhaite étudier le taux de sur-couverture et de sous-couverture dans les différentes solutions. La sous-couverture étant interdite lors de l'optimisation, elle est représentée par les quarts anonymes. Cependant, la génération de quarts anonymes est restreinte par des contraintes notamment de durée minimale et maximale. Ainsi, il est possible qu'un quart anonyme couvre des périodes qui ne sont pas réellement en sous-couverture. Nous calculons donc aussi la sous-couverture réelle (couverture des employés par rapport à la demande) et nous évaluons son coût en prenant la structure de coûts de sur-couverture.

Dans nos tableaux, nous présentons le pourcentage d'heures de sur- et sous-couverture par rapport au nombre d'heures travaillées (temps) mais aussi la place que prennent les coûts de sur- et de sous-couverture dans le coût total de la solution (coût).

5.4.1 Sous-couverture et quarts anonymes

Les résultats pour la sous-couverture et les quarts anonymes sont présentés dans le tableau 5.6. Nous calculons aussi la différence en pourcentage entre le taux de sous-couvertures et le taux de quarts anonymes (colonne Diff.).

Tableau 5.6 Sous-couvertures (SC) et Quarts anonymes (QA) (%).

Propositions			Coût			Temps		
			SC	QA	Diff. (%)	SC	QA	Diff. (%)
init			0	0	–	0	0	–
exact			0,1	0,1	18,2	0,0	0,0	18,5
fixed			11,8	17,0	44,4	1,8	2,7	45,8
*trans*1			9,6	11,3	18,6	1,1	1,4	18,8
*trans*2	-	-	5,7	6,6	17,4	0,6	0,7	17,8
		genOS	5,4	6,2	15,1	0,6	0,7	15,8
		noPen	5,7	6,6	15,9	0,6	0,7	16,4
		morePen	7,0	8,2	17,1	0,8	0,9	17,7
		doublePen	5,6	6,6	18,0	0,6	0,7	18,4
		dayOfMod	5,6	6,6	17,5	0,6	0,7	17,9
		modJobs	7,9	9,0	14,3	0,9	1,0	14,9
lessReplaced			7,2	8,5	17,6	0,8	1,0	18,2
*trans*2	*lessJobs*	-	0,3	0,3	14,0	0,0	0,0	13,9
		modJobs	7,7	8,3	7,8	0,8	0,9	8,2

Comme attendu, le nombre de périodes de quarts anonymes est plus élevé que le nombre de périodes réellement en sous-couverture. Il en va bien entendu de même pour les coûts. Si on

calcule la différence, on obtient à peu près le même ratio sauf pour *fixed* et *trans2_lessJobs* à savoir entre 15% et 18% plus de quarts anonymes que de sous-couverture. La différence est plus grande pour *fixed* parce que l'horaire est fixé. Ainsi, l'ensemble de la sous-couverture engendrée par la modification de la demande doit être couverte par les quarts anonymes sans pouvoir arranger les quarts des employés pour diminuer le nombre de quarts anonymes nécessaires. Ainsi, il est possible qu'on doive ajouter un quart anonyme de 4h (durée minimale) pour une seule période de sous-couverture isolée qu'on aurait normalement couverte avec un quart d'employé allongé.

On constate que les propositions qui ont des coûts plus élevés sont aussi celles qui ont le plus de sous-couverture. C'est notamment le cas de *trans2_modJobs* et *trans2_morePen*. Cela s'explique par le fait que les modifications des quarts des employés sont plus dissuadées dans ces propositions soit par le fait qu'on propose moins de quarts au choix soit par le fait que les pénalités de modification sont plus élevées. On constate cependant qu'il y a toujours (sauf pour *exact*) des quarts anonymes dans la solution optimale alors qu'il n'y en avait pas dans la solution initiale et qu'il n'y en a pas (ou très peu) dans la solution *exact*. Cela signifie que nos propositions de quarts ne sont pas suffisantes pour éviter les quarts anonymes.

On observe également que plus on propose de quarts, moins on doit faire appel aux quarts anonymes ce qui semble logique. Ainsi, si on note $N_{qa}(x)$ le nombre de périodes de quarts anonymes pour une instance avec la proposition x, on a :

$$N_{qa}(trans1) \leq N_{qa}(lessReplaced) \leq N_{qa}(trans2)$$

5.4.2 Sur-couverture

Les résultats pour la sur-couverture sont dans le tableau 5.7. On calcule la sur-couverture réelle mais aussi la sur-couverture en enlevant les quarts anonymes pour vérifier si l'interdiction de la sous-couverture et la production de quarts anonymes induit une forte augmentation de la sur-couverture dû à la couverture non nécessaire de certaines périodes par ces quarts.

On constate que les quarts anonymes n'induisent pas un nombre de sur-couvertures vraiment plus élevé. Le nombre de sur-couvertures est assez constant avec les propositions. *Fixed* mis à part, il tourne autour de 25% pour le coût et de $2,3\%$ pour le temps ce qui est aussi le cas pour la solution initiale. Comme le taux de sur-couverture n'augmente pas beaucoup lors de la ré-optimisation par rapport à la solution initiale, nous pouvons avancer que nos propositions sont performantes sur ce paramètre.

Le taux de sur-couverture élevé peut être expliqué par le fait que la demande n'est pas lisse.

Tableau 5.7 Sur-couvertures (%).

Propositions			Coût		Temps	
			avec QA	sans QA	avec QA	sans QA
init			23,7	23,7	2,0	2,0
exact			26,0	25,9	2,2	2,1
fixed			33,4	28,0	4,6	1,9
*trans*1			23,9	22,0	2,4	1,9
*trans*2	-	-	24,5	23,4	2,3	1,9
		genOS	24,4	23,5	2,2	1,9
		noPen	24,3	23,3	2,2	2,0
		morePen	24,8	23,5	2,4	1,9
		doublePen	24,5	23,5	2,3	1,9
		dayOfMod	25,2	24,1	2,4	2,0
		modJobs	24,0	22,8	2,3	1,9
lessReplaced			24,7	23,3	2,3	2,0
*trans*2	*lessJobs*	-	24,5	24,4	2,1	1,9
		modJobs	23,3	22,7	2,2	1,9

Ainsi, il n'est pas rare d'avoir des créneaux comme nous avons pu le voir sur la figure 4.1. Comme la sous-couverture est interdite, il faut obligatoirement couvrir les périodes dont la valeur de la demande est la plus élevée avec des quarts contraints par une durée minimale créant nécessairement des sur-couvertures.

5.5 Conclusion

En conclusion de l'analyse de ces résultats, nous pouvons observer qu'il y a un arbitrage que l'entreprise doit faire entre le coût de la solution finale et la flexibilité. En effet, le degré de flexibilité varie beaucoup entre les propositions de *fixed* où aucune modification n'est acceptée à *exact* où l'on est prêt à accepter toutes les modifications qui feraient diminuer le coût. De même, le fait de pénaliser davantage les modifications (*trans2_morePen*) implique un nombre de modifications dans la solution finale moins élevé. C'est à l'utilisateur de déterminer à quel point l'entreprise peut se permettre de faire des modifications dans l'horaire des employés et quelles transformations sont acceptables. L'utilisateur peut ainsi jouer sur deux paramètres pour la flexibilité de la ré-optimisation : la valeur des pénalités et les transformations autorisées. Plus il y a de flexibilité, plus le coût de la solution finale sera faible.

Comme nous l'avons vu avec la figure 5.3, il y a également un arbitrage à effectuer entre le coût et le temps de résolution, le temps de résolution étant intimement relié au nombre de

quarts proposés. En effet, on peut avec certaines propositions, gagner en qualité au niveau coût mais au prix d'une grosse perte en temps de calcul. Il faut donc décider quel temps on est prêt à allouer au calcul de l'horaire mis à jour.

Les propositions qui semblent intéressantes au niveau du temps de calcul et du coût de la solution sont *trans2* et *trans2_dayOfMod*. Il peut donc être intéressant d'exploiter ces propositions.

CHAPITRE 6 CONCLUSION

Lors de ce projet de maîtrise, nous avons proposé des solutions pour la mise à jour des horaires de personnel travaillant sur des quarts. Nous concluons ici par une synthèse des travaux (section 6.1), une présentation de leurs limites (section 6.2) et des perspectives d'améliorations futures (section 6.3).

6.1 Synthèse des travaux

Nous avons proposé de résoudre le problème de mise à jour des horaires de personnel de manière exacte avec un modèle explicite. Afin de pouvoir mettre notre modèle en application et de pouvoir tester les solutions proposées, nous avions un premier travail de préparation des données à effectuer. Nous devions ainsi :

- **Générer des scénarios de demande.** Afin de pouvoir tester nos propositions, nous avions besoin d'un nombre représentatif de scénarios de demande. Les données auxquelles nous avions accès étant limitées, nous avons dû développer une méthode pour générer nous-mêmes des scénarios. Nous avons défini différents patrons de modification de la demande qui nous semblaient représentatifs des situations qui peuvent se présenter dans la vie réelle puis nous y avons introduit des paramètres aléatoires afin de pouvoir générer autant de scénarios que nécessaire.
- **Générer des quarts transformés.** Nous proposons un modèle explicite pour lequel nous devons générer les quarts que nous allons proposer comme variables. Nous avons donc proposé différentes transformations possibles puis différentes propositions avec plus ou moins de transformations autorisées et donc plus ou moins de quarts proposés.
- **Établir une structure de coûts.** Nous avons adapté la structure de coût dans une optique de ré-optimisation. Pour cela, nous avons gardé les coûts utilisés lors de l'optimisation initiale et nous y avons ajouté des pénalités de modification cherchant à éviter une trop grande modification de l'horaire.

Pour la génération des quarts transformés et la structure de coûts, nous avons veillé à ce que les propositions soient les plus souples possibles pour que l'utilisateur puisse les adapter selon ses souhaits.

En analysant les résultats selon plusieurs critères, nous avons pu dégager deux propositions qui ont un bon rapport temps de résolution/coût. Les résultats nous permettent surtout de conclure à la nécessité d'un arbitrage entre l'importance de la modification de l'horaire et le coût. Pour obtenir le coût réel le plus faible possible, il faut être prêt à obtenir un

horaire avec d'importantes modifications. Cependant, il ne faut pas perdre de vue que le fait de modifier un horaire a aussi un coût indirect sur l'entreprise. Changer en permanence les horaires risque en effet de mécontenter les employés et d'engendrer une moindre productivité au sein de l'entreprise.

6.2 Limitations de la solution proposée

Une des limitations de la solution proposée est qu'il est nécessaire de faire un arbitrage entre le coût et le temps de résolution. En effet, ici, plus nous réussissons à diminuer le temps de résolution, plus nous avons de la difficulté à atteindre une bonne qualité de la solution au niveau du coût.

De plus, nous avons ici effectué des tests avec un jeu de données de seulement 49 employés. Dans un contexte avec plus d'employés (100 à 150), il est vraisemblable que le temps de résolution augmente de manière inacceptable. Il serait donc peut-être intéressant de développer une heuristique permettant de résoudre le problème en un temps plus court pour les problèmes plus gros.

Une autre limitation est que la solution n'a pas été testée dans un contexte réel. En effet, nos propositions ont été élaborées selon des suppositions théoriques notamment sur les possibilités de transformations. Il serait intéressant pour confirmer nos résultats de les faire tester par un utilisateur réel qui règlerait les transformations autorisées et la valeur des pénalités en fonction de de ses intérêts et de la règlementation à respecter.

6.3 Améliorations futures

Afin de répondre à la première limitation présentée, une des améliorations futures pourrait être de faire des recherches plus poussées sur les quarts effectivement utilisés dans la solution optimale avec d'autres critères. Ainsi, nous pourrions par exemple étudier la longueur des transformations quand le quart est allongé ou raccourci ou encore tenter de relier les transformations subies à la variation de la demande. Cela permettrait peut-être de réduire le temps de calcul tout en gardant la qualité de la solution.

Afin de se rapprocher de la réalité, une des améliorations pourrait également être de faire les mêmes recherches dans un contexte multi-activités. Il faudrait pour cela redéfinir les transformations autorisées puis vérifier que nos propositions restent performantes. L'agrégation avec le placement des pauses pourrait également être une piste de recherche intéressante. Enfin, il serait intéressant de prendre en compte les préférences des employés. Pour cela,

l'exploitation des pénalités de modification pourrait ici être une piste intéressante. En effet, des pénalités de modification individualisées pourraient être mises en place en fonction des préférences exprimées par chacun des employés.

Enfin, il pourrait être intéressant, à titre d'information, de faire plus de tests avec des pénalités différentes afin de pouvoir établir le lien entre le nombre de modifications effectuées et la valeur des pénalités. Cela permettrait à l'utilisateur de pouvoir définir les coûts de manière avisée.

RÉFÉRENCES

T. Aykin, “Optimal Shift Scheduling with Multiple Break Windows”, *Management Science*, vol. 42, no. 4, pp. 591–602, Avril 1996.

S. E. Bechtold et L. W. Jacobs, “Implicit Modeling of Flexible Break Assignments in Optimal Shift Scheduling”, *Management Science*, vol. 36, no. 11, pp. 1339–1351, Novembre 1990.

M. Bouchard, “Coloration de graphes et attribution d’activités dans des quarts de travail”, Thèse de doctorat, École Polytechnique de Montréal, 2008.

M.-C. Côté, B. Gendron, et L.-M. Rousseau, “Modeling the Regular Constraint with Integer Programming”, dans *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, vol. 4510, 2007, pp. 29–43.

——, “Grammar-Based Integer Programming Models for Multiactivity Shift Scheduling”, *Management Science*, vol. 57, pp. 151–163, January 2011.

——, “Grammar-Based Column Generation for Personalized Multi-Activity Shift Scheduling”, *INFORMS Journal on Computing*, vol. 25, no. 3, pp. 461–474, 2012.

S. Dahmen et M. Rekik, “Solving multi-activity multi-day shift scheduling problems with a hybrid heuristic”, *Journal of Scheduling*, vol. 18, no. 2, pp. 207–223, April 2015.

G. B. Dantzig, “A Comment on Edie’s "Traffic Delays at Toll Booths"”, *Journal of the Operations Research Society of America*, vol. 2, no. 3, pp. 339–341, Août 1954.

S. Demassey, G. Pesant, et L.-M. Rousseau, “Constraint Programming based Column Generation for Employee Timetabling”, dans *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, R. Barták et M. Milano, éds., 2005.

L. C. Edie, “Traffic Delays at Toll Booths”, *Journal of the Operations Research Society of America*, vol. 2, no. 2, pp. 107–138, Mai 1954.

A. Ernst, H. Jiang, M.Krishnamoorthy, et D.Sier, “Staff scheduling and rostering : A review of applications, methods and models”, *European Journal of Operational Research*, vol. 153, pp. 3–27, 2004.

A. Legrain, “Génération de scénarios pour la demande en personnels durant plusieurs périodes”, Mémoire de maîtrise, Ecole Polytechnique de Montréal, Décembre 2011.

Q. Lequy, G. Desaulniers, et M. M. Solomon, “A two-stage heuristic for multi-activity and task assignment to work shifts”, *Computers & Industrial Engineering*, vol. 63, no. 4, pp. 831–841, December 2012.

M. Moz et M. V. Pato, “An Integer Multicommodity Flow Model Applied to the Rerostering of Nurse Schedules”, *Annals of Operations Research*, vol. 119, pp. 286–301, 2003.

Z. Omari, “Attribution des activités aux employés travaillant sur des quarts”, Mémoire de maîtrise, École Polytechnique de Montréal, 2002.

S. Proulx, “Génération de scénarios par quantification optimale en dimension élevée”, Mémoire de maîtrise, École Polytechnique de Montréal, Mai 2014.

C.-G. Quimper et L.-M. Rousseau, “A Large Neighbourhood Search Approach to the Multi-Activity Shift Scheduling Problem”, *Journal of Heuristics*, vol. 16, no. 3, pp. pp 373–392, June 2010.

M. Rekik, J.-F. Cordeau, et F. Soumis, “Implicit shift scheduling with multiple breaks and work stretch duration restrictions”, *Journal of Scheduling*, vol. 13, no. 1, pp. 49–75, Février 2010.

M. Segal, “The Operator-Scheduling Problem : A Network-Flow Approach”, *Operations Research*, vol. 22, no. 4, pp. 808–823, Juillet-Août 1974.

J. Van den Bergh, J. Beliën, P. D. Bruecker, E. Demeulemeester, et L. D. Boeck, “Personnel scheduling : A literature review”, *European Journal of Operational Research*, vol. 226, pp. 367–385, 2013.

E. Vatri, “Intégration de la génération de quarts de travail et de l’affectation d’activités”, Mémoire de maîtrise, Université de Montréal, 2001.

ANNEXE A STRUCTURE DE COÛTS

La structure des coûts pour l'optimisation est celle utilisée par le logiciel actuel. Les coûts se divisent en trois parties :

- **Le coût de la main d'oeuvre.** Cela représente le coût pour l'entreprise d'embaucher un employé pour une période.
- **Le coût des quarts anonymes.** Cela représente le coût d'utiliser un quart anonyme pour couvrir une période. Les quarts anonymes sont souvent travaillés ensuite par des employés surnuméraires qui coûtent plus cher et on souhaite donc en avoir le moins possible d'où leur coût élevé.
- **Le coût de la sur-couverture.** Ce coût pénalise le fait qu'une période soit en sur-couverture. On veut éviter le plus possible la sur-couverture et les coûts sont donc plutôt élevés.

Coût de la main d'oeuvre

Le coût linéaire par morceaux pour chaque période d'affectation d'un employé à un quart est donné par :

1. Huit premières heures d'affectation dans la semaine = $75\$/période$
2. Plus de 8 et jusqu'à 16 heures d'affectation dans la semaine = $86,1\$/période$
3. Plus de 16 et jusqu'à 24 heures d'affectation dans la semaine = $99\$/période$
4. Plus de 24 et jusqu'à 32 heures d'affectation dans la semaine = $113,7\$/période$
5. Plus de 32 et jusqu'à 40 heures d'affectation dans la semaine = $130,65\$/période$
6. Chaque période après 40 heures d'affectation dans la semaine = $150\$/période$

Augmenter les coûts en fonction du temps travaillé dans la semaine permet d'assurer une équité parmi les employés. En effet, cela tend à l'égalisation pour tous les employés du nombre d'heures travaillées dans la semaine. Cela coûtera en effet moins cher d'avoir deux employés travaillant chacun 8h plutôt qu'un employé travaillant 16h et un autre ne travaillant pas.

Coût des quarts anonymes

Pour chaque période de l'horizon, il y a un coût linéaire par morceaux pour chaque quart anonyme couvrant cette période. Pour une période donnée on a donc :

1. Premier quart utilisé = 1500$
2. Deuxième quart utilisé = 1723,05$
3. Troisième quart utilisé = 1979,25$
4. Quatrième quart utilisé = 2273,55$
5. Cinquième quart utilisé = 2611,65$
6. Sixième et septième quart utilisé = 3000$
7. Huitième et neuvième quart utilisé = 4846,2$
8. Dixième et onzième quart utilisé = 7828,5$
9. Douzième et treizième quart utilisé = 12646,05$
10. Quatorzième et quinzième quart utilisé =20428,5$
11. Chaque quart supplémentaire utilisé à partir du seizième quart utilisé = 33000$

Coût de sur-couverture

Il y a un coût linéaire par morceaux pour chaque période de sur-couverture d'une activité. Pour une période et une activité données on a donc :

1. Premier employé en sur-couverture = 1500$
2. Deuxième employé en sur-couverture = 1889,85$
3. Troisième employé en sur-couverture = 2381,1$
4. Quatrième, cinquième et sixième employés en sur-couverture = 3000$
5. Septième employé en sur-couverture = 3519,9$
6. Huitième et neuvième employés en sur-couverture = 3779,7$
7. Dixième employé en sur-couverture = 4102,8$
8. Onzième, douzième et treizième employés en sur-couverture = 4762,2$
9. Chaque employé supplémentaire en sur-couverture à partir du quatorzième employé en sur-couverture = 6000$

On constate que les coûts de quarts anonymes et de sur-couverture sont beaucoup plus élevés que les coûts de la main d'oeuvre ce qui confirme le désir de les éviter au maximum.