POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Méthodes heuristiques de planification et de ré-optimisation en temps réel pour les problèmes d'horaires de personnel

RACHID HASSANI
Département de mathématiques et de génie industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Mathématiques

Décembre 2019

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Méthodes heuristiques de planification et de ré-optimisation en temps réel pour les problèmes d'horaires de personnel

présentée par **Rachid HASSANI**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Michel GENDREAU, président
Issmail EL HALLAOUI, membre et directeur de recherche
Guy DESAULNIERS, membre et codirecteur de recherche
Antoine LEGRAIN, membre
Djamal REBAÏNE, membre externe

DÉDICACE

À l'âme de mon père : M'Hammed HASSANI
tu me manques...

CITATION

«Quand on me contredit, on éveille mon attention, mais non ma colère : je m'avance vers celui qui me contredit, qui m'instruit. La cause de la vérité devrait être la cause commune de l'un et de l'autre.»

MONTAIGNE (1572-1592). Essais, III.

REMERCIEMENTS

Je tiens tout d'abord à remercier mes deux directeurs de recherche : Issmail El Hallaoui et Guy Desaulniers. Ils m'ont accueilli au sein de leur équipe de recherche et ont su, malgré leurs responsabilités, me donner de leur temps et de leur patience. Ils m'ont aussi beaucoup aidé lors de la rédaction de cette thèse et de ces articles de recherche. C'est grâce à leurs consignes, recommandations, conseils et encouragements que j'ai pu finir cette thèse à temps. Je me considère très chanceux d'avoir été encadré par eux.

J'exprime également ma reconnaissance et ma considération envers François Soumis avec qui j'ai échangé à plusieurs reprises durant le déroulement de ma thèse.

Je remercie les membres du jury, Michel Gendreau, Antoine Legrain et Djamal Rebaïne, qui m'ont fait l'honneur de juger ce travail doctoral.

Je remercie également le CRSNG et KRONOS Inc. qui ont participé au financement de mon projet.

J'ai eu l'opportunité de côtoyer pendant ces quatre années des femmes et des hommes qui, en plus d'être très compétents, sont particulièrement humains et serviables. Je les remercie pour leur contribution au bon déroulement de cette thèse. Un remerciement spécial à Adil, Chahid, Safae, Bani, Ilyas, Omar, Mayssoun, Tayeb, Salah, Issoufou, Fillipo, Dalia et Hélène.

Merci à certains professeurs que j'ai croisés au cours de ma scolarité, qui m'ont marqué sur les plans personnel et professionnel. Je pense plus particulièrement à Ahmed Akrkouz, Abdelwahed Samid et Hassan Elidrissi.

Merci aussi à mes plus anciens amis : les deux Mehdis, Amine, Mkireb, Yves, Ivan, Victoria, Awatif, Hamza et Anouar. Merci plus particulièrement à Mounia pour son aide inestimable, son soutien permanent et pour m'avoir remonté le moral dans les périodes les plus difficiles, quand le doute s'installait.

Un énorme merci à mes frères et sœurs : Fatima, Mohammed, Hind, Atif et Hasna, qui ont contribué à leur façon pour me soutenir. Une mention toute particulière à Fatima, qui a bien jouée le rôle d'une grande sage sœur, et parfois, le rôle d'une mère et d'un père pour des bonnes périodes de ma vie : c'est grâce à elle que j'ai pu mener un tel parcours scolaire.

Finalement, je souhaite finir par le remerciement le plus grand pour mon épouse Hanae, mon père M'Hammed et ma mère Khadija. Ma chère Hanae, un grand merci pour ton soutien, ton encouragement et ton amour. Merci d'avoir été à mes côtés dans toutes les phases de ces quatre années : des hauts, des bas et souvent des entre-deux. Merci infiniment, mes chers

parents, pour tout ce que vous avez fait pour moi. Merci d'avoir toujours pensé en priorité à ma réussite scolaire et d'avoir toujours œuvré pour que je ne manque de rien.

RÉSUMÉ

Les problèmes d'horaires de personnel visent à déterminer les horaires de travail les moins coûteux pour couvrir la demande d'une ou de plusieurs tâches à chaque période d'un horizon donné. Ces problèmes font de plus en plus l'objet d'études traitées par la recherche opérationnelle. Cela est dû, d'une part, aux besoins économiques, qui révèlent jour après jour de nouveaux défis, et, d'autre part, à la complexité qui caractérise de tels problèmes. Parmi les grands défis, on retrouve la gestion de l'incertitude en temps réel. En effet, durant l'opération, plusieurs perturbations, difficilement contrôlables, peuvent survenir à tout moment, comme les retards des employés, la variation de la demande, etc. Ces perturbations doivent être traitées intelligemment et très rapidement par la mise à jour des horaires de certains employés. Dans cette thèse, on s'intéresse principalement à cette problématique dans un contexte très peu abordé dans la littérature, à savoir la vente au détail, où les employés peuvent être affectés à une grande variété de quarts de travail selon leurs qualifications et leurs disponibilités.

Nous proposons, en premier lieu, une méthode de ré-optimisation en temps réel qui se base principalement sur l'information duale, trouvée lors de l'optimisation initiale, et sur un ensemble de types de décision recommandés par notre partenaire industriel. La méthode détermine, pour chaque type de décision, la meilleure adaptation de l'horaire en estimant un coût ajouté pour chaque adaptation candidate. Les adaptations concernent seulement le jour où la perturbation a eu lieu et visent à palier la perturbation tout en gardant une solution globale quasi-optimale. Nous proposons également une procédure, exploitant une méthode de régression, pour mettre à jour les valeurs duales après la correction de chaque perturbation lorsque plusieurs d'entre elles surviennent durant l'horizon. Les tests menés sur un ensemble contenant 1050 instances de problèmes réels allant jusqu'à 191 employés ont montré l'efficacité de la méthode proposée. Celle-ci arrive à trouver la solution optimale pour plus de 95% de ces instances, et ce, en une seconde en moyenne.

En second lieu, nous proposons une méthode de ré-optimisation en temps réel plus générale que la première : elle ne requiert pas l'information duale et ne se restreint pas seulement au jour où la perturbation a eu lieu, mais s'applique également au reste de l'horizon. La méthode cherche des adaptations qui permettent de corriger la perturbation sans trop s'éloigner de l'horaire planifié en termes du nombre de modifications engendrées par ces adaptations. Le coût ajouté dû à ces modifications doit être le moins élevé possible. À l'aide de certaines décisions de base qui permettent de générer tous les horaires possibles, la méthode cherche

les bonnes séquences, appelées politiques, qui conduisent à des solutions de bonne qualité, appelées solutions non-dominées, obtenues après une optimisation multi-objectif. La méthode fait appel à un réseau qui schématise une région prometteuse de l'enveloppe convexe des solutions réalisables. La construction du réseau se fait d'une manière dynamique au cours de la résolution en se basant sur quelques résultats théoriques. Les tests menés sur des instances dérivées de données réelles impliquant jusqu'à 95 employés démontrent l'efficacité de l'heuristique. En moins d'une seconde en moyenne, elle peut calculer des solutions Pareto-optimales pour plus de 98% des scénarios testés.

L'efficacité et la rapidité de cette dernière méthode de ré-optimisation nous a poussé à être plus ambitieux pour l'adapter afin de faire une optimisation globale. Ceci a abouti à la naissance d'une nouvelle approche intégrée, qui génère et affecte les quarts simultanément et qui vise à optimiser un problème d'horaires de personnel. L'approche se base sur la stimulation/correction itérative d'un ensemble de perturbations bien ciblées selon certaines mesures probabilistes. Ces dernières indiquent la probabilité que la correction de chacune de ces perturbations améliore significativement la solution courante. Deux procédures de groupage sont utilisées afin de générer un ensemble de sous-problèmes qui seront traités parallèlement. Chacun de ces sous-problèmes représente un terrain riche de perturbations visées pour les stimuler, ce qui rend la recherche plus efficace et rapide. Pour chaque sous-problème, nous stimulons une perturbation qui nous mène, généralement, à l'espace irréalisable. Ainsi, nous définissons un voisinage de recherche, sur ce sous-problème, afin de trouver une politique qui nous mène à l'espace réalisable. Nous combinons à chaque itération, de façon optimale, toutes les politiques qui sont compatibles (i.e., leur combinaison produit une solution réalisable) parmi celles trouvées parallèlement dans les sous-problèmes considérés afin de passer à une solution meilleure. L'heuristique proposée a été testée sur des instances réelles du problème impliquant jusqu'à 94 employés et 10 tâches. Notre méthode a alors permis de trouver des solutions de bonne qualité en moins de 2 minutes, en moyenne, dans 97% des cas de test.

ABSTRACT

Personnel scheduling aims at determining the cheapest work schedules to cover the demand for one or more tasks at each period of a given horizon. These problems are increasingly addressed in operations research. This is due, on the one hand, to the economic needs that reveal day by day new challenges, and on the other, the complexity that characterizes such problems. One of the biggest challenges is to manage uncertainty in real time. In fact, during the operation, several disruptions, difficult to control, can occur at any time such as employee delays, demand variation, etc. These disruptions must be handled intelligently and quickly by updating the schedules of certain employees. In this thesis, we are mainly interested in personnel scheduling in a context that is not tackled much in the literature where employees can be assigned to a large variety of shifts depending on their qualifications and availability.

Firstly, we propose a real-time re-scheduling method based on the dual information found during the initial optimization and on a set of decision types provided by our industrial partner. This method determines for each type of decision the best adaptation of the schedule by estimating an additional cost for each possible adaptation. The adaptations are intended to cover the disruption, that occurs during the day, and to keep a quasi-optimal solution. We also propose a procedure using a regression method to update the dual values after the correction of each disruption when several of them occur during the horizon. Computational experiments conducted on a set of 1050 instances derived from real-life datasets involving up to 191 employees show the efficiency of the proposed re-scheduling heuristic: it can compute optimal solutions for more than 95% of these instances in less than one second on average.

Secondly, we propose a real-time re-scheduling method which is more general than the first one. It does not require the dual information and is not limited to the day when the disruption takes place but covers all the rest of the horizon. The method looks for adaptations that enable to correct the disruption without deviating too much from the planned schedule in terms of the number of modifications caused by these adaptations. The additional cost caused by these changes must be as low as possible. Using some basic decisions that can generate all possible schedules, our method aims to find the right sequences, called policies, that lead to non-dominated solutions, obtained after a multi-objective optimization. The method uses a network that schematizes a promising region of the convex hull of the set of feasible solutions. The network's construction is done dynamically during the solution process based on some theoretical results. Computational experiments conducted on a set of instances derived from real-life datasets involving up to 95 employees showed the usefulness of the fundamental

study as well as the efficiency of the proposed heuristic. It can find all the exact solutions that achieve a good compromise cost/modifications, in a search zone set by the employer, in less than one second on average for more than 97% of the scenarios generated.

The efficiency and the speed of the last re-scheduling method encouraged us to integrate it into a global optimization algorithm. As a result, a new integrated approach has been developed which generates and assigns shifts simultaneously and optimizes the personnel scheduling problem. The approach is based on the iterative stimulation/correction of a set of well-targeted disruptions according to certain probabilistic measures. These indicate the probability that the correction of each of these disruptions significantly improves the current solution. Two clustering procedures are used to generate a set of subproblems that are processed in parallel. Each of these subproblems represents a rich area of disruptions intended to stimulate them and what makes the search more efficient and faster. For each subproblem, we stimulate a disruption that generally leads us to the infeasible region. So, we define a search neighborhood in order to find a policy that leads to the feasible region. At each iteration we combine all policies that are compatible (i.e. their combination leads to a feasible solution) among those found in parallel by the subproblems. The proposed heuristic has been tested on real instances of the problem involving up to 94 employees and 10 tasks. The method then made it possible to find good quality solutions in less than 2 minutes, on average, in 97% of the test cases.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Les problèmes liés à la gestion du personnel sont fréquents dans de nombreux domaines tels que la vente au détail, le transport, la santé ou la production industrielle. Souvent, les entreprises gèrent un grand nombre d'employés possédant diverses qualifications et disponibilités. Produire un horaire adapté permettant de satisfaire la demande tout en respectant les contraintes syndicales et législatives peut être un exercice difficile. On compte de nombreuses variantes du problème selon l'environnement dans lequel on se place. En effet, il existe deux types d'environnement : l'environnement non continu, où on travaille la journée avec des heures fixes d'ouverture et de fermeture (magasins à grande surface), et un environnement continu, où le travail s'effectue 24 h/24 h (hôpitaux, usines). Le domaine auquel on s'intéresse est celui de la vente au détail qui fait partie de l'environnement non continu. Dans un tel champ d'expertise, les dépenses liées à la masse salariale constituent les charges majeures de fonctionnement pour une entreprise. Puisque la diminution de ces charges a un effet direct sur la baisse des prix, la concurrence aiguë pousse les entreprises à s'intéresser de plus en plus au développement de nouveaux outils pour améliorer leur gestion du personnel.

1.1 Définitions et concepts de base

Quelques jours à l'avance (voire quelques mois), l'employeur doit être en mesure d'informer ses employés sur leurs jours de travail et de congé ainsi que les tâches qu'ils auront à effectuer pour un horizon donné (généralement une semaine). L'employeur, dans sa planification des horaires, cherche à minimiser les dépenses de l'entreprise liées à la masse salariale tout en prenant en compte les aspects suivants :

- *La satisfaction de la demande* : face à la demande, l'entreprise doit pouvoir répondre à la clientèle par un nombre raisonnable d'employés. S'il y en a trop peu, la demande ne sera évidemment pas satisfaite. La situation ne sera pas idéale non plus s'il y en a trop, car l'entreprise peut a priori en mobiliser moins, tout en gardant sa qualité de service. Dans les deux cas, elle perd de l'argent. Dans notre étude, on considère que le nombre de clients à servir peut se ramener au nombre d'employés nécessaires pour le service. La demande sera exprimée en nombre d'employés nécessaires par la suite.
- *Les contraintes des horaires de travail* : les syndicats ainsi que les lois imposent un certain nombre de contraintes liées aux horaires des employés. Parmi celles-ci, on retrouve une durée minimale et maximale pour un quart de travail (i.e., temps de travail d'un employé pour une journée), un temps de travail maximal dans la semaine, un temps de repos minimal entre deux quarts et un nombre minimal de jours de repos

dans la semaine.

— *La qualification des employés* : chaque employé possède une qualification pour la réalisation d'une tâche. L'employeur n'attribuera pas une tâche à effectuer à un employé qui n'a pas la qualification nécessaire.

— *La disponibilité des employés* : la disponibilité de chaque employé peut varier d'un jour à l'autre. Si un quart chevauche une période d'indisponibilité d'un employé, alors il ne peut être affecté à cet employé.

Lorsqu'on élabore un horaire, on peut rarement couvrir de manière exacte en tout temps la demande. Ainsi, il n'est pas rare, que pour certaines périodes courtes, il y ait trop (resp. peu) d'employés par rapport à la demande. On dit alors qu'on est en situation de sur-couverture (resp. sous-couverture). Pour établir un horaire, on peut soit interdire ce genre de situations, soit les pénaliser pour diminuer leur occurrence. Dans notre cas, nous avons choisi d'interdire la sous-couverture et de pénaliser la sur-couverture. Avec ce choix, nous prenons le risque de ne pas être en mesure d'établir un horaire réalisable s'il n'y a pas assez d'employés disponibles pour couvrir la totalité de la demande. Pour éviter cette irréalisabilité, la notion de quart anonyme est introduite. Les quarts anonymes sont des quarts qui ne sont attribués à aucun employé lors de la construction des horaires du personnel, mais qui sont pris en compte comme des quarts effectués pour répondre à la demande. Si des quarts anonymes apparaissent dans notre horaire, alors des périodes de sous-couverture sont présentes. Les quarts anonymes ont un coût plus élevé, car des travailleurs à temps partiel embauchés à l'occasion ou des employés d'un autre département qui connaît une demande moins forte pourront travailler durant ces quarts.

Tout au long de notre étude, nous ferons l'hypothèse qu'un quart est constitué d'un instant de début et d'un instant de fin sans prendre en compte les pauses. De plus, une seule tâche doit être effectuée durant ce quart. Cette hypothèse facilite la description de certaines contraintes du problème sans toutefois qu'il n'en résulte une perte de généralité.

L'objectif de l'employeur consiste à minimiser les coûts de main-d'œuvre, mais aussi les pénalités de quarts anonymes et de sur-couverture. Ces coûts ne sont pas forcément ceux payés réellement par l'entreprise, mais ils correspondent à une structure pénalisant les situations que nous ne souhaitons pas retrouver dans l'horaire.

On dispose d'une fonction linéaire par morceaux pour les coûts de main-d'œuvre, de sur-couverture et de quarts anonymes. Pour les coûts de main-d'œuvre, cette structure des coûts permet d'assurer une équité au niveau du nombre d'heures travaillées dans la semaine. En effet, il sera moins coûteux d'avoir deux employés travaillant chacun 8 heures plutôt qu'un employé travaillant 16 heures et un autre ne travaillant pas.

Si l'on suppose qu'on dispose d'un ensemble de quarts anonymes proposés, ainsi qu'un ensemble de quarts personnalisés pour chaque employé, alors, on peut voir l'horaire recherché comme une solution d'un programme linéaire en nombres entiers. Les variables de décision de ce programme permettent de sélectionner : i) pour chaque employé, certains quarts parmi ses quarts personnalisés ; ii) un ensemble de quarts anonymes parmi ceux proposés. Si ces ensembles sont construits d'une manière exacte, alors l'horaire établi est un horaire optimal. Sinon, la génération de ces ensembles se fait d'une manière heuristique ; l'horaire retourné représente un horaire non exact pour le problème réel de planification d'une qualité très dépendante de l'efficacité de l'heuristique utilisée pour générer les quarts.

1.2 Éléments de la problématique

Les problèmes de gestion du personnel sont de nature combinatoire et sont très souvent caractérisés par la présence de plusieurs contraintes conflictuelles. Si l'entreprise compte beaucoup d'employés, la résolution de ces problèmes requiert un logiciel d'optimisation. Ainsi, de nombreux projets en recherche opérationnelle sont effectués pour améliorer ces logiciels de génération d'horaires de personnel. KRONOS est une entreprise américaine qui compte parmi les grands fournisseurs de ce type de logiciels. Puisque la résolution exacte de ces problèmes complexes est très coûteuse en termes de temps de calcul, la plupart des méthodes proposées sur le marché sont des heuristiques sacrifiant la garantie d'optimalité, mais permettant d'aboutir à un horaire de très bonne qualité en un temps raisonnable. La recherche se focalise de plus en plus sur la réduction de ce laps de temps tout en gardant une très bonne qualité de l'horaire adopté au final.

Malheureusement, il est rare que l'horaire planifié résiste à l'incertitude. En effet, souvent, des employés sont en retard, s'absentent ou ne peuvent pas finir les heures de travail programmées pour un jour donné. Ces incidents, qu'on appelle petites perturbations, peuvent arriver à tout moment. Ce sont des incidents non contrôlables et qui rendent l'horaire planifié initialement non réalisable. Afin de pallier ces petites perturbations, il faut apporter des modifications. Selon Mercer [1], ces modifications peuvent représenter pour une entreprise un coût équivalant à 35% de la somme totale de la paie. Par conséquent, il est fondamental de les gérer au mieux. Pour cela, une ré-optimisation est nécessaire. En général, la ré-optimisation est l'optimisation d'une instance d'un problème qui a été légèrement perturbée et pour laquelle une solution a déjà été trouvée. Le problème de la ré-optimisation se caractérise, d'une part, par la présence d'une première solution obtenue en résolvant l'instance du problème non perturbé et, d'autre part, par des exigences en termes de temps de calcul plus contraignantes que pour l'optimisation initiale. Aussi, obtenir une solution du problème de ré-optimisation proche de la solution initiale est nécessaire. En effet, on ne peut pas changer le programme

de tous les employés seulement parce que l'un d'eux est en retard.

1.3 Objectifs de recherche

L'objectif qui nous motive dans notre recherche est d'alimenter le logiciel d'optimisation de KRONOS, appelé *Workforce scheduler*, par une méthode efficace de ré-optimisation en temps réel des horaires de personnel suite à une petite perturbation due aux incidents cités ci-dessus. La méthode devra proposer aux employeurs des choix de mises à jour des horaires et évaluer les coûts des modifications occasionnées en tenant compte des coûts immédiats (coûts de gestion) et des coûts futurs déterministes (impact des modifications sur les horaires futurs et la rémunération des employés). À notre connaissance, aucune étude académique traitant de ce problème n'a encore été faite. Les précédentes études ayant un objectif connexe étaient appliquées dans le milieu hospitalier. Nous nous plaçons ici dans un environnement non continu, à savoir le domaine du commerce de détail. Par ailleurs, les résultats du milieu hospitalier sont difficilement transposables au domaine du commerce de détail. En effet, dans le cadre hospitalier, le coût des solutions n'est pas l'objectif principal à minimiser, car on s'intéresse davantage à la qualité du travail fourni. Par exemple, pour éviter qu'elles commettent des erreurs professionnelles engageant la vie des patients, il est plus difficile d'attribuer des heures supplémentaires aux infirmières. Ainsi, on s'attache à trouver des solutions de bonne "qualité" plutôt que les solutions les moins coûteuses. À l'inverse, notre domaine permet généralement beaucoup plus de flexibilité concernant l'horaire.
Nous allons nous pencher sur ce problème en tentant d'atteindre trois objectifs. Le premier consiste à développer une méthode de ré-optimisation très rapide qui corrige la perturbation en ne modifiant que la planification du jour lors duquel la perturbation a eu lieu. On souhaiterait corriger la perturbation en moins d'une seconde et proposer des choix de réadaptation des horaires. Cette méthode utilise principalement l'information duale trouvée lors de la première optimisation. Nous proposons ainsi une méthode de régression pour actualiser l'information duale après la correction de chaque perturbation survenue sans passer par une optimisation exacte. Le deuxième objectif consiste à tenir compte, lors de la ré-optimisation, du jour où la perturbation a eu lieu ainsi que les jours suivants jusqu'à la fin de l'horizon. Pour ne pas trop modifier l'horaire planifié, nous utilisons une optimisation multi-objectif : chaque horaire est évalué par son coût ainsi sa distance à l'horaire planifié. La méthode se base sur un réseau qui schématise une région de l'enveloppe convexe des solutions réalisables. La construction du réseau se fait d'une manière dynamique au cours de la résolution en se basant sur quelques résultats théoriques. Le troisième objectif consiste à généraliser notre méthode de ré-optimisation introduite dans le deuxième objectif afin de l'utiliser pour l'optimisation globale d'un horaire de personnel. Nous voyons le problème d'optimisation comme

une séquence de plusieurs ré-optimisations traitant une séquence de plusieurs perturbations stimulées artificiellement à la base de certaines mesures "probabilistes".

1.4 Plan de la thèse

Cette thèse est organisée comme suit : le chapitre 2 présente une revue de littérature des travaux déjà effectués concernant les problèmes connexes au nôtre. Dans le chapitre 3, nous discutons, en bref, l'organisation des trois chapitres principaux de la thèse. Le chapitre 4 est consacré au premier objectif ; il s'agit d'un article soumis à la revue *Computers and Operations Research*. Le chapitre 5 représente un article qui porte sur notre deuxième objectif, soumis à la revue *Omega*. Le chapitre 6 est consacré au troisième objectif. Le chapitre 7 aborde une discussion générale sur la thèse. Enfin, les conclusions sont synthétisées dans le chapitre 8.

CHAPITRE 2 REVUE DE LITTÉRATURE

Dans cette thèse, on rappelle que notre objectif principal est de proposer des méthodes de ré-optimisation rapides pour corriger les petites perturbations qui peuvent survenir à tout moment dans un horaire de personnel planifié. Le succès de ces méthodes nous a inspiré à les généraliser et proposer une nouvelle heuristique rapide, qui se base sur la stimulation/correction de certaines perturbations bien ciblées, pour faire l'optimisation globale d'un problème d'horaires de personnel. Notre étude comprend donc quatre axes de recherche : la génération des quarts de travail, la résolution des problèmes d'horaires de personnel, les méthodes de ré-optimisation pour traiter l'incertitude dans un problème d'horaires de personnel et l'optimisation en temps réel. Dans la première partie de ce chapitre, nous abordons l'historique du problème d'horaires de personnel, pour ensuite articuler notre revue de littérature autour des quatre axes cités précédemment.

2.1 Problème d'horaires de personnel

Au début des années 1950, Edie [2] a proposé une méthode heuristique dans le but d'optimiser le délai d'attente des véhicules qui se présentent aux postes de péage de New York. Quelques mois plus tard, Dantzig [3] a proposé de résoudre le même problème en utilisant un programme linéaire en nombres entiers, à savoir un problème de recouvrement. Ainsi, il a présenté une modélisation capable de fournir un horaire de personnel pour répondre aux besoins en question à moindre coût. L'étude de Dantzig est devenue une référence puisque, depuis, les chercheurs s'intéressent à l'amélioration de ce modèle pour, entre autres, l'adapter à d'autres problèmes d'horaires de personnel. Dès lors, les problèmes d'horaires de personnel ont été très étudiés sous de multiples angles.

Baker et Magazine [4] ont apporté à la communauté scientifique l'une des premières méthodes de classification d'un problème d'horaires de personnel en distinguant les trois principaux groupes suivants : *Day-Off Scheduling*, *Shift Scheduling Problem* et *Tour Scheduling Problem*. Le Day-Off Scheduling consiste à planifier les jours de repos des employés sur un horizon (généralement une semaine) alors que le Shift Scheduling Problem consiste à construire des quarts de travail pour les employés. Le Tour Scheduling Problem aborde ces deux problèmes simultanément.

Plus tard, dans un état de l'art très complet [5], Ernst *et al*. ont proposé une méthodologie pour un problème générique de gestion de personnel où le processus de construction d'horaires est composé de six modules plus spécifiques :

i) l'estimation de la demande ;

ii) la planification des jours de congé ;
iii) la construction des quarts de travail ;
iv) la construction des lignes de travail ;
v) l'affectation des tâches aux quarts ;
vi) l'affectation des quarts aux employés.

L'optimisation d'un horaire de personnel ne sert à rien si l'on ne connaît pas la demande à satisfaire. Ainsi, dans un premier temps, l'employeur doit prévoir le nombre d'employés requis durant chaque période de l'horizon de planification (étape i). C'est un exercice ardu lorsque la demande comporte beaucoup de facteurs d'incertitude. Ensuite, l'étape ii et l'étape iii correspondent respectivement aux Day-Off Scheduling et Shift Scheduling Problem. À noter que durant l'étape iii, l'employeur construit chaque quart possible. Pour cela, il doit déterminer pour chacun d'entre eux les éléments suivants : son instant de début, son instant de fin et les intervalles de pauses pendant le quart. Lors de la construction des quarts, il ne prend pas en compte les particularités de chaque employé car ce sont des quarts anonymes qui respectent seulement les conventions collectives. L'employeur peut ensuite passer à l'étape iv ou bien directement à l'étape v si l'étape iv ne lui est pas nécessaire. En fait, l'étape iv consiste à regrouper les quarts générés jusque-là, de façon à ce que chaque groupe soit une séquence composée d'horaires de travail et de périodes de repos. Donc, chaque séquence correspond à un horaire individuel qui n'est attribué à aucun employé en particulier. L'ensemble de ces séquences permet à l'employeur de couvrir la demande de l'horizon étudié. Ces séquences sont appelées lignes de travail. Les seules contraintes qui seront prises en compte ici sont celles du repos minimum entre deux quarts consécutifs et le nombre minimum de jours de congé dans chaque ligne de travail. L'étape v consiste à affecter des tâches aux quarts. À cette étape, on distinguera deux cas, soit les cas mono-activité et multi-activité. Pour le premier cas, il suffit d'assigner un quart donné à une tâche donnée. Pour le second, il faut déterminer les tâches qui seront effectuées durant chaque quart donné ainsi que les instants de début et de fin pour chacune de ces tâches. Enfin, l'étape vi vise à affecter chaque quart à un employé (ou chaque ligne de travail à un employé, si on passe par l'étape iv). À cette étape, d'autres contraintes seront prises en compte, par exemple les qualifications et les disponibilités de chaque employé.

Ces six modules sont intégrés selon les caractéristiques du problème en question ; certains parmi eux peuvent être combinés. Dans une revue de littérature récente, Bergh *et al.* [6] ont recensé ainsi plus de 300 travaux effectués sur ces modules en fonction des caractéristiques des problèmes résolus.

2.2 Méthodes de résolution pour les différents types de problème d'horaires de personnel

Les problèmes d'horaires de personnel peuvent être de différentes natures selon le domaine d'application, la taille du problème, la nature de la main d'œuvre, le degré d'incertitude considéré, etc. Différentes méthodes exactes et heuristiques ont alors été proposées dans la littérature. Ces méthodes sont basées sur l'une ou sur une combinaison des techniques suivantes : la programmation linéaire, la programmation par contraintes, la programmation multi-objectif, les heuristiques, etc.

2.2.1 Programmation linéaire

Un programme linéaire (PL) consiste à optimiser une fonction objectif linéaire sur un polyèdre convexe. Les méthodes les plus utilisées dans la programmation linéaire sont les méthodes de points intérieurs et les méthodes basées sur le simplexe. Nous invitons le lecteur à se référer au livre de Vanderbei *et al.* [7] pour plus de détails.
En pratique, l'utilisation de la programmation linéaire pour un problème d'horaires de personnel amène généralement, à s'intéresser uniquement aux solutions entières dans le polyèdre convexe en question. Dans ce cas, on parle d'un programme linéaire en nombres entiers (PLNE). En 1960, Land et Doig [8] ont proposé la méthode, la plus commune pour la résolution d'un PLNE, appelée *branch-and-bound*. Cette méthode se base sur l'exploration récursive d'un arbre de branchement. Grâce au calcul des bornes inférieure/supérieure, on peut accélérer le processus en supprimant des branches de l'arbre. Pour un problème de minimisation, la borne supérieure est donnée par la valeur de la fonction objectif de la meilleure solution entière obtenue. La borne inférieure est déterminée suite à la résolution de plusieurs programmes linéaires rencontrés durant l'exploration de l'arbre. Pour améliorer la qualité de la borne inférieure, on peut ajouter des coupes dans les PLs et dans ce cas, on parle de *branch-and-cut*.
La résolution de PL en un temps raisonnable est réalisable uniquement si la taille du problème étudié est modéré, ce qui n'est généralement pas le cas pour les problèmes d'horaires de personnel. Pour cela, il faut adopter d'autres approches pour la résolution comme la méthode de génération de colonnes. Cette dernière consiste à considérer dans le PL un sous-ensemble de variables et à ajouter progressivement de nouvelles variables retournées par des sous-problèmes. Pour plus de détails, le lecteur est invité à consulter [9–11]. La méthode de *branch-and-price* consiste à utiliser la méthode de génération de colonnes pour résoudre les programmes linéaires rencontrés durant l'exploration de l'arbre de branchement pour la résolution d'un PLNE. Plusieurs solveurs commerciaux, ayant prouvé leur efficacité dans la résolution d'un PLNE, comme CPLEX et XPRESS, sont disponibles sur le marché.

2.2.2 Programmation par contraintes

La programmation par contraintes (PPC) est un paradigme de programmation affleurée dans les années 1970. La modélisation par PPC se base sur un ensemble de variables, chacune de ces variables x_i est associé à un domaine D_{x_i}. Les variables sont liées par différents types de contraintes : arithmétiques, logiques et symboliques. La résolution se base sur la propagation de contraintes consistant à filtrer, de chaque domaine D_{x_i}, les valeurs de x_i qui ne peuvent prendre part à une solution (voir [12]). Il existe plusieurs solveurs commerciaux, comme CP Optimizer, permettant de résoudre un modèle de PPC.
La PPC est une technique très efficace pour chercher des solutions réalisables pour un problème très contraint. En particulier, pour certains problèmes d'horaires de personnel, l'obtention d'une solution réalisable en présence de plusieurs contraintes conflictuelles représente un véritable défi. Dans ce cas, les approches basées sur la PPC semblent être les plus appropriées. Néanmoins, si l'espace combinatoire du problème considéré est très riche, elles seront moins adaptées pour trouver la solution optimale de cet espace. Dans la littérature, les principaux travaux basés sur la PPC concernent le milieu hospitalier, dû à la non flexibilité de ce domaine (Weil *et al.* [13] ; Abdennadher et Schlenker [14] ; Bourdais *et al.* [15]).

2.2.3 Programmation multi-objectif

La programmation multi-objectif (PMO) vise à optimiser plusieurs fonctions objectif à la fois. Cette approche nous mène à une modélisation plus réaliste. En effet, dans la vraie vie une solution peut rarement être évaluée selon un seul critère. Pour les problèmes d'horaires de personnel, plusieurs critères interviennent en pratique : coûts de gestion, préférences des employés/employeurs, etc. Les critères utilisés dans la PMO sont par essence conflictuels (par exemple, minimiser les coûts de la main-d'œuvre et maximiser les préférences des employés). Lorsque l'on utilise la PMO, deux solutions ne sont pas forcément comparables, alors la notion d'optimalité se perd et est substituée par le front de Pareto composé d'un ensemble de solutions. Chacune de ces solutions réalise un bon compromis entre les critères considérés (voir [16]).
La plupart des approches multi-objectif proposées dans la littérature pour les problèmes d'horaires de personnel concerne le milieu hospitalier, que ce soit pour trouver un horaire pour les infirmiers ou pour les médecins (Arthur et Ravindran [17] ; Oskarahan *et al.* [18] ; Beaulieu *et al.* [19] ; Azaiez *et al.* [20]).

2.2.4 Heuristiques

La complexité qui caractérise la plupart des problèmes d'horaires de personnel a rapidement influencé beaucoup de chercheurs à développer des heuristiques/métaheuristiques sacrifiant

la garantie d'optimalité mais permettant d'aboutir, rapidement, à une solution de très bonne qualité. Les approches heuristiques sont très populaires dans les problèmes d'horaires de personnel. Selon Ernst *et al.* [21], cette popularité est due au fait que les heuristiques sont relativement robustes, simples à implémenter, susceptibles d'être adaptées à un problème spécifique et aptes à prendre en compte des objectifs plus complexes. Parmi les métaheuristiques dont l'efficacité a été établie, on retrouve des méthodes de recherche locale et des méthodes évolutives.

Pour les méthodes de recherche locale, la démarche consiste à améliorer itérativement une seule solution à la fois. Ces méthodes sont fondées sur deux éléments indispensables : i) une structure de voisinage ; ii) une procédure qui induit une topologie sur ce voisinage. Le recuit simulé [22], la recherche tabou [23, 24] et la recherche à voisinages variables [25] sont les méthodes de recherche locale les plus populaires.

Les méthodes évolutives sont basées sur l'évaluation d'une population de solutions selon des règles bien précises. Chaque méthode évolutive est basée sur trois éléments fondamentaux : i) objet transformable facilement en solution appelé individu ; ii) une population formée par plusieurs individus ; iii) un mécanisme d'évolution qui permet de décider de la survie des anciens individus et comment produire de nouveaux individus. Les algorithmes génétiques [26] sont les plus populaires dans la famille des méthodes évolutives.

Pour optimiser un horaire, les heuristiques sont très souvent combinées avec une ou plusieurs des approches citées ci-dessus. On retrouve les travaux de Aickelin et Dowsland [27, 28] (approche basée sur la résolution d'un PLNE à l'aide de plusieurs algorithmes évolutionnaires) ; ou ceux de Berrada *et al.* [29] (approche combinant la méthode de recherche tabou et une approche multi-objectif).

2.3 Génération explicite et implicite des quarts de travail

Dans la littérature, nous distinguons deux types de classe d'approches proposées : les approches explicites qui sélectionnent les quarts retenus dans la solution parmi un ensemble de quarts candidats, et les approches implicites pour lesquelles la génération des quarts se fait au fur et à mesure dans le processus de résolution.

2.3.1 Approches explicites

Le modèle de Dantzig [3] est typiquement un modèle explicite. Ce modèle se base sur l'idée qu'on peut, théoriquement, énumérer tous les quarts réalisables, et associer chaque quart à une variable. Pour trouver la solution optimale, il faut résoudre un programme en nombres

entiers. La modélisation, la plus commune, pour ce problème se formule comme suit :

$$min \quad \sum_{q \in \mathcal{Q}} c_q X_q \tag{2.1}$$

$$\text{sujet à :} \quad \sum_{q \in \mathcal{Q}} a_q^p X_q \geq d_p \qquad \forall p \in \mathcal{P} \tag{2.2}$$

$$X_q \geq 0, \text{ entier} \qquad \forall q \in \mathcal{Q} \tag{2.3}$$

où :

- $\mathcal{Q}$: l'ensemble des quarts réalisables ;
- $\mathcal{P}$: l'ensemble des périodes considérées dans la planification ;
- a_q^p : vaut 1 si le quart $q \in \mathcal{Q}$ couvre la période $p \in \mathcal{P}$, 0 sinon ;
- c_q : le coût du quart $q \in \mathcal{Q}$;
- d_p : le nombre des quarts requis durant la période $p \in \mathcal{P}$;
- X_q : variable qui indique le nombre d'utilisations du quart $q \in \mathcal{Q}$.

La fonction objectif (2.1) représente les coûts totaux de travail de travail total. Les contraintes (2.2) permettent de satisfaire la demande pour chaque période considérée.

Tous les modèles basés sur celui de Dantzig sont inclus dans la famille des approches explicites. Certains modèles explicites se basent sur des variables d'affectation pour affecter le bon quart candidat au bon employé (Miller *et al.* [30] ; Warner [31] ; Beaulieu *et al.* [19] ; Filici et Gentile [32]).

Henderson et Berry [33] ont proposé d'améliorer la solution de Dantzig en utilisant plusieurs heuristiques qui génèrent des sous-ensembles de quarts réalisables. Il ont montré qu'avec des sous-ensembles de petite taille, ils pouvaient atteindre une solution quasi-optimale.

Segal [34] a travaillé sur un problème d'horaires de personnel à partir d'une liste de 30 types de quarts prédéterminés par un instant de début, une durée et un placement idéal d'une pause. Le problème est résolu en deux étapes : la première étape consiste à trouver les quarts optimaux en utilisant un modèle de flot et dans la seconde, une heuristique est utilisée pour affecter les pauses aux quarts.

Morris et Showalter [35] ont proposé un modèle général et simple reposant sur des quarts identiques afin de planifier l'horaire pour un horizon de 24 heures en minimisant le nombre de quarts nécessaires pour couvrir la demande. Il a alors utilisé les plans coupants sur la valeur de la borne inférieure du nombre de quarts nécessaires.

2.3.2 Approches implicites

Afin de réduire le nombre de variables de décision dans le problème considéré, plusieurs approches implicites ont été proposées dans la littérature. Dans ces approches, les variables de

décision concernent l'instant de début, la longueur et le placement d'une pause dans un quart. Ces approches ne nécessitent pas de construire des ensembles de quarts candidats générés explicitement. À notre connaissance, dans le domaine de la vente au détail, cette famille d'approches traite les problèmes anonymes pour lesquels les caractéristiques des employés sont relaxées ou identiques, on parle alors d'un personnel homogène (voir l'exemple 1).

Exemple 1. *Soit $\mathcal{E}$ l'ensemble des employés disponibles. Nous voulons trouver un horaire optimal pour satisfaire la demande d'une journée discrétisée en des périodes. Soit $\mathcal{P} = \{1, \ldots, |\mathcal{P}|\}$ l'ensemble de ces périodes, et soit d_p la demande en nombre d'employés pour la période $p \in \mathcal{P}$. On suppose que le personnel est homogène : le coût de rémunération pour n'importe quel employé pour une période de travail est donné par c ; chaque employé peut travailler entre $\underline{l}$ et $\bar{l}$ périodes. On propose une approche implicite.*

$$min \quad \sum_{e \in \mathcal{E}} c(f_e - b_e + 1) \tag{2.4}$$

$$sujet\ à : \quad \sum_{e \in \mathcal{E}} X_p^e \geq d_p \qquad \forall p \in \mathcal{P} \tag{2.5}$$

$$b_e \leq f_e \qquad \forall e \in \mathcal{E} \tag{2.6}$$

$$\underline{l} \leq f_e - b_e + 1 \leq \bar{l} \qquad \forall e \in \mathcal{E} \tag{2.7}$$

$$si\ b_e \leq p \leq f_e\ alors\ X_p^e = 1 \qquad \forall e \in \mathcal{E},\ \forall p \in \mathcal{P} \tag{2.8}$$

$$si\ b_e = 0\ alors\ f_e = 0 \qquad \forall p \in \mathcal{P} \tag{2.9}$$

$$X_p^e \in \{0, 1\} \qquad \forall e \in \mathcal{E},\ \forall p \in \mathcal{P} \tag{2.10}$$

$$b_e, f_e \in \mathcal{P} \cup \{0\} \qquad \forall e \in \mathcal{E},\ \forall p \in \mathcal{P} \tag{2.11}$$

où :

- *X_p^e : variable binaire vaut 1 si la période $p \in \mathcal{P}$ est couverte par l'employé $e \in \mathcal{E}$, 0 sinon ;*
- *b_e (resp. f_e) : variable indique la période pour laquelle le quart de l'employé commence (resp. finit). Si cette variable vaut la période 0, alors un jour de repos est affecté à l'employé $e \in \mathcal{E}$.*

La fonction objectif (2.4) représente le coût total de rémunération. Les contraintes (2.5) permettent de satisfaire la demande. Les contraintes (2.6)-(2.7) concernent la structure des quarts recherchés. Les contraintes logiques (2.8) nous permettent de sélectionner les périodes couvertes par le quart de l'employé e. Finalement, les contraintes (2.9) assurent que l'employé e est en repos si la variable b_e vaut la période 0. À noter que les contraintes logiques (2.8)-(2.9) peuvent être linéarisées afin de passer à un programme linéaire en nombres entiers.

Moondra [36] a proposé un programme linéaire pour résoudre un problème de planification.

Dans ce programme, il y a une variable pour chaque période de début et une variable pour chaque période de fin possible. Ces variables indiquent le nombre nécessaire d'employés qui commencent et finissent à ces périodes. La réalisabilité des quarts sélectionnés est gérée par les contraintes considérées dans le programme.
Bechtold et Jacobs [37] ont traité un problème de placement de pauses dans différents types de quarts considérés. Dans leur étude, les quarts sont générés explicitement et le placement de pause est modélisé implicitement. Ils ont supposé que les fenêtres du temps, pour le placement d'une pause dans chaque type, ne s'achèvent pas. Cette supposition était omise ultérieurement par Addou et Soumis [38], à l'aide des nouvelles contraintes considérées. Dans la littérature, on trouve plusieurs approches implicites proposées pour traiter le problème de placement de pauses, parmi elles, il y a celles de Aykin [39] et Rekik *et al.* [40].
Dans un contexte proche du nôtre, Musliu *et al.* [41] a proposé en 2004 une méthode implicite, basée sur une exploration du voisinage par la méthode de recherche tabou, dans le but de choisir un ensemble de quarts parmi 4 types de quarts (chaque type est caractérisé par une fenêtre d'heure de début et une fenêtre de durée) et déterminer le nombre d'employés nécessaires pour chaque quart de travail. Dans cette étude, le voisinage est déterminé par les mouvements suivants : ajouter/supprimer un quart, changer le début/la durée d'un quart et changer le nombre d'employés nécessaires pour un type de quart durant une journée donnée. Des mouvements composés ont également été utilisés. Afin d'éviter les minimums locaux, deux variétés de liste tabou sont utilisées : une dans laquelle on enregistre l'inverse des mouvements effectués, l'autre dans laquelle on enregistre les solutions trouvées. À noter que l'affectation finale des employés aux quarts, selon leurs disponibilités et qualifications, n'est pas considérée dans leur contexte (contrairement au nôtre). Le même problème a été analysé et amélioré dans des travaux ultérieurs [42–44]. En 2016, Bonutti *et al.* [45] ont travaillé sur le problème de Musliu *et al.* [41] mais en considérant plusieurs tâches (à savoir entre 2 et 3 tâches) et des nouveaux types de quarts qui contiennent des pauses. Ils ont proposé une méthode de recherche locale basée sur un recuit simulé. Le voisinage utilisé a été défini à l'aide des mouvements semblables à ceux définis plus haut. Mais ici, ils considèrent deux types de voisinages dont le premier est formé à l'aide de tous les mouvements de base excepté un mouvement avec lequel le deuxième voisinage est défini. Il s'est avéré que la recherche séquentielle sur ces deux voisinages a mieux fonctionné que la recherche sur le voisinage classique utilisant tous les mouvements considérés à la fois.

2.4 L'incertitude et l'optimisation en temps réel

Dans la pratique, les problèmes d'horaires de personnel sont sujets à beaucoup d'incertitude. À titre d'exemple, supposons qu'on veut construire un horaire pour un centre d'achats. Dans

ces conditions, la demande est très variable. De plus, la variation est très aléatoire puisqu'elle peut dépendre de nombreux facteurs difficilement contrôlables (par exemple, la météo). Ainsi, plusieurs réalisations sont possibles. Pour que la solution soit optimale, l'employeur doit attendre la dernière minute pour observer la vraie réalisation et attribuer les quarts aux employés afin de répondre à la demande exacte de cette réalisation. Cette stratégie, appelée *wait and see*, est non applicable. En effet, l'employé doit connaître son horaire bien en avance. Une approche stochastique est utilisée dans un contexte incertain où les prises de décisions se font en deux étapes. On commence par une décision à long terme appelée *primaire* ou *a priori*, et on prévoit pour chaque réalisation, la bonne décision qu'il faut prendre afin de répondre à la demande de cette réalisation. Ces décisions sont appelées les décisions de *recours*.

Parmi les premières formulations des problèmes d'horaires de personnel stochastiques, on retrouve celle de Aykin [39]. La décision primaire consiste à construire des quarts d'une durée fixe égale à 8 heures, et pour chaque quart, on donne un intervalle de temps pendant lequel une pause devra être prise. En fonction de la demande réelle, on peut prendre une décision de recours : on fixe les pauses ou on allonge les quarts afin de répondre à cette demande. Bard *et al.* [46] ont utilisé une approche stochastique pour le problème de construction des quarts pour un opérateur postal et ils ont réussi à démontrer qu'elle permettait d'économiser 5% du coût total.

La difficulté principale de ce type de problèmes réside dans la résolution. En effet, la multiplicité des réalisations possibles rend la résolution à l'aide d'un solveur classique quasi impossible. Pour y remédier, il faut avoir recours à des techniques de résolution spécialisées. Dans le cas des problèmes stochastiques avec des recours non entiers, la méthode $L-shaped$ est très utilisée. C'est une méthode introduite par Van Slyke et Wets [47] basée sur la décomposition de Benders [48]. Il s'agit d'utiliser un problème maître où les décisions doivent être prises avant que l'incertitude soit "révélée". On utilise également des sous-problèmes de façon à ce que chacun d'entre eux puisse être associé à une réalisation possible. Grâce aux coupes d'optimalité et de faisabilité, ces sous-problèmes remontent les informations sur la qualité des décisions au problème maître. Une généralisation de cette méthode existe pour la résolution des problèmes stochastiques avec des recours entiers (voir le livre de Birge et Louveaux [49]).

Souvent l'incertitude dans les problèmes d'optimisation est révélée petit à petit (exemple : les appels entrants dans un centre d'appels). Un tel aspect "dynamique" rend l'ensemble des réalisations possibles très grand, ce qui complique la résolution. Une optimisation en temps réel peut être alors une bonne option pour remédier à ce problème. L'objectif principal de cette optimisation est de prendre des décisions instantanément lorsqu'un événement survient. Ces décisions ont alors pour but de satisfaire la demande (peu importe le type) liée à cet

événement mais aussi de conserver une très bonne solution globale. À souligner que le critère le plus important est le temps de résolution. Une revue de littérature de Jaillet et Wagner [50] expose les domaines d'application, ainsi que les méthodes en temps réel. Ces méthodes ont été développées pour fournir des solutions "robustes" quels que soient les futurs événements susceptibles de survenir. Généralement, les problèmes étudiés sont *NP-difficiles*. Buchbinder [51] a étendu l'algorithme primal-dual (cet algorithme peut changer, en fonction des décisions passées, la façon de sélectionner la variable entrante en base contrairement à l'algorithme du simplexe) pour le considérer comme une approche en temps réel sur des dizaines de problèmes classiques.

Récemment, les chercheurs ont commencé à s'intéresser aux algorithmes d'optimisation en temps réel en prenant en compte les informations probabilistes du futur. Une telle stratégie augmente la chance de garder une solution globale quasi-optimale. Manshadi *et al.* [52] ont recueilli des statistiques en arrière-plan sur les décisions à l'aide d'un échantillonnage réalisé avec la méthode de Monte Carlo. Ces statistiques ont été utilisées par la suite pour guider les décisions de l'algorithme en ligne. Van Hentenryck et Bent [53] ont suggéré une autre stratégie. En effet, ils ont proposé une métaheuristique de façon à générer un ensemble fini de scénarios possibles juste avant de prendre une décision. Ensuite, ils ont estimé l'impact de chaque décision sur chaque scénario. Enfin, la meilleure décision est choisie en fonction de ces impacts.

2.5 Ré-optimisation d'un horaire de personnel après une perturbation

Au début des années 2000, les chercheurs ont commencé à s'intéresser au problème de la ré-optimisation d'un horaire prévu. La plupart des travaux ont abordé le problème de la ré-optimisation associé aux horaires des infirmières qui comprennent un nombre très limité de quarts de travail possibles (par exemple, 7am à 3pm, 3pm à 11pm, et 11pm à 7am). La demande est souvent exprimée par le nombre de quarts de travail requis pour une tâche. Dans ce contexte, une petite perturbation correspond à l'absence d'une infirmière (resp. des infirmières) pour un quart (resp. plusieurs quarts) de travail. Moz et Pato [54, 55] et Bard et Purnomo [56] ont proposé des algorithmes exacts de branchement pour résoudre les variantes de ce problème. Étant démesurément lents, ces algorithmes ne sont pas applicables en temps réel. Des algorithmes heuristiques qui traitent le même problème ont également été développés : se référer aux travaux de Pato et Moz [54, 57, 58], Maenhout et Vanhoucke [59], et Kitada et Morizawa [60, 61]. Même si certains de ces algorithmes s'exécutent en un temps raisonnable, ils conviennent davantage au domaine de la santé qu'au domaine de la vente au détail qui nous intéresse. Ce domaine offre beaucoup de flexibilité sur la construction des quarts et sur les modifications admissibles pour ajuster un horaire.

En 2016, Gross *et al.* [62] ont travaillé sur la ré-optimisation des horaires des médecins après l'absence de l'un d'entre eux en cherchant un compromis entre la qualité du nouvel horaire et la distance de ce dernier à l'horaire initial. Pour ce faire, un programme linéaire mixte en nombres entiers est résolu. Le temps de résolution pouvait aller jusqu'à 21 secondes. À noter que pour aboutir au meilleur compromis, il fallait faire tourner le programme plusieurs fois avec différents choix de paramètres. C'est d'ailleurs cette donnée qui a influencé principalement le temps total nécessaire pour choisir la bonne ré-optimisation.
En 2015, Froger [63] a travaillé dans le domaine de la vente au détail sur les perturbations à grandes échelles, à savoir les grandes perturbations de la demande. Elle a proposé une méthode de résolution exacte, à partir des quarts prévus pour chaque employé, puis a énuméré des quarts transformés selon certaines règles de modification, et une résolution d'un programme en nombres entiers. Notre projet comporte quelques différences : il est plus contraignant en termes de temps de calcul et on s'intéresse aux perturbations à petites échelles. Cependant, cette forme de solution nous sera utile pour tester l'efficacité de notre méthode de ré-optimisation.
Meignan [64] a développé une approche heuristique de ré-optimisation dans un autre contexte, à savoir l'optimisation interactive [65]. Cette approche est une méta-heuristique de recherche locale appliquée à la ré-optimisation d'un horaire initial. L'employeur essaye d'améliorer l'horaire initial obtenu à l'aide d'un logiciel d'optimisation en le rendant plus réaliste et plus applicable. En effet, lors de l'optimisation initiale, certains critères et événements pouvant se produire sont négligés. Par exemple, des estimations du nombre de demandes sont faites et des aspects comme la ponctualité et l'efficacité des employés ne sont pas pris en compte. Bürgy *et al.* [66] ont essayé d'intégrer certains critères cités ci-dessus (qui requièrent une intervention manuelle de l'employeur et une ré-optimisation) dans un modèle en nombres entiers afin de trouver des horaires initiaux plus réalistes et de permettre à l'employeur de réagir plus efficacement à d'éventuelles perturbations. Pour cela, ils ont utilise à la fois des données de la demande prévue, la probabilité d'avoir une perturbation de demande pour certaines périodes ainsi que la probabilité d'absence et de retard des employés. En bref, cette étude a pour objectif de développer une méthode capable de réagir efficacement aux perturbations probables.

2.6 Sommaire

D'après cette revue de littérature, on peut faire les constats suivants sur le domaine de la vente au détail :

- le domaine est très peu abordé dans la littérature ;
- lorsque le personnel est hétérogène, dans ce domaine, seules les approches explicites

sont utilisées. En effet, toutes les approches implicites proposées visent à trouver un horaire anonyme dans lequel les caractéristiques des employés sont omises. Lorsque ces caractéristiques sont très différentes, l'affectation des quarts anonymes, présents dans l'horaire anonyme, aux employés n'est pas évidente ;

— aucune méthode de ré-optimisation en temps réel n'est proposée pour corriger les petites perturbations dues au manque de personnel qui peuvent survenir à tout moment.

Dans cette thèse, nous abordons l'optimisation globale d'un horaire de personnel en proposant une méthode intégrée qui génère et affecte les quarts simultanément durant la résolution dans le domaine de la vente au détail. On se place dans un contexte qui se caractérise avec une demande qui peut fluctuer à n'importe quel moment de l'horizon étudié, avec un personnel très hétérogène au niveau des disponibilités, des qualifications et des durées de travail autorisées par jour. Nous proposons aussi des méthodes de ré-optimisation des horaires de personnel, dans le même contexte, pour corriger les petites perturbations. On suppose qu'on ne dispose d'aucune information sur la perturbation en question avant son arrivée. Pour cette raison, les méthodes de ré-optimisation proposées fournissent des choix d'adaptation en moins d'une seconde en moyenne.

CHAPITRE 3 ORGANISATION DU TRAVAIL

D'après la revue de la littérature sur les problèmes d'horaires de personnel, exposée dans le chapitre 2, on peut déduire que l'aspect de ré-optimisation en temps réel est rarement étudié dans le domaine de la vente au détail. En particulier, cela est vrai pour la correction d'un horaire de personnel en temps réel suite à une petite perturbation qui engendre un manque de personnel pour couvrir la demande. Notre étude s'inscrit dans cette direction. En effet, nous abordons, dans un premier temps, la ré-optimisation sur une seule journée en tirant profit de l'information duale fournie par le modèle de planification. Dans un deuxième temps, on définit une heuristique qui ne requiert pas cette information duale (qui n'est pas toujours disponible) et qui ré-optimise sur plusieurs journées. Finalement, étant donné que cette dernière heuristique est très rapide, elle est intégrée dans une heuristique pour résoudre le problème de planification en créant diverses perturbations à la base de certaines mesures probabilistes.

Le chapitre 4 propose une méthode de ré-optimisation en temps réel des horaires de personnel suite à une petite perturbation due au retard ou à l'absence d'un employé. La méthode devra proposer des choix de réadaptation des horaires et évaluer les coûts des modifications occasionnées en tenant compte des coûts immédiats (coût de gestion) et des coûts futurs déterministes (impact des modifications sur les horaires futurs et la rémunération des employés). La méthode ne modifie que la planification du jour lors duquel la perturbation a eu lieu et utilise principalement l'information duale trouvée lors de la première optimisation. Cette information est actualisée après chaque correction d'une perturbation à l'aide de la méthode de régression M.A.R.S. Ce travail est exposé dans l'article *Real-time personnel rescheduling after a minor disruption in the retail industry* soumis à *Computers and Operations Research*.

Dans le chapitre 5, on expose une nouvelle approche de ré-optimisation très rapide qui tient en compte, lors de la ré-optimisation, du jour lors duquel la perturbation a eu lieu et aussi des jours restants dans l'horizon. Pour ne pas trop modifier l'horaire planifié, nous avons utilisé une optimisation multi-objectif : chaque horaire est évalué par son coût ainsi que la distance de l'horaire planifié donnée par le nombre de modifications de quarts. L'heuristique peut trouver un ensemble de solutions approximatives Pareto-optimales qui permettent d'obtenir un bon compromis entre le coût et le nombre de modifications de quarts. Elle peut être vue comme un algorithme d'étiquetage qui explore partiellement un réseau défini par les arêtes de l'enveloppe convexe des solutions d'un programme en nombres entiers. Des résultats théoriques sont fournis à l'appui de certaines règles d'accélération. Ce travail est présenté dans

l'article *Real-time bi-objective personnel re-scheduling in the retail industry* soumis à *Omega*. Le chapitre 6 présente une approche heuristique, inspirée de la méthode proposée dans le chapitre 5, qui vise à optimiser un horaire de personnel. L'approche proposée est une approche intégrée qui génère/affecte les quarts simultanément pour chaque employé durant la résolution. Elle se base sur la stimulation/correction itérative d'un ensemble de perturbations bien ciblées selon certaines mesures probabilistes afin de la guider pour effectuer une recherche efficace. Deux procédures de groupage sont utilisés afin de générer un ensemble de sous-problèmes qui seront traités parallèlement. Pour chaque sous-problème, nous stimulons une perturbation qui nous mène, généralement, à l'espace irréalisable. Nous définissons un voisinage sur ce sous-problème afin de corriger la perturbation et ainsi revenir à l'espace réalisable. Nous combinons de façon optimale les améliorations compatibles, obtenues parallèlement dans les sous-problèmes.

CHAPITRE 4 ARTICLE 1 : REAL-TIME PERSONNEL RE-SCHEDULING AFTER A MINOR DISRUPTION IN THE RETAIL INDUSTRY

R. Hassani, G. Desaulniers et I. Elhallaoui ont écrit cet article et l'ont soumis en 2017 dans la revue *Computers and Operations Research*.

4.1 Introduction

Personnel scheduling problems are frequent in numerous domains such as retail, transportation, healthcare, and manufacturing, to name just a few. There exist several problem variants which are defined according to the environment in which they arise. For instance, in hospitals and factories, employees often work 24 hours per day, seven days per week and are assigned to a limited number of shifts (e.g., 7am to 3pm, 3pm to 11pm, and 11pm to 7am). In this case, the demand in employees for each job (task or position) to accomplish is often expressed by shifts (or half-shifts when shifts overlap). On the other hand, in large retail stores, leisure resorts or call centers, demand occurs only during the business hours and may highly fluctuate throughout a day and from one day to another. It needs to be expressed by short time intervals (e.g., 15-minute periods) for each job and not by shifts. To meet these demand fluctuations, numerous shifts (often several hundreds per day), starting and ending at about any time, can be assigned to the employees. In addition, in some contexts, an employee is always assigned to the same job in a shift while, in others, he/she can be assigned to multiple jobs in a shift. Typically, a personnel scheduling problem consists of determining the working schedules (the exact shifts and jobs) of a set of employees such that the demand for each job and each period is met as much as possible at minimum cost while satisfying labor rules. Costs include employee salaries and often various penalties for enforcing the satisfaction of soft constraints or preferences. Solving these problems is often a challenge that is addressed using an optimization algorithm.

Unfortunately, a personnel schedule is rarely operated as planned. Indeed, it often occurs that an employee is late for work, must leave earlier than scheduled, or simply does not show up. Such a *minor disruption* usually makes the planned schedule infeasible and must be addressed as soon as possible. To deal with a minor disruption, re-scheduling must be performed, i.e., the planned schedule must be modified to retrieve a feasible schedule that takes into account the disruption but does not differ much from the planned schedule. Given that minor disruptions are often known just before the operations and the modified employee schedules must be announced to the impacted employees as soon as possible, re-optimization must be done in real time.

In this paper, we address the problem of re-optimizing a personnel schedule to address a minor disruption in real time. We assume that this problem occurs in the retail industry or a similar one, where there are several jobs to cover by skilled employees, the number of employees required for each job may fluctuate by (short) period throughout the planning horizon, and the employees can be assigned to a wide variety of possible shifts, starting and ending at various times and with various durations. In this case, the disruption can cause a lack of personnel for a job just for a few periods (say, one hour) and can thus be addressed, for example, by simply extending the shift of an employee to cover these periods. Other options can be considered to resolve a minor disruption but, in all cases, the number of changes brought to the planned schedules must be rather limited and impact only the day of the disruption.

4.1.1 Literature review

The study of personnel scheduling problems in the operations research literature started at the beginning of the 1950s with the work of Edie [2] who introduced a heuristic method for solving a shift scheduling problem for toll booth operators with the objective of minimizing waiting at the toll booths. A few months later, Dantzig [3] proposed to model this problem as a set covering problem and solve it using integer linear programming techniques. Since then, numerous papers have been published on many personnel scheduling problems. Exhaustive surveys on these works were written by Ernst *et al.* [5], Burke *et al.* [67], and Van den Bergh *et al.* [6].

Literature on personnel re-scheduling is rather scarce. Most works have tackled the nurse re-rostering problem that considers a very limited number of possible shifts (three shifts per day in most cases) and, often, demand by shifts. In this case, a minor disruption corresponds to the absence of a nurse for one or several consecutive shifts and can be resolved by re-assigning the nurses to the shifts for one or several days, or resorting to part-time or on-call nurses. Larger disruptions that span a whole shift and take into account employee absences and revised demands have also been considered. For solving these problem variants, exact branch-and-bound and branch-and-price algorithms were proposed by Moz and Pato [54,55] and Bard and Purnomo [56,68], whereas heuristics were developed by Moz and Pato [54,57], Pato and Moz [58], Maenhout and Vanhoucke [59], and Kitada and Morizawa [60]. Even if some of these algorithms yield fast computational times, they are suited for the nurse re-rostering context, which differs from the retail context addressed in this paper.

Personnel re-scheduling has also been studied in other sectors such as transportation (see Clausen *et al.* [69] and Cacchiani *et al.* [70] for surveys in air and rail transportation, respectively). Because the crews travel during their working days and may even overnight outside

of their homes in some cases, these problems differ substantially from the one considered in this study and, therefore, the proposed models and solution approaches are not suitable.
To the best of our knowledge, no works have been published on personnel re-scheduling problems arising in retail stores or similar settings, beside the recent master's thesis of Froger [63]. This thesis focuses on large disruptions, such as when a relatively large variation of the demand is expected to occur on one or several days due, for instance, to bad weather or an unanticipated sale. The author developed an integer linear program that considers variables associated with the planned employee shifts and additional shifts that are derived from the planned shifts according to parameterized modification rules. Different rules and parameter values are tested. In all cases, the resulting model can be solved by a commercial mixed integer programming solver in less than two minutes for instances involving up to 50 employees. This work has similarities with our study. The initial personnel scheduling problem is the same and the modifications that can be brought to the planned shifts are the same. On the other hand, the sizes of the disruptions differ and the time available to address a disruption is much less in our case (a few seconds versus a few minutes). Consequently, the solution approach proposed by Froger [63] cannot be used for tackling a minor disruption in real time.

4.1.2 Contributions

The previous literature review shows that personnel re-scheduling following a minor disruption in a retail store or a similar environment (where demand is highly fluctuating, numerous shifts are feasible, planned shifts can be modified in various ways, and fast computational times are required) has not been addressed yet. This paper aims at fulfilling this gap. First, we develop an efficient heuristic for re-optimizing a personnel schedule in real time once a minor disruption becomes known. This fast heuristic can compute for a single disruption one or several solutions modifying the planned schedule only on the day of disruption. It is based on the dual solution of the linear relaxation of the model used to compute this planned schedule. Second, we exploit a non-parametric regression method for estimating the evolution of some dual values when multiple disruptions occur, yielding a sequence of re-optimizations. Third, through extensive computational experiments performed on various instances derived from real-world datasets involving between 15 and 195 employees, we show that the proposed heuristic finds in less than two seconds an optimal solution in more than 95% of the test cases. Additional computational results obtained when resolving a sequence of minor disruptions indicate that using the proposed regression method for updating the dual values reduces the gap between the optimal value and the heuristic solution value by an average of 73%.

4.1.3 Paper structure

This paper is organized as follows. In Section 4.2, we first state the initial personnel scheduling problem and then derive from it the personnel re-scheduling problem considered. In Section 4.3, we describe the proposed heuristic and the non-parametric regression method used to update dual values. In Section 4.4, we report and analyze the results of our computational experiments. Finally, conclusions are drawn in Section 4.5.

4.2 Problem description

In this section, we start by stating the personnel scheduling problem solved to establish planned schedules for the employees. We also present an integer linear programing model for this problem because the proposed heuristic exploits the dual variables of its linear relaxation. Finally, we describe the personnel re-scheduling problem by defining the minor disruptions considered and the type of decisions that can be made to resolve them.

4.2.1 The personnel scheduling problem

Before the planning horizon (typically, one week or one month), the personnel manager must disclose the employee working schedules for that horizon, i.e., the days that each employee will work and, for each working day, the exact shift that he/she will work and on which job. Let us present a detailed definition of this personnel scheduling problem when the horizon $\mathcal{H} = \{1, 2, ..., 7\}$ is a set of seven consecutive days numbered from 1 to 7. The model described below can easily be adapted for other horizons. Note, however, that longer horizons would require additional time-related constraints such as a minimum and a maximum number of consecutive working days.

Horizon $\mathcal{H}$ is also divided into a set $\mathcal{P} = \{1, \ldots, p^{max}\}$ of consecutive periods of equal length (for example, 15 minutes) numbered from 1 to p^{max}. Any shift that can be assigned to an employee starts at the beginning of a period and ends at the end of another period. Let $\mathcal{E}$ be the set of available employees and $\mathcal{W}$ the set of jobs to accomplish. In the retail context, a job is a type of a position that serves the customers in real time such as a cashier in a supermarket or a clerk in its bakery department. An employee assigned to a job can be replaced at any time by another employee or can interrupt its assignment at any time to go for a break, move to another job, or end his/her day.

In this paper, we assume that an employee can be assigned to a single job during a shift and that no breaks need to be scheduled in the shifts. This first assumption is common in practice as it simplifies the operations, especially, personnel management, and avoids scheduling short breaks that are often mandatory when a job change is scheduled. This assumption is imposed

only for ease of exposition as our heuristic can be easily adapted if it does not hold (additional decision types similar to the ones proposed below would be needed). The second assumption is also frequent in practice because the breaks are often assigned during the operations depending on the observed demand. Typically, to take breaks into account when planning the personnel schedule, the forecasted demand is increased during certain time intervals. In a context where this assumption does not hold, it would still be possible to adapt the proposed heuristic by introducing new decision types that might, however, be more complex than the ones described below. Indeed, break placement is often subject to regulations that would restrict the possible decisions.

The main goal of the personnel scheduling problem is to assign the employees to shifts such that enough employees are on duty for each job in each period so as to offer a service of good quality to the customers. One of the main input data to this problem is, thus, the required number of employees d_w^p for each job $w \in \mathcal{W}$ and period $p \in \mathcal{P}$. To fulfill this demand, one must determine for each employee $e \in \mathcal{E}$ a working schedule that is composed of shifts taken from sets of potential feasible shifts $\mathcal{S}_e^h$, $h \in \mathcal{H}$, that can be assigned to e. A shift s is defined by a starting period b_s, an end period f_s, a length l_s in number of periods ($l_s = f_s - b_s + 1$), and a job w_s to which the employee is assigned in this shift. It belongs to $\mathcal{S}_e^h$ if : 1) b_s belongs to day h and is an admissible starting period for employee e ; 2) l_s falls into a feasible shift length interval (for instance, between 4 and 10 hours) ; and 3) employee e is skilled for job w_s and available on day h. Like the employee skills, the set of admissible starting periods and the feasible shift length interval may vary from one employee to another. The schedule of employee e is feasible if it contains : 1) at least n^O days off ; 2) at most one shift per working day ; 3) at least n^R periods of rest between shifts assigned on two consecutive days ; and 4) a total of at most n^P working periods. Note that, for horizons longer than one week, additional conditions on the distribution of the days off need to be considered and the maximum number of periods may be imposed for every week, not for the whole horizon.

Given that the demand for each job may fluctuate from one period to the next, it is nearly impossible to cover this demand exactly with a set of feasible employee schedules. In fact, it is not rare that there is too much or too many employees assigned to a job in a period. In the former case, we say that each exceeding employee yields an over-covering. In the latter, each missing employee corresponds to an under-covering. In practice, the under-coverings are usually covered by temporary or on-call employees who are usually scheduled just before the operations if really needed. To exploit this possibility at best, the under-coverings for a job should be bunched together as much as possible so as to be able to cover them with shifts for these extra employees. Consequently, the input to the personnel scheduling problem also includes a set of anonymous shifts, denoted S^A, that can be eventually assigned to these

employees, i.e., which are feasible with respect to their start times and lengths. Beside a starting period b_s, an end period f_s, and a length l_s, an anonymous shift $s \in S^A$ is also defined by a job w_s. Considering these shifts allows to forbid demand under-covering. They should, however, be highly penalized. Note that an anonymous shift can be chosen several times in a solution.

The personnel scheduling problem involves three types of costs : labor costs, anonymous shift penalties, and over-covering penalties. The labor costs correspond to the salaries of the employees that are paid for the time worked. Typically, the most senior employees, those that should be favored to work more, have the largest hourly rates. Consequently, if the real hourly rates were considered in the model, the senior employees would be assigned less working hours than the junior ones. To avoid this, one can consider an hourly rate that is the same for all employees. In our case, to favor an almost even distribution of the working hours among the employees, we use a step-wise function to compute an adjusted labor cost for each employee, that is, the hourly rate increases by step with the number of hours worked. This function is defined by a set of steps $\mathcal{K}^L = \{1, \ldots, m^L\}$, where m^L is the number of steps, and, for each step k, a number of periods n_k^L which define the length of this step and a cost per period c_k^L such that $c_k^L < c_{k+1}^L$ for $k \in \mathcal{K}^L \setminus \{m^L\}$. Notice that $\sum_{k \in \mathcal{K}^L} n_k^L$ corresponds to the maximum number of periods that an employee can work in a week.

Next, each anonymous shift $s \in \mathcal{S}^A$ yields a penalty c_s^Y that is proportional to the shift length l_s. These penalties are large enough compared to the other costs to ensure the minimization of the total number of periods assigned in anonymous shifts. Finally, over-covering penalties are also considered even though the labor costs ensure that there will be no extra over-covering. In fact, we use again a step-wise function to compute the over-covering penalty associated with each job in each period. With this function, the over-covering of a job is more evenly distributed among the various periods. For instance, if two periods of over-covering are unavoidable for a job, it is preferable to have them, if possible, on two different periods rather than at the same period. The over-covering penalty function does not depend on the job and is defined by a set of steps $\mathcal{K}^V = \{1, \ldots, m^V\}$, a number of over-coverings n_k^V in each step k, and a cost per over-covering c_k^V in each step k with $c_k^V < c_{k+1}^V$ for $k \in \mathcal{K}^V \setminus \{m^V\}$.

To model the personnel scheduling problem, we also rely on the following notation. For each period $p \in \mathcal{P}$, job $w \in \mathcal{W}$ and shift $s \in \bigcup_{h \in \mathcal{H}, e \in \mathcal{E}} \mathcal{S}_e^h \cup \mathcal{S}^A$, the binary parameter a_{sw}^p takes value 1 if shift s covers job w in period p and 0 otherwise. Let M be a sufficiently large constant. The variables are defined as follows.

X_{es}^h : Binary variable equal to 1 if shift $s \in \mathcal{S}_e^h$ is assigned to employee $e \in \mathcal{E}$ on day $h \in \mathcal{H}$ and 0 otherwise ;

O_e^h : Binary variable equal to 1 if employee $e \in \mathcal{E}$ has a day off on day $h \in \mathcal{H}$ and 0

otherwise ;

L_e^k : Integer variable specifying the number of periods worked by employee $e \in \mathcal{E}$ on step $k \in \mathcal{K}^L$;

Y_s : Integer variable indicating the number of times that anonymous shift $s \in \mathcal{S}^A$ is used in the solution ;

V_w^{kp} : Integer variable indicating the number of over-coverings for job $w \in \mathcal{W}$ in period $p \in \mathcal{P}$ and on step $k \in \mathcal{K}^V$.

Given this notation, the personnel scheduling problem can be modeled as the following integer program.

$$\text{Minimize} \quad \sum_{e\in\mathcal{E}} \sum_{k\in\mathcal{K}^L} c_k^L L_e^k + \sum_{s\in\mathcal{S}^A} c_s^Y l_s Y_s + \sum_{p\in\mathcal{P}} \sum_{w\in\mathcal{W}} \sum_{k\in\mathcal{K}^V} c_k^V V_w^{kp} \tag{4.1}$$

$$\text{subject to :} \quad \sum_{s\in\mathcal{S}_e^h} X_{es}^h + O_e^h = 1, \qquad \forall e \in \mathcal{E}, h \in \mathcal{H} \tag{4.2}$$

$$\sum_{h\in H} O_e^h \geq n^O, \qquad \forall e \in \mathcal{E} \tag{4.3}$$

$$\sum_{h\in\mathcal{H}} \sum_{s\in\mathcal{S}_e^h} l_s X_{es}^h - \sum_{k\in\mathcal{K}^L} L_e^k = 0, \qquad \forall e \in \mathcal{E} \tag{4.4}$$

$$M\, O_e^{h+1} + \sum_{s\in\mathcal{S}_e^{h+1}} b_s X_{es}^{h+1} - \sum_{s\in\mathcal{S}_e^h} (f_s + 1 + n^R) X_{es}^h \geq 0, \qquad \forall e \in \mathcal{E}, h \in \mathcal{H} \setminus \{7\} \tag{4.5}$$

$$\sum_{e\in\mathcal{E}} \sum_{h\in\mathcal{H}} \sum_{s\in\mathcal{S}_e^h} a_{sw}^p X_{es}^h + \sum_{s\in\mathcal{S}^A} a_{sw}^p Y_s - \sum_{k\in\mathcal{K}^V} V_w^{kp} = d_w^p, \qquad \forall p \in \mathcal{P}, w \in \mathcal{W} \tag{4.6}$$

$$X_{es}^h \in \{0,1\}, \qquad \forall e \in \mathcal{E}, h \in \mathcal{H}, s \in \mathcal{S}_e^h \tag{4.7}$$

$$O_e^h \in \{0,1\}, \qquad \forall e \in \mathcal{E}, h \in \mathcal{H} \tag{4.8}$$

$$L_e^k \in [0, n_k^L], \text{integer}, \qquad \forall e \in \mathcal{E}, k \in \mathcal{K}^L \tag{4.9}$$

$$Y_s \geq 0, \text{integer}, \qquad \forall s \in \mathcal{S}^A \tag{4.10}$$

$$V_w^{kp} \in [0, n_k^V], \text{integer}, \qquad \forall w \in \mathcal{W}, p \in \mathcal{P}, k \in \mathcal{K}^V. \tag{4.11}$$

The objective function (4.1) aims at minimizing the sum of the labor costs, the penalties incurred by the anonymous shifts, and the over-covering penalties. Constraints (4.2) ensure that each employee is assigned to a shift or a day off on each day of the horizon. Constraints (4.3) impose the assignment of at least n^O days off to each employee. The distribution of the periods worked by each employee on the various steps of $\mathcal{K}^L$ are computed through constraints (4.4). Note that the relations between the costs c_k^L ensure that the cheapest steps are always used

first. Constraints (4.5) specify that there must be a rest of at least n^R periods between shifts assigned to the same employee on two consecutive days. For each job and each period, constraints (4.6) guarantee that the demand is covered by a sufficient number of employees or anonymous shifts. They also allow to compute the number of over-coverings per step in $\mathcal{K}^V$. Finally, the domains of the decision variables are restricted by (4.7)–(4.11).
Note that, for a typical instance involving more than 25 employees and 5 jobs, model (4.1)–(4.11) involves more than 1000 constraints and 15,000 variables per day of the horizon (see Section 4.4.1). Consequently, one might not expect very fast computational times with this model when re-scheduling over a single day is required after a minor disruption.

4.2.2 The personnel re-scheduling problem

In general, the small disruptions which can make a planned schedule infeasible are numerous. These disruptions can be triggered, for example, by the lateness or the absence of an employee, a demand increase at certain periods of a day, or an unforeseen event that prevents an employee to complete a planned work shift. In this paper, we focus on the lateness of an employee because most of the other cases can be treated similarly. Indeed, the lateness of an employee creates a demand block (one employee is required for a specific job and a given number of consecutive periods) that must be covered during the operations. All the other examples listed above (employee absence, demand increase, premature shift end) can also be seen as yielding a demand block to cover. For instance, the absence of an employee yields a demand block of the length of the planned shift for this employee. Therefore, in all these cases, the personnel schedule must be adjusted to cover a demand block, using decision types that may be specific to the cause of the disruption. Note that, when a disruption results in a demand decrease, the planned schedule remains feasible but with additional over-coverage. To avoid this new over-coverage and reduce the operational cost, the decision to make is often easy to choose given that reducing the working time of an employee is almost always feasible.
In the following, we characterize a minor disruption due to a lateness by :

- the employee $\hat{e} \in \mathcal{E}$ who is late ;
- the day $\hat{h} \in \mathcal{H}$ when the disruption occurs ;
- the planned shift $\hat{s} \in \mathcal{S}_{\hat{e}}^{\hat{h}}$ of employee $\hat{e}$;
- the lateness duration $\hat{l}$ in periods $(\hat{l} < l_{\hat{s}})$;
- the period $\hat{p} \in \mathcal{P}$ at which the manager is notified that the employee will be late. We assume that $\hat{p}$ is a period prior to $b_{\hat{s}}$, possibly the period just preceding it.

Hereafter, we call the first $\hat{l}$ periods of shift $\hat{s}$ the *uncovered demand block*.
Each company has its own set $\mathcal{D}$ of possible types of decision that can be applied to update its

personnel schedule following a minor disruption $(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$. Each type $d \in \mathcal{D}$ is associated with a set of candidates $\mathcal{Q}_d(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$. A candidate $q \in \mathcal{Q}_d(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$ is a subset $\mathcal{E}^q$ of employees (possibly just one) excluding employee $\hat{e}$ whose schedules can be modified on day $\hat{h}$ to address the disruption without violating any feasibility constraint. Because the disruption is minor, the modifications must only impact a limited number of employees. Furthermore, because a real-time solution is sought, they must only apply to day $\hat{h}$. Note that, given the large number of possible shifts in the retail context and the small size of the disruption, it is almost always possible to adjust the employee schedules only on this day without impacting the feasibility over the whole horizon. Nevertheless, the chosen decision may result in a non-optimal distribution of the working time between the employees (for instance, some employees might fall into overtime for the week). In this case, the manager re-optimizes the schedule at the end of the day to better distribute the working time among the employees. In this paper, we focus on the real-time decision to make on day $\hat{h}$.

Each decision type $d \in \mathcal{D}$ incurs a managerial cost γ_d, which usually accounts for the time spent implementing the schedule changes. These costs can also be used to represent the preferences of the manager for certain decision types. The personnel re-scheduling problem consists of determining the candidate among all decision types that yields the smallest cost increase. This increase is computed as the difference between the cost of the modified schedule (including the managerial cost) and that of the planned schedule. Here, the modified schedule is that obtained after re-optimizing the schedule at the end of the day as discussed in the previous paragraph.

In this paper, we consider the following five basic decision types, i.e., $\mathcal{D} = \{d_1, \ldots, d_5\}$, which are among the most commonly used in practice.

— d_1 : Extend the planned shift $s \in \mathcal{S}_e^{\hat{h}}$ of an employee e working on day $\hat{h}$ to cover the uncovered demand block. Shift s must end at period $\hat{p}$ or after but before $b_{\hat{s}}$.

— d_2 : Assign a shift $s \in \mathcal{S}_e^{\hat{h}}$ that covers the uncovered demand block to an employee e who has a planned day off on day $\hat{h}$. This is possible only if employee e has sufficient time between $\hat{p}$ and $b_{\hat{s}}$ to reach work from home.

— d_3 : Switch shift $\hat{s}$ with the planned shift $s \in \mathcal{S}_e^{\hat{h}}$ of an employee e who is planned to start after the late arrival of employee $\hat{e}$, i.e., such that $b_s \geq b_{\hat{s}} + \hat{l}$. Here also, this is possible only if employee e has sufficient time between $\hat{p}$ and $b_{\hat{s}}$ to arrive at work.

— $d_4(\eta)$: Extend or move forward the planned shifts of at most η employees whose shifts are scheduled to end between $\hat{p}$ and $b_{\hat{s}}$. Figure 4.1 gives an example where moving forward the shift of employee e_1 might be necessary to avoid reaching the maximum working time per week. Parameter η is set by the manager and limits the number of employees involved in this type of decision. Note that type d_4 generalizes type d_1 as

$d_4(1) = d_1$.

— $d_5(\eta)$: Switch the planned shifts of at most $\eta + 1$ employees (including employee $\hat{e}$) such that employee $\hat{e}$ is re-assigned to a shift s with $b_s \geq b_{\hat{s}} + \hat{l}$ (see Figure 4.2). Here again, η imposes a limit on the number of employees involved in the decision (beside employee $\hat{e}$). This decision type generalizes type d_3 as $d_5(1) = d_3$.

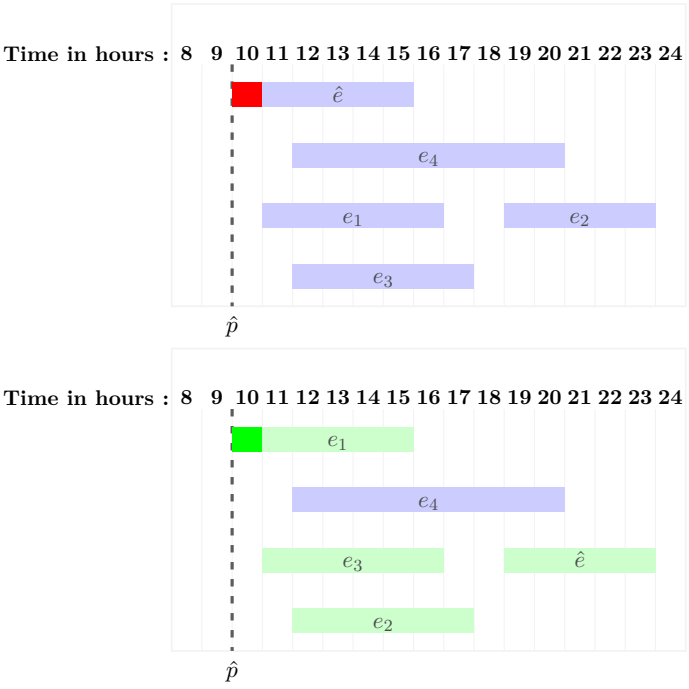

Figure 4.2 Example of a decision of type $d_5(3)$ for a candidate $q = \{e_1, e_2, e_3\}$. The top part shows the planned shifts with the uncovered demand block in red. The bottom part shows the employee shifts after re-scheduling, highlighting in dark green the demand now covered by employee e_1.

Note that, given a candidate, there might exist several combinations of modified shifts to assign to them in order to resolve the disruption. For example, in decision type $d_5(\eta)$, there are several ways to redistribute the shifts among the employees in a candidate. We assume that the best combination can easily be found either by inspection or by using a fast algorithm like the one proposed in the next section for decision $d_5(\eta)$. If this assumption does not hold, the heuristic may be too slow to be used in real time.

The proposed methodology can be adapted to other decision types. For example, when breaks must be scheduled within the shifts, an additional decision type might be to delay the start time of a break to cover a late employee arrival. Note that considering other decision types may yield different computational times. Nevertheless, given that the chosen decision must

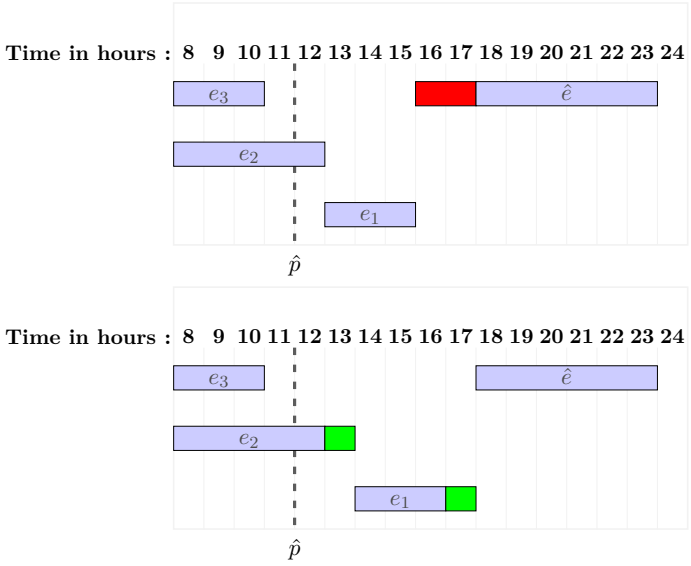

Figure 4.1 Example of a decision of type $d_4(2)$ for a candidate $q = \{e_1, e_2\}$. The top part shows the planned shifts with the uncovered demand block in red. In the bottom part, the shift of employee e_2 has been extended while that of employee e_1 has been moved forward and extended.

impact a limited number of employees on day $\hat{h}$, the number of possible decisions of a given type should be rather limited and the search for the best decision of a given type not much time-consuming.

4.3 Heuristic

In this section, we describe the proposed heuristic to solve the personnel re-scheduling problem. This heuristic is based on dual variable values obtained when solving the personnel scheduling problem. First, we consider the case where a single minor disruption needs to be addressed. Then, we discuss the case where a sequence of minor disruptions must be handled without re-optimizing the personnel scheduling problem (to obtain updated dual values) after each disruption.

4.3.1 A single disruption

Let us consider a minor disruption defined by $(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$. The proposed heuristic is based on an enumeration of all possible candidates for all applicable decision types. It is described in Algorithm 1. In this algorithm, d^* and q^* indicate the best decision type and the best candidate of this type, respectively, yielding an estimated cost increase of Δ^*. When building the set of candidates for a decision type in Step 3, we check for each candidate that the modified shift s of an employee e in this candidate belongs to $\mathcal{S}_e^{\hat{h}}$ and that constraints (4.3)

and (4.5) remain satisfied. In Step 5, the computation of the cost increase estimate Δ_d^q for a candidate q depends on the decision type d as discussed below. Note that, for a given disruption, one can restrict in Step 2, the set of decision types that can be applied. For instance, if the uncovered demand block lasts only 30 minutes, then decision type d_2 might not be interesting given that a guaranteed minimum time largely exceeding 30 minutes might have to be paid. Furthermore, observe that the algorithm can easily be adapted to output not only the best found decision, but several decisions from which the manager can choose the one he/she prefers. In the rest of this section, we propose a fast approach based on the optimal dual values of the linear relaxation of model (4.1)–(4.11) to compute the cost increase estimates Δ_d^q. A minor disruption together with a candidate decision to repair it can be seen as a perturbation of the right-hand side vector b. To evaluate the cost increase yielded by this candidate, sensitivity analysis or parametric optimization in integer linear programming (see, e.g., Geoffrion and Nauss [71], Schrage and Wolsey [72], Jenkins [73], Mitsos and Barton [74]) may be of interest. On the one hand, sensitivity analysis allows to compute a lower bound function on the optimal value when the right-hand side vector changes. This function is computed from all the dual solutions obtained throughout the search tree of a branch-and-cut algorithm when solving the initial planning problem (Schrage and Wolsey [72]). For large trees, this approach can be computationally expensive. On the other hand, parametric optimization aims at computing the optimal values of a family of models defined by parameters (e.g., the right-hand side values) that vary in predetermined ranges. These parametric problems are typically solved as a series of integer linear programs. In our context, this would correspond to solving the planning problem as a highly time-consuming parametric problem. Given that model (4.1)–(4.11) yields very small integrality gaps on the tested real-life instances (see Table 4.1), the linear relaxation solutions seem sufficiently close to their corresponding optimal integer solutions to approximate the candidate cost increases by using linear programming sensitivity analysis theory.

The formulas used to compute the cost increase estimates Δ_d^q are derived as follows. Let us rewrite model (4.1)–(4.11) in the following compact form :

$$\begin{aligned} (\mathrm{P}) \quad \min_{x} \quad & c^\top x \\ \text{subject to :} \quad & Ax = b \\ & x \in \mathbb{N}^n, \end{aligned}$$

where x is the vector of all n decision variables, c is the vector of the corresponding cost coefficients, A is the constraint coefficient matrix, and b is the constraint right-hand side vector. Denote by R the linear relaxation of P. Let B be an optimal basis for R and write

Algorithm 1: Re-scheduling heuristic

input : a minor disruption $(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$;
output: (Δ^*, d^*, q^*) ;
Set $\Delta^* := \infty$, $d^* := NIL$, $q^* := NIL$;
foreach *decision type* $d \in \mathcal{D}$ **do**
  Build the set of candidates $\mathcal{Q}_d(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$;
  foreach *candidate* $q \in \mathcal{Q}_d(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$ **do**
    Estimate the cost increase Δ_d^q ;
    if $\Delta_d^q < \Delta^*$ **then**
      $\Delta^* := \Delta_d^q$, $d^* := d$, $q^* := q$;
Return (Δ^*, d^*, q^*);

$A \equiv [B, N]$, where B and N are the submatrices associated with the basic and nonbasic variables, respectively. Vectors x and c are also partitioned accordingly, i.e., $x^\top \equiv [x_B^\top, x_N^\top]$ and $c^\top \equiv [c_B^\top, c_N^\top]$. Let $\pi^{*\top} \equiv [\pi_B^{*\top}, \pi_N^{*\top}]$ be the vector of dual variables. Basis B yields the optimal solution x^* with $x_B^* = B^{-1}b$ and $x_N^* = \vec{0}$ and is such that $c_N^\top - (\pi_B^*)^\top N \geq (\vec{0})^\top$, where $(\pi_B^*)^\top = c_B^\top B^{-1}$ is the corresponding optimal dual solution. The cost of these primal and dual solutions is $z^* = c_B^\top x_B^* = (\pi_B^*)^\top b$.

From the linear programming sensitivity analysis theory, we know that basis B may remain optimal for a perturbed model obtained by slightly modifying the right-hand side vector b, i.e., by replacing b with $b+\delta b$, where δb is a perturbation vector. In this case, the modified primal solution is given by $\tilde{x}_B^* = B^{-1}(b+\delta b) = x_B^* + B^{-1}\delta b$ and its cost by $\tilde{z}^* = c_B^\top \tilde{x}_B^* = z^* + (\pi_B^*)^\top \delta b$. The cost increase yielded by perturbation vector δb is, thus, equal to $\tilde{z}^* - z^* = (\pi_B^*)^\top \delta b$.

For a candidate q of the decision type d, we compute the cost increase estimate as follows :

$$\Delta_d^q = \gamma_d + (\pi_B^*)^\top \delta b_d^q, \tag{4.12}$$

where the perturbation vector δb_d^q is defined according to the shift variables $X_{es}^{\hat{h}}$, $e \in \mathcal{E}^q \cup \{\hat{e}\}$, $s \in \mathcal{S}_e^{\hat{h}}$, switching either from one to zero or from zero to one in the decision associated with candidate q. More precisely, let A_{es}^h be the column of matrix A associated with a variable X_{es}^h. Furthermore, let $\mathcal{Z}_d^q$ and $\bar{\mathcal{Z}}_d^q$ be the sets of the indices (e, s) of the shift variable $X_{es}^{\hat{h}}$ switching from one to zero and from zero to one, respectively. Then,

$$\delta b_d^q = \sum_{(e,s) \in \bar{\mathcal{Z}}_d^q} A_{es}^{\hat{h}} - \sum_{(e,s) \in \mathcal{Z}_d^q} A_{es}^{\hat{h}}. \tag{4.13}$$

Beside the small observed integrality gaps yielded by model (4.1)–(4.11), we believe that the estimate provided by (4.12) is good because the considered disruption is minor and, thus, the

norm of δb is small and a large number of the columns forming the basis B should remain in the optimal basis of the perturbed model.

In the rest of this section, we present the explicit formula (4.12) for each decision type $d \in \mathcal{D}$. We denote by $\pi_{eh}^{(4.2)}$, $\pi_{e}^{(4.4)}$, $\pi_{eh}^{(4.5)}$, and $\pi_{pw}^{(4.6)}$ the computed optimal dual values associated with constraints (4.2), (4.4)–(4.6), respectively. We also define $\pi_{e0}^{(4.2)} = \pi_{e0}^{(4.5)} = 0$ for all $e \in \mathcal{E}$.

Decision type d_1 **:** For a candidate q of this type, shift $\hat{s}$ for the late employee $\hat{e}$ is replaced by a shift obtained from $\hat{s}$ by omitting the uncovered demand block. Hence, employee $\hat{e}$ works $\hat{l}$ periods less and starts $\hat{l}$ periods later than planned. To cover the uncovered block, the end of the shift s of an employee e is extended by $t_e = \hat{l} + b_{\hat{s}} - f_s - 1$ periods, yielding an additional over-covering for job $w_{\hat{s}}$ in each period from $f_s + 1$ to $b_{\hat{s}} - 1$ if $b_{\hat{s}} - f_s \geq 2$. Consequently, the cost increase estimate is given by

$$\Delta_{d_1}^{q} = \gamma_{d_1} + \hat{l}\pi_{\hat{e}}^{(4.4)} - t_e\pi_{e}^{(4.4)} - \hat{l}\pi_{\hat{e},\hat{h}-1}^{(4.5)} + t_e\pi_{e,\hat{h}}^{(4.5)} + \sum_{p=f_s+1}^{b_{\hat{s}}-1} \pi_{pw_{\hat{s}}}^{(4.6)}.$$

Decision type d_2 **:** For a candidate q of this type, shift $\hat{s}$ for the employee $\hat{e}$ is also replaced by a shorter shift that omits the uncovered demand block. The uncovered block is then covered by a shift s of an employee e that was assigned to a day off on day $\hat{h}$. This shift starts at period $b_{\hat{s}}$ and lasts $\hat{l}$ periods. In practice, such a decision type is admissible only if $\hat{l}$ is greater or equal to the minimum number of periods in a feasible shift. For this decision, the cost increase estimate expresses as :

$$\Delta_{d_2}^{q} = \gamma_{d_2} - \pi_{e\hat{h}}^{(4.2)} + \hat{l}\pi_{\hat{e}}^{(4.4)} - \hat{l}\pi_{e}^{(4.4)} - \hat{l}\pi_{\hat{e},\hat{h}-1}^{(4.5)} + (f_{\hat{s}} + 1 + n^R)\pi_{e,\hat{h}}^{(4.5)} - b_{\hat{s}}\pi_{e,\hat{h}-1}^{(4.5)}.$$

Note that, because a day off on day $\hat{h}$ is planned for employee e, the dual value $\pi_{e\hat{h}}^{(4.2)}$ is equal to 0 and can be omitted.

Decision type d_3 **:** For such a decision candidate q, the shift $\hat{s}$ of employee $\hat{e}$ is switched with a shift s of an employee e. Consequently, the time worked by employee $\hat{e}$ (resp. e) decreases by $l_{\hat{s}} - l_s$ (resp. $l_s - l_{\hat{s}}$). On day $\hat{h}$, the start and end times of the shifts assigned to employee $\hat{e}$ and e are also modified yielding the following cost increase estimate :

$$\Delta_{d_3}^{q} = \gamma_{d_3} + (l_{\hat{s}} - l_s)\pi_{\hat{e}}^{(4.4)} + (l_s - l_{\hat{s}})\pi_{e}^{(4.4)} + (f_s - f_{\hat{s}})\pi_{\hat{e},\hat{h}}^{(4.5)} - (b_s - b_{\hat{s}})\pi_{\hat{e},\hat{h}-1}^{(4.5)} + (f_{\hat{s}} - f_s)\pi_{e,\hat{h}}^{(4.5)} - (b_{\hat{s}} - b_s)\pi_{e,\hat{h}-1}^{(4.5)}.$$

Decision type $d_4(\eta)$ **:** A candidate q of type $d_4(\eta)$ is a subset of m employees $e_1, \ldots, e_m$ assigned on day $\hat{h}$ to shifts $s_1, \ldots, s_m$, respectively, such that $m \leq \eta$, $w_{s_j} = w_{\hat{s}}$ for all $j \in \{1, \ldots, m\}$, $f_{s_j} < b_{s_{j+1}}$ for all $j \in \{1, \ldots, m-1\}$, $f_{s_1} > \hat{p}$, and $f_{s_m} < b_{\hat{s}}$. These employees are re-scheduled to new shifts $s_1^*, \ldots, s_m^*$, respectively. Thus, their work time decreases by

$l_{s_j} - l_{s_j^*}$, $j \in \{1, \ldots, m\}$, and their shift start and end times are modified. Furthermore, the new shifts can yield an additional over-covering for job w in some periods. The set of these periods are denoted $\mathcal{P}_q^V$. For this candidate, the cost increase estimate is given by :

$$\Delta_{d_4}^q = \gamma_{d_4} + \hat{l}\pi_{\hat{e}}^{(4.4)} + \sum_{j=1}^{m} \left[(l_{s_j} - l_{s_j^*})\pi_{e_j}^{(4.4)} + (f_{s_j^*} - f_{s_j})\pi_{e_j,\hat{h}}^{(4.5)} - (b_{s_j^*} - b_{s_j})\pi_{e_j,\hat{h}-1}^{(4.5)} \right] + \sum_{p \in \mathcal{P}_q^V} \pi_{pw_{\hat{s}}}^{(4.6)}.$$

Decision type $d_5(\eta)$ **:** For this type, a candidate q is an ordered subset of m employees $e_1, \ldots, e_m$ assigned on day $\hat{h}$ to shifts $s_1, \ldots, s_m$, respectively, such that $1 \leq m \leq \eta$, $w_{s_j} = w_{\hat{s}}$ for all $j \in \{1, \ldots, m\}$, $b_{s_j} \geq b_{\hat{s}}$ for all $j \in \{1, \ldots, m\}$, and $\max_{j \in \{1,\ldots,m\}} b_{s_j} \geq b_{\hat{s}} + \hat{l}$. Setting $e_0 = \hat{e}$ and $s_0 = \hat{s}$, shifts $s_0, \ldots, s_{m-1}$ are reassigned to employees $e_1, \ldots, e_m$, respectively, and shift s_m to employee e_0 (assuming $b_{s_m} \geq b_{\hat{s}} + \hat{l}$). Therefore, the estimate of the cost increase for candidate q is equal to :

$$\Delta_{d_5}^q = \gamma_{d_5} + \sum_{j=0}^{m} \left[(l_{s_j} - l_{s_{j-1}})\pi_{e_j}^{(4.4)} + (f_{s_{j-1}} - f_{s_j})\pi_{e_j,\hat{h}}^{(4.5)} - (b_{s_{j-1}} - b_{s_j})\pi_{e_j,\hat{h}-1}^{(4.5)} \right],$$

where $j - 1$ is assumed to be equal to m when $j = 0$.

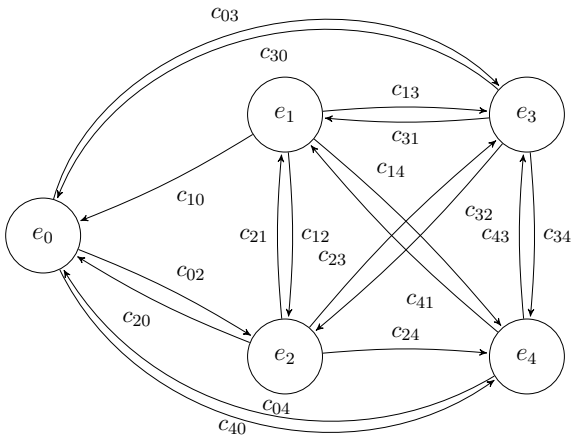

Figure 4.3 Example of a network (with $n^N = 4$) used for decision type $d_5(\eta)$

Given that the number of candidates for decision type $d_5(\eta)$ can be large, the problem of finding the best decision of this type can be formulated as a network flow problem. Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a network, where $\mathcal{N}$ and $\mathcal{A}$ are its node and arc sets, respectively. In $\mathcal{N}$, there is a node for $e_0 = \hat{e}$ and for each employee e that is scheduled to work a shift $s \in \mathcal{S}_e^{\hat{h}}$ on day $\hat{h}$ such that $w_s = w_{\hat{s}}$ and $b_s \geq b_{\hat{s}}$. Let n^N be the number of these latter employees that

we denote e_i, $i \in \{1, \ldots, n^N\}$. Their respective shifts are denoted s_i, $i \in \{1, \ldots, n^N\}$. In $\mathcal{A}$, there exists an arc between every pair of nodes e_i and e_j if employee e_i can be assigned to shift s_j. This arc (e_i, e_j) represents the re-scheduling of employee e_i to the shift s_j and bears a cost

$$c_{ij} = (l_{s_i} - l_{s_j})\pi^{(4.4)}_{e_i} + (f_{s_j} - f_{s_i})\pi^{(4.5)}_{e_i,\hat{h}} - (b_{s_j} - b_{s_i})\pi^{(4.5)}_{e_i,\hat{h}-1}.$$

Figure 4.3 shows an example of a network $\mathcal{G}$ where $n^N = 4$. This network is almost complete : only the arcs (e_0, e_1) and (e_4, e_2) are missing. These arcs may not exist because, for example, shift s_1 begins too early for the late employee e_0 and shift s_2 ends too late for employee e_4 in order to respect the minimum rest time between days $\hat{h}$ and $\hat{h} + 1$.

Given a network $\mathcal{G}$, the problem of finding the best decision of type $d_5(\eta)$ consists of finding a least-cost elementary circuit passing through node e_0 and containing at most $\eta + 1$ arcs. This problem can be modeled as the following integer program :

$$\text{Minimize} \quad \sum_{(e_i,e_j)\in\mathcal{A}} c_{ij}x_{ij} \tag{4.14}$$

$$\text{subject to :} \quad \sum_{(e_0,e_i)\in\mathcal{A}} x_{ij} = 1, \tag{4.15}$$

$$\sum_{(e_i,e_j)\in\mathcal{A}} x_{ij} = \sum_{(e_j,e_i)\in\mathcal{A}} x_{ji}, \qquad \forall e_i \in \mathcal{N} \tag{4.16}$$

$$\sum_{(e_i,e_i)\in\mathcal{A}} x_{ij} \leq \eta + 1, \tag{4.17}$$

$$\sum_{(e_i,e_j)\in\mathcal{A}\,|\,e_i,e_j\in S} x_{ij} \leq |S| - 1, \qquad \forall S \subset \mathcal{N} \setminus \{e_0\} \text{ such that } |S| \leq \eta \tag{4.18}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall (e_j, e_i) \in \mathcal{A}, \tag{4.19}$$

where binary variable x_{ij}, $(e_i, e_j) \in \mathcal{A}$, takes value 1 if arc (e_i, e_j) is selected in the solution and 0 otherwise. Constraints (4.15) and (4.16) ensure that the solution forms a circuit passing through node e_0. Constraint (4.17) impose a maximum of $\eta + 1$ arcs in this circuit. Constraints (4.18) forbid sub-circuits and enforce elementarity. This model is solved by a commercial mixed-integer programming (MIP) solver. Note that, because there may be a huge number of constraints (4.18), only those for which $|S| \leq 5$ are considered initially. The others are generated as needed. The computed optimal circuit indicates the employees involved in the decision and the switching order of their corresponding shifts. The cost increase estimate is given by the sum of the costs of the arcs forming this circuit plus the managerial cost γ_{d_5}.

In Algorithm 1, decision type $d_5(\eta)$ is, therefore, treated differently than the other types and

omitted from the loop in Step 2. After this loop, the integer linear program based on network $\mathcal{G}$ is constructed and solved using a commercial mixed-integer programming solver. The cost increase estimate is then compared to the best one found for the other types.

It should be noted that the minimum rest time constraints (4.5) are often not binding in practice and their corresponding dual values $\pi_{e,h}^{(4.5)}$ are, thus, often equal to 0. To speed up the proposed heuristic, we have decided to omit the terms involving these dual values in all cost increase estimates presented above. For certain decision types, the resulting speed up is easy to evaluate. For instance, for decision type d_3, one can observe that computing $\Delta_{d_3}^q$ for a candidate q requires 12 additions/subtractions and 6 multiplications if these dual values are considered, whereas only 4 additions/subtractions and 2 multiplications are required without them, yielding an approximate speed up of a factor 3. The speed up is similar for type d_2 and somewhat less for type d_4. For type d_1, the speed up factor is slightly less than 2, whereas for type d_5, it is difficult to evaluate given that this simplification only impact network construction. Note also that, when building the candidate set in Step 3 of Algorithm 1, some feasibility tests are performed to detect infeasible candidates. For instance, if a candidate includes an employee e whose shift needs to be extended, then the minimum rest time constraint (4.5) associated with employee e on day $\hat{h}$ is checked to determine whether this extension is feasible. If this is not the case, the candidate is rejected. Otherwise, this constraint is not binding, its dual value is thus equal to 0 (by the complementary slackness theorem) and the candidate may be kept depending on the other constraints. Consequently, omitting this dual from the computation of the cost increase estimate incurs no error in this particular but frequent case. An error can occur when, in a candidate, the start of the shift of an employee is moved forward while its associated constraint (4.5) on day $\hat{h}-1$ is binding or the end of its shift is moved backward while its associated constraint (4.5) on day $\hat{h}$ is binding.

4.3.2 A sequence of disruptions

To solve the personnel re-scheduling problem, we proposed in the previous section a heuristic that is based on the dual values obtained when solving the personnel scheduling problem, that is, when planning the initial schedule. Typically, more than one minor disruption occurs during a week. Hence, we can apply the heuristic described in Algorithm 1 for each disruption as they occur. However, when solving the personnel re-scheduling problem for the second and subsequent disruptions, the dual values derived from the initial personnel scheduling problem are not accurate anymore given that the schedules of some employees have changed when resolving the previous disruptions. In fact, the dual values should be updated after each disruption so that they can reflect the schedule changes. Solving an updated per-

sonnel scheduling problem might not be applicable, for example, when there is not enough time between two disruptions. Furthermore, this personnel scheduling problem would require additional constraints to enforce the current schedule, yielding a very different dual solution. In this section, we propose a simple procedure to update the dual values without solving a personnel scheduling problem.

The formulas used to estimate the cost increase of each candidate of each decision type involve the dual variables $\pi_e^{(4.4)}$, $e \in \mathcal{E}$, and $\pi_{pw}^{(4.6)}$, $p \in \mathcal{P}$, $w \in \mathcal{W}$. Observe, however, that for a given minor disruption $(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$, not all decisions may yield additional over-coverings and, if this occurs, the additional over-coverings take place in between periods $\hat{p}$ and $b_{\hat{s}}$ for task $w_{\hat{s}}$. Consequently, most probably, only the dual values $\pi_{pw}^{(4.6)}$ for a period $p \in [\hat{p}+1, b_{\hat{s}-1}]$ and task $w_{\hat{s}}$ can be affected by this disruption. Given that most subsequent disruptions will become known later than $b_{\hat{s}}$ or will involve a task $w \neq w_{\hat{s}}$, there is no need to update the dual values $\pi_{pw}^{(4.6)}$, $p \in \mathcal{P}$, $w \in \mathcal{W}$. The proposed procedure, thus, concentrates on the dual values $\pi_e^{(4.4)}$, $e \in \mathcal{E}$.

To update these dual values, we first assume that there exists a real function ψ such that, for each employee $e \in \mathcal{E}$, we have $\pi_e^{(4.4)} \backsimeq \psi(L_e)$, where $L_e = \sum_{k \in \mathcal{K}^L} L_e^k$ is the total time worked by employee e in the current solution. We propose to find a function $\tilde{\psi}$ to approximate function ψ. Given that the shape of ψ is unknown, we resort to a nonparametric regression method to determine $\tilde{\psi}$, namely, the *Multivariate Adaptive Regression Splines* (MARS) method introduced by Friedman [75] in 1991. To the best of our knowledge, this method has never been used to update dual values. We chose it because the observed points $(L_e, \pi_e^{(4.4)})$ for our datasets (see, for example, Figure 4.5 for dataset I_2) show that this function is linear on certain intervals and nonlinear on others. The MARS method is known to model adequately such irregularities. To approximate a real function $f(x_1, \ldots, x_n)$ of one or several independent variables, the MARS method assumes that the approximate function $\tilde{f}$ belongs to a vectorial space generated by a set of basis functions. These basis functions can take the form of a constant, a linear spline $(x \mapsto (x-t)_+$ where t is a constant and the $+$ subscript indicates a zero value for a negative argument), a quadratic spline $(x \mapsto (x-t)_+^2)$ and a product of these functions. This vectorial space is larger than the one used for parametric regression methods, which may explain its efficiency with respect to classical regression methods. The MARS method determines a model for the approximate function $\tilde{f}$ in two phases. First, it selects iteratively the basis functions. Second, it eliminates iteratively the components that do not have a large impact on the estimation error.

In our case, to find an approximate function $\tilde{\psi}$, we consider the set of sample points $\{(L_e, \pi_e^{(4.4)}) | e \in \mathcal{E}\}$ obtained from the linear relaxation of the initial personnel scheduling problem. Then, we apply the MARS method on this set of points to yield a function $\tilde{\psi}$ such that $\pi_e^{(4.4)} \backsimeq \tilde{\psi}(L_e)$

for all $e \in \mathcal{E}$. Given this approximate function, the dual values are updated before resolving each disruption occurring during the same week except the first one. The decision made to resolve disruption k modifies the schedules of a subset of employees denoted $\mathcal{E}^k$ and incurs a variation of λ_e^k in the total number of periods worked by employee $e \in \mathcal{E}^k$. Before resolving each disruption $k \geq 2$, the dual value update is performed as follows :

$$\left(\pi_e^{(4.4)}\right)^k = \begin{cases} \tilde{\psi}(L_e + \sum_{i=1}^{k-1} \lambda_e^i) & \text{if } e \in \mathcal{E}^{k-1} \\ \left(\pi_e^{(4.4)}\right)^{k-1} & \text{otherwise,} \end{cases}$$

where $\left(\pi_e^{(4.4)}\right)^{k-1}$, $e \in \mathcal{E}$, are the dual values used to solve disruption $k-1$ and $\left(\pi_e^{(4.4)}\right)^1 = \pi_e^{(4.4)}$, $e \in \mathcal{E}$.

4.4 Computational experiments

To assess the efficiency of the re-scheduling heuristic described in Algorithm 1 and of the procedure updating the dual values introduced in Section 4.3.2, we conducted intensive computational experiments. The results of these experiments are reported in Section 4.4.2 for the single-disruption case and in Section 4.4.3 for the multiple-disruption case. Beforehand, we describe in Section 4.4.1 the datasets used for these tests and how the disruption scenarios were generated, resulting in instances with very small integrality gaps. Finally, in Section 4.4.4, we present some computational results for instances that have larger integrality gaps.

All algorithms were implemented in C++ and all integer programs were solved using the CPLEX MIP solver, version 12.6.1.0. All tests were executed on a Linux machine equipped with an 8-core Intel Core i7 processor clocked at 3.4GHz and a RAM of 16Gb.

4.4.1 Datasets and disruption scenarios

To perform our tests, we use seven real-world datasets provided by our industrial partner that consider a one-week horizon divided into 15-minute periods. Note that the length of the horizon and the length of the periods do not impact the computational time of the proposed heuristic. Indeed, the algorithm evaluates possible decisions occurring only on the day of the disruption and, for each decision type, the number of candidates does not depend on the period length.

Table 4.1 gives the characteristics of the seven datasets and some statistics on the solution process for computing the initial schedule : in order, the number of employees, the number of jobs, the numbers of variables and constraints model (4.1)–(4.11), the integrality gap yielded by this model (given by $(z_{IP} - z_{LP})/z_{LP}$, where z_{IP} and z_{LP} are the optimal values for the

integer program and its linear relaxation), the numbers of branch-and-bound nodes explored and cuts generated throughout the solution process, and the total computational times in seconds. We observe that the gaps are very small and support our assumption that the dual values derived from the linear relaxation solution carry valuable information with respect to the integer solution. The numbers of cuts and branch-and-bound nodes show, however, that the linear relaxation solutions are not integer and, therefore, the dual values do not necessarily provide a perfect information.

For the single-disruption case, our tests focused on one decision type at a time. For each type $d \in \mathcal{D}$ and dataset I_k, $k \in \{1, \ldots, 7\}$, 30 minor disruptions $(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$ were generated as follows. First, day $\hat{h}$ is selected using a uniform distribution over $\mathcal{H}$. Second, employee $\hat{e}$ is drawn from a uniform distribution over the set of employees working on day $\hat{h}$. Shift $\hat{s}$ is the shift worked by employee $\hat{e}$ on day $\hat{h}$. Third, the length of the lateness $\hat{l}$ in number of periods is chosen using another uniform distribution in the interval $[2, l_{\hat{s}} - 4]$. Fourth, the period $\hat{p}$ at which the manager learns that employee $\hat{e}$ will be late is drawn from a uniform distribution in the interval starting at the beginning of day $\hat{h}$ and ending at period $b_{\hat{s}} - 1$ if $d \in \{d_1, d_4(\eta)\}$. For the other decision types, period $\hat{p}$ has no impact on the optimal decisions as long as it provides enough time for an employee to reach work from home. Finally, the set of candidates $\mathcal{Q}_d(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$ is built. If it contains less than three candidates, the disruption is rejected because the corresponding re-scheduling problem is considered too simple and, after removing employee $\hat{e}$ from the possible choices on day $\hat{h}$, the process is repeated from the second step to find a new disruption on day $\hat{h}$.

For the multiple-disruption case, we also consider for testing purposes one decision type at a time, i.e., all disruptions will be resolved using the same decision type. For each type $d \in \mathcal{D}$ and each dataset I_k, $k \in \{1, \ldots, 7\}$, we generated 30 sequences of minor disruptions. To determine each sequence, we first draw a number of potential disruptions n^D from a uniform distribution in $[2, 50]$. Then, for each $j \in \{1, \ldots, n^D\}$, we try to generate a minor disruption as in the single-disruption case described above, except that there is no repetition if the disruption is rejected, i.e., if the candidate set contains less than three candidates. Every time that a disruption $(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$ is accepted, we force the next disruption to occur later than at period $b_{\hat{s}}$ by removing from the possible choices all days $h < \hat{h}$ and, if day $\hat{h}$ is drawn again, all employees e working a shift s on day $\hat{h}$ such that $b_s \leq b_{\hat{s}}$. Note that the proposed approach can easily handle multiple decision types simultaneously at the expense of increasing the computational time.

Table 4.1 Characteristics of the datasets

Dataset	No. emp.	No. jobs	No. var.	No. const.	Gap (%)	No. BB	No. cuts	T (s)
I_1	15	5	118229	9708	0.06	6	7	58
I_2	25	5	114928	9908	0.06	1	14	30
I_3	29	6	193092	8644	0.00	1	18	116
I_4	32	3	75327	5248	0.10	523	88	49
I_5	49	7	204090	13076	0.11	294659	97	6579
I_6	95	7	190586	13996	0.14	38012	309	508
I_7	191	44	852312	61612	0.03	454476	361	69432

4.4.2 Computational results for the single-disruption case

As mentioned above, we tested each decision type $d \in \mathcal{D}$ separately. This allows to see if the cost increase estimate computed by the re-scheduling heuristic is accurate for each type individually. For each decision type, 30 disruptions are generated for each dataset, yielding a total of 210 test instances for each type. For decision type $d_4(\eta)$, we set $\eta = 2$ because it is difficult to find more than two employees in a candidate. For type $d_5(\eta)$, we chose to set $\eta = \left|\mathcal{Q}_{d_3}(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})\right|$ for a disruption $(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$, allowing the largest flexibility in this case. Note that, because we do not compare decisions of different types, the managerial costs γ_d, $d \in \mathcal{D}$, are irrelevant and have been set to 0.

Consider a decision type $d \in \mathcal{D}$ and an instance for this type whose disruption is defined by $(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$. Let $q^* \in \mathcal{Q}_d(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$ be a candidate yielding the least cost increase estimate $\Delta_d^{q^*}$ for this instance. To assess the quality of the corresponding decision, we will compare it to the decisions yielded by two other methods for addressing the same disruption, namely, a greedy and an exact method.

The greedy method corresponds to a simple procedure that might be used in practice by a manager who does not rely on an algorithm to resolve a disruption. The candidate $q^G \in \mathcal{Q}_d(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$ returned by this method depends on the decision type :

- d_1 : Choose the candidate (an employee) whose shift ends the latest before $b_{\hat{s}}$. If several candidates end at the same period, then choose the one with the least working time during the horizon.
- d_2 : Choose the candidate (an employee having a day off) with the least working time during the horizon.
- d_3 : Choose the candidate whose shift begins the earliest after $b_{\hat{s}} + \hat{l} - 1$, i.e., the last period of the uncovered demand block.
- $d_4(\eta)$: Choose the same candidate as for type d_1.
- $d_5(\eta)$: Choose the same candidate as for type d_3.

The exact method is based on the assumption that the disruption is resolved by modifying

shifts on day $\hat{h}$ only but the manager may revise the employee schedules for the rest of the horizon at the end of day $\hat{h}$. Indeed, this often occurs in practice because the decision made in real time to address a disruption may bring an employee in overtime if he/she works the rest of his/her scheduled shifts. The manager may want to cut this overtime as much as possible. In this case, schedule changes affecting one or several employees are applied. To fully test the robustness of the decisions proposed by our re-scheduling heuristic, we assume for our tests that these changes might impact all employees. Consequently, the exact method consists of solving a personnel scheduling problem (through model (4.1)–(4.11)) restricted as follows. All planned shifts scheduled on a day prior to $\hat{h}$ are fixed. On day $\hat{h}$, all planned shifts starting before period $\hat{p}$ are also fixed, while the others are considered except shift $\hat{s}$. On day $\hat{h}$, we also consider all shifts associated with a candidate in $\mathcal{Q}_d(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$. Finally, all shifts, planned or not, are considered for the days after day $\hat{h}$.

To compare fairly the decisions proposed by the three methods, we produced two new solutions, one from each decision imposed by the re-scheduling heuristic and the greedy heuristic. These new solutions were obtained by fixing the corresponding decision on day $\hat{h}$ and by re-optimizing the rest of the week as in the exact method. Let σ^i, $i = R, G, E$, be the solutions derived from the re-scheduling heuristic ($i = R$), the greedy heuristic ($i = G$), and the exact method ($i = E$). Furthermore, for $i = R, G, E$, let $z_d(\sigma^i)$ be the cost of solution σ^i and ζ_d^i be the cost increase between $z_d(\sigma^i)$ and the cost of the initial planned solution.

Given the large number of results obtained (3 methods, 5 decision types and 210 instances for each method and type), we present them in an aggregate form which allows to highlight the performance of the re-scheduling heuristic for each decision type. For each decision type $d \in \mathcal{D}$ and each instance, we compare the values ζ_d^R, ζ_d^G, and ζ_d^E and classify the instance into one of the three following categories. If $\zeta_d^R = \zeta_d^E$, we say that the decision proposed by the re-scheduling heuristic is *optimal*. If $\zeta_d^E < \zeta_d^R \leq \zeta_d^G$, we say that the re-scheduling heuristic obtained a *success*. Otherwise ($\zeta_d^R > \zeta_d^G$), the result is categorized as a *failure*.

The results by decision type and success category are reported in Table 4.2, where the last line also gives the results when aggregating all decision types. For each decision type $d \in \mathcal{D}$, the average number of candidates considered per instance, denoted μ_d, is also indicated and varies between 4.7 and $6 \cdot 10^7$. The results are impressive : the re-scheduling heuristic computes an optimal decision for more than 95% of the instances and yields a failure in less than 2% of the cases. Furthermore, for more than 25% of the instances, this heuristic finds a better solution than the greedy heuristic. We observed that a large number of failures occur for instances based on dataset I_3. For this dataset, the ψ function is almost constant as shown in Figure 4.4, contrarily to the other datasets (for example, see Figure 4.5 for dataset I_2). In fact, in the initial schedule, the numbers of periods worked by most employees fall on the

Table 4.2 Success rate by category of success for each decision type

Decision type	μ_d	Rate by category (%)		
		Optimal	Success	Failure
d_1	5.6	97.6	2.4	0.0
d_2	4.7	97.1	0.5	2.4
d_3	6.1	94.3	1.9	3.8
d_4	7.9	94.8	5.2	0.0
d_5	$\simeq 6 \cdot 10^7$	91.8	5.3	2.9
All	-	95.1	3.1	1.8

same step $k \in \mathcal{K}^L$ of the labor cost step-wise function. When the duals $\pi_e^{(4.4)}$, $e \in \mathcal{E}$, are all equal, the cost increase estimates Δ_d^q for all candidates $q \in \mathcal{Q}_d(\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$ of a decision type $d \in \{d_2, d_3, d_5\}$ are all equal and, therefore, all decisions seem equivalent even if this is not the case. Consequently, poor dual information seems to cause the observed failures.

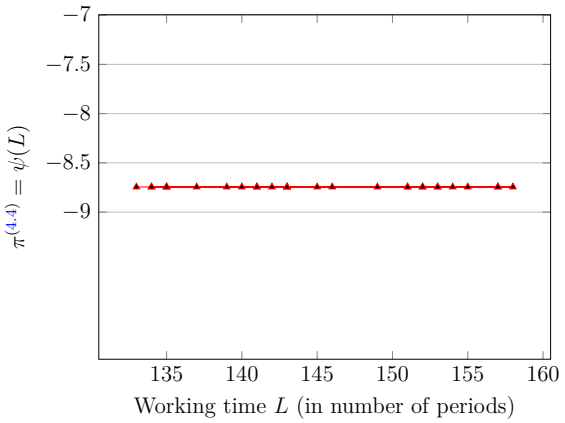

Figure 4.4 Function ψ for dataset I_3

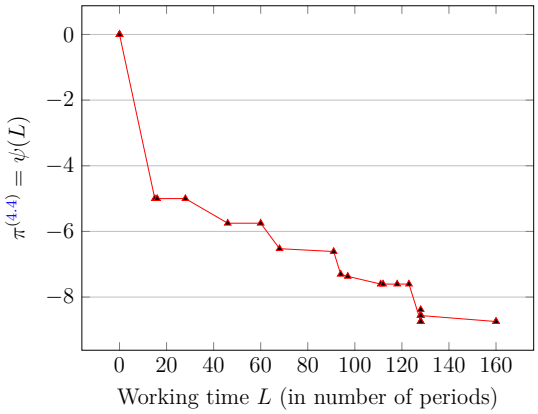

Figure 4.5 Function ψ for dataset I_2

To better compare the re-scheduling and the greedy heuristic, we report in Table 4.3 for each decision type and for all of them together, the averages ζ_{avg}^R and ζ_{avg}^G of the cost increases for both heuristics, as well as the p-value of a Welch t-test. This test, which is adequate when comparing samples with unequal variances, allows to determine the statistical significance of the difference between the two averages. A p-value smaller than 0.05 indicates that the difference is significant. From these results, we observe that the re-scheduling heuristic yields for every decision type a lower average cost increase than the greedy heuristic. Furthermore, the p-values indicate that the domination of the re-scheduling heuristic is statistically significant. This domination is more significant for the more complex decision types d_3 and d_5. It should be noted that, when the solution provided by the greedy heuristic is worse than the one computed by the re-scheduling heuristic, it can be of very poor quality, while the

opposite rarely occurs.

Table 4.3 Additional results for the single-disruption case

	d_1	d_2	d_3	d_4	d_5	All
ζ^R_{avg}	0.31	0.37	0.63	0.34	1.16	0.56
ζ^G_{avg}	1.67	0.56	5.04	1.53	6.56	3.07
p-value	$4 \cdot 10^{-7}$	$1 \cdot 10^{-1}$	$3 \cdot 10^{-12}$	$6 \cdot 10^{-6}$	$5 \cdot 10^{-12}$	$2 \cdot 10^{-16}$

The observed computational times for the re-scheduling heuristic are sufficiently small to use this heuristic in real time. Indeed, for the tested 1050 instances, the average computational time is less than one second. The computational times for the greedy heuristic are also very small, while the exact method can require computational times that are much larger for certain instances (more than 10 minutes in some cases). We also ran tests with the exact algorithm applied only on day $\hat{h}$ for each specific decision type and obtained larger computational times than the re-scheduling heuristic. For instance, for decision type d_5, the computational times range between 10 and 30 seconds for the instances derived from datasets I_5 and I_6 (i.e., with 49 and 95 employees, respectively) and between 30 and 175 seconds for the I_7 ones with 191 employees.

Table 4.4 Comparison of the three best solutions derived from the re-scheduling heuristic for decision type $d_4(2)$

Candidate rank	Cost ranking			Cost difference with best		
	First	Second	Third	Avg.	Min.	Max.
First	30	0	0	0.0	0.0	0.0
Second	7	21	2	56.0	0.0	174.5
Third	3	1	26	124.1	0.0	316.1

Finally, in Table 4.4, we report additional results to evaluate if the ranking of the decisions performed by the re-scheduling heuristic based on the cost increase estimate corresponds to the ranking of the costs obtained after re-optimizing the schedule for the rest of the week. These results focus on the 30 instances tested for decision type $d_4(2)$ and dataset I_6 (similar results were observed for the other instances and datasets). For each instance, we considered the three solutions derived from the best three candidates (in this order) identified by this heuristic and ranked them according to their costs. For each candidate rank, Table 4.4 indicates the total number of times that the corresponding solution is ranked first, second or third according to the solution costs. Note that, for an instance, two or three candidates can obtain the same rank if they have the same cost. This table also provides for each candidate rank the average cost difference with respect to the best cost found, as well as the minimum

and maximum differences over the 30 tested instances. From these results, we deduce that the cost increase estimate seems to be positively correlated with the decision quality.

4.4.3 Computational results for the multiple-disruption case

For the multiple-disruption case, we compare three algorithms, namely, the re-scheduling heuristic without updating the dual values, the re-scheduling heuristic with an update of the dual values after each disruption, and the greedy heuristic. For each decision type and each instance, the experiment unfolds as follows. For each disruption (treated in chronological order), the three heuristics and the exact method are applied to propose a decision each. Three solutions derived from the decisions of the three heuristics are computed as in the single-disruption case by fixing the decision for the disruption day and by re-optimizing the end of the week as in the exact method. These solutions and that of the exact method are denoted σ^i, $i = R, U, G, E$ (U for re-scheduling with dual value updates). Their costs are denoted by $z_d(\sigma^i)$, $i = R, U, G, E$, and we compute the nonnegative cost difference $z_d(\sigma^i) - z_d(\sigma^E)$, $i = R, U, G$, of each heuristic solution with respect to the exact solution. Next, the shift modifications on the day of the disruption suggested by the exact method are implemented before moving on to the next disruption. We denote by Γ^i_d, $i = R, U, G$, the sum of the cost differences incurred over all disruptions by the re-scheduling heuristic without and with dual value updates and the greedy heuristic, respectively.

Recall that, for each decision type and each dataset, 30 instances were generated. These instances differ by the sequence of disruptions considered. The results of our experiments are reported in Table 4.5. For each dataset and each decision type, this table specifies the average and the maximum number of disruptions per instance (n^D_{avg} and n^D_{max}), the average and the maximum computational time (T_{avg} and T_{max}) in seconds to resolve one disruption by the re-scheduling heuristic with dual value updates (including the time to update the dual values), the average sums of the cost differences (Γ^i_{avg}, $i = R, U, G$) obtained by the re-scheduling heuristic without dual value updates, with dual value updates and by the greedy heuristic, the standard deviations of these sums (Γ^i_{sd}, $i = R, U, G$), and their maximum values (Γ^i_{max}, $i = R, U, G$).

From these results, we make the following observations. The number of disruptions per instance tends to increase with the dataset size (number of employees). This is due to the rule that a disruption is accepted only if there are at least three candidates to resolve it. Concerning the computational times of the re-scheduling heuristic with dual value updates, the results show that they are very acceptable, reaching a largest average time of 1.64 seconds for datasets involving up to 191 employees. Now, comparing the average values Γ^R_{avg} and Γ^U_{avg} (and their corresponding standard deviations and maximum values), we deduce that

Table 4.5 Results for the multiple-disruption case

Dataset	Decision type	n^D_{avg}	n^D_{max}	Re-scheduling								Greedy		
				T_{avg} (s)	T_{max} (s)	Γ^R_{avg}	Γ^R_{sd}	Γ^R_{max}	Γ^U_{avg}	Γ^U_{sd}	Γ^U_{max}	Γ^G_{avg}	Γ^G_{sd}	Γ^G_{max}
I_1	d_1	2.7	4	0.07	0.12	0	0	0	0	0	0	0	0	0
	d_2	3	3	0.03	0.06	0	0	0	0	0	0	0	0	0
	d_3	2.5	3	0.05	0.07	0.3	0.7	2.6	1	3	11.8	13.9	3	40.3
	d_4	2.6	4	0.05	0.09	0	0	0	0	0	0	0	0	0
	d_5	2.3	3	0.05	0.11	5.1	7.6	26.2	5.7	7.9	26.2	11.4	11.7	36.9
I_2	d_1	2.2	4	0.06	0.09	0	0	0	0	0	0	0.8	2.6	11.4
	d_2	6.1	9	0.02	0.03	**22.2**	20.9	64.3	**0**	0	0	**0**	0	0
	d_3	2.1	4	0.04	0.05	1.4	3.4	12.5	0.7	1.6	7.7	44.1	27.6	97.2
	d_4	2.2	4	0.04	0.10	0	0	0	0	0	0	2.9	4.1	9.1
	d_5	3.3	5	0.05	0.17	8.5	9.3	36.5	4.9	6.9	20.5	38.3	32.1	140.6
I_3	d_1	2.4	4	0.19	0.21	0	0	0	0	0	0	0	0	0
	d_2	7.9	12	0.19	0.21	0	0	0	0	0	0	0	0	0
	d_3	2.3	5	0.23	0.25	8.6	9.6	27	1.5	3.6	12.5	2.8	5.2	18.4
	d_4	2.1	3	0.21	0.27	3.2	4.3	11.8	3.6	4.2	11.8	4.1	4.2	11.8
	d_5	5.3	8	0.20	0.30	7	15.4	63.2	1.2	2.3	9.1	4	12.1	56.4
I_4	d_1	2.5	6	0.10	0.15	0	0	0	0	0	0	0	0	0
	d_2	5	7	0.05	0.07	4.5	5.8	14.9	0	0	0	1.5	3.1	7.5
	d_3	2.7	5	0.07	0.10	0.1	0.4	2	0.1	0.7	4	4.4	6.6	26.4
	d_4	4.9	10	0.08	0.19	1.8	2.5	8.9	0.9	1.6	4	2.8	3.5	11.9
	d_5	4.6	8	0.09	0.21	2.3	2.9	10.9	1.2	2.6	11.9	6	5.5	20.4
I_5	d_1	2.2	4	0.18	0.27	0.2	0.6	2.2	0.2	0.6	2.2	0.2	0.6	2.2
	d_2	4.7	7	0.15	0.17	3.5	8.2	25.8	0	0	0	0.1	0.2	1
	d_3	3.5	7	0.21	0.22	4.5	6	18	3	3.5	11.8	22.9	17.5	59.7
	d_4	2.2	4	0.21	0.29	0.2	0.6	2.2	0.1	0.5	2.2	0.2	0.6	2.2
	d_5	6.1	11	0.24	0.38	11.9	10	38.6	11.3	8.6	40.7	33.5	14.1	58.8
I_6	d_1	2.4	4	0.29	0.35	1.9	4.5	20.6	1.1	4.4	20.6	2.9	5.3	20.6
	d_2	17.4	24	0.11	0.13	**26.7**	16.1	58.1	**8.1**	11.3	24.1	**37**	17.7	71.3
	d_3	6.1	13	0.20	0.23	4.7	6	15..5	2.5	3.5	13.7	56.6	29.2	131.8
	d_4	2.7	5	0.33	0.61	0.7	1.4	4	0	0	0	2.9	5.3	20.6
	d_5	9.6	15	0.34	0.74	30.7	17.8	67	11.6	7.6	27.3	56.5	20.4	98.5
I_7	d_1	7	14	1.42	1.59	11.8	10	38.6	3.1	5.2	14.2	18.7	16.8	69
	d_2	**27.1**	**37**	0.51	0.64	**65.9**	55.9	212	**8.0**	12.2	50.1	**8.1**	13.6	59
	d_3	8.5	18	1.22	1.29	21.1	21.4	71.4	4.5	8.8	24.4	38.7	25.3	99.8
	d_4	7.6	14	**1.64**	2.12	7.0	10.5	33.4	1.1	7.5	17.8	20	18.6	57.8
	d_5	17.9	28	1.54	**2.29**	**33.3**	29.7	107.2	**4.1**	14.1	50.1	**45.3**	35.3	118.6

updating the dual values can be highly useful in several cases, especially for decision type d_5 for which a decision may impact several employees at the same time and, thus, there is a relatively high probability that the schedule of the same employee be modified more than once during the week. We also observe a large impact for decision type d_2 where the number of hours worked by the employee who is asked to work in a day off may change considerably. On the other hand, in certain cases (for instance, dataset I_1 and type d_5), updating the dual values may slightly deteriorate the solution quality. This is not surprising given that we are comparing heuristics. Nevertheless, the improvements obtained for several cases (for instance, dataset I_2 and type d_2 or dataset I_7 and type d_5) are substantial. In all cases, the standard deviations and maximal values show that these observations are consistent throughout the instances.

Next, let us compare the values of Γ^U_{avg} and Γ^G_{avg}. In all cases, we get $\Gamma^U_{avg} \leq \Gamma^G_{avg}$, showing the superiority of the re-scheduling heuristic with dual value updates over the greedy heuristic. In certain cases, the value of Γ^G_{avg} is much larger than the value of Γ^U_{avg}, indicating that the greedy heuristic can compute bad-quality solutions in some cases. Note, however, that without updates, the re-scheduling heuristic does not always yield a better average cost difference Γ^R_{avg} than that of the greedy heuristic Γ^G_{avg}. Finally, from these results, we observe that the average cost difference with respect to the cost increase of the exact method is relatively small. This can be explained by the results obtained for the single-disruption case which showed that the re-scheduling heuristic returns the optimal decision most of the times. On average, the sum of the cost differences decreases by 73.3% when updating the dual values between consecutive disruptions.

To complete the comparison of the three heuristics, we report in Table 4.6 for each decision type and for all of them together, the averages of the cost increases for the three heuristics (ζ^i_{avg}, $i = R, U, G$) and the p-values of two Welch t-tests comparing the averages derived by the R and G heuristics and those obtained by the U and R heuristics. These results clearly show the effectiveness of the re-scheduling heuristic with dual value updates compared to the other two approaches for all decision types and overall. The re-scheduling heuristic without updates outperforms the greedy heuristic except for decision type d_2 where it is the opposite. For this type (assigning the uncovered block to an employee e with a day off), a decision might increase substantially the time worked by this employee, making the information provided by the dual value $\pi^{(4.4)}_e$ less accurate if it is not updated.

Table 4.6 Additional results for the multiple-disruption case

	d_1	d_2	d_3	d_4	d_5	All
ζ^R_{avg}	2.08	18.02	6.14	2.19	14.58	8.60
ζ^U_{avg}	0.72	2.75	2.22	1.13	6.24	2.61
ζ^G_{avg}	3.32	7.13	26.51	4.95	28.32	14.05
p-value (R vs G)	$5 \cdot 10^{-2}$	$9 \cdot 10^{-6}$	$2 \cdot 10^{-16}$	$3 \cdot 10^{-4}$	$7 \cdot 10^{-9}$	$2 \cdot 10^{-9}$
p-value (U vs R)	$1 \cdot 10^{-3}$	$2 \cdot 10^{-10}$	$3 \cdot 10^{-6}$	$3 \cdot 10^{-3}$	$1 \cdot 10^{-8}$	$2 \cdot 10^{-16}$

4.4.4 Results for instances with larger integrality gaps

The integrality gaps for the seven datasets used in the previous tests are very small as reported in Table 4.1, with a maximum of 0.14%. This can be explained by the fact that the solution space is very wide and that the minimum rest constraints (4.5) are not very restrictive given that there is no demand during a certain period of the night. To assess the impact of these integrality gaps on the performance of the heuristics in the multi-disruption case, we slightly modified three arbitrary datasets I_2, I_5 and I_6 to obtain larger gaps. To do so, we considered

a hard constraint which is sometimes used in practice. Each employee $e \in \mathcal{E}$ is associated with a main job $\hat{w}_e \in \mathcal{W}$ and must be assigned to at least $\tau\%$ of its working time to this job. These constraints can be written as follows :

$$\sum_{h\in\mathcal{H}}\sum_{s\in\mathcal{S}_e^h} l_s^{\hat{w}_e} X_{es}^h \geq \frac{\tau}{100} \sum_{h\in\mathcal{H}}\sum_{s\in\mathcal{S}_e^h} l_s X_{es}^h, \qquad \forall e \in \mathcal{E}, \tag{4.20}$$

where $l_s^{\hat{w}_e} = l_s$ if $w_s = \hat{w}_e$ and 0 otherwise. The characteristics of the modified datasets, denoted I_2', I_5' and I_6', are given in Table 4.7. Note that the value of parameter τ has been adjusted for each dataset to yield the largest possible gap for each of them. The resulting gaps are not very large but they are significantly larger than those of the original datasets. The numbers of branch-and-bound nodes and cuts clearly show that the linear relaxation solutions are not very close to the optimal integer solutions.

Table 4.7 Characteristics of the datasets with larger integrality gaps.

Dataset	No. emp.	No. jobs	τ	No. var.	No. const.	Gap (%)	No. BB	No. cuts	T (s)
I_2'	25	5	80	114953	9958	3.42	41342	132	1779
I_5'	49	7	50	204139	13174	2.01	13699	630	18981
I_6'	95	7	65	190681	14186	0.83	24318	230	7344

Table 4.8 Results for the multiple-disruption case on the instances with larger integrality gaps

Dataset	Decision type	n^D_{avg}	n^D_{max}	Re-scheduling									Greedy		
				T_{avg} (s)	T_{max} (s)	Γ^R_{avg}	Γ^R_{sd}	Γ^R_{max}	Γ^U_{avg}	Γ^U_{sd}	Γ^U_{max}	Γ^G_{avg}	Γ^G_{sd}	Γ^G_{max}
I_2'	d_1	2	2	0.07	0.09	0	0	0	0	0	0	0	0	0
	d_2	3	3	0.02	0.03	**41.9**	40.8	122.1	**0**	0	0	**0**	0	0
	d_3	2.2	6	0.03	0.05	**52**	32.2	127.3	**0.9**	2	4.4	**44.3**	30.6	121.2
	d_4	1.2	2	0.04	0.06	0	0	0	0	0	0	0	0	0
	d_5	2.4	4	0.06	0.16	**60.4**	40.8	145.7	**2.7**	3.5	13.5	**42.4**	35.2	126.2
I_5'	d_1	1.9	5	0.17	0.19	0	0	0	0	0	0	0	0	0
	d_2	5.6	8	0.13	0.21	**50.4**	18.8	79.9	**1**	2.2	7.1	**1.5**	2.4	7.1
	d_3	3.8	7	0.18	0.19	**20.1**	21.5	79.6	**1.6**	2.9	7.4	**61.8**	40.84	142.6
	d_4	2.2	4	0.20	0.22	0	0	0	0	0	0	0	0	0
	d_5	6.7	11	0.19	0.20	**40**	28.6	132.4	**5.9**	6	24.1	**58.3**	35.1	149.2
I_6'	d_1	2.3	5	0.20	0.27	3.9	8.5	27.5	3.9	8.5	27.5	9.5	13.3	36.3
	d_2	**9.3**	**13**	0.12	0.13	**49.8**	25.4	94.3	**3.5**	6.4	18.5	**14.7**	12.8	45.8
	d_3	5.2	9	0.19	0.21	**9.2**	10.9	38.4	**1.7**	3	9.9	**84.6**	37.6	179.4
	d_4	2.3	6	0.23	**0.45**	0.2	1	5.4	0.2	1	5.4	6.2	14.3	62
	d_5	7.6	12	**0.26**	0.33	**24.2**	19.9	68.6	**6.3**	6.5	27.5	**82.4**	39.1	184.2

For each modified dataset, we generated 30 disruption scenarios as described in Section 4.4.1 to create 30 instances for each dataset. Each instance was then solved using the re-scheduling heuristics with and without dual value updates and the greedy heuristic. Note, however, that to obtain better results with the re-scheduling heuristic with updates, we use the $\tilde{\psi}$ function to estimate the dual values $\pi_e^{(4.4)}$, $e \in \mathcal{E}$, before resolving the first disruption. The results

of these experiments are reported in Table 4.8. We observe that the quality of the solutions produced by the re-scheduling heuristic without updates for types d_2, d_3 and d_5 is much worse than for the instances used in the previous section. This seems to be due to the poorer quality of the initial dual information. On the other hand, one can notice that the heuristic with dual value updates is as efficient as before : it computes optimal or near-optimal solutions for all tested instances. Therefore, it seems that approximating the dual values through the $\tilde{\psi}$ function before each disruption plays a determining role in the effectiveness of the algorithm for decision types d_2, d_3 and d_5. For the other two types (d_1 and d_4), the two re-scheduling heuristics compute the same solutions. The greedy heuristic does so as well except for the last dataset and type d_4 where it finds worst-quality solutions than the other two heuristics. We conclude this section by reporting in Table 4.9 additional results similar to those presented in the previous section. These results support the observations made above, especially that the re-scheduling heuristic with dual value updates outperforms the other two heuristics in general.

Table 4.9 Additional results for the multiple-disruption case on the instances with larger integrality gaps

	d_1	d_2	d_3	d_4	d_5	All
ζ^R_{avg}	1.29	47.45	27.14	0.06	41.79	23.5
ζ^U_{avg}	1.29	1.57	1.38	0.06	5.22	1.90
ζ^G_{avg}	3.16	5.47	63.56	2.06	61.30	27.1
p-value (R vs G)	$4 \cdot 10^{-2}$	$2 \cdot 10^{-16}$	$2 \cdot 10^{-11}$	$2 \cdot 10^{-2}$	$2 \cdot 10^{-4}$	$6 \cdot 10^{-1}$
p-value (U vs R)	$5 \cdot 10^{-1}$	$2 \cdot 10^{-16}$	$5 \cdot 10^{-13}$	$5 \cdot 10^{-1}$	$2 \cdot 10^{-16}$	$2 \cdot 10^{-16}$

4.5 Conclusion

In this paper, we considered the personnel re-scheduling problem that must be solved in real time after the occurrence of a minor disruption in a context where the employees can be assigned to a wide variety of shifts, starting and ending at various times. To solve this problem, we propose a fast re-scheduling heuristic that can handle five different types of decision. This heuristic is based on the dual values obtained when solving in the planning phase the linear relaxation of the personnel scheduling problem. The computational results obtained on 1050 instances derived from real-world datasets involving up to 191 employees showed the efficiency of this re-scheduling heuristic. It computed in less than one second on average an optimal solution for more than 95% of these instances. We also developed a procedure for updating the dual values after each disruption when several disruptions must be dealt with during the same week. Additional computational results showed the usefulness of this updating procedure which reduced the sum of the gaps between the optimal value

and the heuristic solution value by more than 73% in our tests. Additional tests on instances where the initial dual information is not so accurate showed that the proposed heuristic with dual value updates can still perform very well.

Several research avenues can be pursued after this work, including the following two. First, in certain companies, the planned personnel schedules are not necessarily computed by solving a mixed integer program. They are rather determined manually from past schedules. In this case, no dual values are available and designing an algorithm to determine accurate dual values would be a valuable complement to our work. Second, the proposed re-scheduling heuristic is myopic in the sense that it considers only the current disruption without anticipating future disruptions. Developing a stochastic personnel re-scheduling algorithm that evaluate the expected future costs for each possible decision constitutes a challenging research topic.

Acknowledgments

The authors are grateful to the personnel of Kronos Canadian Systems Inc. for proposing this problem and corresponding datasets. This work was funded by Kronos Canadian Systems and the Natural Sciences and Engineering Research Council of Canada under the grant #RDCPJ 468 716-14. This financial support was greatly appreciated.

CHAPITRE 5 ARTICLE 2 : REAL-TIME BI-OBJECTIVE PERSONNEL RE-SCHEDULING IN THE RETAIL INDUSTRY

R. Hassani, G. Desaulniers et I. Elhallaoui ont écrit cet article et l'ont soumis en 2019 dans la revue *Omega*.

5.1 Introduction

Personnel scheduling arises in many areas such as retail, health, postal services, transportation and industrial production. The personnel scheduling process can vary from one area to another but the goal, generally, remains the same. It consists of constructing the work schedule of a set of employees over a given time horizon to satisfy the demand in employees induced by a set of tasks or jobs. The computed schedule must respect various working, union and legislative constraints, and must be optimal with respect to one or several selection criteria such as the employee salaries, the quality of the work provided or the employee preferences. Finding such a schedule is a very delicate exercise which is better performed using an optimization software.

In practice, personnel scheduling is subject to a great deal of uncertainty. Indeed, employees may be late, absent or the observed demand may also differ from the expected demand for certain periods of the horizon. These incidents, called minor disruptions, are revealed dynamically during the operations. When such a disruption occurs, the planned schedule becomes infeasible and must be updated, generally in real time, to retrieve a feasible schedule. Usually, for a minor disruption, the re-optimized schedule deviates very little from the planned schedule, i.e., it is obtained by applying only a few changes to the current schedule. This is very desirable in practice : the planned work shifts cannot be changed for a large number of employees just because one of them is late. The quality of the new schedule should, therefore, be evaluated with respect to two criteria : the number of modifications made to the planned schedule and the cost generated by these modifications. This leads to a real-time bi-objective personnel re-scheduling problem (RBPRP).

In this paper, we aim at developing an effective heuristic for solving the RBPRP where the demand in employees can vary frequently during the day and the number of possible work shifts is very large. In particular, we have in mind problems arising in the retail industry where the set of feasible shifts may include shifts that can start at various times of the day (e.g., at every 15 minutes between 7h00am and 6h00pm) and last various durations (e.g., between 3 and 9 hours, by increment of 15 minutes). Furthermore, given its variability, the demand is not specified by shifts but rather by short time periods (e.g., 15-minute periods).

The proposed heuristic must be fast and provide to the store manager a small set of updated schedules that are Pareto-optimal with respect to the number of changes brought to the planned schedule and the costs. From this set, the manager can then choose the schedule which seems the best from his/her point of view. The cost of a new schedule is computed by taking into account the management and operating costs on the day of the disruption as well as future costs to be incurred from additional schedule changes that may be needed on the subsequent days to rebalance the employee schedules and avoid weekly overtime as much as possible.

5.1.1 Literature review

Since the 1950s, personnel scheduling has been widely studied in the operations research literature as shown in the comprehensive surveys [5,6,67]. Research on personnel re-scheduling is much more recent as it began in the early 2000s. Most works have addressed the nurse re-scheduling problem which considers a very limited number of possible shifts (for example, from 7am to 3pm, from 3pm to 11pm, and from 11pm to 7am). Furthermore, the demand for a given task is often expressed as a number of nurses required by shift. In this context, a minor disruption corresponds to the absence of one or several nurses for a whole shift. Moz and Pato [54, 55] proposed exact branch-and-bound algorithms to solve this problem. The computational times of these algorithms being disproportionately long, they are not applicable in real time. Various heuristic algorithms dealing with the same problem have also been developed : see the works of Moz and Pato [54, 57], Pato and Moz [58], Maenhout and Vanhoucke [59], and Kitada *et al.* [60, 61]. Although some of these heuristics provide fast computational times, they are more relevant to the health field than to the retail domain which offers much more flexibility on the possible shifts and on the allowable changes to adjust the planned schedule. Note that larger disruptions in a health care setting (induced by a revision of the demand and offer of a whole 8-hour shift) have also been treated by exact branch-and-bound and branch-and-price algorithms in Bard and Purnomo [56, 68].

In 2016, Gross *et al.* [62] worked on re-scheduling doctors in a hospital, after the absence of one of them, by seeking a compromise between the quality of the new schedule and the distance of the latter from the initial schedule. In their setting, the number of work shifts (or duties) is equal to 17, demand is expressed by shift and the planning horizon spans 28 days. They formulated their problem as a mixed integer linear program that can be solved using a commercial branch-and-cut algorithm. They report computational time of up to 21 seconds. Note that to achieve the best compromise between the two objectives, the program was run several times with different parameter choices.

In 2015, Froger [63] studied a personnel re-scheduling problem arising in a retail context

identical to our study except that large disruptions (e.g., when a relatively large variation of the demand is expected over one or several days due to bad weather forecasts or an unplanned sale) are considered instead of minor ones. She modeled the problem as a mixed integer linear program where the variables are defined for each planned shift of each employee and each possible transformation of these shifts according to some modification rules. This work has many similarities with ours. Indeed, the initial personnel scheduling problem is the same and the changes that can be made to the scheduled shifts are also the same. However, the size of the disruptions differs and the time available to deal with them is much less important in our case (a few seconds against a few minutes). Therefore, the solution approach developed by Froger [63] cannot be applied to a minor disruption in real time.

To the best of our knowledge, no works have been published to handle a minor disruption in a retail store or in a similar context, with the exception of the recent paper by Hassani *et al.* [76]. They introduced a heuristic that relies on the dual information obtained while solving the initial planning problem. This information is updated using a regression method after each time that the schedule is adjusted following a disruption. The proposed heuristic returns a small set of re-scheduling options that impact only the day of the disruption, ensuring that the rest of the schedule remains feasible but not necessarily optimal or near-optimal.

5.1.2 Contributions

In this paper, we introduce a new heuristic for the RBPRP that is characterized by efficiency and speed. This heuristic re-optimizes the planned schedule in real time when a minor disruption occurs with the bi-objective of minimizing costs and the number of required changes. Contrarily to what was proposed in Hassani *et al.* [76], the changes here can affect the day of the disruption as well as the subsequent days. Using basic decisions that can generate all possible schedules, our heuristic aims at finding decision sequences, called policies, that lead to Pareto-optimal solutions. All feasible policies can be modeled using a network where each vertex represents a feasible solution and each arc a basic decision. Based on theoretical insights, we significantly reduce the part of this network that is explored to yield a fast heuristic. The resulting heuristic is tested on RBPRP instances derived from real-life datasets and involving up to 95 employees. In more than 98% of the test cases, it can find all the Pareto-optimal solutions in an average computational time of less than one second.

5.1.3 Paper structure

This paper is organized as follows. In Section 5.2, we define the RBPRP and formulate it as a constrained shortest path problem. In Section 5.3, we first present some theoretical results that support the design of the proposed heuristic and then describe this heuristic. In

Section 5.4, we report and analyze the computational results obtained on RBPRP instances derived from real-world datasets. Finally, conclusions are drawn in Section 5.5.

5.2 Problem definition and formulation

In this section, we begin by stating the RBPRP. Then, we introduce some terminology and concepts that are necessary to formulate the problem. Finally, we formulate it as a constrained shortest path problem and highlight the relationship between the underlying network and the convex hull of the set of feasible solutions.

5.2.1 Definition of the RBPRP

To define the RBPRP, we start by stating the personnel scheduling problem that needs to be solved to establish an initial planned schedule. Let $\mathcal{H}$ be a planning horizon (typically, one week or one month), where the days are numbered from 1 to $|\mathcal{H}|$. $\mathcal{H}$ is discretized in periods of equal length (say, 15 minutes) which form the set $\mathcal{P}$, numbered from 1 to $|\mathcal{P}|$. Furthermore, let $\mathcal{W}$ be a set of jobs that need to be covered in each period of $\mathcal{P}$ with a given number of employees that may depend on the period. To cover these demands, a set of skilled employees $\mathcal{E}$ is available. Each employee $e \in \mathcal{E}$ is qualified to work only on a subset of the jobs $\mathcal{W}_e$. For each employee $e \in \mathcal{E}$, a subset of possible mono-job shifts $\mathcal{S}_e$ on which he/she can work is computed a priori by a heuristic procedure and given as an input. A shift $s \in \mathcal{S}_e$ is defined by a starting period $b_s \in \mathcal{P}$, an end period $f_s \in \mathcal{P}$, and a job $w_s \in \mathcal{W}_e$. We denote by $l_s (= f_s - b_s + 1)$ its length, h_s its assignment day (i.e., that containing period b_s), and $e_s \in \mathcal{E}$ its associated employee. Furthermore, for $e \in \mathcal{E}$ and $h \in \mathcal{H}$, we define $\mathcal{S}_e^h \subseteq \mathcal{S}_e$ as the (possibly empty) subset of shifts that can be assigned to employee e on day h.

The planning problem consists of finding feasible work schedules for the employees in $\mathcal{E}$ such that they cover at least cost the demands of each job in $\mathcal{W}$ over the horizon $\mathcal{H}$. A feasible schedule for an employee e is composed of a subset of the available shifts in $\mathcal{S}_e$ such that a certain number of working rules are satisfied. For example, the schedule of each employee must contain at most one shift per day, a minimum number of days off per week, a minimum rest time between two consecutive work shifts and a maximum number of worked hours. Under-coverage of a demand at a given period (i.e., assigning fewer employees than requested) is prohibited but can be avoided by scheduling highly penalized anonymous shifts to be dispatched later to temporary employees. Demand over-coverage is accepted but also penalized. The cost of a complete schedule is computed as the sum of these penalties and labor costs. These costs do not correspond to the employee salaries (otherwise, minimizing costs would result in less working hours for the more experienced employees) but ensure through non-decreasing stepwise functions that the total working time is more or less balanced

among the employees. A detailed definition of this problem over a one-week horizon is given in Hassani *et al.* [76].

The RBPRP must be solved when a minor disruption occurs and makes the planned schedule infeasible. Such a disruption can result, for example, from the lateness or the absence of an employee, the increase/decrease in the demand of a job for a limited number of periods, or an unforeseen event that prevents an employee to finish a work shift. In this paper, we focus on a minor disruption triggered by the lateness of an employee as most of the other situations can be treated similarly (see Hassani *et al.* [76]). A disruption due to a lateness is characterized by :

- the employee $\hat{e} \in \mathcal{E}$ who is late ;
- the day $\hat{h} \in \mathcal{H}$ when the disruption occurs ;
- the planned shift $\hat{s} \in \mathcal{S}_{\hat{e}}^{\hat{h}}$ of employee $\hat{e}$ on day $\hat{h}$;
- the lateness duration $\hat{l}$ in periods ($\hat{l} < l_{\hat{s}}$) ;
- the period $\hat{p} \in \mathcal{P}$ at which the manager is notified that employee $\hat{e}$ will be late ($\hat{p} \leq b_{\hat{s}}$).

This disruption is denoted $\hat{\Delta} = (\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$. When it occurs, the planned schedule becomes infeasible because shift $\hat{s}$ cannot be operated as planned and re-scheduling must be performed to retrieve an updated feasible solution. Thus, the RBPRP consists of finding a new feasible solution to the original planning problem such that all operated shifts up to period $\hat{p}$ are fixed and all other planned shifts can be modified including those on days $\hat{h}+1, \hat{h}+2, \ldots, |\mathcal{H}|$. Note that a planned shift that is ongoing at period $\hat{p}$ but not finished can also be changed after period $\hat{p}$. Hence, the new schedule must be feasible with respect to the constraints of the original planning problem. In addition, two objectives should be sought. First, the new schedule should be similar to the planned schedule in terms of the number of modified shifts. Second, the additional cost incurred by the modifications should be minimized. Given that these two objectives might be contradictory, the RBPRP falls into the class of bi-objective optimization problems and a small set of Pareto-optimal solutions should be produced. Finally, because delay between period $\hat{p}$ and the disruption starting period $b_{\hat{s}}$ may be very small, even null, re-scheduling must often be performed in real time (i.e., some Pareto-optimal solutions must be found in less than a few seconds).

5.2.2 Concepts and terminology

The initial personnel scheduling problem can be modeled as the integer linear program (ILP) proposed by Hassani *et al.* [76] and presented in details in A.1. For ease of exposition and because every anonymous shift selected during planning is subsequently assigned to a temporary employee for the operations, we discard the anonymous shifts from our discussion. Let $x = ((x_e^h)_{\substack{h \in \mathcal{H} \\ e \in \mathcal{E}}}, x^r)^\top$ be the vector of variables, where x_e^h represents a sub-vector of binary

variables of dimension $|\mathcal{S}_e^h|$ such that $x_e^h(s) = 1$ if the proposed shift $s \in \mathcal{S}_e^h$ is assigned to employee e on day h and 0 otherwise. Furthermore, x^r is a sub-vector containing all the remaining integer variables that are necessary to compute the penalties and the labor costs. The variables in x^r are totally dependent on the binary shift variables $x^b = (x_e^h)_{\substack{h\in\mathcal{H}\\ e\in\mathcal{E}}}$ when the sum of the costs and penalties is minimized. Let $\mathcal{X}$ be the set of feasible solutions to ILP and denote by $c(x) = c^\top x$ its objective function. Consequently, ILP writes as : $\min_{x\in\mathcal{X}} c^\top x$.
Because any feasible solution $x \in \mathcal{X}$ describes a feasible work schedule for all employees in $\mathcal{E}$, x is also called a schedule afterwards. For a schedule $x \in \mathcal{X}$, denote by $s_x(e, h)$ the shift assigned to employee $e \in \mathcal{E}$ on day $h \in \mathcal{H}$ in this schedule. By breach of terminology, we say that an employee is assigned on day $h \in \mathcal{H}$ to a nil shift denoted $s_0(h)$, if h is a day off for this employee. In this case, we have $s_x(e, h) = s_0(h)$. For the sake of notational conciseness, we assume that vector x_e^h has an additional component for this nil shift.
Consider an optimal planned schedule $x_0 \in \mathcal{X}$. When a disruption $\hat{\Delta} = (\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$ occurs and the corresponding RBPRP must be solved, at least another solution $x \in \mathcal{X}$ such that $x_{\hat{e}}^{\hat{h}}(\hat{s}) = 0$ (among other restrictions) must be found. Observe that both solutions x_0 and x belong to the same domain $\mathcal{X}$. To find a solution x from a starting solution x_0, we consider the following three types of decisions g^S, g^X and g^O that we call the *generating decisions* :

- $g^S(s_1, s_2)$: Swap the employees for shifts s_1 and s_2 ;
- $g^X(s_1, s_2, q)$: Extend shift s_2 to cover the first, or the last, q periods of shift s_1 which is shortened by the same number of periods ;
- $g^O(s, e, h)$: If $s \neq s_0(h)$, assign shift s on day h to employee e who was assigned to a day off ; otherwise ($s = s_0(h)$), assign a day off on day h to employee e and deleting his/her planned shift on day h.

Note that any such decision may assign an infeasible shift s to an employee e (i.e., $s \notin \mathcal{S}_e$). To reach a feasible schedule, this shift needs to be replaced in a subsequent decision.
In the following, we use a vector $(g_1, g_2, \ldots, g_m)$ to denote a sequence of m generating decisions that are applied in the order g_1 to g_m. This order is important because the decisions are not commutative. For example, consider three shifts s_1, s_2 and s_3 that are assigned to three employees e_1, e_2 and e_3, respectively, on the same day and which can be swapped by them. Swapping first the shifts of e_1 and e_2 before swapping the shifts of e_2 and e_3 results in a solution where e_1 is assigned to s_2, e_2 to s_3, and e_3 to s_1. On the other hand, swapping first the shifts of e_2 and e_3 before swapping those of e_1 and e_2 yields a different solution where e_1 is assigned to s_3, e_2 to s_1 and e_3 to s_2. Because some shifts are replaced by others in each generating decision, we denote by $r(s)$ the shift that is assigned to employee e_s on day h_s after applying at least one decision involving a shift s. With this notation, the sequence of decisions in the previous two examples would be written as $(g^S(s_1, s_2), g^S(r(s_2), s_3))$ and

$(g^S(s_2, s_3), g^S(s_1, r(s_3)))$, respectively.

Applying a sequence of m generating decisions from the initial schedule x_0 can yield a sequence of schedules $x_0, x_1, x_2, \ldots, x_m$ that might not be all feasible. Nevertheless, the following observation highlights the usefulness of the generating decisions.

Observation 1. *Let $x, y \in \mathcal{X}$. Schedule y can always be obtained from schedule x by applying a finite sequence of generating decisions.*

Indeed, for each employee $e \in \mathcal{E}$ and each day $h \in \mathcal{H}$ such that $s_x(e,h) \neq s_y(e,h)$, one can add to the sequence the decisions $g^O(s_0(h), e, h)$, if $s_x(e,h) \neq s_0(h)$, and $g^O(s_y(e,h), e, h)$, if $s_y(e,h) \neq s_0(h)$, which first assigns a day off to employee e on day h and then reassigns employee e to shift $s_y(e,h)$. Note that the resulting sequence proposed is trivial but other sequences using the other two decision types g^S and g^X might exist to move from x to y. In fact, in the proposed heuristic, the decision type g^O is used with parsimony.

Algebraically, the application of a sequence of generating decisions (possibly a single one) can be seen as a transition vector d between a solution x and another solution $y = x + d$, i.e., $d = y - x$. As mentioned above, the obtained solution y might be infeasible. Consequently, we are not interested in all sequences of generating decisions, but only in those that yield a feasible solution. The following definition introduces minimal ones.

Definition 1. *Given a solution $x \in \mathcal{X}$, a sequence of m generating decisions $(g_1, g_2, \ldots, g_m)$, represented by its transition vector d, is called an* elementary decision (from x) *if $x + d \in \mathcal{X}$ and the sequence is minimal in the sense that any of its prefixes $(g_1, g_2, \ldots, g_n)$ with $n < m$ leads to an infeasible solution.*

An elementary decision can also be represented by a transition vector $d = (d^b, d^r)^\top$ with $d^b = (d_e^h)_{\substack{h \in \mathcal{H} \\ e \in \mathcal{E}}}$. If this elementary decision does not change the shift assigned to an employee e on a day h, then all components of the sub-vector d_e^h are equal to 0. Otherwise, this sub-vector contains two non-zero components : the component associated with the initial shift is equal to -1, whereas that associated with the new shift is equal to 1. Note that a transition vector carries less information than the corresponding elementary decision as it does not keep track of the sequence of the generating decisions. In fact, several elementary decisions may yield the same transition vector.

Let $\mathcal{D}(x)$ be the set of elementary decisions from solution x. Each decision $d \in \mathcal{D}(x)$ generates a number of modifications n_d and a cost c_d which are given by :

$$n_d = |Supp^+(d^b)| \quad \text{and} \quad c_d = c^\top d,$$

where $Supp^+(u) = \{j \in J \mid u_j > 0\}$ denotes the index subset of the positive-valued components in a vector $u = (u_j)_{j \in J}$. Consequently, n_d counts one modification each time that

the final assignment of an employee on a day does not correspond to its initial assignment. It is independent on the sequence of generating decisions in the sense that, if the sequence modifies the assignment several times, it is still counted as a single modification. The cost c_d is equal to the cost difference between the solutions x and $x + d$. It can be positive, negative or null.

To illustrate these concepts, let us consider the following example that involves four employees assigned to the shifts s_1, s_2, s_3 and $\hat{s}$ on a given day $\hat{h}$ (see Figure 5.1). We assume that these shifts cover the same job and that the period length is 15 minutes although the figure indicates the time in hours. Furthermore, the working rules defining the feasibility of a shift may be specific to each employee and specify that employee $e_{\hat{s}}$ cannot work more than 8 hours per day and employee e_{s_2} must work exactly 8 hours per day. No specific restrictions are considered for employees e_{s_1} and e_{s_3}.

The following disruption occurs on day $\hat{h}$ and becomes known at period $\hat{p}$ at around 9h45am : the employee $e_{\hat{s}}$ who was scheduled to start the shift $\hat{s}$ at 12pm will be late by 8 periods, i.e., he/she will arrive at 2pm as shown by the red block in Figure 5.1. The following four elementary decisions $d_1, d_2, d_3, d_4 \in \mathcal{D}(x_0)$ can be applied to handle this disruption. The changes made by these decisions are illustrated in green in Figures 5.2 to 5.5.

- $d_1 = g^X(\hat{s}, s_1, 8)$ with $n_{d_1} = 2$: It consists of extending shift s_1 to cover the 8 periods of lateness of employee $e_{\hat{s}}$ (see Figure 5.2).
- $d_2 = (g^X(\hat{s}, s_2, 8), g^X(r(s_2), s_3, 8))$ with $n_{d_2} = 3$: It consists to start shift s_2 8 periods earlier to cover the lateness of employee $e_{\hat{s}}$ and then to cover the last 8 periods of the modified shift $r(s_2)$ by starting shift s_3 earlier (see Figure 5.3). Note that applying only $g^X(\hat{s}, s_2, 8)$ does not produce a feasible solution because employee e_{s_2} must work exactly 8 hours.
- $d_3 = g^S(\hat{s}, s_2)$ with $n_{d_3} = 2$: It consists of swapping the initial shift assignments of employees $e_{\hat{s}}$ and e_{s_2} (see Figure 5.4).
- $d_4 = (g^X(\hat{s}, s_1, 4), g^X(r(\hat{s}), s_2, 4), g^X(r(s_2), s_3, 4)$ with $n_{d_4} = 4$: It consists of extending

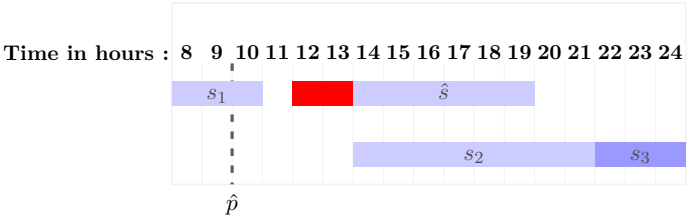

Figure 5.1 Example of a schedule x_0 disrupted on day $\hat{h}$.

shift s_1 to cover the first 4 periods of lateness of employee $e_{\hat{s}}$ before moving forward the start of shift s_2 for 4 periods in order to cover the 4 remaining periods of lateness. Finally, to assign only 8 hours of work to e_{s_2}, the last 4 periods of his/her modified shift $r(s_2)$ are removed from this shift and covered by starting shift s_3 4 periods earlier (see Figure 5.5).

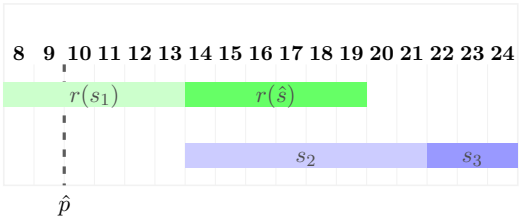

Figure 5.2 Elementary decision d_1

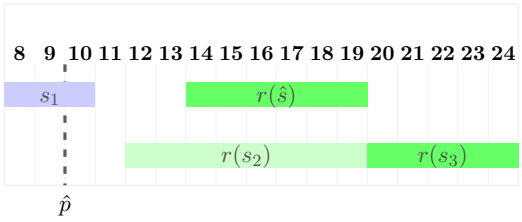

Figure 5.3 Elementary decision d_2

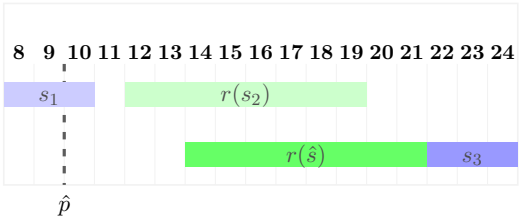

Figure 5.4 Elementary decision d_3

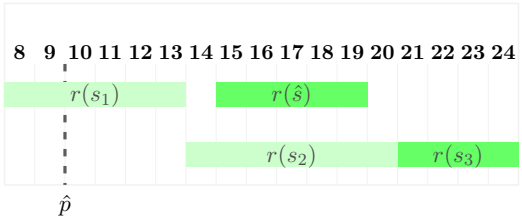

Figure 5.5 Elementary decision d_4

When a disruption occurs during the operations, an elementary decision similar to those presented above and that often impacts only the day of the disruption must be made as quickly as possible to rectify the disruption. If the planned schedule is optimal, the cost of this decision is often strictly positive because it typically yields additional working time for certain employees (unbalancing the working time between the employees) or additional demand over-coverage. To reduce this cost increase, a sequence of elementary decisions (possibly involving shift modifications on multiple days) can be applied. Such a decision sequence must generate a sequence of feasible solutions and aims at achieving a tradeoff between cost and total number of modifications. These decision sequences are called admissible policies.

Definition 2. *Given a schedule $x \in \mathcal{X}$, a sequence of m elementary decisions $(d_1, d_2, \ldots, d_m)$ is called an* admissible policy *for x if*

$$d_i \in \mathcal{D}(x + \sum_{k=1}^{i-1} d_k), \qquad \forall i \in \{1, \ldots, m\}.$$

The set of all admissible policies for a schedule x is denoted $\Pi(x)$.
According to Observation 1, we deduce that, when a disruption $\hat{\Delta} = (\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$ occurs and perturbs an initial schedule x_0, any feasible solution in $\mathcal{X} \setminus \{x_0\}$ can be reached using a policy in $\Pi(x_0)$. However, the manager is not interested by all these solutions but rather by a subset of so-called admissible solutions, denoted $\hat{\mathcal{X}}$. A solution in $\hat{\mathcal{X}}$: i) corrects the disruption $\hat{\Delta}$; ii) does not contain any over-coverage that can be avoided by simply reducing the length of a shift or removing a complete shift ; and iii) respects the planned schedule x_0 up until period $\hat{p}$. The set of all admissible policies for x_0 that conduct to the solutions in $\hat{\mathcal{X}}$ is denoted $\hat{\Pi}(x_0)$. To take these restrictions into account, we also limit the set of elementary decisions to those yielding an admissible solution in $\hat{\mathcal{X}}$.
To evaluate an admissible policy $\delta = (d_1, d_2, \ldots, d_m) \in \hat{\Pi}(x_0)$ composed of m elementary decisions, we introduce the two criteria :

$$\varphi^c(\delta) = \sum_{k=1}^{m} c_{d_k} \qquad \text{and} \qquad \varphi^n(\delta) = |Supp^+(\sum_{k=1}^{m} d_k)|,$$

which compute the total cost and the total number of modifications incurred by policy δ, respectively. Note that $\varphi^n(\delta)$ is not necessarily equal to $\sum_{k=1}^{m} n_{d_k}$ because the shift or day off of an employee on a given day can be modified by more than one elementary decision in δ.
Criteria φ^c and φ^n are inherently conflicting. Indeed, the more we admit modifications, the more we are likely to lower the cost. We introduce the following strict ordering relation between two policies which is denoted by the symbol $\prec$:

$$\delta_1 \prec \delta_2 \Longleftrightarrow \Big(\varphi^c(\delta_1) \leq \varphi^c(\delta_2)\Big) \wedge \Big(\varphi^n(\delta_1) \leq \varphi^n(\delta_2)\Big) \wedge \Big((\varphi^c(\delta_1), \varphi^n(\delta_1)) \neq (\varphi^c(\delta_2), \varphi^n(\delta_2))\Big).$$

Relation $\prec$ is clearly a partial ordering relation, in which case, two policies may not be comparable. This leads us to introduce the concepts of dominated and Pareto-optimal policies.

Definition 3. *Given a schedule $x \in \mathcal{X}$, a policy $\delta \in \hat{\Pi}(x)$ is said to be* dominated *if there exists a policy $\delta' \in \hat{\Pi}(x)$ such that $\delta' \prec \delta$. When there exist no such policies, policy δ is said to be* Pareto-optimal.

The set of Pareto-optimal admissible policies for a schedule x is denoted $\hat{\Pi}^*(x)$. Applying a dominated (resp. Pareto-optimal) policy in $\hat{\Pi}^*(x_0)$ to the planned schedule $x_0 \in \mathcal{X}$ produces a dominated (resp. Pareto-optimal) solution. The set of Pareto-optimal solutions forms the so-called Pareto front, which is denoted $\hat{\mathcal{X}}^*$ and can contain a relatively large number of solutions.
In practice, any solution of $\hat{\mathcal{X}}^*$ with a relatively high cost or number of modifications will never be adopted by the manager. For this reason, the manager can set a search zone parameterized

by two non-negative parameters Φ^c and Φ^n that provide upper bounds on the criteria φ^c and φ^n, respectively. Consequently, for a given planned scheduled $x_0 \in \mathcal{X}$, the RBPRP is equivalent to searching in the set of admissible policies $\hat{\Pi}(x_0, \Phi^c, \Phi^n) = \{\delta \in \hat{\Pi}(x_0) \mid \varphi^c(\delta) \leq \Phi^c \wedge \varphi^n(\delta) \leq \Phi^n\}$, the set of Pareto-optimal policies, denoted $\hat{\Pi}^*(x_0, \Phi^c, \Phi^n)$, that can be applied to produce Pareto-optimal solutions respecting the conditions fixed by the manager. The set of these solutions is denoted $\hat{\mathcal{X}}^*(x_0, \Phi^c, \Phi^n)$.

5.2.3 Problem formulation

The RBPRP consists of finding sequences of elementary decisions forming Pareto-optimal admissible policies in $\hat{\Pi}(x_0, \Phi^c, \Phi^n)$ that describes the Pareto front. It can be formulated on a network $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ is its vertex set and $\mathcal{A}$ its arc set. Set $\mathcal{V}$ contains a source vertex $o = v(x_0)$ representing the initial solution x_0, a sink vertex t, and one vertex $v(x)$ for each solution $x \in \hat{\mathcal{X}}$. In $\mathcal{A}$, there is an arc a between two vertices $v(x_i)$ and $v(x_j)$ if and only if there exists an elementary decision d_a from solution x_i such that $x_i + d_a = x_j$. This arc $a = (v(x_i), v(x_j))$ is associated with a cost $\gamma_a = c_{d_a}$. Furthermore, there is an arc a with $\gamma_a = 0$ between every vertex $v(x)$, $x \in \hat{\mathcal{X}}$, and the sink vertex t. Note that there are no arcs entering the source vertex o because x_0 is infeasible and, therefore, cannot be reached using an elementary decision.

An $o-t$ path $\rho = (o, v(x_1), v(x_2), \ldots, v(x_m), t)$ in network $\mathcal{G}$, also denoted by its arc sequence $\langle a_1, a_2, \ldots, a_{m+1} \rangle$, represents an admissible policy $\delta_\rho = (d_1, d_2, \ldots, d_m)$ in $\hat{\Pi}(x_0)$. Its cost is given by $\varphi^c(\delta_\rho) = \sum_{k=1}^{m+1} \gamma_{a_k}$ and its number of modifications by $\varphi^n(\delta_\rho) = |Supp^+(\sum_{k=1}^{m} d_k)|$. Policy δ_ρ belongs to $\hat{\Pi}(x_0, \Phi^c, \Phi^n)$ if $\varphi^c(\delta_\rho) \leq \Phi^c$ and $\varphi^n(\delta_\rho) \leq \Phi^n$. Furthermore, it belongs to the set of Pareto-optimal admissible policies $\hat{\Pi}^*(x_0, \Phi^c, \Phi^n)$ if there exists no other $o - t$ path ρ' representing an admissible policy $\delta_{\rho'}$ such that $\varphi^c(\delta_{\rho'}) \leq \varphi^c(\delta_\rho)$, $\varphi^n(\delta_{\rho'}) \leq \varphi^n(\delta_\rho)$ and $(\varphi^c(\delta_{\rho'}), \varphi^n(\delta_{\rho'})) \neq (\varphi^c(\delta_\rho), \varphi^n(\delta_\rho))$. Thus, the RBPRP consists of finding Pareto-optimal $o - t$ paths ρ in $\mathcal{G}$ such that $\varphi^c(\delta_\rho) \leq \Phi^c$ and $\varphi^n(\delta_\rho) \leq \Phi^n$. This may be seen as equivalent to solving a shortest path problem with resource constraints (SPPRC) on $\mathcal{G}$ by dynamic programming (see, e.g., Irnich and Desaulniers [77]), where the objective is to minimize path cost under the constraint that the number of modifications should not exceed Φ^n. However, in our case, all paths reaching a vertex $v(x)$ are equivalent because both criterion φ^c and φ^n only depend on the solution x and not the path used to reach it.

Network $\mathcal{G}$ is not acyclic. However, even if there are negative arc costs, there is no need to impose elementarity requirements in this SPPRC because every cycle in $\mathcal{G}$ incurs no cost and no modification and can, thus, be discarded. The main difficulty with this SPPRC formulation is, thus, the size of network $\mathcal{G}$. In practice, it cannot be built a priori as it requires the enumeration of all admissible solutions in $\hat{\mathcal{X}}$. In Section 5.3, we introduce a heuristic that

explores only a part of this network which is built dynamically.

5.2.4 Relationship with the set of admissible solutions

To conclude Section 5.2, we highlight a relationship between network $\mathcal{G}$ and the convex hull of the set of admissible solutions $\hat{\mathcal{X}}$. Recall that a solution $x \in \hat{\mathcal{X}}$ is written as $x = (x^b, x^r)^\top$, where $x^b = (x_e^h)_{\substack{h \in \mathcal{H} \\ e \in \mathcal{E}}}$ is a vector of binary shift variables and x^r is dependent on x^b. Therefore, $\hat{\mathcal{X}}$ is bounded and its convex hull $conv(\hat{\mathcal{X}})$ is a polytope.

Proposition 1. *$x \in \hat{\mathcal{X}}$ if and only if x is an extreme point of $conv(\hat{\mathcal{X}})$.*

proof. *See B.1*

From this proposition, we deduce that, in network $\mathcal{G}$, every vertex in $\mathcal{V} \setminus \{o, t\}$ corresponds to an extreme point of the polytope $conv(\hat{\mathcal{X}})$. The following proposition indicates that there is a bijection between the edges of this polytope and the arcs linking two vertices in $\mathcal{V} \setminus \{o, t\}$.

Proposition 2. *Let $x, y \in \hat{\mathcal{X}}$ be two extreme points of the polytope $conv(\hat{\mathcal{X}})$. These extreme points are adjacent in $conv(\hat{\mathcal{X}})$ if and only if there exists an elementary decision $d \in \mathcal{D}(x)$ such that $y = x + d$.*

proof. *See B.2*

Propositions 1 and 2 show that network $\mathcal{G}$ enables the exploration of $conv(\hat{\mathcal{X}})$ by moving along the edges of this polytope. The arcs exiting vertex o allow to move from the infeasible solution x_0 to admissible solutions in $conv(\hat{\mathcal{X}})$. Similarly as above, it can be proven that these admissible solutions are adjacent to x_0 in the polytope $conv(\mathcal{X})$ because they are obtained using elementary decisions from x_0. Finally, all arcs entering vertex t are used to gather admissible policies yielding different admissible solutions at a common vertex in order to find Pareto-optimal ones.

5.3 Solution algorithm

As mentioned in Section 5.2.3, solving the RBPRP is equivalent to solving a SPPRC defined on $\mathcal{G}$. However, $\mathcal{G}$ is so large in practice that it cannot be built a priori. Consequently, to solve the RBPRP, we propose a heuristic labeling algorithm that explores a relatively small subnetwork of $\mathcal{G}$ that is built dynamically. This heuristic is based on theoretical insights that are presented first.

5.3.1 Theoretical insights

Some results in this section (but not the proposed heuristic) are limited to a subset of the admissible policies in $\hat{\Pi}(x_0, \Phi^c, \Phi^n)$. This subset is denoted $\underline{\hat{\Pi}}(x_0, \Phi^c, \Phi^n)$ and composed of policies $\delta = (d_1, d_2, \ldots, d_m)$ such that each elementary decision d_k, $k = 1, \ldots, m$, is formed only of generating decisions of type g^O and the number of modifications strictly increases from one elementary decision to the next, i.e., $\varphi^n(\delta_i) < \varphi^n(\delta_{i+1})$ for all $i = 1, \ldots, m-1$, where $\delta_i = (d_1, d_2, \ldots, d_i)$ is the policy made up of the first i decisions of δ. As highlighted by the next proposition, focusing on this subset of policies is not restrictive.

Proposition 3. *Let $x_0 \in \mathcal{X}$ and $x \in \hat{\mathcal{X}}^*(x_0, \Phi^c, \Phi^n)$. Solution x can always be obtained from solution x_0 by applying a policy in $\underline{\hat{\Pi}}(x_0, \Phi^c, \Phi^n)$.*

proof. *See B.3.*

Denote by $\underline{\hat{\Pi}}^*(x_0, \Phi^c, \Phi^n) = \underline{\hat{\Pi}}(x_0, \Phi^c, \Phi^n) \cap \hat{\Pi}^*(x_0, \Phi^c, \Phi^n)$ the subset of policies in $\underline{\hat{\Pi}}(x_0, \Phi^c, \Phi^n)$ which yield a Pareto-optimal solution in $\hat{\mathcal{X}}^*(x_0, \Phi^c, \Phi^n)$. Observe that, if there exist two policies $\underline{\delta} \in \underline{\hat{\Pi}}^*(x_0, \Phi^c, \Phi^n)$ and $\delta \in \hat{\Pi}^*(x_0, \Phi^c, \Phi^n) \setminus \underline{\hat{\Pi}}(x_0, \Phi^c, \Phi^n)$ that allow to move from solution x_0 to the same Pareto-optimal solution x, then $\varphi^c(\underline{\delta}) = \varphi^c(\delta)$ and $\varphi^n(\underline{\delta}) = \varphi^n(\delta)$. Consequently, for any policy in $\hat{\Pi}^*(x_0, \Phi^c, \Phi^n)$, there exists one in $\underline{\hat{\Pi}}(x_0, \Phi^c, \Phi^n)$ that yields the same solution. Thus, to find Pareto-optimal solutions, it is sufficient to only look for policies in $\underline{\hat{\Pi}}(x_0, \Phi^c, \Phi^n)$.
Let $\delta_c^*, \delta_n^* \in \underline{\hat{\Pi}}(x_0, \Phi^c, \Phi^n)$ be two policies such that

$$\delta_i^* \in \underset{\delta \in \underline{\hat{\Pi}}(x_0, \Phi^c, \Phi^n)}{\arg\min} \varphi^i(\delta), \qquad i \in \{c, n\}.$$

Thus, δ_c^* and δ_n^* are policies that minimize the cost and the number of modifications, respectively. By definition, they both belong to $\underline{\hat{\Pi}}^*(x_0, \Phi^c, \Phi^n)$ and can be used to restrict the search space as shown in Figure 5.6. In this figure, the dots represent Pareto-optimal policies. The dashed rectangle shows the initial search space where such policies can be found. The rectangles in gray and black represent the regions containing the policies dominated by δ_c^* and δ_n^*, respectively. Finding these policies may, thus, reduce significantly the search space. Finally, note that policies δ_c^* and δ_n^* are not necessarily unique. However, in the following, we use δ_c^* and δ_n^* to denote representative policies.
The following proposition shows that policy δ_n^* can easily be found.

Proposition 4. *There exists an elementary decision $d \in \mathcal{D}(x_0)$ such that $\delta_n^* = (d)$.*

proof. *See B.4*

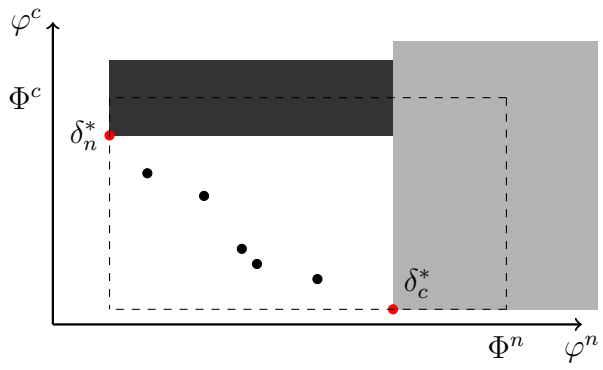

Figure 5.6 Restricted search space induced by policies δ_c^* and δ_n^*.

Finding policy δ_c^* is much more difficult. However, if we find a policy δ such that $\varphi^c(\delta) = 0$, then $\delta = \delta_c^*$ because x_0 is optimal for ILP and, thus, $\varphi^c(\delta_c^*) \geq 0$. In this case, Φ^n can be replaced by $\varphi^n(\delta_c^*)$ to speed up the algorithm.

For any policy $\delta = (d_1, d_2, \ldots, d_m) \in \underline{\hat{\Pi}}^*(x_0, \Phi^c, \Phi^n)$ (even in $\hat{\Pi}^*(x_0, \Phi^c, \Phi^n)$) with $m \geq 2$, there exists an index $l(\delta) \in \{2, 3, \ldots, m\}$ such that $c_{d_{l(\delta)-1}} \geq 0$ and $c_{d_k} < 0, \forall k \geq l(\delta)$. Otherwise, there would be a contradiction with the fact that δ is Pareto-optimal as removing the last decision would yield a dominating policy. On the other hand, it is well known from linear programming theory that, from any non-optimal extreme point x_1 of a polyhedron, there exists at least one sequence of adjacent extreme points $(x_1, x_2, \ldots, x_m)$ of decreasing costs that reaches an optimal extreme point x_m. In our case, this sequence can be generated using elementary decisions $d_2, d_3, \ldots, d_m$ with costs $c_{d_k} < 0$, $k \in \{2, 3, \ldots, m\}$. A first elementary decision d_1 with a non-negative cost is also required to move from the infeasible solution x_0 to feasible solution x_1. Nevertheless, it is not clear if there exists such a policy $\delta = (d_1, d_2, \ldots, d_m)$ in the restricted set $\underline{\hat{\Pi}}^*(x_0, \Phi^c, \Phi^n)$. Therefore, based on these observations, we conjecture that, to find a set of Pareto-optimal solutions in $\hat{\mathcal{X}}^*(x_0, \Phi^c, \Phi^n)$ that describes the Pareto front, it is sufficient to only consider the policies $(d_1, d_2, \ldots, d_m) \in \hat{\Pi}^*(x_0, \Phi^c, \Phi^n)$ such that $c_{d_k} < 0$, $\forall k \geq 2$. The proposed heuristic restricts its search to such policies.

The existence of an elementary decision in a Pareto-optimal policy can be used to predict the existence of other decisions in the same policy. Indeed, if a decision forces an employee e_1 to work two hours more and an employee e_2 to work two hours less, the balance between these two employees might be restored in a subsequent decision by making e_1 (resp., e_2) work less (resp., more). Such a decision is said to be *compatible* with the first decision made. Let us introduce a measure of compatibility.

For an elementary decision d (resp., policy δ), denote by $\mathcal{E}_d$ (resp., $\mathcal{E}_\delta$) the set of employees

affected by this decision (resp., policy). Let $\delta = (d_1, d_2, \ldots, d_m) \in \hat{\Pi}(x_0, \Phi^c, \Phi^n)$ be an admissible policy that produces the feasible solution $x_\delta \in \hat{\mathcal{X}}(x_0, \Phi^c, \Phi^n)$. The following function $\mu(x_0, \delta, d)$ that returns a value in $[0, 1]$ provides a "measure" of the compatibility of each elementary decision $d \in \mathcal{D}(x_\delta)$ with respect to policy δ :

$$\mu(x_0, \delta, d) = \sqrt[m]{\frac{|\mathcal{E}_\delta \cap \mathcal{E}_d|}{|\mathcal{E}_\delta \cup \mathcal{E}_d|}} \cdot \frac{\min\limits_{d' \in \mathcal{D}(x_\delta)} \varphi^n(\delta \oplus d')}{\varphi^n(\delta \oplus d)} \cdot \prod_{e \in \mathcal{E}_\delta \cap \mathcal{E}_d} \frac{\min\{N_{x_0}(e), N_{x_\delta + d}(e)\}}{\max\{N_{x_0}(e), N_{x_\delta + d}(e)\}}, \tag{5.1}$$

where the symbol $\oplus$ indicates the concatenation of a decision to a policy, and $N_y(e)$ is equal to the number of periods worked by an employee $e \in \mathcal{E}$ in a schedule $y \in \hat{\mathcal{X}}$. For a decision $d \in \mathcal{D}(x_\delta)$, a value $\mu(x_0, \delta, d)$ close to 1 indicates that d is highly compatible with δ; at the opposite, a value $\mu(x_0, \delta, d)$ close to 0 means that d is highly incompatible with δ. Indeed, the first term is equal to 1 if the employees affected by decision d are all involved in policy δ, to 0 if none of them are, and to a value between 0 and 1 otherwise. The second term evaluates the impact of decision d on the criterion φ^n compared to the minimal impact over all possible decisions. A decision yielding a relatively large number of modifications compared to the others is disfavored. Finally, the third term measures the relative change in the total working time of each employee in $\mathcal{E}_\delta \cap \mathcal{E}_d$. The focus is put on these employees instead of all employees in $\mathcal{E}_d$ because the largest changes mostly occur for the employees in the first decisions of a policy. A large relative change (either much more working time or much less) for an employee decreases the value of $\mu(x_0, \delta, d)$. Note that the mth root function applied in the first term is used to weight the first term against the others. Indeed, the more decisions are made, the more employees are likely to be involved in the modifications.

To extend a policy δ in the proposed heuristic, we consider only elementary decisions d that yield a sufficiently large value of function $\mu(x_0, \delta, d)$. In fact, in the following proposition, we show under an additional assumption often holding in practice that extending δ with a decision d such that $\mu(x_0, \delta, d) = 1$ is the best possible choice. This assumption stipulates that every elementary decision of a policy does not modify demand coverage for each job and each period. Indeed, in practice, the manager tries to cover all demands as planned so as to avoid a loss in service quality or a wage increase. We denote by $\hat{\underline{\Pi}}^{=}(x_0, \Phi^c, \Phi^n)$ the subset of policies in $\hat{\underline{\Pi}}(x_0, \Phi^c, \Phi^n)$ that respect this assumption.

Proposition 5. *Let $\delta = (d_1, d_2, \ldots, d_m) \in \hat{\underline{\Pi}}^{=}(x_0, \Phi^c, \Phi^n)$ be an admissible policy from x_0 that generates a solution x_δ. If there exists a decision $d \in \mathcal{D}(x_\delta)$ such that $\mu(x_0, \delta, d) = 1$ and $(\delta \oplus d) \in \hat{\underline{\Pi}}^{=}(x_0, \Phi^c, \Phi^n)$, then policy $\delta \oplus d$ is not dominated by any policy $\delta' = \delta \oplus (d_{m+1}, \ldots, d_{m+q}) \in \hat{\underline{\Pi}}^{=}(x_0, \Phi^c, \Phi^n)$ with $d_{m+1} \neq d$ and $q \geq 1$.*

proof. *See B.5*

5.3.2 Heuristic

In this section, we describe the heuristic that we propose to solve the RBPRP. This heuristic is a labeling algorithm that explores the network $\mathcal{G}$ defined in Section 5.2.3 but only partially. Indeed, to speed up the search, it eliminates admissible elementary decisions based on various rules. These rules are :

1. An elementary decision involves shifts of a same day.
2. An elementary decision contains at most n^g generating decisions ($n^g = 2$ for our tests).
3. Generating decisions of type g^O are only used to initially correct the lateness of employee $\hat{e}$ if no other decision types can.
4. A policy $\delta = (d_1, d_2, \ldots, d_m)$ is such that $t_{d_k} \leq t_{d_{k+1}}$ for all $k = 1, 2, \ldots, m-1$, where $t_d \in \mathcal{P}$ is the earliest starting period b_s of a shift s involved in decision d.
5. A policy δ yielding solution x_δ cannot be extended with an elementary decision $d \in \mathcal{D}(x_\delta)$ if $\mu(x_0, \delta, d) \geq \frac{\varphi^n(\delta \oplus d)}{\Phi^n} - \epsilon$, where ϵ is a small positive tolerance.
6. A policy δ containing at least one elementary decision and yielding a solution x_δ can only be extended with an elementary decision $d \in \mathcal{D}(x_\delta)$ such that $c_d < 0$.

Rules 5 and 6 are derived from the discussions made in Section 5.3.1. The heuristic may also reduce the value of Φ^n and the total number of policies explored when it finds a policy δ_c^* such that $\varphi^c(\delta_c^*) = 0$.

In the proposed re-scheduling heuristic, we use the concept of a label ℓ to store a policy $\delta(\ell)$ and its corresponding solution $x_{\delta(\ell)}$. The generated labels are grouped in sets $\mathcal{L}_k$, $k = 1, 2, \ldots, K$, that contain all labels associated with a policy with k elementary decisions, where K is a sufficiently large integer. The labels representing non-dominated solutions are kept in set Ω^*. By breach of terminology, a label is said to be dominated (resp. non-dominated) when its associated solution is.

We start by describing the heuristic main procedure, before presenting some of the embedded functions. The pseudo-code of this main procedure is given in Algorithm 2. First, in Step 2, it finds through function `findCorrectiveGeneratingDecisions` a set G of generating decisions that correct disruption $\hat{X}$. Because these decisions do not necessarily correspond to elementary decisions, function `findAdjacentSolutions` is called in Step 4 for each decision $d \in D$ to find elementary decisions starting with decision g. These elementary decisions form one-decision policies represented by labels that are stored in $\mathcal{L}_1$ in Step 6. The labels in set $\mathcal{L}_1$ are then transferred to set Ω^*. Next, in the while loop (Steps 9–13), the algorithm extends all generated labels in set $\mathcal{L}_i$, not only the non-dominated ones, because labels are typically associated with different solutions to which different elementary decisions can be further applied. In each iteration i, it first propagates (Step 11) each label $\ell \in \mathcal{L}_i$ using function `propagate`

to possibly create multiple labels stored in set $\mathcal{L}_{i+1}$ and representing policies that extends policy $\delta(\ell)$ with a single elementary decision in $\mathcal{D}(x_{\delta(\ell)})$. This loop stops when no more labels can be extended. Finally, a function `applyDominance` is to find the non-dominated labels in set Ω^* (Step 14).

Now, let us detail the three functions `findCorrectiveGeneratingDecisions`, `findAdjacent-Solutions`, and `propagate`. The pseudo-code of the first function is given in Algorithm 3. This function produces a set of generating decisions G that correct disruption $\hat{X}$. For each employee $e \in \mathcal{E} \setminus \{\hat{e}\}$ qualified for job $w_{\hat{s}}$, it finds the generating decisions that modify its current assignment and can correct the lateness of employee $\hat{e}$. If e is assigned to a shift s on day $\hat{h}$ (Step 4), it verifies if shift s can be switched with shift $\hat{s}$ (Step 5) or extended to cover the lateness interval $[b_{\hat{s}}, b_{\hat{s}} + \hat{l}]$ (Step 7). In both cases, the decision is added to set G only if it is considered as a candidate, i.e., if the resulting shifts respect employee availability and qualifications, as well as the minimum rest time between consecutive shifts. Note that the duration of these shifts might not be valid, yielding an infeasible solution to be corrected with one or several subsequent generating decisions. On the other hand, if employee e is assigned to a day off and no decisions of type g^S or g^X have been generated yet (Step 9), then the algorithm looks for potential decisions of type g^O that would assign employee e to a shift of minimal duration covering the lateness interval. Such a decision is added to set G^O if it respects the conditions to be a candidate stated above as well as the minimum number of days off. Decisions in G^O are only added to set G in Step 14 if no decisions of type g^S or g^X have been found.

Function `findAdjacentSolutions` is called in Step 4 of Algorithm 2 to create elementary decisions starting with a given generating decision. The pseudo-code of this function is presented in Algorithm 4. The elementary decisions are stored in set D. Set S contains sequences of generating decisions currently under investigation. When one of them does not yield a feasible solution, it is extended in various ways and the resulting sequences are added to the set S^+. This function starts with a single sequence, composed of the generating decision g_0, in set S. Then, it enters a repeat loop that checks for every sequence $\sigma \in S$ if the schedule of every employee is feasible (Step 5). This check is performed by function `employeesWithInfeasibleSchedule` which returns the set of employees E with an infeasible schedule. If this set is empty, the sequence σ is added to set D. Otherwise, function `lengthenSequence` is called in Step 9 to create new sequences obtained by appending an additional generating decision g to σ. Such a generating decision must correct the infeasibility of the schedule of at least one employee in E. The repeat loop stops when either all sequences do not need to be extended or the maximum number of generating decisions is reached.

Algorithm 2: Re-scheduling Heuristic

input : initial schedule x_0, minor disruption $\hat{X}$;
output: set of labels Ω^*;

1 $\mathcal{L}_k \leftarrow \emptyset, k = 1, 2, \ldots, K$;
2 $G \leftarrow$ `findCorrectiveGeneratingDecisions`$(x_0, \hat{X})$;
3 **foreach** $g \in G$ **do**
4 $D \leftarrow$ `findAdjacentSolutions`$(x_0, \emptyset, g)$;
5 **foreach** $d \in D$ **do**
6 $\mathcal{L}_1 \leftarrow \mathcal{L}_1 \cup \{(x_d, d)\}$;
7 $\Omega^* \leftarrow \mathcal{L}_1$;
8 $i \leftarrow 1$;
9 **while** $\mathcal{L}_i \neq \emptyset$ **do**
10 **foreach** $\ell \in \mathcal{L}_i$ **do**
11 $\mathcal{L}_{i+1} \leftarrow \mathcal{L}_{i+1} \cup$ `propagate`(x_0, ℓ) ;
12 $\Omega^* \leftarrow \Omega^* \cup \mathcal{L}_{i+1}$;
13 $i \leftarrow i + 1$;
14 $\Omega^* \leftarrow$`applyDominance`(Ω^*) ;

Algorithm 5 describes function `propagate` which is called in Step 11 of Algorithm 2 to extend a label ℓ and create new ones that are stored in set $\mathcal{L}$. It starts by calling function `findCandidateElementaryDecisions` (defined below) that produces a relatively large set D of candidate elementary decisions (Step 2). Then, each elementary decision $d \in D$ must pass three selection tests (Steps 4 and 5) to be retained for label creation. First, it must lead to a policy that does not exceed the maximum number of modified shifts. Second, the cost of decision d must be negative as stipulated in Rule 6. Finally, the μ function value for decision d must be sufficiently large according to Rule 5. When all these tests are met, a new label $(x_{\delta(\ell)\oplus d}, \delta(\ell) \oplus d)$ is created and added to set $\mathcal{L}$ (Step 6). Finally, upon finding a zero-cost policy $\delta(\ell) \oplus d$, the value of Φ^n might be updated in Step 8.

For a label ℓ and its solution x_{δ_ℓ}, function `findCandidateElementaryDecisions` builds a list of elementary decisions D that can potentially improve this solution by re-allocating as much as possible the same amount of working time to an employee as in its initial schedule. It loops over all employees affected by the current policy $\delta(\ell)$ and computes for each employee the difference ΔN in its working time between its initial schedule and its current schedule (Step 3). If this difference is not equal to zero, then all possible generating decisions of type g^S or g^X that can reduce (resp. increase) its working time if ΔN is positive (resp. negative) are enumerated through function `findGeneratingDecisions` in Step 5 and stored in set G. Then, for each decision $g \in G$, function `findAdjacentSolutions` is called to create

Algorithm 3: Function `findCorrectiveGeneratingDecisions`

input : initial schedule x_0 ; minor disruption $\hat{X}$;
output: set of generating decisions G ;

$G \leftarrow \emptyset$, $G^O \leftarrow \emptyset$;
foreach $e \in \mathcal{E} \setminus \{\hat{e}\}$ *such that* $w_{\hat{s}} \in \mathcal{W}_e$ **do**
  $s \leftarrow s_{x_0}(e, \hat{h})$;
  if $s \neq s_0(\hat{h})$ **then**
    if $b_{\hat{s}} + \hat{l} \leq b_s$ *and* $g^S(s, \hat{s})$ *is candidate* **then**
      $G \leftarrow G \cup \{g^S(s, \hat{s})\}$;
    if $[b_{\hat{s}}, b_{\hat{s}} + \hat{l}] \cap [b_s, f_s] = \emptyset$, $w_s = w_{\hat{s}}$, $\hat{p} \leq f_s$ *and* $g^X(\hat{s}, s, \hat{l})$ *is candidate* **then**
      $G \leftarrow G \cup \{g^X(\hat{s}, s, \hat{l})\}$;
  else if $G = \emptyset$ **then**
    foreach $s' \in \mathcal{S}_e^{\hat{h}}$ *of minimal duration such that* $[b_{\hat{s}}, b_{\hat{s}} + \hat{l}] \subseteq [b_{s'}, f_{s'}]$ **do**
      if $g^O(s', e, \hat{h})$ *is candidate* **then**
        $G^O \leftarrow G^O \cup \{g^O(s', e, \hat{h})\}$;
if $G = \emptyset$ **then**
  $G \leftarrow G^O$

elementary decisions starting with decision g.

Because it applies Rules 1 to 6, the proposed re-scheduling heuristic is fast and partially explores network $\mathcal{G}$ which is implicitly constructed simultaneously. It offers no guarantee that it will find a set of Pareto-optimal solutions that will describe the Pareto-front. However, as presented in the next section, our computational experiments showed that it is very efficient at this task.

5.4 Computational experiments

To assess the efficiency of the heuristic described in Algorithm 2 and denoted H in the following, we performed computational experiments on different datasets using sets of disruption scenarios. In these tests, we compared heuristic H with the following two solution algorithms :

- An exact algorithm E that is used as a reference on the quality of the solutions returned by H. It consists of solving by a commercial solver the initial planning model described in A.1, modified as follows. First, all shifts or parts of shifts scheduled prior to period $\hat{p}$ in the initial solution x_0 are fixed. Second, all variables associated with shifts of employee $\hat{e}$ that are in conflict with the late arrival are set to 0. Finally, a constraint is added to ensure that at most Φ^n shifts are changed from the initial

Algorithm 4: Function `findAdjacentSolutions`

input : schedule x_δ, policy δ, generating decision g_0 ;
output: set of elementary decisions D ;

$D \leftarrow \emptyset$, $k \leftarrow 1$, $S \leftarrow \{(g_0)\}$;
repeat
  $S^+ \leftarrow \emptyset$;
  foreach $\sigma \in S$ **do**
    $E \leftarrow$ `employeesWithInfeasibleSchedule`(x_δ, σ) ;
    if $E = \emptyset$ **then**
      $D \leftarrow D \cup \{\sigma\}$;
    else
      $S^+ \leftarrow S^+ \cup$ `lengthenSequence`(x_δ, σ, E) ;
  $S \leftarrow S^+$, $k \leftarrow k + 1$;
until $S = \emptyset$ *or* $k = n^g$;

Algorithm 5: Function `propagate`

input : initial schedule x_0, label ℓ, small tolerance $\epsilon > 0$;
output: set of new labels $\mathcal{L}$;

$\mathcal{L} \leftarrow \emptyset$;
$D \leftarrow$ `findCandidateElementaryDecisions`(x_0, ℓ);
foreach $d \in D$ **do**
  if $\varphi^n(\delta(\ell) \oplus d) \leq \Phi^n$ *and* $c_d < 0$ **then**
    if $\mu(x_0, \delta(\ell), d) \geq \frac{\varphi^n(\delta(\ell) \oplus d)}{\Phi^n} - \epsilon$ **then**
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{(x_{\delta(\ell)\oplus d}, \delta(\ell) \oplus d)\}$;
      if $\varphi^c(\delta(\ell) \oplus d) = 0$ **then**
        $\Phi^n \leftarrow \min\{\Phi^n, \varphi^n(\delta(\ell) \oplus d)\}$;

solution x_0.

- A variant of H, referred to as heuristic R, that replaces the test on the μ function in Step 5 of the propagate function described in Algorithm 5 by a random test which allows the selection of exactly the same number of elementary decisions selected using the test on μ. This heuristic is used to confirm the usefulness of function μ.

All algorithms were implemented in C++. All tests were executed on a Linux machine equipped with an 8-core Intel Core i7 processor clocked at 3.4GHz and 16Gb of RAM. The mixed-integer programming solver Cplex, version 12.6.1.0, was used for algorithm E.

Below, we describe first the instances used for our experiments, including the procedure generating the disruption scenarios. Then, we present and analyze the computational results

Algorithm 6: Function `findCandidateElementaryDecisions`

input : initial schedule x_0, label ℓ ;
output: set of elementary decisions D ;

$D \leftarrow \emptyset$;
foreach $e \in \mathcal{E}_{\delta(\ell)}$ **do**
  $\Delta N \leftarrow N_{x_{\delta(\ell)}}(e) - N_{x_0}(e)$;
  if $\Delta N \neq 0$ **then**
    $G \leftarrow$ `findGeneratingDecisions`$(x_{\delta(\ell)}, e, \Delta N)$;
    foreach $g \in G$ **do**
      $D \leftarrow D \cup$ `findAdjacentSolutions`$(x_{\delta(\ell)}, g)$;

obtained.

5.4.1 Datasets and disruption scenarios

Our experiments were conducted using seven real-world datasets provided by our industrial partner. These datasets provide, for each period $p \in \mathcal{P}$ and job $w \in \mathcal{W}$, the number of employees required for job w in period p and, for each employee $e \in \mathcal{E}$ and day $h \in \mathcal{H}$, a set of potential shifts that can be assigned to employee e on day h. The characteristics of these datasets are given in Table 5.1. For each dataset, an optimal initial schedule was obtained by solving the integer linear program presented in A.1.

For each dataset I_k, $k \in \{1, \ldots, 7\}$, we generated 30 scenarios, where each scenario is formed by a disruption $\hat{X} = (\hat{e}, \hat{h}, \hat{s}, \hat{l}, \hat{p})$ that perturbs the planned schedule. This disruption is generated as follows. First, day $\hat{h}$ is chosen using a uniform discrete distribution over horizon $\mathcal{H}$. Then, we choose employee $\hat{e}$ uniformly in the set of employees who work on day $\hat{h}$. The selection of $\hat{h}$ and $\hat{e}$ yields the shift $\hat{s}$. The lateness duration $\hat{l}$, in number of periods, is drawn from another discrete uniform distribution over the set $\{1, \ldots, \left\lfloor \frac{l_{\hat{s}}}{2} \right\rfloor\}$. Finally, the period $\hat{p}$, when the employer becomes aware of the disruption, is chosen from a discrete uniform distribution over the set $\{b_{\hat{s}} - 8, \ldots, b_{\hat{s}}\}$. Note that a check is made to ensure that a scenario is not repeated.

5.4.2 Computational results

To devise a fast heuristic H, several choices were made to limit the labeling process. Consequently, there is no guarantee that heuristic H can find a feasible solution even if one exists, especially if parameter Φ^n is set to a low value, which might be desirable to avoid changing the schedules of many employees just because one employee is late. We performed a first series of tests to determine how frequently heuristic H fails to find a feasible solution in

Table 5.1 Characteristics of the datasets

Dataset	No. employees	No. jobs	Horizon (in days)
I_1	15	5	7
I_2	25	5	7
I_3	29	6	7
I_4	32	3	7
I_5	47	6	7
I_6	49	7	7
I_7	95	7	7

function of Φ^n. For values of $\Phi^n \in \{2, 3, 4, 5, 6\}$, Table 5.2 reports the percentage of failures obtained for each dataset for the corresponding 30 disruption scenarios. This percentage is computed as $100\frac{(n^E - n^H)}{n^E}$, where n^E (resp. n^H) is the number of scenarios for which a feasible solution, with no more than Φ^n shift changes, was found by algorithm E (resp. H). These results show that heuristic H always succeeds to find feasible solutions except for a small number of scenarios when the number of shift changes is restricted to a maximum of $\Phi^n = 2$.

Table 5.2 Percentage of failures with heuristic H

Dataset	$\Phi^n = 2$	$\Phi^n = 3$	$\Phi^n = 4$	$\Phi^n = 5$	$\Phi^n = 6$
I_1	0	0	0	0	0
I_2	0	0	0	0	0
I_3	0	0	0	0	0
I_4	3.3	0	0	0	0
I_5	3.3	0	0	0	0
I_6	0	0	0	0	0
I_7	0	0	0	0	0
Average	0.9	0	0	0	0

Now, let us analyze the quality of the solutions produced by heuristic H on the scenarios for which it did not fail for any value of Φ^n, i.e., 30 scenarios for all datasets except I_4 to I_5, and 29 scenarios for these two datasets. For each algorithm $i \in \{E, H, R\}$, let $\delta_i^* \in \underset{\delta \in \mathcal{Z}(\Phi^n)}{\operatorname{argmax}} \varphi^n(\delta)$ be the least-cost policy it found in the zone $\mathcal{Z}(\Phi^n) = \{\delta \in \hat{\Pi}(x_0, \Phi^c, \Phi^n) | \varphi^n(\delta) \leq \Phi^n\}$ and denote by $\gamma^i(\Phi^n)$ the cost of this policy. Furthermore, let $\Gamma^i(\Phi^n) = \gamma^i(\Phi^n) - \gamma^E(\Phi^n)$ be the error made by heuristic $i \in \{H, R\}$ in the search area $\mathcal{Z}(\Phi^n)$. For a given dataset I_k, $k = 1, \ldots, 7$, the average (resp. standard deviation and maximum) of this error over the n_H scenarios for which heuristic H did not fail, is denoted $\Gamma^i_{avg}(I_k, \Phi^n)$ (resp. $\Gamma^i_{sd}(I_k, \Phi^n)$ and $\Gamma^i_{max}(I_k, \Phi^n)$) for heuristic $i \in \{H, R\}$. Finally, the overall error average, i.e., over all datasets, for a given heuristic i is denoted $\Gamma^i_{AVG}(\Phi^n)$.

The results achieved by heuristics R and H compared to the exact algorithm E are summarized in Table 5.3 for each value of $\Phi^n \in \{2, 3, 4, 5, 6\}$. These results reveal a high success rate

Table 5.3 Percentage of optimal policies and errors for heuristics R and H

Dataset	$\Phi^n = 2$								$\Phi^n = 3$							
	heuristic R				heuristic H				heuristic R				heuristic H			
	$OPT(\%)$	Γ^R_{avg}	Γ^R_{sd}	Γ^R_{max}	$OPT(\%)$	Γ^H_{avg}	Γ^H_{sd}	Γ^H_{max}	$OPT(\%)$	Γ^R_{avg}	Γ^R_{sd}	Γ^R_{max}	$OPT(\%)$	Γ^H_{avg}	Γ^H_{sd}	Γ^H_{max}
I_1	100	0	0	0	100	0	0	0	100	0	0	0	100	0	0	0
I_2	100	0	0	0	100	0	0	0	100	0	0	0	100	0	0	0
I_3	100	0	0	0	100	0	0	0	100	0	0	0	100	0	0	0
I_4	83.3	0.31	0.72	1.88	83.3	0.31	0.72	1.88	80	0.46	1.01	3.96	100	0	0	0
I_5	93.1	0.27	1.15	5.95	96.5	0.07	0.37	1.98	93.1	0.27	1.15	5.95	96.5	0.07	0.37	1.98
I_6	100	0	0	0	100	0	0	0	10	5.36	1.81	5.95	100	0	0	0
I_7	76.6	6.87	13.43	36.74	96.6	0.33	1.81	9.95	40	9.23	13.47	13.47	96.6	0.04	0.24	1.33
Γ^i_{AVG}		1.21				**0.1**				2.19				**0.02**		
$OPT(\%)$	93.3				**96.6**				74.7				**99.0**			

Dataset	$\Phi^n = 4$								$\Phi^n = 5$							
	heuristic R				heuristic H				heuristic R				heuristic H			
	$OPT(\%)$	Γ^R_{avg}	Γ^R_{sd}	Γ^R_{max}	$OPT(\%)$	Γ^H_{avg}	Γ^H_{sd}	Γ^H_{max}	$OPT(\%)$	Γ^R_{avg}	Γ^R_{sd}	Γ^R_{max}	$OPT(\%)$	Γ^H_{avg}	Γ^H_{sd}	Γ^H_{max}
I_1	100	0	0	0	100	0	0	0	100	0	0	0	100	0	0	0
I_2	100	0	0	0	100	0	0	0	100	0	0	0	100	0	0	0
I_3	3.3	10.22	6.21	26.23	96.6	0.07	0.41	2.23	3.3	9.69	5.43	26.23	96.6	0.07	0.41	2.23
I_4	26.6	2.37	2.22	5.95	96.6	0.06	0.34	1.89	26.6	2.37	2.22	5.95	96.5	0.06	0.34	1.89
I_5	24.1	3.35	3.56	9.91	96.5	0.07	0.37	1.97	24.1	3.35	3.56	9.91	100	0	0	0
I_6	10.0	5.36	1.81	5.95	100	0	0	0	0	32.62	10.40	38.99	100	0	0	0
I_7	10.0	14.24	9.63	37.34	100	0	0	0	6.6	15.39	9.01	34.5	100	0	0	0
Γ^i_{AVG}		5.08				**0.03**				9.06				**0.02**		
$OPT(\%)$	39.1				**98.5**				37.2				**99.0**			

Dataset	$\Phi^n = 6$							
	heuristic R				heuristic H			
	$OPT(\%)$	Γ^R_{avg}	Γ^R_{sd}	Γ^R_{max}	$OPT(\%)$	Γ^H_{avg}	Γ^H_{sd}	Γ^H_{max}
I_1	100	0	0	0	100	0	0	0
I_2	100	0	0	0	100	0	0	0
I_3	3.3	9.69	5.43	26.23	96.6	0.07	0.41	2.23
I_4	23.3	2.51	2.54	7.90	100	0	0	0
I_5	24.1	3.35	3.56	9.91	100	0	0	0
I_6	0	34.01	7.90	38.99	100	0	0	0
I_7	3.3	16.28	39.07	9.18	100	0	0	0
Γ^i_{AVG}		9.41				**0.01**		
$OPT(\%)$	36.3				**99.5**			

for heuristic H, which succeeds to compute a least-cost solution for 96.6% to 99.5% of the scenarios for each tested value of Φ^n. Given that, with $\Phi^n = 6$, heuristic H also computes all the solutions generated with $\Phi^n = 2, 3, 4, 5$, we deduce that it can generate Pareto-optimal solutions for almost all values of $\Phi^n \in \{2, 3, 4, 5, 6\}$. Furthermore, when it cannot find a Pareto-optimal solution for a given Φ^n value, the cost error is very small as highlighted by the Γ^H_{avg} values. Notably, observe that, when $\Phi^n \geq 3$, heuristic H finds a least-cost solution for most scenarios, in particular, for all scenarios except one when $\Phi^n = 6$. We believe that this is due to the fact that any solution $x \in \hat{\mathcal{X}}^*(x_0, \Phi^c, \Phi^n)$ can be reached in network $\mathcal{G}$ using several paths and that larger values of Φ^n yield more chances to find one of them. In contrast, the performance of heuristic R deteriorates as the value of Φ^n increases from 2 to 4 and remains stable for larger values. In fact, the first elementary decisions which are not affected by the random selection often allow to find least-cost solutions for $\Phi^n = 2, 3$. Hence,

the random selection only starts to play an important role for larger Φ^n values. The poor performance of heuristic R clearly shows the efficiency of function μ to reduce the number of elementary decisions explored.

Next, we investigate the impact of the value of parameter Φ^n on the quality of the least-cost solution obtained by heuristic H. To do so, for each dataset, scenario and value of $\Phi^n \in \{3, 4, 5, 6\}$, we define $\mathcal{G}^H(2, \Phi^n) = 100\frac{(\gamma^H(2)-\gamma^H(\Phi^n))}{\gamma^H(2)}$ as the difference (gain) in percentage between the costs of the least-cost policies obtained by heuristic H when allowing Φ^n shift changes compared to only two shift changes. For a given dataset, the average gain over all scenarios is denoted $\mathcal{G}^H_{avg}(2, \Phi^n)$. Furthermore, for each dataset, we compute the number of scenarios $n^H(\Phi^n - 1, \Phi^n)$, $\Phi^n = 3, 4, 5, 6$, for which the cost of the solution obtained with at most Φ^n shift changes is better than that of the solution with at most $\Phi^n - 1$ changes. Table 5.4 reports the average gains and the number of scenarios with an improved cost obtained for all values of $\Phi^n \in \{3, 4, 5, 6\}$ and all datasets I_3 to I_7. It turned out that, for the small datasets I_1 and I_2, no gain is realized for each scenario and each value of $\Phi^n \geq 3$, i.e., the least-cost policy always involves two shift changes. This is not the case for the other datasets where increasing the allowed number of shift changes up to six yields average gains varying between 33% and 100%, with an overall average gain of 75.4%. Note that an average gain of 100% (for datasets I_6 and I_7) means that a zero-cost (i.e., least-cost) solution has been found for all scenarios with a maximum of six shift changes. On the other hand, a gain of less than 100% does not mean that the algorithm does not perform well because least-cost solutions might have positive costs. From the results in Table 5.4, observe that, for datasets I_3, I_4 and I_5, most improvement occurred when switching from three to four allowed shift changes. For the other two datasets (I_6 and I_7), better solutions for several scenarios are found as the value of Φ^n increases until finding a least-cost solution for each scenario. All these results show that additional cost reduction can be achieved in many cases by modifying only a few additional shifts but that, in most cases, a very limited number of shift changes (up to six) is sufficient.

In Table 5.5, we provide, for each dataset, the average and maximum computational time (T_{avg} and T_{max}) over the 30 scenarios required by heuristic H when $\Phi^n = 6$ and by the exact algorithm E when $\Phi^n = 2$ to 6. For heuristic H, we observe an overall average time of 0.79 second and a maximum time of 4.13 seconds. These computational times are more than reasonable compared to the times required by the exact algorithm which takes on average 30 seconds for a given value of Φ^n, with a maximum exceeding two minutes in some cases. Note that, to generate a set of Pareto-optimal solutions, the exact algorithm needs to solve a modified ILP for each value of Φ^n resulting in a total average time of approximately 150 seconds.

Finally, to better understand the fast computational times obtained by heuristic H, we report

Table 5.4 Average gain $\mathcal{G}^H_{avg}(2, \Phi^n)$ in percentage

Dataset	$\mathcal{G}^H_{avg}(2,3)$	$\mathcal{G}^H_{avg}(2,4)$	$\mathcal{G}^H_{avg}(2,5)$	$\mathcal{G}^H_{avg}(2,6)$	$n^H(2,3)$	$n^H(3,4)$	$n^H(4,5)$	$n^H(5,6)$
I_3	0	73.4	73.4	73.4	0	30	0	0
I_4	7.3	67.5	67.5	70.7	7	23	1	1
I_5	0	32.5	33.1	33.1	0	22	1	0
I_6	35.6	35.6	96.9	100	30	0	30	7
I_7	30.2	82.1	90.8	100	19	18	11	17
Average	14.6	58.2	72.3	75.4	11.2	18.2	8.6	5

Table 5.5 Average and maximum computational time for heuristic H and algorithm E

Dataset	Heuristic H		Algorithm E									
			$\Phi_n = 2$		$\Phi_n = 3$		$\Phi_n = 4$		$\Phi_n = 5$		$\Phi_n = 6$	
	$T_{avg}(s)$	$T_{max}(s)$	$T_{avg}(s)$	$T_{max}(s)$	$T_{avg}(s)$	$T_{max}(s)$	$T_{avg}(s)$	$T_{max}(s)$	$T_{avg}(s)$	$T_{max}(s)$	$T_{avg}(s)$	$T_{max}(s)$
I_1	0.03	0.04	12.28	14.04	12.81	14.59	12.16	15.23	12.95	35.84	12.95	19.53
I_2	0.02	0.05	8.27	12.73	9.32	26.36	8.77	10.47	8.55	11.32	8.73	19.29
I_3	1.39	4.13	53.99	59.86	69.73	133.17	70.82	104.15	74.87	117.22	73.3	143.01
I_4	0.42	1.33	10.49	14,85	12.79	18.63	13.74	18.38	14.54	20.85	14.69	20.75
I_5	0.46	1.72	26.33	30.96	31.76	41.38	34.66	44.11	35.95	52.63	35.4	45.5
I_6	1.78	2.51	36.16	38.06	36.57	44.82	36.98	46.46	38.34	51.19	37.98	48,65
I_5	1.44	3.67	29.82	32.84	36.23	47.86	39.13	54.17	40.06	65.96	41.56	70.1
Average	0.79		25.33		29.89		32.86		32.18		32.09	

in Table 5.6 the average numbers of policies that are treated before and after some tests in heuristic H when $\Phi^n = 6$: n^{P0}_{avg} is the average number of policies $\delta(\ell) \oplus d$ considered in the tests of Step 4 of the propagate function (see Algorithm 5), n^{P1}_{avg} is the average number of policies surviving these two tests, and n^{P2}_{avg} is the average number surviving the test in Step 5 and, thus, producing a label. Note that these statistics are provided only for the datasets that require more than two shift modifications to find a least-cost solution for some scenarios. From these results, we compute that, on average, 55% of the generated policies (n^{P0}_{avg}) are rejected by the tests in Step 4, whereas 65% of the remaining ones (n^{P1}_{avg}) do not survive the test in Step 5. These results show that the proposed tests are efficient at significantly reducing the number of policies to consider and, therefore, the computational times. As highlighted by the previous results, this computational effort reduction is not made at the expense of bad-quality solutions.

Table 5.6 Average numbers of policies treated before and after some tests in heuristic H

Dataset	n_{avg}^{P0}	n_{avg}^{P1}	n_{avg}^{P2}
I_3	1558.9	1039.6	58.6
I_4	219.1	89.4	62.5
I_5	572.6	158.8	48.6
I_6	3516.7	1700.7	393.8
I_7	722.2	246.5	63.2

5.5 Conclusion

In this paper, we considered the RBPRP in a retail context where employees can be assigned to a wide variety of shifts. To solve this problem, we developed a fast heuristic that aims at correcting a minor disruption by proposing a set of solutions that achieve a good compromise between the cost and the number of shift modifications. Computational experiments conducted on 210 disruption scenarios generated from real-world datasets showed the effectiveness of this heuristic : it can compute the exact Pareto-optimal solutions in less than one second on average for more than 98% of the test cases.

We believe that the framework of the proposed heuristic is generic and could be used to tackle other optimization problems. In fact, it allows the exploration of the extreme points of a polyhedron and, to adapt it to a different problem, would require to redefine the generating decisions and the μ function. Such an adaptation would constitute an interesting research avenue. Other opportunities include the design of a robust heuristic that would take into account, in a stochastic fashion, the potential future disruptions as well as the integration of this re-scheduling tool in a heuristic for solving the initial planning problem.

Acknowledgements : This work was funded by Kronos Canadian Systems, Prompt, and the Natural Sciences and Engineering Research Council (NSERC) of Canada under grant #RDC 530544-18. This financial support is greatly appreciated. The authors are also grateful to the personnel of Kronos for describing the problem and providing datasets.

CHAPITRE 6 STIMULATION PARALLÈLE DE PERTURBATIONS POUR L'OPTIMISATION GLOBALE D'UN HORAIRE DE PERSONNEL

Dans ce chapitre, nous développons une heuristique qui optimise un horaire de personnel. Cette heuristique est inspirée de celle proposée dans le chapitre 5 sur la ré-optimisation en temps réel d'un horaire de personnel suite à une petite perturbation. Nous rappelons qu'on est toujours placé dans un contexte multi-tâches avec un personnel très hétérogène aux niveaux des disponibilités, des qualifications et des durées de travail autorisées par jour. Notre contexte se caractérise aussi avec une demande qui peut fluctuer à n'importe quel moment de l'horizon étudié. Cela veut dire que nous pouvons avoir des quarts qui peuvent débuter et s'achever à différents moments. D'une part, cela offre une certaine flexibilité pour la construction de l'horaire. D'autre part, cette flexibilité mène à des problèmes plus difficiles à résoudre, surtout avec les approches explicites [3, 19, 30, 31]. Concernant les approches implicites que nous avons trouvées dans la littérature [41–45], la notion de type de quart, utilisée dans ces approches, se perd compte tenu de l'hétérogénéité de personnel dont nous disposons. Il est ainsi très difficile de transposer ces études à notre contexte. Par ailleurs, avec ces approches, une affectation aux employés n'aboutira pas nécessairement à un horaire de bonne qualité car la plupart des caractéristiques de personnel dans notre cas ne sont pas prises en compte lors de la génération des quarts composant la solution. Une adaptation des longueurs de ces quarts et de leur instants de début peut être requise. Ceci nous amène à la résolution d'un problème combinatoire.

Notre démarche consiste alors à considérer une approche intégrée qui génère/affecte les quarts simultanément pour chaque employé spécifique durant la résolution. Pour ce faire, notre heuristique se base sur des mouvements de base, appelés décisions génératrices, permettant de générer de nouvelles solutions réalisables/irréalisables à partir d'une solution donnée. Nous stimulons/corrigeons itérativement un ensemble de perturbations bien ciblées selon certaines mesures probabilistes afin de guider notre méthode à effectuer une recherche efficace. Deux procédures de groupage seront utilisées afin de générer un ensemble de sous-problèmes qui seront traités parallèlement. Sur chaque sous-problème, nous stimulons une perturbation qui nous mène, généralement, à l'espace irréalisable. Ainsi, nous définissons un voisinage sur ce sous-problème de rayon déterminé par une borne sur le nombre de modifications dues à la correction de cette perturbation en appliquant une séquence de décisions génératrices afin de revenir à l'espace réalisable. La notion de compatibilité entre les sous-problèmes (ou les voisinages) nous a permis de combiner de façon optimale les améliorations obtenues en parallèle dans les sous-problèmes compatibles.

Ce chapitre est organisé comme suit. Dans la section 6.1, nous rappelons le problème étudié et sa formulation qui est complétée par quelques nouvelles notations et notions. Dans la section 6.2, nous commençons par introduire la terminologie et les concepts qui nous seront utiles pour décrire notre heuristique. Par la suite, nous décrivons l'heuristique proposée ainsi que les algorithmes sur lesquels elle se base. Dans la section 6.3, nous présentons et analysons les résultats de nos expériences numériques. Enfin, les conclusions seront tirées dans la section 6.4.

6.1 Rappels et notions générales

On s'intéresse au problème de planification de base décrit dans la section 4.2. Il consiste à trouver un horaire de travail optimal (ou quasi-optimal) pour un ensemble d'employés désigné par $\mathcal{E}$ et pour un ensemble de tâches désigné par $\mathcal{W}$, sur un horizon de jours $\mathcal{H}$. L'horizon est discrétisé en périodes de 15 minutes (une discrétisation différente pourrait être utilisée). Soit $\mathcal{P}$ l'ensemble de ces périodes. On note par $\mathcal{P}^h$ la restriction de l'ensemble $\mathcal{P}$ au jour $h \in \mathcal{H}$. Par la suite, on caractérisera un quart s par sa période de début $b_s \in \mathcal{P}$, sa période de fin $f_s \in \mathcal{P}$, sa longueur en périodes $l_s (= f_s - b_s + 1)$. Nous ferons également l'hypothèse que ce quart est associé à une seule tâche $w_s \in \mathcal{W}$. La demande est exprimée en nombre d'employés nécessaires pour chaque tâche à chaque période. Chaque employé $e \in \mathcal{E}$ est qualifié pour un ensemble de tâches $\mathcal{W}_e \subseteq \mathcal{W}$ et disponible pour les périodes $\mathcal{P}_e \subseteq \mathcal{P}$. On note par $\mathcal{P}_e^h$ l'ensemble des périodes de $\mathcal{P}_e$ du jour $h \in \mathcal{H}$. On dispose de grands ensembles de quarts personnalisés générés en avance, noté par $\mathcal{S}_e^h$, pour chaque employé $e \in \mathcal{E}$ durant chaque jour $h \in \mathcal{H}$. Un quart s peut être proposé à un employé e, si : i) $w_s \in \mathcal{W}_e$; ii) $\{b_s, \ldots, f_s\} \subset \mathcal{P}_e$; iii) $\underline{l}_e \leqslant l_s \leqslant \bar{l}_e$, où $\underline{l}_e$ (resp. $\bar{l}_e$) est la durée minimale (resp. maximale) pendant laquelle l'employé e peut continûment travailler durant une journée. La sous-couverture (avoir moins d'employés que la demande à une période donnée) est interdite et la sur-couverture (avoir plus d'employés que la demande à une période donnée) est pénalisée. Dans ces conditions, on risque de ne pas trouver un horaire réalisable s'il n'y a pas assez d'employés disponibles pour couvrir la totalité de la demande. Pour éviter cette irréalisabilité, on autorise l'utilisation 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 effectués pour répondre à la demande. Généralement, on affecte ces quarts à des employés d'un autre département ou à des intérimaires juste avant les opérations. On note par $\mathcal{S}_h^A$ l'ensemble des quarts anonymes proposés durant le jour $h \in \mathcal{H}$. La longueur de chaque quart anonyme doit être comprise entre $\underline{l}^A$ et $\bar{l}^A$.

Une solution réalisable x du problème représente un horaire de travail pour les employés de $\mathcal{E}$ qui répondent à la demande de chaque tâche de $\mathcal{W}$ durant chaque période de $\mathcal{P}$. Le coût de l'horaire x est donné par $c(x) = c^\top x$, où c est le vecteur de coût qui comporte

les pénalités de quarts anonymes et de sur-couverture, et les coûts de la main-d'œuvre. Ces coûts ne sont pas les coûts réels payés par l'entreprise mais ils correspondent à une structure pénalisant les situations que nous ne souhaitons pas retrouver dans l'horaire. Ils assurent également une équité entre les employés au niveau du nombre d'heures travaillées dans l'horizon. En effet, on dispose d'une fonction strictement croissante sur les paliers de rémunérations $\mathcal{K} = \{1, 2, \ldots, |\mathcal{K}|\}$. Chaque palier est constitué de 8 heures de travail (une durée différente pourrait être utilisée). Par exemple, si un employé e travaille 20 heures, alors les premières 8 heures seront payées à hauteur de c_0 par période, les deuxièmes 8 heures à hauteur de τc_0 par période (où τ est le taux de croissance) et enfin les dernières 4 heures seront payées à hauteur de $\tau^2 c_0$ par période. On dit alors que la rémunération de e atteint le palier 3. Généralement, la rémunération dans un palier $k \in \mathcal{K}$ est donnée par $\tau^{k-1} c_0$ par période.

Soit $\mathcal{X}$ l'ensemble d'horaires réalisables de ce problème. Pour un horaire réalisable $x \in \mathcal{X}$, on note $\mathcal{S}_x^A$, l'ensemble des quarts anonymes présents dans l'horaire x. Ainsi $\mathcal{S}_x(e, h)$ représente le quart réservé pour l'employé $e \in \mathcal{E}$ au jour $h \in \mathcal{H}$. Par abus de langage, on dit qu'un employé est affecté à un quart nul, noté $s_0(h)$, si cet employé est en repos le jour $h \in \mathcal{H}$. Dans ce cas, on a $\mathcal{S}_x(e, h) = s_0(h)$. On considère les deux opérateurs suivants :

- $\mathcal{D}(e, [d_1, d_2])$ vaut 1 si l'employé $e \in \mathcal{E}$ est disponible entre les périodes d_1 et d_2 et peut travailler pendant la durée $d_2 - d_1 + 1$, 0 sinon ;
- $\mathcal{Q}(e, w)$ vaut 1 si l'employé $e \in \mathcal{E}$ est qualifié pour la tâche $w \in \mathcal{W}$, 0 sinon.

Ainsi un quart s peut être affecté à l'employé e si $\mathcal{A}(e, s) = \mathcal{D}(e, [b_s, f_s]) \times \mathcal{Q}(e, w_s) = 1$.

6.2 Algorithme de résolution

Dans la première partie de cette section, nous introduisons la terminologie et les concepts qui nous seront utiles pour décrire et formaliser nos démarches. Ensuite, nous décrivons les algorithmes de groupages utilisés avant de représenter les mesures sur lesquelles on se base pour choisir le type de perturbation qu'il faut stimuler, dans chaque sous-problème, afin de générer rapidement une décision élémentaire de coût négatif. Finalement, nous détaillons l'heuristique proposée.

6.2.1 Concepts et terminologie

Afin de se déplacer dans l'espace de solutions $\mathcal{X}$, on introduit les types de décisions g^P, g^M, g^{PA} et g^{MA}, appelés décisions génératrices, suivants :

- $g^P(h, e_1, e_2)$: permute l'affectation des quarts pour les employés e_1 et e_2 durant le jour h ;
- $g^M(h, e_1, e_2, l)$: allonge le quart de e_1 de façon à couvrir les l premières ou dernières

périodes du quart de e_2 durant le jour h ;

- $g^{PA}(h, e, s)$: affecte le quart anonyme s à l'employé e durant le jour h, et rend le quart réservé initialement à ce dernier, si différent de $s_0(h)$, à un quart anonyme ;
- $g^{MA}(h, e, s, l)$: consiste à allonger le quart de e de façon à couvrir les l premières ou dernières périodes du quart anonyme s durant le jour h.

L'application d'une décision génératrice peut conduire à une solution non réalisable (par exemple pour $g^{PA}(h, e, s)$ si $\mathcal{A}(e, s) = 0$). Dans ce cas, cela impliquera la nécessité d'appliquer une séquence de décisions génératrices $(g_1, g_2, \ldots, g_m)$ dans l'ordre de g_1 à g_m afin d'obtenir une solution réalisable. Lorsque la séquence $(g_1, g_2, \ldots, g_m)$ est minimale, dans le sens où chaque sous-séquence $(g_1, g_2, \ldots, g_n)$ avec $n < m$ conduit à une solution non réalisable, on dit que la séquence $(g_1, g_2, \ldots, g_m)$ représente une décision élémentaire. Algébriquement, l'application d'une décision élémentaire $(g_1, g_2, \ldots, g_m)$ peut être vue comme un vecteur de transition d entre une solution réalisable $x \in \mathcal{X}$ et une autre solution réalisable $y \in \mathcal{X}$, i.e., $d = y - x$. On note par $\mathcal{D}(x)$ l'ensemble des décisions élémentaires admissibles à partir de la solution $x \in \mathcal{X}$. Chaque décision $d \in \mathcal{D}(x)$ génère un nombre de modifications n_d et engendre un coût $c_d = c^\top d$. À noter que l'on appelle modification, tout changement apporté à l'horaire d'un employé ou à un quart anonyme après la prise d'une séquence de décisions. Lorsque cette séquence apporte plusieurs changements à l'horaire d'un même employé ou au même quart anonyme pendant le même jour, ces changements seront comptabilisés par une seule modification.

Par la suite, on appellera δ une politique admissible de x_0, toute séquence de décisions élémentaires. Si $\delta = (d_1, d_2, \ldots, d_m)$, on dit que δ est une politique de taille m. L'application de cette politique génère une suite de solutions réalisables $\{x_i = x_{i-1} + d_i\}_{i=1}^m$. Dans ce cas, nous caractérisons la politique δ par le coût $\varphi_c(\delta) = \sum_{i=1}^m c_{d_i}$ et par le nombre de modifications $\varphi_n(\delta)$, (les modifications apportées à l'horaire x_0 afin d'obtenir la solution x_m). Soit $\Pi(x)$ l'ensemble des politiques admissibles à partir de $x \in \mathcal{X}$. On note par $\Pi(x, \Phi^c, \Phi^n) = \{\delta \in \Pi(x) | \varphi^c(\delta) < \Phi^c \text{ et } \varphi^n(\delta) \leq \Phi^n\}$ l'ensemble des politiques telles que leurs coûts ne dépassent pas Φ^c et leur nombre de modifications engendrées est inférieur à Φ^n. Soit $\mathcal{E}_\delta$ (resp. $\mathcal{S}_\delta$) l'ensemble des employés (resp. des quarts anonymes) impliqués dans la politique δ.

Soit $x \in \mathcal{X}$ une solution réalisable, soit $n_x^A(h)$ le nombre de quarts anonymes du jour $h \in \mathcal{H}$ et soit $\mathcal{E}_x(k) \subset \mathcal{E}$ l'ensemble d'employés tels que leur rémunération atteint le palier $k \in \mathcal{K}$. On note par $n_x(k)$ le cardinal de cet ensemble. Pour chaque employé $e \in \mathcal{E}$, on note par $p_x(e)$ (resp. $r_x(e)$) le nombre total de périodes programmées pour l'employé e (resp. le nombre de jours de repos) dans l'horaire x. Avec ces notations, on définit les vecteurs $N_x^A = (n_x^A(h))_{h \in \mathcal{H}}$, $N_x = (n_x(k))_{k \in \mathcal{K}}$, $P_x(\mathcal{E}') = (p_x(e))_{e \in \mathcal{E}'}$ et $R_x(\mathcal{E}') = (r_x(e))_{e \in \mathcal{E}'}$ où $\mathcal{E}' \subset \mathcal{E}$.

Pour chaque jour $h \in \mathcal{H}$, on définit la fonction $dist_h$ qui va nous aider à "mesurer" la flexibilité

des modifications, qu'on peut éventuellement effectuer, entre les horaires de deux employés :

$$dist_h : \mathcal{E} \times \mathcal{E} \longrightarrow [0,1]$$
$$(e_1, e_2) \longmapsto \left(1 - \frac{|\mathcal{P}^h_{e_1} \cap \mathcal{P}^h_{e_2}|}{|\mathcal{P}^h_{e_1} \cup \mathcal{P}^h_{e_2}|}\right)^{|\mathcal{W}_{e_1} \cap \mathcal{W}_{e_2}|}.$$

Le terme $1 - \frac{|\mathcal{P}^h_{e_1} \cap \mathcal{P}^h_{e_2}|}{|\mathcal{P}^h_{e_1} \cup \mathcal{P}^h_{e_2}|}$ mesure le pourcentage de chevauchements entre les sous-ensembles $\mathcal{P}^h_{e_1}$ et $\mathcal{P}^h_{e_2}$ pour les deux employés e_1 et e_2 durant le jour h. On choisit de pondérer ce pourcentage en fonction du nombre de tâches pour lesquelles les deux employés e_1 et e_2 sont qualifiés. De plus, cette mesure ne dépend que des données initiales du problème et non d'une solution en particulier. Par la suite, pour un vecteur donné $q = (q_i)_{1 \leq i \leq n}$ où $q_i \in [a, b]$, on définit $var(q)$:

$$var(q) = \frac{\frac{1}{n}\sum_{i=1}^{n} q_i^2 - (\frac{1}{n}\sum_{i=1}^{n} q_i)^2}{b^2 - a^2}.$$

Plus $var(q)$ est proche de 1 plus la dispersion des valeurs q_i est importante. Par la suite, on se base sur $var(N^A_x)$, $var(N_x)$, $var(P_x)$, $var(R_x)$ et $dist_h$ afin de définir des groupages pour le problème et choisir le type de perturbation qu'il faut stimuler dans chaque sous-problème considéré.

On appelle $(\mathcal{C}_1(x), \mathcal{C}_2(x), \ldots, \mathcal{C}_p(x))$ un groupage de problème à partir de la solution x de p sous-problèmes si $\mathcal{C}_i(x) = (\mathcal{E}_i, \mathcal{S}^A_i(x)), \forall i \in \{1, \ldots, p\}$ tels que :

$$\begin{cases} \bigcup_{i=1}^{p} \mathcal{E}_i & = \mathcal{E} \\ \mathcal{E}_i \bigcap \mathcal{E}_j & = \emptyset, \quad \forall i \neq j \\ \bigcup_{i=1}^{p} \mathcal{S}^A_i(x) & = \mathcal{S}^A(x) \\ \mathcal{S}^A_i(x) \bigcap \mathcal{S}^A_j(x) & = \emptyset, \quad \forall i \neq j. \end{cases}$$

Pour chaque sous-problème $\mathcal{C}(x)$, on note par $\Pi(\mathcal{C}(x), \Phi^c, \Phi^n)$ l'ensemble des politiques de $\Pi(x, \Phi^c, \Phi^n)$ et qui implique seulement les employés et les quarts anonymes présents dans le sous-problème $\mathcal{C}(x)$. Si $\delta \in \Pi(\mathcal{C}(x), \Phi^c, \Phi^n,)$ et $\delta' \in \Pi(\mathcal{C}'(x), \Phi^c, \Phi^n)$, où $\mathcal{C}(x)$ et $\mathcal{C}'(x)$ sont deux sous-problèmes ne provenant pas forcément du même groupage, on dit que les deux politiques sont compatibles si $\mathcal{E}_\delta \cap \mathcal{E}_{\delta'} = \emptyset$ et $\mathcal{S}^A_\delta \cap \mathcal{S}^A_{\delta'} = \emptyset$. Dans ce cas, nous avons $\delta \oplus \delta' \in \Pi(x)$, où le symbole $\oplus$ indique la concaténation des politiques δ et δ'. Par essence, toutes les politiques associées aux sous-problèmes du même groupage sont compatibles.

On considère un sous-problème $\bar{\mathcal{C}}(x) = (\bar{\mathcal{E}}, \bar{\mathcal{S}}^A(x))$. L'application d'une décision génératrice g^X, avec $X \in \{P, M, PA, MA\}$, $\mathcal{E}_{g^X} \subset \bar{\mathcal{E}}$ et $\mathcal{S}^A_{g^X} \subset \bar{\mathcal{S}}^A(x)$, peut être vue comme une stimulation de perturbation dans le sous-problème $\bar{\mathcal{C}}(x)$. Si cette décision génératrice nous conduit à une solution non réalisable, alors cette perturbation nécessite une correction en appliquant

d'autres décisions génératrices afin d'aboutir à une solution réalisable. Sinon, alors cette décision, en soi, représente une décision élémentaire. Lorsqu'une décision génératrice g^X, de type $X \in \{P, M, PA, MA\}$, joue le rôle de stimulant de perturbation, nous utilisons la notation suivante $\widehat{g}^X$. Afin de trouver une politique $\delta \in \Pi(\bar{\mathcal{C}}(x), \Phi^c, \Phi^n)$ formée par m décisions élémentaires, nous avons besoin de stimuler au moins m perturbations. De plus, si le coût de cette politique $\varphi^c(\delta)$ est négatif, alors on dit que la politique δ est améliorante. Par la suite, on note par $\bar{\Pi}(\bar{\mathcal{C}}(x), \Phi^n)$ (resp. $\bar{\Pi}(x, \Phi^n)$) l'ensemble des politiques améliorantes de $\Pi(\bar{\mathcal{C}}(x), \Phi^c, \Phi^n)$ (resp. $\Pi(x, \Phi^c, \Phi^n)$). Soit $\bar{\delta}(x, \Phi^n) = \underset{\delta \in \bar{\Pi}(x, \Phi^n)}{argmin} \varphi^c(\delta)$. Il est facile de voir que la solution $x + \bar{\delta}(x, \infty) \in \mathcal{X}$ est une solution optimale du problème. Notre objectif alors est de trouver ou approximer la politique $\bar{\delta}(x, \infty)$. Pour cela, nous proposons de se servir des politiques de $\bar{\Pi}(\bar{\mathcal{C}}(x), \Phi^n)$, $i \in \{1, 2, \ldots, p\}$, où p est un nombre fini de sous-problèmes considérés. Notre travail consistera à trouver des sous-problèmes de bonne qualité, sur lesquels nous aurons besoin de stimuler et corriger des perturbations $\widehat{g}^X$. Le choix de chaque type de perturbation est basé sur des mesures qui indiquent la probabilité que la perturbation $\widehat{g}^X$ nous conduise à une politique de $\bar{\Pi}(\bar{\mathcal{C}}(x), \Phi^n)$. Les sous-problèmes considérés et leur nombre seront altérés dynamiquement, en fonction de certaines grandeurs (définies par la suite), au cours de la résolution. La notion de compatibilité entre les politiques définie nous permet d'utiliser la programmation parallèle : les sous-problèmes considérés seront traitées simultanément.

6.2.2 Les groupages et la stimulation d'une perturbation

Pour construire les sous-problèmes qui seront utilisés par la suite dans l'heuristique proposée, nous avons besoin de regrouper certains employés avec certains quarts anonymes présents dans un horaire réalisable donné. Dans l'algorithme 7, on commence par le partitionnement de l'ensemble des employés $\mathcal{E}$ (étapes 4-8) de telle sorte que chaque ensemble $\mathcal{E}_i$ contienne le nombre minimum d'employés de $\mathcal{E}_x(k)$ pour chaque palier $k \in \mathcal{K}$. Cela nous permet d'avoir des ensembles $\mathcal{E}_i$ tels que $var(P_x(\mathcal{E}_i))$ est assez significative. En effet, l'ensemble $\mathcal{E}_i$ contient des employés travaillant trop dans l'horaire x, et d'autres travaillant peu. Ainsi, pour chaque quart anonyme $s \in \mathcal{S}^A(x)$ et chaque ensemble $\mathcal{E}_i$, on calcule la mesure $\mu_i(s)$ (étape 13). Cette mesure nous permet de calculer la "probabilité" que ce quart anonyme soit travaillé, totalement ou partiellement, par les employés de $\mathcal{E}_i$. Cette mesure prend en considération les qualifications et les disponibilités des employés en question. A noter que le terme associé aux qualifications, à savoir n_q, est plus pondéré car même si un employé e n'est pas disponible pour travailler pendant l'intégralité du quart anonyme s (i.e., $\mathcal{D}(e, [b_s, f_s]) = 0$), il peut toutefois travailler partiellement pendant ce quart (par exemple à l'aide de la décision g^{MA}). Finalement, le quart anonyme s est affecté au sous-problème de mesure la plus élevée.

Algorithme 7: groupage 1

entrées: x horaire initial ;
p nombre de sous-problèmes ;
sorties : $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_p)$ les p sous-problèmes ;

1 **pour** $i \in \{1, \ldots, p\}$ **faire**
2 $\quad \mathcal{E}_i \leftarrow \emptyset$; $\mathcal{S}_i^A(x) \leftarrow \emptyset$;
3 $i \leftarrow 1$;
4 **pour** *chaque palier* $k \in \mathcal{K}$ **faire**
5 $\quad$ **pour** *chaque employé* $e \in \mathcal{E}_x(k)$ **faire**
6 $\quad\quad \mathcal{E}_i \leftarrow \mathcal{E}_i \cup \{e\}$;
7 $\quad\quad$ **si** $i < p$ **alors** $i \leftarrow i + 1$;
8 $\quad\quad$ **sinon** $i \leftarrow 1$;
9 **pour** *chaque quart anonyme* $s \in \mathcal{S}^A(x)$ **faire**
10 $\quad$ **pour** $i \in \{1, \ldots, p\}$ **faire**
11 $\quad\quad n_q \leftarrow \frac{|\{e \in \mathcal{E}_i | \mathcal{Q}(e, w_s) = 1\}|}{|\mathcal{E}_i|}$;
12 $\quad\quad n_d \leftarrow \frac{|\{e \in \mathcal{E}_i | \mathcal{D}(e, [b_s, f_s]) = 1\}|}{|\mathcal{E}_i|}$;
13 $\quad\quad \mu_i(s) \leftarrow n_d^2.n_q$;
14 $\quad i_0 \leftarrow argmax\{\mu_i(s) | i \in \{1, \ldots, p\}\}$;
15 $\quad \mathcal{S}_{i_0}^A(x) \leftarrow \mathcal{S}_{i_0}^A(x) \cup \{s\}$;
16 **pour** $i \in \{1, \ldots, p\}$ **faire**
17 $\quad \mathcal{C}_i \leftarrow (\mathcal{E}_i, \mathcal{S}_i^A(x))$;

Contrairement à l'algorithme 7, l'algorithme 8 commence par former les ensembles $\mathcal{S}^A(x)$. Ensuite, nous affectons les employés au sous-problème qui contient le plus de quarts anonymes susceptibles d'être travaillés par l'employé en question (étape 12). On note par $\mathcal{C}^E(x, p)$ (resp. $\mathcal{C}^A(x, p)$) l'ensemble des sous-problèmes retournés par l'algorithme 7 (resp. 8).

On considère un sous-problème $\bar{\mathcal{C}}(x) = (\bar{\mathcal{E}}, \bar{\mathcal{S}}^A(x))$ de $\mathcal{C}^E(x, p) \cup \mathcal{C}^A(x, p)$ tel que $x \in \mathcal{X}$ et $p \in \mathbb{N}^*$. Vu qu'on ne s'intéresse qu'aux politiques de $\bar{\Pi}(\bar{\mathcal{C}}(x), \Phi^n)$, nous ne stimulons que les décisions génératrices susceptibles de générer des décisions élémentaires avec un coût "très" négatif. Stimuler plusieurs perturbations de différents types et choisir la meilleure, celle du coût le plus négatif, semble très contraignant en termes de temps. Pour cela, on définit un ordre de recherche à parcourir afin de stimuler, le plus rapidement possible, une perturbation d'un type $X \in \{P, M, PA, MA\}$. L'ordre (PA, MA, P, M) indique que nous essayons de stimuler une perturbation de type g^{PA}. Si cela est impossible, nous essayerons de stimuler une perturbation de type g^{MA}, etc. Afin de trouver un ordre qui maximise la chance de stimuler rapidement une "bonne" perturbation, on définit les mesures μ^P, μ^M, μ^{PA} et μ^{MA},

Algorithme 8: groupage 2

entrées: x horaire initial ;
p nombre de sous-problèmes ;
sorties : $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_p)$ les p sous-problèmes ;

pour $i \in \{1, \ldots, p\}$ **faire**
  $\mathcal{E}_i \leftarrow \emptyset$; $\mathcal{S}_i^A(x) \leftarrow \emptyset$;
$i \leftarrow 1$;
pour *chaque quart anonyme* $s \in \mathcal{S}_x^A$ **faire**
  $\mathcal{S}_i^A(x) \leftarrow \mathcal{S}_i^A(x) \cup \{s\}$;
  si $i < p$ **alors** $i \leftarrow i + 1$;
  sinon $i \leftarrow 1$;
pour *chaque employé* $e \in \mathcal{E}$ **faire**
  pour $i \in \{1, \ldots, p\}$ **faire**
    $\rho_i(e) \leftarrow |\{s \in \mathcal{S}_i^A(x) | \mathcal{A}(e, s) = 1\}|$;
  $i_0 \leftarrow argmax\{\rho_i(s) | i \in \{1, \ldots, p\}\}$;
  $\mathcal{E}_{i_0} \leftarrow \mathcal{E}_{i_0} \cup \{e\}$;
pour $i \in \{1, \ldots, p\}$ **faire**
  $\mathcal{C}_i \leftarrow (\mathcal{E}_i, \mathcal{S}_i^A(x))$;

telles que :

$$\begin{cases} \mu^P(\bar{\mathcal{C}}(x)) &= \left[var(P_x(\bar{\mathcal{E}}))\right]^{1-var(R_x(\bar{\mathcal{E}}))} \\ \mu^M(\bar{\mathcal{C}}(x)) &= \left[var(P_x(\bar{\mathcal{E}}))\right]^{var(R_x(\bar{\mathcal{E}}))} \\ \mu^{PA}(\bar{\mathcal{C}}(x)) &= \sqrt[|\bar{\mathcal{S}}^A(x)|]{\frac{\sum_{e \in \bar{\mathcal{E}}} r_x(e)}{|\mathcal{H}||\bar{\mathcal{E}}|}} \\ \mu^{MA}(\bar{\mathcal{C}}(x)) &= \sqrt[|\bar{\mathcal{S}}^A(x)|]{1 - \frac{\sum_{e \in \bar{\mathcal{E}}} r_x(e)}{|\mathcal{H}||\bar{\mathcal{E}}|}}. \end{cases}$$

La mesure $\mu^X(\bar{\mathcal{C}}(x))$, $X \in \{P, M, PA, MA\}$, permet de calculer la probabilité que la stimulation de la perturbation $\widehat{g}^X$ de type $X \in \{P, M, PA, MA\}$ permette d'améliorer la solution courante en obtenant une décision élémentaire de $\mathcal{D}(x)$ avec un coût négatif. En d'autres mots, $\mu^X(\bar{\mathcal{C}}(x))$ permet de voir si le sous-problème $\bar{\mathcal{C}}(x)$ représente un bon terrain afin de stimuler des perturbations de type $X \in \{P, M, PA, MA\}$ qui permettent de générer, par la suite, des politiques améliorantes.

On note par $\bar{\mathcal{E}}^-$ (resp. $\bar{\mathcal{E}}^+$) l'ensemble des employés de $\bar{\mathcal{E}}$ qui travaillent moins (resp. plus) que la moyenne $\bar{p}_x = \frac{\sum_{e \in \bar{\mathcal{E}}} p_x(e)}{|\bar{\mathcal{E}}|}$. Si $var(P_x(\bar{\mathcal{E}}))$ est élevée (i.e. la variabilité des valeurs $p_x(e)$, $e \in \bar{\mathcal{E}}$, est importante) nous aurons plus de chance de réduire le coût avec g^P et g^M en faisant travailler les employés de $\mathcal{E}^-$ (resp. $\mathcal{E}^+$) plus (resp. moins). Lorsque $var(R_x(\bar{\mathcal{E}}))$ est élevée, nous avons alors des employés qui ont beaucoup de jours de repos et d'autres qui en ont peu.

Dans ce cas, la stimulation de perturbation à l'aide de $\widehat{g}^P$ pourra éventuellement nous conduire à des décisions élémentaires de $\mathcal{D}(x)$ de coût négatif. Si $var(R_x(\bar{\mathcal{E}}))$ est faible, cela impliquera que l'affectation des jours de repos est assez homogène entre les employés de $\bar{\mathcal{E}}$. Ainsi, dans ce cas, $\mu^M(\bar{\mathcal{C}}(x))$ favorise les décisions génératrices g^M. D'autre part, lorsque $|\bar{\mathcal{S}}^A(x)|$, le nombre de quarts anonymes dans le sous-problème $\bar{\mathcal{C}}(x)$ est élevé alors la stimulation d'une perturbation à l'aide de g^{PA} et g^{MA} semble plus intéressante. En effet, par nature, ces deux décisions réduisent le nombre de périodes couvertes par les quarts anonymes, ce qui implique la réduction du coût de la solution courante. Afin de choisir entre g^{PA} et g^{MA}, on se base sur la valeur $\frac{\sum_{e\in\bar{\mathcal{E}}} r_x(e)}{|\mathcal{H}||\bar{\mathcal{E}}|}$ qui représente le taux de présence des jours de repos, pour les employés de $\bar{\mathcal{E}}$, sur l'horizon $\mathcal{H}$. Si cette valeur est élevée alors nous favorisons la décision g^{PA}, dans le but de supprimer un quart anonyme et l'affecter à un employé censé être en repos dans la solution x. Sinon, si $1 - \frac{\sum_{e\in\bar{\mathcal{E}}} r_x(e)}{|\mathcal{H}||\bar{\mathcal{E}}|}$ est élevée, alors on favorise les décisions g^{MA}. En effet, dans ce cas nous disposons de plusieurs quarts, non nuls, affectés aux employés. Ces quarts peuvent être modifiés afin de raccourcir les quarts anonymes $\bar{\mathcal{S}}^A(x)$. Nous n'allons

Algorithme 9: Fonction `trouverUneDécisionElémentaire`

entrées: $\bar{\mathcal{C}}(x) = (\bar{\mathcal{E}}, \bar{\mathcal{S}}^A(x))$ sous-problème ;
$\bar{\varphi}^n$ nombre de modifications maximal ;
sorties : d^- décision élémentaire ;

1 $(X_1, X_2, X_3, X_4) =$ `ordre`$\{PA, MA, P, M\}$;
2 $i \leftarrow 1$;
3 **pour** $i \in \{1, \ldots, 4\}$ **faire**
4 $\quad g \leftarrow$ `stimulerUnePerturbationX`$_i(\bar{\mathcal{C}}, g)$;
5 $\quad$ **si** *`bestAdjacentSolution`*$(\bar{\mathcal{C}}, g, \bar{\varphi}^n)$ *existe* **alors**
6 $\quad\quad d^- =$`bestAdjacentSolution`$(\bar{\mathcal{C}}, g, \bar{\varphi}^n)$;
7 $\quad\quad$ retourner d^- ;

pas comparer ces mesures entre elles afin de trouver le bon ordre à parcourir pour stimuler une perturbation car trivialement l'ordre de parcours est (PA, MA, P, M). En effet, avec la décision g^{PA} (resp. g^{MA}), nous pouvons supprimer (resp. raccourcir) des quarts anonymes de $\mathcal{S}^A(x)$. Avec g^P, on peut faire travailler un employé au lieu d'un autre et finalement, avec g^M, on peut allonger (resp. raccourcir) les quarts des employés de $\bar{\mathcal{E}}^-$ (resp. $\bar{\mathcal{E}}^+$). Si $\mu^X(\bar{\mathcal{C}}(x)) < \mu_0^X$, où $X \in \{P, M, PA, MA\}$ et μ_0^X est un paramètre de $]0, 1[$, alors on saute le type X dans l'ordre classique (PA, MA, P, M) et on le place à la fin au besoin. Par exemple, si $\mu^{PA}(\bar{\mathcal{C}}(x)) < \mu_0^{PA}$, $\mu^{MA}(\bar{\mathcal{C}}(x)) \geq \mu_0^{MA}$, $\mu^P(\bar{\mathcal{C}}(x)) < \mu_0^P$ et $\mu^M(\bar{\mathcal{C}}(x)) \geq \mu_0^M$ alors nous obtenons l'ordre de parcours suivant (MA, M, PA, P).

Dans l'algorithme 9, on commence par trouver cet ordre (X_1, X_2, X_3, X_4) à l'étape 1, où

$X_i \in \{P, M, PA, MA\}$ et $X_i \neq X_j$ si $i \neq j$. Dans l'ordre (X_1, X_2, X_3, X_4), on utilise la fonction `stimulerUnePerturbationX`$_i$ (étape 4) afin de stimuler une perturbation de type X_i. Par la suite, l'algorithme cherche à trouver une décision élémentaire de $\mathcal{D}(\bar{\mathcal{C}}(x))$ en utilisant la fonction `bestAdjacentSolution` (étape 5). Cette dernière retourne la décision élémentaire, si elle existe, de coût le plus négatif avec un nombre de modifications qui ne dépasse pas $\bar{\varphi}^n$ parmi celles trouvées à l'aide de la fonction `findAdjacentSolutions` (algorithme 4 défini dans la section 5.3).

Algorithme 10: Fonction `stimulerUnePerturbationPA`

entrées: $\bar{\mathcal{C}} = (\bar{\mathcal{E}}, \bar{\mathcal{S}}^A(x))$ sous-problème ;
sorties : g^{PA} décision génératrice ;

```
1 pour quart anonyme s ∈ S̄^A(x) faire
2     pour chaque palier k ∈ K faire
3         pour chaque employé e ∈ Ē ∩ E_x(k) faire
4             s̄ ← S_x(e, h_s) ;
5             si l_s̄ > l_s et g^PA(h_s, e, s) est candidate alors
6                 retourner g^PA(h_s, e, s) ;
```

Algorithme 11: Fonction `stimulerUnePerturbationMA`

entrées: $\bar{\mathcal{C}} = (\bar{\mathcal{E}}, \bar{\mathcal{S}}^A(x))$ sous-problème ;
sorties : g^{MA} décision génératrice ;

```
1 pour chaque quart anonyme s ∈ S̄^A(x) faire
2     pour k ∈ K faire
3         pour chaque employé e ∈ Ē ∩ E_x(k) | S_x(e, h_s) ≠ s_0(h_s) faire
4             pour l = l_s à l = 1 faire
5                 si g^MA(h_s, e, s, l) est candidate alors
6                     retourner g^MA(h_s, e, s, l) ;
7                 l ← l − 1 ;
```

Maintenant, nous détaillons ci-dessous les quatre fonctions décrites dans les algorithmes (10,11,12,13).En effet, dans ces algorithmes, on dit qu'une décision génératrice g^X, où $X \in \{P, M, PA, MA\}$, est candidate si la prise de la décision g^X produit un horaire de travail qui respecte : i) les disponibilités/qualifications des employés ; ii) la durée minimale de repos entre deux quarts consécutifs. Dans la fonction `stimulerUnePerturbationP` (resp. `stimulerUnePerturbationM`), soit l'algorithme 12 (resp. 13), on cherche une décision génératrice g^P (resp. g^M) afin de permuter (resp. modifier) les deux quarts planifiés pour deux

employés e_1 et e_2 pour lesquels la différence entre les deux paliers atteints par ces employés est maximale. Dans ce cas, le coût de la décision générée pourrait être significativement négatif. A noter que la distance entre chaque couple d'employés est utilisée dans ces deux algorithmes avec un seuil d'élimination, donné par $dist^0$, afin d'accélérer notre recherche (étape 5). En effet, tant que la distance est importante entre deux employés pour une décision génératrice g^P(ou g^M), la flexibilité est minime entre les horaires de ces deux employés. Il sera alors difficile (voire impossible) de générer une décision élémentaire à partir de cette décision génératrice. La fonction `stimulerUnePerturbationPA` (algorithme 10) permet de trouver une décision génératrice g^{PA} faisant plus travailler un employé dans le but de réduire les longueurs des quarts anonymes. Lorsque l'employé en question ne travaille pas, le quart anonyme en question est éliminé. À l'étape 4, la démarche consiste à impliquer les employés dont la rémunération est la plus basse possible. Dans la fonction `stimulerUnePerturbationMA` (algorithme 11), cette même stratégie est utilisée lors de la recherche d'une décision génératrice g^{MA} qui permet d'éliminer ou raccourcir un quart anonyme $s \in \bar{\mathcal{S}}^A(x)$. L'élimination se fait dans le cas idéal lorsque $l = l_s$ à l'étape 4.

Algorithme 12: Fonction `stimulerUnePerturbationP`

entrées: $\bar{\mathcal{C}} = (\bar{\mathcal{E}}, \bar{\mathcal{S}}^A(x))$ sous-problème ;
sorties : g^P décision génératrice ;

$\bar{k} \leftarrow |\mathcal{K}|$;
tant que $\bar{k} \geq 2$ **faire**
  pour *chaque employé* $e_1 \in \bar{\mathcal{E}} \cap \mathcal{E}_x(\bar{k})$ **faire**
    pour *chaque jour* $h \in \mathcal{H}$ *et chaque palier* $k \in \{1, \ldots, \bar{k}-1\}$ **faire**
      pour *chaque employé* $e_2 \in \bar{\mathcal{E}} \cap \mathcal{E}_x(k) \mid dist_h(e_1, e_2) \leq dist^0$ **faire**
        $s_1 \leftarrow \mathcal{S}_x(e_1, h)$;
        $s_2 \leftarrow \mathcal{S}_x(e_1, h)$;
        si $l_{s_1} > l_{s_2}$ *et* $g^P(h, e_1, e_2)$ *est candidate* **alors**
          retourner $g^P(h, e_1, e_2)$;
  $\bar{k} \leftarrow \bar{k} - 1$;

6.2.3 Heuristique

L'algorithme 16 décrit l'heuristique proposée. Cette heuristique prend comme entrées une solution initiale x_0 et un paramètre p qui représente le nombre initial de sous-problèmes à considérer dans les algorithmes de groupage (7,8). La fonction `parallèleRésolution` décrite dans l'algorithme 14 est la pierre angulaire de l'heuristique. Dans cette fonction, on cherche une politique δ améliorante d'une solution donnée $x \in \mathcal{X}$. Cette politique est une combinaison

Algorithme 13: Fonction `stimulerUnePerturbationM`

entrées: $\bar{\mathcal{C}} = (\bar{\mathcal{E}}, \bar{\mathcal{S}}^A(x))$ sous-problème ;
sorties : g^M décision génératrice ;

```
1  k̄ ← |𝒦| ;
2  tant que k̄ ≥ 1 faire
3      pour chaque employé e1 ∈ Ē ∩ 𝓔x(k̄) faire
4          pour chaque jour h ∈ ℋ et chaque palier k ∈ {1, ..., k̄ − 1} faire
5              pour chaque employé e2 ∈ Ē ∩ 𝓔x(k) | disth(e1, e2) ≤ dist^0 faire
6                  s1 ← 𝓢x(e1, h) ;
7                  s2 ← 𝓢x(e2, h) ;
8                  l̄ = ;
9                  pour l = l_{s2} − 1 à l = 1 faire
10                     si g^M(h, e1, e2, l) est candidate alors
11                         retourner g^M(h, e1, e2, l) ;
12                     l ← l − 1 ;
13     k̄ ← k̄ − 1 ;
```

de plusieurs politiques compatibles, trouvées dans les $2p$ sous-problèmes pris en considération. Par la suite, à l'aide de l'algorithme 7, on trouve le groupage $\mathcal{C}^E$ formé par les p premiers sous-problèmes et à l'aide de l'algorithme 8, on forme les p sous-problèmes de $\mathcal{C}^A$. On note par $(\mathcal{C}_1(x), \mathcal{C}_2(x), \ldots, \mathcal{C}_{2p}(x))$ l'ensemble de tous les sous-problèmes dont nous disposons. À l'étape 4, on réserve $2p$ processeurs.

Sur chaque processeur i, les étapes (6-15) sont exécutées en parallèle. Sur chaque processeur $i \in \{1, \ldots, 2p\}$, on cherche à trouver une politique $\delta_i \in \bar{\Pi}(\mathcal{C}_i(x), \Phi^n)$. Pour former la politique δ_i, on utilise la fonction `trouverUneDécisionElémentaire`$(\mathcal{C}_i(x), \bar{\varphi}^n)$, après l'actualisation de la borne $\bar{\varphi}^n$ à l'étape 10, d'une façon récursive (étape 12). Si δ_i contient au moins une décision élémentaire, on dit que le sous-problème $\mathcal{C}_i(x)$ est actif. À l'étape 16, le modèle (P) est résolu, où w_i est une variable binaire qui vaut 1 si la politique δ_i est sélectionnée. Ce modèle nous retourne la combinaison optimale de politiques compatibles $(\delta_i)_{1 \leqslant i \leqslant 2p}$. Les contraintes (6.2) assurent que la politique adoptée à l'étape 17 est dans $\bar{\Pi}(x)$.

$$(\mathrm{P}) : \min_{w} \quad \sum_{i=1}^{2p} w_i \phi^c(\delta_i) \tag{6.1}$$

$$\text{sujet à :} \quad w_i + w_j \leq 1 \qquad \forall i \neq j | \mathcal{E}_{\delta_i} \cap \mathcal{E}_{\delta_j} \neq \emptyset \vee \mathcal{S}^A_{\delta_i} \cap \mathcal{S}^A_{\delta_j} \neq \emptyset \tag{6.2}$$

$$w_i \in \{0, 1\} \qquad \forall i \in \{1, \ldots, 2p\}. \tag{6.3}$$

Finalement, à l'étape 18, a représente le nombre de sous-problèmes actifs. La politique fi-

Algorithme 14: Fonction `parallèleRésolution`

entrées: x solution courante ;
p nombre de sous-problèmes ;
sorties : δ politique de $\bar{\Pi}(x)$;
a nombre de sous-problèmes actifs ;

1 $\mathcal{C}^E \leftarrow$`groupage1`(x, p) ;
2 $\mathcal{C}^A \leftarrow$`groupage2`(x, p) ;
3 $(\mathcal{C}_1(x), \mathcal{C}_2(x), \ldots, \mathcal{C}_{2p}(x)) \leftarrow \mathcal{C}^E \cup \mathcal{C}^A$;
4 `réserverProcesseurs`$(2p)$;
5 **pour** *chaque processeur* i **faire**
6 $\quad \delta_i = (), a_i = 0$;
7 $\quad drapeau \leftarrow vrai$;
8 $\quad$ **tant que** $\phi^n(\delta_i) < \Phi^n$ *et* $drapeau = vrai$ **faire**
9 $\quad\quad drapeau \leftarrow faux$;
10 $\quad\quad \bar{\varphi}^n \leftarrow \Phi^n - \phi^n(\delta_i)$;
11 $\quad\quad$ **si** *`trouverUneDécisionElémentaire`*$(\mathcal{C}_i(x), \bar{\varphi}^n)$ *existe* **alors**
12 $\quad\quad\quad d^- \leftarrow$`trouverUneDécisionElémentaire`$(\mathcal{C}_i(x), \bar{\varphi}^n)$;
13 $\quad\quad\quad \delta_i \leftarrow \delta_i \oplus (d^-)$;
14 $\quad\quad\quad drapeau \leftarrow vrai$;
15 $\quad$ **si** $\delta_i \neq ()$ **alors** $a_i = 1$;
16 $w = argmin(P)$;
17 $\delta = \bigoplus_{i=1}^{2p} w_i \delta_i$;
18 $a = \sum_{i=1}^{2p} a_i$;

nale trouvée δ nous permettra d'améliorer la solution courante dans un voisinage de rayon déterminé par un nombre de modifications donné par $a \times \Phi^n$.

La fonction `parallèleRésolution` sera appelée récursivement par l'heuristique (algorithme 16, étapes 9-15). A chaque appel nous calculons $t_2 - t_1$ la durée totale prise par la fonction afin de trouver la politique $\delta \in \bar{\Pi}(\bar{x})$ où $\bar{x} \in \mathcal{X}$ est la solution courante. À l'étape 15, nous actualisons le nombre de sous-problèmes $\bar{p}$. En effet, si le nombre de sous-problèmes actifs a est inférieur à 25% du nombre total de sous-problèmes considérés (donné par $2\bar{p}$), alors nous réduisons ce dernier. Car avec un nombre $\bar{p}$ plus petit, nous augmentons la chance de stimuler des perturbations et ainsi de générer des politiques qui peuvent améliorer la solution courante. Soit c_0 le coût minimum qu'il faut réduire par une seconde de recherche. Tant que, $\varphi^c(\delta) \leqslant c^0(t_2 - t_1)$, nous estimons que l'amélioration, à partir de la solution courante, est significative. Autrement, la solution courante est quasi-optimale ou bien sa décomposition ne nous permet pas d'avoir des sous-problèmes pour lesquels nous pouvons facilement stimuler des perturbations. Dans ce deuxième cas de figure, on est obligé, à l'étape 6, de modifier la structure de la solution courante (le coût total de cette modification est c^M) avant de rappeler,

à nouveau, récursivement la fonction `parallèleRésolution`. Ce processus est répété tant que le gain total apporté $-c^-$ (où c^- est le coût total réduit de l'étape 9 à l'étape 15) est au moins équivalent à $(1+\epsilon)c^M$.
La fonction `modifierSolution`, algorithme 15, modifie une solution courante (on note par

Algorithme 15: Fonction `modifierSolution`

entrées: x solution réalisable ;
n^P nombre entier ;
sorties : x la solution réalisable modifiée ;
c^M coût de la modification ;

1 $k \leftarrow |\mathcal{K}|$, $i \leftarrow 0$, $c^0 \leftarrow c^\top x$;
2 **pour** *chaque employé* $e \in \mathcal{E} \cap \mathcal{E}_x(k)$ **faire**
3 $\quad i \leftarrow i+1$;
4 $\quad$ **pour** *chaque jour* $h \in \mathcal{H} \mid \mathcal{S}_x(e,h) \neq s_0(h)$ **faire**
5 $\quad\quad \mathcal{S}^A(x) \leftarrow \mathcal{S}^A(x) \cup \{\mathcal{S}_x(e,h)\}$;
6 $\quad\quad \mathcal{S}_x(e,h) \leftarrow s_0(h)$;
7 $\quad$ **si** $i = |\mathcal{E}_x(k)|$ **alors** $k \leftarrow k-1$;
8 $\quad$ **si** $i = n^P$ **alors**
9 $\quad\quad c^M = c^\top x - c^0$;
10 $\quad\quad$ retourner (x, c^M) ;

c^0 le coût de cette solution) en changeant l'horaire de n^P employés. Nous choisissons les employés qui travaillent le plus dans l'horaire x. Nous transformons, par la suite, tous les quarts affectés à ces employés à des quarts anonymes. Nous aurons alors une nouvelle solution avec au moins n^P employés qui ne travaillent pas, et des nouveaux quarts anonymes. Cela nous permet de changer la structure de la solution initiale et d'avoir accès à de nouvelles décisions génératrices. En effet, notre méthode cherche à affecter ces nouveaux quarts anonymes à des employés et équilibrer les durées totales travaillées par les employés. On note par c^M le coût ajouté dû à ces modifications. La condition de l'étape 17 de l'algorithme 16 qui dépend de c^M, nous assure qu'au cours de la résolution nous n'obtenons pas de cycle.

6.3 Résultats numériques

Afin de tester l'efficacité de l'heuristique (notée H par la suite) décrite par l'algorithme 16, nous avons réalisé des tests expérimentaux sur des instances réelles. Dans ces tests, nous avons adopté trois méthodes de comparaison :

- une méthode exacte E que nous utilisons comme référence sur la qualité des solutions retournées par notre heuristique ;

Algorithme 16: Heuristique

entrées: x_0 solution initiale ;
p nombre de sous-problèmes ;
sorties : x^* solution retenue ;

$drapeau \leftarrow vrai$, $\bar{p} \leftarrow p$, $\bar{x} \leftarrow x_0$, $i \leftarrow 0$, $c^M \leftarrow \infty$;
tant que $drapeau = vrai$ **faire**
  si $i \neq 0$ **alors**
    $\bar{k} \leftarrow \underset{k}{argmax}\{\mathcal{E}_x(k) | \mathcal{E}_x(k) \neq \emptyset\}$;
    $n^P \leftarrow |\mathcal{E}_x(\bar{k})|$;
    $(\bar{x}, c^M) \leftarrow$ `modifierSolution`$(\bar{x}, n^P)$;
    $\bar{p} \leftarrow p$;
  $c^- \leftarrow 0$; $t_1 \leftarrow 0$; $t_2 \leftarrow \infty$;
  tant que $\phi^c(\delta) \leq c^0(t_2 - t_1)$ **faire**
    $t_1 \leftarrow$`clock()` ;
    $(\delta, a) \leftarrow$`parallèleRésolution`$(\bar{x}, \bar{p})$;
    $t_2 \leftarrow$`clock()` ;
    $\bar{x} \leftarrow \bar{x} + \delta$;
    $c^- \leftarrow c^- + \phi^c(\delta)$;
    si $a \leq \frac{\bar{p}}{2}$ *et* $\bar{p} \geq 2$ **alors** $\bar{p} \leftarrow \bar{p} - 1$;
  si $c^M < -c^-$ *ou* $c^M = \infty$ **alors** $x^* \leftarrow \bar{x}$;
  si $-c^- < (1 + \epsilon)c^M$ **alors** $drapeau \leftarrow faux$;
  $i \leftarrow i + 1$;

- deux heuristiques A^2 et A^3 qui représentent des variantes connues dans la pratique de la méthode exacte.

Dans la première sous-section, on décrit les instances utilisées et on définit les méthodes E, A^2 et A^3. Dans la deuxième sous-section, nous analysons les résultats obtenus.
Tous nos algorithmes ont été implémentés en C++. Par ailleurs, tous les tests ont été exécutés sur une machine Linux équipée d'un processeur Intel Core i7 12 cœurs cadencé à 3,4 GHz et d'une RAM de 16 Go. En ce qui concerne le calcul parallèle, nous avons utilisé l'interface OpenMP pour le calcul parallèle sur architecture à mémoire partagée.

6.3.1 Instances et méthodes de comparaison

Nous avons accès à 14 instances réelles, fournies par notre partenaire industriel, dont le nombre d'employés varie entre 17 et 94 et le nombre de tâches varie entre 2 et 10. Sur un horizon d'une semaine, nous disposons d'une demande, exprimée en nombre d'employés, de chaque période de la semaine pour chaque tâche de $\mathcal{W}$. Aussi, nous disposons pour chaque employé $e \in \mathcal{E}$: ses qualifications $\mathcal{W}_e$; ses disponibilités $\mathcal{P}_e$; la durée minimale $\underline{l}_e$ et maximale

Tableau 6.1 Caractéristiques des instances

Instances	*Nombre d'employés*	*Nombre de tâches*	*Horizon (en jours)*
I_1,I_2,I_3	17	2	7
I_4,I_5,I_6	27	2	7
I_7,I_8,I_9	34	2	7
I_{10},I_{11},I_{12}	54	2	7
I_{13},I_{14},I_{15}	34	4	7
I_{16},I_{17},I_{18}	47	4	7
I_{19},I_{20}	54	4	7
I_{21}	94	4	7
I_{22},I_{23}	25	5	7
I_{24},I_{25},I_{26}	50	5	7
I_{27},I_{28},I_{29}	37	7	7
I_{30}	72	7	7
I_{31}	94	8	7
I_{32},I_{33}	50	10	7

$\bar{l}_e$ de travail autorisée par jour ; la durée minimale r_e requise entre deux quarts consécutifs. De plus, nous disposons de la longueur minimale $\underline{l}^A$ et maximale $\bar{l}^A$ pour les quarts anonymes. À partir de ces instances, nous générons de nouvelles instances, en permutant les qualifications ou bien les disponibilités des employés, en ne gardant que les instances qui ont une structure suffisamment différente de celle de l'instance initiale. Les caractéristiques de toutes ces instances sont données dans le tableau 6.1. La première instance sur chaque ligne de ce tableau est l'instance réelle, les autres s'il y en a, sont générées à partir de cette instance réelle.
Les deux algorithmes 17 et 18 seront utilisés ultérieurement par les méthodes de comparaisons E, A^2 et A^3, afin de générer les quarts proposés $\mathcal{S}_e^h$ pour chaque employé $e \in \mathcal{E}$ durant chaque jour $h \in \mathcal{H}$ ainsi que les quarts anonymes proposés $\mathcal{S}_h^A$ pour chaque journée $h \in \mathcal{H}$. Dans ces deux algorithmes, p^+ représente le pas de transition de l'instant de début de quarts (étape 8). Donc, pour n'importe quelle période $p \in \mathcal{P}^h$ et pour un pas p^+ donné, au plus nous avons un seul quart s de $\mathcal{S}_e^h$ (de même pour les quarts de $\mathcal{S}_h^A$) qui débute à une période $b_s \in [p, p+p^+[$. C'est facile de voir que, lorsque $p^+ = 1$, la génération des quarts se fait d'une manière exacte (méthode E). La méthode A^2 (resp. A^3) consiste à utiliser un pas $p^+ = 2$ (resp. $p^+ = 3$). Pour ces trois méthodes, un programme linéaire en nombre entiers (ILP), donné dans la section 4.2, est résolu afin d'affecter les bons quarts aux bons employés et sélectionner un sous-ensemble de quarts anonymes qui vont être utilisés pour couvrir la demande totale.

6.3.2 Résultats et analyse

La résolution du programme (ILP), après la génération des quarts, se fait à l'aide du solveur CPLEX (version 12.9.0.0). Dans le tableau 6.2, on remarque que l'obtention d'une solution exacte (avec un gap d'optimalité de 10^{-4}) est une action très coûteuse en termes de temps,

Algorithme 17: Génération de l'ensemble $\mathcal{S}_e^h$

entrées: e employé ;
h jour ;
p^+ pas de mouvement ;
sorties : $\mathcal{S}_e^h$;

$b_s \leftarrow min(\mathcal{P}_e^h)$, $\mathcal{S}_e^h \leftarrow \emptyset$;
tant que $b_s + \underline{l}_e \leq max(\mathcal{P}_e^h)$ **faire**
  $\bar{l} \leftarrow min(\bar{l}_e, max(\mathcal{P}_e^h) - b_s)$;
  pour $l \in [\underline{l}_e, \bar{l}]$ **faire**
    $f_s \leftarrow b_s + l$;
    pour *chaque tâche* $w_s \in \mathcal{W}_e$ **faire**
      $\mathcal{S}_e^h \leftarrow \mathcal{S}_e^h \cup \{\texttt{quart}(b_s, f_s, w_s)\}$;
  $b_s \leftarrow b_s + p^+$;

Algorithme 18: Génération de l'ensemble $\mathcal{S}_h^A$

entrées: h jour ;
p^+ pas de mouvement ;
sorties : $\mathcal{S}_h^A$;

$b_s \leftarrow min(\mathcal{P}^h)$, $\mathcal{S}_h^A \leftarrow \emptyset$;
tant que $b_s + \underline{l}^A \leq max(\mathcal{P}^h)$ **faire**
  $\bar{l} \leftarrow min(\bar{l}^A, max(\mathcal{P}^h) - b_s)$;
  pour $l \in [\underline{l}^A, \bar{l}]$ **faire**
    $f_s \leftarrow b_s + l$;
    pour *chaque tâche* $w_s \in \mathcal{W}$ **faire**
      $\mathcal{S}_h^A \leftarrow \mathcal{S}_h^A \cup \{\texttt{quart}(b_s, f_s, w_s)\}$;
  $b_s \leftarrow b_s + p^+$;

ce qui était attendu puisque la méthode exacte E prend en considération tous les quarts réalisables pour chaque employé ainsi que les quarts anonymes possibles ($p^+ = 1$). De plus, le solveur essaye de prouver l'optimalité. À noter que pour un gap d'optimalité de 0.01 et 0.05, les durées restent relativement importantes.

Par la suite, on utilise la solution retournée par la méthode exacte E (avec un gap d'optimalité de 10^{-4}) comme une référence pour calculer l'erreur relative commise par les méthodes H, A^2 et A^3. En pratique, une solution avec une erreur relative inférieure à 5%, obtenue dans un laps de temps le plus petit possible, est considérée comme une solution qui satisfait notre partenaire industriel. Cette plage autorisée pour l'erreur est justifiée par le fait que les problèmes de gestion d'horaires de personnel sont sujets à beaucoup d'incertitude. Donc,

durant l'opération, plusieurs ré-optimisations en temps réel auront lieu. Ces ré-optimisations peuvent réduire l'erreur initiale et aboutir à un horaire final de très bonne qualité. L'exigence au niveau de temps, par notre partenaire industriel, est justifiée par la centralisation de la planification pour la plupart des clients de notre partenaire industriel. A titre d'exemple, une grande entreprise X qui dispose de 300 magasins à grande surface en Amérique du nord planifie dans le même endroit les horaires hebdomadaires de ses 300 magasins. Cette entreprise ne va pas réserver 300 machines pour cette action, car une réservation d'une telle quantité à la fois engendre un pic de coût important.

Tableau 6.2 Résultats pour les méthodes : E, A^2 et A^3.

Instance	***Méthode*** A^2		***Méthode*** A^3		***Temps(s) de la méthode*** E		
	Err.(%)	***Temps(s)***	***Err.(%)***	***Temps(s)***	***Gap***= 5%	***Gap***= 1%	***Gap***= 0.01%
I_1	3.35	41.6	8.96	27.5	41.6	119.5	146.2
I_2	3.35	78.6	7.83	23.1	101.1	98.1	125.3
I_3	3.35	49.5	7.83	49.5	146.9	150.6	148.6
I_4	4.23	160.4	11.11	102.1	174.9	186.3	981.1
I_5	4.23	179.9	9.8	216.4	489.9	489	1737.4
I_6	4.23	266.6	10	172.4	401.6	398.4	1112.4
I_7	3.35	71.4	9.67	27.1	118.6	118.9	168.2
I_8	3.45	40.2	9.26	26.3	103	102.4	152.1
I_9	3.32	37.2	9.06	27.1	88.3	87.9	100.2
I_{10}	4.23	1541.6	11.16	890.254	428.4	434.9	7674.1
I_{11}	4.22	1396	10.62	4181.5	464.2	463.1	6425.3
I_{12}	4.23	1453.6	9.8	2223.4	781.5	778.1	9157.6
I_{13}	3.35	247.1	9.7	65.2	286.8	306.2	455.2
I_{14}	3.35	612.8	9	112.1	819.9	819.2	1712.2
I_{15}	3.35	144.4	8.8	1068.5	913.2	917.9	932.8
I_{16}	2.83	648.2	7.7	648.2	228	230.2	5374.8
I_{17}	2.89	1913.84	6.65	385	1641.4	2123.1	10119.2
I_{18}	2.44	10037.4	6.24	855.5	3157.3	10072.9	10083.8
I_{19}	4.1	5031.4	11	4663.1	1442.8	1516.5	10080.6
I_{20}	4.23	2815.5	10.63	10032.9	1390	1408.7	9881.7
I_{21}	2.73	18588.6	7.67	573.8	2992	13608.1	20100.4
I_{22}	5.18	286.2	13.9	165.8	595.8	572.2	601.1
I_{23}	5.18	207	13.9	127.1	444.5	437.7	464.2
I_{24}	5.18	459.2	13.9	303.5	1017.2	1012.9	1025.3
I_{25}	5.18	450.6	13.9	286.1	1095.5	1083.9	1105
I_{26}	5.18	353.6	13.9	160.24	1480.3	1476.3	1542.1
I_{27}	6.89	425	17.24	213.1	2053.6	2026.6	2620.7
I_{28}	6.89	382.3	17.24	382.3	766	787.1	895.9
I_{29}	6.89	977.9	17.24	208.1	977.9	829.6	1067.6
I_{30}	6.81	19696.3	6.81	15113.3	4191.7	6761.6	18873.8
I_{31}	2.71	20229.8	7.59	1730	5100.5	5104.4	20450.8
I_{32}	5.18	1874.9	13.89	1372.1	4381.2	4298.9	5840.4
I_{33}	5.18	1807.3	13.89	1262.3	4339.3	4285.8	4266.1
Moyenne	4.28	2803.2	10.79	1445.3	1292.6	1912.3	4709.8

Pour la méthode A^2, nous avons moins de quarts considérés, ce qui réduit les temps de calcul. Cependant, cette réduction n'est pas assez significative pour la plupart des instances. Nous avons aussi obtenu des erreurs qui ne satisfaisaient pas notre partenaire industriel pour

plus de 30% des instances. Avec la méthode A^3, nous obtenons des erreurs comprises entre 6.2% et 17.2%. Donc, pour aucune instance nous obtenons une erreur acceptable par notre partenaire industriel. En ce qui concerne les temps de calcul, nous obtenons les solutions entre 10 minutes et 251 minutes pour plus de 36% des instances. On peut y conclure que l'augmentation du pas de transition p^+ lors de la génération des quarts réduit les temps de calcul (pas nécessairement au niveau de la satisfaction) mais engendre aussi une erreur importante. Dans notre heuristique, on ne se base pas sur la génération initiale de quarts puisque c'est fait d'une façon dynamique. En effet, durant la résolution, on modifie, à l'aide des décisions génératrices g^P, g^M, g^{PA} et g^{MA}, les quarts présents dans la solution courante afin d'en avoir des nouveaux.

L'heuristique requiert une solution initiale. On choisit comme solution initiale, un horaire quasiment "anonyme", noté x_0, obtenu en minimisant uniquement les pénalités de sur-couverture. Un tel horaire sera caractérisé par la présence importante des quarts anonymes, ainsi qu'une différence significative au niveau de la durée totale travaillée par chaque employé (la valeur $var(P_{x_0}(\mathcal{E}))$ est proche de 1). Les temps nécessaires pour obtenir la solution x_0, en utilisant CPLEX, sont donnés dans le tableau 6.3. On remarque que ces temps sont négligeables par rapport à ceux obtenus dans le tableau 6.2 en minimisant les coûts de la sur-couverture et aussi les pénalités de l'utilisation des quarts anonymes et les coûts de la masse salariale. Même si la solution x_0 a une erreur très significative (entre 45.3% et 106%), elle représente un terrain très riche de décisions élémentaires de coût négatif. Par conséquent, on peut remarquer sur la figure 6.1 (où on donne l'erreur en fonction du temps pour certaines instances), que l'erreur décroît exponentiellement dans la première phase de résolution (avant d'utiliser la fonction `modifierSolution`). Lorsque la méthode commence à prendre plus de temps pour améliorer la solution courante (lorsque les triangles de la figure 6.1 deviennent plus espacés), elle fait appel à la fonction `modifierSolution`. Cet appel est traduit par un pic de l'erreur (due au coût de modification ajouté c^M). Sur la même figure, on peut voir que très rapidement la méthode arrive à passer à une nouvelle solution de bonne qualité par rapport à celle trouvée avant l'appel de la fonction `modifierSolution`.

Dans le tableau 6.3, on peut voir que pour chaque instance, à l'exception de l'instance I_8, nous avons obtenu une erreur acceptable par notre partenaire industriel (inférieure à 5%), dans un laps de temps moyen de 114.3 secondes. Pour plus de 75% des instances, ce laps est inférieur à 2 minutes. Dans la dernière colonne du même tableau, on peut aussi constater que la première solution acceptable, ayant une erreur inférieure à 5%, est trouvée dans un laps de temps compris entre 1.3 et 172.4 secondes (excepté pour l'instance I_8) et avec une moyenne de 55.4 secondes. À noter que pour la plupart des instances, la première solution acceptable (i.e. ayant une erreur inférieure à 5%) a été trouvée avant le premier appel de

la fonction `modifierSolution`. Comme nous l'avons mentionné précédemment, cet appel n'est pas coûteux au niveau du temps. Il nous permet de changer la structure d'une solution courante afin de passer à une autre solution qui représente un nouveau terrain riche de décisions élémentaires de coût négatif. Cette modification s'active lorsque la méthode commence à prendre plus de temps pour générer des décisions élémentaires de coût négatif. Cela nous permet d'accélérer notre heuristique.

Tableau 6.3 Résultats de l'heuristique

Instance	***Solution initiale***		***Méthode H***					***Méthode H^R***	
	Err.(%)	***Temps(s)***	***Err.(%)***	***Temps(s)***		***N. sous-prob.***		***Temps(s)=600***	
				Total	***Err.***$< 5\%$	***Max***	***N. chang.***	***Err.(%)***	***Réd.(%)***
I_1	63.40	12	1.11	19.9	2.8	4	0	1.11	0
I_2	63.81	11.8	0.50	5.4	1.3	4	0	0.33	34
I_3	53.85	12.2	0.93	28.9	1.5	4	0	0.93	0
I_4	71.08	8.7	2.21	49.3	13.9	4	0	1.41	36.2
I_5	59.72	14.4	0.24	26.8	3.7	4	0	0.21	12.5
I_6	71.11	9.3	2.40	54.3	31	4	0	1.79	25.4
I_7	76.16	27.1	4.66	63	19.8	6	0	4.35	6.7
I_8	81.15	11.8	5.51	103.3	∞	6	0	1.98	64.1
I_9	79.2	11.6	4.47	61.2	49.2	6	6	1.67	62.6
I_{10}	74.51	9.7	2.36	87.9	58	10	7	1.76	25.4
I_{11}	74.03	9.4	2.7	81.7	45.2	10	4	2.34	13.3
I_{12}	62.4	18.2	0.7	63.1	24.1	10	7	0.7	0
I_{13}	73.17	30.3	4.36	38.9	24	6	3	4.36	0
I_{14}	58.39	41.5	1.69	19.5	4.3	6	5	1.26	25.4
I_{15}	74.7	28	0.62	39.3	7.3	6	2	0.62	0
I_{16}	53.73	21.9	4.04	163.8	101.8	6	5	4.04	0
I_{17}	38.12	36.1	0.95	70.4	24.9	6	3	0.93	2.1
I_{18}	51.4	32.6	0.88	74.3	47.4	6	0	0.39	55.7
I_{19}	65.77	24.4	4.29	56.4	39.1	10	2	1.97	54.1
I_{20}	65.72	24.3	1.95	114.2	59	10	1	1.95	0
I_{21}	51.2	28.3	2.59	309.3	153	12	1	2.59	0
I_{22}	56.88	64.8	1.27	61.2	8.7	4	0	0.2	84.3
I_{23}	60	53	0.21	0.21	24.2	4	0	0.21	0
I_{24}	59.23	91.5	3.86	345	172.4	10	0	1.63	57.8
I_{25}	58.93	92.6	1.13	212	139.4	10	0	1.13	0
I_{26}	58.57	49.7	2.81	181.4	117.8	10	0	1.31	53.4
I_{27}	62.68	82.1	0.21	53.2	53.2	6	2	0.05	76.2
I_{28}	66.66	83.4	0.04	92.5	37.7	6	3	0.04	0
I_{29}	62 .4	83.6	0.39	44.6	32.8	6	3	0.39	0
I_{30}	71.65	132.4	1.36	117.6	109.5	12	5	1.2	11.8
I_{31}	45.3	118.1	3.25	272.6	111.5	12	7	1.05	67.7
I_{32}	104.96	356.5	0.41	540.7	94.4	10	7	0.41	0
I_{33}	106.14	355.5	0	319.5	161	10	5	0	0
Moyenne	64.05	60.2	1.94	114.3	55.4	7.3	2	1.34	23.3

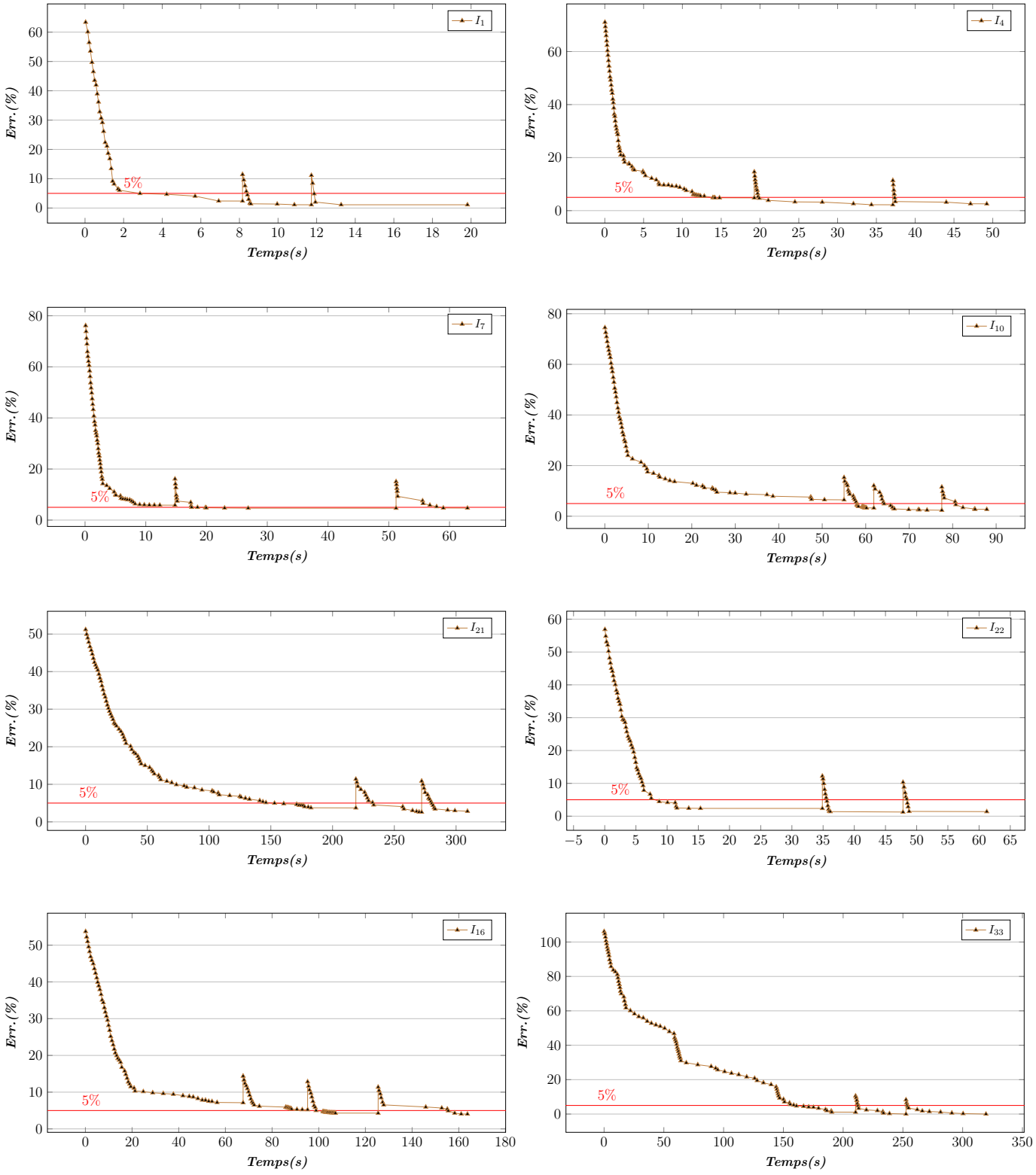

Figure 6.1 Réduction de l'erreur en fonction du temps pour certaines instances

À noter que le nombre de sous-problèmes utilisés n'est pas très grand (la borne supérieure sur ce nombre varie entre 4 et 12 selon le nombre d'employés dans chaque instance), ce nombre s'actualise dynamiquement au cours de la résolution dans le but d'essayer de maximiser le nombre de sous-problèmes actifs parmi ceux considérés. Dans la colonne ***N.chang.*** du tableau 6.3, on donne le nombre de changements apportés au nombre de sous-problèmes considérés. On peut constater que ce nombre est petit (en moyenne, deux changements au cours de toute la résolution). Donc, la plupart des sous-problèmes considérés sont actifs. Cela est dû à la fixation de la borne Φ^n à une valeur relativement petite (pour nos tests, nous avons choisi $\Phi^n = 6$). En effet, avec une telle valeur, nous assurons une très bonne synchronisation entre les tâches exécutées en parallèle. Si, par exemple, $\Phi^n = 20$ et on dispose de 4 sous-problèmes dont un est actif, noté $\mathcal{C}$, alors la méthode continue à stimuler/corriger des perturbations dans le sous-problème actif $\mathcal{C}$ dans le but de générer une politique $\delta \in \bar{\Pi}(\mathcal{C}, \Phi^n)$. Durant tout ce temps, nous ne profitons pas des autres sous-problèmes, ce qui influence négativement les temps de calcul.
Le critère d'arrêt, utilisé par l'heuristique (algorithme 16) à l'étape 16, impose à la méthode d'arrêter le plus rapidement possible à cause de la nature très contraignante, en terme de temps, du problème étudié. Afin de voir la capacité de l'heuristique à améliorer la solution (en réduisant l'erreur) avec plus de temps accordé, nous avons relaxé la méthode en supprimant ce critère et en fixant une durée de 10 minutes pour l'exécution. La méthode qui résulte de cette modification est notée H^R. Sur la dernière colonne du tableau 6.3, on remarque qu'avec l'heuristique H^R, on peut réduire l'erreur pour certaines instances avec un pourcentage moyen de réduction de 23.3%. Ce pourcentage est très significatif pour certaines instances (atteint 84.3% pour I_{22}). Pour les autres instances où ce pourcentage est faible ou bien nul, excepté I_{13} et I_7, nous remarquons que la solution trouvée par l'heuristique H est plutôt de très bonne qualité, ce qui rend l'heuristique H^R incapable de battre une telle qualité.

6.4 Conclusion

Dans ce chapitre, nous avons considéré le problème d'optimisation d'un horaire du personnel dans un contexte où les employés peuvent être assignés à une grande variété de quarts, commençant et se terminant à divers moments. Nous avons proposé une heuristique rapide basée sur la stimulation/correction de certaines perturbations. Le choix du type de ces perturbations est basé sur des mesures qui nous permettent d'estimer la probabilité qu'une correction d'une telle perturbation nous conduise à une solution meilleure. L'utilisation des algorithmes de groupages ainsi que la notion de compatibilité entre les politiques nous a permis d'utiliser la programmation parallèle. Les tests numériques obtenus sur 33 instances (dont 14 sont réelles) ont montré l'efficacité de cette heuristique. Celle-ci arrive à trouver, sur 97% des

instances, des solutions de très bonne qualité en moins de 2 minutes en moyenne. Les tests ont aussi prouvé l'efficacité (resp. la limite) de la génération dynamique (resp. la génération classique) des quarts.

Nous croyons que notre heuristique peut être appliquée dans d'autres contextes. Pour ce faire, deux adaptations sont nécessaires : adapter les décisions génératrices avec le nouveau contexte et adapter les mesures μ^X. On peut également voir qu'une solution trouvée par notre heuristique sera facilement re-optimisable dans le cas de la révélation d'une petite ou grande perturbation ou le cas d'une optimisation interactive. En effet, toute modification, imposée, de la solution trouvée peut être vue comme un appel de la fonction `modifierSolution`. Ainsi les modifications entre les horaires des employés seront flexibles lors de la ré-optimisation, étant donné qu'avec les mêmes types de modifications, la solution à ré-optimiser avait été trouvée.

CHAPITRE 7 DISCUSSION GÉNÉRALE

Cette thèse a donné naissance à deux nouvelles méthodes de ré-optimisation en temps réel pour corriger des événements perturbant un horaire planifié. Ces événements non contrôlables peuvent survenir à tout moment de l'horizon. La correction devait être réalisée sans disposer d'aucune information sur la perturbation avant sa réalisation, tout en proposant une solution efficace dans un temps très court. Le domaine d'application est celui de la vente de détail, un domaine très peu abordé dans la littérature. Nous avons également montré comment une méthode de ré-optimisation efficace et rapide peut être généralisée pour effectuer une optimisation globale.

L'horaire de travail des employés d'une entreprise du domaine de la vente de détail peut être réalisé à l'aide d'un modèle mathématique appelé modèle de base. Le premier article s'intéresse à l'utilité de l'information duale de la relaxation linéaire de ce modèle de base. En effet, une perturbation de l'horaire planifié correspond à une perturbation de la solution de ce modèle au niveau de son membre de droite, et l'information duale est utilisée pour effectuer une ré-optimisation très locale. Cette ré-optimisation est très rapide et fournit des horaires modifiés de très bonne qualité par rapport à l'optimum. Cependant, si plusieurs perturbations surviennent, cette information duale initiale peut perdre en richesse. L'article montre également qu'une simple étude statistique et d'apprentissage permet d'éviter une optimisation exacte, à chaque fois, pour trouver les nouvelles valeurs duales à associer à l'horaire trouvé après la correction de plusieurs perturbations.

Le problème traité dans le deuxième article consiste à chercher de bonnes décisions pour corriger une perturbation en adaptant le reste de l'horizon. Deux objectifs doivent être visés lors de cette ré-optimisation : la minimisation des coûts et le nombre de modifications effectuées à l'horaire initial. Un réseau est utilisé pour atteindre ces deux objectifs contradictoires. La construction d'un tel réseau en temps réel est impossible de par sa taille. Cependant, des résultats théoriques sur le problème nous permettent de réduire très significativement la taille du réseau nécessaire, ce qui rend la construction et la résolution possibles.

Dans la troisième partie de cette thèse, on généralise la méthode de l'article 2 pour réaliser une optimisation globale d'un horaire de personnel. Les défis dans ce travail viennent du besoin de cibler, itérativement, un ensemble de perturbations telles que leurs corrections aboutissent à un nouvel horaire améliorant la solution initiale (avant perturbation). De plus, il a été nécessaire de créer des sous-problèmes afin de traiter le maximum de ces perturbations en parallèle.

À la connaissance de l'auteur, cette thèse est le premier ouvrage scientifique qui traite la

problématique de ré-optimisation en temps réel dans le domaine de la vente au détail. Les précédentes études ayant un objectif connexe étaient appliquées au domaine hospitalier. Cependant, les résultats dans le milieu hospitalier sont difficilement transposables au domaine de la vente au détail. De plus, contrairement à l'intuition, la flexibilité qui caractérise généralement le domaine de la vente au détail, ne le rend pas nécessairement plus facile lors d'une ré-optimisation. En effet, l'espace combinatoire pour traiter une petite perturbation peut être très grand surtout si cette ré-optimisation touche plusieurs jours de l'horizon.
La littérature sur l'optimisation globale d'un horaire de personnel repose sur deux approches principales pour la génération des quarts : l'approche explicite et l'approche implicite. Les caractéristiques de notre contexte ne nous ont pas permis d'utiliser directement les approches implicites proposées. Ainsi, nous avons proposé une nouvelle méthode intégrée, qui génère/affecte les quarts simultanément, qui s'est avérée très efficace, même dans un contexte de personnel très hétérogène.

CHAPITRE 8 CONCLUSION ET RECOMMANDATIONS

Dans cette thèse, nous avons présenté deux méthodes de ré-optimisation en temps réel pour corriger efficacement et très rapidement les petites perturbations qui peuvent survenir à tout moment dans un horaire de personnel (chapitres 4 et 5). Nous avons également montré comment on peut se servir d'une méthode de ré-optimisation efficace et rapide pour faire l'optimisation globale d'un horaire de personnel en présentant une nouvelle méthode au chapitre 6.

Dans ce chapitre, nous synthétisons nos travaux à la section 8.1. Dans la section 8.2, on expose certaines limitations des méthodes proposées et des améliorations futures en perspective.

8.1 Synthèse des travaux

Dans le chapitre 4, nous avons proposé une méthode de ré-optimisation rapide susceptible de gérer cinq types de décision différents. Cette méthode se base sur les valeurs duales obtenues en résolvant, dans la phase de planification, la relaxation linéaire du problème d'horaire de personnel. Les tests effectués sur 1050 scénarios dérivés de données réelles impliquant jusqu'à 191 employés ont démontré l'efficacité de cette méthode. La méthode a permis d'obtenir en moins d'une seconde en moyenne une solution optimale pour plus de 95% de ces scénarios. Nous avons également développé une procédure de mises à jour des valeurs duales après chaque perturbation lorsque plusieurs doivent être traitées au cours d'une même semaine. D'autres résultats ont montré l'utilité de cette procédure de mises à jour réduisant l'erreur de plus de 73%. Des tests supplémentaires effectués sur des cas où les valeurs duales ne sont pas aussi précises ont montré que l'approche reposant sur les mises à jour des valeurs duales reste toujours opérationnelle.

Dans le chapitre 5, nous avons proposé une méthode plus générale que celle proposée dans le chapitre 4. En effet, dans cette nouvelle méthode, nous n'avons pas besoin de l'information duale. De plus, la ré-optimisation concerne tous les jours restants de l'horizon. La méthode fait appel à l'optimisation multi-objectif afin de ne pas trop s'éloigner de l'horaire planifié, et elle se base sur des résultats théoriques. La méthode propose un ensemble de solutions permettant d'obtenir un bon compromis entre le coût et le nombre de modifications effectuées. Les tests menés sur 210 scénarios de perturbation générés aléatoirement, à partir des instances de données réelles, ont démontré l'efficacité de cette méthode : elle peut calculer les solutions exactes Pareto-optimales en moins d'une seconde en moyenne pour plus de 98% des scénarios de test.

Dans le chapitre 6, nous avons adapté la méthode proposée dans le chapitre 5 pour traiter un

autre problème : l'optimisation globale d'un horaire de personnel. Pour ce faire, nous avons considéré ce problème comme une séquence de ré-optimisations à mener afin de corriger une séquence de perturbations. Celles-ci sont stimulées à la base de certaines mesures indiquant la probabilité que la correction de chacune de ses perturbations améliore significativement la solution courante. La méthode se base sur des algorithmes de groupage qui retournent plusieurs sous-problèmes à chaque itération, où chacun de ces sous-problèmes représente un terrain riche de perturbations visées. Cette nouvelle approche proposée est considérée comme une approche intégrée qui génère/affecte les quarts simultanément. Elle était implémentée à l'aide de la programmation parallèle, ce qui a influencé très positivement les temps de calcul. Les tests effectués sur 33 instances (dont 14 sont réelles) ont montré l'efficacité de cette heuristique. Celle-ci arrive à produire, pour 97% des instances, des solutions de très bonne qualité en moins de 2 minutes en moyenne. Les tests ont également révélé l'efficacité (resp. la limite) de la génération dynamique (resp. la génération classique) des quarts.

8.2 Limitations des méthodes proposées et améliorations futures

La limite principale des méthodes de ré-optimisation en temps réel proposées dans les chapitres 4 et 5 repose sur le fait qu'elles n'anticipent pas, lors du traitement d'une perturbation, les autres événements futurs qui peuvent survenir aléatoirement durant le reste de l'horizon. Ce point fait l'objet d'une perspective de travail de recherche intéressante susceptible d'aboutir à la naissance d'une nouvelle méthode de ré-optimisation stochastique en temps réel. Cela nécessitera d'estimer un coût futur pour chaque employé en se basant sur l'historique de leur ponctualité et sur les variations de demande.

Pour améliorer ces deux méthodes, une autre option consisterait à inclure les préférences de l'employeur. En effet, pour corriger une perturbation, l'employeur a souvent des préférences. Celles-ci reposent sur l'historique et la perception que l'employeur a de ses employés : efficacité, rapidité, crédibilité, etc. Afin de tenir compte de ces critères, on peut utiliser l'optimisation interactive. Une fois les choix de réadaptation retournés par la méthode en question, l'employeur peut donner son retour et une nouvelle ré-optimisation peut alors être effectuée en conséquence. Afin de minimiser les retours de l'employeur, il est envisageable d'enregistrer dans une base de données toutes les décisions qui l'ont satisfait. Ensuite, il sera possible d'utiliser l'apprentissage automatique pour prévoir les préférences de correction de l'employeur. L'intégration de tous ces objectifs dans un seul logiciel peut mener à un produit très efficace.

Pour l'approche proposée dans le chapitre 6, la limitation principale concerne les instances disponibles pour les tests. Nous n'avons considéré que les règles génériques toujours présentes dans un problème d'horaire de personnel. Néanmoins, chaque client peut être concerné par de nouvelles règles qui lui sont spécifiques. La méthode doit être robuste avec l'intégration de ces

nouvelles règles. Il semble aussi intéressant de tester la méthode dans d'autres contextes. Dans le cas de règles nouvelles ou de nouveau contexte, nous pensons que la différence se situerait uniquement au niveau de la génération des nouveaux types de décisions génératrices, et qu'il suffirait d'adapter les mesures considérées pour cibler les bonnes perturbations à stimuler. Si la fonction objectif dans le nouveau contexte n'a pas la même structure que celle utilisée dans notre étude, i.e., si la fonction objectif n'est pas croissante et linéaire par morceaux, une adaptation des procédures de groupage utilisées semble nécessaire dans ce cas.

RÉFÉRENCES

[1] Mercer. (2008) The Total Financial Impact of Employee Absences. [En ligne]. Disponible : https://www.fmlainsights.com/wp-content/uploads/sites/813/2011/09/mercer-survey-highlights1.pdf

[2] L. C. Edie, "Traffic Delays at Toll Booths," *Journal of the Operations Research Society of America*, vol. 2, n°. 2, p. 107–138, 1954.

[3] G. B. Dantzig, "Letter to the editor-A Comment on Edie's "Traffic Delays at Toll Booths"," *Journal of the Operations Research Society of America*, vol. 2, n°. 3, p. 339–341, 1954.

[4] K. R. Baker et M. J. Magazine, "Workforce Scheduling with Cyclic Demands and Day-Off Constraints," *Management Science*, vol. 24, n°. 2, p. 161–167, 1977.

[5] A. T. Ernst, H. Jiang, M. Krishnamoorthy, B. Owens et D. Sier, "An annotated bibliography of personnel scheduling and rostering," *Annals of Operations Research*, vol. 127, n°. 1-4, p. 21–144, 2004.

[6] J. Van den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester et L. De Boeck, "Personnel scheduling : A literature review," *European Journal of Operational Research*, vol. 226, n°. 3, p. 367–385, 2013.

[7] R. J. Vanderbei *et al.*, *Linear programming*. Springer, 2015.

[8] A. Land et A. Doig, "An Automatic Method of Solving Discrete Programming Problems," *Econometrica 28*, p. 497–520, 1960.

[9] L. S. Lasdon, *Optimization theory for large systems*. Courier Corporation, 2002.

[10] M. Minoux, *Programmation mathématique. Théorie et algorithmes*. Lavoisier, 2008.

[11] M. E. Lübbecke et J. Desrosiers, "Selected Topics in Column Generation," *Operations Research*, vol. 53, n°. 6, p. 1007–1023, 2005.

[12] K. Apt, *Principles of Constraint Programming*. Cambridge university press, 2003.

[13] G. Weil, K. Heus, P. Francois et M. Poujade, "Constraint programming for nurse scheduling," *IEEE Engineering in medicine and biology magazine*, vol. 14, n°. 4, p. 417–422, 1995.

[14] S. Abdennadher et H. Schlenker, "Nurse Scheduling Using Constraint Logic Programming," dans *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative*

Applications of Artificial Intelligence, ser. AAAI '99/IAAI '99. Menlo Park, CA, USA : American Association for Artificial Intelligence, 1999, p. 838–843.

[15] S. Bourdais, P. Galinier et G. Pesant, "Hibiscus : A Constraint Programming Application to Staff Scheduling in Health Care," dans *International Conference on Principles and Practice of Constraint Programming*. Springer, 2003, p. 153–167.

[16] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001, vol. 16.

[17] J. L. Arthur et A. Ravindran, "A Multiple Objective Nurse Scheduling Model," *AIIE Transactions*, vol. 13, n°. 1, p. 55–60, 1981.

[18] I. Ozkarahan et J. E. Bailey, "Goal Programming Model Subsystem of a Flexible Nurse Scheduling Support System," *IIE Transactions*, vol. 20, n°. 3, p. 306–316, 1988.

[19] H. Beaulieu, J. A. Ferland, B. Gendron et P. Michelon, "A mathematical programming approach for scheduling physicians in the emergency room," *Health Care Management Science*, vol. 3, n°. 3, p. 193–200, 2000.

[20] M. N. Azaiez et S. S. Al Sharif, "A 0-1 goal programming model for nurse scheduling," *Computers & Operations Research*, vol. 32, n°. 3, p. 491–507, 2005.

[21] A. T. 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, n°. 1, p. 3–27, 2004.

[22] S. Kirkpatrick, C. D. Gelatt et M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, n°. 4598, p. 671–680, 1983.

[23] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research*, vol. 13, n°. 5, p. 533–549, 1986.

[24] P. Hansen, "The steepest ascent mildest descent heuristic for combinatorial programming," dans *Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy*, 1986, p. 70–145.

[25] N. Mladenović et P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, n°. 11, p. 1097–1100, 1997.

[26] J. H. Holland, "Outline for a Logical Theory of Adaptive Systems," *Journal of the ACM (JACM)*, vol. 9, n°. 3, p. 297–314, 1962.

[27] U. Aickelin et K. A. Dowsland, "Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem," *Journal of Scheduling*, vol. 3, n°. 3, p. 139–153, 2000.

[28] ——, "An Indirect Genetic Glgorithm for a Nurse-Scheduling Problem," *Computers & Operations Research*, vol. 31, n°. 5, p. 761–778, 2004.

[29] I. Berrada, J. A. Ferland et P. Michelon, "A multi-objective approach to nurse scheduling with both hard and soft constraints," *Socio-Economic Planning Sciences*, vol. 30, n°. 3, p. 183–193, 1996.

[30] H. E. Miller, W. P. Pierskalla et G. J. Rath, "Nurse Scheduling Using Mathematical Programming," *Operations Research*, vol. 24, n°. 5, p. 857–870, 1976.

[31] D. M. Warner, "Scheduling Nursing Personnel According to Nursing Preference : A Mathematical Programming Approach," *Operations Research*, vol. 24, n°. 5, p. 842–856, 1976.

[32] G. Felici et C. Gentile, "A Polyhedral Approach for the Staff Rostering Problem," *Management Science*, vol. 50, n°. 3, p. 381–393, 2004.

[33] W. B. Henderson et W. L. Berry, "Heuristic Methods for Telephone Operator Shift Scheduling : An Experimental Analysis," *Management Science*, vol. 22, n°. 12, p. 1372–1380, 1976.

[34] M. Segal, "The operator-scheduling problem : A network-flow approach," *Operations Research*, vol. 22, n°. 4, p. 808–823, 1974.

[35] J. G. Morris et M. J. Showalter, "Simple approaches to shift, days-off and tour scheduling problems," *Management Science*, vol. 29, n°. 8, p. 942–950, 1983.

[36] S. L. Moondra, "An LP model for work force scheduling for banks," *Journal of Bank Research*, vol. 7, n°. 4, p. 299–301, 1976.

[37] S. E. Bechtold et L. W. Jacobs, "Implicit modeling of flexible break assignments in optimal shift scheduling," *Management Science*, vol. 36, n°. 11, p. 1339–1351, 1990.

[38] I. Addou et F. Soumis, "Bechtold-Jacobs generalized model for shift scheduling with extraordinary overlap," *Annals of Operations Research*, vol. 155, n°. 1, p. 177–205, 2007.

[39] T. Aykin, "Optimal Shift Scheduling with Multiple Break Windows," *Management Science*, vol. 42, n°. 4, p. 591–602, 1996.

[40] M. Rekik, J.-F. Cordeau et F. Soumis, "Implicit shift scheduling with multiple breaks and work stretch duration restrictions," *Journal of Scheduling*, vol. 13, n°. 1, p. 49–75, 2010.

[41] N. Musliu, A. Schaerf et W. Slany, "Local Search for Shift Design," *European Journal of Operational Research*, vol. 153, n°. 1, p. 51–64, 2004.

[42] L. Di Gaspero, J. Gärtner, G. Kortsarz, N. Musliu, A. Schaerf et W. Slany, "The minimum shift design problem," *Annals of Operations Research*, vol. 155, nº. 1, p. 79–105, 2007.

[43] M. Widl et N. Musliu, "The break scheduling problem : complexity results and practical algorithms," *Memetic Computing*, vol. 6, nº. 2, p. 97–112, 2014.

[44] A. Beer, J. Gärtner, N. Musliu, W. Schafhauser et W. Slany, "Scheduling Breaks in Shift Plans for Call Centers," dans *Proc. of the 7th Int. Conf. on the Practice and Theory of Automated Timetabling, Montréal, Canada*, 2008.

[45] A. Bonutti, S. Ceschia, F. De Cesco, N. Musliu et A. Schaerf, "Modeling and solving a real-life multi-skill shift design problem," *Annals of Operations Research*, vol. 252, nº. 2, p. 365–382, 2017.

[46] J. F. Bard, D. P. Morton et Y. M. Wang, "Workforce planning at USPS mail processing and distribution centers using stochastic optimization," *Annals of Operations Research*, vol. 155, nº. 1, p. 51–78, 2007.

[47] R. M. Van Slyke et R. Wets, "L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming," *SIAM Journal on Applied Mathematics*, vol. 17, nº. 4, p. 638–663, 1969.

[48] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische mathematik*, vol. 4, nº. 1, p. 238–252, 1962.

[49] J. R. Birge et F. Louveaux, *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[50] P. Jaillet et M. R. Wagner, "Online Optimization-An Introduction," *TutORials in Operations Research : Risk and Optimization in an Uncertain World, INFORMS*, p. 142–152, 2010.

[51] N. Buchbinder et J. Naor, "The Design of Competitive Online Algorithms via a Primal : Dual Approach," *Foundations and Trends® in Theoretical Computer Science*, vol. 3, nº. 2–3, p. 93–263, 2009.

[52] V. H. Manshadi, S. O. Gharan et A. Saberi, "Online Stochastic Matching : Online Actions Based on Offline Statistics," *Mathematics of Operations Research*, vol. 37, nº. 4, p. 559–573, 2012.

[53] P. V. Hentenryck et R. Bent, *Online Stochastic Combinatorial Optimization*. The MIT Press, 2009.

[54] M. Moz et M. V. Pato, "An Integer Multicommodity Flow Model Applied to the Rerostering of Nurse Schedules," *Annals of Operations Research*, vol. 119, nº. 1-4, p. 285–301, 2003.

[55] ——, "Solving the Problem of Rerostering Nurse Schedules with Hard Constraints : New Multicommodity Flow Models," *Annals of Operations Research*, vol. 128, n°. 1–4, p. 179–197, 2004.

[56] J. F. Bard et H. W. Purnomo, "Hospital-wide reactive scheduling of nurses with preference considerations," *IIE Transactions*, vol. 37, n°. 7, p. 589–608, 2005.

[57] M. Moz et M. V. Pato, "A genetic algorithm approach to a nurse rerostering problem," *Computers & Operations Research*, vol. 34, n°. 3, p. 667–691, 2007.

[58] M. V. Pato et M. Moz, "Solving a bi-objective nurse rerostering problem by using a utopic Pareto genetic heuristic," *Journal of Heuristics*, vol. 14, n°. 4, p. 359–374, 2008.

[59] B. Mahenhout et M. Vanhoucke, "An evolutionary approach for the nurse rerostering problem," *Computers & Operations Research*, vol. 38, n°. 10, p. 1400–1411, 2011.

[60] M. Kitada et K. Morizawa, "A Heuristic Method for Nurse Rerostering Problem with a Sudden Absence for Several Consecutive Days," *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, n°. 11, p. 353–361, 2013.

[61] M. Kitada, K. Morizawa et H. Nagasawa, "A Heuristic Method in Nurse Rerostering Following a Sudden Absence of Nurses," dans *Proc. 11st Asia Pacific Industrial Engineering & Management Systems Conference*, vol. 6, 2010.

[62] C. N. Gross, A. Fügener et J. O. Brunner, "Online rescheduling of physicians in hospitals," *Flexible Services and Manufacturing Journal*, vol. 30, n°. 1, p. 296–328, 2018.

[63] C. Froger, "Mise à jour des horaires de personnel travaillant sur des quarts," *Mémoire de maîtrise. École Polytechnique de Montréal*, 2015.

[64] D. Meignan, "A heuristic approach to schedule reoptimization in the context of interactive optimization," dans *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2014, p. 461–468.

[65] M. Fisher, "Interactive optimization," *Annals of Operations Research*, vol. 5, n°. 1-4, p. 541–556, 1986.

[66] R. Bürgy, H. Michon-Lacaze et G. Desaulniers, "Employee scheduling with short demand perturbations and extensible shifts," *Omega*, vol. 89, p. 177–192, 2019.

[67] E. Burke, P. De Causmaecker et G. Vanden Berghe, "The State of the Art of Nurse Rostering," *Journal of Scheduling*, vol. 7, n°. 6, p. 441–499, 2004.

[68] J. F. Bard et H. W. Purnomo, "Short-Term Nurse Scheduling in Response to Daily Fluctuations in Supply and Demand," *Health Care Management Science*, vol. 8, n°. 4, p. 315–324, 2005.

[69] J. Clausen, J. Larsen et N. Rezanova, “Disruption management in the airline industry – Concepts, models and methods,” *Computers and Operations Research*, vol. 37, n°. 5, p. 809–821, 2010.

[70] V. Cacchiani, D. Huisman, M. Kidd, L. Kroon, P. Toth, L. Veelenturf et J. Wagenaar, “An overview of recovery models and algorithms for real-time railway rescheduling,” *Transportation Research Part B*, vol. 63, n°. 1, p. 15–37, 2014.

[71] A. M. Geoffrion et R. Nauss, “Exceptional Paper—Parametric and Postoptimality Analysis in Integer Linear Programming,” *Management Science*, vol. 23, n°. 5, p. 453–466, 1977.

[72] L. Schrage et L. Wolsey, “Sensitivity Analysis for Branch and Bound Integer Programming,” *Operations Research*, vol. 33, n°. 5, p. 1008–1023, 1985.

[73] L. Jenkins, “Parametric methods in integer linear programming,” *Annals of Operations Research*, vol. 27, p. 77–96, 1990.

[74] A. Mitsos et P. Barton, “Parametric mixed integer 0–1 linear programming : The general case for a single parameter,” *European Journal of Operational Research*, vol. 194, p. 663–686, 2009.

[75] J. H. Friedman, “Multivariate adaptive regression splines,” *Annals of Statistics*, p. 1–67, 1991.

[76] R. Hassani, G. Desaulniers et I. Elhallaoui, “Real-time personnel re-scheduling after a minor disruption,” *Rapport technique, Les Cahiers du GERAD G-2017-27, HEC Montréal, Montréal.*, 2017.

[77] S. Irnich et G. Desaulniers, “Shortest path problems with resource constraints,” dans *Column Generation*, G. Desaulniers, J. Desrosiers et M. M. Solomon, édit. Boston, MA : Springer US, 2005, ch. 2, p. 33–65.

ANNEXE A

Dans cette annexe, on représente le modèle de base cité dans l'article représenté au chapitre 5.

A.1 Mathematical model of the personnel scheduling problem

In this section, we present the integer programming model proposed by Hassani *et al.* [76] for the initial personnel scheduling problem. This model assumes that the horizon has seven days, numbered from 1 to 7. We describe first the required notation that has not be defined in the main text.

$\mathcal{S}^A$: Set of anonymous shifts used to avoid under-coverage ;

n^O : Minimum number of days off to assign to each employee ;

n^R : Minimum number of periods of rest between two consecutive shifts assigned to the same employee ;

d_w^p : Number of employees required for job w in period p ;

a_{sw}^p : Binary parameter equal to 1 if shift s covers job w in period p and 0 otherwise ;

$\mathcal{K}^L$: Set of steps in the step function defining the labor costs ;

n_k^L : Number of periods on step $k \in \mathcal{K}^L$;

c_k^L : Unit penalty on step $k \in \mathcal{K}^L$ (with $c_k^L < c_{k+1}^L$) ;

$\mathcal{K}^V$: Set of steps in the step function defining the over-coverage penalties ;

n_k^V : Number of periods on step $k \in \mathcal{K}^V$;

c_k^V : Unit penalty on step $k \in \mathcal{K}^V$ (with $c_k^V < c_{k+1}^V$) ;

c_s^Y : Penalty for each period in an anonymous shift s ;

M : Large constant ;

X_{es}^h : Binary variable equal to 1 if shift s is assigned to employee e on day h and 0 otherwise ;

Y_s : Nonnegative integer variable equal to the number of times that anonymous shift s is used ;

L_e^k : Nonnegative integer variable equal to the number of periods worked by employee e on step $k \in \mathcal{K}^L$;

V_w^{kp} : Nonnegative integer variable equal to the number of employees assigned to job w in over-coverage in period p on step $k \in \mathcal{K}^V$;

O_e^h : Binary variable equal to 1 if employee e has a day off on day h and 0 otherwise.

With this notation, the initial personnel scheduling problem can be modeled as the following integer program :

$$\text{Minimize} \quad \sum_{e\in\mathcal{E}}\sum_{k\in\mathcal{K}^L} c_k^L L_e^k + \sum_{s\in\mathcal{S}^A} c_s^Y l_s Y_s + \sum_{p\in\mathcal{P}}\sum_{w\in\mathcal{W}}\sum_{k\in\mathcal{K}^V} c_k^V V_w^{kp} \tag{A.1}$$

$$\text{subject to :} \quad \sum_{s\in\mathcal{S}_e^h} X_{es}^h + O_e^h = 1, \qquad \forall e\in\mathcal{E}, h\in\mathcal{H} \tag{A.2}$$

$$\sum_{h\in H} O_e^h \geq n^O, \qquad \forall e\in\mathcal{E} \tag{A.3}$$

$$\sum_{h\in\mathcal{H}}\sum_{s\in\mathcal{S}_e^h} l_s X_{es}^h - \sum_{k\in\mathcal{K}^L} L_e^k = 0, \qquad \forall e\in\mathcal{E} \tag{A.4}$$

$$M\, O_e^{h+1} + \sum_{s\in\mathcal{S}_e^{h+1}} b_s X_{es}^{h+1} - \sum_{s\in\mathcal{S}_e^h} (f_s + 1 + n^R) X_{es}^h \geq 0, \qquad \forall e\in\mathcal{E}, h\in\mathcal{H}\setminus\{7\} \tag{A.5}$$

$$\sum_{e\in\mathcal{E}}\sum_{h\in\mathcal{H}}\sum_{s\in\mathcal{S}_e^h} a_{sw}^p X_{es}^h + \sum_{s\in\mathcal{S}^A} a_{sw}^p Y_s - \sum_{k\in\mathcal{K}^V} V_w^{kp} = d_w^p, \qquad \forall p\in\mathcal{P}, w\in\mathcal{W} \tag{A.6}$$

$$X_{es}^h \in \{0,1\}, \qquad \forall e\in\mathcal{E}, h\in\mathcal{H}, s\in\mathcal{S}_e^h \tag{A.7}$$

$$O_e^h \in \{0,1\}, \qquad \forall e\in\mathcal{E}, h\in\mathcal{H} \tag{A.8}$$

$$L_e^k \in [0, n_k^L], \text{integer}, \qquad \forall e\in\mathcal{E}, k\in\mathcal{K}^L \tag{A.9}$$

$$Y_s \geq 0, \text{integer}, \qquad \forall s\in\mathcal{S}^A \tag{A.10}$$

$$V_w^{kp} \in [0, n_k^V], \text{integer}, \qquad \forall w\in\mathcal{W}, p\in\mathcal{P}, k\in\mathcal{K}^V \tag{A.11}$$

Objective function (A.1) minimizes the sum of the labor costs and the penalties incurred by the anonymous shifts and the over-coverage. Constraints (A.2) ensure that, on each day, each employee is assigned to a shift or a day off. Constraints (A.3) impose a minimum of n^O days off for each employee. Constraints (A.4) distribute the periods worked by each employee on the various steps of $\mathcal{K}^L$. Constraints (A.5) enforce a rest of at least n^R periods between shifts assigned to the same employee on two consecutive days. Finally, constraints (A.6) guarantee that, for each job and each period, a sufficient number of employees or anonymous shifts covers the demand. They also allow the computation of the number of employees in over-coverage per step in $\mathcal{K}^V$.

ANNEXE B

Dans cette annexe, on donne les preuves des propositions de l'article représenté au chapitre 5.

B.1 Proof of Proposition 1

proof. ($\Leftarrow$) *By the definition of a convex hull, all extreme points of any set Q belong to Q.* ($\Rightarrow$) *Assume that x is not an extreme point of $conv(\hat{\mathcal{X}})$. In this case, x can be written as a convex combination of two other solutions in $\hat{\mathcal{X}} \setminus \{x\}$, i.e., there exists $x_1, x_2 \in \hat{\mathcal{X}} \setminus \{x\}$ and $\alpha \in]0,1[$ such that $x = \alpha x_1 + (1-\alpha)x_2$. In particular, we have $x^b = \alpha x_1^b + (1-\alpha)x_2^b$. Because x^b, x_1^b and x_2^b are binary vectors, this equality implies that $x_1^b = x_2^b = x^b$. Since x_1^r, x_2^r and x^r depend on x_1^b, x_2^b and x^b, we get $x_1 = x_2 = x$, which contradicts $x_1, x_2 \in \hat{\mathcal{X}} \setminus \{x\}$. Consequently, the assumption is false and x is an extreme point of $conv(\hat{\mathcal{X}})$.*

B.2 Proof of Proposition 2

proof. ($\Rightarrow$) *Assume that x and y are adjacent in $conv(\hat{\mathcal{X}})$. Therefore, they are the extreme points of an edge of $conv(\hat{\mathcal{X}})$. From the primal simplex algorithm, we know that, to move along the edge between x and y, a minimal set of shift variables indexed by S must be entered into the basis. This set defines a direction d such that $y = x + d$, where*

$$d_j^b = \begin{cases} 1 & \text{if } j \in S \\ -1 & \text{if } j \in Supp^+(x^b) \setminus Supp^+(y^b) \\ 0 & \text{otherwise} \end{cases}$$

and d^r is derived from d^b. Each component $d_j^b = -1$ corresponds to a generating decision of type g^O that proposes a day off to an employee that was assigned to a shift on this day and each component $d_j^b = 1$ corresponds to a generating decision of type g^O that, at the opposite, assigns a shift to an employee that had a day off. Sequencing these decisions starting with the former ones yields an elementary decision because of the minimality of set S.
($\Leftarrow$) *Assume that there exists an elementary decision $d = (d^b, d^r)^\top \in \mathcal{D}(x)$ such that $y = x + d$. Let $S = Supp^+(d^b)$. Because decision d is elementary, entering into the basis the shift variables indexed by S in the order provided by d allows to move directly from x to y and the move only occurs when the last of these variables is pivoted in. This shows that x and y are adjacent in the polytope $conv(\hat{\mathcal{X}})$.*

B.3 Proof of Proposition 3

proof. *Let $d = x - x_0$ be the transition vector from x_0 to x. For each employee $e \in \mathcal{E}$ and day $h \in \mathcal{H}$ such that the sub-vector d_e^h is not null (starting with $\hat{e}$ and day $\hat{h}$), define the generating decisions $g^O(s_0(h), e, h)$ if $s_{x_0}(e,h) \neq s_0(h)$ (i.e., if $d_e^h(s_{x_0}(e,h)) = -1$) and $g^O(s_x(e,h), e, h)$ if $s_x(e,h) \neq s_0(h)$ (i.e., if $d_e^h(s_x(e,h)) = 1$) which allow to reassign employee e from shift $s_{x_0}(e,h)$ to shift $s_x(e,h)$ on day h and put them in a list in this order. Once all employees and days have been treated, this list forms a sequence of generating decisions of type g^O allowing a transition between x_0 and x. Scanning this list in order, one can identify a sequence of elementary decisions $d_1, d_2, \ldots, d_m$ by testing after each generating decision whether it yields a feasible solution or not. Note that $n_{d_i} \geq 1$ for all $i = 1, \ldots, m$. Indeed, for n_{d_i} to be equal to 0, all generating decisions in d_i would need to assign an employee to a day off on a given day. This is not possible for d_1 because this decision involves employee $\hat{e}$ who receives a new shift on day $\hat{h}$. Moreover, this is not possible for the other decisions d_i, $i = 2, \ldots, m$, because it would mean that all employees assigned to a day off in this decision were in over-coverage during their whole shift and, therefore, the solution $x_0 + \sum_{k=1}^{i-1} d_k$ is not in $\hat{\mathcal{X}}$, i.e., decision d_{i-1} is not a feasible elementary decision.*
Given that, for each employee e and day h, there is at most one vector d_i, $i = 1, \ldots, m$, with a positive component in the sub-vector d_e^h, we deduce that $\varphi^n(\delta_i) = \sum_{k=1}^{i} n_{d_k}$ for all $i = 1, \ldots, m$. Furthermore, because $n_{d_i} \geq 1$ for all $i = 1, \ldots, m$, we get that $\varphi^n(\delta_i) = \sum_{k=1}^{i} n_{d_k} < \sum_{k=1}^{i+1} n_{d_k} = \varphi^n(\delta_{i+1})$ for all $i = 1, \ldots, m-1$. Therefore, the sequence of elementary decisions $d_1, d_2, \ldots, d_m$ forms a policy in $\underline{\hat{\Pi}}(x_0, \Phi^c, \Phi^n)$ that can be applied to move from x_0 to x.

B.4 Proof of Proposition 4

proof. *Let $\delta = (d_1, d_2, \ldots, d_m) \in \underline{\hat{\Pi}}(x_0, \Phi^c, \Phi^n)$ with $m \geq 2$ and denote by $\delta' = (d_1, d_2, \ldots, d_{m-1}) \in \underline{\hat{\Pi}}(x_0, \Phi^c, \Phi^n)$ the policy obtained by omitting the last decision d_m from δ. Because $\varphi^n(\delta') < \varphi^n(\delta)$ by the definition of set $\underline{\hat{\Pi}}(x_0, \Phi^c, \Phi^n)$, policy δ cannot have a minimum number of modifications, i.e., $\delta \neq \delta_n^*$. Consequently, δ_n^* must have a single elementary decision.*

B.5 Proof of Proposition 5

proof. *Assume that there exists such an elementary decision d. Because $\mu(x_0, \delta, d) = 1$, we deduce that*

$$\frac{\min\limits_{d' \in \mathcal{D}(x_\delta)} \varphi^n(\delta \oplus d')}{\varphi^n(\delta \oplus d)} = 1, \quad \sqrt[m]{\frac{|\mathcal{E}_\delta \cap \mathcal{E}_d|}{|\mathcal{E}_\delta \cup \mathcal{E}_d|}} = 1, \quad \frac{\min\{N_{x_0}(e), N_{x_\delta + d}(e)\}}{\max\{N_{x_0}(e), N_{x_\delta + d}(e)\}} = 1, \forall e \in \mathcal{E}_\delta \cap \mathcal{E}_d.$$

$$
\Rightarrow \quad \begin{cases} \min\limits_{d^{'} \in \mathcal{D}(x_\delta)} \varphi^n(\delta \oplus d^{'}) = \varphi^n(\delta \oplus d), \\ \mathcal{E}_\delta \cap \mathcal{E}_d = \mathcal{E}_\delta \cup \mathcal{E}_d \\ \min\left\{N_{x_0}(e), N_{x_\delta + d}(e)\right\} = \max\left\{N_{x_0}(e), N_{x_\delta + d}(e)\right\}, \quad \forall e \in \mathcal{E}_\delta \cap \mathcal{E}_d \end{cases}
$$

$$
\Rightarrow \quad \begin{cases} \varphi^n(\delta \oplus d) \leq \varphi^n(\delta \oplus d^{'}), \quad \forall d^{'} \in \mathcal{D}(x_\delta) \\ \mathcal{E}_\delta = \mathcal{E}_d \\ N_{x_0}(e) = N_{x_\delta + d}(e), \qquad \forall e \in \mathcal{E}_\delta \end{cases}
$$

$$
\Rightarrow \quad \begin{cases} \varphi^n(\delta \oplus d) \leq \varphi^n(\delta \oplus d^{'}), \quad \forall d^{'} \in \mathcal{D}(x_\delta) \\ c(x_0) = c(x_\delta + d) \end{cases}
$$

$$
\Rightarrow \quad \begin{cases} \varphi^n(\delta \oplus d) \leq \varphi^n(\delta \oplus d^{'}), \quad \forall d^{'} \in \mathcal{D}(x_\delta) \\ \varphi^c(\delta \oplus d) = 0. \end{cases}
$$

Clearly, any policy $\delta^{'} = \delta \oplus (d_{m+1}, \ldots, d_{m+q}) \in \hat{\underline{\Pi}}^{=}(x_0, \Phi^c, \Phi^n)$ *with* $d_{m+1} \neq d$ *and* $q \geq 1$ *cannot dominate* $\delta \oplus d$ *because* $\varphi^n(\delta^{'}) \geq \varphi^n(\delta \oplus d_{m+1}) \geq \varphi^n(\delta \oplus d)$ *and* $\varphi^c(\delta^{'}) \geq 0 = \varphi^c(\delta \oplus d)$.