POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Optimisation de roulements de chauffeurs d'autobus

SAFAE ER-RBIB
Département de mathématiques et de génie industriel

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

Mai 2020

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Optimisation de roulements de chauffeurs d'autobus

présentée par **Safae ER-RBIB**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

François SOUMIS, président
Guy DESAULNIERS, membre et directeur de recherche
Issmaïl ELHALLAOUI, membre et codirecteur de recherche
Louis-Martin ROUSSEAU, membre
Mustapha OUHIMMOU, membre externe

DÉDICACE

À l'âme de L'Mima Aicha,
Tu inondes toujours mes pensées...

«Life is like riding a bicycle. To keep your balance you must keep moving.»

Albert Einstein (1879-1955).

REMERCIEMENTS

Mon parcours doctoral était sans doute l'expérience la plus marquante de ma vie, un voyage rempli de défis et de difficultés. Les moments de doute semblaient plus longs, et le stress était plus grand que les fruits de la réussite. Des gens très spéciaux étaient là, pour me soutenir, m'encourager et pour me montrer à quel point je peux être déterminée et motivée.

Ma profonde gratitude, et respect s'adressent à mes directeurs de recherche Pr. Guy Desaulniers et Pr. Issmail Elhallaoui, sans votre soutien bienveillant ce projet n'aurait jamais pu aboutir, un grand merci pour le soutien tant moral que financier. Merci à Guy pour la confiance, l'encadrement et l'accompagnement tout au long de la thèse. Merci pour votre rigueur et discipline, merci de votre aide énorme dans la rédaction, et vos judicieux conseils dans les présentations. Merci à Issmail pour tous les conseils et les bouffées de motivation, pour l'aide technique, l'attention aux détails et l'accompagnement tout au long de mon parcours. Merci pour toutes ces réunions spontanées et fructueuses que nous avons pu avoir.

Je remercie les membres du jury, François Soumis, Louis-Martin Rousseau et Mustapha Ouhimmou, qui m'ont fait l'honneur de juger ce travail doctoral. Je remercie également le CRSNG et GIRO Inc. qui ont participé au financement de ce projet. Je suis aussi reconnaissante à Charles Fleurent et Marc Gendron de GIRO Inc. pour l'accompagnement et la définition du projet.

Un merci spécial à Abderrahmane Bani pour l'aide en programmation, sans ton soutien et tes astuces, je ne saurais comment me débrouiller en code, merci aussi pour l'ami que j'ai trouvé en toi tout au long de ces années. Je remercie aussi Patrick Monroe, j'ai eu la chance de te rencontrer à la fin de mon parcours, mais tes idées et ton aide en rédaction ont été si précieuses. Un grand merci à Pierre Girard et Edoh Logo pour l'aide technique et tout ce que vous m'avez appris sur Linux. Merci à Edoh pour nos petites discussions matinales et la bouffée d'encouragement et de motivation que tu me donnais pour que je puisse avancer sur mon projet, et aussi pour les déblocages 'ssh', même pendant les fins de semaines.

Je tiens à exprimer mes sentiments de gratitude aux professeurs que j'ai côtoyés durant tout mon parcours scolaire, et qui ont laissé un grand impact dans ma vie : Fatiha, Khadija, Rachid, Dolkifl, Lmkhantar, Laayel, Abdelhak, Habiba, Mohammed, Nizar, Alain, Charles... À ceux qui nous ont quitté trop tôt, que vos âmes reposent en paix.

J'ai eu l'occasion de rencontrer des gens très spéciaux, qui sont devenus comme une famille au sein du Gerad, Merci à Rachid, Adil, Youssouf, Ilyas, Salah, Omar, Chahid, Dina, Cherifa,

Adham, Sarah... pour la bonne humeur, le respect, les belles discussions et les anecdotes.

J'ai pu aussi avoir un cercle d'amies qui sont maintenant plus qu'une famille, je les remercie d'être à mes côtés dans les moments les plus difficiles et les plus joyeux aussi de ma vie Fadwa, Amal, Fatiha, Zeyneb, Mayssoun, Manar, Greta, Meryam, Lucie...

Merci à ma chère famille, mes parents Hnyia et Abdelhak qui me soutiennent depuis le tout début de mon parcours, merci d'être là, dans mon cœur, malgré l'océan qui nous sépare, merci de n'avoir jamais épargné aucun effort pour me soutenir, merci d'avoir forgé ma personnalité, et de m'avoir préparée à accueillir à bras ouvert la vie. Un grand merci à ma sœur Rim et mon frère Taha, vous êtes les prunelles de mes yeux, merci d'exister. Finalement merci à mon conjoint, Kazem, pour tous les gestes directs et indirects qui m'ont aidé à bien mener ma thèse, merci pour les tons fermes d'encouragement et de motivation. Merci pour l'amour la patience et le soutien.

RÉSUMÉ

Le problème de roulements de chauffeurs d'autobus vise à déterminer les horaires de travail des chauffeurs d'autobus sur un horizon donné. Il s'agit d'un problème où des séquences de jours de repos et de journées de travail sont construites. Les journées de travail sont générées lors de la résolution du problème de construction de journées de travail. Ce problème a pour objectif de générer des journées de travail anonymes afin d'assurer, à un coût minimum, la couverture complète des horaires d'autobus. Plusieurs règles doivent être respectées lors de la résolution, entre autres, l'amplitude maximale ou minimale d'une journée de travail, le temps maximal ou minimal de travail, etc. Lorsque les journées de travail sont déterminées, elles sont affectées aux différents chauffeurs disponibles et les roulements des chauffeurs sont construits à cette étape. Les journées de travail sont affectées en respectant un ensemble de règles dérivées des conventions collectives, et chaque journée de travail est effectuée par un chauffeur durant un jour de la semaine. Dans notre contexte, les roulements de chauffeurs d'autobus sont cycliques et définis sur une semaine pour un certain horizon de planification. Ainsi les journées de travail peuvent varier d'un jour à l'autre mais se répètent d'une semaine à une autre.

Le problème de roulements avec jours de repos fixés vise à affecter les journées de travail aux différents chauffeurs dans les jours de travail (c'est-à-dire, les jours qui ne sont pas des jours de repos). Nous proposons, d'abord, une nouvelle formulation forte en nombres entiers du problème de roulements avec repos fixés. Les règles d'affectation des journées de travail sont diverses et compliquées, surtout qu'elles impliquent des contraintes de repos de nuit entre deux journées de travail et des contraintes qui s'étendent sur plusieurs jours et parfois sur plusieurs semaines. La fonction objectif vise à équilibrer le plus possible la charge de travail entre tous les chauffeurs. Ceci a été traduit par la minimisation des déviations positives par rapport à la moyenne des charges de travail totale par semaine de toutes les journées de travail. Différentes modélisations des contraintes de repos de nuit ont été proposées, ainsi qu'une deuxième formulation de la fonction objectif, mais qui vise aussi à équilibrer la charge de travail entre les chauffeurs d'autobus. Nous avons montré que la nouvelle formulation permet de reserrer l'espace de recherche lors du branchement, ce qui permet d'avoir des solutions entières plus rapidement.

Ensuite, une approche est proposée pour résoudre le problème de roulements intégré de construction de jours de repos et d'affectation de journées de travail. Le problème est modélisé comme un programme linéaire mixte en nombres entiers. Étant donné que le problème ne

contient pas de règles de quarts de travail ni de règles souples (des préférences par exemple), le problème présente beaucoup de symétrie. Le modèle s'est avéré très difficile à résoudre à l'optimalité avec le solveur commercial CPLEX malgré l'ajustement très poussé des paramètres et l'utilisation des méthodes avancées de programmation en nombres entiers (fixation de variables, branchement priorisé, ...). Sur la base de ce modèle, nous avons introduit une matheuristique à deux étapes qui permet de trouver des solutions de très bonne qualité. En utilisant une telle solution comme donnée d'entrée dans un solveur commercial, le modèle intégré peut être résolu beaucoup plus rapidement. Nos expériences de calcul testées sur des instances réelles de grande taille ont montré l'efficacité de la matheuristique. Des solutions optimales ont été obtenues dans des temps de calcul relativement courts (3.5 heures pour le cas impliquant jusqu'à 333 chauffeurs). En outre, en fournissant ces solutions comme solutions initiales au solveur CPLEX, de grandes accélérations (jusqu'à 99%) ont été obtenues pour résoudre le problème intégré avec une optimalité prouvée. L'article intitulé "Integrated and sequential solution methods for the cyclic bus driver rostering problem" traitant cet objectif a été publié dans la revue "Journal of the Operational Research Society"

Finalement, nous avons intégré des règles relatives aux préférences des chauffeurs dans le modèle de roulements. Le nouveau modèle vise à affecter les journées de travail aux différents chauffeurs sur un horizon prédéfini, tout en respectant les règles strictes d'affectation, en équilibrant la charge de travail entre les chauffeurs et en minimisant le plus possible les violations des règles souples (les préférences). Deux nouvelles matheuristiques ont été proposées. La première limite l'espace de recherche en pré-assignant les journées de travail aux roulements avec des jours de repos fixés. La deuxième matheuristique utilise un problème de partitionnement d'ensemble pour décomposer les roulements de grande taille en sous-roulements de tailles petites à moyennes. Dans une série d'expériences de calcul menées sur des instances réelles, nous montrons que ces matheuristiques peuvent être utilisées pour produire des solutions de bonne qualité pour des grandes instances (333 chauffeurs et 1509 journées de travail) dans des temps de calcul relativement courts. L'article intitulé "Preference-based bus driver rostering problem with fixed days off" traitant cet objectif a été soumis à la revue "Public Transport"

ABSTRACT

The bus driver rostering problem aims at building the work schedules of bus drivers over a given period of time. Solving such problem results in sequences of days off and duties. The duties are constructed via the duty scheduling problem, which creates anonymous duties in order to ensure, at a minimum cost, complete coverage of a set of bus trips. Several rules must be respected while solving this problem, i.e. maximum or minimum span of a duty, maximum or minimum working time, etc. The resulting duties must then be assigned to the different available drivers, creating their rosters. This process complies with a set of rules derived from collective agreements. Every duty is performed by one driver on one day of the week. Here in this context, bus driver rosters are cyclic, and defined over a week for a certain planning horizon. Thus, duties may vary from a day to another, but they are repeated weekly.

The rostering problem with fixed days off aims at assigning duties to drivers in working days. First, a new mixed integer formulation of the problem is proposed. The assignment rules are diverse and complicated, especially since they involve night rest constraints between two duties and constraints that are extended over several days, and sometimes over several weeks. The objective function is to balance the workload among all the drivers. This has been achieved by minimizing positive deviations from the average total workload per week. Furthermore, different formulations of the night rest constraints are presented, as well as, a second formulation of the objective function that minimizes the sum of the absolute values of the deviations from the average workload per week. It is shown that the first proposed formulation makes it possible to tighten the search space during the branch-and-bound process and, consequently, helps finding integer solutions more rapidly.

Next, an approach is proposed to solve the integrated days off scheduling and rostering problem. First the problem is modeled as a mixed integer linear program. In this problem, there are no shifts, and therefore, no shift related rules that reduce the solution space, nor shift related preferences that can reduce symmetry in the branch-and-bound process and ease the search for integer solutions. This model turns out to be very hard to solve to optimality without providing an initial solution. Based on this model, we introduce a new two-step matheuristic that can compute high-quality solutions. Using such a solution as an input to a commercial solver, the integrated model can be solved much more rapidly. Our computational results obtained on real-world instances involving up to 333 drivers and 1509 duties show that these initial solutions are optimal in most cases and, consequently, that the

proposed matheuristic is very efficient by itself.

Finally, we integrated the bus driver preference rules to the rostering problem. The new model aims at assigning duties to different drivers over a predefined cyclic horizon, while respecting a set of rules (hard constraints), balancing the workload among the drivers and satisfying as much as possible the driver preferences (soft constraints). We first model the problem as a mixed integer linear program that minimizes the number of preference violations while maintaining the workload balance of the solutions within a certain margin relative to the optimal one. Since this model is hard to solve for large instances, we propose two new matheuristics. The first one restricts the search space by preassigning duties to rosters based on an optimal solution to the duty assignment problem with fixed days off. The second algorithm makes use of a set partitioning problem to decompose rosters consisting of a large number of positions into sub-rosters of smaller sizes. In a series of computational experiments conducted on real-world instances, we show that these matheuristics can be used to produce high-quality solutions for large instances of the problem, within short computational times.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

STM	Société de transport de Montréal
MIP	Programme linéaire mixte
LP	Programme linéaire
PNE	Programme en nombre entier
GIRO	Le Groupe en Informatique et Recherche Opérationnelle
RCRF	Roulements cyclique avec jours de repos fixés
CDT	Contraintes de différence de temps
MF	Modélisation forte
H-Ind	Heuristique industrielle
CBDRP	Cyclic bus driver rostering problem
DOS	Days-off scheduling
DDT	Difference-duty-times
DA	Duty assignment
TSM	Two-step matheuristic
INTIS	Integrated-model-based algorithm with an initial solution
IND	Industrial algorithm
DAPDP	Duty assignment problem with consideration of driver preferences
PDR	Preassigning Duties to Rosters
PRP	Partitioning Roster Positions

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Le transport urbain est un domaine en forte croissance. La STM a, par exemple, enregistré une hausse d'environ 18% d'achalandage entre les années 2008 et 2018 (STM 2008, STM 2018). Cette situation s'applique à l'échelle mondiale et ne cesse d'augmenter. Le transport urbain contribue aux solutions sociales, écologiques et économiques des multiples défis auxquels les villes et régions sont confrontées : réduction des émissions polluantes, des coûts d'entretien et de construction de voies publiques, ainsi que la réduction des embouteillages routiers et des besoins en stationnement. Afin de garantir un service fiable et efficace, tout en gérant les coûts d'exploitation et profits, les compagnies de transport urbain se trouvent dans l'obligation d'améliorer leurs approches et de développer de nouvelles techniques à tous les niveaux de planification : stratégique, tactique et planification des opérations.

L'objectif de cette thèse est d'étudier le problème de roulements des chauffeurs d'autobus. Ce problème appartient à la phase de la planification des opérations, et consiste à définir un calendrier de travail et de repos pour chaque chauffeur sur une période de planification. En général, ce problème vise à équilibrer la charge de travail entre les chauffeurs et à maximiser leurs satisfactions. Nous proposons différentes approches de résolution afin d'obtenir des roulements à moindres coûts et de bonne qualité.

Dans la section 1.1, nous présentons les différentes définitions et concepts de base relatifs au problème de roulements. Ensuite, la section 1.2 définit les éléments de la problématique abordée dans cette thèse. Les objectifs de recherche et le plan de la thèse sont présentés dans les sections 1.3 et 1.4 respectivement.

1.1 Définitions et concepts de base

Dans le texte suivant, nous ferons appel à la terminologie suivante :

- *Voyage :* Parcours ou trajet que prend un autobus d'un point à un autre ;
- *Voyage actif :* Voyage durant lequel l'autobus transporte des passagers ;
- *Voyage de positionnement :* Voyage durant lequel il n'y a pas de passagers à bord de l'autobus ;
- *Journée de travail :* Série ordonnée de voyages effectués par un chauffeur durant un jour de la semaine ;
- *Roulement :* Horaire d'un groupe de chauffeurs d'autobus durant une période de temps prédéfinie. Chaque groupe comprend des chauffeurs qui ont les mêmes règles et préfé-

rences ;

— *Position :* Horaire d'un chauffeur d'autobus durant une semaine ;

— *Horaire d'autobus :* Ensemble de voyages actifs qui doivent être opérés par les autobus d'un dépôt (ici : garage ou espace de stationnement) durant une plage horaire définie ;

Comme décrit dans Desaulniers et Hickman (2007), le processus de planification pour le transport urbain passe généralement par les trois phases suivantes : stratégique, tactique et opérationnelle.

La planification stratégique consiste à prendre des décisions à long terme. À ce niveau de planification, les compagnies de transport urbain essayent de maximiser la qualité de service tout en respectant les restrictions budgétaires. C'est durant cette phase que le réseau est conçu et l'ensemble des lignes d'autobus sont développées ; c'est-à-dire le tracé de chaque ligne et les points d'arrêt. La planification tactique concerne les étapes intermédiaires dans le processus de planification et est effectuée sur une base saisonnière. Dans cette étape de planification, les fréquences de voyage, c'est-à-dire le nombre de départs sur chaque ligne et l'heure de chacun de ces départs sont déterminées. Ces fréquences sont sélectionnées de façon à satisfaire au mieux les besoins des passagers (par exemple : effectuer facilement des correspondances entre plusieurs lignes d'autobus).

Lors de la phase opérationnelle, les compagnies de transport visent à minimiser le coût total des opérations. Les problèmes sont généralement résolus sur une base mensuelle et hebdomadaire et parfois même sur une base quotidienne. Le problème d'horaires d'autobus représente la première étape de la planification des opérations. Le problème d'horaires d'autobus vise à construire des horaires réalisables et de moindre coût respectant plusieurs critères. Un horaire doit être composé d'un voyage de départ à partir d'un dépôt, d'une séquence de voyages actifs (tous les trajets planifiés) séparés par des voyages de positionnement, et d'un voyage de retour vers le même dépôt. Ce problème correspond soit à un problème d'horaires de véhicules à un seul dépôt, soit à dépôts multiples selon si la compagnie exploite sa flotte d'autobus d'un seul dépôt ou de dépôts multiples, ou si plusieurs types de véhicule sont utilisés. La complexité de ce dernier dépend du contexte opérationnel et pratique. À titre d'exemple, dans le cas où la compagnie opère une flotte d'autobus homogène d'un seul dépôt, le problème se réduit à un problème de flot, et donc à un problème facile à résoudre. En revanche, si le problème requiert plusieurs dépôts, ou si la flotte d'autobus opérée est hétérogène, le problème devient NP-difficile.

Le deuxième problème résolu dans cette phase de planification est le problème d'horaires des chauffeurs d'autobus. Les chauffeurs sont également affectés à des dépôts comme les autobus et, par conséquent, ce problème est séparable par dépôt. C'est un problème complexe,

notamment pour des instances de grande taille, d'où la nécessité de le résoudre en deux étapes importantes. La première concerne la construction des journées de travail (en anglais, *duties*), et la seconde la construction des roulements (en anglais, *rosters*).

Le problème de construction des journées de travail sert à générer des journées de travail anonymes afin d'assurer, à un coût minimum, la couverture complète des horaire d'autobus. Plusieurs règles doivent être respectées lors de la résolution de ce problème, par exemple l'amplitude maximale ou minimale d'une journée de travail, le temps maximum de travail sans pause, etc.

Finalement, lorsque les journées de travail sont déterminées, elles sont affectées aux différents chauffeurs disponibles et les roulements des chauffeurs sont construits à cette étape. Il s'agit d'un problème où on détermine des séquences de journées de travail et de repos sur un horizon défini de planification tout en respectant un ensemble de règles dérivées des conventions collectives. À ce niveau de planification, une journée de travail est effectuée par un chauffeur durant un jour de la semaine. Dans notre contexte, les roulements des chauffeurs d'autobus sont cycliques et en général, définis sur une semaine pour un certain horizon de planification. Ainsi, les journées de travail peuvent varier d'un jour à l'autre mais se répètent d'une semaine à une autre.

1.2 Éléments de la problématique

Le problème de roulements des chauffeurs d'autobus *bus driver rostering problem* vise à affecter des jours de repos et des journées de travail aux différents chauffeurs. Les journées de travail représentent une donnée d'entrée au problème de roulements. Elles sont générées lors de la résolution du problème indépendant de construction de journées de travail *Duty scheduling problem*. Ce dernier est une question financière majeure pour une compagnie de transport urbain, puisqu'il détermine la plus grande partie des salaires payés aux chauffeurs. Ce problème consiste à déterminer des journées de travail des chauffeurs basés dans un dépôt. Chaque journée de travail générée est caractérisée par une heure de début, une heure de fin, une charge de travail rémunérée, le jour de la semaine auquel elle appartient, le ou les groupes de chauffeurs auxquels elle peut être assignée, ainsi que les lignes qu'elle peut couvrir.

Tout comme la construction de journées de travail, la validité des roulements est régie par les règles de sécurité et de convention collective. Par exemple, un chauffeur ne peut pas travailler plus qu'un certain nombre de jours consécutifs. Dans la plupart des compagnies de transport urbain nord-américaines, les roulements sont construits par ordre de séniorité, laissant peu de place pour l'optimisation. De l'autre côté de l'océan, plusieurs compagnies européennes

construisent leurs roulements en considérant comme objectif principal la distribution équitable de la charge de travail entre les chauffeurs, ce qui produit un problème intéressant d'optimisation. Les chauffeurs d'autobus sont partitionnés par roulement : chaque chauffeur effectue une position donnée à la première semaine, la position suivante dans la semaine qui suit et ainsi de suite. Le chauffeur qui effectue la dernière position sera assigné à la première position dans la semaine suivante. Ainsi, pour un roulement de n positions, tous les chauffeurs effectuent l'horaire de chaque position à toutes les n semaines, assurant ainsi un horaire de travail équitable entre ceux-ci.

L'objectif de la construction de roulements est double ; il consiste d'une part à minimiser le nombre de journées de travail non affectées et, d'autre part, à balancer la moyenne hebdomadaire des heures de travail entre les chauffeurs d'autobus. D'un point de vue de programmation mathématique, le problème de roulements peut être formulé comme un problème de partitionnement d'ensemble ou un problème de recouvrement d'ensemble. Cependant, la résolution d'un tel modèle n'est pas populaire pour les problèmes de roulements en transport urbain, contrairement au transport aérien ou ferroviaire de longue distance. Ceci réside surtout dans le caractère cyclique des solutions recherchées. En outre, les problèmes de transport urbain ne sont pas séparables par type de véhicules, comme c'est le cas dans le transport aérien puisque tous les chauffeurs sont généralement autorisés à conduire tous les autobus. De plus, dans ces problèmes, les tâches à couvrir correspondent à des journées de travail individuelles, alors que ces tâches, appelées rotations, peuvent couvrir plusieurs journées allant jusqu'à six jours dans le transport aérien.

Le problème de roulements est traditionnellement résolu suivant une approche séquentielle, qui consiste à construire des patrons de jours de repos et ensuite à affecter les journées de travail. Les jours de repos sont affectés sans prendre en considération aucun élément crucial de l'affectation des journées de travail (comme équilibrer la charge de travail par exemple). De plus, dans plusieurs problèmes de roulements, le processus passe par une étape intermédiaire, celle de l'affectation des quarts de travail auxquels les journées de travail sont assignées. Par exemple, dans plusieurs cas, les journées de travail sont caractérisées par un type de quart de travail, entre autres, jour, soir ou nuit. Il est commun d'affecter les quarts de travail en premier, et ensuite leurs assigner des journées de travail dépendamment du type du quart de travail.

La méthode séquentielle facilite la tâche de la modélisation mathématique et réduit considérablement la taille du problème (nombre de variables et de contraintes), mais peut malheureusement générer des solutions de piètre qualité. En effet, il se peut que la configuration des patrons de repos ne soit pas bien adaptée aux journées de travail à assigner, ce qui mène à

des journées de travail non affectées ou à des roulements mal équilibrés.

Un grand défi rencontré est celui d'intégrer les deux problèmes de configuration de jours de repos et d'affectation des journées de travail, ce qui permettrait sans doute d'améliorer la qualité des roulements en équilibrant le plus possible la charge de travail entre les employés, et en affectant le plus de journées de travail possible. Toutefois, en raison de sa complexité, la résolution du problème intégré requiert en général de grands temps de calcul.

D'autre part, les compagnies de transport urbain accordent une grande importance aux employés (chauffeurs d'autobus). Dans l'objectif d'atteindre un meilleur service aux clients, il faut répondre d'abord aux satisfactions du personnel au travail. En plus des règles strictes qui régissent la construction des roulements, les préférences (comme le choix de certaines journées de travail ou de jours de repos) d'employés figurent dans les préoccupations majeures des compagnies de transport urbain. Ces préférences, qui peuvent être traduites par des règles souples, sont déterminantes dans le choix des horaires, la qualité et l'harmonie des différentes journées de travail et le choix des jours de repos. L'objectif de répondre au maximum aux préférences des chauffeurs peut évidemment générer des roulements moins équilibrés. Cependant, même si l'intégration des préférences dans le problème de roulement peut nuire à l'équité des charges de travail totales, ceci ne présentera pas un souci d'équité des charges dans le même roulement d'employés, puisqu'ils cyclent sur les mêmes positions.

1.3 Objectifs de recherche

L'objectif ultime de cette thèse est de concevoir des algorithmes qui permettront de résoudre efficacement des instances très difficiles du problème de roulements en transport urbain, à savoir le problème intégré de jours de repos et d'affectation des journées de travail. Dans notre problème, nous travaillons sur des journées de travail individuelles qui ne peuvent pas être réduites à des quarts de travail. En revanche, un seul problème de roulements qui considère des types ou quarts (par exemple : jour, soir, nuit) de journées de travail est étudié. Pourtant, aucune étape intermidiaire considérant seulement l'affectation des types de journées de travail n'est appliquée. Dans ce cas, la combinatoire et la symétrie du problème sont augmentées, ce qui paralyse complétement la méthode *Branch & Bound* utilisée par un solveur commercial aussi sophistiqué soit-il.

Le problème de roulements est difficile pour plusieurs raisons. D'abord, les instances sont de grande taille et peuvent aller jusqu'à 333 chauffeurs et plus de 1500 journées de travail à affecter, ce qui engendre un très grand nombre de variables et de contraintes à considérer. Ensuite, la nature des règles strictes d'affectation est complexe : ces règles peuvent s'étaler

sur plusieurs semaines. Finalement, la taille de l'horizon de planification n'est pas fixe pour tous les roulements, mais dépend plutôt de leurs tailles. En outre, lors de l'intégration du problème de jours de repos avec le problème d'affectation des journées de travail, le problème devient encore plus complexe. La symétrie augmente énormément, puisque la plupart des journées de travail peuvent être affectées à tous les roulements.

Nous proposons tout d'abord des modèles de programmation en nombres entiers (MILP), qui peuvent être résolus en utilisant des solveurs commerciaux. La plupart des instances (à savoir les instances de grande taille) ont été résolues à l'optimalité en des temps de calcul raisonnables.

Nous montrons que l'approche séquentielle aide à réduire la symétrie de nos modèles, sur la base desquels nous développons ensuite une matheuristique pour le modèle intégré. Cette matheuristique est conçue en deux étapes qui peuvent produire des solutions allant à l'optimalité, en des temps de calcul relativement rapides pour les instances de grande taille.

Finalement, nous développons un processus de décomposition des roulements de grande taille, pour permettre de violer le moins possible les contraintes (souples) de préférences au cours de la résolution, dans le but de satisfaire les préférences des chauffeurs d'autobus envers leurs horaires.

1.4 Plan de la thèse

Cette thèse est structurée de la façon suivante. Le chapitre 2 présente une revue de littérature concernant les problèmes de roulements dans le transport urbain et dans d'autres domaines connexes. Le chapitre 3 décrit l'organisation des trois axes principaux de la thèse. Le chapitre 4 présente le premier objectif de la thèse qui s'intéresse au problème de roulements avec jours de repos fixés. Le chapitre 5 est consacré au deuxième objectif portant sur la résolution du problème intégré de jours de repos et d'affectation de journées de travail. Ce deuxième objectif a fait l'objet d'un article publié dans la revue *Journal of the Operational Research Society*. Le chapitre 6 concerne le dernier objectif portant sur le traitement des préférences des chauffeurs d'autobus dans les problèmes de roulements, un article soumis à la revue *Public Transport*. Une discussion générale sera abordée dans le chapitre 7 et finalement, les conclusions sont présentées dans le chapitre 8.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre passe en revue la littérature sur le problème de construction de roulements de personnel, en particuler les roulements de chauffeurs d'autobus, ainsi que les méthodes utilisées récemment pour le résoudre. La section 2.1 présente un aperçu sur les étapes de planification de roulements de personnel en général. Dans la section 2.2 seront définies les méthodes exactes et les métaheuristiques employées pour la résolution de ce type de problèmes. Dans la section 2.3, nous présentons les trois axes spécifiques de la thèse et comment ils ont été abordés dans la littérature. Finalement, la section 2.4 synthétise ce qui a été traité dans la revue de littérature et énonce brièvement les objectifs de la thèse.

2.1 Problème de roulements

Le problème de roulements de personnel sert à construire les horaires d'employés d'une organisation donnée afin qu'elle puisse répondre à la demande de ses produits ou services. Ernst et al. (2004a) font un état de l'art complet sur ce sujet. D'après l'article, la construction d'horaires de personnel passe en général par un nombre d'étapes clairement définies en l'occurrence : modélisation de la demande, affectation des séquences de repos et de jours de travail, planification des équipes, création des lignes de travail et finalement, affectation des tâches de travail et du personnel. Plusieurs étapes peuvent être combinées dans le processus de planification et leurs exigences dépendent des différents domaines d'application : les centres d'appels, les systèmes de santé, le domaine du transport, etc.

Le domaine d'application qui motive cette thèse est le transport urbain. Le processus de construction d'horaires de chauffeurs d'autobus passe par des étapes plus ou moins analogues aux étapes décrites par Ernst et al. (2004a). D'après Desaulniers et Hickman (2007), le processus complet de planification en transport urbain est très complexe, et il est divisé en plusieurs phases et étapes (voir la section 1). Ici, nous nous concentrons sur le problème de roulements de chauffeurs d'autobus qui est inclu dans la dernière phase de plannification : la phase opérationnelle. Cette dernière concerne principalement les problèmes d'horaires d'autobus, la construction de journées de travail et le problème de roulements des chauffeurs d'autobus.

Le problème d'horaires d'autobus constitue un élément très important dans la construction d'horaires de chauffeurs d'autobus. Il représente la première étape de la phase opérationnelle et vise à minimiser les coûts d'acquisition et d'opération des autobus. L'étape suivante

concerne la construction des journées de travail (*duty scheduling* en anglais). Elle est importante d'un point de vue économique, puisqu'elle détermine les salaires des chauffeurs d'autobus. Le problème de construction de journées de travail, comme son nom l'indique, vise à concevoir des journées de travail. Ce problème est très complexe et il est généralement formulé comme un problème de partitionnement d'ensemble avec contraintes additionnelles. Le problème de construction de journées de travail vise à minimiser le nombre de journées de travail ainsi que les coûts totaux. Une journée de travail est composée d'une ou de plusieurs pièces de travail séparées par des pauses (une pièce de travail contient des voyages consécutifs d'un point à un autre au long d'un segment de trajet de l'autobus). Chaque journée de travail générée doit se conformer aux règles de travail, par exemple : l'amplitude de la journée de travail, le temps maximum des pièces de travail sans pause, le temps minimum ou maximum des pauses, etc.

Les journées de travail représentent les données d'entrée du problème de roulements. Ce dernier est résolu lors de la dernière étape de la phase opérationnelle. Les journées de travail et les jours de repos sont affectés aux différents chauffeurs d'autobus, tout en respectant les règles de conventions collectives, ainsi que les préférences des chauffeurs. Les résultats de ce problème constituent les roulements des chauffeurs d'autobus.

2.2 Méthodes de résolution

Comme indiqué précédemment, le problème de roulements de personnel est présent dans plusieurs domaines d'application. Il est modélisé par différents modèles qui sont difficiles à résoudre à l'optimalité. Ainsi, tel que mentionné dans Ernst et al. (2004b) et Van den Bergh et al. (2013), il existe une panoplie de méthodes de résolution pour le problème de roulements. Dans cette section, nous présentons les approches les plus présentes et, à l'occasion, nous citons quelques articles qui sont spécifiques à ces méthodes de résolution.

2.2.1 Méthodes exactes de résolution

D'après Van den Bergh et al. (2013), les approches de programmation mathématique regroupent la plupart des méthodes de résolution exactes considérées. Le problème de roulements est souvent modélisé comme un programme linéaire mixte en nombre entiers. Dans cette section, nous présentons des méthodes exactes pour la résolution du problème de roulements.

Programmation linéaire en nombres entiers - *Branch-and-Bound* Un programme linéaire mixte (MIP) est formulé comme suit :

$$\min \quad c^T x + h^T y \tag{2.1}$$

$$\text{sujet à :} \quad Ax + Gy \leq b \tag{2.2}$$

$$x \in \mathbb{N}, y \geq 0 \tag{2.3}$$

tels que $x \in \mathbb{N}^n$ et $y \in \mathbb{R}^p$ sont deux vecteurs à n et p variables de décision respectivement, $c \in \mathbb{R}^n$ et $h \in \mathbb{R}^p$ sont deux vecteurs de coefficients de la fonction objectif, $b \in \mathbb{R}^m$ est un vecteur de scalaires et $A \in \mathbb{R}^{m*n}$ et $G \in \mathbb{R}^{m*p}$ sont deux matrices de scalaires. Si x est fixé, le problème est réduit à un programme linéaire (LP). Cependant, si $y = 0$, alors le problème est un programme en nombres entiers (pur PNE). Un cas particulier de programme en nombres entiers est lorsque la variable x est binaire. Dans ce cas, on parle de programme en nombres entiers binaires (PNE_{0-1}). Les problèmes MIP et PNE sont prouvés NP-difficiles (Wolsey, 1998), contrairement aux problèmes LP qui peuvent être résolus en temps pseudo-polynomial. Les méthodes les plus utilisées pour résoudre les problèmes LP sont les méthodes du simplexe (Dantzig et al., 1955) et les méthodes de points intérieurs (Karmarkar, 1984).

La plupart des MIP sont résolus par la méthode de *branch-and bound*. “Diviser pour régner“, telle est la devise dans Wolsey (1998). Il s'agit d'une méthode exacte de résolution. L'idée de cette méthode repose sur l'exploration récursive d'un arbre de branchement. La racine de l'arbre représente la relaxation linéaire du problème original. Une solution fractionnaire est généralement obtenue et le processus d'énumération implicite (appelé ici branchement ou *branching*) commence pour avoir une solution entière. Chaque décision de branchement est imposée dans un noeud de l'arbre. Si le branchement est exhaustif, alors la solution obtenue est optimale.

Les nœuds de l'arbre sont sélectionnés en se basant sur des stratégies d'exploration telles que profondeur d'abord, meilleur d'abord, ou d'autres. Durant l'exploration de l'arbre, les calculs des bornes inférieurs, obtenues en relâchant les contraintes d'intégralité, permettent d'accélérer le processus en élaguant les branches de l'arbre qui ne sont pas intéressantes de l'arbre.

Décomposition Dantzig-Wolfe et génération de colonnes. La méthode de décomposition de Dantzig-Wolfe peut être utile pour la résolution d'un PNE de grande taille. Le principe de cette décomposition repose sur la séparation des contraintes principales du problème en deux groupes. L'un des deux groupes doit avoir une structure particulière, telle que si les contraintes de l'autre groupe sont omises, le PNE devient plus facile à résoudre.

La génération de colonnes (GC) est une méthode basée sur la décomposition de Dantzig-

Wolfe. La GC considère un sous-ensemble de variables dans un problème-maître restreint (avec un premier groupe de contraintes) et ajoute progressivement des variables (colonnes) retournées par des sous-problèmes (contenant l'autre groupe de contraintes). Lorsque la GC est utilisée pour la résolution du problème LP dans un algorithme de *branch-and bound*, l'algorithme est appelé *branch-and-price*. La GC, dans ce cas, est utilisée pour résoudre chaque nœud de l'arbre de branchement.

La méthode de GC a été utilisée dans plusieurs travaux. Brunner et Edenharter (2011) ont proposé un modèle de programmation mixte (MIP) pour modéliser la demande de médecin dans un hôpital sur un horizon d'un an. La GC a été utilisée pour obtenir une solution entière optimale. Bard et Purnomo (2005a) ont utilisé la GC pour construire les horaires d'infirmières avec des préférences individuelles. Naudin et al. (2012) présentent trois modèles de roulements d'infirmières. Ces modèles sont résolus par la méthode de *branch-and-bound* et la méthode de *branch-and-price* où différentes stratégies de branchement ont été appliquées.

2.2.2 Métaheuristiques

Les métaheuristiques constituent une classe importante de méthodes de résolution les problèmes de roulements du personnel (Van den Bergh et al., 2013). Elles sont conçues pour traiter des problèmes d'optimisation complexes lorsque les autres méthodes d'optimisation exactes ne réussissent pas à le faire efficacement. L'avantage pratique des métaheuristiques réside dans la production de solutions réalisables dans un temps de calcul raisonnable. Cependant, l'utilisation des métaheuristiques ne donne aucune garantie sur l'optimalité des solutions produites. Les métaheuristiques sont souvent basées sur des algorithmes de recherche locale. Ce sont des méthodes générales pour résoudre des problèmes d'optimisation combinatoire. À partir d'une solution de départ, la recherche locale explore un voisinage de solutions candidates afin d'y trouver une solution améliorante. Le même processus se répète pour chaque itération de l'algorithme. La recherche s'arrête lorsqu'aucune solution améliorante ne peut être trouvée ou lorsque le temps imparti est dépassé.

Recuit simulé. Le recuit simulé (Kirkpatrick et al., 1983) est une méthode d'optimisation inspirée par le processus de refroidissement en métallurgie. C'est un algorithme qui empêche les solutions de rester bloquées dans un optimum local. Le recuit simulé permet des changements qui améliorent les solutions mais acceptent aussi de passer temporairement à des solutions de moindre bonne qualité dans le but d'élargir le domaine de recherche.

Peng et al. (2015) proposent une approche de recuit simulé à objectifs multiples pour concilier des objectifs contradictoires dans la recherche de solutions à un problème de roulements d'employés personnalisés. Dans le problème de roulements pour infirmières traité par Bailey

et al. (1997), des approches de recuit simulé et d'algorithmes génétiques (décrits ci-bas) ont été appliquées. Les résultats montrent qu'ils peuvent obtenir des solutions quasi-optimales dans un temps de calcul court. Xie (2013) a implémenté différentes métaheuristiques dont une approche de recuit simulé pour le problème de roulements. Cette approche était le meilleur choix pour résoudre le problème.

Algorithme génétique. Un algorithme génétique est une métaheuristique basée sur l'évolution des espèces biologiques. La génération des solutions et leurs améliorations sont réalisées en appliquant des opérateurs d'inspiration biologique : la sélection pour l'amélioration de solution, le croisement pour la simulation de la reproduction et la mutation pour le changement de solution (Barbosa, 2018). Dans Moz et Pato (2007), un algorithme génétique est utilisé pour résoudre le problème de roulements d'infirmières. L'algorithme traite le cas d'absence d'infirmières et génère de nouveaux roulements à partir d'un roulement original préexistant et ce, en en apportant des petites différences. Gröbner et Wilke (2001) proposent des algorithmes génétiques pour optimiser des problèmes généraux de roulements. Les solutions non réalisables sont corrigées en utilisant des opérateurs de correction.

Des problèmes de roulements ont été résolus également par des algorithmes mémétiques. Ces derniers sont une extension de l'algorithme génétique par l'utilisation de la recherche locale, leur but est d'améliorer les solutions lors des itérations génétiques. Un exemple d'application d'algorithme mémétique est présenté par Özcan (2005), qui traite le problème de roulements d'infirmières en utilisant différents types d'opérateurs génétiques et une méthode d'escalade hiérarchique auto-adaptative dirigée par violation (*self adaptive directed hierarchical hill climbing method*).

Méthodes hybrides. Dans Nurmi et al. (2011), un problème de roulements de chauffeurs d'autobus a été résolu par une méthode de recherche locale. Plusieurs procédures de recherche locale (recherche locale coopérative, algorithme évolutionnaire,...) coexistent dans le but d'éviter le blocage dans des solutions d'optimums locaux. Dans Monfroglio (1996), un algorithme génétique est utilisé pour optimiser les paramètres d'une heuristique glouton, qui est aussi hybridée avec un algorithme génétique pour optimiser des roulements dans une compagnie ferroviaire.

2.3 Problème de roulements de chauffeurs d'autobus

Comme décrit dans Desaulniers et Hickman (2007), la taille des instances réelles dans le transport urbain est très grande. En comparant avec le transport aérien et ferroviaire de longue distance, la notion de journée de travail (*duty* en transport urbain) est différente de

la notion de rotation (*pairing* en transport aérien). Ceci est dû au fait qu'une rotation peut s'étaler sur plusieurs jours, alors qu'une journée de travail est effectuée en un seul jour. Un grand nombre de journées de travail créées est considéré lors de la création des roulements. De plus, tous les chauffeurs sont capables de conduire tous les types de véhicules, contrairement au transport aérien, ce qui augmente la complexité du problème puisqu'il ne peut pas être décomposé par type de flotte. Les problèmes de roulements dans le transport ferroviaire peuvent s'étendre aux problèmes de roulements pour le transport urbain dans le cas où les trains urbains sont considérés et non pas les trains de longue distance.

2.3.1 Approche séquentielle de résolution

L'approche séquentielle vise à assigner les jours de repos et ensuite les quarts de travail ou les journées de travail. C'est l'approche la plus utilisée en transport urbain. Son efficacité a été démontrée par la génération rapide de solutions réalisables.

Affectation des jours de repos. L'affectation des jours de repos est l'étape qui définit les séquences de jours de repos et de jours de travail dans un horaire. D'après Bennett (1967), la construction de jours de repos est faite de façon à maximiser les jours de repos consécutifs tout en équilibrant le plus possible la distribution de repos entre les employés. Les jours de repos isolés ne sont pas souhaitables. L'auteur se base sur une routine qui calcule le nombre de séquences de repos consécutifs (4, 3 ou 2 repos), et qui distribue ensuite ces séquences de repos entre les employés d'une façon équitable. Les repos isolés restants sont distribués par la suite.

Dans Gendron (2012), le problème de construction des séquences de repos est traité tout en déterminant la taille des effectifs dans des horaires cycliques lorsqu'il s'agit de plusieurs groupes d'employés. Il a présenté deux modélisations : un modèle MIP et un modèle de flots.

Alfares (1998) propose aussi un MIP pour le problème de jours de repos dans un contexte particulier et présente une approche de résolution à deux étapes. Cette dernière consiste à inclure une contrainte, exprimant le nombre minimal d'employés. Le problème proposé a une formulation idéale donc toute solution du LP donne une solution entière à coût minimum.

Prakash et al. (1984) utilisent la programmation par objectif (*goal programming*) pour résoudre le problème multi-objectif de jours de repos. Deux jours de repos par semaine sont garantis pour chaque chauffeur. Les objectifs sont de limiter le nombre total d'employés et de minimiser le nombre d'employés ayant des jours de repos non consécutifs.

Un modèle de recouvrement d'ensemble est présenté pour le problème de construction de jours de repos dans Pedrosa et Constantino (2001), où le nombre total des employés est minimisé.

Dans ce problème, la génération de colonnes est utilisée avec un modèle de recouvrement d'ensemble comme problème-maître, et un problème de plus court chemin avec contraintes de ressources comme sous-problème.

Kyngäs et Nurmi (2011) ont utilisé une méthode de recherche locale pour résoudre le problème de jours de repos dans un contexte de roulement non cyclique. L'objectif est de minimiser les violations relatives aux contraintes souples concernant les préférences des chauffeurs, comme l'interdiction des jours de repos isolés tout en satisfaisant l'ensemble des contraintes strictes (e.g., exigences réglementaires et opérationnelles, comme le nombre maximum de jours de travail consécutifs). Le nombre de chauffeurs va jusqu'à 56 et l'horizon de planification est de 36 semaines. Nurmi et al. (2012) ont résolu le même problème en utilisant des métaheuristiques de recuit simulé et de recherche tabou pour minimiser le nombre de violations des contraintes souples.

Problèmes de roulements avec jours de repos fixés. Le problème de roulements avec jours de repos fixés attribue des journées de travail spécifiques à chaque chauffeur à chaque jour de travail dans un calendrier donné. Lorsqu'il s'agit de ce type de problèmes, la configuration des jours de repos présente la donnée d'entrée du problème de roulements.

Lezaun et al. (2006) considèrent un ensemble de patrons de repos déjà fixé pour le problème de roulements. Ils proposent un modèle en nombres entiers qui est résolu par étapes. Sur une base annuelle, les journées de travail de réserve sont d'abord attribuées. Ensuite, sur une base hebdomadaire, les séquences de jours de repos et de quarts de travail sont établis. Les patrons les plus intéressants sont sélectionnés et affectés finalement aux différents chauffeurs. Sodhi et Norris (2004) décomposent le problème de roulements en deux étapes : i) affectation des séquences de jours de repos et de quarts de travail, ii) affectation des journées de travail. Un modèle de réseau en nombres entiers est utilisé pour trouver un chemin cyclique optimal pour couvrir tout le roulement. Dans ce sens, Xie et Suhl (2015) présentent une heuristique à deux étapes. D'abord, l'affectation de jours de repos et de quarts de travail, ensuite, l'affectation des journées de travail. Un problème de réseau multi-flots est formulé pour la résolution de la première étape, ensuite un MIP est résolu pour la deuxième étape afin d'assigner les journées de travail aux quarts de travail sélectionnés dans la première étape. Dans Carraresi et Gallo (1984), le problème de roulements avec affectation de quarts de travail est modélisé comme un problème d'affectation avec goulot d'étranglement (*bottelneck assignment*) à plusieurs niveaux, et ce, en minimisant la charge de travail maximale des chauffeurs. Une heuristique a été utilisée pour produire des solutions asymptotiquement optimales. En utilisant la même modélisation de goulot d'étranglement, Bianco et al. (1992) résolvent à l'aide d'une heuristique itérative le problème d'affectation des journées de travail. L'algorithme vise à trouver

une combinaison de roulements à coût minimum, tout en minimisant la charge de travail maximale. À chaque itération, les roulements hebdomadaires sont générés sur l'ensemble des journées de travail non affectées. L'algorithme prend fin si aucune journée de travail n'est disponible ou s'il devient impossible de générer des roulements.

Quant à Hartog et al. (2009), ils commencent par l'attribution des ensembles de journées de travail à chaque groupe de chauffeurs en utilisant un MIP. La construction de roulements passe par deux étapes, d'abord, l'affectation des types de journées de travail (chaque journée de travail appartient à un type : tôt, tard ou nuit) et de jours de repos à chaque groupe de chauffeurs, ensuite l'affectation des journées de travail spécifiques à chaque chauffeur.

2.3.2 Approche integrée

L'approche intégrée est moins utilisée dans le transport urbain. Différentes méthodes de résolution ont été utilisées dans divers domaines d'application. Ernst et al. (1998) présentent une approche de recuit simulé pour résoudre le problème de roulements cyclique dans le domaine ferroviaire. Ils construisent les roulements à partir des voyages de train en fonction du nombre de conducteurs dans chaque dépôt. Caprara et al. (1998) introduisent un système qui se base sur un algorithme heuristique pour trouver des roulements avec un nombre optimal de conducteurs. Une borne lagrangienne inférieure est décrite en se basant sur un graphe et est utilisée pour améliorer l'efficacité de l'algorithme.

Un MIP a été présenté par Xie et Suhl (2015). Ils y ont proposé une approche intégrée qui consiste à résoudre le problème en combinant deux étapes de l'approche heuristique mentionnées dans l'approche séquentielle (Section 2.3.1). Les auteurs ont testé cette approche seulement sur une instance de problème cyclique incluant 9 groupes d'employés, 11 types de quarts de travail et 1013 journées de travail avec 12% d'activité pré-assignées.

Dans un contexte métaheuristique, Moz et al. (2009) ont proposé des heuristiques évolutives pour résoudre le problème bi-objectif non cyclique de roulements. La charge de travail supplémentaire maximale est minimisée ainsi que le nombre de chauffeurs par période de roulements. Mesquita et al. (2011) utilisent une matheuristique basée sur la programmation en nombres entiers dans le même objectif de minimisation de la charge de travail supplémentaire et le nombre de chauffeurs. Les solutions mensuelles obtenues sont quasi-optimales ou optimales et comportent jusqu'à 70 chauffeurs avec 1400 journées de travail. Dans Borndörfer et al. (2015), le problème de roulements a été résolu avec une méthode de recherche locale, avec un objectif de minimisation des coûts de roulements et aussi les pénalités relatives aux violations de préférence des chauffeurs.

Nishi et al. (2014) ont développé un algorithme de décomposition de Benders dans lequel il est supposé que le même ensemble de journées de travail soit effectué chaque jour et que toutes les copies de la même journée de travail doivent être affectées au même roulement. Dans cette approche, le problème-maître de Benders vise à assigner les journées de travail aux roulements et à déterminer la durée de chaque roulement qui ne peut pas être un multiple de sept jours. Les sous-problèmes de Benders sont des problèmes de faisabilité qui déterminent la position de chaque journée de travail et chaque jour de repos dans chaque roulement. Les auteurs ont pu résoudre, à l'optimalité, seulement des instances de petite taille (15 journées de travail par jour).

2.3.3 Problème de roulements avec préférences d'employés

Au cours des dernières années, considérer les préférences d'employés est devenu un aspect important dans la construction de roulements. Hanne et al. (2009) présentent les avantages de l'affectation d'horaires flexibles aux employés. Ils expliquent comment les préférences peuvent réduire l'absentéisme et améliorer le moral, ce qui est traduit par moins de coûts opérationnels et moins de ré-optimisations.

Plusieurs articles ont abordé le problème de roulements en considérant les préférences d'employés, en particulier pour les horaires d'infirmières. Dans ce domaine d'application, les préférences sont généralement exprimées sous la forme d'activités désirables ou au contraire comme des séquences indésirables d'activités. Certains auteurs Knust et Xie (2019), Bard et Purnomo (2005a) et Bard et Purnomo (2005b) cherchent à minimiser le nombre de violations des préférences dans la fonction objectif. La non-satisfaction d'une préférence correspond à la présence d'une séquence non désirable, dans laquelle deux jours de repos sont séparés par un jour de travail isolé. Ce problème est résolu par différentes approches : Knust et Xie (2019) utilisent la formulation d'un réseau de flot et le recuit simulé, tandis que Bard et Purnomo (2005a) et Bard et Purnomo (2005b) ont recours à un programme en nombres entiers et la génération de colonnes.

Dans le même secteur d'application, Bard et Purnomo (2007) et Purnomo et Bard (2007) ajoutent aux préférences précédentes les séquences d'activités indésirables (e.g. séquence jour-soir-jour, sans un jour de repos intermédiaire). Dans les deux articles, l'objectif est de minimiser les pénalités relatives aux contraintes souples (les préférences d'employés). Bard et Purnomo (2007) résolvent le problème en utilisant une approche de fixation de variables au lieu du *Branch-and-bound* traditionnel. D'autre part, Purnomo et Bard (2007) utilisent un algorithme de branchement combiné avec une décomposition de Dantzig-Wolfe.

Une autre façon de traiter les préférences pour la planification des horaires d'infirmières est

présentée dans Wong et al. (2014), où les auteurs sélectionnent les préférences les plus importantes et les plus demandées (par exemple : trois quarts de soir consécutifs au maximum) et les considèrent comme des contraintes dures tout en pénalisant la non-satisfaction des autres préférences dans la fonction objectif (c'est-à-dire : les traiter comme des contraintes souples). Les roulements sont ensuite construits à l'aide d'une heuristique de recherche locale.

Dans d'autres domaines, les préférences consistant à interdire certaines séquences de travail ou de rotations sont considérées. Par exemple, Borndörfer et al. (2015) pénalisent les rotations à rebours (lorsque la journée de travail au jour $j + 1$ commence plus tôt que la journée de travail au jour j) dans un contexte de contrôle de péages. Le problème de roulements est résolu en utilisant la programmation en nombres entiers.

Dans notre contexte, nous nous concentrons sur le problème de roulements dans le transport urbain. Dans le transport ferroviaire et transport en métro, il est courant de prendre en compte les préférences des conducteurs lors de la construction des roulements. Par exemple Hartog et al. (2009) préconisent les jours de repos adjacents et l'affectation des journées de travail de même type dans la même séquence de travail. Suivant le même principe, Sodhi et Norris (2004) interdisent les heures supplémentaires excessives et préfèrent des horaires dans lesquels les journées de travail précoces commencent progressivement plus tard, et les journées de travail tardives se terminent progressivement plus tôt, au fur et à mesure qu'on avance dans la semaine. Dans les deux travaux de recherche, la programmation en nombre entiers est utilisée pour résoudre le problème.

Dans un autre exemple en transport ferroviaire, Jütte et al. (2017) pénalisent l'affectation des journées de travail indésirables dans la fonction objectif et résolvent le modèle résultant avec un algorithme basé sur la génération de colonnes.

Lors de la construction de roulements, en transport urbain, les quarts ou les journées de travail indésirables sont généralement pondérés et pénalisés dans la fonction objectif. Cette approche est utilisée par exemple dans Kisielewski et al. (2019), Lezaun et al. (2006), Mesquita et al. (2015), Nurmi et al. (2011) et Xie et Suhl (2015), où les auteurs cherchent à minimiser le nombre de séquences de journées de travail indésirables ou à les répartir équitablement entre les chauffeurs. Les méthodes utilisées sont variables, allant de la programmation en nombres entiers aux algorithmes génétiques. Par exemple, Nurmi et al. (2011) utilisent une heuristique évolutive pour résoudre le problème. Dans leur modèle, en plus d'éviter l'affectation des jours de repos et des jours de travail isolés, ils permettent aux chauffeurs de choisir s'ils préfèrent ou non avoir le même quart de travail avant et après une séquence de jours de repos. Moz et al. (2009) utilisent également une heuristique évolutive pour résoudre un problème de roulements dans lequel ils visent à minimiser le nombre maximum d'heures supplémentaires

par chauffeur ainsi que le nombre total de chauffeurs dont la charge de travail est incomplète. Dans leur approche, les préférences ne sont pas pénalisées, mais elle sont plutôt exprimées implicitement dans la fonction objectif.

2.4 Sommaire

Dans l'état de l'art fait, nous avons décrit les différents modèles mathématiques et les approches de résolution utilisés pour résoudre les problèmes de roulements d'employés, en particulier, les problèmes de roulements des chauffeurs d'autobus.

Afin de faire face aux règles complexes du problème de roulements des chauffeurs d'autobus, les planificateurs optent pour la décomposition du problème pour en réduire sa complexité et le résoudre d'une façon séquentielle. À notre connaissance, aucun algorithme exact n'a été proposé pour résoudre le problème général de roulements cycliques avec des instances de grande taille, et sans faire appel aux types de journées de travail ou aux considérations des quarts de travail dans l'optimisation. En plus de la réduction des coûts, qui est un objectif implicite de l'optimisation du problème de roulements, les préférences des chauffeurs d'autobus sont considérées comme importantes. Plusieurs publications traitent les préférences d'employés dans les roulements. Toutefois, la plupart des problèmes proposés et des approches utilisées traitent les quarts de travail et les activités désirables ou indésirables d'employés.

L'objectif principal de cette thèse est d'étudier et de développer des approches de résolution séquentielle et intégrée du problème de roulements des chauffeurs d'autobus tout en équilibrant les charges de travail entre les employés. Nous présentons aussi une variante du problème de roulements qui prend en considération les préférences de chauffeurs d'autobus pour obtenir non seulement des roulements bien équilibrés mais qui répondent le plus aux préférences des chauffeurs.

CHAPITRE 3 ORGANISATION DU TRAVAIL

Cette thèse traite essentiellement l'optimisation des roulements des chauffeurs d'autobus. Dans la revue de littérature, nous avons décrit l'état de l'art concernant les différentes méthodes utilisées pour résoudre les problèmes de roulements. Nous avons aussi pu conclure que les problèmes de roulements en transport urbain sont peu traités en comparaison à ceux en transport aérien et ferroviaire. La plupart des processus de construction de roulements commencent par l'attribution des types ou des quarts de journées de travail aux chauffeurs et ensuite l'affectation des journées de travail. L'affectation directe des journées de travail aux chauffeurs est rarement proposée, et les approches de résolution se basent la plupart du temps sur la relaxation de certaines contraintes. Ceci aide à réduire la complexité des problèmes certes, mais peut produire des solutions de piètre qualité. À notre connaissance, aucun algorithme exact n'a été proposé pour la résolution des problèmes de roulements de grande taille et sans l'aide de la décomposition par quart de travail.

Le chapitre 4 propose une méthode exacte et des formulations efficaces pour les problèmes de roulements des chauffeurs d'autobus avec jours de repos fixés, et ce dans deux contextes cycliques différents : un contexte relatif à une grande compagnie de transport urbain traitant plusieurs groupes de chauffeurs à la fois, et un contexte relatif à une compagnie de taille moyenne traitant un seul groupe de chauffeurs. Les deux contextes traitent l'affectation des journées de travail en respectant plusieurs règles de repos de nuit, des règles qui s'étendent sur plusieurs jours (notamment le premier contexte) et des règles sur les types des journées de travail (notamment le deuxième contexte). Les journées de travail sont affectées en équilibrant le plus possible la charge de travail hebdomadaire entre les employés. Notons que la résolution des problèmes dans les deux contextes a été faite en respectant toutes les règles, aucun problème n'a été résolu en relaxant des contraintes ni en ré-optimisant des solutions. L'efficacité des modèles a été démontrée pour des instances comportant jusqu'à 202 chauffeurs et les solutions obtenues ont été utiles pour le traitement des deux autres objectifs de la thèse.

Le deuxième objectif de la thèse (chapitre 5) porte sur le problème de roulements intégré d'affectation de jours de repos et des journées de travail. Le problème a été modélisé comme un programme linéaire mixte en nombres entiers, qui peut être résolu en utilisant un solveur commercial MILP comme CPLEX. Avec ce modèle, nous avons pu obtenir des solutions optimales pour des instances ayant jusqu'à 146 chauffeurs. Nous proposons une matheuristique à deux étapes, qui est une approche modifiée de la résolution séquentielle de ce type

de problème. L'approche consiste, d'abord, en l'affectation simultanée des jours de repos et des journées de travail tout en équilibrant la charge de travail hebdomadaire entre les chauffeurs, mais en relaxant les contraintes relatives à l'affectation des journées de travail (à savoir les règles de repos de nuit,...). Évidemment, les contraintes de couverture des journées de travail sont toujours présentes. Cette procédure a permis de réduire la symétrie du modèle intégré, puisque les jours de repos sont devenus fixes. La solution obtenue est présentée par une configuration optimale de jours de repos. Les journées de travail sont ensuite affectées en utilisant le modèle formulé dans le chapitre 4. La troisième approche utilisée est la résolution du problème intégré en fournissant une solution initiale (provenant de la matheuristique) pour résoudre à l'optimalité toutes les instances notamment les plus grandes. Le démarrage à chaud (*warm start*) montre que les temps de calcul peuvent être réduits significativement, comparés à la résolution directe par le solveur MILP.

Le dernier objectif de la thèse, présenté dans le chapitre 6, traite les préférences des chauffeurs d'autobus. Nous proposons une extension du problème de roulements avec repos fixés (chapitre 4) qui vise à ré-optimiser la solution obtenue dans le deuxième objectif (chapitre 5) en prenant en considération les préférences des chauffeurs. Nous nous intéressons aux préférences relatives aux temps de début des journées de travail. Ces nouvelles règles sont formulées comme des contraintes souples, et la fonction objectif vise à réduire le nombre de contraintes violées. Une contrainte qui vise à limiter le budget de dépassement du coût original (excès des charges de travail hebdomadaire par rapport à la moyenne) a été ajoutée. Le problème devient plus complexe et la résolution par un solveur MILP prend beaucoup de temps de calcul (plus de 9 heures de calcul pour fournir une première solution entière pour une instance de 165 chauffeurs), surtout pour les grandes instances. Un algorithme de décomposition basé sur un modèle de partitionnement a permis la décomposition des grands groupes (*rosters*) de chauffeurs en plusieurs sous-groupes de différentes tailles. La solution obtenue avec le modèle du premier objectif (chapitre 4) a permis de restreindre les possibilités d'affectation des journées de travail, ce qui réduit significativement les temps de calcul et le nombre des contraintes souples violées. Les roulements obtenus sont de bonne qualité, bien équilibrés et répondent à la plupart des préférences des chauffeurs d'autobus.

CHAPITRE 4 PROBLÈME DE ROULEMENTS DE CHAUFFEURS D'AUTOBUS AVEC JOURS DE REPOS FIXÉS

Ce chapitre présente le premier objectif de la thèse. Nous y proposons deux variantes du problème d'affectation des journées de travail avec jours de repos fixés. Ces deux variantes sont différentes par leurs règles d'affectation de journées de travail et par la taille de leurs instances. Nous traitons des instances réelles de deux compagnies de transport urbain. Nous présentons d'abord les notions générales et les éléments essentiels à la construction d'un horaire de chauffeurs d'autobus, avant d'introduire le problème d'affectation des journées de travail et d'expliquer les différentes règles le régissant. Nous développons ensuite les modèles mathématiques. Finalement, nous présentons les résultats numériques obtenus et les comparons avec ceux de notre partenaire industriel GIRO. Ce dernier est un leader mondial dans le développement et la commercialisation de logiciels d'optimisation pour les sociétés de transport urbain.

4.1 Introduction

Nous considérons le problème cyclique de construction d'horaires de chauffeurs d'autobus. Ce problème est défini sur un horizon d'une semaine et peut être résolu en deux sous-étapes opérationnelles : d'abord, affecter les jours de repos ; ensuite, affecter les journées de travail. Dans notre contexte, nous travaillons sur des horaires avec des jours de repos fixés a priori. Par ailleurs, cet objectif de la thèse sera utile aux (chapitre 5 et chapitre 6) pour la résolution du problème intégré d'affectation de journées de travail et de jours de repos.

Une journée de travail se définit comme une séquence de voyages actifs séparés par des pauses et des voyages de positionnement. Chaque journée de travail a une heure de début, une heure de fin, une charge de travail (temps rémunéré) et contient, en général, des voyages d'une ou deux lignes d'autobus. D'autre part, une journée de réserve reprend les mêmes éléments d'une journée de travail mais sert à remplir les jours pour lesquels nous n'arrivons pas à affecter une journée de travail ou un jour de repos. Les journées de réserve sont aussi utiles pour faire face aux cas d'absences d'employés avec des journées de travail. Dans un deuxième cas étudié dans ce chapitre et, en plus des éléments cités, nous évoquons la notion de types de journées de travail. Ces derniers prennent la forme de types de journées de travail : "très tôt", "tôt", "jour", "midi" et "soir".

Nous retrouvons ensuite la notion de position qui est définie comme l'horaire d'un chauffeur

durant une semaine, c'est-à-dire, une succession de journées de travail et de jours de repos. La construction d'un tel horaire est régie par un ensemble de règles qui dictent l'affectation des journées de travail et de repos.

Enfin, un roulement est un concept à double sens : il réfère à la fois à un groupe d'employés et aussi à l'horaire de ces derniers. Chaque roulement est construit suivant un ensemble de règles. Il réunit alors un nombre d'employés partageant les mêmes préférences, les mêmes caractéristiques de journées de travail (par exemple, des journées longues ou courtes de travail) ou tout simplement le même nombre de jours de repos souhaités. Par exemple, les chauffeurs souhaitant avoir plusieurs jours de repos par semaine et effectuant de longues journées de travail sont regroupés dans le même roulement. Le nombre d'employés dans un roulement définit le nombre de positions et la taille du cycle de ce roulement. Un cycle représente le nombre de semaines sur lesquelles les chauffeurs font des rotations. Si, par exemple, la taille du cycle vaut 4, alors le roulement contient 4 chauffeurs qui exécutent 4 positions à tour de rôle.

Le tableau 4.1 présente un exemple de deux roulements : l'un avec 4 positions et l'autre avec 3 positions. Les cases avec R définissent les jours de repos et celles avec D_i ($i = 1, \ldots, 9$) représentent les journées de travail. Nous remarquons que la même journée de travail peut apparaître plusieurs fois durant le même roulement. Dans cet exemple, nous remarquons aussi que les deux roulements sont différents, l'un contient deux jours de repos par semaine et l'autre trois jours de repos. Ils diffèrent également par la taille des cycles.

Tableau 4.1 Roulements avec deux cycles différents

Roulement	Position	Lun	Mar	Mer	Jeu	Ven	Sam	Dim
1	1	D_1	D_7	R	R	D_2	D_3	D_4
	2	D_2	D_8	D_1	R	R	D_4	D_5
	3	D_3	D_9	D_2	D_6	R	R	D_7
	4	D_4	D_2	D_3	D_7	D_8	R	R
2	1	R	R	D_4	D_8	D_9	D_5	R
	2	D_5	R	R	D_9	D_1	R	D_8
	3	D_6	D_1	D_5	R	R	R	D_9

Le problème de roulements dans cette thèse présente un cas particulier où les journées de travail sont différentes d'un jour à l'autre, elles sont affectées à plusieurs positions à la fois et l'objectif est de répartir équitablement la charge de travail entre les chauffeurs. Dans la première semaine, chaque chauffeur est affecté à une position, et la semaine suivante, le chauffeur sera assigné à la position suivante et ainsi de suite. Le chauffeur assigné à la dernière

position dans la première semaine sera attribué à la première position la semaine suivante, ce qui donne lieu à un roulement cyclique où tous les chauffeurs du même groupe ont la même charge de travail à la fin du cycle.
Dans le tableau 4.2, nous présentons un roulement cyclique avec 4 employés lorsque nous déroulons l'horaire. La taille du cycle tel que présenté est de 4.

Tableau 4.2 Roulement cyclique sur 4 semaines

Employé	Semaine 1	Semaine 2	Semaine 3	Semaine 4
Chauffeur 1	Position 1	Position 2	Position 3	Position 4
Chauffeur 2	Position 2	Position 3	Position 4	Position 1
Chauffeur 3	Position 3	Position 4	Position 1	Position 2
Chauffeur 4	Position 4	Position 1	Position 2	Position 3

4.2 Règles d'affectation

Le problème cyclique de roulements de journées de travail avec jours de repos fixés ($RCRF$) est défini sur un horizon d'une semaine et vise à affecter les journées de travail aux différentes positions. Le problème est modélisé comme un programme en nombres entiers. Dans ce chapitre, nous traitons des données réelles provenant de deux compagnies de transport urbain européennes.

Les roulements doivent se conformer aux règles de sécurité et aux règles de conventions collectives qui définissent des contraintes strictes, telles que la durée de repos minimale entre deux journées de travail pendant une semaine, le nombre de jours de travail consécutifs qui incluent des voyages d'autobus sur une même ligne, etc.

Le RCRF tient compte d'un ensemble de contraintes dures et cherche à équilibrer le plus possible la charge de travail entre tous les employés. Cela se traduit par la minimisation de la somme des déviations positives par rapport à la moyenne des charges de travail totales par semaine. En minimisant la somme de ces écarts positifs, la somme des écarts négatifs est minimisée aussi.

La configuration de jours de repos a été fournie par notre partenaire industriel GIRO ; le problème d'affectation de jours de repos y est modélisé comme un programme linéaire mixte (Gendron, 2012) et résolu par un solveur commercial MILP. Le résultat présente les données d'entrée du problème de roulement en question. Le problème d'affectation des jours de repos est résolu généralement en respectant un ensemble de contraintes, notamment le nombre de jours de repos par semaine, le nombre maximum de jours de travail consécutifs, etc.

Pour la première variante du problème de roulements cycliques avec jours de repos fixés A ($RCRF$-A), les jours de repos ont été affectés en respectant les contraintes suivantes :

- Un maximum de 6 jours de travail consécutifs ;
- Au moins 2 jours de repos consécutifs par semaine (Repos double - étant donné la nature cyclique du problème, les repos Dimanche-Lundi sont considérés comme un repos double mais comptent seulement pour une semaine) ;
- 2 jours de repos par semaine pour le type de roulements 1 et 3 jours de repos par semaine pour le type de roulements 2.

Pour la deuxième variante du problème de roulements cycliques avec jours de repos fixé Y ($RCRF$-Y), les jours de repos ont été affectés en respectant les contraintes suivantes :

- Au moins 2 jours de repos par semaine ;
- Entre 9 et 10 jours de repos par période de 4 semaines, soit des semaines 1 à 4, 5 à 8, 9 à 12, etc.
- Au moins 2 jours de repos consécutifs par semaine.

Rappelons que ce chapitre ne traite pas la partie d'affectation des jours de repos. Donc nous nous intéressons seulement aux règles d'affectation des journées de travail pour modéliser les problèmes. Pour le problème $RCRF$-A, les journées de travail doivent être attribuées en respectant les règles suivantes :

- Il doit y avoir au moins 9 heures de repos entre deux journées de travail consécutives ;
- Il doit y avoir au moins 10 heures de repos entre deux journées de travail consécutives si elles sont immédiatement suivies par une troisième journée de travail ;
- Une période (libre) de repos d'au moins 57 heures dont au moins deux jours de repos consécutifs doit être attribuée à chaque position.
- Pour toute période de 28 jours consécutifs finissant par une journée de travail, le temps de repos moyen entre les journées de travail doit être d'au moins 12 heures ;
- Au plus une journée de travail d'amplitude de 13 heures ou plus peut être affectée à chaque position. L'amplitude d'une journée de travail est égale à la durée entre son temps de début et son temps de fin ;
- Au plus deux journées de travail desservant la même ligne d'autobus peuvent être affectées consécutivement.

Tandis que pour le problème $RCRF$-Y, les journées de travail doivent être attribuées en respectant les règles suivantes :

- Il doit y avoir au moins 10 heures de repos entre deux journées de travail consécutives.
- Une période (libre) de repos d'au moins 58 heures dont au moins deux jours de repos

consécutifs doit être attribuée à chaque position.
— Au plus deux types de journées de travail peuvent être attribués dans la même séquence de travail.
— Au maximum 5 heures d'écart entre les heures de fin minimum et maximum des journées de travail dans la même séquence de travail.

4.3 Formulations mathématiques

Dans cette partie, nous présentons les modèles mathématiques des problèmes *RCRF-A* et *RCRF-Y*. Les deux problèmes sont modélisés comme des programmes linéaires en nombres entiers et l'affectation des journées de travail est effectuée en exploitant la connaissance des jours de repos.

Étant donné la nature cyclique du modèle proposé, nous avons besoin d'identifier les paires de position et jour (p^*, j^*) atteintes en se déplaçant de k jours en avant ou en arrière à partir de la paire (p, j) dans le cycle (par exemple, si nous voulons imposer au plus 6 jours de travail consécutifs, il faut identifier à partir d'une paire de position et jour (p, j) les 5 jours suivants, c-à-d, pour k= $1, 2, ..., 5$). Pour ce faire, définissons d'abord, les deux éléments suivants :

$r(p)$: Roulement de la position p ;
P_r : Ensemble des positions du roulement r.

Notons que les positions d'un roulement r sont numérotées de 0 à $|P_r - 1|$ et les jours de la semaine sont numérotés de 0 à 6.

Donc, par exemple, pour un roulement de 4 positions $|P_r| = 4$, (p^*, j^*) = $(3, 6)$ si (p, j) = $(4, 0)$ et $k = -1$ et (p^*, j^*) = $(0, 2)$ si $(p, j) = (4, 5)$ et $k = 4$. Pour cette raison, nous introduisons une fonction vectorielle f :

$$f(p, j, k) = \left(\left\lfloor p + \frac{(j + k)}{7} \right\rfloor \bmod \left| P_{r(p)} \right| \ , \ (j + k) \bmod 7 \right), \tag{4.1}$$

où la fonction modulo est définie par : $a \mod n = a - n \left\lfloor \frac{a}{n} \right\rfloor$.

Nous utilisons la notation suivante pour définir les différents éléments des deux modèles.

4.3.1 Ensembles

$\mathbf{J}$: Ensemble des jours de la semaine, du lundi au dimanche ;
$\mathbf{D}_j^R$: Ensemble des journées de travail qui doivent être assignées au jour j ;
$\mathbf{D}_j^S$: Ensemble des journées de réserve qui peuvent être assignées au jour j ;

- $\mathbf{R}$: Ensemble des roulements ;
- $\mathbf{R}_d$: Ensemble des roulements auxquels on peut assigner la journée de travail d ;
- $\mathbf{P}$: Ensemble de toutes les positions dans tous les roulements ;
- $\mathbf{P}_r$: Ensemble des positions dans le roulement r ;
- $\mathbf{D}_{pj}^R$: Ensemble des journées de travail qui peuvent être assignées à la position p le jour j ;
- $\mathbf{D}_{pj}$: Ensemble des journées de travail et journées de réserve $\mathbf{D}_{pj}^R \cup \mathbf{D}_j^S$ qui peuvent être assignées à la position p le jour j ;
- $\mathbf{L}$: Ensemble des lignes d'autobus ;
- $\mathbf{L}_d$: Ensemble des lignes d'autobus qui couvrent la journée de travail d ;
- $\mathbf{J}_p^W$: Ensemble des jours de travail dans la position p ;
- $\mathbf{J}_p^9$: Ensemble des jours de travail dans la position p qui sont immédiatement suivis par un jour de travail et ensuite un jour de repos ;
- $\mathbf{J}_p^{10}$: Ensemble des jours de travail dans la position p qui sont suivis immédiatement par au moins deux jours de travail ;
- $\mathbf{J}_p^{57}$: Ensemble des jours simples de travail dans la position p qui sont suivis par deux jours de repos (repos double) ;
- $\mathbf{J}_p^{12}$: Ensemble des jours dans la position p qui commencent une séquence de 28 jours finissant par un jour de travail ;
- $\mathbf{K}_{pj}$: Sous-ensemble de $\{0, 1, \ldots, 27\}$ indiquant les paires de jours de travail consécutifs dans une séquence de 28 jours commençant par la paire position-jour (p, j), c'est-à-dire, $k \in K_{pj}$ si les jours de travail sont assignés aux deux paires $f(p, j, k)$ et $f(p, j, k+1)$;
- $\mathbf{T}_{pj}$: Ensemble de tous les temps de fin des journées de travail dans $\mathbf{D}_{pj}$.

En plus des ensembles précédents, nous ajoutons les ensembles suivants relatifs au problème *RCRF-Y* :

- $\mathbf{T}$: Ensemble de tous les types de journées de travail ;
- $\mathbf{D}_a$: Ensemble des journées de travail de type $a \in \mathbf{T}$;
- $\mathbf{Y}_p^{10}$: Ensemble de jours de travail dans la position p qui sont suivis immédiatement par un jour de travail ;
- $\mathbf{Y}_p^{58}$: Ensemble de jours simples de travail dans la position p qui sont suivis par deux jours de repos (repos double) ;
- $\mathbf{S}$: Ensemble des séquences s de jours de travail. Le début de chaque séquence s est désigné par une paire (p^s, j^s) de position-jour. Chaque séquence doit être précédée et suivie par un jour de repos ;
- $\mathbf{Y}^s$: Ensemble des paires (p, j) de position-jour dans la séquence de travail s.

4.3.2 Paramètres

W_d : Charge de travail de la journée de travail d (en minutes) ;

S_d : Temps de début de la journée de travail d (en minutes) ;

E_d : Temps de fin de la journée de travail d (en minutes) ;

I_d^{13} : Indicateur binaire égal à 1 si l'amplitude de la journée de travail d dépasse 13 heures ;

I_{dl} : Indicateur binaire égal à 1 si la journée de travail d dessert la ligne d'autobus l, c-à-d, si $l \in \mathbf{L}_d$;

A : Moyenne de charge de travail par position $\left(A = \dfrac{\sum_{j \in \mathbf{J}} \sum_{d \in \mathbf{D}_j^R} W_d}{\sum_{r \in \mathbf{R}} |P_r|} \right)$;

F_{dt}^{+} : Indicateur binaire égal à 1 si la journée de travail d termine au temps t ou après ;

B_{dt}^{-} : Indicateur binaire égal à 1 si la journée de travail d commence avant le temps t (excluant le temps t) ;

U_{pj} : Nombre total des jours de travail qui sont immédiatement suivis par un jour de travail dans une séquence de 28 jours commençant par la paire position-jour (p, j).

En plus des paramètres précédents, nous ajoutons les paramètres suivants relatifs au problème *RCRF-Y* :

K^s : nombre de jours dans la séquence de travail s.

4.3.3 Variables

x_{pj}^d : Variable binaire égale à 1 si la journée de travail d est assignée à la position p le jour j, et 0 sinon ;

z_p : Variable positive égale à la charge de travail excédentaire de la position p par rapport à la charge de travail moyenne par position c-à-d, $z_p = \max\{0, \sum_{j \in \mathbf{J}} \sum_{d \in \mathbf{D}_{pj}^R} W_d x_{pj}^d - A\}$.

En plus des variables précédentes, nous ajoutons les variables suivantes au problème *RCRF-Y* :

y_a^s : Variable binaire égale à 1 si la séquence $s \in \mathbf{S}$ contient une journée de travail de type a, et 0 sinon.

4.3.4 Modèles mathématiques

Problème RCRF-A : Le modèle mathématique du problème de roulement est formulé comme suit :

$$\min \quad \sum_{p \in \mathbf{P}} z_p \tag{4.2}$$

$$\text{sujet à :} \quad \sum_{j \in \mathbf{J}_p^W} \sum_{d \in \mathbf{D}_{pj}^R} W_d x_{pj}^d - A \leq z_p \qquad \forall p \in \mathbf{P} \tag{4.3}$$

$$z_p \geq 0 \qquad \forall p \in \mathbf{P} \tag{4.4}$$

$$\sum_{r \in \mathbf{R}_d} \sum_{p \in \mathbf{P}_r} x_{pj}^d = 1 \qquad \forall j \in \mathbf{J},\, d \in \mathbf{D}_j^R \tag{4.5}$$

$$\sum_{d \in \mathbf{D}_{pj}} x_{pj}^d = 1 \qquad \forall p \in P,\, j \in \mathbf{J}_p^W \tag{4.6}$$

$$\sum_{d \in \mathbf{D}_{pj}} F_{dt}^+ x_{pj}^d + \sum_{d \in \mathbf{D}_{f(p,j,1)}} B_{d,t+540}^- x_{f(p,j,1)}^d \leq 1 \qquad \forall p \in \mathbf{P},\, j \in \mathbf{J}_p^9, t \in \mathbf{T}_{pj} \tag{4.7}$$

$$\sum_{d \in \mathbf{D}_{pj}} F_{dt}^+ x_{pj}^d + \sum_{d \in \mathbf{D}_{f(p,j,1)}} B_{d,t+600}^- x_{f(p,j,1)}^d \leq 1 \qquad \forall p \in \mathbf{P},\, j \in \mathbf{J}_p^{10},\, t \in \mathbf{T}_{pj} \tag{4.8}$$

$$\sum_{d \in \mathbf{D}_{pj}} F_{dt}^+ x_{pj}^d + \sum_{d \in \mathbf{D}_{f(p,j,3)}} B_{d,t+3420}^- x_{f(p,j,3)}^d \leq 1 \qquad \forall p \in \mathbf{P},\, j \in \mathbf{J}_p^{57},\, t \in \mathbf{T}_{pj} \tag{4.9}$$

$$\sum_{k \in \mathbf{K}_{pj}} \left(\sum_{d \in \mathbf{D}_{f(p,j,k+1)}} S_d x_{f(p,j,k+1)}^d - \sum_{d \in \mathbf{D}_{f(p,j,k)}} E_d x_{f(p,j,k)}^d \right) \geq 720 U_{pj} \quad \forall p \in \mathbf{P},\, j \in \mathbf{J}_p^{12} \tag{4.10}$$

$$\sum_{j \in \mathbf{J}_p^W} \sum_{d \in \mathbf{D}_{pj}} I_d^{13} x_{pj}^d \leq 1 \qquad \forall p \in \mathbf{P} \tag{4.11}$$

$$\sum_{d \in \mathbf{D}_{pj}} I_{dl} x_{pj}^d + \sum_{d \in \mathbf{D}_{f(p,j,1)}} I_{dl} x_{f(p,j,1)}^d + \sum_{d \in \mathbf{D}_{f(p,j,2)}} I_{dl} x_{f(p,j,2)}^d \leq 2 \qquad \forall p \in \mathbf{P},\, j \in \mathbf{J}_p^{10},\, l \in \mathbf{L} \tag{4.12}$$

$$x_{pj}^d \in \{0,1\} \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}_p^W, d \in \mathbf{D}_{pj} \tag{4.13}$$

La fonction objectif (4.2) avec les contraintes (4.3) et (4.4) minimisent la somme de la charge de travail excédentaire par rapport à la moyenne A dans le but d'équilibrer le plus possible la charge de travail entre les différentes positions. Les contraintes (4.5) vérifient que chaque journée de travail est affectée à une position. Les contraintes (4.6) vérifient que chaque journée de travail est assignée à un jour de travail dans chaque position. Les contraintes (4.7), (4.8) et (4.9) imposent les règles de repos de nuit (9 heures et 10 heures) et de période libre (57 heures), respectivement. Ces trois contraintes interdisent l'affectation de combinaisons de journées de travail pour lesquelles les temps de début et de fin chevauchent avec l'intervalle de temps $[t, t+i]$ ($i = 540, 600, 3420$ pour chaque contrainte, respectivement). Les contraintes (4.10) imposent que la règle de la moyenne des repos de nuit sur une période de 28 jours soit d'au moins 12 heures de repos. La règle d'avoir au plus une journée de travail d'amplitude égale à 13 heures ou plus par semaine est vérifiée par les contraintes (4.11). Finalement, la

règle sur le nombre maximal de journées de travail consécutives desservant la même ligne d'autobus est exprimée par les contraintes (4.12). Les contraintes (4.13) imposent, quant à elles, l'intégralité des variables de décision.

Problème RCRF-Y : Le modèle mathématique du problème de roulement est formulé comme suit :

$$\min \quad \sum_{p\in\mathbf{P}} z_p \tag{4.14}$$

$$\text{sujet à :} \quad \sum_{j\in\mathbf{J}_p^W}\sum_{d\in\mathbf{D}_{pj}^R} W_d x_{pj}^d - A \le z_p \qquad \forall p\in\mathbf{P} \tag{4.15}$$

$$z_p \ge 0 \qquad \forall p\in\mathbf{P} \tag{4.16}$$

$$\sum_{p\in\mathbf{P}} x_{pj}^d = 1 \qquad \forall j\in\mathbf{J}, d\in\mathbf{D}_j^R \tag{4.17}$$

$$\sum_{d\in\mathbf{D}_{pj}} x_{pj}^d = 1 \qquad \forall p\in\mathbf{P}, j\in\mathbf{J}_p^W \tag{4.18}$$

$$\sum_{d\in\mathbf{D}_{pj}} F_{dt}^+ x_{pj}^d + \sum_{d\in\mathbf{D}_{f(p,j,1)}} B_{d,t+600}^- x_{f(p,j,1)}^d \le 1 \qquad \forall p\in\mathbf{P}, j\in\mathbf{Y}_p^{10}, t\in\mathbf{T}_{pj} \tag{4.19}$$

$$\sum_{d\in\mathbf{D}_{pj}} F_{dt}^+ x_{pj}^d + \sum_{d\in\mathbf{D}_{f(p,j,3)}} B_{d,t+3480}^- x_{f(p,j,3)}^d \le 1 \qquad \forall p\in\mathbf{P}, j\in\mathbf{Y}_p^{58}, t\in\mathbf{T}_{pj} \tag{4.20}$$

$$\sum_{d\in\mathbf{D}_a}\sum_{i=0}^{K^s-1} x_{f(p^s,j^s,i)}^d \le K^s y_a^s \qquad \forall a\in\mathbf{T}, s\in\mathbf{S} \tag{4.21}$$

$$\sum_{a\in\mathbf{T}} y_a^s \le 2 \qquad \forall s\in\mathbf{S} \tag{4.22}$$

$$\sum_{d\in\mathbf{D}_{pj}} E_d x_{pj}^d - \sum_{d\in\mathbf{D}_{p'j'}} E_d x_{p'j'}^d \le 300 \qquad \forall s\in\mathbf{S}, (p,j),(p',j')\in\mathbf{Y}^s, (p,j)\ne(p',j') \tag{4.23}$$

$$\sum_{d\in\mathbf{D}_{p'j'}} E_d x_{p'j'}^d - \sum_{d\in\mathbf{D}_{pj}} E_d x_{pj}^d \le 300 \qquad \forall s\in\mathbf{S}, (p,j),(p',j')\in\mathbf{Y}^s, (p,j)\ne(p',j') \tag{4.24}$$

$$x_{pj}^d \in \{0,1\} \quad \forall p\in\mathbf{P}, j\in\mathbf{J}_p^W, d\in\mathbf{D}_{pj} \tag{4.25}$$

$$y_a^s \in \{0,1\} \quad \forall s\in\mathbf{S}, a\in\mathbf{T} \tag{4.26}$$

Pareillement au modèle précèdent, la fonction objectif (4.14) avec les contraintes (4.15) et (4.16) équilibrent la charge de travail entre les chauffeurs. Les contraintes (4.17)-(4.20) servent à assurer l'affectation des jounrées de travail tout en respectant les règles de repos de nuit et de période libre. Les contraintes (4.21) et (4.22) interdisent d'avoir plus que deux types

de journées de travail par séquence de travail, et les contraintes (4.23) et (4.24) assurent un maximum de 5 heures de gap entre le temps de fin minimum et temps de fin maximum des journées de travail dans une même séquence. Finalement, les contraintes (4.25) et (4.26) assurent l'intégralité des variables de décision.

4.4 Discussion et formulations alternatives

Dans nos données, la plupart des journées de travail sont valides pour presque tous les roulements (le cas A) et pour plusieurs jours de la semaine. La fixation initiale des jours de repos permet sans doute d'éliminer certaines variables, ce qui réduit partiellement la symétrie dans le modèle. Malgré ce fait, la symétrie reste élevée.

Supposons qu'il y a 4 journées de travail d_1, d_2, d_3, d_4 qui sont valides pour tous les jours de la semaine. Supposons aussi que ces journées de travail satisfont les contraintes de faisabilité d'un jour à l'autre (par exemple les règles de repos entre deux journées de travail). Les deux exemples de roulements dans le tableau 4.6 illustrent la symétrie du modèle en question. Nous remarquons que, dans les deux solutions, les mêmes journées de travail sont affectées à chaque position, mais à différents jours (par exemple, la position 1 contient les journées de travail d_2, d_3 et d_4).

Tableau 4.6 Deux solutions symétriques

Solution 1							
Position	Lun	Mar	Mer	Jeu	Ven	Sam	Dim
1	d_4	R	R	d_3	d_4	R	d_2
2	R	R	d_1	d_2	d_3	d_4	R
3	d_1	d_2	R	R	R	d_3	d_4
4	R	d_4	d_2	d_1	R	R	d_3

Solution 2							
Position	Lun	Mar	Mer	Jeu	Ven	Sam	Dim
1	d_3	R	R	d_4	d_2	R	d_4
2	R	R	d_2	d_1	d_3	d_4	R
3	d_4	d_1	R	R	R	d_3	d_2
4	R	d_4	d_1	d_2	R	R	d_3

Notons que les charges de travail par position sont les mêmes dans les deux exemples, malgré les différents positionnements des journées de travail dans le roulement (la charge de travail par position est la somme des charges de travail des journées de travail qui lui sont assignées).

En plus des règles de repos de nuit entre deux journées de travail, les règles régissant la couverture des lignes d'autobus et les règles s'étalant sur plusieurs jours (la règle sur 28 jours par exemple) permettent de réduire la symétrie du modèle. Ces dernières interdisent certaines séquences de journées de travail, ce qui élimine plusieurs possibilités d'affectation.

La fonction objectif visant à minimiser la somme des écarts qui dépassent la moyenne semble réduire la symétrie aussi. Parmi plusieurs séquences de journées de travail, le modèle sélectionne les séquences qui donnent une charge de travail dépassant le moins possible la moyenne totale.

Dans les différentes expériences menées pour la résolution du RCRF, nous avons essayé d'équilibrer la charge de travail entre les chauffeurs avec une nouvelle fonction objectif. Cette dernière vise à minimiser la somme des valeurs absolues des écarts par rapport à la moyenne. Les fonctions objectifs (4.2) et (4.14) avec les contraintes (4.4)-(4.3) et (4.16)-(4.15) ont été remplacées par la fonction objectif et les contraintes suivantes :

$$\min \quad \sum_{p \in \mathbf{P}} z_p \tag{4.27}$$

$$\text{sujet à :} \quad \sum_{j \in \mathbf{J}_p^W} \sum_{d \in \mathbf{D}_{pj}^R} W_d x_{pj}^d - A \leq z_p \qquad \forall p \in \mathbf{P} \tag{4.28}$$

$$A - \sum_{j \in \mathbf{J}_p^W} \sum_{d \in \mathbf{D}_{pj}^R} W_d x_{pj}^d \leq z_p \qquad \forall p \in \mathbf{P} \tag{4.29}$$

Par ailleurs, les règles sur les repos entre les journées de travail ont aussi été modélisées en calculant la différence entre le temps de fin et le temps de début de deux journées de travail. La contrainte suivante constitue un exemple de ce type (faible) de modélisation pour la règle de 10h de repos dans *RCRF-A* :

$$\sum_{d \in \mathbf{D}_{f(p,j,1)}} S_d x_{f(p,j,1)}^d - \sum_{d \in \mathbf{D}_{pj}} E_d x_{pj}^d \geq 600 \qquad \forall p \in \mathbf{P},\, j \in \mathbf{J}_p^{10} \tag{4.30}$$

Nous appellerons ces contraintes : *contraintes de différence de temps entre les journées de travail (CDT)*. Nous pensons que la modélisation forte (MF) des contraintes de repos entre les journées de travail, (4.7)-(4.9) et (4.19)-(4.20), permet d'avoir des relaxations linéaires plus serrées avec moins de variables fractionnaires (voir figure 4.5 et figure 4.6). Malgré le fait que cette modélisation augmente le nombre de contraintes dans le modèle, elle permet

de réduire le nombre de coupes et de nœuds explorés dans l'arbre de branchement, ce qui accélère le processus de résolution.

La différence entre les deux modèles ($RCRF$-A et $RCRF$-Y) et leurs instances relatives réside en premier lieu dans le nombre de roulements dans chaque instance. L'instance A contient plusieurs roulements alors que l'instance Y en contient un seul. Chacun des roulements de l'instance A possède différentes possibilités d'affectation de journées de travail pour un même jour j ; ces mêmes journées de travail sont possibles pour plusieurs roulements à la fois. Une journée de travail le jour j sera assignée à seulement une position dans un roulement. Ceci augmente le nombre de variables décidant quelle journée de travail doit être affectée à quel roulement. Quant au roulement Y, chaque journée de travail possible est une journée de travail qui est assignée à un jour de travail, il ne reste seulement qu'à décider dans quelle position.

4.5 Résultats numériques

Les modèles précédents ont été implémentés en utilisant la technologie Concert de IBM et résolus en utilisant le solveur CPLEX, version 12.7.0.0 ; les paramètres de CPLEX sont fixés aux valeurs par défaut du solveur. Les tests ont été effectués sur une machine Linux équipée d'un processeur Intel Xeon E5-2637 V2, cadencé à 3,5 GHz et d'une RAM de 128 Go.

4.5.1 Instances

Nous avons testé les modèles sur plusieurs instances réelles provenant de deux ensembles de données (A et Y) fournies par notre partenaire industriel GIRO : La première instance A contient 14 roulements, 202 positions, 881 journées de travail et 14 lignes d'autobus. La taille des roulements varie de 6 à 35 et les roulements contiennent de deux à trois jours de repos par semaine selon le type du roulement. La deuxième instance Y, quant à elle, contient un seul roulement de 52 positions et 238 journées de travail de 5 types différents. Dans ce même roulement, nous avons des semaines avec deux jours de repos et d'autres avec trois jours de repos.

L'instance A contient beaucoup de roulements et un grand nombre de chauffeurs d'autobus, d'où l'intérêt d'en extraire d'autres instances, de tailles variables, pour tester le fonctionnement du modèle sur différents nombres de chauffeurs. L'extraction de nouvelles instances ne changera pas la structure du modèle puisque les roulements resteront les mêmes. Quant au cas de l'instance Y, il s'agit d'un seul roulement avec un petit nombre de chauffeurs, et si nous essayons d'en extraire d'autres roulements, la structure du modèle changera (par exemple, la

rotation des chauffeurs sur les positions).

À partir de l'instance A, nous avons généré aléatoirement 4 instances additionnelles de petite, moyenne et grande tailles comme suit :

— Nous avons choisi pour chaque instance un sous-ensemble de roulements tel que le nombre total des positions de ces roulements soit entre 70 et 180 positions.
— Pour chaque instance et pour chaque roulement, nous avons gardé la même configuration de repos que celle de l'instance originale A.
— Nous avons calculé un nombre n_{ij} de journées de travail nécessaires pour l'instance i dans le jour j.
— Pour chaque instance i et pour chaque jour de travail j, nous avons sélectionné aléatoirement un ensemble de n_{ij} journées de travail qui peuvent être affectées à au moins un roulement dans l'instance, tout en s'assurant que l'instance soit réalisable.

Le tableau 4.7 présente les instances générées, les instances originales A et Y, et leurs caractéristiques, ce qui inclut, entre autres, le modèle de résolution, le nombre de roulements, le nombre de positions et le nombre de journées de travail. Le nom d'une instance $I_{(R,P)}$ précise l'ensemble de données d'origine (I), le nombre de roulements (R) et le nombre de positions (P). Notons que les deux dernières instances correspondent aux ensembles de données complets A et Y.

Tableau 4.7 Instances et leurs caractéristiques

Instance	Modèle	#Roulements	#Positions	#Journées de Travail
$A_{(4,71)}$	$RCRF-A$	4	71	234
$A_{(5,110)}$	$RCRF-A$	5	110	343
$A_{(7,124)}$	$RCRF-A$	7	124	343
$A_{(10,146)}$	$RCRF-A$	10	146	375
$A_{(11,178)}$	$RCRF-A$	11	178	447
$A_{(14,202)}$	$RCRF-A$	14	202	881
$Y_{(1,52)}$	$RCRF-Y$	1	52	238

4.5.2 Résultats comparatifs

Dans cette partie du chapitre, nous présentons d'abord les résultats obtenus avec l'utilisation des deux fonctions objectifs : la minimisation des écarts positifs des charges de travail par rapport à la moyenne, (4.2) et (4.14), et la minimisation de la valeur absolue de ces écarts par rapport à la moyenne (4.27). Ensuite, nous comparons les deux modélisations (MF et CDT) des contraintes de repos entre les journées de travail.

Toutes les journées de travail sont affectées en respectant toutes les contraintes. Des journées de réserve ont aussi été affectées aux jours restants (jours de travail) dans les roulements correspondant au problème $RCRF\text{-}A$. En ce qui concerne le problème $RCRF\text{-}Y$, aucune journée de réserve n'a été affectée étant donné que le nombre de journées de travail est égal exactement au nombre de jours de travail.

Pour les deux instances complètes $A_{(14,202)}$ et $Y_{(1,52)}$, les deux fonctions objectifs donnent des résultats semblables comme nous pouvons le constater en comparant les graphiques des figures 4.1 et 4.2 (instance $A_{(14,202)}$) et ceux des figures 4.3 et 4.4 (instance $Y_{(1,52)}$). Pour l'instance $A_{(14,202)}$, nous remarquons que l'écart entre la plus grande charge de travail et la plus petite est de 475 min lorsqu'il s'agit de minimiser la somme des écarts positifs, et de 478 min lorsqu'il s'agit de minimiser la somme des valeurs absolues. Pour l'instance $Y_{(1,52)}$ ces écarts sont de 441 min et de 462 respectivement.

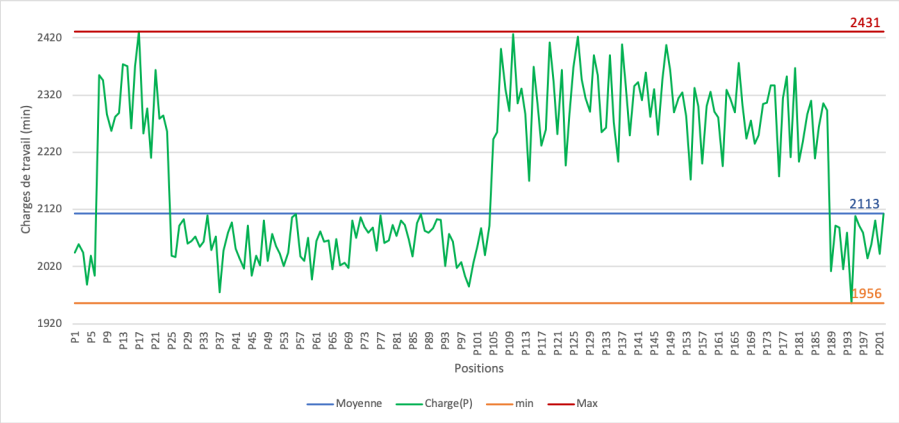

Figure 4.1 Minimisation de la somme des écarts positifs $A_{(14,202)}$

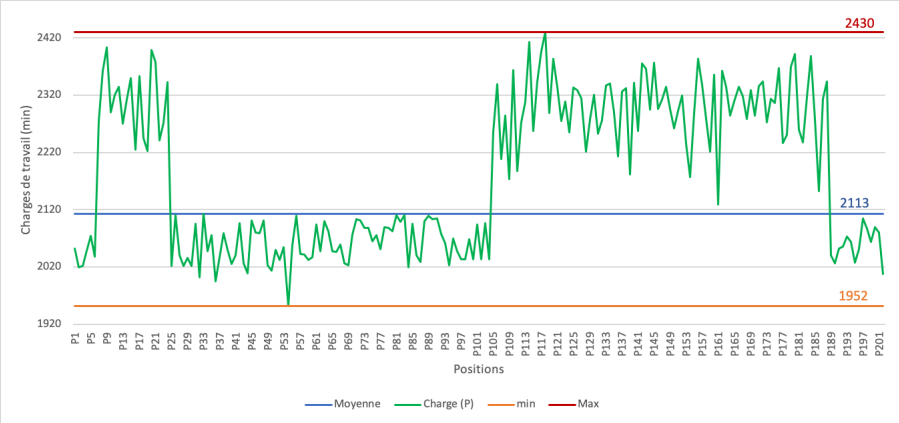

Figure 4.2 Minimisation de la somme des valeurs absolues des écarts $A_{(14,202)}$

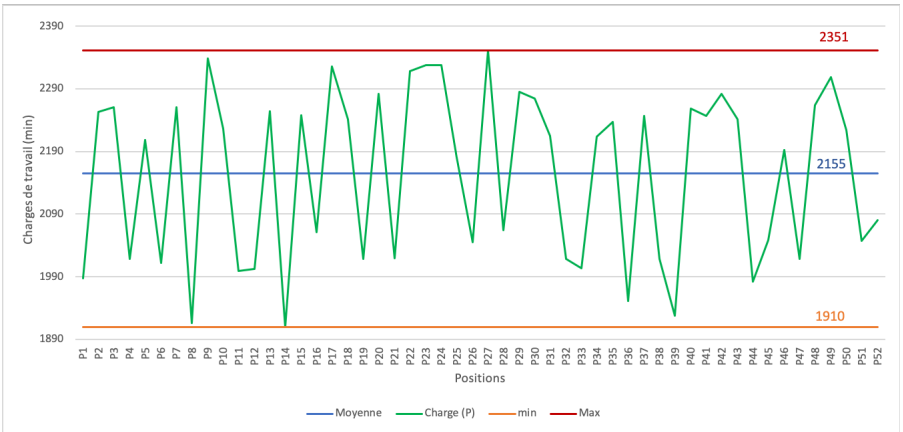

Figure 4.3 Minimisation de la somme des écarts positifs $Y_{(1,52)}$

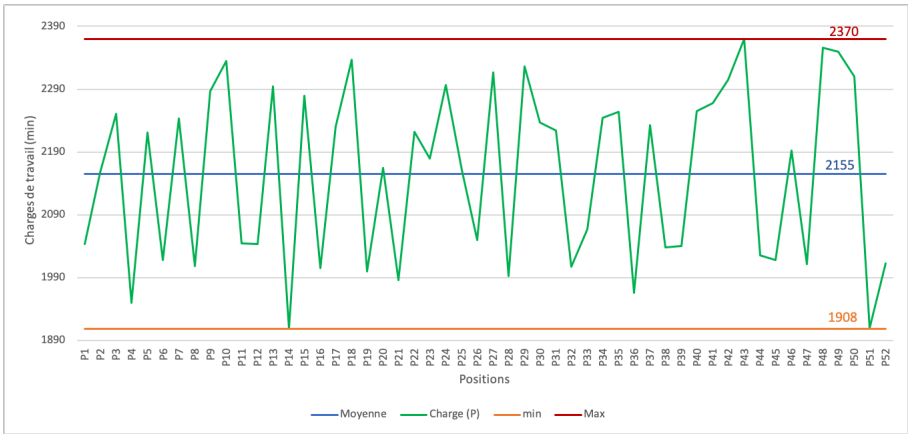

Figure 4.4 Minimisation de la somme des valeurs absolues des écarts $Y_{(1,52)}$

Dans ce qui suit, nous présentons les résultats numériques associés aux deux modèles $RCRF$-A et $RCRF$-Y avec les deux fonctions objectifs pour la formulation MF des contraintes de repos. Les résultats sont présentés dans les tableaux 4.8 et 4.9. Pour chaque instance, nous présentons les nombres de variables et de contraintes, le nombre de nœuds de branchement, le nombre de coupes appliquées et le temps total de calcul (en secondes). Nous présentons aussi les coûts des solutions obtenues.

En comparant les deux tableaux 4.8 et 4.9, nous constatons que les temps de calcul sont légèrement plus longs avec la fonction objectif minimisant la somme des valeurs absolues. Le nombre de coupes est aussi plus grand dans la plupart des instances comparé aux résultats des roulements avec la fonction objectif minimisant la somme des écarts positifs. Notons que les solutions sont obtenues dans des temps de calcul courts même lorsqu'il s'agit d'instances

Tableau 4.8 Résultats $RCRF$ avec l'objectif des écarts positifs et la MF

Instances	#Variables	#Contraintes	#Noeuds	#Coupes	Temps(s)	Coût(min)
$A_{(4,71)}$	18237	22310	0	15	5	6261
$A_{(5,110)}$	38533	44864	0	62	19	9571
$A_{(7,124)}$	48337	51103	0	79	21	10225
$A_{(10,146)}$	66601	60002	0	27	42	12552
$A_{(11,178)}$	97888	84631	0	117	83	16580
$A_{(14,202)}$	128630	105945	0	86	99	19339
$Y_{(1,52)}$	12945	16506	0	214	4	3258.5

Tableau 4.9 Résultats $RCRF$ avec l'objectif des valeurs absolues et la MF

Instances	#Variables	#Contraintes	#Noeuds	#Coupes	Temps(s)	Coût(min)
$A_{(4,71)}$	18237	22310	0	59	7	8227
$A_{(5,110)}$	38533	44864	0	61	21	12905
$A_{(7,124)}$	48337	51103	0	110	23	12874
$A_{(10,146)}$	66601	60002	0	110	50	15504
$A_{(11,178)}$	97888	84631	0	91	78	21187
$A_{(14,202)}$	128630	105945	0	131	117	24726
$Y_{(1,52)}$	12945	16506	0	380	12	6516.8

de grande taille ($A_{14,202}$ en moins de deux minutes avec les deux objectifs).

Dans ce ce qui suit, nous présentons les résultats obtenus avec les formulations (MF) et (CDT). Tel que mentionné dans la section précédente, la formulation MF permet d'avoir moins de variables fractionnaires (dans le cas de l'instance A). Le graphique de la figure 4.5 présente le nombre des variables fractionnaires x^d_{pj} dans les solutions des relaxations linéaires des modèles $RCRF$ pour les instances A et l'instance Y. Nous remarquons que le nombre des variables fractionnaires avec la modélisation CDT est plus que le double du nombre de celles-ci avec la modélisation MF.

En plus du nombre de variables fractionnaires par type de modélisation, nous présentons dans la figure 4.6 les distributions des valeurs de variables fractionnaires des deux instances $A_{(14,202)}$ et $Y_{(1,52)}$. La figure montre que le graphique A, CDT contient un grand nombre de variables fractionnaires dont les valeurs sont inférieures ou égales à 0.5, ce qui est moins dense dans le graphique A, MF.

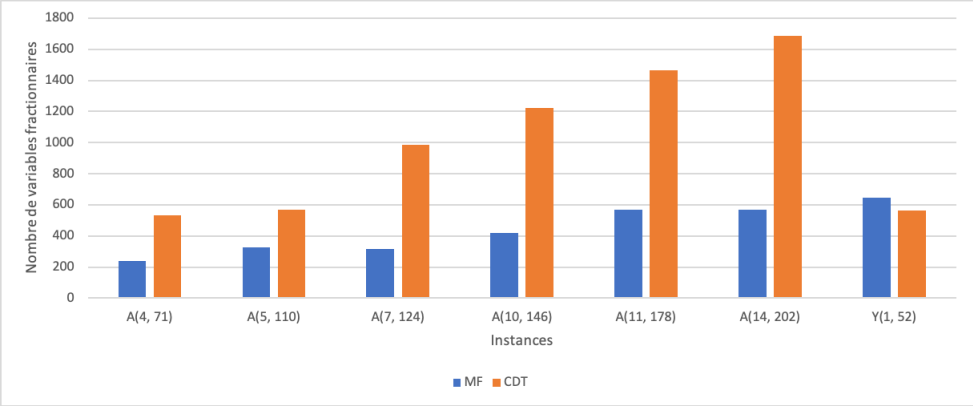

Figure 4.5 Nombre de variables fractionnaires des instances A et l'instance Y

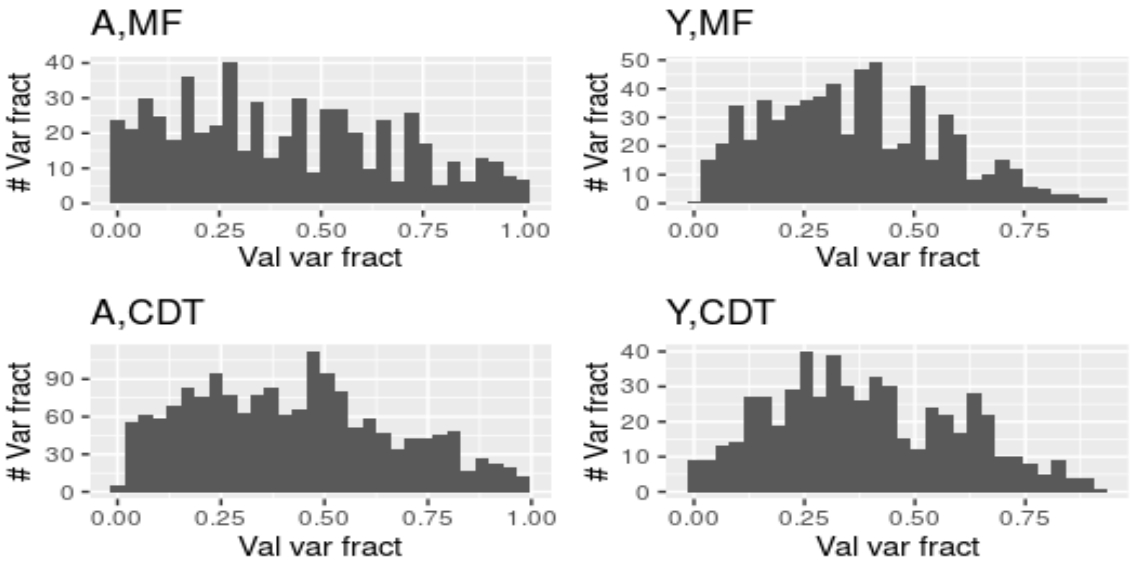

Figure 4.6 Distributions des valeurs des variables fractionnaires des instances $A_{(14,202)}$ et $Y_{(1,52)}$ par type de modélisation

Le tableau 4.10 présente les résultats obtenus lors de la résolution des modèles $RCRF$ avec la modélisation CDT et l'objectif de minimiser les écarts positifs. Les résultats de ce tableau doivent être comparés à ceux du tableau 4.8. Il est clair que le nombre de nœuds de branchement et de coupes a largement augmenté et que le temps de calcul a plus que doublé dans la plupart des instances. Par conséquent, bien que les modèles $RCRF$ aient été résolus à l'optimalité dans des temps de calcul très raisonnables avec les deux types de modélisation (MF et CDT), la modélisation MF pourrait s'avérer plus avantageuse pour des instances de plus grande taille ou dans des contextes plus complexes, par exemple, sans jours de repos fixés.

Tableau 4.10 Résultats $RCRF$ avec l'objectif des écarts positifs et la modélisation CDT

Instance	#Variables	#Contraintes	#Noeuds	#Coupes	Temps(s)	Coût(min)
$A_{(4,71)}$	18237	3617	331	156	10	6261
$A_{(5,110)}$	38533	5855	1384	359	33	9571
$A_{(7,124)}$	48337	6436	2222	419	51	10225
$A_{(10,146)}$	66601	7497	5455	426	129	12552
$A_{(11,178)}$	97888	9572	4622	591	250	16580
$A_{(14,202)}$	128630	11176	0	320	169	19339
$Y_{(1,52)}$	12945	14270	739	675	10	3258.5

4.5.3 Comparaison avec les solutions du partenaire industriel

Notre partenaire industriel utilise un algorithme heuristique pour résoudre les problèmes RCRF qui résout une suite de sous-problèmes séquentiellement. Chaque sous-problème est défini par un sous-ensemble de positions et de jours, formulé comme un problème de réseau avec contraintes additionnelles et résolu avec un solveur MILP. Pour assurer la faisabilité de la solution globale, des contraintes dérivées des solutions des sous-problèmes précédemment résolus peuvent être imposées aux sous-problèmes ultérieurs. Cette heuristique est dénotée $H - Ind$.

Le tableau 4.11 suivant présente les coûts des solutions obtenues avec les deux algorithmes pour les deux instances $A_{(14,202)}$ et $Y_{(1,52)}$. Le même tableau présente aussi le gain (en minutes) obtenu sur le coût par notre approche dénotée $RCRF$.

Tableau 4.11 Comparaison entre les résultats de RCRF et H-Ind

	Coût(min)		
	$RCRF$	$H - Ind$	Gain (%)
A_{202}	19339	19395	0.2
Y_{52}	3259	3486	6.5

Notons que les solutions obtenues par l'algorithme industriel sont issues des mêmes configurations de jours de repos que nous avons utilisées dans nos modèles. L'algorithme H-Ind a été exécuté par notre partenaire industriel et qu'il nous a seulement communiqué le temps de calcul obtenu pour l'instance $A_{(14,202)}$, soit environ une heure. Il s'agit d'un temps nettement supérieur aux 2 minutes requises par notre approche. Cette comparaison montre que la solution obtenue est mieux équilibrée par rapport à la solution industrielle. On obtient un gain moyen de 3.35%, qui représente environ 5 heures d'excédents par rapport à la charge de

travail moyenne par semaine.

4.6 Conclusion

Dans ce chapitre, nous avons traité deux contextes différents de problèmes de roulements cycliques avec des jours de repos fixés. Nous avons affecté les journées de travail aux différents roulements tout en équilibrant le plus possible la charge de travail hebdomadaire entre les chauffeurs d'autobus. Nous avons proposé deux modèles mathématiques avec une formulation forte (MF) des contraintes de repos entre deux journées de travail et une formulation de type CDT. Nous avons montré que le modèle MF permet d'obtenir des solutions optimales dans des temps de calcul courts avec moins de branchements et moins de coupes. Deux différentes formulations de la fonction objectif ont aussi été étudiées dans le but d'équilibrer la charge de travail entre les chauffeurs.

Les résultats numériques obtenus montrent l'efficacité de nos modèles. Nous avons obtenu des solutions meilleures que celles de notre partenaire industriel avec un gain sur les excédents par rapport à la moyenne des charges de travail. Nos modèles ne sont pas restreints à l'affectation des journées de travail avec jours de repos fixés. En effet, en apportant quelques adaptations, ils pourront être utilisés pour la résolution d'autres types de problèmes d'horaire et ils pourront être étendus à la résolution intégrée de problèmes de roulements et de jours de repos comme présenté dans le chapitre suivant.

CHAPITRE 5 ARTICLE 1 : INTEGRATED AND SEQUENTIAL SOLUTION METHODS FOR THE CYCLIC BUS DRIVER ROSTERING PROBLEM

Cet article a été écrit par S.Er-Rbib, G.Desaulniers, I.Elhallaoui et A.Bani, et a été publié en Janvier 2020 dans la revue *Journal of the Operational Research Society*.

5.1 Introduction

Public transit companies are constantly required to improve the quality of their services to enhance population mobility and reduce traffic congestion. Operating costs impact the service quality, which prompts transit companies to improve their processes in order to manage as efficiently as possible their resources (buses, drivers, ...) at all planning levels: strategic, tactical and operational. In this introduction, we first present an overview of the major planning steps in public transport and briefly state the problem of interest for this paper, namely, the cyclic bus driver rostering problem (CBDRP). We then review the literature on this topic and position our work with respect to this literature.

5.1.1 Planning process and problem overview

As described in Desaulniers and Hickman (2007), the operations planning process in a public transit system is usually divided in the following three planning levels. Strategic planning involves taking long-term decisions, such as network (route) design and resource acquisition. At this level, the companies typically try to maximize quality of service (e.g., number of passengers transported) while respecting budgetary restrictions. Tactical planning is performed on a seasonal basis. It mostly aims at establishing service frequencies on the bus routes and the exact timetable such that service quality is maximized taking into account the available resources. Finally, at the operational level, the decisions concern the scheduling of the buses and the drivers as well as the planning of bus maintenance operations and overnight bus parking. These problems are solved on a monthly to a weekly basis and, sometimes, on a daily basis. Unlike the previous levels, the operational phase aims at minimizing total cost.

The three most important problems of the operational planning phase are *bus scheduling*, *duty scheduling*, and *driver rostering*. Bus scheduling consists of determining a least-cost feasible set of schedules for the available fleet of buses such that it covers all trips of a planned timetable (hereafter called active trips) over a given day. The buses may be assigned to one

or several depots and may be of different types. A bus schedule is defined by a sequence of active trips to cover. It starts and ends at the same depot and may contain deadhead trips (travels without passengers) between two consecutive active trips or between the depot and an active trip.

Drivers are also assigned to depots, i.e., each driver can only drive a bus assigned to its depot. Therefore, the problem of determining the schedules of the drivers is separable by depot. For medium to large-sized instances, this problem is complex and is usually solved in two steps. In the duty scheduling step, anonymous duties are built to cover at minimum cost a set of trip segments operated out of the same depot. A *duty* represents the activities to be performed by a driver on a given day. It is defined as a sequence of (active or deadhead) trip segments, possibly interspersed with breaks and repositioning moves, that respects various feasibility rules (e.g., a maximum working time per duty and a maximum working time without a break). Given that the duties are anonymous, the problem can be separated by day. Global constraints on the maximum number of duties of each type (straight or split, long or short, early, afternoon, or late, etc.) that can be built are often considered. These constraints are derived from the set of available drivers and their preferences. The objective of the problem is to minimize the driver wages. Given that the trip timetable is defined over a week, the duties may vary from one day of the week to another but are the same from one week to another over a long planning horizon (e.g., one year) except on Holidays.

Once the duties are computed, drivers must be assigned to them. This assignment process is realized in the driver rostering step which determines at the same time the working days and the days off of each driver over a planning horizon. In the majority of the North American companies, this process is performed in order of seniority, leaving little room for optimization. In Europe, equity among the employees often prevails and the main objective is to balance the workload between the drivers as much as possible over a week. Moreover, to increase fairness in the schedules assigned to the drivers over the planning horizon, the drivers are partitioned into groups of drivers having similar preferences with regard to their schedules (e.g., working four days per week or early duties) and who will share their weekly schedules in a cyclic manner. More specifically, for a group of n drivers, if driver i, $i = 1, \ldots, n$, is assigned to schedule s_i in a given week, then he/she is assigned to schedule s_{i+1} (or s_1 if $i = n$) in the following week. This set of schedules is called a *roster* and the resulting problem of computing simultaneously the rosters for all groups is called the CBDRP. Note that its definition depends on the labor law and collective agreement rules that each roster must respect (e.g., a minimum number of days off per week and a minimum rest time between two consecutive working days) and may, thus, differ from one bus operator to another. Additional features (such as shifts and standby duties) might also be considered to yield different CBDRP variants. In Section 5.2,

we provide a detailed definition of the CBDRP faced by some European companies.

5.1.2 Literature review

Rostering or personnel scheduling problems arise in various domains of applications (see Ernst et al. (2004a) and Van den Bergh et al. (2013)): transportation, healthcare, production, retail, hospitality, finance, etc. From one domain to another, their definition may vary in different ways. Broadly speaking, these problems aim at determining the work schedule of each available employee over a planning horizon. These schedules, which must respect various operational (labor law and union) rules that differ from one domain to another, specify the days off and working days for each employee and, for each working day, the corresponding schedule and the task(s) to be accomplished. In public transit, the task assigned to a driver on a given day is a duty and, given that most duties do not have the same schedule, the drivers may be assigned to a wide variety of working schedules (essentially, a duty can start at any time during the day and last any duration in minutes between, e.g., 3 and 10 hours). In other transportation modes such as extra-urban rail and air transportation where the crew members travel long distances, crew members are not assigned to daily tasks but rather to rotations (pairings) which contain several duties that are separated by rest periods outside their home base. Furthermore, in many other sectors such as healthcare, production and call centers, employees work on a very limited number of shifts (e.g., three to five per day). In this case, the rostering problem corresponds to a generalized set covering problem (a number of employees is required for each shift), rather than a set partitioning problem (one employee per duty) like in public transit. For these reasons, we focus below on works published on public transit rostering.

In the last twenty-five years, several papers have been published on public transit rostering. Given the wide variety of contexts arising in practice, these papers have tackled different problem variants and proposed different solution approaches. Table 5.1 lists them and summarizes some of their characteristics. In the following, we specify the meaning of each column in this table and, if worthwhile, comment on the corresponding characteristic. The second column provides the transportation mode (bus, commuter train or subway) on which the paper focused. Most papers concern bus transportation. The next column indicates whether or not cyclic rosters are built. Non-cyclic rosters may be sought when, e.g., individual driver preferences or vacations must be taken into account or when the trip timetable is not highly regular from one week to another. In the fourth column, we indicate if days off scheduling is explicitly considered. In the three papers Bianco et al. (1992),Carraresi and Gallo (1984),Ma et al. (2014) where it is not, sequences of consecutive duties are computed without mention-

Table 5.1 Papers on rostering in public transit and some of their characteristics.

Reference	Mode	Cyclic	Days off	Shifts	Minimize	Approach	Exact	Methodology
Carraresi and Gallo (1984)	bus	no	no	no	maximum workload	duty sequencing only	no	bottleneck assignment heuristic
Bianco et al. (1992)	bus	no	no	no	roster costs maximum workload	duty sequencing only	no	linear programming & bottleneck assignment heuristic
Caprara et al. (1998)	rail	yes	yes	no	number of drivers	integrated	no	Lagrangian relaxation & constructive heuristic
Sodhi and Norris (2004)	subway	yes	yes	yes	preference-related penalties soft constraint violations	1) day-off/shift assignment 2) duty assignment	no	integer programming
Lezaun et al. (2006)	bus	no	from fixed pattern set	yes	preference-related penalties shift assignment variance	1) yearly reserve scheduling 2) weekly day-off/shift assignment 3) seasonal schedule construction 4) driver assignment	no	integer programming
Moz et al. (2009)	bus	no	yes	no	maximum workload number of under-utilized employees	integrated	no	evolutionary heuristics
Hartog et al. (2009)	train	yes	yes	yes	preference-related penalties	1) duty set partitioning 2) day-off/shift assignment 3) duty assignment	no	integer programming
Nurmi et al. (2011)	bus	no	yes	yes	soft constraint violations	1) day-off assignment 2) shift assignment	no	evolutionary heuristic
Ma et al. (2014)	bus	no	no	no	workload variance	duty sequencing only	no	genetic algorithm
Nishi et al. (2014)	train	yes	yes	no	maximum workload number of rosters	integrated	yes	Benders decomposition
Xie and Suhl (2015)	bus	yes/no	yes	yes	maximum workload soft constraint violations preference-related penalties	integrated & 1) days-off/shift assignment 2) duty assignment	yes/no	integer programming
Borndörfer et al. (2015)	bus	yes	yes	no	roster costs soft constraint violations	integrated	no	local search
Mesquita et al. (2015)	bus	no	from fixed pattern set	yes	roster costs preference-related penalties	integrated	no	integer programming matheuristic

ing how these sequences are assembled to form rosters including days-off. Note also that, in Lezaun et al. (2006), Mesquita et al. (2015), it is assumed that a set of possible days-off patterns is provided as an input to the problem.

The fifth column specifies whether shifts (or duty types) are considered in the problem. When they are, they allow to partition the duties by shift as each duty belongs to a single shift (e.g., morning, afternoon, evening). Shifts are used to define operational rules and preferences. For instance, instead of imposing a minimum rest time between duties on two consecutive days, valid shift sequences (e.g., an evening shift can be followed by an afternoon or an evening shift only) are stipulated Lezaun et al. (2006), Mesquita et al. (2015), Nurmi et al. (2011). Furthermore, some valid sequences might be preferable to others (e.g., a morning shift followed by a morning shift is better than followed by an evening shift) Hartog et al. (2009), Lezaun et al. (2006), Sodhi and Norris (2004). In some cases Lezaun et al. (2006), Mesquita et al. (2015), instead of requesting one driver for each duty, the demand is expressed as a number of required drivers by shift, greatly reducing the combinatorial difficulty of the problem.

The next column is devoted to the objective function considered. As can be observed, multiple objectives, which are combined into a single one using weights, are often taken into account. A frequent objective is to balance the workload between the drivers as much as possible, either by minimizing the maximum workload or the workload variance. Other objectives include minimizing the roster costs which essentially boils down to minimizing the number of drivers, minimizing soft constraint violations, and minimizing preference-related penalties (e.g., to avoid assigning a morning shift followed by an evening shift).

The general solution approach selected to tackle the problem is provided in the seventh column. Five papers proposed an integrated approach, four a sequential one and Xie and Suhl (2015) developed both approaches. The remaining three papers Bianco et al. (1992), Carraresi and Gallo (1984), Ma et al. (2014) focused on duty sequencing only. The eighth column indicates whether the overall solution algorithm is exact or not. Note that, to have an exact algorithm, an integrated approach must have been proposed. One can observe that only two exact algorithms were developed by Nishi et al. (2014) and Xie and Suhl (2015). Finally, the last column stipulates the types of methodology used, namely, mathematical programming tools and various types of heuristics.

Among the papers listed in Table 5.1, those of Nishi et al. (2014) and Xie and Suhl (2015) are the most interesting from our point of view because we aim at solving exactly large-sized CBDRP instances (i.e., with more than 1000 duties). Nishi et al. (2014) developed a Benders decomposition algorithm where it is assumed that the same set of duties is operated each day

and that all copies of the same duty must be assigned to the same roster. In this approach, the Benders master problem aims at assigning the duties to the rosters and to determine the length of each roster which might not be a multiple of seven days. The Benders subproblems are feasibility problems that determine the position of each duty and each day off in each roster. With this algorithm, the authors were able to solve to optimality only small-sized instances (with up to 15 duties per day) within a 1-hour time limit.

On the other hand, Xie and Suhl (2015) considered a cyclic and a non-cyclic rostering problem for which they designed a two-step heuristic approach (day-off/shift assignment first, duty assignment second) that can solve very large problem instances in relatively fast computational times as well as an exact integrated approach. In the first step of the heuristic, a multi-commodity network flow formulation is solved by a commercial MILP solver to assign days off and shifts. There is a network per driver (for the non-cyclic case) or per roster (for the cyclic case) which contains a layer for each day of the considered horizon. In each layer, there is a node for each feasible activity (e.g., day-off, morning shift, evening shift, standby) on this day. The arcs link nodes in consecutive layers when it is not forbidden to perform the activities associated with these nodes consecutively. In this formulation, there are variables to specify the flow on each arc of each network as well as variables to determine the violation of the soft constraints. In the second step, an integer program is solved to determine which duty must be assigned to each shift selected in the first step. In this program, the main variables indicate whether or not a specific duty should be assign to a selected shift for a driver on a given day. The proposed integrated approach consists of solving a mixed-integer linear program that combines the models proposed in the two steps of the heuristic approach. Among other computational experiments, the authors tested this integrated approach on a single cyclic problem instance involving 9 rosters, 11 shifts, 1013 duties and 303 standby duties, which was solved in 91 minutes. It should be noted, however, that around 12% of the activities are preassigned for this instance and that prohibitions/preferences for certain shifts and shift sequences restrict significantly the solution space. Interestingly, the authors showed that with the integrated approach, they were able to significantly improve the quality of the solutions produced by the sequential approach.

5.1.3 Contributions and paper structure

The research project conducted for this paper has been defined in collaboration with the company Giro Inc., a world-leader in the development and commercialization of optimization softwares for public transport companies with more than 200 customers worldwide. Its goal is to devise an exact algorithm for solving large-scale instances of one of the most difficult

CBDRPs faced by some of Giro's customers. In this problem, there are no shifts and, therefore, no shift-related rules that reduce the solution space, nor shift-related preferences that can reduce symmetry in the branch-and-bound process and ease the search for integer solutions. Furthermore, all duties can be assigned to most rosters if not all of them. Under these conditions, the network underlying the model of Xie and Suhl (2015) would contain (for an instance of similar size) much more arcs than the one they tested and the computational time would get much larger as the authors acknowledged in their paper.

Consequently, to the best of our knowledge, no exact algorithm has been proposed in the literature to solve large-scale CBDRP instances without shifts that are faced by some customers (transit companies) of our industrial partner. Our main goal consists of designing an integrated solution approach for solving these CBDRP instances. In the context of the CBDRP without shifts, the main contributions of this paper are:

— We introduce a mixed-integer linear programming (MILP) model that can be solved using a commercial MILP solver. With this model, medium-sized CBDRP instances with up to 146 drivers and 640 duties are solved to optimality within two hours of computational time.

— Based on this model, we devise a two-step matheuristic that can produce high-quality solutions (optimal ones, most of the times) in relatively fast computational times for large-sized CBDRP instances.

— We show that warm-starting the integrated approach with the solution computed by the sequential approach allows to significantly reduce the computational times of the MILP solver and solve to optimality large-sized CBDRP instances.

Our computational results also show that the proposed approaches can produce better quality solutions than the solution algorithm currently used by our industrial partner Giro.

This paper is organized as follows. First, the CBDRP considered in this paper is stated in details in Section 5.2. Then, the integrated MILP model for this problem is introduced in Section 5.3, before describing the proposed solution algorithms in Section 5.4. Next, the computational results obtained on real-world instances provided by Giro are presented and discussed in Section 5.5. Finally, conclusions are drawn in Section 5.6.

5.2 Problem statement

The CBDRP is defined over a one-week horizon $\mathbf{J}$ (from Monday to Sunday) which is assumed to repeat for a long period of time (e.g., a season or one year). For each day $j \in \mathbf{J}$, a set of (regular) duties $\mathbf{D}_j^R$ must be operated. Each duty $d \in \mathbf{D}_j^R$, $j \in \mathbf{J}$, is defined by a starting time S_d, an ending time E_d, a set of bus lines served in the duty $\mathbf{L}_d$, and a workload W_d.

The duty workload corresponds to the time paid to the assigned driver. Depending on the duty type, it may differ from the duty length (which is equal to $E_d - S_d$). For instance, the workload is less than the duty length for a split duty which is composed of two parts usually covering the peak hours, separated by a long break period (several hours).

The duties must be assigned to a set of available drivers which are partitioned into groups of drivers having similar preferences. For instance, all drivers preferring duties with a high workload to work only four days per week may be grouped together. The weekly schedules assigned to the drivers of a group form a roster that must be cyclic in the following sense. These schedules are assigned to a position in a cycle (from one to the number of schedules). A driver assigned to position i in a given week will be assigned to position $i + 1$ (or to position 1 if he/she was in last position) in the following week. Consequently, for a roster with n positions, the cycle has a length of n weeks, which means that, after n weeks, the drivers assigned to this roster will have worked the same exact schedules but on different weeks. Note that the roster feasibility rules (stated below) apply to this cycle: for example, if a driver cannot work more than six consecutive days, then the sum of the number of work days at the end of the schedule in any position i and at the beginning of the schedule in position $i + 1$ must not exceed six.

Table 5.2 presents an example of a roster with four positions. In this table, OFF indicates a day off while d_i, $i = 1, \ldots, 8$, represent duties assigned to the drivers on the remaining days. Notice that the same duty may appear more than once but on different days. In this example, the roster respects a maximum of six consecutive working days and the assignment of two days off per week (position).

The set of rosters $\mathbf{R}$ to construct and the set of positions $\mathbf{P}_r$ to fill in each roster $r \in \mathbf{R}$ are given as input to the CBDRP. Given the type of schedules sought for each roster, it may be forbidden to assign some duties to a roster. Consequently, each duty $d \in \mathbf{D}_j^R$, $j \in \mathbf{J}$, is associated with a set of rosters $\mathbf{R}_d$ to which it can be assigned.

Given the complexity of the roster feasibility rules and that the number of duties may vary from one day to another, it is difficult to perfectly match the number of available drivers with the number of duties to operate each day of the week. Therefore, it often happens that there is an excess of drivers on a given day. Instead of assigning extra days off to some drivers, it is common practice to assign them to a standby duty during which they may be called to replace another driver in case of absence. Thus, for each day $j \in \mathbf{J}$, there is a set of standby duties $\mathbf{D}_j^S$ that might or might not be assigned to one or several drivers. Each standby duty $d \in \mathbf{D}_j^S$ is defined by a starting time S_d and an ending time E_d. Given that the driver assigned to such a duty might not be called in, no workload is associated with the

Table 5.2 Example of a roster with four positions (drivers).

Position	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	d_1	d_5	OFF	OFF	d_6	d_8	d_4
2	d_2	d_6	d_1	OFF	OFF	d_3	d_5
3	d_3	d_7	d_2	d_4	OFF	OFF	d_6
4	d_4	d_8	d_3	d_5	d_7	OFF	OFF

standby duties. Note that, in general, the number of available drivers is sufficient to cover all regular duties when the groups of drivers are well designed. Hence, we enforce the covering of all these duties.

Rosters must conform to safety regulations and collective agreement rules that define hard constraints to meet. These rules may vary from one company to another. According to our industrial partner, those stated in Table 5.3 are the most important and common ones. The specific values indicated in this table (e.g., 13 hours in Rule 3) are those used for our computational experiments. They might slightly differ for other practical instances. Note that, for our experiments, there are two roster types: one denoted $\mathbf{R}^2$ in which two days off must be granted to each position, and another denoted $\mathbf{R}^3$ with three days off per position (thus, $\mathbf{R} = \mathbf{R}^2 \cup \mathbf{R}^3$). We consider that all these roster feasibility rules are strict, i.e., they cannot be violated.

The CBDRP can be stated as follows. Given the sets of regular and standby duties $\mathbf{D}_j^R$ and $\mathbf{D}_j^S$ for $j \in \mathbf{J}$, and the set of rosters $\mathbf{R}$, assign duties and days off to the rosters such that all regular duties are assigned to exactly one position on the days they must be operated, standby duties are assigned only if needed, all rosters are feasible with respect to Rules 1 to 8, and the total workload is balanced as much as possible between all positions. This balancing can be formulated in different ways. For instance, if $A = \frac{\sum_{j \in \mathbf{J}} \sum_{d \in \mathbf{D}_j^R} W_d}{\sum_{r \in \mathbf{R}} |P_r|}$ denotes the average workload per position, then one can minimize the sum over all positions of the absolute deviations from A or the squares of these deviations. In our case, our objective function seeks to minimize the sum of the positive deviations, i.e., only over the positions for which the workload exceeds A.

5.3 A mathematical model for the CBDRP

In this section, we present a mathematical model for the CBDRP defined in the previous section and discuss some of its characteristics. Beforehand, we introduce the necessary notation.

Table 5.3 Roster feasibility rules.

Rule	Description
1	Depending on the roster type, there must be exactly 2 or 3 days off per position, with at least two consecutive days off in each position. Due to the cyclic nature of the rosters, consecutive days off on Sunday and Monday can serve as a double day off on either position (that with the Sunday off or that with the Monday off) but not both.
2	The number of consecutive days without a day off cannot exceed 6.
3	There must be at least 9 hours of rest between two consecutive duties.
4	There must be at least 10 hours of rest between two consecutive duties if they are immediately followed by a third duty.
5	A 57-hour period of rest, including at least two consecutive days off, must be assigned to each position.
6	In any period of 28 consecutive days ending with a working day, the average rest time between the duties must be at least 12 hours.
7	At most one duty lasting 13 hours or more can be assigned to each position.
8	At most two duties serving the same bus line can be assigned consecutively.

5.3.1 Notation

For the sake of completeness, we re-introduce the notation presented above. Note that this notation respects the following convention: sets are denoted using bold capital letters, variables using lowercase letters, and parameters using capital letters.

Sets

$\mathbf{J}$: Set of days in a week, numbered from 0 (Monday) to 6 (Sunday);
$\mathbf{D}_j^R$: Set of duties that must be assigned on day j;
$\mathbf{D}_j^S$: Set of standby duties that can be assigned (once or more) on day j;
$\mathbf{R}$: Set of all rosters;
$\mathbf{R}^i$: Set of rosters with i days off per position, $i \in \{2, 3\}$;
$\mathbf{R}_d$: Set of rosters to which duty d can be assigned;
$\mathbf{P}$: Set of all positions in all rosters;
$\mathbf{P}_r$: Set of positions in roster r, numbered from 0 to $|\mathbf{P}_r| - 1$;
$\mathbf{D}_{pj}^R$: Set of regular duties that can be assigned to position p on day j;

$\mathbf{D}_{pj}$: Set of regular and standby duties that can be assigned to position p on day j, $\mathbf{D}_{pj} = \mathbf{D}^R_{pj} \cup \mathbf{D}^S_j$;

$\mathbf{L}$: Set of bus lines;

$\mathbf{L}_d$: Set of bus lines served in duty d;

Parameters

W_d: Workload in duty d (in minutes);

S_d: Starting time of duty d (in minutes from the beginning of the week);

E_d: Ending time of duty d (in minutes from the beginning of the week);

I^{13}_d: Binary indicator equal to 1 if the workload of duty d exceeds 13 hours;

I_{dl}: Binary indicator equal to 1 if duty d serves bus line l, i.e., if $l \in L_d$;

A: Average workload per position $\left(A = \frac{\sum_{j\in\mathbf{J}}\sum_{d\in\mathbf{D}_j} W_d}{\sum_{r\in\mathbf{R}}|P_r|}\right)$;

M: A sufficiently large constant.

Variables

x^d_{pj}: Binary variable equal to 1 if duty d is assigned to position p on day j, and to 0 otherwise;

y^2_{pj}: Binary variable equal to 1 if a double day off is assigned to position p starting on day j, and to 0 otherwise;

y^1_{pj}: Binary variable equal to 1 if a single day off is assigned to position p on day j, and to 0 otherwise;

z_p: Nonnegative variable equal to the excess workload in position p with respect to the average workload per position, i.e., $z_p = \max\left\{0, \sum_{j\in\mathbf{J}}\sum_{d\in\mathbf{D}^R_{pj}} W_d x^d_{pj} - A\right\}$;

u_{pj}: Binary variable equal to 1 if the driver in position p is assigned to a duty on day j and on the following day, and to 0 otherwise;

v_{pj}: Nonnegative variable equal to the duration of the rest period between day j and the following day if the driver in position p is assigned to a duty on day j and on the following day, and to 0 otherwise.

Functions

$r(p)$: Roster of position p.

A roster r can be seen as the following cycle of pairs (p, j) of position p and day j: $(0,0) - (0,1) - \ldots - (0,6) - (1,0) - (1,1) - \ldots - (|P_r|-1,5) - (|P_r|-1,6) - (0,0)$. In the proposed model, we often need to identify which pair (p^*, j^*) is reached when moving k days forward

(if $k > 0$) or backward (if $k < 0$) from a pair (p, j) in the cycle. For instance, for a roster r with $|P_r| = 4$ positions, $(p^*, j^*) = (3, 6)$ if $(p, j) = (4, 0)$ and $k = -1$ and $(p^*, j^*) = (0, 2)$ if $(p, j) = (4, 5)$ and $k = 4$. In this regard, we introduce a vectorial function [1]

$$f(p, j, k) = \left(\left\lfloor p + \frac{(j+k)}{7} \right\rfloor \bmod |P_{r(p)}| \, , \, (j+k) \bmod 7 \right) \tag{5.1}$$

which returns this position-day pair (p^*, j^*) for a given triplet of position p, day j and number of days shifted k. With this function, writing $x^d_{f(4,5,4)}$ is equivalent to writing $x^d_{0,2}$ if $|P_{r(4)}| = 4$ as in the last example above.

5.3.2 Model

Using the notation presented above, the CBDRP can be modeled as the following mixed-integer linear program, which is called the *integrated model* hereafter:

$$\min \quad \sum_{p \in \mathbf{P}} z_p \tag{5.2}$$

$$\text{subject to :} \quad \sum_{j \in \mathbf{J}} \sum_{d \in \mathbf{D}^R_{pj}} W_d x^d_{pj} - A \leq z_p \qquad \forall p \in \mathbf{P} \tag{5.3}$$

$$z_p \geq 0 \qquad \forall p \in \mathbf{P} \tag{5.4}$$

$$\sum_{r \in \mathbf{R}_d} \sum_{p \in \mathbf{P}_r} x^d_{pj} = 1 \qquad \forall j \in \mathbf{J}, d \in \mathbf{D}^R_j \tag{5.5}$$

$$\sum_{d \in \mathbf{D}_{pj}} x^d_{pj} + y^1_{pj} + y^2_{pj} + y^2_{f(p,j,-1)} = 1 \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{5.6}$$

$$\sum_{j \in \mathbf{J}} y^2_{pj} = 1 \qquad \forall p \in \mathbf{P} \tag{5.7}$$

$$\sum_{j \in \mathbf{J}} y^1_{pj} = 1 \qquad \forall r \in \mathbf{R}^3, p \in \mathbf{P}_r \tag{5.8}$$

$$y^2_{f(p,j,-1)} + \sum_{k=0}^{6} \left(y^2_{f(p,j,k)} + y^1_{f(p,j,k)} \right) \geq 1 \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{5.9}$$

$$2u_{pj} \leq \sum_{d \in \mathbf{D}_{pj}} x^d_{pj} + \sum_{d \in \mathbf{D}_{f(p,j,1)}} x^d_{f(p,j,1)} \leq 1 + u_{pj} \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{5.10}$$

$$M \sum_{d \in \mathbf{D}_{pj}} x^d_{pj} \geq v_{pj} \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{5.11}$$

$$M \sum_{d \in \mathbf{D}_{f(p,j,1)}} x^d_{f(p,j,1)} \geq v_{pj} \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{5.12}$$

1. To always return a nonnegative value, we define the modulo function as follows: $a \mod n = a - n \left\lfloor \frac{a}{n} \right\rfloor$.

$$v_{pj} - \left(\sum_{d \in \mathbf{D}_{f(p,j,1)}} S_d x^d_{f(p,j,1)} - \sum_{d \in \mathbf{D}_{pj}} E_d x^d_{pj} \right) \leq M(1 - u_{pj}) \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{5.13}$$

$$v_{pj} - \left(\sum_{d \in \mathbf{D}_{f(p,j,1)}} S_d x^d_{f(p,j,1)} - \sum_{d \in \mathbf{D}_{pj}} E_d x^d_{pj} \right) \geq -M(1 - u_{pj}) \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{5.14}$$

$$v_{pj} \geq 540 u_{pj} \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{5.15}$$

$$600 u_{pj} - M \left(1 - \sum_{d \in \mathbf{D}_{f(p,j,2)}} x^d_{f(p,j,2)} \right) \leq v_{pj} \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{5.16}$$

$$\sum_{d \in \mathbf{D}_{f(p,j,3)}} S_d x^d_{f(p,j,3)} - \sum_{d \in \mathbf{D}_{pj}} E_d x^d_{pj}$$
$$\geq 3420 - M \left(3 - y^2_{f(p,j,1)} - \sum_{d \in \mathbf{D}_{pj}} x^d_{pj} - \sum_{d \in \mathbf{D}_{f(p,j,3)}} x^d_{f(p,j,3)} \right) \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{5.17}$$

$$720 \sum_{k=0}^{27} u_{f(p,j,k)} - M \left(1 - \sum_{d \in \mathbf{D}_{f(p,j,27)}} x^d_{f(p,j,27)} \right) \leq \sum_{k=0}^{27} v_{f(p,j,k)} \qquad \forall p \in \mathbf{P}, j \in \mathbf{J} \tag{5.18}$$

$$\sum_{j \in \mathbf{J}} \sum_{d \in \mathbf{D}_{pj}} I^{13}_d x^d_{pj} \leq 1 \qquad \forall p \in \mathbf{P} \tag{5.19}$$

$$\sum_{d \in \mathbf{D}_{pj}} I_{dl} x^d_{pj} + \sum_{d \in \mathbf{D}_{f(p,j,1)}} I_{dl} x^d_{f(p,j,1)} + \sum_{d \in \mathbf{D}_{f(p,j,2)}} I_{dl} x^d_{f(p,j,2)} \leq 2 \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}, l \in \mathbf{L} \tag{5.20}$$

$$x^d_{pj} \in \{0, 1\} \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}, d \in \mathbf{D}_{pj} \tag{5.21}$$

$$y^2_{pj}, y^1_{pj}, u_{pj} \in \{0, 1\} \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}. \tag{5.22}$$

Combined with constraints (5.3) and (5.4), the objective function (5.2) minimizes the sum over all positions of the workload exceeding the average workload per position in order to balance as much as possible the workload among the positions. Constraints (5.5) ensure that every regular duty is assigned to a position. Constraints (5.6) specify that a duty or a day off must be assigned to each position on each day. Rule 1 on the days off assignment per position is imposed through constraints (5.7) and (5.8). Constraints (5.9) limit the number of consecutive working days as dictated by Rule 2. In fact, it says that there must be at least one day off in every period of seven consecutive days.

Next, constraints (5.10) set the values of the u_{pj} variables in function of the x^d_{pj} variables, while the values of the v_{pj} variables are computed through the constraints (5.11)–(5.14). Recall that v_{pj} must be set to zero if a duty is not assigned to position-day pair (p, j) or position-day pair $f(p, j, 1)$. When not equal to zero, v_{pj} is equal to rest duration between the duties assigned on the position-day pairs $f(p, j, 1)$ and (p, j), i.e., the difference between the start time of the first of these duties and the end time of the second. As imposed through

constraints (5.15) and (5.16), it must be larger than 9 hours (540 minutes) according to Rule 3 or 10 hours (600 minutes) according to Rule 4, respectively. Rule 5 on a rest period of at least 57 hours (3420 minute) per position is enforced through constraints (5.17). Given that there is exactly one double day off per position and that this rest must include it, it becomes possible to model this rule using one constraint per day and position. Note that, in constraints (5.14) and (5.17), one week (i.e., 10800 minutes) must be added to all parameters S_d if position p differs from the position associated with $f(p, j, 1)$ or $f(p, j, 3)$, respectively.

Rule 6 on the average rest time on every period of 28 days is imposed through constraints (5.18). In these constraints, the first sum computes the total rest time between two consecutive working days while the second sum computes the total number of pairs of consecutive working days. When the last of the 28 days is a working day (i.e., $\sum_{d \in \mathbf{D}_{f(p,j,27)}} x^d_{f(p,j,27)} = 1$), the ratio of these two first sums gives the average rest time over these 28 days which must exceed 12 hours (720 minutes). Rule 7 limiting the number of duties with 13 hours or more per position is imposed by constraints (5.19), while Rule 8 on the maximum number of consecutive days servicing the same line is enforced by constraints (5.20). Finally, binary restrictions on the decision variables are expressed through (5.21) and (5.22).

5.3.3 Model characteristics

Assuming that $|\mathbf{J}|$ does not vary ($|\mathbf{J}| = 7$), model (5.2)–(5.22) contains $O(|\mathbf{P}|^2)$ variables and $O(|\mathbf{P}||\mathbf{L}|)$ constraints because $|\mathbf{D}^R_j| \leq |\mathbf{P}|$ for all $j \in \mathbf{J}$ in any feasible instance. Thus, the size of this model grows rapidly with the number of positions. In fact, as it will be reported in Section 5.5, the number of variables counts in hundreds of thousands of variables for instances with a few hundreds positions.

The integrated model (5.2)–(5.22) involves many big-M constraints, namely, constraints (5.11)–(5.14) and (5.16)–(5.18). Such constraints typically yield relatively loose linear relaxations and a large number of fractional-valued variables in their optimal solutions. Therefore, when solving them by a MILP solver, large branch-and-bound search trees should be expected. Note that these big-M constraints could be avoided by replacing each of them with a relatively large number of constraints as it will be proposed in Section 5.4.1, but given the already large size of the integrated model, we prefer to keep them.

One last characteristic of this model is that it contains a large number of symmetric solutions which can significantly increase the number of nodes in the search tree. Indeed, given that the weekly schedules are anonymous, it is always possible to shift all schedules assigned to a roster by one position to obtain an equivalent solution. Furthermore, many duties appear on different days of the week and, when a subset of them are assigned on different positions on

different days, it may be possible to permute them without violating any rest time constraints. Also, there often exist several duties that can be assigned to the same roster on the same day that have the same workload but maybe slightly different starting and ending times. These duties can often be permuted from one position to another without violating the rest time constraints. Finally, when such duties can be assigned to different duties, it increases the number of symmetric solutions.

To illustrate some of these symmetry cases, Tables 5.5 and 5.6 present an example with one roster, four positions and four duties that must be assigned overall on sixteen days. We assume that three days off must be assigned to each position and that all presented solutions satisfy the roster feasibility rules. The duties and their characteristics are given in Table 5.5. For each duty d, it provides its starting time S_d, its ending time E_d, its workload W_d and its set of days $\mathbf{J}_d$ on which it must be assigned. For convenience, the starting and ending times are given in the format $hh:mm$ and the workload in hours. Table 5.6 presents three symmetric solutions with the same cost. The second solution is obtained from the first by shifting the weekly schedules forward by one position, i.e., the schedule in position i, $i = 0, 1, 2$ is moved to position $i + 1$ and that of position 3 is moved to position 0. The third solution is obtained from the second by exchanging the duties d_1 and d_2 assigned to positions 1 and 3 on Wednesday and Thursday.

Table 5.5 The four duties and their characteristics.

Duty	S_d	E_d	W_d	$\mathbf{J}_d$
d_1	7:00	19:00	$9h$	Mon, Wed, Thu
d_2	6:00	18:00	$10h$	Tue, Wed, Thu, Sun
d_3	7:00	15:00	$7h$	Thu, Fri, Sat, Sun
d_4	9:00	18:00	$8h$	Mon, Tue, Fri, Sat, Sun

To break a part of the symmetry, one can impose for each roster r the following constraints:

$$\sum_{j \in \mathbf{J}} \sum_{d \in \mathbf{D}^R_{0j}} W_d x^d_{0j} \geq \sum_{j \in \mathbf{J}} \sum_{d \in \mathbf{D}^R_{pj}} W_d x^d_{pj}, \qquad \forall p \in P_r \setminus \{0\}, \tag{5.23}$$

which ensure that the first position in roster r has the maximum workload among the positions in this roster. Our computational experiments showed, however, that these constraints have little impact on the computational times.

Clearly, these characteristics of the integrated model make it difficult to solve to optimality, especially for large CBDRP instances. In the following section, we introduce, among others, a matheuristic that will turn out to also be useful for solving the problem to optimality.

Table 5.6 Three symmetric solutions.

Solution 1							
Position	Mon	Tue	Wed	Thu	Fri	Sat	Sun
0	OFF	OFF	d_1	d_2	d_3	d_4	OFF
1	d_1	d_2	OFF	OFF	OFF	d_3	d_4
2	OFF	d_4	d_2	d_1	OFF	OFF	d_3
3	d_4	OFF	OFF	d_3	d_4	OFF	d_2

Solution 2							
Position	Mon	Tue	Wed	Thu	Fri	Sat	Sun
0	d_4	OFF	OFF	d_3	d_4	OFF	d_2
1	OFF	OFF	d_1	d_2	d_3	d_4	OFF
2	d_1	d_2	OFF	OFF	OFF	d_3	d_4
3	OFF	d_4	d_2	d_1	OFF	OFF	d_3

Solution 3							
Position	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	d_4	OFF	OFF	d_3	d_4	OFF	d_2
2	OFF	OFF	d_2	d_1	d_3	d_4	OFF
3	d_1	d_2	OFF	OFF	OFF	d_3	d_4
4	OFF	d_4	d_1	d_2	OFF	OFF	d_3

5.4 Solution algorithms

In this section, we describe three solution algorithms for the CBDRP. The first algorithm is heuristic while the second and third ones are exact. In fact, the last algorithm combines the first two.

5.4.1 A two-step matheuristic

We propose a two-step matheuristic that schedules the days off before assigning the duties to the working days. This approach differs from the sequential ones listed in our literature review (see Table 5.1) by considering duty assignment variables in the days-off scheduling step. This allows to position the days off in the rosters while taking into account the workload balancing between the drivers. Let us describe each step of this algorithm which we denote TSM for two-step matheuristic.

Step 1: Days-off scheduling

In the days-off scheduling step, we consider the relaxation of the integrated model (5.2)–(5.22) obtained by dropping all the constraints related to Rules 3 to 8. Thus, the resulting model, called the *DOS model* hereafter for days-off scheduling, is given by (5.2)–(5.9), (5.21) and (5.22) but without the binary requirements on the u_{ij} variables that are not needed here. Note that the v_{ij} variables are neither necessary and that all constraints involving a big-M constant in the complete model have been omitted. The DOS model is solved using a commercial MILP solver. From the computed solution, we retain only the scheduled days off as there is no guarantee that the duty assignment satisfies all relaxed constraints.

One can expect that the DOS model is much easier to solve than the complete model (5.2)–(5.22) for the following reasons. First, it involves less variables and much less constraints. In fact, the number of constraints is drastically reduced by about 90% as it will be shown in Section 5.5. Second, all constraints with a big-M constant have been removed. Such constraints are known to yield looser linear relaxations. Finally, given the much smaller number of constraints, the number of fractional-valued variables in the linear relaxation solution at the root node of the search tree should be substantially reduced. Typically, less fractional-valued variables induce less cuts to add and less branch-and-bound nodes to explore. In this case, the large number of symmetric solutions should have less impact on the solution process.

Step 2: Duty assignment

Given the days off computed in the first step, the duty assignment step consists of assigning duties to the working days in the rosters while taking into account only the constraints that depend on the duty assignment. Therefore, Rules 1 and 2 are not relevant. This duty assignment problem can be formulated with the integrated model (5.2)–(5.22) after fixing all y variables to the values they take in the computed solution to the DOS model and removing the unnecessary constraints. Here we propose another model that exploits the knowledge of the days off and avoids all big-M constraints.

In addition to the previous notation, we need the following one.

Sets:

$\mathbf{J}_p^W$: Set of working days in position p;

$\mathbf{J}_p^9$: Set of working days in position p which are immediately followed by a working day and a day off (Rule 3 must be checked);

$\mathbf{J}_p^{10}$: Set of working days in position p which are immediately followed by at least two consecutive days (Rules 4 and 8 must be checked on those days);

$\mathbf{J}_p^{57}$: Set containing the single working day in position p which is followed by a double day off (Rule 5 must be checked);

$\mathbf{J}_p^{12}$: Set of days in position p which start a sequence of 28 consecutive days ending with a working day (Rule 6 must be checked);

$\mathbf{K}_{pj}$: Subset of $\{0, 1, \ldots, 27\}$ indicating the pairs of consecutive working days in a sequence of 28 consecutive days starting in position-day pair (p, j), i.e., $k \in \mathbf{K}_{pj}$ if working days are assigned to both position-day pairs $f(p, j, k)$ and $f(p, j, k + 1)$;

$\mathbf{T}_{pj}$: Set of all ending times of the duties in $\mathbf{D}_{pj}$.

Parameters:

F_{dt}^+: Binary indicator equal to 1 if duty d finishes at time t or after;

B_{dt}^-: Binary indicator equal to 1 if duty d begins before time t (excluding time t);

U_{pj}: Total number of working days immediately followed by a working day in a sequence of 28 consecutive days starting in position-day pair (p, j).

With this notation, we model the duty assignment problem as the following mixed-integer linear program, called the *DA model* hereafter:

$$\min \quad \sum_{p \in \mathbf{P}} z_p \tag{5.24}$$

$$\text{subject to} \quad \sum_{j \in \mathbf{J}} \sum_{d \in \mathbf{D}_{pj}^R} W_d x_{pj}^d - A \leq z_p \qquad \forall p \in \mathbf{P} \tag{5.25}$$

$$z_p \geq 0 \qquad \forall p \in \mathbf{P} \tag{5.26}$$

$$\sum_{r \in \mathbf{R}_d} \sum_{p \in \mathbf{P}_r} x_{pj}^d = 1 \qquad \forall j \in \mathbf{J}, d \in \mathbf{D}_j^R \tag{5.27}$$

$$\sum_{d \in \mathbf{D}_{pj}} x_{pj}^d = 1 \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}_p^W \tag{5.28}$$

$$\sum_{d \in \mathbf{D}_{pj}} F_{dt}^+ x_{pj}^d + \sum_{d \in \mathbf{D}_{f(p,j,1)}} B_{d,t+540}^- x_{f(p,j,1)}^d \leq 1 \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}_p^9, t \in \mathbf{T}_{pj} \tag{5.29}$$

$$\sum_{d \in \mathbf{D}_{pj}} F_{dt}^+ x_{pj}^d + \sum_{d \in \mathbf{D}_{f(p,j,1)}} B_{d,t+600}^- x_{f(p,j,1)}^d \leq 1 \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}_p^{10}, t \in \mathbf{T}_{pj} \tag{5.30}$$

$$\sum_{d\in\mathbf{D}_{pj}} F^{+}_{dt} x^{d}_{pj} + \sum_{d\in\mathbf{D}_{f(p,j,3)}} B^{-}_{d,t+3420} x^{d}_{f(p,j,3)} \leq 1 \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}^{57}_{p}, t \in \mathbf{T}_{pj} \qquad (5.31)$$

$$\sum_{k\in\mathbf{K}_{pj}} \left(\sum_{d\in\mathbf{D}_{f(p,j,k+1)}} S_d x^{d}_{f(p,j,k+1)} - \sum_{d\in\mathbf{D}_{f(p,j,k)}} E_d x^{d}_{f(p,j,k)} \right) \geq 720 U_{pj} \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}^{12}_{p} \qquad (5.32)$$

$$\sum_{j\in\mathbf{J}^{W}_{p}} \sum_{d\in\mathbf{D}_{pj}} I^{13}_{d} x^{d}_{pj} \leq 1 \qquad \forall p \in \mathbf{P} \qquad (5.33)$$

$$\sum_{d\in\mathbf{D}_{pj}} I_{dl} x^{d}_{pj} + \sum_{d\in\mathbf{D}_{f(p,j,1)}} I_{dl} x^{d}_{f(p,j,1)} + \sum_{d\in\mathbf{D}_{f(p,j,2)}} I_{dl} x^{d}_{f(p,j,2)} \leq 2 \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}^{10}_{p}, l \in \mathbf{L} \qquad (5.34)$$

$$x^{d}_{pj} \in \{0,1\} \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}^{W}_{p}, d \in \mathbf{D}_{pj}. \qquad (5.35)$$

Objective function (5.24), constraints (5.25)–(5.28) and (5.33)–(5.35) correspond to (5.2)–(5.6) and (5.19)–(5.21) in the integrated model, except that constraints (5.28) and (5.34) are restricted to the days where they apply. Constraints (5.29)–(5.31) impose Rules 3, 4 and 5, respectively. They are all based on the same rationale and, therefore, we explain only the first constraint set. For a given possible duty ending time t at position-day pair (p, j), a constraint (5.29) stipulates that it is not possible to assign more than one duty to position-day pairs (p, j) and $f(p, j, 1)$ that overlaps with time interval $[t, t + 540[$ without violating Rule 3 (at least 9 hours of rest between two consecutive duties). In fact, this constraint is lifted by also including all variables associated with the duties that can be assigned to pair (p, j) and start at time $t + 540$ or after. Notice here that 10800 minutes must subtracted from $t+540$ (or $t+600$ or $t+3420$) when $t+540 > 10800$. Finally, constraints (5.32) ensure that Rule 6 is satisfied. The left-hand side computes the total rest time in the considered sequence of 28 days which must be greater than 12 hours multiplied by the number of rests between two consecutive duties in this sequence. As for the integrated model, one week (i.e., 10800 minutes) must be added to all parameters S_d if the position of pair $f(p, j, k)$ differs from the position of pair $f(p, j, k + 1)$.

Observe that the minimum rest time Rules 3, 4 and 5 could have been modeled differently, namely, by computing the rest time as the difference between the start time and the end time of two duties and imposing that this rest time be greater or equal to the minimum rest time (like in the integrated model). For example, for position/day pairs (p, j) and $f(p, j, 1)$ that requires a minimum rest time of 600 minutes, such a constraint would write as:

$$\sum_{d\in\mathbf{D}_{f(p,j,1)}} S_d x^{d}_{f(p,j,1)} - \sum_{d\in\mathbf{D}_{pj}} E_d x^{d}_{pj} \geq 600. \qquad (5.36)$$

We call these constraints the difference-duty-times (DDT) constraints. We have rather chosen to model Rules 3, 4 and 5 by constraints (5.29)–(5.31) which yield tighter linear relaxations

than the DDT constraints and speed up the solution process by reducing the number of cuts generated or the number of nodes explored in the search tree. On the other hand, this way of modeling the rest time rules requires a large number of constraints and becomes possible in the DA model because of the fixed days off. We also tried a similar modeling for the integrated model but, in this case, the benefits of obtaining tighter linear relaxations were outweighed by the burden of dealing with a too large number of additional constraints. This type of modeling for rest time rules has also been used by other authors for the CBDRP. However, none have proposed such a strong version of the constraints, where two subsets of variables are summed, namely, one associated with day t and another with the other day (associated with time $t + 540$, $t + 600$ or $t + 3420$). Indeed, Moz et al. (2009) proposed a model with rest time constraints involving one variable on day t and a subset on the other day, whereas the models of Hartog et al. (2009) and Xie and Suhl (2015) involve constraints with a single variable on both days.

The DA model (5.24)–(5.35) is solved using a commercial MILP solver. In general, it is much easier to solve than the integrated because it has a reduced number of variables, no big-M constraints, and prone to less symmetry because of the fixed days off. Combining the computed solution of the DA model with the days off schedule obtained in the first step of the TSM yields a complete feasible solution to the CBDRP.

Observe that, in the second step of the TSM, the duty assignment problem may be infeasible even if the CBDRP is feasible. Indeed, because the fixed days off schedule was computed while relaxing some constraints to the CBDRP, there is no guarantee that a feasible duty assignment can be found with this schedule. However, given the flexibility provided by the large number of duties that can be assigned to each roster and vice versa, this situation did not occur in our computational tests. It should also be noted that this flexibility helps finding optimal or near-optimal solutions with the TSM.

5.4.2 Two exact algorithms

We propose two exact algorithms for the CBDRP. The first one simply consists of solving the integrated model (5.2)–(5.22) using a commercial MILP solver. This algorithm is denoted INT for integrated-model-based algorithm. It is well-known that state-of-the-art MILP solvers have several components which can help to speed up the overall solution process. One of them is a set of heuristics that searches for a good feasible solution and that is invoked at various stages of the solution process and, in particular, just after solving the first linear relaxation. A feasible solution provides an upper bound on the optimal value which can be used to fix the values of some variables according to their reduced costs (see, e.g.,

Wolsey and Nemhauser (1999), p. 389). A better upper bound yields a larger number of variables that can be fixed at one of their bounds, typically at 0 in our case. Fixing a large number of variables may have several positive impacts on the solution process, including further preprocessing to fix other variables and remove redundant constraints, faster linear relaxation computational times incurred by the problem size reduction, increased opportunity to find a better solution around this initial solution when applying the solver heuristics, and tighter subsequent linear relaxations. As our computational experiments will show in the next section, the heuristic embedded in the state-of-the-art MILP solver that we used (the CPLEX MILP solver) was unable to find good-quality feasible solutions and to solve the largest instance that we tested.

As a second exact algorithm, we propose to combine the TSM and INT algorithms. The TSM is first executed to compute a high-quality feasible solution which is then inputted as an initial solution to the INT algorithm. The INT algorithm can then proceed to variable fixing early in the solution process and benefit from the ensuing positive impacts. This algorithm is denoted INTIS in the following for integrated-model-based algorithm with an initial solution.

5.5 Computational results

To compare the performance of the three proposed algorithms, we conducted computational experiments on two real-world instances and on five additional instances derived from them. In this section, we first describe these instances. Then, we report and discuss the computational results obtained from these experiments. Finally, we compare our results with those produced by the current solution approach used by our industrial partner.

All proposed models were implemented using the IBM concert technology and solved using the CPLEX MILP solver version 12.7.0.0 with default parameter values. The computational tests were conducted on a Linux machine with an Intel Xeon E5-2637 V2 processor, clocked at 3.5 GHz and a RAM of 128 GB.

5.5.1 Instances

Our industrial partner provided two real-world datasets, denoted by A and G, that come from two different European public transit companies. Instance A contains 14 rosters, 202 positions, 881 duties and 14 bus lines; the rosters differ by the number of days off per position (2 or 3), by the number of positions (between 6 and 35) and by the duties they can operate though most duties can be assigned to almost all rosters. Instance G contains 2 rosters, 333

positions, 1509 duties and 28 bus lines: two days off must be assigned to the roster with 180 drivers whereas three days off must be assigned to that with 153 drivers. For this instance, all duties can be assigned to both rosters.

From these two datasets, we randomly generated five additional smaller-sized instances as follows:

1. From dataset A, we created three instances by choosing a subset of the rosters in instance A such that these rosters contain in total between 100 and 150 positions. From dataset G which contains only two rosters, we created two instances with two rosters containing a total of 165 and 180 positions, respectively.
2. For each instance i, we computed a feasible days off schedule and, for each day of the week j, established the number n_{ij} of duties to assign according to this schedule.
3. For each instance i and each day of the week j, we selected randomly a set of n_{ij} duties that can be assigned to at least one selected roster on day j while ensuring that the resulting instance is feasible.

Table 5.7 presents the instances and their characteristics, namely, the dataset from which it was generated, the number of rosters, the number of positions, and the number of duties it contains. The name of an instance $I_{S,R,P}$ specifies the original dataset (S), the number of rosters (R) and the number of positions (P). Note that the last two instances $I_{A,14,202}$ and $I_{G,2,333}$ correspond to the complete datasets. Additional details on these instances are provided in Appendix A.

5.5.2 Comparative results for the INT, TSM and INTIS algorithms

We begin by presenting the results obtained by the INT algorithm, which is the most straightforward algorithm among the proposed algorithms. These results are reported in Table 5.8. For each instance, this table indicates the numbers of variables and constraints in the integrated model, the number of branch-and-bound nodes explored (excluding the root node of the search tree) during the solution process, the number of cuts applied, the total computational time in seconds, the relative integrality gap before and after adding cuts at the root node (G^B and G^A), and the cost of the computed solution (in minutes). These integrality gaps are computed as $(z^* - LB)/z^*$, where z^* is the cost of the solution found with the INTIS algorithm (see Table 5.8) and LB is the optimal value of the linear relaxation before or after adding cuts. Note that the solution process was stopped after twelve hours of computing time (and only three branch-and-bound nodes) for the largest instance $I_{G,2,333}$, with an optimality gap of $(34682 - z^*)/34682 = 60.1\%$. All the other instances were solved within this time limit.

Table 5.7 Instances and their characteristics.

Instance	Dataset	#Rosters	#Positions	#Duties
$I_{A,5,110}$	A	5	110	490
$I_{A,6,124}$	A	6	124	414
$I_{A,10,146}$	A	10	146	640
$I_{G,2,165}$	G	2	165	745
$I_{G,2,180}$	G	2	180	813
$I_{A,14,202}$	A	14	202	881
$I_{G,2,333}$	G	2	333	1509

Table 5.8 Results for the INT algorithm.

	#Variables	#Constraints	#Nodes	#Cuts	Time (s)	G^B (%)	G^A (%)	Cost
$I_{A,5,110}$	57645	20623	14285	226	2386	0.76	0	4966
$I_{A,6,124}$	51964	19844	19921	1931	5259	0.14	0	3461
$I_{A,10,146}$	98414	27342	16935	998	7005	0	0	6291
$I_{G,2,165}$	128537	47125	88147	480	41516	76.11	0	6222
$I_{G,2,180}$	152462	51409	80099	1232	40687	67.43	0	7853
$I_{A,14,202}$	189692	40691	17947	614	11969	0	0	18532
$I_{G,2,333}$	521813	99773	3	1082	> 43200	3.99	0.1	34682

From these results, we observe that the initial integrality gaps (i.e., before adding cuts) are relatively small for all instances, except for instances $I_{G,2,165}$ and $I_{G,2,180}$. In all cases, the cuts are very effective at closing this gap. In fact, for the first six instances, they completely close the gap. Nevertheless, a very large number of nodes in the search tree needs to be explored for finding an optimal solution. As discussed in Section 5.3.3, this can be explained by the large number of symmetric solutions. Without this symmetry, the computational times would be much smaller.

Next, in Tables 5.9 and 5.10, we present the results obtained in each step of the TSM. The days off scheduling results (Table 5.9) show that, for all instances, the DOS model can be solved in less than 20 minutes. Remark that no branching was required and that a much smaller number of cuts were generated than for the integrated model (see Table 5.8). As expected, relaxing a large number of constraints from the integrated model to obtain the DOS model yields linear relaxation solutions with much less fractional-valued variables. For example, there are 4462 and 7005 fractional-valued variables in the linear relaxation solutions of the integrated model for the complete instances $I_{A,14,202}$ and $I_{G,2,333}$, respectively, while there are only 2363 and 4229 such variables for the DOS model. Clearly, this helps finding an optimal solution much more rapidly. One can observe that no branching is performed

for instance $I_{A,14,202}$ even if there remains an optimality gap of 0.01%, which is less than or equal to the default tolerance of 0.01% accepted by CPLEX. One can observe that, for the first five instances, the cost of the computed solution is the same as that of the solution produced by the INT algorithm and differs by only one for the sixth instance. It shows that the constraints (Rules 3 to 8) that are relaxed to define the DOS model are not much binding because the large set of duties offers great flexibility to satisfy them.

Table 5.10 gives the results obtained in the duty assignment step of the TSM. These results are similar to those obtained for the first step: all instances are solved within short computational times except for the largest instance which required around 40 minutes. One can observe that, within the CPLEX tolerance, there is no integrality gap (even before adding cuts) for all instances. Nevertheless, some cuts are needed to find an optimal solution, but no branch-and-bound nodes. With fixed days off, a relatively easy solution process was expected. Surprisingly, the costs of the solutions computed for the first and second steps are all equal for the first five instances and differ by one for the last two. This proves that the TSM has computed optimal solutions for all tested instances within the CPLEX optimality gap tolerance.

To assess some modeling choices we made for the TSM, we performed two additional tests on the complete instances $I_{A,14,202}$ and $I_{G,2,333}$. First, in the days off scheduling step, we transformed the problem into a feasibility one, i.e., only a feasible day off schedule is sought, ensuring that all duties can be assigned. In this case, the DOS model has no objective function and the constraints (5.3) and (5.4) are omitted. With this strategy, the TSM produces worse solutions: the costs of the computed solutions for the instances $I_{A,14,202}$ and $I_{G,2,333}$ are 18688 and 15647, respectively, instead of 18532 and 13531 when the objective function (5.2) is considered in the DOS model. These results clearly show the importance of considering the right objective function in the first step of the TSM. In the second test, we replaced in the DA model the minimum rest time constraints (5.29)–(5.31) by DDT constraints (see Section 5.4.1). We obtained the same solutions but the computational times of the duty assignment step of the TSM increased from 141 and 2368 seconds for $I_{A,14,202}$ and $I_{G,2,333}$, respectively, to 275 seconds and more than 12 hours. When modeling these rules as in Moz et al. (2009) or as in Hartog et al. (2009) and Xie (2014) (see Section 5.4.1), the computational times are also much larger for instance $I_{G,2,333}$, namely, close to 5 and 12 hours, respectively. Thus, modeling the rest time rules adequately allows a substantial time reduction.

In Table 5.11, we report the results obtained by the INTIS algorithm which considers the initial solution produced by the TSM. Note that the reported statistics exclude the computation of the initial solution. In particular, the computational time does not include the time

Table 5.9 Results of the days off scheduling step of the TSM.

Instance	#Variables	#Constraints	#Nodes	#Cuts	Time (s)	G^B (%)	G^A (%)	Cost
$I_{A,5,110}$	55727	2033	0	10	9	0.76	0	4966
$I_{A,6,124}$	50228	1912	0	51	23	0.14	0	3461
$I_{A,10,146}$	95950	2668	0	23	28	0	0	6291
$I_{G,2,165}$	125604	3070	0	133	111	76.11	0	6222
$I_{G,2,180}$	149263	3349	0	230	78	67.43	0	7853
$I_{A,14,202}$	186150	3725	0	3	49	0.01	0.01	18533
$I_{G,2,333}$	515891	6200	0	252	1001	3.98	0	13531

Table 5.10 Results of the duty assignment step of the TSM.

Instance	#Variables	#Constraints	#Nodes	#Cuts	Time (s)	G^B (%)	G^A (%)	Cost
$I_{A,5,110}$	40319	44461	0	39	15	0	0	4966
$I_{A,6,124}$	34791	40695	0	32	11	0	0	3461
$I_{A,10,146}$	67268	62229	0	66	42	0	0	6291
$I_{G,2,165}$	89809	102694	0	113	310	0	0	6222
$I_{G,2,180}$	106747	107416	0	79	168	0	0	7853
$I_{A,14,202}$	128715	113080	0	174	141	0	0	18532
$I_{G,2,333}$	352868	235891	0	101	2368	0	0	13531

for computing the initial solution. The total time will be discussed afterwards. From these results, we observe that providing high-quality initial solutions allow to solve to optimality the integrated model for all instances in less than three hours of computational time. Only the first instance requires very limited branching; the cuts completely close the gaps for four instances; no cuts, nor branching was needed to prove the optimality of the initial solution for two instances. Notice that the computational times are much smaller than those obtained without considering an initial solution (see Table 5.8).

Finally, let us compare the total computational times of the INT and INTIS algorithms, including the time required to compute the initial solution in the latter algorithm. These times are reported in Table 5.12 together with the gain in computational time (in percentage) achieved by the INTIS algorithm over the INT algorithm. The gain exceeds 90% for five of the seven instances and reaches a maximum of 99% for one instance. Obviously, these large speedups depend on the quality of the computed initial solutions.

5.5.3 Comparison with the current industrial solution algorithm

In this section, we compare the performance of the INTIS algorithm and that of the algorithm currently used by our industrial partner. Their algorithm, denoted IND hereafter, is a two-step matheuristic which also solves a days off scheduling problem first and a duty assignment

Table 5.11 Results for the INTIS algorithm.

	#Variables	#Constraints	#Nodes	#Cuts	Time† (s)	G^B (%)	G^A (%)	Cost
$I_{A,5,110}$	57645	20623	6	392	486	0.76	0.76	4966
$I_{A,6,124}$	51964	19844	0	627	172	0.14	0	3461
$I_{A,10,146}$	98414	27342	0	0	14	0	0	6291
$I_{G,2,165}$	128537	47125	0	665	249	76.11	0	6222
$I_{G,2,180}$	152462	51409	0	642	3328	67.43	0	7853
$I_{A,14,202}$	189692	40691	0	0	53	0	0	18532
$I_{G,2,333}$	521813	99773	0	1223	8270	3.99	0	13532

†: Excluding the time required to compute the initial solution

Table 5.12 Total computational times for the INT and INTIS algorithms.

	Time (s)		
	INTIS	INT	Gain (%)
$I_{A,5,110}$	510	2386	79
$I_{A,6,124}$	206	5259	96
$I_{A,10,146}$	84	7005	99
$I_{G,2,165}$	670	41516	98
$I_{G,2,180}$	3574	40687	91
$I_{A,14,202}$	243	11969	98
$I_{G,2,333}$	11639	> 43200	> 73

problem second. In the first step, the problem is formulated as a mixed-integer linear program (see Gendron (2012)) which is solved using a commercial MILP solver. Contrarily to the first step of the TSM, it involves no duty assignment variables. In the second step, the duty assignment problem is decomposed into a sequence of subproblems, each one being defined by a subset of positions and days. Each subproblem is formulated as a network-flow problem with side constraints and solved by a MILP solver. The subproblems are solved sequentially and, to ensure the continuity of the overall solution, constraints derived from the solutions of the subproblems previously solved may be imposed to a subproblem.

To compare the INTIS and IND algorithms, we report in Table 5.13 the costs of the solutions obtained by both algorithms on the full instances $I_{A,14,202}$ and $I_{G,2,333}$. This table also gives the gain (in percentage) obtained by the INTIS solution, the average workload per position (A which is the same for both algorithms), and the minimum and maximum workload over all positions for the solutions computed by both algorithms. Note that the solutions produced by the IND algorithm were computed in approximately one and three hours for the instances $I_{A,14,202}$ and $I_{G,2,333}$, respectively.

Table 5.13 Comparative results between the INTIS and IND algorithms.

	Cost			Workload per position (min.)		
	INTIS	IND	Gain (%)	A	INTIS [Min, Max]	IND [Min, Max]
$I_{A,14,202}$	18532	19395	4.5	2113	[2002, 2428]	[1962, 2309]
$I_{G,2,333}$	13532	16173	16.3	2305	[2151, 2552]	[1936, 2621]

From these results, we observe significant gains realized by the proposed INTIS algorithm, namely, 4.5% (or 863 minutes) and 16.3% (or 2641 minutes) for the $I_{A,14,202}$ and $I_{G,2,333}$, respectively. Looking at the minimum and maximum workloads for the instance $I_{G,2,333}$, one can notice that the solution computed by the INTIS algorithm is better balanced than the IND solution, which is the goal of the CBDRP. For the other instance, the minimum and maximum workloads do not highlight this though the total number of minutes exceeding the average workload per position is much smaller for the INTIS solution.

5.6 Conclusion

In this paper, we addressed the CBDRP defined by our industrial partner and which does not involve shift-related rules but offers wide flexibility for duty assignment. We first modeled this problem as a mixed-integer linear program. Then, we proposed three algorithms to solve it. One is a two-step matheuristic, the other two consist of solving the integrated model using a MILP solver with or without an initial solution. Our computational experiments carried on two real-world instances and five additional instances derived from them showed that the two-step matheuristic yields, for these instances, optimal solutions in relatively short computational times (up to 3.5 hours for instances involving up to 333 drivers). Furthermore, when providing these solutions as initial solutions to the MILP solver, very large speedups (up to 99%) were achieved to solve the CBDRP to proven optimality. Finally, we compared the computed optimal solutions with those produced by the matheuristic currently used by our industrial partner and observed significant gains in the solution cost, yielding a much better balanced workload between the positions.

Direct extensions of this work consist of designing efficient algorithms for solving the CBDRP with a variable number of positions per roster and the integrated duty scheduling and cyclic bus driver rostering problem in which the duties would be determined at the same time as the rosters.

Acknowledgements

We are thankful to Charles Fleurent and Marc Gendron from GIRO Inc. for introducing this problem to us, providing the data, and discussing our progress throughout the project. We gratefully acknowledge the financial support provided by GIRO Inc. and the Natural Sciences and Engineering Research Council of Canada under the grant RDCPJ 4634633-14.

CHAPITRE 6 ARTICLE 2 : PREFERENCE-BASED BUS DRIVER ROSTERING PROBLEM WITH FIXED DAYS OFF

Cet article a été écrit par S.Er-Rbib, G.Desaulniers, I.Elhallaoui et P.Munroe, et a été soumis en Mars 2020 à la revue *Public transport*.

6.1 Introduction

Crew rostering problems arise in several domains, namely in public transportation where creating good-quality schedules is crucial, not only from the costumer point of view, but also for the employees. A high-quality schedule should, therefore, reduce the operational costs and maximize crew satisfaction by taking into account their preferences. In this paper, we focus on a rostering problem variant arising in public transit that considers some driver preferences.

Because of the complexity of the planning process in public transit, decisions are performed at different planning levels (Desaulniers and Hickman, 2007): strategic, tactical and operational planning. Strategic planning involves making long term decisions such as the network design, where all the routes, bus lines and stops are determined. On the other hand, tactical planning concerns intermediate steps in the planning process. In this phase, service quality represents the main concern. Travel frequencies, i.e., the number of departures on each line and their schedule, are set in order to maximize customer service and facilitate connection between several buses. The results of this phase serve as the input data of the operational planning phase, the main purpose of which is to minimize the total cost for operating the service offer established in the previous phases. The operational problems are generally solved on a monthly to weekly basis, and sometimes even on a daily basis. They include building bus and employee schedules, bus parking and maintenance plans among others.

Bus scheduling is the first step in operational planning. A feasible bus schedule consists of a journey starting from the depot, active trip sequences separated by deadhead trips, and a trip back to the depot. The second step is duty scheduling. A duty is defined as an ordered sequence of trips and pauses performed by a driver during a weekday. Anonymous duties are built to ensure complete coverage of the bus schedule. At this level of planning, the number of duties is minimized as well as the total workload.

The next step consists in assigning the duties to the drivers, which is done by solving the rostering problem. This problem can be modeled as a set partitioning or set covering problem.

It aims at assigning the days off and the anonymous duties to the available drivers in a particular depot with the twofold objective of minimizing the number of unassigned duties and balancing the workload among the drivers. This results in schedules that are called rosters.

The validity of a roster is determined by safety regulations and collective agreement rules. For example, a driver cannot work more than a certain number of days per week. In the usual approach, the rostering problem is solved in two steps: days off scheduling and duty assignment. In this work, we propose a way to improve the quality of the rosters obtained from the second step. More precisely, we will solve the duty assignment problem with consideration of driver preferences (DAPDP), which consists of building driver work schedules over a one-week horizon while respecting all the feasibility rules and satisfying as much as possible some driver preferences.

6.1.1 Literature review

In the last few decades, crew rostering problems have received wide attention from the scientific community as well as practitioners (Van den Bergh et al., 2013). They occur in many domains of application, including mainly healthcare, production and transportation (Ernst et al., 2004b,a). Even though crew rostering problems vary from one domain of application to another, they always aim at building employee schedules over a certain planning horizon. These schedules specify the sequences of duties and days off according to which employees will work. Moreover, they must comply with safety regulations and collective rules that are generally specific to the sector of application.

In public transit, crew rostering still represents a vital step in the operational planning process. There are several reasons for this. The size of the problem has grown through the years due to the fast growth of bus traffic. In addition to this, the labor rules have become more complex. Finally, the crew costs are a challenging component in the rostering problem, and it is important to consider driver preferences while devising the schedules (Xie, 2014).

As stated in Desaulniers and Hickman (2007), while, in public transit, a duty is a daily task assigned to a driver on a given day, in the world of air/rail transportation, it is a pairing that can cover several days. This makes the number of duties to be considered in public transit quite large. In addition, since drivers can operate almost any vehicle, they are generally able to perform any duty and rostering is not separable by vehicle fleet. All of this increases the complexity of the rostering problem and the size of real-world instances as compared to the air/rail transportation context. Furthermore, given that most of the duties are different from one day to another (starting times, ending times, workload, length, ...), drivers can be

assigned a variety of schedules. Moreover, not only tasks can be different from one day to another, but also schedules may vary from one week to another. The driver rostering problem is then modeled as a set partitioning problem, where every driver is assigned one duty on a working day. In other domains such as health care, nurses and other physicians work on a shift basis. This type of rostering generally corresponds to a set covering problem (Erhard et al., 2018), in which the goal is to cover shifts with employees, subject to a set of hard and soft constraints. These soft constraints usually correspond to task assignment preferences. Therefore, this review will not be restricted to public transport, but will also include a few other domains of application in which employee preferences are considered in the rostering problem. Most advances may be relevant for the rostering problem with regard to driver preferences.

In recent years, taking into account employee preferences has become an important aspect of rostering. Hanne et al. (2009) mention the advantages of assigning flexible schedules to employees. They explain how preferences can reduce absenteeism and increase morale, which results in less operational re-scheduling and lower operational costs.

Many papers have tackled rostering problems with a consideration of employee preferences, especially in the context of nurse scheduling. In this field, preferences are usually expressed as desirable activities or (negatively) as undesirable sequences of activities. Some authors (Knust and Xie, 2019; Bard and Purnomo, 2005a,b) seek to minimize the number of preference violations in the objective function. The non-satisfaction of a preference corresponds to the presence of an undesirable shift, in which two days off are separated by an isolated working day. This problem is solved using different approaches: Knust and Xie (2019) uses network flow formulation and simulated annealing, whereas Bard and Purnomo (2005a,b) resorts to integer programming and column generation.

In the same sector of application, Bard and Purnomo (2007) and Purnomo and Bard (2007) add to the previous preferences the undesirable shift sequences (e.g., a day-evening-day sequence without an intervening day off). In both papers, the objective is to minimize the related soft constraint penalties. Bard and Purnomo (2007) solves the problem using a variable fixing approach instead of the traditional branch and bound. On the other hand, Purnomo and Bard (2007) uses a branch and bound algorithm combined with a Dantzig-Wolfe decomposition. Another way to tackle preferences for nurse scheduling is presented in Wong et al. (2014), where the authors select the most important and requested preferences (e.g., three consecutive evening shifts at most) and consider them as hard constraints while penalizing non-satisfaction of the remaining preferences in the objective function (i.e., as soft constraints). The schedules are then built with the help of a local search heuristic.

In other domains, preferences consisting in forbidding some rotation/work sequences are considered. For example, Borndörfer et al. (2015) penalize backward rotations (i.e., when the duty on day $j+1$ starts earlier than the duty on day j) in a toll enforcement problem. The rostering problem is then solved using integer programming.

The focus of the current paper is on the rostering problem in public transport. In railway/subway transport, it is common to take into account driver preferences while devising work schedules. For example, Hartog et al. (2009) favor adjacent days off and assignment of duties of the same type in the same work sequence. In the same vein, Sodhi and Norris (2004) forbid excessive overtime and prefer schedules in which early duties start progressively later, and late duties end progressively earlier as we move towards the end of the week. In both works, integer programming is used to solve the problem. In another example of railway driver rostering, Jütte et al. (2017) penalize the assignment of undesirable duties in the objective function and solve the resulting model with a column generation-based algorithm.

In bus driver rostering, undesirable shifts or duties are generally weighted and penalized in the objective function. This approach is used, for example, in Kisielewski et al. (2019), Lezaun et al. (2006), Mesquita et al. (2015), Nurmi et al. (2011), and Xie and Suhl (2015), where the authors seek to minimize the number of unfavorable sequences of duties or to distribute them fairly among the drivers. The methods used vary from integer programming to genetic algorithms. For example, Nurmi et al. (2011) uses an evolutionary heuristic to solve the problem. In their model, in addition to avoiding the assignment of single days off and single working days, they let the drivers choose whether or not they prefer to have the same shift before and after a days-off sequence. Moz et al. (2009) also uses an evolutionary heuristic to solve a rostering problem in which they aim at minimizing the maximum overtime per driver as well as the total number of drivers with incomplete workload. In their approach, preferences are not penalized, but are rather expressed implicitly as the two components of a vector-valued objective function.

6.1.2 Contribution and organization

In this paper, we propose a new mathematical programming approach to the cyclic bus driver rostering problem in which preferences of the drivers are taken into account. We introduce new duty preference types that can be seen as a duty analogue to the requirement of specific shift patterns (see, e.g., Xie et al., 2012). To the best of our knowledge, no model considering these preference types while solving the duty assignment problem for bus drivers has ever been proposed in the literature. The main contributions of this paper are as follows:

— We introduce a mixed-integer linear program (MILP) that models the DAPDP. Using

a commercial solver, this MILP can be solved to optimality for small-sized instances with up to 124 drivers and 490 duties while keeping the workload among the positions close to the optimal one.

— We propose a first matheuristic that solves the DAPDP by preassigning duties to rosters based on an optimal solution to the duty assignment problem (without consideration of preferences). We show that this algorithm can generate, within short computational times, high-quality solutions for medium- and large-sized instances of the DAPDP with up to 202 drivers and 881 duties.

— We devise a second matheuristic that first decomposes the original DAPDP and then solve the resulting model. With this algorithm, high-quality solutions of real-world large instances (with up to 333 drivers and 1509 duties) can be obtained in less than five minutes.

The rest of the paper is organized as follows. In Section 6.2, we state precisely the DAPDP, to which this paper is dedicated. This problem is then formulated as a mixed-integer program in Section 6.3. Next, we propose two matheuristics to solve this model in Section 6.4. The results of computational experiments that were carried out on real-world instances in order to test these algorithms are then presented in Section 6.5. Finally, we draw conclusions and outline future research directions in Section 6.6.

6.2 Problem statement

On a given week, drivers have to perform duties according to their *weekly work schedule*, i.e., a specific assignment of duties and days off to each day of the week. Given a one-week horizon $\mathbf{J}$, the set of all (regular) duties that must be assigned on day $j \in \mathbf{J}$ is denoted by $\mathbf{D}_j^R$. Each duty is characterized by a starting time S_d and an ending time E_d, a set of bus lines L_d served in this duty, and its associated workload W_d. Contrary to the duty length (i.e., E_d - S_d), the workload W_d corresponds to the time actually paid to the worker for the duty d. For example, the value of W_d can be smaller than $E_d - S_d$ if the duty d is composed of two parts separated by a long break.

Work schedules are assigned to the drivers in the following way. Drivers are grouped together based on their work preferences (e.g., they prefer the same pattern of days off or a specific number of working days per week) or their personal capacities to perform certain types of duties (e.g., longer or shorter duties). Each group of drivers will then be assigned to a *roster*, i.e., a set of weekly work schedules according to which the drivers of the group will work. These weekly work schedules should all have more or less the same characteristics (e.g., the same number of days off) based on the drivers' preferences of the group.

Each work schedule is identified by its position $p \in \mathbf{P}_r$ in the roster $r \in \mathbf{R}$, where $\mathbf{R}$ denotes the set of rosters to build (one per driver group), and $\mathbf{P}_r$ the set of all the positions in r. A roster has as many different positions as the number of drivers in the group. The drivers will go cyclically over all of them. In other words, the driver assigned to a certain position in a given week will be assigned to the next position in the following week, and the driver assigned to the last position of the roster will be assigned to the first position in the following week. At the end of the cycle, each driver will have had the same work schedules, but in different weeks. Hence, in the long run, the average workload should be the same among all the drivers of a same roster.

Rosters must conform to different assignment rules, and due to the cyclic nature of the schedules, in addition to applying to each position individually, these rules must also be respected when moving from one position to the next. For instance, two consecutive duties should be interleaved with a night rest period and this rule should also apply to the duties assigned to Sunday and Monday of two consecutive positions (assuming that Monday is the first day of a week).

The DAPDP is a variant of the driver rostering problem. Days off scheduling is performed before duty assignment, and the fixed days off solution obtained is given as an input to the DAPDP. Rosters are defined in a way such that some assignments of duties to rosters are possible, while others are forbidden. The set of rosters to which a duty d can be assigned is denoted by $\mathbf{R}_d$.

Sometimes, the number of drivers can be different from the number of duties to be performed on a given day. In case of an excess of drivers, it is common to compensate for the missing regular duties by adding extra duties, that are called *standby duties*. They have the same characteristics as regular duties, except that no workload is associated with them. Indeed, a driver that is assigned to a standby duty on some day will have to work on that day only in the case of absence of another driver. Moreover, almost all the assignment rules apply to standby duties. The set of standby duties that can be assigned (once or more) on day j is denoted by $\mathbf{D}_j^S$.

Rosters must comply with safety and collective agreement rules and are thus hard constraints. To increase driver satisfaction, preference rules can also be considered as soft constraints. The roster feasibility rules stated below are the most common and important ones according to our industrial partner (although some of the constants may vary from one bus company to another) and were also used by Er-Rbib et al. (2020):

1. There must be at least 9 hours of resting time between two consecutive duties, and

10 hours of resting time if they are followed by a third duty;

2. There must be at least 57 hours of resting time including at least two consecutive days off per week;
3. There must be an average resting time of at least 12 hours in any period of 28 days ending with a working day;
4. There must be at most one duty lasting 13 hours or more assigned to each position;
5. There must be at most two consecutive duties serving the same bus line in each position;
6. Depending on the roster type, each position must have exactly 2 or 3 days off per position, and two of them must be consecutive. Days off falling on Sunday and Monday of two consecutive positions can be considered as two consecutive days off for either of the two positions but not for both.
7. There is no more than 6 consecutive days without a day off.

Throughout the rest of the paper, these rules will be referred to as Rules 1 to 7. Note that Rules 6 and 7 are important for days off scheduling but are not directly used for duty assignment. Note also that Rule 6 implies that Rule 2 can only be satisfied once per position, namely, around its double or triple day off.

The preference rules, which are related to the starting time of two consecutive duties, can be formulated as follows. On a given position:

1. A duty assigned to some day should not start earlier than the duty assigned to the previous day (i.e., increasing starting time);
2. A duty does not begin more than $\overline{H}$ minutes later than the duty of the previous day;
3. A duty does not begin more than $\underline{H}$ minutes earlier than the duty of the previous day.

Typically, $\overline{H}$ is slightly larger than 1440 (i.e., one day) and $\underline{H}$ is slightly less than 1440. While the first preference (Rule 8) is equivalent to the concept of favoring *forward rotations* when working with shifts, preferences of the second and third types (Rules 9 and 10), when used together, should increase *steadiness* in the resulting work schedules. Since these two goals are not necessarily compatible, it will generally be the case that either the first preference, or both the second and third are used together.

The DAPDP can be stated as follows. Given a set of regular and standby duties $\mathbf{D}_j^R \cup \mathbf{D}_j^S$ for each day $j \in \mathbf{J}$, a set of rosters $\mathbf{R}$ (one per group of drivers) and, for each roster $r \in \mathbf{R}$, a set of positions $\mathbf{P}_r$ and a days off schedule, the DAPDP consists in assigning the duties to positions such that each regular duty $d \in \mathbf{D}_j^R$, $j \in \mathbf{J}$, is assigned to exactly one position on

day j in a roster of $\mathbf{R}_d$. Standby duties are assigned only if needed. All the hard constraints must be respected, and preferences are soft constraints that should be satisfied whenever possible. Moreover, workload must be balanced as much as possible among all the positions.

6.3 Mathematical formulation

In this section we formulate the problem described in Section 6.2 as an integer program. Before presenting the corresponding mathematical model, we introduce first the required notation.

6.3.1 Notation

The different sets, parameters and variables that will be used throughout the rest of the paper (including those already defined above) are listed below. The following convention will be used: sets are denoted using bold capital letters, parameters using capital letters, and variables using lowercase letters. Note that most of the notation that we use is the same as the one introduced by Er-Rbib et al. (2020). Therefore, for the sake of simplicity, the definitions below are reproduced (almost) word for word from their work when possible.

Moreover, like in Er-Rbib et al. (2020), we will make use of the following vectorial function:

$$f(p, j, k) = \left(\left\lfloor p + \frac{(j+k)}{7} \right\rfloor \bmod \left| P_{r(p)} \right| \ , \ (j+k) \bmod 7 \right) \tag{6.1}$$

where $r(p)$ is the roster that contains position p, the positions in $\mathbf{P}_{r(p)}$ are number from 0 to $\left|\mathbf{P}_{r(p)}\right| - 1$, and $a \mod n$ is equal to $a - n \left\lfloor \frac{a}{n} \right\rfloor$. This function returns the position-day pair occurring k days after (if $k > 0$) or before (if $k < 0$) position-day pair (p, j) in roster $r(p)$. For example, if a roster contains the positions 0 to 3, then $f(3, 5, 3) = (0, 1)$ (i.e., in this roster, 3 days after day 5 in position 3 corresponds to day 1 in position 0) and $f(1, 3, -5) = (0, 5)$.

Sets

- $\mathbf{J}$: Set of days in a week, numbered from 0 (Monday) to 6 (Sunday);
- $\mathbf{D}_j^R$: Set of regular duties that must be assigned on day j;
- $\mathbf{D}_j^S$: Set of standby duties that can be assigned (once or more) on day j;
- $\mathbf{R}$: Set of all rosters;
- $\mathbf{R}_d$: Set of rosters to which duty d can be assigned;
- $\mathbf{P}$: Set of all positions in all rosters;

$\mathbf{P}_r$: Set of positions in roster r, numbered from 0 to $|\mathbf{P}_r| - 1$;

$\mathbf{D}_{pj}^R$: Set of regular duties that can be assigned to position p on day j;

$\mathbf{D}_{pj}$: Set of regular and standby duties $\mathbf{D}_{pj}^R \cup \mathbf{D}_j^S$ that can be assigned to position p on day j;

$\mathbf{L}$: Set of bus lines;

$\mathbf{L}_d$: Set of bus lines served in duty d;

$\mathbf{J}_p^W$: Set of working days in position p;

$\mathbf{J}_p^C$: Set of working days in position p which are immediately followed by a working day (Rules 8 to 10 must be checked);

$\mathbf{J}_p^9$: Set of working days in position p which are immediately followed by a working day and a day off (Rule 1 must be checked);

$\mathbf{J}_p^{10}$: Set of working days in position p which are immediately followed by at least two working days (Rules 1 and 5 must be checked);

$\mathbf{J}_p^{57}$: Set containing the single working day in position p which is followed by a double day off (Rules 2 and 6 must be checked);

$\mathbf{J}_p^{12}$: Set of days in position p which start a sequence of 28 consecutive days ending with a working day (Rule 3 must be checked);

$\mathbf{K}_{pj}$: Subset of $\{0, 1, \ldots, 27\}$ indicating the pairs of consecutive working days in a sequence of 28 consecutive days starting in position-day pair (p, j), i.e., $k \in K_{pj}$ if working days are assigned to both position-day pairs $f(p, j, k)$ and $f(p, j, k + 1)$;

$\mathbf{T}_{pj}^S$: Set of all starting times of the duties in D_{pj}.

Parameters

W_d: Workload in duty d (in minutes);

S_d: Starting time of duty d (in minutes from the beginning of the week);

E_d: Ending time of duty d (in minutes from the beginning of the week);

I_d^{13}: Binary indicator equal to 1 if the workload of duty d exceeds 13 hours;

I_{dl}: Binary indicator equal to 1 if duty d serves bus line l, i.e., if $l \in L_d$;

A: Average workload per position ($A = \frac{\sum_{j \in \mathbf{J}} \sum_{d \in \mathbf{D}_j^R} W_d}{\sum_{r \in \mathbf{R}} |\mathbf{P}_r|}$);

F_{dt}^+: Binary indicator equal to 1 if duty d finishes at time t or after;

B_{dt}^+: Binary indicator equal to 1 if duty d begins at time t or after;

B_{dt}^-: Binary indicator equal to 1 if duty d begins before time t (excluding time t);

U_{pj}: Total number of working days immediately followed by a working day in a sequence of 28 consecutive days starting in position-day pair (p, j);

$\overline{H}, \underline{H}$: Parameters defining Rules 9 and 10.

For the sake of notational conciseness, we assume that S_d is increased by 10080 (i.e., one week) whenever the context dictates it, for example, if d is operated on day 0 and a time on day 6 must be subtracted from S_d. Also, for parameters F^+_{dt}, B^+_{dt} and B^-_{dt}, any value of $t > 10080$ should be replaced by t mod 10080.

Variables

x^d_{pj}: Binary variable equal to 1 if duty d is assigned to position p on day j, and to 0 otherwise;

z_p: Nonnegative variable equal to the excess workload in position p with respect to the average workload per position, i.e., $z_p = \max\{0, \sum_{j\in\mathbf{J}^W_p} \sum_{d\in\mathbf{D}^R_{pj}} W_d x^d_{pj} - A\}$;

e_{pj}: Binary variable equal to 1 if the preference of position p and day j is not respected, and to 0 otherwise.

6.3.2 DAPDP model

Before modeling the DAPDP, we present a model for the duty assignment problem (DAP) which is a special case of the DAPDP that is obtained by omitting all preference-related soft constraints. The DAP has been formulated by Er-Rbib et al. (2020) as the following integer program:

$$\min \quad \sum_{p\in\mathbf{P}} z_p \tag{6.2}$$

$$\text{subject to} \quad \sum_{j\in\mathbf{J}^W_p} \sum_{d\in\mathbf{D}^R_{pj}} W_d x^d_{pj} - A \le z_p \qquad \forall p \in \mathbf{P} \tag{6.3}$$

$$z_p \ge 0 \qquad \forall p \in \mathbf{P} \tag{6.4}$$

$$\sum_{r\in\mathbf{R}_d} \sum_{p\in\mathbf{P}_r} x^d_{pj} = 1 \qquad \forall j \in \mathbf{J}, d \in \mathbf{D}^R_j \tag{6.5}$$

$$\sum_{d\in\mathbf{D}_{pj}} x^d_{pj} = 1 \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}^W_p \tag{6.6}$$

$$\sum_{d\in\mathbf{D}_{pj}} F^+_{dt} x^d_{pj} + \sum_{d\in\mathbf{D}_{f(p,j,1)}} B^-_{d,t+540} x^d_{f(p,j,1)} \le 1 \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}^9_p, t \in \mathbf{T}_{pj} \tag{6.7}$$

$$\sum_{d\in\mathbf{D}_{pj}} F^+_{dt} x^d_{pj} + \sum_{d\in\mathbf{D}_{f(p,j,1)}} B^-_{d,t+600} x^d_{f(p,j,1)} \le 1 \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}^{10}_p, t \in \mathbf{T}_{pj} \tag{6.8}$$

$$\sum_{d\in\mathbf{D}_{pj}} F^+_{dt} x^d_{pj} + \sum_{d\in\mathbf{D}_{f(p,j,3)}} B^-_{d,t+3420} x^d_{f(p,j,3)} \le 1 \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}^{57}_p, t \in \mathbf{T}_{pj} \tag{6.9}$$

$$\sum_{k\in\mathbf{K}_{pj}}\left(\sum_{d\in\mathbf{D}_{f(p,j,k+1)}} S_d x^d_{f(p,j,k+1)} - \sum_{d\in\mathbf{D}_{f(p,j,k)}} E_d x^d_{f(p,j,k)}\right) \geq 720 U_{pj} \qquad \forall p\in\mathbf{P}, j\in\mathbf{J}^{12}_p \tag{6.10}$$

$$\sum_{j\in\mathbf{J}^W_p}\sum_{d\in\mathbf{D}_{pj}} I^{13}_d x^d_{pj} \leq 1 \qquad \forall p\in\mathbf{P} \tag{6.11}$$

$$\sum_{d\in\mathbf{D}_{pj}} I_{dl} x^d_{pj} + \sum_{d\in\mathbf{D}_{f(p,j,1)}} I_{dl} x^d_{f(p,j,1)} + \sum_{d\in\mathbf{D}_{f(p,j,2)}} I_{dl} x^d_{f(p,j,2)} \leq 2 \qquad \forall p\in\mathbf{P}, j\in\mathbf{J}^{10}_p, l\in\mathbf{L} \tag{6.12}$$

$$x^d_{pj}\in\{0,1\} \qquad \forall p\in\mathbf{P}, j\in\mathbf{J}^W_p, d\in\mathbf{D}_{pj} \tag{6.13}$$

This model aims at balancing as much as possible the workload among the positions. Combined with constraints (6.3) and (6.4), the objective function (6.2) minimizes the sum over all positions of the workload exceeding the average workload per position. Constraints (6.5) specify that each regular duty has to be assigned to a position. Constraints (6.6) impose that a duty is assigned on each working day. As stipulated by Rule 1, constraints (6.7) and (6.8) enforce at least 9 or 10 hours of rest time between two consecutive duties by forbidding the assignment of a duty to position-day pair (p, j) and another to position-day pair $f(p, j, 1)$ that would violate the required minimum rest time. Constraints (6.9) – (6.12) ensure that Rules 2 to 5 are satisfied.

The rest of this section is dedicated to introducing a variant of this model that addresses the DAPDP. In addition to the collective agreement rules already considered in the DAP model, the DAPDP model should be able to include the three preferences stated at the end of Section 6.2. The first preference (Rule 8, favoring increasing starting times) can be expressed as the linear soft constraints

$$\sum_{d\in\mathbf{D}_{pj}} B^+_{dt} x^d_{pj} + \sum_{d\in\mathbf{D}_{f(p,j,1)}} B^-_{d,t+1440} x^d_{f(p,j,1)} \leq 1 + e^1_{pj} \quad \forall p\in\mathbf{P}, j\in\mathbf{J}^C_p, t\in\mathbf{T}^S_{pj},$$

where $e^1_{pj}\in\{0,1\}$ is a penalty variable (to be included in the objective function) that takes value 1 if the preference is not respected on day j of position p, and 0 otherwise. Note that this preference could also have been modeled as follows:

$$\sum_{d\in\mathbf{D}^R_{pj}} S_d x^d_{pj} - \sum_{d\in\mathbf{D}^R_{f(p,j,1)}} S_d x^d_{f(p,j,1)} \leq \epsilon_{pj} \qquad \forall p\in\mathbf{P}, j\in\mathbf{J}^C_p,$$

where ϵ_{pj} is a nonnegative variable equal to the difference between the starting times of the chosen duties for position-day pairs (p, j) and $f(p, j, 1)$. However, as explained in Er-Rbib et al. (2020), the first formulation yields tighter linear relaxations, which is expected to reduce solution time.

Similarly, the second and the third preferences (Rules 9 and 10) can be expressed by introducing the following two sets of soft constraints and their respective binary penalty variables e^2_{pj} and e^3_{pj}:

$$\sum_{d\in D_{pj}} B^-_{d,t+1} x^d_{pj} + \sum_{d\in D_{f(p,j,1)}} B^+_{d,t+\overline{H}+1} x^d_{f(p,j,1)} \leq 1 + e^2_{pj} \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}^C_p, t \in \mathbf{T}^S_{pj},$$

$$\sum_{d\in D_{pj}} B^+_{dt} x^d_{pj} + \sum_{d\in D_{f(p,j,1)}} B^-_{d,t+\underline{H}} x^d_{f(p,j,1)} \leq 1 + e^3_{pj} \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}^C_p, t \in \mathbf{T}^S_{pj}.$$

The employee preferences are satisfied as much as possible when the total sum of the penalty variables e^k_{pj} is minimized, which can be expressed as the following objective function:

$$\min \qquad \sum_{p\in\mathbf{P}} \sum_{j\in\mathbf{J}^C_p} (e^1_{pj} + e^2_{pj} + e^3_{pj}).$$

However, as mentioned at the end of Section 6.2, it will often be desirable in practice to include in the model only a subset of the three preferences described above: either the first one, or both the second and third ones. To this end, we introduce the binary parameters Q_i, $i = 1, \ldots, 3$, taking the value 1 if and only if the preference type associated with the penalty variables e^i_{pj} should be taken into account by the model. With these parameters, the objective function becomes

$$\min \qquad \sum_{p\in\mathbf{P}} \sum_{j\in\mathbf{J}^C_p} (Q_1 e^1_{pj} + Q_2 e^2_{pj} + Q_3 e^3_{pj}). \tag{6.14}$$

On the other hand, since employee preferences generally compete with the original objective of balancing as much as possible the workload among the positions, the original objective function (6.2) should not be replaced simply with (6.14), which could lead to very unbalanced solutions. In order to deal with this issue, an approach consists in adding the sum of the considered e^k_{pj} variables (or a constant multiple C thereof) to the objective function (6.2):

$$\min \qquad \sum_{p\in\mathbf{P}} \left(z_p + C \left(\sum_{j\in\mathbf{J}^C_p} Q_1 e^1_{pj} + Q_2 e^2_{pj} + Q_3 e^3_{pj} \right) \right).$$

The constant C is chosen to be more or less large depending on the relative importance of preferences in comparison to workload balance.

Another approach consists in using (6.14) in conjunction with a constraint that limits the deterioration of the optimal value Z^*_{DAP} of the DAP model (6.2) – (6.13) up to a certain

percentage B:

$$\sum_{p \in \mathbf{P}} z_p \leq (1 + B)\ Z^*_{DAP}. \tag{6.15}$$

This approach is the one used in this paper.

By adding preference constraints and changing the objective function, the DAP model can be transformed into a single-objective integer program that addresses the DAPDP:

$$\min \sum_{p \in \mathbf{P}} \sum_{j \in \mathbf{J}^C_p} Q_1 e^1_{pj} + Q_2 e^2_{pj} + Q_3 e^3_{pj} \tag{6.14}$$

$$\text{subject to} \quad \sum_{p \in \mathbf{P}} z_p \leq (1 + B) Z^*_{DAP} \tag{6.15}$$

$$\sum_{j \in \mathbf{J}^W_p} \sum_{d \in \mathbf{D}^R_{pj}} W_d x^d_{pj} - A \leq z_p \qquad \forall p \in \mathbf{P} \tag{6.16}$$

$$z_p \geq 0 \qquad \forall p \in \mathbf{P} \tag{6.17}$$

$$\sum_{r \in \mathbf{R}_d} \sum_{p \in \mathbf{P}_r} x^d_{pj} = 1 \qquad \forall j \in \mathbf{J}, d \in \mathbf{D}^R_j \tag{6.18}$$

$$\vdots \qquad \vdots$$

$$x^d_{pj} \in \{0, 1\}, \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}^W_p, d \in \mathbf{D}_{pj}, \tag{6.26}$$

$$\sum_{d \in \mathbf{D}_{pj}} B^+_{dt} x^d_{pj} + \sum_{d \in \mathbf{D}_{f(p,j,1)}} B^-_{d,t+1440} x^d_{f(p,j,1)} \leq 1 + e^1_{pj} \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}^C_p, t \in \mathbf{T}^S_{pj} \tag{6.27}$$

$$\sum_{d \in \mathbf{D}_{pj}} B^-_{d,t+1} x^d_{pj} + \sum_{d \in \mathbf{D}_{f(p,j,1)}} B^+_{d,t+\overline{H}+1} x^d_{f(p,j,1)} \leq 1 + e^2_{pj} \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}^C_p, t \in \mathbf{T}^S_{pj}, \tag{6.28}$$

$$\sum_{d \in \mathbf{D}_{pj}} B^+_{dt} x^d_{pj} + \sum_{d \in \mathbf{D}_{f(p,j,1)}} B^-_{d,t+\underline{H}} x^d_{f(p,j,1)} \leq 1 + e^3_{pj} \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}^C_p, t \in \mathbf{T}^S_{pj} \tag{6.29}$$

$$e^k_{pj} \in \{0, 1\}, \qquad \forall p \in \mathbf{P}, j \in \mathbf{J}^C_p, d \in \mathbf{D}_{pj}, k \in \{1, 2, 3\}\,. \tag{6.30}$$

Combined with the preference constraints (6.27) – (6.30), the new objective function (6.14) minimizes the number of times that preferences are not respected, subject to the budget constraint (6.15). Note that constraints (6.16) – (6.26) are exactly the same as constraints (6.3) – (6.13) of the DAP model. From that point on, model (6.14) – (6.30) will be referred to as the *DAPDP model*.

6.4 Solution algorithms

The integer program (6.14) – (6.30) can be implemented directly in a commercial MILP solver. However, due to its complexity, it is practically impossible to solve it for large instances. In this section, we describe two matheuristics that make it possible to obtain good solutions even if the instances under consideration are large.

6.4.1 Preassigning duties to rosters

In general, on a day $j \in \mathbf{J}$, a duty $d \in \mathbf{D}_j^R$ can be assigned to a position $p \in \mathbf{P}_r$ associated with any roster $r \in \mathbf{R}_d$. In the case of instances where the size of $\mathbf{R}_d$ is relatively large, this flexibility in the duty/roster assignment may make the resulting model too complex to be solved in reasonable time. To make the problem tractable, a simple solution would be to restrict the assignment of a duty to a single roster. In other words, each duty is preassigned to a specific roster before the DAPDP model is solved. Since this preassignment can have a meaningful impact on the quality of the solution, care should be taken when choosing this roster. Considering that balancing the workload among the positions remains an important goal of the DAPDP (see constraints (6.15)), a good candidate for this roster would be the roster to which the duty was assigned in an optimal solution to the original DAP.

Based on the above observation, we propose the following matheuristic (named PDR for Preassigning Duties to Rosters):

1. Solve the DAP model (6.2) – (6.13). Let $\overline{x}_{pj}^d$ be the duty assignment variable values of an optimal solution.
2. For each $j \in \mathbf{J}$ and $d \in \mathbf{D}_j^R$, let r_{dj} be the roster to which the duty d is assigned in this solution. More precisely, r_{dj} is the only element of $\mathbf{R}_d$ such that $\overline{x}_{pj}^d = 1$ and $p \in \mathbf{P}_{r_{dj}}$.
3. Solve the model (6.14) – (6.30) where (6.18) is replaced with

$$\sum_{p \in \mathbf{P}_{r_{dj}}} x_{pj}^d = 1 \qquad \forall j \in \mathbf{J}, d \in \mathbf{D}_j^R,$$

   and $x_{pj}^d = 0$ for all $p \in \mathbf{P}$, $j \in \mathbf{J}_p^W$, $d \in \mathbf{D}_{pj}$ such that $p \notin \mathbf{P}_{r_{dj}}$.

Note that in the last step, the optimal solution of the DAP model (obtained in the first step) can be used as an initial solution to perform a warm-start.

6.4.2 Partitioning the positions of a roster

The PDR matheuristic presented in Section 6.4.1 simplifies considerably the integer program (6.14) – (6.30) in the case of instances in which each duty can be assigned to a relatively large number of rosters. On the other hand, as will be seen in Section 6.5, for large instances in which the number of rosters to which duties can be assigned is small, this duty/roster preassignment might not be sufficient to make the problem tractable. Indeed, in such a situation, the complexity of the model mainly comes from the large number of positions in each roster, and not from the number of rosters available for each duty.

To deal with this issue, we propose a matheuristic that subdivides each roster by partitioning the positions it contains, creating new rosters (called subrosters) for drivers with the same characteristics/preferences. The final number of subrosters should be large enough so that applying the PDR algorithm would substantially cut down the solution time, but the number of positions per subroster should not be too small, in which case all the high-quality solutions could be excluded from the search space.

Moreover, since Rules 6 and 7 related to the assignment of days off have to be respected by the subrosters as well, not all partitions give rise, in general, to a set of valid subrosters. To guarantee that positions i to j form a valid subroster, we will require that the days off of position $j+1$ are assigned to the same days of the week as the days off of position i. Note that this condition is stronger than the requirement that $\{i, \ldots, j\}$ is a valid subroster.

Before presenting the integer program that produces a partition having the characteristics mentioned above, we introduce new notation. Let $n_{\min}$ be a positive integer corresponding to the minimal allowed number of positions that a subroster can contain. Define the set of admissible position pairs of roster $r \in \mathbf{R}$ as

$$\mathbf{A}_r := \{(i,j) \in \mathbf{P}_r \times \mathbf{P}_r : i \sim j+1 \text{ and } j-i+1 \geq n_{\min}\},$$

where $i \sim j+1$ if and only if the positions i and $j+1$ share the same days off. An admissible position pair $(i,j) \in \mathbf{A}_r$ defines a potential subroster of roster r that is composed of the positions i to j. Note that set $\mathbf{A}_r$ does not allow the definition of a subroster from a position i to a position j such that $i > j$, i.e., which includes both the first and the last position of roster r. This possibility might be helpful to define smaller subrosters but was not required for our tests.

For all $(i,j) \in \mathbf{A}_r$ and $p \in \mathbf{P}_r$, let

$$a_{ij}^p = \begin{cases} 1 & \text{if } i \leq p \leq j, \\ 0 & \text{otherwise} \end{cases}$$

be a parameter indicating whether position p is included in a potential subroster between positions i and j. Furthermore, let y_{ij}^r be a binary variable that takes value 1 if positions i to j of roster r define one of the subrosters of r.

With the above notation, the problem of finding for roster $r \in \mathbf{R}$ a partition of positions (i.e., a subset of disjoint subrosters) meeting the above requirements can be modeled as the following integer program:

$$\min \; v \tag{6.31}$$

$$\text{subject to} \quad v \geq (j-i+1) y_{ij}^r \qquad \forall (i,j) \in \mathbf{A}_r \tag{6.32}$$

$$\sum_{(i,j) \in \mathbf{A}_r} a_{ij}^p y_{ij}^r = 1 \qquad \forall p \in \mathbf{P}_r \tag{6.33}$$

$$y_{ij}^r \in \{0,1\} \qquad \forall (i,j) \in \mathbf{A}_r \tag{6.34}$$

Combined with (6.32), the objective function (6.31) minimizes the maximum number of positions in a selected subroster expressed by variable v. Constraints (6.33) make sure that the selected subrosters are disjoint and cover exactly once each position of roster r.

Based on this partitioning of the rosters, we propose the following matheuristic (named PRP for Partitioning Roster Positions):

1. For each roster $r \in \mathbf{R}$, solve the integer program (6.31) – (6.34) to create a set of subrosters

   $$\mathbf{R}'_r := \left\{ r'_1, r'_2, \ldots, r'_{n_r} \right\},$$

   where $n_r := \sum_{(i,j) \in \mathbf{A}_r} y_{ij}^r$ is the number of created subrosters.

2. Define

   $$\mathbf{R}' := \cup_{r \in \mathbf{R}} \mathbf{R}'_r \quad \text{and}$$
   $$\mathbf{R}'_d := \{ r' \in \mathbf{R}' : \exists\, r \in \mathbf{R}_d \text{ such that } r' \in \mathbf{R}'_r \}, \qquad \forall j \in \mathbf{J}, d \in \mathbf{D}_j^R.$$

3. Apply the duty/roster preassignment algorithm described in Section 6.4.1 where $\mathbf{R}$

and $\mathbf{R}_d$, $j \in \mathbf{J}$, $d \in \mathbf{D}_j^R$, are replaced with $\mathbf{R}'$ and $\mathbf{R}'_d$, $j \in \mathbf{J}$, $d \in \mathbf{D}_j^R$, respectively.

Note that the PRP algorithm replaces the original rosters by a set of subrosters. In the resulting schedule, a driver is then assigned cyclically to all the positions of a subroster, but not to all the positions of the original roster. In this sense, the duty assignment produced by this heuristic might not yield a feasible solution to the original DAPDP (i.e., where the drivers would cycle through the positions of the original rosters) because Rules 1 and 3 might be violated when they involve duties assigned to the last position of a subroster and the first of another subroster. However, if the drivers cycle through the positions in the subrosters (instead of the positions in the rosters), the solution is feasible but may yield a lower balance of the workload between the drivers of a roster split into several subrosters. For example, a roster with 10 positions with an average workload of 40 hours per week may be split into two subrosters of 5 positions each, with average workloads of 39 and 41 hours, respectively. Given that the drivers of the first subroster will not cycle through the positions of the second subroster, the workloads of the drivers in these two subrosters will remain unequal even in the long run. Nevertheless, as our computational results will show, the difference in the workload balancing is not significant.

6.5 Computational experiments

The main goal of this section is to present the results of computational experiments that were carried out to compare the performance of the models and algorithms introduced in Sections 6.3 and 6.4.

All the proposed models were implemented in the C++ language and the IBM Concert Technology, and solved using the commercial MILP solver CPLEX version 12.7.0.0 with default parameter values. All the computational experiments were conducted on a Linux personal computer equipped with an Intel Core i7-8700K processor, clocked at 3.70 GHz and a RAM of 32 GB. For all our tests, a time limit of twelve hours (43 200 seconds) was imposed.

6.5.1 Instances

The computational experiments were conducted on the same seven instances as the one used in Er-Rbib et al. (2020). They are derived from two real-world datasets, A and G, coming from two different European public transit companies. Their characteristics are reported in Table 6.3. A precise definition of the information reported in the last three columns, i.e., the "Workload Cost", the "# Violations" and "% of Violated Preferences", will be given in Section 6.5.2 below. The indices in the name of an instance indicate the original dataset, the

number of rosters and the number of positions, respectively. The full datasets correspond to instances $I_{A,14,202}$ and $I_{G,2,333}$. The additional five datasets were generated randomly from A and G. For more information on how these smaller-sized instances were derived, the reader is referred to Section 5.1 in Er-Rbib et al. (2020).

Table 6.3 Instances and their characteristics

Instance	Dataset	# Rosters	# Positions	# Duties	Workload Cost	# Violations	% of Violated Preferences
$I_{A,5,110}$	A	5	110	490	4966	183	53
$I_{A,6,124}$	A	6	124	414	3461	155	48
$I_{A,10,146}$	A	10	146	640	6291	232	52
$I_{A,14,202}$	A	14	202	881	18532	331	50
$I_{G,2,165}$	G	2	165	745	6222	291	54
$I_{G,2,180}$	G	2	180	813	7853	303	50
$I_{G,2,333}$	G	2	333	1509	13531	597	54

6.5.2 Results and discussion

In all the computational experiments discussed below, we assumed that the work schedules should favor increasing starting time (Rule 8), which corresponds to constraints (6.27). In other words, Rules 9 and 10, modeled through constraints (6.28) and (6.29), were left out (or, equivalently, $Q_1 = 1$ and $Q_2 = Q_3 = 0$) of our tests because preliminary results showed that similar conclusions to those presented next for Rule 8 would be drawn.

As explained in Section 6.3.2, a solution of the DAPDP should minimize the number of preference violations (i.e., the value of the objective function of a solution to (6.14) – (6.30)) without compromising on the workload balance among the drivers. For this reason, in each experiment, solutions will be evaluated both in terms of the number of violations, i.e.,

$$\pi(x) := \sum_{p \in \mathbf{P}} \sum_{j \in \mathbf{J}_p^C} e_{pj}^1,$$

and in terms of the sum over all positions of the workload exceeding the average workload per position, i.e.,

$$\omega(x) := \sum_{p \in \mathbf{P}} \max \left\{ 0, \sum_{j \in \mathbf{J}_p^W} \sum_{d \in \mathbf{D}_{pj}^R} W_d x_{pj}^d - A \right\} \quad \left(= \sum_{p \in \mathbf{P}} z_p \right),$$

where $x = (x^d_{pj})$ represents a solution of the DAP or the DAPDP. Moreover, our solutions of the DAPDP will be compared with the optimal solution of the DAP, denoted by $\tilde{x}$, found in Er-Rbib et al. (2020). For each instance, the values of $\omega(\tilde{x})$ and $\pi(\tilde{x})$ are reported in Table 6.3 as the "Workload Cost" and the "# Violations", respectively. In the last column, we included information about the proportion of preferences that are not satisfied. More precisely, the "% of Violated Preferences" is equal to

$$100\ \frac{\pi(\tilde{x})}{\sum_{p\in\mathbf{P}} \left|\mathbf{J}^C_p\right|},$$

where the denominator corresponds to the maximal number of preferences that can be violated. As can be seen from this column, for each instance, close to half of the preferences are not satisfied in the optimal solution to the DAP. This large proportion of preference violations justifies the need of considering driver preferences directly in the duty assignment step.

The results of each experiment that were conducted are presented in the form of a table that reports, for each instance, information about the size of the model solved, the solution process, and the quality of the solution obtained. For the model size, the numbers of variables (# Var.) and constraints (# Const.) in the model are reported. Next, the table reports information about the solution process of the MILP involved. Note that no matter whether the solution to the DAPDP is obtained by solving model (6.14) – (6.30) directly or by using one of the heuristics presented in Section 6.4, a MILP consisting in a variation of model (6.14) – (6.30) has to be solved at some point in order to produce the solution. The solution process of this MILP is described in terms of the number of branch-and-bound nodes explored in the search tree (excluding the root node), the total number of cuts applied, the solution time, and the time required to obtain the best integer solution found (BIT). Finally, the quality of the solution to the DAPDP, denoted by x, with respect to the quality of the optimal solution to the DAP, $\tilde{x}$, is expressed in terms of the fraction of violations remaining (i.e., $\pi(x)/\pi(\tilde{x})$) and the workload increase (i.e., $\omega(x)/\omega(\tilde{x})$).

DAPDP model

To determine by how much the number of preference violations can be reduced, the DAPDP model (i.e., (6.14) – (6.30)) was first solved without the budget constraint (6.15) (or, equivalently, with a budget $B = \infty$). The results are presented in Table 6.4. An optimal solution that contains almost no preference violations was found for all the A-instances. On the other hand, for each of them, the workload cost has more than doubled in comparison to the op-

timal one. For the largest of these instances, $I_{A,14,202}$, the workload has even increased by a factor of around 5. As for the instances derived from the dataset G, none of them could be solved to optimality within the time limit. However, for the smallest of these instances, $I_{G,2,165}$, it was possible to find a non-optimal solution that contains almost no preference violations (only about 5.9 % left) and that increases the workload cost by a factor of about 2.5. For the slightly larger instance $I_{G,2,180}$, a non-optimal solution that reduces the number of preference violations by about 95% was found, whereas no solution at all could be obtained for the largest instances $I_{G,2,333}$.

Table 6.4 Results for the DAPDP model without budget.

Instance	# Var.	# Const.	# Nodes	# Cuts	Time (s)	BIT (s)	$\frac{\pi(x)}{\pi(\tilde{x})}$	$\frac{\omega(x)}{\omega(\tilde{x})}$
$I_{A,5,110}$	40564	112602	7	70	317	297	0.011	2.50
$I_{A,6,124}$	34994	102438	0	54	211	209	0.006	3.23
$I_{A,10,146}$	67758	161550	0	39	1499	1499	0.000	2.77
$I_{A,14,202}$	129593	287632	0	26	17748	17748	0.000	4.99
$I_{G,2,165}$	90488	266867	1605	154	43200	34802	0.059	2.48
$I_{G,2,180}$	107671	292372	1749	110	43200	39836	0.052	1.21
$I_{G,2,333}$	356186	653439	0	44	43200			

By solving the DAPDP model without budget, it was possible to obtain good solutions as far as the number of preferences is concerned, but not in terms of workload. To deal with this issue, a budget of 20%, which seems reasonable from the point of view of applications, was added to the model (i.e., constraints (6.15) with $B = 0.2$). The results appear in Table 6.5. Within the time limit, it was possible to find a feasible solution for all the instances except for the largest one. On the other hand, the solution was proven to be optimal only for $I_{A,5,110}$ and $I_{A,6,124}$. These optimal solutions offer exactly the same reductions of preference violations as the ones of the solutions to the DAPDP model without budget, but with a workload increase limited to 20%. However, it should be pointed out that the computational time required to obtain these solutions is larger by at least an order of magnitude than the one required to solve the DAPDP model without budget on the same instances. As for the remaining instances, a solution that reduces significantly the number of preference violations was found only for the slightly larger instance $I_{A,10,146}$.

The large values of $\frac{\omega(x)}{\omega(\tilde{x})}$ in Table 6.4 indicate that the solutions obtained by solving the DAPDP model without budget are substantially different from the solutions produced by the DAP model. Adding the budget constraint to the DAPDP model allows to limit the deterioration of the excess workload cost, but at the expense of much larger computational

Table 6.5 Results for the DAPDP model with a budget of 20%.

Instance	# Var.	# Const.	# Nodes	# Cuts	Time (s)	BIT (s)	$\frac{\pi(x)}{\pi(\tilde{x})}$	$\frac{\omega(x)}{\omega(\tilde{x})}$
$I_{A,5,110}$	40564	112603	45236	116	34499	33391	0.011	1.20
$I_{A,6,124}$	34994	102439	44505	134	39925	39778	0.006	1.20
$I_{A,10,146}$	67758	161551	4706	70	43200	41076	0.061	1.20
$I_{A,14,202}$	129593	287633	0	26	43200	41031	0.88	1.20
$I_{G,2,165}$	90488	266868	0	246	43200	5687	1.056	1.20
$I_{G,2,180}$	107671	292373	0	64	43200	39256	0.997	1.20
$I_{G,2,333}$	356186	653440	0	216	43200			

times.

In view of these observations, we conducted an experiment in which we looked at the impact of warm-starting the DAPDP model without budget with an optimal solution to the DAP. By providing an initial solution that has $\frac{\omega(x)}{\omega(\tilde{x})} = 1$, it seems reasonable to expect that the final solution will stay relatively "close" to the initial one, which should then imply a relatively low value of the excess workload cost. The results of this experiment are reported in Table 6.6. As expected, the value of $\frac{\omega(x)}{\omega(\tilde{x})}$ are smaller for all the instances than those reported in Table 6.4. However, this reduction is of only about 3.4 %. Solution times are also slightly larger in general.

Table 6.6 Results for the DAPDP model with a warm-start and without budget.

Instance	# Var.	# Const.	# Nodes	# Cuts	Time (s)	BIT (s)	$\frac{\pi(x)}{\pi(\tilde{x})}$	$\frac{\omega(x)}{\omega(\tilde{x})}$
$I_{A,5,110}$	40319	44461	208	86	4632	590	0.011	2.41
$I_{A,6,124}$	34791	40695	0	52	357	357	0.006	3.09
$I_{A,10,146}$	67268	62229	0	50	1243	1243	0.000	2.56
$I_{A,14,202}$	128662	112631	1	32	43200	26136	0.003	4.89
$I_{G,2,165}$	89809	102694	1668	251	43200	42766	0.035	2.48
$I_{G,2,180}$	106747	107416	193	60	43200	43200	0.123	1.17
$I_{G,2,333}$	352868	235891	0	49	43200	9	1.000	1.00

It appears from the results above that the DAPDP model without budget does not seem to be appropriate for solving the DAPDP due to the considerable deterioration of the excess workload cost, even with a warm-start. On the other hand, the DAPDP model with budget seems to be suitable to deal with small instances, but not with medium- or large-sized ones.

PDR algorithm results

Next, we performed computational experiments on our instances after preassigning duties to rosters according to the PDR matheuristic described in Section 6.4.1. The results for an infinite budget, a budget of 20% and a budget of 5% are reported in Tables 6.7, 6.8 and 6.9, respectively. From these results, we observe that for all budget scenarios, an optimal solution to the MILP involved in the PDR matheuristic was found for almost all the instances derived from the dataset A. The only exception is $I_{A,5,110}$ with an infinite budget, for which, however, the best integer solution was found after only 173 seconds. On the other hand, for each instance coming from the dataset G, the time limit was reached and the MILP could not be solved to optimality. This could be explained by the fact that the G-instances have only 2 rosters each, whereas each A-instance has between 5 and 14 rosters. Indeed, the effect of restricting every duty to a single roster should have a greater effect when there is a larger number of rosters to choose from. This phenomenon can be seen when looking at the reduction of the number of variables. Preassigning duties to rosters decreases the number of variables by an average of about 60% among all the instances. However, this reduction is considerably larger in the case of the instances derived from the dataset A ($\approx 75\%$) than for the instances coming from the dataset G ($\approx 37\%$). As for the number of constraints, it undergoes only a small decrease (by an average of about 4%), which is slightly larger for the A-instances ($\approx 6\%$) than for the G-instances ($\approx 1\%$).

Table 6.7 Results of the PDR matheuristic without budget.

Instance	# Var.	# Const.	# Nodes	# Cuts	Time (s)	BIT (s)	$\frac{\pi(x)}{\pi(\tilde{x})}$	$\frac{\omega(x)}{\omega(\tilde{x})}$
$I_{A,5,110}$	13095	109036	589710	2007	43200	173	0.337	1.03
$I_{A,6,124}$	11213	98032	3934	395	61	61	0.269	1.09
$I_{A,10,146}$	14482	151071	5857	910	119	46	0.342	1.02
$I_{A,14,202}$	19884	260890	4262	915	147	147	0.374	1.02
$I_{G,2,165}$	60732	264147	2449	370	43200	41958	0.528	1.45
$I_{G,2,180}$	72527	291001	3223	367	43200	41386	0.532	1.34
$I_{G,2,333}$	191602	640684	0	85	43200	28764	0.617	1.28

This significant reduction of the size of the models has for effect to shorten considerably the solution time for all the instances derived from A, except for $I_{A,5,110}$, for which it increases markedly. When no budget is allocated, the time limit has even been reached before optimality could be proved. Nonetheless, the best solution was found after only 173 seconds.

It should be noted that the solutions computed by the PDR matheuristic contain many more

Table 6.8 Results of the PDR matheuristic with a budget of 20%.

Instance	# Var.	# Const.	# Nodes	# Cuts	Time (s)	BIT (s)	$\frac{\pi(x)}{\pi(\tilde{x})}$	$\frac{\omega(x)}{\omega(\tilde{x})}$
$I_{A,5,110}$	13095	109037	193074	1868	6697	212	0.337	1.20
$I_{A,6,124}$	11213	98033	6168	480	123	116	0.269	1.20
$I_{A,10,146}$	14482	151072	4412	773	70	59	0.342	1.20
$I_{A,14,202}$	19884	260891	5033	1040	167	124	0.374	1.20
$I_{G,2,165}$	60732	264150	0	126	43200	20566	0.698	1.20
$I_{G,2,180}$	72527	291005	0	437	43200	1579	1.013	1.20
$I_{G,2,333}$	191602	640685	0	80	43200	34012	1.020	1.20

Table 6.9 Results of the PDR matheuristic with a budget of 5%.

Instance	# Var.	# Const.	# Nodes	# Cuts	Time (s)	BIT (s)	$\frac{\pi(x)}{\pi(\tilde{x})}$	$\frac{\omega(x)}{\omega(\tilde{x})}$
$I_{A,5,110}$	13095	109037	435357	1578	23201	215	0.337	1.05
$I_{A,6,124}$	11213	98033	4766	399	118	115	0.269	1.05
$I_{A,10,146}$	14482	151072	7723	731	237	53	0.342	1.05
$I_{A,14,202}$	19884	260891	5753	1218	194	98	0.374	1.05
$I_{G,2,165}$	60732	264150	0	222	43200	1012	1.007	1.05
$I_{G,2,180}$	72527	291005	0	261	43200	1601	1.061	1.05
$I_{G,2,333}$	191602	640685	0	51	43200			

preference violations than the solutions obtained by solving the original DAPDP model to optimality. As can be seen from Tables 6.4 and 6.5, the instances for which an optimal solution to the original DAPDP model could be found satisfy almost all the preference constraints. This is in stark contrast with the PDR solutions, for which about a third of the preference violations still remain. This can be attributed to the fact that the restricted model resulting from this heuristic cannot give rise to solutions that differ too greatly from the solution that was used to preassign the duties. This forced resemblance of the solutions also explains why workload cost of optimal solutions increases by only 4% on average when no budget is allocated. Nevertheless, removing close to two thirds of the preference violations is not to be neglected.

Finally, one can observe from Tables 6.7, 6.8 and 6.9 that the ratio $\frac{\pi(x)}{\pi(\tilde{x})}$ of each A-instance is the same, no matter the value of the budget. Note that by adding a finite budget (constraint (6.15)) to the model, the value of $\pi(x)$ (the objective function) can only increase or stay the same. The fact that $\pi(x)$ did not increase for any of the A-instance indicates that solutions that are as good (in terms of the number of preference violations) could be found after

restricting the deterioration of workload balance to only 5%. It is expected that the ratio $\frac{\pi(x)}{\pi(\tilde{x})}$ would increase (up to a maximum that is bounded above by 1) by reducing the budget even more.

PRP algorithm results

As can be seen from Tables 6.7, 6.8 and 6.9, preassigning duties to rosters did not allow any of the instances derived from the dataset G to be solved to optimality. As discussed above, the inefficiency of the PDR matheuristic to solve G-instances seems to be related to the fact that each instance consists of only two rosters. To deal with this issue, we solved these instances again after partitioning them according to the PRP algorithm described in Section 6.4.2. For each instance, the parameter $n_{\min}$, that corresponds to the minimal number of positions in a subroster, was fixed to 6 based on the results of preliminary computational experiments. The results for instance $I_{G,2,333}$ with values of $n_{\min}$ ranging from 6 to 12 in the no-budget case are presented in Table 6.10. In addition to the usual information, this table contains the size of the largest subroster $v^* = \max_{r' \in \mathbf{R}'} |r'|$, which is equal to the optimal value of model (6.31) – (6.34). Moreover, since $\pi(\tilde{x})$ of the partitioned instances may differ from the one of the original instance $I_{G,2,333}$, the column containing the ratio $\frac{\pi(x)}{\pi(\tilde{x})}$ was replaced with two columns, reporting $\pi(\tilde{x})$ and $\pi(x)$. On the other hand, the value of $\omega(\tilde{x})$, which, in general, may also differ from the one of the original instance, was not reported in this table because it happens to stay the same (i.e., $\omega(\tilde{x}) = 13531$), no matter the value of $n_{\min}$. Note that the time required to solve the partitioning model (6.31) – (6.34) does not appear in the table since it is negligible (i.e., less than 1 second).

As can be seen from Table 6.10, as the value of $n_{\min}$ increases, the solution time and the deterioration of workload balance increase as well, whereas the number of preference violations tends to decrease. The selection of $n_{\min}$ can then be done based on a trade-off between the solution time, the number of preference violations and the workload balance. In the case at hand, solution time increases considerably for $n_{\min} \geq 9$, while the reduction in the value of $\pi(x)$ is not very large. For this reason, we restricted our choice to $n_{\min} \leq 8$. For the experiments, we decided to select $n_{\min} = 6$ because we opted for a smaller deterioration of the workload balance rather than a greater reduction of the number of preference violations. That said, any of 6, 7 or 8 seems to be an appropriate choice for $n_{\min}$.

Since partitioning an instance modifies the original problem by adding new (sub)rosters, the number of preference violations and the optimal excess workload cost may not be the same as the ones of the original instance presented in Table 6.3. These values are reported in Table 6.11 for the partitioned instances $\tilde{I}_{G,22,165}$, $\tilde{I}_{G,24,180}$ and $\tilde{I}_{G,47,333}$ coming from $I_{G,2,165}$, $I_{G,2,180}$

Table 6.10 Results when varying the minimum number of positions per subroster.

$n_{\min}$	v^*	# Var.	# Const.	# Nodes	# Cuts	Time (s)	BIT (s)	$\pi(\tilde{x})$	$\pi(x)$	$\frac{\omega(x)}{\omega(\tilde{x})}$
6	16	23120	510312	4059	563	46	23	558	315	1.21
7	16	24321	518526	8023	371	60	19	567	300	1.22
8	16	25374	522536	4060	276	78	28	567	266	1.25
9	24	28080	530335	54662	1242	1248	80	566	247	1.29
10	24	27848	531712	63715	390	1649	124	565	248	1.30
11	24	29711	539565	1117447	1044	43200	290	579	224	1.26
12	93	66806	568626	1691	1254	43200	42486	560	216	1.27

and $I_{G,2,333}$, respectively. Note that, as shown by the instance names, the partitioning of these instances resulted in 22, 24, and 47 subrosters, respectively.

Table 6.11 DAP solution characteristics of the partitioned instances.

Instance	Workload Cost	# Violations	% of Violated Preferences
$\tilde{I}_{G,22,165}$	6222	296	55
$\tilde{I}_{G,24,180}$	7853	328	54
$\tilde{I}_{G,47,333}$	13531	558	50

Moreover, as mentioned at the end of Section 6.4.2, in the solution obtained after partitioning the instances, the drivers are assigned cyclically to all the positions of a subroster, but not to all the positions of the original roster. By cycling over less positions, the drivers assigned to a subroster are not expected to have exactly the same workload on average as they could have by cycling over the positions of the original roster. To evaluate the impact of using subrosters on the balancing of the workload per driver over a long period of time, we collected workload information from the solution of the partitioned instance $\tilde{I}_{G,47,333}$ computed by the PRP heuristic. Before presenting these results, we introduce some notation. Given a solution $\tilde{x}$ to instance $\tilde{I}_{G,47,333}$, define the *average workload per position of the subroster* $r' \in \mathbf{R}'$ as

$$\mu(r') := \frac{1}{|\mathbf{P}_{r'}|} \left(\sum_{p \in \mathbf{P}_{r'}} \sum_{j \in \mathbf{J}_p^W} \sum_{d \in \mathbf{D}_{pj}^R} W_d \tilde{x}_{pj}^d \right).$$

The same definition can be applied to a given roster $r \in \mathbf{R}$ of the original instance $I_{G,2,333}$. Let $r_1, \ldots, r_{|\mathbf{R}|}$ be all the rosters of this instance, and let $r_i^1, r_i^2, \ldots, r_i^{n_{r_i}}$ be all the subrosters of r_i, $i = 1, \ldots, |\mathbf{R}|$ of the instance $\tilde{I}_{G,47,333}$. Define the *relative average workload difference*

between the i-th roster and its j-th subroster as

$$\delta(i,j) := \frac{\mu(r_i^j) - \mu(r_i)}{\mu(r_i)},$$

where $i = 1, \ldots, |\mathbf{R}|$ and $j = 1, \ldots, n_{r_i}$.

The values of $\left|\mathbf{P}_{r_i^j}\right|$, $\mu(r_i^j)$ and $\delta(i,j)$ for each pair of roster and subroster are reported in Table 6.12. Note that $n_{r_1} = 25$, $n_{r_2} = 22$ and the average workloads per position of the original rosters are $\mu(r_1) = 2380.17$ and $\mu(r_2) = 2216.39$. It appears from this table that the difference in average workload per position between each subroster and its parent roster is less than 2%, which, in terms of paid time, represents a difference of less than 48 minutes for an entire week. This suggests that splitting a roster into several subrosters according to our partitioning model does not have an important impact on the workload balance of the individual (sub)rosters. Moreover, it should be pointed out that cycling over a very large roster can take several years (e.g., it takes almost 3.5 years to cycle over roster r_1, that contains 180 positions), which, in practice, is often greater than the period of time during which an actual work schedule is valid. Indeed, changes in the network or in the collective agreement rules often require the work schedules to be updated more frequently. As a consequence, even though the average workload per position is the same for all the drivers of a large roster, the total workload over the actual validity period of a work schedule (which is often smaller than the cycle-length of the roster) may vary significantly from one driver to another. By decomposing large cycle-length rosters into smaller cycle-length ones, we increase the chance that the actual weekly average workload of a driver is closer to the average workload per position of the roster to which it is assigned.

In Tables 6.13, 6.14 and 6.15, we present the results obtained by the PRP algorithm with $B = \infty$, $B = 20\%$ and $B = 5\%$, respectively. For any value of the budget, an optimal solution to the MILP involved in step 3 of the PRP matheuristic was found in less than 30 seconds for the two smallest instances and in less than 5 minutes for the largest one. These relatively short computational times can likely be attributed to a substantial decrease (of more than 83% on average) in the number of variables in the model resulting from the partitioned instances. As in the previous subsection, we observe again that, for all instances, the ratio $\frac{\pi(x)}{\pi(\tilde{x})}$ does not change with the allocated budget, showing again that there are multiple solutions with the same number of preference violations but with different workload costs.

The optimal solutions of the partitioned instances offer an average reduction of 41% in the number of preference violations, which is considerably greater than the average reduction obtained by the PDR solutions when considering a finite budget (namely, $\approx 9\%$ for $B = 5\%$

Table 6.12 Average workload per position of the subrosters of the partitioned instance $\tilde{I}_{G,47,333}$.

r_1				r_2			
j	$\left\lvert\mathbf{P}_{r_1^j}\right\rvert$	$\mu(r_1^j)$	$\delta(1,j)$ (%)	j	$\left\lvert\mathbf{P}_{r_2^j}\right\rvert$	$\mu(r_2^j)$	$\delta(2,j)$ (%)
1	16	2410.00	1.25	1	11	2203.27	-0.59
2	8	2352.50	-1.16	2	10	2210.60	-0.26
3	11	2360.05	-0.85	3	7	2211.71	-0.21
4	7	2426.71	1.96	4	6	2221.17	0.22
5	6	2388.17	0.34	5	7	2203.29	-0.59
6	6	2361.00	-0.81	6	7	2242.43	1.17
7	6	2354.67	-1.07	7	7	2231.71	0.69
8	6	2412.00	1.34	8	6	2209.50	-0.31
9	6	2378.00	-0.09	9	6	2215.00	-0.06
10	6	2371.67	-0.36	10	8	2212.88	-0.16
11	6	2393.00	0.54	11	8	2238.88	1.01
12	6	2403.00	0.96	12	8	2232.88	0.74
13	6	2400.50	0.85	13	7	2215.43	-0.04
14	6	2403.17	0.97	14	6	2215.17	-0.06
15	6	2358.33	-0.92	15	6	2218.50	0.10
16	6	2396.67	0.69	16	6	2210.83	-0.25
17	6	2340.33	-1.67	17	6	2219.83	0.16
18	8	2377.38	-0.12	18	6	2228.50	0.55
19	9	2360.44	-0.83	19	6	2201.50	-0.67
20	7	2370.86	-0.39	20	6	2208.00	-0.38
21	6	2359.67	-0.86	21	7	2209.71	-0.30
22	6	2378.83	-0.06	22	6	2200.17	-0.73
23	6	2351.67	-1.20				
24	6	2405.33	1.06				
25	12	2370.67	-0.40				

and $\approx -3\%$ for $B = 5\%$), and is similar to the average reduction achieved by the PDR solutions with an infinite budget (namely, $\approx 44\%$). However, it should be pointed out that, in this latter case, the workload cost increases significantly (i.e., by about 35%). Partitioning the G-instances has then allowed to find, within short computational times, solutions that offer a good tradeoff between workload balancing and preference satisfaction.

6.6 Conclusion and perspective

In this paper, we studied a variant of the driver rostering problem, the DAPDP, that considers drivers' preferences when assigning the duties to the rosters. We then modeled the

Table 6.13 Results of the PRP matheuristic without budget.

Instance	# Var.	# Const.	# Nodes	# Cuts	Time (s)	BIT (s)	$\frac{\pi(x)}{\pi(\tilde{x})}$	$\frac{\omega(x)}{\omega(\tilde{x})}$
$\tilde{I}_{G,22,165}$	11802	223454	4086	414	11	6	0.625	1.32
$\tilde{I}_{G,24,180}$	13351	240427	0	507	7	7	0.579	1.29
$\tilde{I}_{G,47,333}$	23120	510312	4059	563	46	23	0.565	1.21

Table 6.14 Results of the PRP matheuristic with a budget of 20%.

Instance	# Var.	# Const.	# Nodes	# Cuts	Time (s)	BIT (s)	$\frac{\pi(x)}{\pi(\tilde{x})}$	$\frac{\omega(x)}{\omega(\tilde{x})}$
$\tilde{I}_{G,22,165}$	11802	223455	2903	626	14	8	0.625	1.20
$\tilde{I}_{G,24,180}$	13351	240428	518	723	13	13	0.579	1.20
$\tilde{I}_{G,47,333}$	23120	510313	19566	1339	258	59	0.565	1.20

DAPDP as a MILP. We showed that a commercial solver can be used directly to solve small instances of this problem (i.e., with up to 124 drivers and 490 duties) while maintaining the excess workload cost within 20 % of the optimal one. For larger instances, we proposed two matheuristics based on this MILP. The first one restricts the number of duty assignment variables by preassigning duties to rosters. With this algorithm, it was possible to solve medium- and large-sized instances (i.e., with up to 202 drivers and 881 duties) that contain at least 5 rosters. The second heuristic makes use of a set partitioning model to decompose large cycle-length rosters into smaller cycle-length subrosters. The decomposed model can then be solved directly or by applying the first matheuristic. We showed that this algorithm was able to solve within short computational time medium- and large-sized instances that consist of a relatively small number of rosters, each containing a large number of positions. Moreover, the solutions produced by these two matheuristics maintain the total workload in excess of the average position workload within 5 % of the optimal one.

Table 6.15 Results of the PRP matheuristic with a budget of 5%.

Instance	# Var.	# Const.	# Nodes	# Cuts	Time (s)	BIT (s)	$\frac{\pi(x)}{\pi(\tilde{x})}$	$\frac{\omega(x)}{\omega(\tilde{x})}$
$\tilde{I}_{G,22,165}$	11802	223455	5146	665	23	21	0.625	1.05
$\tilde{I}_{G,24,180}$	13351	240428	3971	692	26	25	0.579	1.05
$\tilde{I}_{G,47,333}$	23120	510313	11949	1259	240	235	0.565	1.05

As a direct extension of this work, further computational experiments could be carried out in order to test the steadiness preferences (6.28) – (6.29). In particular, one could investigate the effect of the parameters $\underline{H}$ and $\overline{H}$ (see Section 6.3.2) on the solution. Moreover, it would be interesting to devise an algorithm to determine the optimal value of $n_{\min}$ in the partitioning model presented in Section 6.4.2.

On a more general note, one could attempt to solve DAPDP at the same time as the days off scheduling problem. An integrated approach would give more flexibility in the construction of the sequences of duties, which should allow to achieve a lower number of preference violations in conjunction with a better workload balance. Additionally, further research could be conducted to study the possibility of taking preferences into account in the duty scheduling step.

Acknowledgements

We would like to thank the personnel of Giro Inc. for providing real-world datasets.

CHAPITRE 7 DISCUSSION GÉNÉRALE

La recherche menée dans cette thèse a pour objectif ultime de concevoir des algorithmes efficaces de résolution de problèmes de roulements de chauffeurs d'autobus. Ce domaine d'application est peu abordé dans la littérature en comparaison à d'autres domaines. Ce projet a été défini en collaboration avec GIRO Inc., et présente l'un des problèmes les plus difficiles auxquels sont confrontés certains clients européens de GIRO.

Les algorithmes en question visent à résoudre en particulier des instances de grande taille. Comme expliqué dans les chapitres 4 et 5, ce type de problème ne comprend pas de règles liées aux quarts de travail ni de règles souples d'affectation qui peuvent réduire l'espace de recherche. En outre, la plupart des journées de travail peut être assignée à tous les roulements, ce qui complique encore plus le problème et ne facilite pas le processus de recherche de solution. À notre connaissance, aucun algorithme n'a été proposé pour résoudre ce type de problème spécifique.

Le premier objectif de la thèse est de tester différentes formulations du problème de roulements avec jours de repos fixés. Les instances réelles ont été résolues à l'optimalité dans des temps de calcul raisonnables. Des formulations fortes ont été utilisées pour modéliser les règles relatives aux repos de nuit. Ces formulations ont permis d'accélérer le processus de résolution pour des grandes instances comportant jusqu'à 202 chauffeurs. Les roulements construits sont de bonne qualité, bien équilibrés et satisfont toutes les règles strictes d'affectation des journées de travail.

Le problème de roulements traité dans le deuxième objectif intègre l'affectation des jours de repos et des journées de travail. Nous avons présenté un modèle mathématique en nombres entiers. À l'aide d'un solveur commercial MILP, nous avons pu résoudre à l'optimalité des instances de taille moyenne allant à 146 chauffeurs et 640 journées de travail. Sur la base de ce modèle, nous avons conçu une approche de résolution à deux étapes qui a produit des solutions de bonne qualité, et qui sont optimales la plupart du temps. Les temps de calcul étaient relativement courts vu la grande taille des instances. Nous avons montré que les solutions obtenues par l'approche séquentielle ont permis de résoudre le problème intégré en utilisant un démarrage à chaud. Ceci a résulté en des solutions optimales dans des temps de calcul très réduits.

Dans le troisième objectif de la thèse, nous avons traité les préférences des chauffeurs d'autobus dans la construction des roulements. Nous avons proposé une nouvelle approche de résolution pour ce type de problème. D'abord, nous avons proposé un modèle linéaire mixte

en nombres entiers, ce dernier a été résolu à l'optimalité pour des petites instances allant à 124 chauffeurs et 490 journées de travail. Pour le reste des instances, nous avons développé un algorithme se basant d'abord sur l'affectation des journées de travail aux roulements sans prendre en considération les préférences. Les solutions obtenues ont permis de créer de nouvelles possibilités réduites d'affectation des journées de travail, qui ont été utilisées dans la résolution du problème de roulements avec préférences de chauffeurs d'autobus. Un modèle de partitionnement d'ensemble a été utilisé pour permettre la décomposition des grands groupes d'employés en sous-groupes de petite à moyenne taille. Finalement, l'approche réduisant les possibilités d'affectation des journées de travail a été appliquée pour ces nouveaux sous-groupes d'employés, ce qui a permis d'avoir une réduction remarquable des temps de calcul.

À notre connaissance, cette thèse présente le premier ouvrage scientifique qui traite ces cas spécifiques de problèmes de roulements en transport urbain. Nous avons pu résoudre des instances compliquées et de grande taille jusqu'à l'optimalité. Nous avons utilisé la haute symétrie et la combinatoire du modèle en notre faveur, en montrant que la méthode séquentielle peut être bénéfique pour la résolution intégrée des problèmes de roulements et en pratique. L'intégration des préférences dans le problème de roulements a montré que la complexité ne diminue pas toujours avec l'ajout de contraintes souples surtout lors de la conception d'algorithmes exacts de résolution. Finalement, nos travaux ont permis de proposer des méthodes de programmation mathématique pour résoudre des problèmes pratiques et de taille industrielle.

CHAPITRE 8 CONCLUSION ET RECOMMANDATIONS

Dans cette thèse, nous avons poursuivi trois objectifs visant à résoudre efficacement le problème de roulements des chauffeurs d'autobus. Le premier objectif vise à résoudre le problème de roulements avec jours de repos fixé. La résolution de ce dernier est utile pour les deux autres objectifs de la thèse. L'un des premiers modèles mathématiques proposés sera utilisé dans la résolution du problème intégré de roulements et dans le problème de roulements avec préférences de chauffeurs d'autobus.

Dans ce chapitre, nous synthétisons, dans la section 8.1, l'ensemble des travaux effectués dans les chapitres principaux (4, 5 et 6). Ensuite, dans la section 8.2, nous présentons quelques limitations des approches proposées tout en suggérant des pistes d'améliorations futures.

8.1 Synthèse des travaux

Dans le chapitre 4, nous avons modélisé deux types de problème de roulements avec jours de repos fixés. Les deux problèmes sont différents en termes de taille des instances et aussi du type de règles qui régissent l'affectation des journées de travail. Une formulation forte des contraintes relatives au repos de nuit a été proposée et comparée à une autre formulation proposée dans la littérature. Nous avons montré que cette nouvelle modélisation donne des relaxations linéaires plus serrées et engendre une résolution avec un nombre réduit de coupes générées et de nœuds explorés dans l'arbre de branchement. Les modèles proposés ont été testés avec deux fonctions objectifs différentes mais qui visent à équilibrer la charge de travail entre les chauffeurs.

Pour le deuxième objectif (chapitre 5), nous avons conçu un algorithme exact pour résoudre le problème de roulements intégré de construction de jours de repos et d'affectation de journées de travail. L'algorithme peut résoudre des instances de grande taille. Dans notre problème, il n'y a pas de notion de quarts de travail, et par conséquent, pas de règles liées aux quarts de travail, ni de règles souples (préférences d'employés par exemple) qui peuvent réduire l'espace de solution. De plus, le modèle proposé présente beaucoup de symétrie puisque la plupart des journées de travail peuvent être affectées à tous les groupes de chauffeurs. Nous avons introduit un modèle de programmation linéaire mixte qui peut être résolu en utilisant un solveur MILP commercial. Avec ce modèle, les instances de taille moyenne allant à 146 chauffeurs et 640 journées de travail sont résolues à l'optimalité dans moins de deux heures de temps de calcul. Sur la base de ce modèle, nous avons proposé une matheuristique à deux

étapes pouvant produire des solutions de haute qualité (optimales, la plupart du temps) en des temps de calcul relativement courts pour les instances de grande taille. Nous nous sommes servis de la solution obtenue pour effectuer un démarrage à chaud pour la résolution intégrée du problème. Les tests effectués sur deux instances réelles, et cinq autres instances qui en dérivent, ont montré que la matheuristique donne des solutions optimales dans des temps de calculs relativement courts (moins de 4 heures pour la grande instance impliquant 333 chauffeurs).

Dans le chapitre 6, nous avons présenté une nouvelle variante du problème de roulements avec jours de repos fixés. Celle-ci prend en compte les préférences des chauffeurs d'autobus. Nous avons développé un nouveau programme linéaire mixte, qui peut être utilisé directement pour la résolution de petites instances allant à 124 chauffeurs et 490 journées de travail. Nous avons pu maintenir un coût de l'objectif qui n'excède pas le coût optimal de plus de 20%. Pour les instances les plus difficiles, nous avons conçu deux matheuristiques. La première matheuristique se base sur la solution du problème d'affectation des journées de travail issue de l'objectif 2 (chapitre 5) et permet de réduire les possibilités d'affectation des journées de travail aux différents roulements. Grâce à cette réduction, nous avons pu résoudre le problème de roulements avec préférences pour des moyennes et grandes instances allant jusqu'à 202 chauffeurs et 881 journées de travail. La deuxième matheuristique fait appel, d'abord, à un modèle de partitionnement d'ensemble pour décomposer les roulements de grande taille (allant à 180 chauffeurs dans le même roulement) en des sous-roulements de taille petite à moyenne. Ensuite, le modèle est résolu pour les nouvelles instances décomposées en appliquant la première matheuristique. L'algorithme était capable de résoudre, en un temps de calcul court, des instances de moyenne et grande tailles. De plus, les valeurs de l'objectif, équilibrant la charge de travail entre les chauffeurs, obtenues ne dépassent pas 5% des coûts optimaux.

8.2 Limitations des solutions proposées et améliorations futures

L'étude effectuée dans le cadre de cette thèse concerne des problèmes de roulements issus de compagnies de transport européennes. Donc la résolution de ces problèmes et les approches proposées sont fortement liées au type des problèmes et des règles qui les régissent. Ceci ne contredit pas le fait que les approches peuvent s'étendre à d'autres types de problèmes de roulements, que cela soit en transport urbain ou autre. Cependant, il faut adapter ces approches aux différents problèmes à étudier.

Le problème de roulements est un problème complexe, qui présente beaucoup de symétrie et de combinatoire. Malgré le fait que les résultats obtenus dans notre contexte soient de bonne

qualité, il se peut que notre approche génère des solutions de piètre qualité dans d'autres domaines d'application, ou même dans un autre contexte de transport urbain (contexte non cyclique, ou avec des règles différentes d'affectation de journées de travail). Cette limitation réside dans le fait que les journées de travail sont construites de façon indépendante et ne sont pas prises en charge lors de la construction des roulements. Une recherche future peut considérer la construction des journées de travail simultanément avec le problème de roulements, tout en minimisant le temps inactif dans les journées de travail et en maximisant l'équité entre les roulements ou les chauffeurs d'autobus. Ce nouveau type de problème peut être modélisé comme un programme mathématique et résolu avec une approche de génération de colonnes combinée avec une méthode d'agrégation des contraintes. Les temps de calcul peuvent être accélérés par une méthode de recherche à grand voisinage. Le nouveau modèle servira à générer dynamiquement les journées de travail et les affecter simultanément aux roulements.

Le problème de roulements avec préférences d'employés a été résolu en se basant sur le problème de roulements avec jours de repos fixés. Un autre défi potentiel est de résoudre le problème intégré de jours de repos et roulements avec préférences. Une approche intégrée apporterait plus de souplesse lors de la construction des séquences de jours de repos et de journées de travail, ce qui diminuerait le nombre de violations des préférences et permettrait une meilleure équité de charge de travail. En outre, d'autres recherches pourraient être menées pour intégrer le traitement des préférences dans l'étape de construction des journées de travail.

RÉFÉRENCES

H. K. Alfares, “An efficient two-phase algorithm for cyclic days-off scheduling”, *Computers & Operations Research*, vol. 25, no. 11, pp. 913–923, 1998.

R. Bailey, K. Garner, et M. Hobbs, “Using simulated annealing and genetic algorithms to solve staff-scheduling problems”, *Asia-Pacific Journal of Operational Research*, vol. 14, no. 2, p. 27, 1997.

V. M. M. Barbosa, “Bus driver rostering by hybrid methods based on column generation”, Thèse de doctorat, Universidade de Lisboa (Portugal), 2018.

J. F. Bard et H. W. Purnomo, “A column generation-based approach to solve the preference scheduling problem for nurses with downgrading”, *Socio-Economic Planning Sciences*, vol. 39, no. 3, pp. 193–213, 2005.

J. F. Bard et H. W. Purnomo, “Preference scheduling for nurses using column generation”, *European Journal of Operational Research*, vol. 164, no. 2, pp. 510–534, 2005.

J. F. Bard et H. W. Purnomo, “Cyclic preference scheduling of nurses using a lagrangian-based heuristic”, *Journal of Scheduling*, vol. 10, no. 1, pp. 5–23, 2007.

B. T. Bennett, “Optimization of bus crew rosters : an application of combinatorial mathematics.” Thèse de doctorat, University of Adelaide, 1967.

L. Bianco, M. Bielli, A. Mingozzi, S. Ricciardelli, et M. Spadoni, “A heuristic procedure for the crew rostering problem”, *European journal of operational research*, vol. 58, no. 2, pp. 272–283, 1992.

R. Borndörfer, M. Reuther, T. Schlechte, C. Schulz, E. Swarat, et S. Weider, “Duty rostering in public transport - facing preferences, fairness, and fatigue”, ZIB, Takustr. 7, 14195 Berlin, Rapp. tech. 15-44, 2015.

J. O. Brunner et G. M. Edenharter, “Long term staff scheduling of physicians with different experience levels in hospitals using column generation”, *Health care management science*, vol. 14, no. 2, pp. 189–202, 2011.

A. Caprara, P. Toth, D. Vigo, et M. Fischetti, “Modeling and solving the crew rostering problem”, *Operations research*, vol. 46, no. 6, pp. 820–830, 1998.

P. Carraresi et G. Gallo, “A multi-level bottleneck assignment approach to the bus drivers’ rostering problem”, *European Journal of Operational Research*, vol. 16, no. 2, pp. 163–173, 1984.

G. B. Dantzig, A. Orden, et P. Wolfe, “The generalized simplex method for minimizing a linear form under linear inequality restraints”, *Pacific Journal of Mathematics*, vol. 5, no. 2, pp. 183–195, 1955.

G. Desaulniers et M. D. Hickman, “Public transit”, *Handbooks in operations research and management science*, vol. 14, pp. 69–127, 2007.

S. Er-Rbib, G. Desaulniers, I. El Hallaoui, et A. Bani, “Integrated and sequential solution methods for the cyclic bus driver rostering problem”, *Journal of the Operational Research Society*, pp. 1–16, 2020.

M. Erhard, J. Schoenfelder, A. Fügener, et J. O. Brunner, “State of the art in physician scheduling”, *European Journal of Operational Research*, vol. 265, no. 1, pp. 1 – 18, 2018.

A. Ernst, M. Krishnamoorthy, et D. Dowling, “Train crew rostering using simulated annealing”, *Proceedings of ICOTAÕ98, Perth*, 1998.

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, no. 1-4, pp. 21–144, 2004.

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, no. 1, pp. 3–27, 2004.

M. Gendron, “Détermination de la taille des effectifs et affectation des séquences de repos dans les horaires d’employés de compagnies de transport public”, Mémoire de maîtrise, Polytechnique Montréal, 2012.

M. Gröbner et P. Wilke, “Optimizing employee schedules by a hybrid genetic algorithm”, dans *Workshops on Applications of Evolutionary Computation*. Springer, 2001, pp. 463–472.

T. Hanne, R. Dornberger, et L. Frey, “Multiobjective and preference-based decision support for rail crew rostering”, dans *2009 IEEE Congress on Evolutionary Computation*. IEEE, 2009, pp. 990–996.

A. Hartog, D. Huisman, E. J. Abbink, et L. G. Kroon, “Decision support for crew rostering at ns”, *Public Transport*, vol. 1, no. 2, pp. 121–133, 2009.

S. Jütte, D. Müller, et U. W. Thonemann, “Optimizing railway crew schedules with fairness preferences”, *Journal of Scheduling*, vol. 20, no. 1, pp. 43–55, 2017.

N. Karmarkar, “A new polynomial-time algorithm for linear programming”, dans *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, 1984, pp. 302–311.

S. Kirkpatrick, C. D. Gelatt, et M. P. Vecchi, “Optimization by simulated annealing”, *science*, vol. 220, no. 4598, pp. 671–680, 1983.

P. Kisielewski, M. Dańda, et M. Bauer, “Optimization of rosters in public transport”, dans *2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. IEEE, 2019, pp. 1–8.

F. Knust et L. Xie, “Simulated annealing approach to nurse rostering benchmark and real-world instances”, *Annals of Operations Research*, vol. 272, no. 1, pp. 187–216, Jan 2019.

J. Kyngäs et K. Nurmi, “Days-off scheduling for a bus transportation company”, *International Journal of Innovative Computing and Applications*, vol. 3, no. 1, pp. 42–49, 2011.

M. Lezaun, G. Pérez, et E. Sáinz de la Maza, “Crew rostering problem in a public transport company”, *Journal of the Operational Research Society*, vol. 57, no. 10, pp. 1173–1179, 2006.

J. Ma, T. Liu, et W. Zhang, “A genetic algorithm approach to the balanced bus crew rostering problem”, *Journal of Traffic and Logistics Engineering Vol*, vol. 2, no. 1, 2014.

M. Mesquita, M. Moz, A. Paias, J. Paixão, M. Pato, et A. Respício, “A new model for the integrated vehicle-crew-rostering problem and a computational study on rosters”, *Journal of Scheduling*, vol. 14, no. 4, pp. 319–334, 2011.

M. Mesquita, M. Moz, A. Paias, et M. Pato, “A decompose-and-fix heuristic based on multi-commodity flow models for driver rostering with days-off pattern”, *European Journal of Operational Research*, vol. 245, no. 2, pp. 423–437, 2015.

A. Monfroglio, “Hybrid genetic algorithms for a rostering problem”, *Software : Practice and Experience*, vol. 26, no. 7, pp. 851–862, 1996.

M. Moz et M. V. Pato, “A genetic algorithm approach to a nurse rerostering problem”, *Computers & Operations Research*, vol. 34, no. 3, pp. 667–691, 2007.

M. Moz, A. Respício, et M. V. Pato, “Bi-objective evolutionary heuristics for bus driver rostering”, *Public Transport*, vol. 1, no. 3, p. 189, 2009.

É. Naudin, P. Y. Chan, M. Hiroux, T. Zemmouri, et G. Weil, “Analysis of three mathematical models of the staff rostering problem”, *Journal of Scheduling*, vol. 15, no. 1, pp. 23–38, 2012.

T. Nishi, T. Sugiyama, et M. Inuiguchi, “Two-level decomposition algorithm for crew rostering problems with fair working condition”, *European Journal of Operational Research*, vol. 237, no. 2, pp. 465–473, 2014.

K. Nurmi, J. Kyngas, et G. Post, “Driver rostering for bus transit companies”, *Engineering Letters*, vol. 19, no. 2, p. 125, 2011.

K. Nurmi, J. Kyngäs, et G. Post, “Driver rostering for a finnish bus transportation company”, dans *Iaeng Transactions On Engineering Technologies Volume 7*. World Scientific, 2012, pp. 15–35.

E. Özcan, “Memetic algorithms for nurse rostering”, dans *International Symposium on Computer and Information Sciences*. Springer, 2005, pp. 482–492.

D. Pedrosa et M. Constantino, “Days-off scheduling in public transport companies”, dans *Computer-Aided Scheduling of Public Transport*. Springer, 2001, pp. 215–232.

K. Peng, Y. Shen, et J. Li, “A multi-objective simulated annealing for bus driver rostering”, dans *Bio-Inspired Computing-Theories and Applications*. Springer, 2015, pp. 315–330.

J. Prakash, S. Sinha, et S. Sahay, “Bus transportation crews planning by goal programming”, *Socio-Economic Planning Sciences*, vol. 18, no. 3, pp. 207–210, 1984.

H. W. Purnomo et J. F. Bard, “Cyclic preference scheduling for nurses using branch and price”, *Naval Research Logistics (NRL)*, vol. 54, no. 2, pp. 200–220, 2007.

M. S. Sodhi et S. Norris, “A flexible, fast, and optimal modeling approach applied to crew rostering at london underground”, *Annals of Operations Research*, vol. 127, no. 1-4, pp. 259–281, 2004.

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, no. 3, pp. 367–385, 2013.

L. A. Wolsey, *Integer programming.* Wiley, 1998.

L. A. Wolsey et G. L. Nemhauser, *Integer and combinatorial optimization.* John Wiley & Sons, 1999, vol. 55.

T. Wong, M. Xu, et K.-S. Chin, "A two-stage heuristic approach for nurse scheduling problem : A case study in an emergency department", *Computers & Operations Research*, vol. 51, pp. 99–110, 2014.

L. Xie, "Metaheuristics approach for solving multi-objective crew rostering problem in public transit", Working paper. University of Paderborn, Germany, Rapp. tech., 2013.

L. Xie, *Decision Support for Crew Rostering in Public Transit : Web-Based Optimization System for Cyclic and Non-Cyclic Rostering.* Springer, 2014.

L. Xie et L. Suhl, "Cyclic and non-cyclic crew rostering problems in public bus transit", *Or Spectrum*, vol. 37, no. 1, pp. 99–136, 2015.

L. Xie, N. Kliewer, et L. Suhl, "Integrated driver rostering problem in public bus transit", *Procedia - Social and Behavioral Sciences*, vol. 54, pp. 656 – 665, 2012, proceedings of EWGT2012 - 15th Meeting of the EURO Working Group on Transportation, September 2012, Paris.

ANNEXE A ADDITIONAL INFORMATION ON THE CBDRP INSTANCES

For each roster of the instance $I_{A,14,202}$, Table A.1 indicates the number of positions it contains and the number of days off required per week.

Table A.1 Characteristics of the rosters in instance $I_{A,14,202}$.

Roster Id	#Positions	#Days off	Roster Id	#Positions	#Days off
RA_1	18	2	RA_{7b}	18	2
RA_3	6	2	RA_8	6	3
RA_4	12	2	RA_9	12	3
RA_6	18	2	RA_{9m}	12	3
RA_{6b}	6	2	RA_{10}	35	3
RA_7	12	2	RA_{11}	21	3
RA_{7a}	12	2	RA_{11m}	14	3

The three instances $I_{A,5,110}$, $I_{A,6,124}$ and $I_{A,10,146}$ were created by choosing the rosters from instance $I_{A,14,202}$ that are listed in Table A.2.

Table A.2 Rosters chosen in instances $I_{A,5,110}$, $I_{A,6,124}$ and $I_{A,10,146}$.

Instance	List of rosters
$I_{A,5,110}$	RA_1, RA_6, RA_{7b}, RA_{10}, RA_{11}
$I_{A,6,124}$	RA_3, RA_4, RA_7, RA_{10}, RA_{11}, RA_{11m}
$I_{A,10,146}$	RA_1, RA_4, RA_{6b}, RA_7, RA_{7a}, RA_8, RA_9, RA_{9m}, RA_{10}, RA_{11}

Recall that instance $I_{G,2,333}$ contains two rosters: RG_1 with 180 positions and two days off per week, and RG_2 with 153 positions and three days off per week. Instance $I_{G,2,165}$ contains 89 positions of roster RG_1 and 76 positions of RG_2, whereas instance $I_{G,2,180}$ involves 97 positions of RG_1 and 83 positions of RG_2.