Mathematical models and algorithms for the home care routing and scheduling problem

Mohamed Cissé

UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

École Doctorale Mathématiques, Informatique, Physique théorique, Ingénierie des systèmes

LABORATOIRE d'INFORMATIQUE

THÈSE présentée par :

CISSÉ Mohamed

soutenu le 21 Juin 2017

pour obtenir le grade de : Docteur de l'Université François - Rabelais de Tours

Discipline/Spécialité : INFORMATIQUE

Modèles mathématiques et algorithmes pour la résolution du problème de tournées du personnel de soins à domicile

THÈSE dirigée par :

Lenté Christophe	Maître de conférences (HDR),	Université François Rabelais de Tours
Kergosien Yannick	Maître de conférences,	Université François Rabelais de Tours

RAPPORTEURS :

Feillet Dominique	Professeur des universités,	École des Mines de Saint-Etienne
Prins Christian	Professeur des universités,	Université de technologie de Troyes

JURY :

Prins Christian	Professeur des universités,	Université de technologie de Troyes
Feillet Dominique	Professeur des universités,	École des Mines de Saint-Etienne
Ruiz Angel	Professeur titulaire,	Université Laval
Garaix Thierry	Assitant Professeur,	École des Mines de Saint-Etienne
Lenté Christophe	Maître de conférences (HDR),	Université François Rabelais de Tours
Kergosien Yannick	Maître de conférences,	Université François Rabelais de Tours

Résumé

Le soin à domicile est un secteur en plein essor ces dernières années. Cela est dû au vieillissement de la population, à la volonté de réduire les coûts hospitaliers et d'assurer le bien-être du patient en le gardant dans son cadre familial tout en maintenant la qualité des soins. L'organisation de ces soins nécessite une prise de décisions aux niveaux stratégique, tactique et opérationnel. Cette thèse s'articule autour de l'étude de problèmes apparaissant uniquement au niveau opérationnel. Ces problèmes traitent de la planification des tournées du personnel de soins à domicile. La première étape de cette étude a consisté à faire une revue de la littérature. De nombreux modèles mathématiques ont été formulés dans la littérature. Cependant, ces modèles étaient dédiés à une structure de soins à domicile spécifique et pouvaient être difficilement transposés. Nous proposons ici une approche générique tant du point de vue de la modélisation que des méthodes de résolutions. À cet effet, nous avons identifié les caractéristiques fréquemment rencontrées dans la littérature à travers cette revue de la littérature. Un modèle générique a été proposé prenant en compte la plupart des caractéristiques. Ce modèle générique constitue un socle pour la construction de méthodes de résolution. Deux méthodes de résolution ont été conçues. La première méthode est une méthode par décomposition et la deuxième méthode est un algorithme hybride génétique avec gestion de la population. Ces deux méthodes utilisent des représentations d'une solution issues de la littérature et adaptées aux caractéristiques du problème. Des expérimentations numériques ont été réalisées dans le but d'évaluer les méthodes proposées et de se comparer à la littérature.

Mots clés : Recherche Opérationnelle, Santé, Tournées de véhicules, Soins à domicile, Méta-heuristique

Table des matières

Liste des tableaux

Table des figures

Introduction générale

Cette thèse s'inscrit dans le cadre du projet "Planification et Optimisation au Niveau Opérationnel des Soins à Domicile" (PONOSAD) financé par l'Agence Nationale de la Recherche. Ce projet vise à développer des outils d'aide à la décision pour les professionnels de soins à domicile. Dans un contexte actuel de vieillissement de la population, nous assistons au développement significatif des soins à domicile. Ces soins permettent de désengorger les hôpitaux et par la même occasion de réduire les coûts hospitaliers. À notre connaissance, peu d'outils d'aide à la décision sont adaptés aux besoins de gestion opérationnelle du fait des spécificités de ces soins. Le but du projet PONOSAD est de contribuer à rendre ces soins plus efficients, économes et de meilleure qualité. Ces objectifs peuvent être atteints grâce à une meilleure planification et optimisation des opérations. Ces objectifs passent donc par la conception d'outils génériques, flexibles et robustes afin de les aider dans leurs tâches de planification. Le défi principal est la modélisation et la résolution d'un ensemble de problèmes se rattachant à la gestion des soins à domicile. Ces problèmes sont de nature combinatoire et donc très complexes, et nécessitent par conséquent des outils d'aide à la décision pertinents et performants. Ces problèmes de nature statique ou dynamique qui s'apparentent à certains problèmes répertoriés dans la littérature des problèmes de transports et de tournées de véhicules, sont abordés par des techniques de la Recherche Opérationnelle.

Ces dernières années, de nombreuses études traitant de ces problèmes ont été publiées. Ces études proposaient à fois une formulation d'un problème et une méthode de résolution. Cependant, elles étaient dédiées à une structure de soins à domicile spécifique et ne pouvaient par conséquent pas être facilement transposées à une autre structure. De plus, il était difficile de comparer ces études entre elles. L'objectif de cette thèse est de proposer une approche unificatrice tant du point de vue de la formulation que des méthodes de résolution. L'idée sous-jacente est de couvrir autant que possible la plupart des travaux sus-mentionnés. Cette thèse est constituée de cinq chapitres. Pour chaque chapitre, après une brève introduction où nous soulignons sa portée, nous rappelons systématiquement nos contributions.

Le premier chapitre dresse un état de l'art complet en lien avec cette étude, composé d'un état des lieux des différentes structures de soins à domicile et d'une revue de littérature autour de ce domaine. Il rend également compte des différents modèles développés. Les caractéristiques du problème ont été décrites à partir d'un regroupement basé sur : les patients, les soignants et la structure de soins à domicile. La revue de la littérature a permis de mettre en avant la manière dont ces caractéristiques ont été prises en compte dans les modèles et les différentes méthodes de résolution proposées.

À partir de l'état de l'art, le deuxième chapitre présente une modélisation générique. Cette modélisation générique couvre la plupart des caractéristiques rencontrées dans la littérature. Nous avons tenu compte des exigences des professionnels (ASSAD-HAD en touraine) de soins à domicile avec lesquels nous avons échangés. Ce deuxième chapitre est une tentative d'unification des différents modèles rencontrés dans la littérature. Nous proposons une formulation mathématique et décrivons une méthode de génération d'instances. Ce chapitre constitue le socle à partir duquel la suite de la thèse s'articule.

Les trois derniers chapitres traitent de la conception de méthodes de résolution approchée pour le problème formulé au deuxième chapitre. Nous tenions à proposer des méthodes de résolution générales, flexibles et robustes prenant en compte la plupart des caractéristiques du problème en question.

Le troisième chapitre traite de la représentation d'une solution et de son évaluation. La représentation d'une solution et son évaluation jouent un rôle central dans le processus de conception d'une méthode de résolution approchée. En s'inspirant de la littérature du problème de tournées de véhicules, nous proposons donc deux encodages et plusieurs algorithmes d'évaluation. Des expérimentations numériques clôturent ce troisième chapitre et nous permettent de valider la pertinence de notre approche.

Dans le quatrième chapitre, nous proposons une première méthode de résolution approchée. Cette dernière décompose le problème en différents sous-problèmes et résout séquentiellement ces derniers de manière exacte ou approchée. La solution initiale générée est améliorée grâce à un processus d'intensification à base d'une recherche locale. Afin d'explorer d'autres zone de l'espace de recherche, plusieurs procédures ont été mises en place afin de revenir sur les décisions prises dans les différents sous-problèmes. Grâce aux expérimentations conduites, cette méthode est comparée aux méthodes existantes de la littérature.

Le cinquième chapitre porte sur un algorithme hybride génétique avec gestion de population. Ces dernières années, l'algorithme hybride génétique s'est révélé très efficace quant à la résolution du problème de tournées de véhicules. Ce succès nous a conduit à proposer aussi un algorithme hybride génétique. Différentes stratégies de recherche ont été inclues dans cet algorithme puisque de nombreux travaux ont montré qu'elles l'amélioraient de façon substantielle. De même, les expérimentations menées nous ont permis d'évaluer et de comparer cette méthode de résolution.

Cette thèse se clôt par une conclusion où nous résumons le travail réalisé et dressons les perspectives de recherche que nous envisageons à court, moyen et long terme. Ces perspectives sont à la fois d'ordre théorique et pratique et elles visent à explorer de nouvelles idées.

Chapitre 1

État de l'art

1.1 Introduction

Les soins à domicile sont un secteur en plein essor ces dernières années. Cela est dû au vieillissement de la population, à la volonté de réduire les coûts hospitaliers et d'assurer le bien-être du patient en le gardant dans son cadre familial tout en maintenant la qualité des soins.

Dans cet état de l'art, une revue de la littérature concernant la planification des soins à domicile a été réalisée, en nous focalisant plus particulièrement sur les problèmes de tournées et d'affectations afférents. L'originalité de ces problèmes réside dans le fait que le patient soit au cœur du système de décision, ce qui fait apparaître des contraintes non-usuelles dans les problèmes de tournées. Afin de classifier les articles auxquels nous avons eu accès, les caractéristiques de chaque problème traité ont été extraites. Nous avons identifié les informations sur les types de données à prendre en compte, les contraintes à satisfaire, les différentes modalités de soins, les objectifs à optimiser, les souplesses décisionnelles, etc. L'ensemble de ces informations permet d'analyser les processus opérationnels des structures de soins à domicile, d'identifier clairement les différents besoins d'outils d'aide à la décision et les problèmes combinatoires à résoudre.

Plusieurs revues de la littérature ayant trait à la fois aux soins à domicile et à la recherche opérationnelle ont déjà été publiées [B. Bashir, M. Chabrol, 2012, Gutiérrez et Vidal, 2013] et [Bennett Milburn, 2012]. Ces travaux mettent en exergue les différents problèmes de recherche opérationnelle auxquels font face les structures de soins à domicile. Parmi ces problèmes, quatre ont été unanimement mis en avant : au niveau stratégique, le problème de sectorisation (Districting Problem) [Blais *et al.*, 2003] consiste à diviser une zone géographique afin de regrouper les patients en secteur et affecter à chaque secteur des ressources, ici du personnel de soins ; au niveau tactique, le problème du dimensionnement de ressources (Resource Dimensioning Problem) [Busby et Carter, 2006] permet de déterminer le niveau de ressources nécessaire pour la planification ; au niveau opérationnel, le problème d'affection du personnel de soins (Operator Assignment Problem) [Lanzarone *et al.*, 2012] et le problème de tournées du personnel de soins à domicile (Home Health Care Routing Problem) [Nickel *et al.*, 2012, Braekers *et al.*, 2016] peuvent être identifiés. Ces deux problèmes peuvent être résolus séparément ou simultanément. Le premier consiste

à déterminer le personnel de soins adéquat pour la réalisation d'un ensemble de services tandis que le second détermine en plus l'ordre dans lequel ces services vont être réalisés. Dans le cadre de cette thèse, seul le problème de tournées du personnel de soins à domicile est abordé.

Ce chapitre est organisé de la manière suivante : dans la section 1.2, le problème de tournées du personnel de soins à domicile est brièvement décrit tout en faisant le lien avec le problème de tournées de véhicules. La section 1.3 présente les caractéristiques du problème de tournées du personnel de soins à domicile. Dans la section 1.4, l'ensemble des contraintes couvertes par la littérature est détaillé avec les différentes déclinaisons existantes. La section 1.5 met l'accent sur les différentes fonctions économiques prises en compte afin d'évaluer la qualité d'une planification. La section 1.6 traite des méthodes de résolution proposées. Ce chapitre est clôturé par une conclusion.

1.2 Le problème de tournées du personnel de soins à domicile

Le Problème de Tournées de Véhicules (Vehicle Routing Problem – VRP) est un problème largement étudié d'optimisation combinatoire intervenant dans de nombreux domaines d'application tels que la livraison d'essence [Dantzig, 1954], la livraison de repas [Russell, 1995], les tournées de techniciens [Pillac *et al.*, 2012], les tournées de véhicules en prenant en compte à la fois le coût environnemental et opérationnel [Bektas et Laporte, 2011].

Le VRP peut être défini comme suit : soit $G = (S, A)$ un graphe avec V l'ensemble des sommets et A l'ensemble des arêtes. L'ensemble des sommets représente les clients à servir et un dépôt unique à partir duquel chaque véhicule débute et termine sa tournée. L'objectif du problème est de déterminer un ensemble de tournées minimisant la distance totale parcourue tel que chaque client soit servi une seule fois. Depuis les travaux séminaux de [Dantzig, 1954], où le Problème de Tournées de Véhicules à capacité a été introduit, pour lequel la somme des demandes des clients d'une même tournée ne doit pas dépasser une capacité K identique pour tout véhicule, de nombreuses extensions ont été étudiées. Par exemple,le VRP avec fenêtres de temps (Vehicle Routing Problem with Time Windows – VRPTW) [Bräysy et Gendreau, 2005b, Bräysy et Gendreau, 2005a] constitue une extension fortement étudiée. Dans le cas du VRPTW, chaque client i doit être servi dans un intervalle de temps ou fenêtre de temps $[e_i, l_i]$ où e_i et l_i représentent respectivement la date de début au plus tôt et au plus tard du service.

Le Problème de Tournées du personnel de soins à domicile (Home Health Care Routing and Shceduling Problem – HCRSP) est une extension du VRP dans laquelle plusieurs contraintes issues des spécificités des soins à domicile ont été ajoutées. Dans la littérature, ce problème peut être rencontré sous différentes dénominations : Home Health Care Scheduling Problem [Begur *et al.*, 1997], Home Health Care Routing and Scheduling Problem [Mankowska *et al.*, 2014], Home Health Care Problem [Bertels et Fahle, 2006], Home Care Crew Scheduling Problem [Rasmussen *et al.*, 2012], Home Care Worker Scheduling [Akjiratikarl *et al.*, 2007], Home Help Staff Scheduling [Ikegami et Uno, 2007] et Therapist Routing and Scheduling Problem [Bard *et al.*, 2014]. D'après [Trautsamwieser et Hirsch, 2011], les premières études traitant du HCRSP datent de 1997 [Begur *et al.*, 1997] et 1998

[Cheng et Rich, 1998].

Le HCRSP peut être résumé comme suit : un ensemble de *patients* réparti sur une zone géographique nécessite des soins/services à domicile. Ces soins/services sont réalisés par des *soignants*. Le problème consiste par conséquent à déterminer un ensemble de tournées sur un horizon de planification d'un ou plusieurs jours afin de minimiser le coût du service ou maximiser la qualité du service tout en respectant l'ensemble des contraintes auquel le problème est soumis. Ce problème est par conséquent similaire au VRP puisque la principale décision consiste à construire un ensemble de tournées pour servir des patients. Cependant, le HCRSP diffère du VRP car il intègre en plus des caractéristiques particulières formulées à travers des contraintes spécifiques. Ces caractéristiques sont :

— la continuité des soins : elle permet d'assurer à chaque patient d'être soigné par un même soignant ou un nombre restreint de soignants
— la dépendance temporelle entre les services : des délais minimaux et/ou maximaux peuvent séparer les débuts de réalisation de services. Par exemple, un service doit être planifié trois heures après un autre ou deux services doivent être réalisés en même temps. Les services qui sont impliqués dans une dépendance temporelle sont fortement liés puisque le calcul de la date de début d'un service dépendra de celle des autres
— les spécificités et préférences des soignants et des patients : le personnel de soins couvre différents domaines d'expertise ce qui soumet donc les décisions d'affectation à des contraintes de qualification

À cause des deux premières caractéristiques, la décision d'affectation d'un soin/service à un soignant ne peut être faite indépendamment des autres services. La dernière caractéristique augmente la complexité de l'affectation pour le HCRSP. Ces caractéristiques sont décrites plus en détail dans la section 1.3.

Un des premiers aspects qui peut être souligné dans la littérature du HCRSP est la diversité des modèles. Cela peut être expliqué par la variété des caractéristiques intégrées dans la modélisation et des choix de la formulation. Le tableau 1.1 classe les modèles du HCRSP selon l'extension du VRP dont ils se rapprochent. La plupart des principaux modèles du HCRSP de la littérature sont formulés à partir du VRP ou d'une de ses extensions. Parmi ces modèles, nous pouvons donc trouver :

— le VRP [Toth et Vigo, 2001]
— le VRPTW [Bräysy et Gendreau, 2005a, Bräysy et Gendreau, 2005b]
— le Problème de Voyageur de Commerce à dépot multiple avec fenêtres de temps (Multiple Depot Traveling Salesman Problem with Time Windows – MDTSPW) correspond à un VRPTW pour lequel les véhicules peuvent débuter et terminer les tournées [Bektas, 2006] à des dépôts différents
— le Problème de Tournées de Véhicules périodique (Periodic Vehicle Routing Problem – PVRP) consiste à réaliser des tournées sur un horizon de planification de plusieurs jours [Francis *et al.*, 2008] où une fréquence de service doit être respectée
— le Problème de Tournées de Véhicules périodique avec fenêtres de temps (Periodic Vehicle Routing Problem with Time Windows – PVRPTW) [Francis *et al.*, 2008] est un PVRP pour lequel les services doivent être réalisés dans une fenêtre de temps donnée

Extensions	Références
VRP	[Bennett, 2010], [Chahed *et al.*, 2009]
VRPTW	[Bertels et Fahle, 2006], [Bowers *et al.*, 2014], [Bräysy *et al.*, 2009] [Bredström et Rönnqvist, 2008], [Braekers *et al.*, 2016], [Cire et Hooker, 2012] [Di Mascolo *et al.*, 2014], [Eveborn *et al.*, 2006], [**?**], [Jemai *et al.*, 2013] [Mankowska *et al.*, 2014], [Rasmussen *et al.*, 2012], [Redjem, 2013], [Rendl *et al.*, 2012], [Rest et Hirsch, 2013], [Thomsen, 2006], [Liu *et al.*, 2013] [Trautsamwieser *et al.*, 2011], [Trautsamwieser et Hirsch, 2011],
MDTSPTW	[Allaoua *et al.*, 2013], [Akjiratikarl *et al.*, 2007], [Cheng et Rich, 1998], [Hiermann *et al.*, 2013]
PVRP	[Cappanera et Scutellà, 2015], [Ikegami et Uno, 2007]
PVRPTW	[Bard *et al.*, 2014], [Bard *et al.*, 2013], [Akjiratikarl *et al.*, 2007] [Macdonald, 2010], [Maya Duque *et al.*, 2015], [Nickel *et al.*, 2012]
SP	[Bredström et Rönnqvist, 2008], [Eveborn *et al.*, 2006], [Rasmussen *et al.*, 2012]

TABLE 1.1 – Extensions du VRP exploitées pour formuler le HCRSP

D'autres études utilisent des modèles basés sur le Problème de Partitionnement d'Ensemble (Set Partitioning Problem – SP). Dans les lignes qui suivent, les expressions « soin » et « service » sont synonymes. De même que les expressions « soignant » et « personnel de soins ».

1.3 Caractéristiques prises en compte dans les modèles

Les caractéristiques prises en compte dans les modèles reflètent la diversité du fonctionnement du soin à domicile. Les études réalisées dans ce domaine font état d'un large éventail de modèles qui dépendent des caractéristiques qu'elles intègrent à la formulation. La revue de la littérature qui a été conduite nous a amené à classifier ces caractéristiques en trois groupes en fonction de leurs liens avec :

— la structure de soins à domicile
— le personnel de soins
— le patient

Un modèle de HCRSP est par conséquent composé en partie d'une combinaison de caractéristiques prises en compte dans la formulation du problème. Chaque caractéristique est soit une donnée, soit une variable, soit une contrainte à respecter, soit une fonction économique à optimiser. Les caractéristiques principales sont décrites dans les paragraphes qui suivent.

1.3.1 Caractéristiques relevant de la structure de soins à domicile

Les caractéristiques relevant de la structure de soins à domicile qui ont un impact sur le modèle du HCRSP peuvent être décrites comme suit :

- l'**horizon de planification** : cet horizon correspond à une période au cours de laquelle la structure de soins à domicile planifie l'ensemble des services. La taille de cet horizon de planification dépend de l'information disponible à la fois sur le plan qualitatif et quantitatif. En effet, plus l'horizon de planification est grand, moins l'information est fiable.
- la **périodicité des décisions de planification** : elle fait référence au nombre de fois qu'une décision est répétée sur un horizon de planification. L'idée ici est de reproduire les tournées déjà construites un ou plusieurs jours après.
- la **diversité des services** à planifier est une autre caractéristique affectant la formulation du modèle du HCRSP. En effet, les services peuvent être des services médicaux, des services ménagers. Selon le type de service offert et le personnel de soins participant à leur réalisation, les modèles de tournées doivent donc intégrer les spécificités des services et du personnel de soins.
- la **continuité des soins** : la structure de soins à domicile affecte à chaque patient un unique soignant afin de réaliser l'ensemble des services qu'il nécessite ou cherche à minimiser le nombre de soignants différents qu'un même patient est amené à rencontrer. Une telle prestation de services permet d'une part de minimiser la perte d'information liée au patient et d'autre part, d'assurer un meilleur confort pour le patient. Lorsqu'aucune continuité des soins n'est assurée, la structure de soins à domicile n'est pas tenue de prendre de compte les décisions d'affectation entre un patient et un soignant issues des périodes antérieures. Chaque nouvelle période commence par un nouveau processus d'affectation dans lequel tous les soignants peuvent être potentiellement affectés à un patient selon leurs qualifications et les préférences du patient.
- la **gestion de l'équipe de soin** : la structure de soins à domicile peut décider de diviser le personnel de soins en plusieurs équipes en fonction de critères relevant du partitionnement géographique, des qualifications des soignants et des préférences des patients. Une telle approche permet de réduire le temps de trajet des soignants (à l'intérieur d'une même zone géographique) et facilite la coordination d'une équipe. Cependant, ce choix peut entraîner la construction d'une planification sous-optimale.

1.3.2 Caractéristiques relevant du patient

Plusieurs caractériques relevant du patient affectent la formulation des modèles du HCRSP. Ces caractériques changent la manière dont les requêtes du patient sont exprimées dans le modèle du HCRSP. Ces caractériques peuvent être décrites comme suit :

- lorsqu'un patient est admis dans la structure de soins à domicile, le décideur met en place son projet thérapeutique, qui consiste à définir la **nature et la périodicité** des services que nécessitent ce patient. En fonction de l'information disponible, un profil est associé au patient à partir du type des services qu'il nécessite. Le

profil prend également en compte la périodicité des services, qu'elle soit journalière, hebdomadaire ou qu'elle suive un motif particulier.
— les services que nécessite un même patient peuvent dépendre les uns des autres. En effet, certains patients nécessitent deux ou plusieurs services en même temps, entraînant la mise en place de **services synchronisés**. Dans d'autres cas, certains services doivent être réalisés avant un autre créant ainsi une **dépendance temporelle**. Enfin, une **disjonction entre services** peut également intervenir lorsque ces services ne peuvent être réalisés en même temps chez un patient. Ces services sont qualifiés de **services disjonctifs**. Par exemple, une prise de sang et une séance de kinésithérapie ne peuvent pas être effectuées simultanément.
— les patients peuvent avoir des **préférences** en ce qui concerne les services qu'ils nécessitent. Ces préférences peuvent porter au jour de planification, aux horaires pour lesquelles le patient est disponible.

1.3.3 Caractéristiques relevant du personnel de soins

Les services que nécessitent les patients peuvent être réalisés par différents types de soignant tels que les infirmiers, les médecins, les kinésithérapeutes, les psychologues, les assistants sociaux ... Chaque soignant est caractérisé par un certain nombre de spécificités :
— la **capacité** d'un soignant correspond à la durée maximale où il peut être amené à soigner un ou plusieurs patients selon son **contrat de travail**. Le soignant peut être employé à temps complet, à mi-temps par la structure de soins à domicile ou être un intérimaire. Le **temps de travail supplémentaire** correspond aux heures excédant cette capacité. Dans un tel cas, la structure de soins à domicile est amené à payer plus cher pour chaque heure de travail supplémentaire effectuée. La **disponibilité** d'un soignant correspond à la période au cours de laquelle le soignant est disponible pour réaliser des services. Cela exclut les pauses, les vacances, les congés.
— le personnel de soins dispose d'une **qualification** lui permettant de réaliser différents soins. Ces qualifications correspondent aux compétences médicales, à la langue parlée ... Ces qualifications se déclinent en niveau de qualification ou en qualification particulière. Dans le premier cas, un soignant ne peut réaliser un service que si son niveau de qualification est supérieur ou égal au niveau de qualification que ce service nécessite. Dans le deuxième cas, la qualification particulière d'un soignant doit concorder avec celle que le service nécessite.

1.4 Contraintes prises en compte dans les modèles

Dans les modèles du HCRSP existants, les trois groupes de caractéristiques décrits ci-dessus sont pris en compte comme contraintes à respecter ou des fonctions économiques à optimiser. Pour chaque caractéristique, les contraintes sont classées en fonction de la catégorie à laquelle elles appartiennent (voir Tableau 1.2). Trois catégories de contraintes peuvent être identifiées :
— les contraintes temporelles
— les contraintes d'affectation
— les contraintes géographiques

Agents	Contraintes temporelles	Contraintes d'affectation	Contraintes géographiques
Structure de soin	- Horizon de planification - Périodicité	- Continuité des soins	- Zone géographique - Secteurs - Typologie des services
Patient	- Régularité de service - Fenêtre de temps - Dépendance temporelle - Services disjonctifs	- Préferences	- Topologie du réseau
Soignant	- Type de contrat - Capacité	- Qualification - Charge de travail	- Localisation des soignants

TABLE 1.2 – Schéma de classification basé sur les contraintes prises en compte

La première catégorie de contraintes regroupe celles qui vont influencer la prise de décision au niveau temporel (par exemple, l'heure à laquelle un service commencera ou quand le soignant réalisera un soin en fonction du planning établi). La seconde catégorie de contraintes qui regroupe les contraintes d'affectation intervient dans la relation entre le soignant et le patient en excluant l'aspect temporel. Elles influenceront le choix du soignant qui réalisera un service requis par un patient. Enfin, dans la troisième catégorie, les contraintes géographiques traiteront de la typologie de la zone géographique, de la localisation à la fois des patients et des soignants. Dans cette partie, une revue des travaux existants est établi en fonction de la manière dont les caractéristiques du HCRSP ont été modélisées en tant que contraintes. Les tableaux **??**–**??** dans l'appendice donnent pour chaque article les contraintes prises en compte.

1.4.1 Contraintes relevant de la structure de soins à domicile

Contraintes temporelles Dans la plupart des travaux traitant du HCRSP, l'horizon se réduit à un seul jour [Bredström et Rönnqvist, 2008] ou à une semaine [Bard *et al.*, 2014] La périodicité des décisions de planification prise en compte dans la littérature correspond à un ensemble de décisions intervenant avec régularité d'une période à une autre.

Contraintes d'affectation Les contraintes d'affectation d'une structure de soins à domicile sont essentiellement liées à la continuité des soins. La notion de continuité des soins exprimée dans les modèles de HCRSP est proche de la notion de cohérence de service dans le PVRP [Kovacs *et al.*, 2014], également appelé Consistent Vehicle Routing Problem (ConVRP). Dans ce problème, les clients doivent être servis par le même livreur tout au long de l'horizon de planification. Dans la plupart des cas, il est difficile de s'assurer que tous les soins d'un même patient soient prodigués par un même soignant tout en respectant à la fois les contraintes temporelles et d'affectation. Ainsi, cette contrainte est relaxée comme dans [Macdonald, 2010] où elle devient une partie de la fonction économique. Cependant, selon [Freeman, 2010], assurer une totale continuité des soins permet d'établir une relation de confiance entre le soignant et le patient. Une telle relation est très importante lors d'un suivi d'une maternité [Bowers *et al.*, 2014], où les soins prénataux et postnataux sont réalisés par une même sage-femme. Dans [Maya Duque *et al.*, 2015], une continuité

des soins à la fois totale et partielle peut être rencontrée. En effet, le même soignant sera affecté à un patient nécessitant un ou deux soins par semaine. Si plus de deux soins par semaine sont nécessaires, au moins deux soignants seront affectés au patient. Ce choix a pour objectif d'atténuer les effets de l'absentéisme.

Contraintes géographiques Les contraintes géographiques prises en compte dans les modèles du HCRSP sont issues de l'existence de secteurs et du fait que les soignants et les patients soient trop dispersés sur une zone géographique donnée. L'existence d'un unique secteur est l'approche la plus communément utilisée [Mankowska *et al.*, 2014], où les soignants ne sont pas séparés en plusieurs groupes. Dans le cas de plusieurs secteurs, un soignant est affecté à un unique secteur et sera amené à servir uniquement les patients de ce secteur [Eveborn *et al.*, 2006]. Chaque secteur peut être géré indépendamment comme un centre de décision autonome [Maya Duque *et al.*, 2015] ou de façon intégrée [Lanzarone *et al.*, 2012]. Cette contrainte peut être modélisée comme une contrainte de qualification.

Différents types de service ont été modélisés dans la littérature. En plus des services usuels qui ont trait aux services et soins à la personne, d'autres types de services ont été pris en compte [Bräysy *et al.*, 2009, Liu *et al.*, 2013, Liu *et al.*, 2014], où les auteurs prennent en compte la collection d'échantillons biologiques (sang, urine) et/ou de la livraison de médicaments ou de matériels. En terme de modélisation, ces problèmes sont similaires au Problème de Tournées de Véhicules avec collecte et livraison (Vehicle Routing Problem with Pickup and Delivery) [Toth et Vigo, 2001, Chapter 9]. La structure de soins à domicile est dans ce cas un système de soin décentralisé dans lequel la localisation des différentes entités participant à la réalisation du processus de livraison des soins a un impact important sur les décisions de planification. La localisation de la structure de soins à domicile, des laboratoires médicaux et des pharmacies est donc un paramètre à prendre en compte dans les modèles du HCRSP. Les tournées doivent être construites en tenant compte de ces localisations. Par exemple, [Kergosien *et al.*, 2014] propose un modèle pour planifier la collecte de sang à domicile. Dans ce contexte, certains arrêts au laboratoire doivent être impérativement planifiés de sorte à prendre en compte le délai d'expiration des échantillons de sang. Lorsque des échantillons biologiques doivent être analysés, une contrainte limitant la durée maximale de transport est à considérer. Cette durée maximale est très importante dans le cas de livraison de chimiothérapies du fait de l'aspect périssable des échantillons [Chahed *et al.*, 2009]. Un sommet fictif est introduit afin de modéliser la localisation d'un laboratoire. Ce type de contrainte rend difficile la construction de tournées réalisables à cause du calcul des dates de livraison au domicile du patient qui dépendent de la date de départ du soignant du laboratoire avec l'échantillon. Dans le cas d'une collecte d'échantillon biologique, la date de retour des échantillons dépendra de l'heure de la collecte. La contrainte mettant en oeuvre un temps de transport maximal peut être également rencontré dans la littérature du VRP avec la livraison de repas périssable [Amorim et Almada-Lobo, 2014]. [Bräysy *et al.*, 2009] traite de la livraison de repas à domicile consistant à livrer des plateaux repas à partir d'une centrale de cuisine et à collecter ceux qui sont réutilisables. Dans [Liu *et al.*, 2013], les auteurs étudient un Problème de Tournées de Véhicules avec collecte et livraison dans le cadre du transport de médicaments et de matériels médicaux entre la structure de soins à domicile et le domicile des patients, du prélèvement d'échantillons de sang du domicile du patient au laboratoire, et de la livraison de médicaments

spécifiques de l'hôpital au domicile du patient.

1.4.2 Contraintes relevant du patient

Contraintes temporelles Dans certains travaux, afin de prendre en compte la périodicité, les patients sont soignés une fois par jour [Hiermann *et al.*, 2013] ou une fois par semaine [Bard *et al.*, 2014]. D'autres travaux tiennent compte du fait qu'un patient puisse être soigné plusieurs fois par jour [Rasmussen *et al.*, 2012] ou plusieurs fois par semaine [Nickel *et al.*, 2012].

L'une des plus importantes contraintes temporelles relevant du patient est l'existence des fenêtres de temps. Une fenêtre de temps, c'est-à-dire un intervalle de temps au cours duquel le patient est disponible à son domicile pour recevoir des soins, est associée à chaque patient. Deux types de fenêtres de temps sont modélisés : dure [Akjiratikarl *et al.*, 2007, Bard *et al.*, 2014] or souple [Mankowska *et al.*, 2014, Trautsamwieser et Hirsch, 2011, Hiermann *et al.*, 2013, Nickel *et al.*, 2012]. Dans le premier cas, la date de début d'un service doit forcément appartenir à la fenêtre de temps. Dans le second cas, la date de début d'un service peut être déterminée avec un retard au prix d'une pénalité dans la fonction économique. Dans les travaux de [Bertels et Fahle, 2006], deux types de fenêtre de temps sont associés à chaque service : une fenêtre de temps souple incluse dans une fenêtre de temps dure. D'autres contraintes liées à la date de début des services existent dans la littérature. [Bertels et Fahle, 2006] impose que la date de début des services appartienne à un ensemble de date de début prédéfinies. Cet ensemble est composé de dates cycliques, par exemple tous les quarts d'heure. Enfin, uniquement [Braekers *et al.*, 2016] propose à la fois le respect de fenêtres de temps et une date de début préférée.

De nombreux travaux traitant du HCRSP admettent une régularité de service imposant la réalisation d'un soin que nécessite un patient, le même jour, à la même heure, d'une semaine à l'autre (par exemple, tous les jeudis à 14h). Ce type de contrainte est pris en compte par [Maya Duque *et al.*, 2015, Bennett et Erera, 2011, Bennett, 2010]. Dans les deux derniers travaux, un horizon de planification roulant est mis en place afin de résoudre un Problème de Tournées de Véhicules dynamique dans lequel une fréquence hebdomadaire de visites sur plusieurs semaines est respectée. À l'échelle d'une semaine, la réalisation des services intervient les mêmes jours durant les mêmes horaires en respectant un motif reproduit de semaine en semaine. Dans les travaux de [Maya Duque *et al.*, 2015], un nombre fixe de services est affecté à chaque patient chaque semaine en évitant de soigner un patient deux jours consécutifs (sauf s'il nécessite plus de quatre services). La régularité de service peut être également trouvée dans d'autres problèmes de tournées où des services réguliers doivent être réalisés comme la collection de déchets [Baptista *et al.*, 2002].

Dans un horizon de planification de plusieurs jours, les services que nécessitent un patient seront réalisés uniquement dans un sous-ensemble de jours. Cet ensemble de jours représente les disponibilités des patients, leurs préférences ou encore des décisions d'affectation journalière prises en amont. Un service donné sera donc planifié uniquement pour un jour appartenant à cet ensemble. Étant donné que les soignants peuvent avoir des jours d'indisponibilité, le décideur doit faire correspondre à la fois les disponibilités des patients et des soignants. Cette contrainte est rencontrée dans quelques études [Bard *et al.*, 2014, Bard *et al.*, 2013].

Dans certains cas, on exige la planification d'un nombre minimum de services et/ou la séparation des services d'une ou plusieurs périodes. Par exemple, un patient sera soigné au moins deux fois par semaine avec un intervalle minimal de deux jours entre les soins [Begur *et al.*, 1997]. Cette situation nécessite de définir un motif d'affectation à chaque patient [Bard *et al.*, 2014, Cappanera et Scutellà, 2015, Liu *et al.*, 2014]. Un motif d'affectation correspond à un sous-ensemble de jours sur un horizon (par exemple, une semaine) dans lequel les services doivent être réalisés. Ce nombre limité de combinaisons permet de planifier des services réalisables à la fois pour le soignant et pour le patient. Par exemple, pour un horizon de cinq jours ouvrables, un patient nécessitant deux services séparés de deux jours peut être soigné mardi et vendredi ou lundi et jeudi.

Des services associés à un soignant donné peuvent être reliés par une dépendance temporelle. Ces dépendances lient des services appartenant à la fois à des tournées distinctes ou des services appartenant à la même tournée. Dans le premier cas (dépendance temporelle inter-tournée), la réalisation de ces services nécessite la synchronisation de deux tournées. Par exemple, un patient lourdement handicapé nécessitera deux soignants pour le déplacer et entraîne donc la coordination de leurs tournées. Ces services synchronisés sont pris en compte notamment par [Eveborn *et al.*, 2006, Thomsen, 2006]. Comme l'a montré [Thomsen, 2006], il est difficile de prendre en compte à la fois des services synchronisés et une capacité de travail des soignants bornée. Contrairement, au VRP ou au VRPTW classique, il n'est plus possible dans ce cas d'évaluer indépendamment les tournées. Déterminer dans ce cas l'heure de début d'un service synchronisé, et donc l'heure de début de l'ensemble des services de la tournée à laquelle il appartient, doit être effectué en tenant compte des autres services auxquels il est lié. La Figure 1.1 illustre un exemple de quatre tournées liées par trois situations de synchronisation. Comme on peut le constater, les tournées R1 et R4 sont liées par une synchronisation (A). Les dates de début des deux services impliqués dans cette synchronisation doivent être identiques. De même, pour les services impliqués dans les synchronisations (B) et (C). En outre, les dates de début des services qui suivent les services impliqués dans ces synchronisations dépendent aussi des dates de début de ces synchronisations. La contrainte traitant de services synchronisés peut être vue comme un cas spécial de dépendance temporelle pour lequel la durée séparant deux services n'est pas nécessairement égale à zéro. La durée séparant deux services synchronisés peut être fixée ou bornée par une valeur maximale et/ou minimale [Mankowska *et al.*, 2014]. Ainsi, une synchronisation où les deux services commencent en même temps est modélisée par un délai nul ou éventuellement un délai maximal faible. Par exemple, la prise de médicaments d'un patient lourdement handicapé doit précéder de plusieurs heures son déjeuner. Les modèles prenant en compte à la fois une durée minimale et maximale formalisent également la contrainte de précédence entre les services [Bredström et Rönnqvist, 2008]. Cette dernière force la réalisation d'un service à précéder de plusieurs heures celle d'un autre. Une revue de la littérature traitant des différentes formes de synchronisation et proposant une classification a été réalisée par [Drexl, 2012]. Dans le second cas (dépendance temporelle intra-tournée), un service est réalisé avant un autre dans le cas de relation de précédence entre eux. En plus de la dépendance temporelle, [Kergosien *et al.*, 2009] introduit les services disjonctifs. Des services disjonctifs ne peuvent être réalisés en même temps, leur chevauchement est interdit. Cette situation met en exergue des incompatibilités entre services d'un même patient. Par exemple, une séance de kinésithérapie ne peut

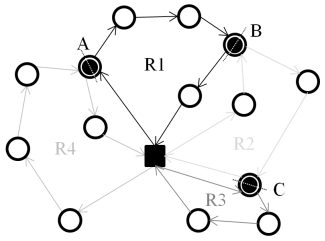


FIGURE 1.1 – Exemple de 3 situations (A,B et C) de synchronisation impliquant 4 tournées

être réalisée en même temps qu'une prise de sang. Cette contrainte ajoute un niveau de décision supplémentaire lors de la planification des services. Les services synchronisés, la précédence et les services disjonctifs ont été étudiés par [Rasmussen *et al.*, 2012]. L'auteur présente cinq types de dépendance temporelle : la synchronisation, le chevauchement, le délai minimum, le délai maximum, et le délai minimum et maximum à la fois.

Contraintes d'affectation Les contraintes d'affectation formalisent des préférences des patients en terme de soignant [Trautsamwieser *et al.*, 2011] ou des incompatibilités de genre [Rendl, 2011]. En effet, ces préférences interviennent lorsqu'une certaine affinité s'est installée entre un patient et un soignant. Un patient préférera un soignant spécifique qui lui a déjà prodigué des soins. Par contre, un patient peut également formuler le désir de ne pas être soigné par un soignant donné. Ce désir peut être motivé par le fait que le soignant soit de sexe différent.

Contraintes géographiques La typologie du réseau est une caractéristique importante du HCRSP. Selon le contexte (rural ou urbain) et en fonction de la dispersion des patients, le temps de trajet sera plus ou moins important. [Rest *et al.*, 2012] souligne les différences entre les services en zone rurale et en zone urbaine. En effet, le ratio entre le temps imparti aux soins et celui imparti au trajet sera différent dans ces zones. L'estimation des temps de trajet, qui sont une donnée essentielle dans tout problème de transport, reste une tâche difficile. Bien que la plupart des travaux existants utilise la distance euclidienne, les temps de trajet pris en compte dans certains travaux sont déterminés à partir d'un système d'information géographique [Begur *et al.*, 1997, Weigel et Cao, 1999]. Quelques travaux [Rendl *et al.*, 2012, Hiermann *et al.*, 2013] ont pris en compte un réseau de transport multimodal qui semble être plus proche de la réalité en milieu urbain. Dans ces travaux, trois modes de transport sont pris en compte : la voiture, la marche à pied et les transports en commun. En fonction du mode de transport utilisé, le temps de trajet séparant deux points du réseau est différent. Le décideur doit donc déterminer le mode de transport adéquat pour une planification de bonne facture. En outre, les temps de trajet séparant deux points du réseau varient selon l'heure de la journée à cause des aléas. Dans ce cas, [Rest et Hirsch, 2015] propose un modèle avec des temps de trajet dépendant de l'heure de la journée car ces durées peuvent changer considérablement selon l'heure de la journée en raison d'embouteillages, d'accidents de la circulation ou d'aléas.

1.4.3 Contraintes relevant du personnel de soins

Contraintes temporelles La plupart des travaux prennent en compte des horaires de travail pour le personnel de soins au cours desquelles ils seront amenés à prodiguer des soins aux patients. En fonction de l'étude, ces horaires, représentés par une fenêtre de temps, peuvent être fixes [Hiermann *et al.*, 2013, Braekers *et al.*, 2016] ou souples [Trautsamwieser *et al.*, 2011]. Dans le premier cas, une planification nécessitant des heures supplémentaires sera irréalisable tandis que dans le second cas, ces heures supplémentaires seront pénalisées dans la fonction économique. Cependant, il n'est pas possible pour un soignant de commencer à travailler avant le début de sa fenêtre de travail. [Bertels et Fahle, 2006] prend en compte à la fois des fenêtres de temps dures et souples. Les secondes incluses dans les premières limitent le nombre d'heures supplémentaires possibles. Avec un horizon de planification de plusieurs jours, ces horaires de travail peuvent dépendre du jour [Bard *et al.*, 2014]. Cela est bien entendu plus proche de la réalité si la structure de soins à domicile fait face à une baisse d'activité, évitant ainsi de faire travailler des soignants malgré la charge de travail faible. Dans certains cas, l'heure de début du travail est inconnue et par conséquent elle doit être déterminée afin de calculer le temps de travail journalier. Les réglementations du temps de travail journalier et hebdomadaire sont liées aux dispositions légales du pays dans lequel la structure de soins à domicile opère. Par exemple, le temps de travail journalier est de 7 heures 30 minutes au Royaume Uni [Akjiratikarl *et al.*, 2007] et de 8 heures en Finlande [Bräysy et Gendreau, 2005b]. Lorsque le temps de travail journalier est dépassé, chaque nouvelle heure effectuée par le soignant représente une heure supplémentaire [Nickel *et al.*, 2012]. Chaque heure supplémentaire a un coût plus élevé qu'une heure effectuée durant le temps de travail légal. Cette limite peut être étendue à l'échelle de la semaine [Maya Duque *et al.*, 2015], auquel cas, le soignant est autorisé à travailler un nombre maximum d'heures par semaine selon son contrat de travail. Dans le cas d'horaires de travail, l'heure de début des services peut être déterminée en estimant que le soignant quitte son dépôt à l'heure d'ouverture de l'horaire de travail. Mais en tenant en plus compte d'un temps de travail réglementaire, l'heure de départ du soignant de son dépôt dépendra des fenêtres de temps des patients qui lui ont été assignés. Il serait préférable de retarder le plus possible l'heure de départ en utilisant une approche prenant en compte le temps restant possible (Forward Time Slack en anglais) [Savelsbergh, 1992].

Les dispositions légales forcent le décideur à accorder au personnel de soins une pause. Deux approches coexistent dans la littérature : la pause doit être prise dans une fenêtre de temps [Trautsamwieser et Hirsch, 2011] ; la pause doit être prise lorsque la durée de la tournée a atteint une certaine limite [Thomsen, 2006]. Dans ce cas, la tournée d'un soignant oblige le décideur à prendre en compte le lieu et/ou la durée de la pause, s'ils sont connus, impliquant la création de nouvelles variables de décision. Certains travaux prennent en compte la pause en introduisant des patients fictifs ou un temps mort entre deux services [Bard *et al.*, 2013], simulant ainsi une pause. Certains travaux prennent en compte plusieurs contrats de travail. Par exemple, certains modèles distinguent les soignants à temps plein et à temps partiel [Begur *et al.*, 1997]. Ces deux soignants ne sont pas soumis aux mêmes contraintes et le coût associé est différent.

Enfin, pour un horizon de planification à court ou moyen terme (la semaine ou le mois), le décideur doit prendre en compte les indisponibilités en jour des soignants [Bard *et al.*,

2013, Nickel *et al.*, 2012]. Ces indisponibilités sont liées aux congés, vacances, et préférences concernant les jours de travail.

Contraintes d'affectation Les contraintes de qualification sont les premiers types de contrainte prises en compte dans la littérature. Les soignants doivent disposer de qualifications suffisantes pour prodiguer certains services [Mankowska *et al.*, 2014, Braekers *et al.*, 2016]. Les qualifications sont essentiellement liées à la fonction (médecin, infirmier, kinésithérapeute) qu'occupe le soignant, mais peut être aussi liées au genre ou à la langue parlée [Eveborn *et al.*, 2009]. Ce type de contrainte est généralement modélisé de trois manières. Dans la première, une qualification unique est assignée à chaque soignant et chaque service nécessite une seule qualification spécifique. L'idée est donc de déterminer des couples compatibles afin de respecter toutes les affectations. S'il n'existe aucune relation entre les services, le problème de tournées est composé de plusieurs problèmes de tournées indépendants, un pour chaque qualification. La deuxième consiste à affecter plusieurs qualifications à chaque soignant [Bertels et Fahle, 2006] ; c'est une généralisation de la première. La dernière, qui peut être également prise en compte par la deuxième, est basée sur le niveau de qualification de chaque soignant comme c'est le cas dans [Cire et Hooker, 2012]. Ces derniers associent à chaque soignant un niveau de qualification. Ainsi, quelque soit le service, un niveau de qualification minimum est exigé à tout soignant appelé à réaliser le service en question. Une description générale du VRP avec contraintes de qualification a été proposée par [Cappanera *et al.*, 2011].

Certains travaux tiennent compte des préférences du personnel de soins car elles peuvent affecter le processus d'affectation. Par exemple, un soignant peut rejeter un patient pour des raisons personnelles [Trautsamwieser et Hirsch, 2011] ou pour des questions d'allergies (présence d'un chien) [Rendl *et al.*, 2012, Rest et Hirsch, 2013, Hiermann *et al.*, 2013]. En termes de modélisation, ces contraintes sont très proches des contraintes de qualification décrites ci-dessus.

Un autre type de contraintes d'affectation existant dans la littérature traite de l'équilibre de la charge de travail. Il s'agit dans ce cas de maintenir une certaine équité au sein du personnel de soins [Ikegami et Uno, 2007]. Cette contrainte est généralement une fonction économique et donc une contrainte souple. L'évaluation de la charge de travail d'un soignant peut être calculée grâce à deux indicateurs : le nombre de patients soignés ou le nombre de d'heures de travail effectuées.

Contraintes géographiques La plupart des études considère une seule structure de soins à domicile à partir de laquelle les soignants partent et reviennent, également appelée dépôt dans le VRP. Le cas de plusieurs dépôts [Akjiratikarl *et al.*, 2007] reste peu étudié dans la littérature. Si la zone géographique est immense et difficilement accessible du fait de la topologie du réseau (zone urbaine ou rurale), plusieurs structures de soins à domicile peuvent exister. Le personnel de soins peut également partir et revenir de leur domicile ce qui conduit à modéliser un VRP à multiples dépôts. [Trautsamwieser *et al.*, 2011] présente trois types de soignants selon le lieu de début et de fin de leur tournée : les soignants qui partent et reviennent à l'hôpital, les soignants qui partent et reviennent à leur domicile et les soignants qui partent de leur domicile et y reviennent à leur domicile mais dont

le trajet aller-retour entre le domicile du soignant et celui d'un patient n'est pas pris en compte dans la tournée. Quelques travaux [Bard *et al.*, 2014, Akjiratikarl *et al.*, 2007] prennent en compte l'ensemble de ces cas en dissociant les lieux de départ et de retour de chaque soignant. Ainsi, ces travaux couvrent à la fois les aspects mono et multi dépôt.

1.5 Fonctions économiques à optimiser

De nombreuses fonctions économiques ont été proposées dans la littérature afin d'évaluer la qualité d'une solution au problème de planification. Ces fonctions économiques sont soit mono–objectif soit multi–objectif. Dans le cas multi–objectif, le nombre de critères varie entre deux [Akjiratikarl *et al.*, 2007] et sept [Trautsamwieser et Hirsch, 2011] et même treize [Hiermann *et al.*, 2013]. Dans la plupart des travaux, une somme pondérée est mise en place pour traiter l'aspect multi–objectif même si les différents critères n'ont pas la même unité [Bertels et Fahle, 2006, Steeg et Schröder, 2008]. L'utilisation d'une somme pondérée ne permet malheureusement pas d'obtenir les solutions non-supportées. [Braekers *et al.*, 2016] propose d'énumérer l'ensemble du front de Pareto en utilisant l'approche ϵ–contrainte tandis que [Maya Duque *et al.*, 2015] utilise une approche lexicographique.

Dans cette section, les critères les plus fréquemment utilisés pour évaluer une solution sont présentés. Quelques uns sont issus de la relaxation de contraintes difficiles à satisfaire. Par exemple, bien que la continuité des soins et l'équilibre de la charge de travail soient des contraintes du problème dans de nombreux travaux (voir section 1.3), d'autres les considèrent comme des fonctions économiques à optimiser.

Les critères à optimiser peuvent être classés dans les catégories suivantes :
— minimiser le coût total des tournées
— minimiser le nombre de services non réalisés
— minimiser le nombre de soignants
— maximiser la satisfaction

Plusieurs formulations pour chacune des catégories ci-dessus ont été proposées

1.5.1 Minimisation du coût total des tournées

Le coût total des tournées est un critère standard du VRP. Le coût couvre en effet de nombreuses données : le temps total de trajet, la distance totale parcourue ou encore le temps total de travail. Ces différents coûts sont fortement corrélés. Dans la littérature du HCRSP, cette fonction économique est prise en compte par la majeure partie des travaux, approximativement 90% et correspond à la somme des distances parcourues par les soignants de la structure de soins à domicile. Lorsque ce critère est une composante d'une fonction économique, à travers une somme pondérée, il est en général pris en compte avec un poids plus faible comparé au coût de sous-traitance ou de retard [Di Gaspero et Urli, 2014, Braekers *et al.*, 2016, Nickel *et al.*, 2012].

1.5.2 Minimisation du nombre de services non-réalisés

Une structure de soins à domicile peut faire face à un manque de soignants pour réaliser l'ensemble des services exigés. Les services ne pouvant pas être réalisés sont sous-traités [Akjiratikarl *et al.*, 2007, Hiermann *et al.*, 2013, Nickel *et al.*, 2012] à un coût plus élevé. Le coût de sous-traitance étant plus élevé, il s'agira donc pour le décideur de minimiser le nombre de services sous-traités. D'autre part, au lieu de minimiser le nombre de services sous-traités, le décideur peut maximiser le nombre de services réalisés ou le nombre de patients soignés [Bennett, 2010, Bennett et Erera, 2011]. Dans les deux cas, il faut uniquement déterminer un sous-ensemble de patients à soigner par la structure de soins à domicile. Ce type de problématique nous renvoie à la littérature du VRP et à des problèmes bien connus dont : selective-TSP, maximum collection problem et orienteering problem [Gunawan *et al.*, 2016].

1.5.3 Minimisation du nombre de soignants

Certaines études [Allaoua *et al.*, 2013, Ikegami et Uno, 2007] considèrent le nombre de soignants disponibles pour prodiguer des soins comme une variable de décision. Ainsi, la fonction économique à optimiser consiste à minimiser le nombre de soignants qui serviront tous les patients. Bien que cette approche permet de réduire le coût du personnel de soins, la structure de soins à domicile doit être suffisamment flexible pour ajuster le nombre de soignants à chaque jour.

1.5.4 Maximisation de la satisfaction

La satisfaction des patients doit clairement être distincte de celle du personnel de soins. Du point de vue des patients, la satisfaction renvoie au respect de leurs préférences horaires (fenêtre de temps, jours de traitement) ou de soignants. Étant donné qu'il n'est pas toujours aisé de respecter ces préférences, les contraintes sont habituellement relâchées et leur non-respect est pénalisé dans la fonction économique. Ces pénalités peuvent être considérées comme une mesure du désagrément des patients. Par exemple, le non-respect des fenêtres de temps est pénalisé dans la fonction économique [Mankowska *et al.*, 2014]. Une alternative proposée par [Braekers *et al.*, 2016] consiste à minimiser l'écart séparant l'heure de début de service désirée à l'heure de début de service effective.

Du point de vue du personnel de soins, l'équilibre de la charge de travail constitue l'un des objectifs à optimiser. L'idée consiste à minimiser l'écart de charge de travail séparant deux soignants. Afin de prendre en compte ce critère, deux approches peuvent être mises en place : répartir de façon équitable le nombre de services ou équilibrer le temps de travail au sein du personnel de soins. Un autre critère relevant de la satisfaction des préférences des soignants consiste à minimiser le nombre d'heures supplémentaires. Par exemple, [Bard *et al.*, 2013] tient compte du nombre d'heures supplémentaires hebdomadaires réalisées tandis que [Trautsamwieser *et al.*, 2011] ne prend en compte que le nombre d'heures supplémentaires journalières.

Enfin, la satisfaction à la fois du patient et du soignant a été pris en compte par [Maya Duque *et al.*, 2015] en utilisant des facteurs de préférence. Les patients et les soignants ont

une fenêtre de temps préférentielle. Une planification est évaluée au regard de ces préférences tout en minimisant la distance totale parcourue. Puisque le niveau de service reste le critère principal, les auteurs proposent une approche hiérarchique : d'abord maximiser le premier critère à savoir le niveau de service sans tenir du second et ensuite minimiser la distance totale parcourue.

1.6 Méthodes de résolution

Dans cette section, les méthodes de résolution proposées pour résoudre le HCRSP seront présentées. Ces méthodes de résolution seront d'abord regroupées en fonction de l'horizon de planification considéré : un horizon de planification d'un seul jour et un horizon de planification de plusieurs jours. Pour chaque cas, les méthodes de résolution exactes et approchées seront décrites. Les méthodes de résolution approchées seront enfin divisées selon leurs caractéristiques principales : les méthodes basées sur une solution unique ; les méthodes basées sur une population de solutions ; les méthodes hybrides.

À cause de l'absence d'instances de référence et de la diversité des structures de soins à domicile, la plupart des méthodes de résolution proposées n'ont malheureusement pas été comparées les unes aux autres. Il n'est par conséquent pas toujours possible d'estimer la performance d'une méthode de résolution par rapport à une autre. Cependant, ces méthodes de résolution peuvent être analysées selon les stratégies gagnantes de [Vidal *et al.*, 2013] qu'elles mettent en œuvre. Le tableau 1.3 répertorie les stratégies gagnantes mises en œuvre dans ces méthodes. Il permet d'identifier les spécificités de chaque méthode et catalogue les techniques de recherche opérationnelle utilisées par les méthodes.

Les deux premières colonnes font référence à l'espace de recherche (Search Space – SP). La colonne RELAXATION indique l'utilisation de l'oscillation stratégique, c'est-à-dire l'exploitation tout au long du processus de recherche de solutions irréalisables. L'emploi d'une représentation indirecte des solutions (encodage indirect) est spécifié dans la colonne IND.REPRESENT.

Les cinq colonnes suivantes étiquetées NEIGHBOUR. traitent du type de voisinage employé. La première, MULT. NEIGHB., correspond à l'utilisation combinée de plusieurs voisinages. La deuxième, ENUMERATIVE, indique l'exploration des solutions voisines en temps polynomial. Ces deux caractéristiques sont prises en compte dans la plupart des travaux contrairement à l'utilisation d'une procédure d'élagage (PRUNING), d'un voisinage large (LARGE) et de recombinaison de solutions (RECOMBIN.).

Les quatres colonnes qui suivent ont trait à la trajectoire de recherche (TRAJEC.), c'est-à-dire le chemin exploré lorsque l'on sonde le voisinage. La trajectoire peut être aléatoire (RANDOMNESS), continue (CONTINUOUS – deux solutions successives partageant de nombreux éléments), discontinue (DISCONTINOUS – deux solutions successives partageant peu ou pas d'éléments) ou mixte (MIXED – combinant à la fois les aspects continus et discontinus). La plupart des travaux prennent en compte une trajectoire continue.

Les colonnes POPULATION, DIV. MANAG., PARAM. ADAPT. et GUIDANCE, regroupées sous l'étiquette MEM. & CONTR. correspondent à la manière dont la méthode va apprendre des événements passés au cours de la résolution en utilisant respectivement : une

	SP.		NEIGHBOUR.					TRAJEC.				MEM. & CONTR.							
	RELAXATION	IND.REPRESENT.	MULT. NEIGHB.	ENUMERATIVE	PRUNING	LARGE	RECOMBIN.	RANDOMNESS	CONTINUOUS	DISCONTINOUS	MIXED	POPULATION	DIV. MANAG.	PARAM. ADAPT.	GUIDANCE	HYBRIDIZATION	MATHEURISTICS	PARALLELISM	DECOMPOSITION
[Akjiratikarl *et al.*, 2007]	✓	✓	✓	✓				✓	✓			✓			✓				
[Allaoua *et al.*, 2013]																✓	✓		✓
[Bard *et al.*, 2014]			✓	✓				✓	✓					✓					✓
[Begur *et al.*, 1997]																			
[Bertels et Fahle, 2006]			✓		✓			✓				✓	✓		✓		✓		
[Bowers *et al.*, 2014]																		✓	✓
[Braekers *et al.*, 2016]			✓			✓		✓			✓		✓						
[Bräysy *et al.*, 2009]			✓				✓		✓										
[Bredström et Rönnqvist, 2008]								✓								✓	✓		✓
[Maya Duque *et al.*, 2015]			✓	✓				✓	✓							✓	✓		✓
[Eveborn *et al.*, 2006]			✓	✓				✓	✓								✓		
[Hiermann *et al.*, 2013]		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				✓	✓		
[Issaoui *et al.*, 2015]			✓																✓
[Liu *et al.*, 2013]	✓	✓	✓	✓				✓	✓			✓							
[Liu *et al.*, 2014]	✓		✓	✓				✓	✓					✓	✓	✓			
[Mankowska *et al.*, 2014]		✓	✓	✓					✓										
[Nickel *et al.*, 2012]			✓			✓		✓	✓					✓	✓	✓	✓		✓
[Rendl *et al.*, 2012]		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				✓	✓		
[Rest et Hirsch, 2013]	✓	✓							✓				✓	✓	✓				
[Trautsamwieser *et al.*, 2011]			✓	✓	✓					✓									
[Trautsamwieser et Hirsch, 2011]	✓		✓					✓	✓					✓					

Table 1.3 – Métaheuristiques pour le HCRSP et stratégies mises en oeuvre

population de solutions, des stratégies de diversification pour explorer de nouveaux espaces de recherche, la mise à jour des paramètres et des mécanismes d'orientation. Bien que ces approches n'ont pas été intégrées dans la plupart des méthodes de résolution et qu'elles soient difficiles à mettre en œuvre, elles participent de façon conséquente à l'amélioration de la performance des méthodes de résolution.

Les dernières colonnes indiquent l'intégration d'une technique d'hybridation (HYBRIDIZATION), la résolution de sous-problèmes de façon exacte grâce à la programmation mathématique (MATHEURISTICS), la mise en place d'une parallélisation mettant en œuvre des techniques de coopération ou (PARALLELISM), et la décomposition du problème en sous-problèmes résolus séparément tout en coopérant les uns avec les autres (DECOMPOSITION). Concernant ces dernières techniques, aucune recherche, à notre connaissance, n'a tenu compte des aspects ayant trait à la parallélisation.

1.6.1 Méthodes traitant un horizon de planification à court terme

Dans cette section, seules les méthodes de résolution opérant sur un horizon de planification d'un jour seront présentées.

1.6.1.1 Méthodes exactes

[Rasmussen *et al.*, 2012] propose une méthode de type branch-and-price utilisant la génération de colonnes. Le problème maître a pour objectif de déterminer une planification à coût minimum pour chaque soignant parmi un ensemble de planifications possibles. Cet ensemble de planifications est généré grâce à un problème esclave qui est en réalité un problème de plus court chemin avec fenêtres de temps. Le problème esclave est composé d'autant de problèmes indépendants qu'il y a de soignants. Cette approche est testée sur des instances réelles (7-15 soignants, 60-150 services), des instances générées aléatoirement (7-15 soignants, 150 services).

1.6.1.2 Méthodes approchées

Méthodes à solution unique Une heuristique à deux phases a été implémentée par [Cheng et Rich, 1998]. Dans la première phase, une procédure basée sur un algorithme glouton randomisé permet de construire en parallèle les tournées. Pour ce faire, un patient est sélectionné à chaque fois pour être inséré dans la tournée d'un soignant. Deux possibilités sont mises en œuvre : l'insertion du patient en fin de tournée ; la prise d'une pause puis l'insertion du patient en fin de tournée. À la suite de cette insertion, le coût total de la solution est mis à jour tout en vérifiant sa réalisabilité. Dans la seconde phase de l'heuristique, la solution obtenue dans la phase précédente est améliorée. Pour ce faire, deux tournées sont choisies l'une de coût faible et l'autre de coût élevé. Le coût élevé d'une tournée est dû essentiellement au nombre d'heures supplémentaires entamées. Il s'agit donc de réduire ces heures supplémentaires en réaffectant des patients d'une tournée à l'autre. L'ordre de quelques patients sont fixés et ces tournées sont réoptimisées. Enfin, la méthode tente d'insérer les patients qui n'ont pas encore été planifiés. Afin d'évaluer la méthode, deux groupes d'instances ont été utilisées. Le premier ensemble est constitué de 40 instances

générées aléatoirement de 4 soignants et 10 patients. L'heuristique résout en moins d'une seconde ces instances et retrouve la solution optimale pour 22 d'entre elles ; le deuxième, plus restreint, est composé de 3 instances réelles de 294 soignants et 900 patients. L'heuristique est capable de fournir des solutions proches de celles de la structure de soins à domicile.

[Eveborn *et al.*, 2006] décrit un système d'aide à la décision, Laps Care, pour la planification des soins dans une structure de soins à domicile en Suède. Ce système se base sur un modèle de partitionnement d'ensemble séparant les patients en sous-groupe et les affectant aux tournées des soignants. Le HCRSP sous-jacent est résolu à partir d'un algorithme de concordance itéré (repeated matching algorithm) partant d'une concordance initiale et créant de nouvelles concordances grâce à des améliorations locales. Cette approche permet de favoriser certaines spécificités d'une solution telles que le temps de trajet et les préférences du patient. Des expérimentations numériques ont été conduites sur des instances réelles de 12-20 soignants et 86-123 services démontrant des gains de 7% sur le temps de travail total et 20% sur le temps total de trajet par rapport à des solutions obtenues manuellement. Ces résultats sont obtenus avec des temps de calcul n'excédant pas 3 minutes.

Une recherche adaptative à voisinage variable (Adaptive Variable Neighborhood Search, VNS) a été proposée par [Mankowska *et al.*, 2014]. Une solution est représentée grâce à une matrice encodant l'affectation entre un patient et un soignant, la synchronisation et le séquencement. La solution initiale est déterminée à partir d'une heuristique gloutonne qui favorise d'abord les patients urgents en les affectant aux soignants qualifiés pouvant intervenir au plus tôt. Huit differents mouvements sont proposés par les auteurs pour améliorer la solution initale. Quatre de ces mouvements consistent à : déplacer un patient à l'intérieur d'une même tournée ; échanger la place de deux patients à l'intérieur d'une même tournée ; déplacer un patient d'une tournée à une autre ; échanger deux patients appartenant à deux tournées distinctes. Les quatre autres mouvements sont identiques aux précédents à la différence qu'ils concernent des patients impliqués dans une synchronisation. Ces huit mouvements sont utilisés séquentiellement dans une descente en profondeur pour améliorer la solution initiale. L'algorithme est évalué sur des instances générées aléatoirement de 19 à 300 patients et de 3 à 40 soignants. Les solutions obtenues sont comparées à celles trouvées par le solveur CPLEX. En dix heures tout au plus, ce dernier prouve l'optimalité de petites instances de 10 patients et 30 soignants et la réalisabilité d'instances de 25 patients et 5 soignants avec un gap de 35.1% avec la borne inférieure. Le solveur n'est pas capable de trouver des solutions réalisables (sauf dans un seul cas) pour des instances de plus grandes en moins de 10h. L'heuristique trouve les solutions optimales pour de petites instances en moins d'une seconde et de meilleures solutions que celles obtenues par CPLEX pour les instances de 25 patients. L'heuristique trouve des solutions en quelques secondes pour des instances d'au plus 100 patients et 20 soignants mais nécessite plus de 2 heures de calcul pour des instances de 300 patients.

[Trautsamwieser *et al.*, 2011] propose également une recherche à voisinage variable pour le HCRSP dans laquelle la solution initiale est déterminée à partir d'une heuristique de liste. Celle-ci affecte les services aux soignants en commençant par les trier en fonction du barycentre des fenêtres de temps tout en respectant les contraintes de qualification et de temps de travail. L'heuristique favorise l'affectation d'un service à la tournée qui est

la plus proche du patient qui le nécessite grâce aux fenêtres de temps et au temps de travail. Une procédure combinant perturbation et recherche locale est mise en œuvre dans le but d'améliorer la solution. La perturbation est utilisée pour diversifier la recherche en atteignant de nouveaux espaces de recherche. Une séquence de patients est déplacée d'une tournée à une autre ou deux séquences de patients issues de tournées distinctes sont échangées. Les tournées sont ensuite améliorées en exploitant le voisinage généré grâce à l'opérateur 3-opt qui détruit trois arêtes et les remplacent par trois nouvelles arêtes. Des expérimentations ont été conduites sur deux types d'instances : d'une part, des instances générées aléatoirement disposant de 4 à 20 soignants et 20 à 100 services, d'autre part, des instances réelles de 13 à 75 soignants, 86 à 411 patients et 86 à 512 services. Sur les instances générées aléatoirement, un écart de moins de 2% de l'optimal est obtenu pour les solutions trouvées par le solveur Xpress. Les solutions obtenues à partir d'instances réelles ont été comparées à la planification actuellement utilisée et une amélioration d'au plus 50% est constatée. Bien qu'en terme de qualité des solutions, ces résultats semblent concluants, ils nécessitent des temps de calcul plus élevés.

[Liu *et al.*, 2013] traite du HCRSP où des médicaments et du matériel médical provenant des pharmacies doivent être livrés au domicile des patients et des échantillons biologiques doivent être récupérés du domicile des patients pour être livrés aux laboratoires. Dans ces travaux, deux méthodes de résolution sont proposées : un algorithme hybride génétique décrit dans la section suivante et une recherche taboue. Les voisinages de la recherche taboue sont définis par deux mouvements : le premier mouvement consiste à déplacer un patient d'une tournée à une autre tandis que le deuxième mouvement échange deux patients appartenant à deux tournées disctinctes. Chaque mouvement est suivi par une procédure de ré-optimization constituée d'une méthode constructive et d'une recherche locale. Si aucune amélioration n'est obtenue en un certain nombre d'itérations, la recherche taboue est ré-initialisée à partir de la deuxième meilleure solution obtenue dans la population initiale de l'algorithme génétique. La recherche taboue est évaluée sur deux ensembles d'instances modifiées issus de la littérature du VRPTW et du Problème de Tournées de Véhicules avec retrait mixte et fenêtres de temps (VRPMBTW). Deux critères d'arrêt de la recherche taboue ont été imposés : soit un nombre maximum d'itérations et un nombre maximum d'itérations sans améliorations (TS1), soit un temps d'exécution (TS2). Le premier ensemble d'instances est obtenu en modifiant les instances de [Solomon, 1987] et [Gehring et Homberger, 1999] constituées respectivement de 40 à 120 services, de 10 à 20 soignants, et de 200 services et 50 soignants. Pour ce premier ensemble d'instances, des gains moyens de 17.1% sur le coût sont constatés par rapport aux solutions obtenues par le solveur CPLEX avec des temps de calcul raisonnables de 863.5 secondes en moyenne pour TS1 et 951.8 secondes pour TS2. Le deuxième ensemble d'instances regroupe les instances de [Hasama, 1998] composées de 100 patients et au plus 22 soignants avec une partie des patients (10%, 30% or 50%) nécessitant du matériel. Seul TS2 a été utilisé pour tester ce deuxième ensemble d'instances et des gains de 6.27 % sont obtenus par rapport à l'algorithme de recuit simulé proposé par [Hasama, 1998].

Méthodes à population [Akjiratikarl *et al.*, 2007] propose un algorithme d'essaims particulaires (Particle Swarm Optimization, PSO). Une particule encodant une solution dans l'espace de recherche, est représentée par une matrice de dimension 2 (la première

dimension étant les soignants et la deuxième, les services). Cette matrice permet de déterminer à la fois l'affectation des services aux soignants et le séquencement des services dans une tournée. La population initiale est construite à l'aide d'un algorithme minimisant la distance totale parcourue par les soignants. L'algorithme fait évoluer la population grâce à trois composantes : la vitesse, la meilleure solution connue et la meilleure solution connue d'un voisinage. Des expérimentations sont conduites sur différentes versions de cet algorithme dans lesquelles certaines composantes ont été supprimées. Les instances testées s'échelonnent sur 100 services, 50 patients et 12 soignants par jour. Les solutions obtenues ont été comparées à la planification établie manuellement et celle obtenue avec ILOG Dispatcher 4.0. La version de l'algorithme avec toutes ses composantes est plus performante que les autres versions et la planification établie manuellement. Des gains de 11% à 31% sur le kilométrage total ont été obtenus par rapport à ILOG Dispatcher 4.0 avec des temps de calcul l'inférieur à 3.5 minutes.

Comme cela a été déjà mentionné, [Liu *et al.*, 2013] a également développé un algorithme hybride génétique. L'algorithme s'inspire des travaux de [Prins, 2004]. Une solution est représentée par une permutation de patients sans délimiteurs. La population initiale est obtenue à partir de trois heuristiques de gain et d'une heuristique de plus proche voisin. L'opérateur de croisement pouvant générer des solutions non réalisables, il est appelé tant qu'une solution réalisable n'est pas obtenue. Un mécanisme d'élitisme est également mis en place afin de transmettre aux générations ultérieures de solutions, les meilleures spécificités des générations antérieures. Cet algorithme génétique est testé avec les mêmes configurations d'arrêt que la recherche taboue décrite précédemment. Les résultats expérimentaux démontrent des gains d'au plus 16.4 % sur le coût par rapport aux solutions obtenues par le solveur CPLEX avec un temps limite de 72 heures et 6.35% par rapport aux solutions obtenues par le recuit simulé proposé par [Hasama, 1998]. Cependant, ces résultats ne permettent pas conclure pas sur la supériorité de l'algorithme génétique sur la recherche taboue.

Méthodes hybrides [Bredström et Rönnqvist, 2008] développe une méthode hybride en utilisant des heuristiques pour la génération de colonnes. La méthode se décompose en sept étapes réparties en deux groupes. Les étapes du premier groupe permettent de générer une solution initiale :

- tout d'abord, chaque service est pré-affecté aux soignants ;
- ensuite, un programme en nombres entiers mixtes basé sur ces pré-affectations est résolu afin de réduire la taille de ces pré-affectations et peut-être identifier les services qui ne peuvent pas être affectés à un soignant ;
- enfin, à partir de ces affectations, un programme en nombres entiers de taille moindre est encore résolu pour trouver une solution réalisable.

Le deuxième groupe composé de 4 étapes est une procédure itérative. Cette dernière permet d'améliorer à chaque itération la meilleure solution réalisable en générant aléatoirement de nouvelles pré-affectations ; cela en résolvant le programme en nombres entiers résultant dans un temps limité. Cette méthode hybride est testée sur des instances générées aléatoirement. Ces instances sont composées d'au plus 16 soignants et 80 services et les solutions obtenues sont comparées à celles obtenues par le solveur CPLEX. Une analyse de sensitivité est proposée pour évaluer l'impact des contraintes de synchronisation, de la taille des fenêtres

de temps et le pourcentage des services synchronisés sur la résolution. La méthode permet d'obtenir des solutions de qualité équivalente en 2 minutes à celles obtenues par le solveur CPLEX en une heure.

[Bertels et Fahle, 2006] propose une méthode combinant la programmation par contraintes, la programmation linéaire et la recherche taboue ou le recuit simulé. La méthode est composée de deux phases : l'affectation des services au personnel de soins, et le séquencement des services pour chaque tournée. La programmation par contraintes affecte un sous-ensemble de services à chaque soignant et la programmation linéaire détermine le séquencement. Enfin, une recherche taboue ou un recuit simulé est appliqué pour améliorer la solution obtenue. L'ensemble de la méthode est limitée à un temps d'exécution de 15 minutes. Les expérimentations numériques ont été conduites sur 120 instances artificielles d'au plus 50 soignants, 200 patients et 600 services. Ces expérimentations montrent que la combinaison d'une procédure de programmation par contraintes et d'une recherche taboue représente la meilleure combinaison en améliorant les solutions de plusieurs instances.

Une heuristique basée sur la décomposition de benders est proposée par [Cire et Hooker, 2012]. Deux étapes composent cette heuristique. Elles consistent à résoudre un problème maître qui affecte à chaque soignant un sous-ensemble de services, en satisfaisant les contraintes de qualification et à résoudre les sous problèmes – un pour chaque soignant – pour le séquencement des services. Le problème maître est résolu grâce à une heuristique gloutonne favorisant les services nécessitant les qualifications les plus élevées et dont le nombre de soignants pour les réaliser est faible. Contrairement, au problème maître, chaque sous-problème est résolu à l'optimal avec la programmation par contraintes. La réalisabilité des sous-problèmes permet de renforcer le problème maître en y ajoutant de nouvelles coupes. Ce processus est itéré jusqu'à ce que la meilleure solution connue ne puisse plus être améliorée. Huit instances réelles d'au plus 11 soignants et 92 patients permettent de mener des expérimentations numériques en comparant les solutions obtenues par l'heuristique à celles obtenues de manière exacte par la programmation par contraintes.

Une méthode par décomposition similaire est proposée par [Allaoua *et al.*, 2013]. Dans la première phase, les services nécessitant les mêmes qualifications sont regroupés ensemble. Dans la deuxième phase, un Problème de Voyageur de Commerce avec fenêtres de temps est résolu pour chaque groupe. Contrairement à l'heuristique précédemment décrite, un programme en nombres entiers mixte est utilisé à la place de la programmation par contraintes. La méthode de [Allaoua *et al.*, 2013] est évaluée sur 90 instances générées aléatoirement avec au plus 9 soignants et 30 soignants. Pour 50.52% des instances, les solutions obtenues sont identiques à celles obtenues par le solver CPLEX.

[Rendl *et al.*, 2012] et [Hiermann *et al.*, 2013] proposent plusieurs méthodes hybrides pour la résolution du HCRSP mutlimodal. Un encodage direct est mis en œuvre pour représenter une solution où une liste de services est associée à chaque soignant. Cette liste encode à la fois le séquencement et l'affectation des services au soignant. Quatre métaheuristiques sont implémentées : une recherche à voisinage variable (Variable Neighborhood Search, VNS), un algorithme mémétique (Memetic Algorithm, MA), un recuit simulé (Simulated Annealing, SA), une recherche dispersée (Scatter Search, SS). La population initale est générée aléatoirement ou à l'aide de la programmation par contraintes. L'ensemble des métaheuristiques implémentées sont évaluées sur des instances réelles de 482/518 soignants et

679/717 services. Toutes ces métaheuristiques obtiennent des solutions nécessitant moins de soignants que disponibles (approximativement 35% des soignants disponibles). Les résultats expérimentaux montrent que ces métaheuristiques fournissent des solutions dans des temps d'exécution raisonnables. L'algorithme mémétique dépasse sur le plan qualitatif les autres métaheuristiques. Les récents succès de l'algorithme mémétique dans la résolution du Problème de Tournées multi-attributs [Vidal *et al.*, 2014] viennent renforcer ces conclusions.

1.6.2 Méthodes traitant un horizon de planification à long terme

1.6.2.1 Méthodes exactes

[Trautsamwieser et Hirsch, 2014] propose un algorithme branch–and–price–and–cut (BPC) pour résoudre le HCRSP hebdomadaire. Le problème est décomposé grâce à une décomposition de Dantzig-Wolfe où le problème maître va combiner plusieurs tournées journalières réalisables pour obtenir une solution au problème planification hebdomadaire en ayant pour objectif de minimiser le temps total de travail. Ces tournées journalières réalisables sont obtenues en résolvant les problèmes esclaves de la décomposition. De nombreuses règles de branchement ont été introduites afin d'obtenir des solutions entières. La solution initiale est obtenue grâce à une recherche à voisinage variable permettant d'avoir des bornes supérieures pour l'algorithme BPC. La méthode proposée est évaluée pour résoudre en moins d'une heure des instances de taille réelle avec au plus 9 soignants, 45 patients et 203 services.

1.6.2.2 Méthodes approchées

Méthodes à solution unique Deux procédures de recherche adaptative randomisée gloutonne (greed randomized adaptive search procedure, GRASP) ont été proposées par [Bard *et al.*, 2014] et [Bard *et al.*, 2013]. Elles s'inscrivent dans deux perspectives de résolution : séquentielle et parallèle. Certains patients doivent être soignés au moins une fois par les soignants. Les résultats expérimentaux tendent à montrer que la version séquentielle de la procédure est plus performante que la version parallèle. Dans [Bard *et al.*, 2014], la solution initiale est construite en trois étapes. Dans la première étape, un jour de soin est déterminé pour chaque patient en fonction des disponibilités des soignants. La deuxième étape permet de construire les couples (soignant, jour) pour toutes les tournées de l'horizon de planification. Les jours maximisant la disponibilité globale des soignants et les soignants ayant les salaires horaires les plus faibles sont favorisés. Dans la troisième étape, les patients sont sélectionnés aléatoirement pour réaliser le séquencement pour chaque tournée. Les étapes 2 et 3 sont réitérées jusqu'à ce que l'ensemble des tournées soient complètes. La solution initiale étant construite, une recherche locale utilisant des échanges et des insertions intra et inter tournées ont été implémentées dans le but d'améliorer la solution. 5 instances réelles de 38 à 113 patients, 93 à 274 services, 4 à 12 soignants et 80 instances générées aléatoirement sont utilisées pour les expérimentations numériques. Les solutions obtenues sur les instances réelles par la procédure séquentielle réduisent de 26.57% le coût par rapport à la planification actuelle. La version séquentielle obtient des résultats supé-

rieurs à la version parallèle lorsque la taille des séquences augmente avec en moyenne une réduction des coûts de 5.67% pour 4 instances réelles. Des conclusions similaires peuvent être faites sur les instances générées aléatoirement.

[Maya Duque *et al.*, 2015] propose un algorithme à deux phases pour la résolution du HCRSP où la maximisation du niveau de service est pris en compte dans la première phase, et la minimisation de la distance totale est considérée dans la seconde phase. L'ensemble des combinaisons de jour est énuméré exhaustivement dans la première phase pour la mise en place d'une planification des services des patients. À partir de ces combinaisons, un nombre limité de combinaisons est généré pour obtenir la meilleure affectation entre les soignants et les patients. La deuxième phase minimise la distance totale tout en acceptant la détérioration de la solution courante avec une certaine tolérance. Une recherche locale randomisée échangeant les patients de tournées distinctes est implémentée. La méthode proposée est testée sur 30 instances générées aléatoirement de 26 à 109 patients et de 3 à 25 soignants. La méthode fournit des solutions avec un bon niveau de service, à plus de 89.35% au dessus de la valeur théorique idéale. En plus, une réduction de 11.67% de la distance totale est observée en moyenne.

Méthodes hybrides [Liu *et al.*, 2014] combine une recherche taboue et des procédures de recherche locale. La recherche taboue est basée sur la structure de recherche taboue proposée par [Cordeau *et al.*, 1997]. Un critère a été ajouté à la fonction économique pour mettre en place une oscillation stratégique entre les espaces réalisables et irréalisables. La solution initiale est obtenue grâce à une heuristique d'insertion. Les fenêtres de temps peuvent être violées et ainsi conduire à générer une solution irréalisable. Cette étape est suivie d'une exploration du voisinage à travers des mouvements entre tournées. La recherche taboue est donc hybridée par des procédures de recherche locale intra tournées pour intensifier la recherche. Pour ce faire, les mouvements 1-1, 1-0, 2-opt sont implémentées tout en tolérant les mouvements irréalisables. De larges expérimentations numériques sont conduites sur différentes instances de test incluant à la fois des extensions d'instances de la littérature ([Solomon, 1987] et [Gehring et Homberger, 1999]) et des instances réelles. Les résultats révèlent que la prise en compte de solutions irréalisables dans le processus de recherche est profitable quant à l'amélioration de la qualité des solutions. Une amélioration de 9% à 16% est constatée par rapport à la planification existante sur les 5 instances réelles de 2 soignants, 49-58 patients et 5 jours.

[Di Gaspero et Urli, 2014] a développé une approche hybride en combinant la programmation par contraintes et la recherche à voisinage large. La programmation par contraintes est mise en œuvre pour trouver une solution initiale avec un schéma de branchement calculant d'abord le nombre de soignants nécessaires chaque jour, puis leurs tournées. Dans ces travaux, la recherche à voisinage large permet de réinitialiser les variables de décision et réparer la solution. La performance de l'algorithme est évaluée sur 540 instances groupées en 18 familles. Chaque instance dispose d'au plus 6 jours de travail et 40 services. D'après ces résultats, la recherche à voisinage large obtient de meilleures performances que l'approche de programmation par contraintes.

1.7 Conclusion

Ce chapitre rend compte des différents modèles développés pour le Problème de Tournées du personnel de soins à domicile. Les caractéristiques du problème ont été décrites à partir d'un regroupement basé sur : les patients, les soignants et la structure de soins à domicile. La revue de la littérature a permis de mettre en lumière la manière dont ces caractéristiques ont été prises en compte dans les modèles soit comme contrainte, soit comme fonction économique. En ce qui concerne les contraintes, trois types de contraintes ont été identifiés : les contraintes temporelles, les contraintes d'affection et les contraintes géographiques. Finalement, les travaux ont été également classés en fonction de la méthode de résolution mise en œuvre pour résoudre le problème en soulignant les stratégies gagnantes qu'elles prennent en compte.

Cet état de l'art montre que de nombreuses approches de modélisation ont été proposées dans la littérature. Le nombre élevé de modèles nous révèle la grande diversité des structures de soins à domicile qui diffèrent selon les pays, selon les législations, selon les patients soignés, ... Une des conséquences qui découle de ce constat est que les méthodes de résolution mises en œuvre sont difficiles à comparer. Cela nous conduit donc à proposer une perspective de recherche qui consiste à développer des modèles et méthodes génériques couvrant autant que possible la diversité des structures de soins à domicile. Il est à observer également que les modèles proposés sont statiques et déterministes. D'une part, les patients entrent et sortent dans une structure de soins à domicile, l'aspect dynamique est indispensable pour prendre compte ces spécificités liées au soin à domicile. D'autre part, les données étant dans la réalité incertaines, l'aspect stochastique voire robuste serait une autre piste de recherche à explorer.

Dans les chapitres qui suivent, nous nous sommes attelés à mettre en place des modèles et des méthodes génériques dans le but de couvrir la plupart des caractéristiques importantes de ce problème.

Chapitre 2

Modèle générique pour la planification des tournées du personnel de soins à domicile

Dans le chapitre précédent, une revue de la littérature a été proposée décrivant les caractéristiques des différents problèmes ainsi que les algorithmes mis en œuvre pour les résoudre. Comme nous l'avons déjà souligné, chaque étude se focalise sur un problème auquel fait face une structure de soins à domicile spécifique ne touchant qu'un sous-ensemble de caractéristiques. Les modèles sont donc difficilement comparables. Ce chapitre est une tentative d'unification de ces modèles. Ce chapitre est organisé comme suit. Dans la section 2.1, nous récapitulons l'ensemble de nos contributions du chapitre courant. Une formulation du problème aussi générique que possible est proposée dans les sections 2.2 et 2.3. Dans ces sections, nous décrivons d'abord le problème en présentant les données et les notations, puis les contraintes et enfin les fonctions économiques. Les limitations de ce modèle sont exposées dans la section 2.4 et dans la section 2.5, la génération d'instances. Enfin, les expérimentations menées ont été rassemblées dans la section 2.6.

2.1 Contributions

Les contributions de ce chapitre sont de deux ordres. D'une part, un modèle générique du problème de tournées du personnel de soins à domicile est proposé couvrant ainsi la plupart des travaux de la littérature. Ce modèle tient compte des caractéristiques du problème les plus communément rencontrées et comble un des manques de la littérature qui a été souligné : l'absence de modèle générique. Ce modèle servira de base à la construction de méthodes de résolution générales. D'autre part, une génération d'instances est réalisée à partir de ce modèle. Les instances générées sont regroupées en plusieurs classes. Chacune d'entre elles représente une combinaison des caractéristiques du problème. Plusieurs ensembles d'instances ont été également repris de la littérature et convertis dans ce même format. Enfin, des expérimentations numériques ont été conduites afin de valider le modèle à l'aide du solveur de programmation mathématique CPLEX.

2.2 Description du problème

Le problème considéré a été formulé de manière à être le plus générique possible et en supposant que toutes les données sont déterministes et connues à l'avance. L'objectif est de planifier les tournées d'un ensemble $\mathscr{K}$ de soignants sur un horizon de planification d'un ensemble $\mathscr{H}$ d'un ou plusieurs jours afin de satisfaire la demande d'un ensemble $\mathscr{P}$ de patients. Cette demande correspond à un ensemble $\mathscr{S}$ de services. L'horizon de planification est divisé en un ensemble $\mathscr{W}$ de semaines. Chaque patient $p \in \mathscr{P}$ nécessite un ensemble de services noté $\mathscr{S}_p$.

Un service peut représenter bien sûr un soin à prodiguer, mais peut être aussi une livraison de médicaments, un prélèvement de sang, une consultation médicale, un service à la personne, etc. Un service sera réalisé au plus une fois dans l'horizon de planification et pour un seul patient. Les services ne pouvant être réalisés, seront sous-traités et affectés au soignant fictif k_f. Ce dernier a la faculté de réaliser tous les services qui lui sont affectés et les contraintes auxquelles sont soumis ces services ne sont pas respectées ici. Le coût de sous-traitance associé à un service i donné est noté cs_i. La sous-traitance peut intervenir lorsque la structure de soins à domicile fait face à une situation de sous-effectif ou qu'aucun soignant disponible n'est en mesure de réaliser un service. La réalisation d'un service i de durée π_i, ne peut intervenir que dans l'ensemble $\mathscr{H}_i$ des jours pour lequel le patient qui le nécessite est disponible. Pour un jour donné d, chaque service dispose de γ_{id} fenêtres de temps, qui correspond aux différentes plages horaires auxquelles le patient qui le nécessite est disponible ce jour. Chaque fenêtre de temps sera notée $[e^n_{id}, l^n_{id}]$, avec $n \in \{1, \ldots, \gamma_{id}\}$. Un service périodique, c'est-à-dire répété dans le temps (ex : tous les 3 jours), sera modélisé par plusieurs services contraints par des délais Δ^{min}_{ij} et Δ^{max}_{ij} en jours à respecter entre chaque service. Si un soin nécessite plusieurs soignants alors il sera modélisé par autant de services que de soignants nécessaires et ces services seront contraints de sorte à ce qu'ils soient planifier à la même heure. Deux services liés par une contrainte de précédence ou de délai seront séparés par des délais min δ^{min}_{ij} et/ou max δ^{max}_{ij} selon la contrainte traitée. Les services en exclusion appartiennent à un ensemble ϵ de services et sont contraints de sorte à empêcher tout chevauchement. L'ensemble de tous les ensembles de services en exclusion est noté $\mathscr{E}$.

Afin de réaliser ces services, les soignants devront faire des tournées qui débuteront au sommet $\mathfrak{o}_k$ et finiront au sommet $\mathfrak{d}_k$. Ces sommets ne sont pas nécessairement disjoints et peuvent correspondre à un hôpital, à une structure de soins à domicile ou au domicile du soignant. Ces tournées interviendront uniquement dans un ensemble $\mathscr{H}_k$ de jours pour lequel il est disponible. Les soignants peuvent être des infirmiers, des médecins, des kinésithérapeutes, etc. Par conséquent, selon le service qu'ils seront amenés à réaliser, ils devront impérativement disposer de la qualification q_{ik} requise. Cette qualification rend simplement compte de la possibilité pour un soignant de réaliser un service, pas de son niveau de compétence. Dans une situation de partitionnement géographique, cette qualification pourra être considérée comme une autorisation à se rendre dans le secteur auquel un service appartient. En plus de disposer de la qualification requise, la réalisation d'un service génère un coût cp_{ikd} qui correspond aux frais nécessaires pour sa réalisation. Ce coût peut également être considéré comme le prix que facture la structure au patient ; auquel cas, c'est un gain. Pour un jour donné d, le début d'une tournée sera contraint par une plage horaire $[e_{kd}, l_{kd}]$.

Afin de répondre aux dispositions légales selon les pays, une durée légale de travail ω_{kd} a été introduite. Au delà, chaque heure entamée sera considérée comme une heure supplémentaire de coût unitaire co_{kd}, qui dépend du jour. Chaque soignant est autorisé à faire au plus ot_k^{jour} heures supplémentaires journalières et ot_k^{heb} heures supplémentaires hebdomadaires. Une autre disposition légale impose une pause lorsque la tournée d'un soignant est trop longue. Ainsi, une durée de travail minimale θ_k^{min} et maximale θ_k^{max} sans pause encadre la date de début de la pause. Celle-ci de durée ρ_k sera effectuée au sommet $\mathbf{b}_{kd}$.

Étant soumis à un horizon de planification de plusieurs jours, les soins prodigués aux patients s'inscrivent dans la durée. Une relation de confiance entre un soignant et un patient s'installe et le suivi d'un patient par un même soignant se révèle dans ces conditions préférable. Afin de prendre en compte cette situation, nous disposons d'un historique des soins prodigués pour un même patient. Cet historique correspond à l'ensemble des soignants $\mathscr{K}_p$ ayant déjà soigné un patient donné au cours des périodes précédentes. Son complémentaire noté $\overline{\mathscr{K}}_p$ correspond donc à l'ensemble des soignants n'ayant pas soigné ce patient. L'idée est d'éviter de faire appel aux soignants n'ayant pas soigné ce patient. Au plus, ζ_p soignants de $\overline{\mathscr{K}}_p$ peuvent réaliser des services du patient p.

L'ensemble du réseau routier est modélisé par un graphe $\mathscr{G} = (\mathscr{V}, \mathscr{A})$, un graphe non-orienté avec $\mathscr{V} = \mathscr{S} \cup \mathscr{O} \cup \mathscr{D} \cup \mathscr{B}$, l'ensemble des sommets et $\mathscr{A} = \{(i,j) \mid i \in \mathscr{V}, j \in \mathscr{V}\}$, l'ensemble des arêtes du graphe. L'ensemble $\mathscr{O}$ (respectivement, $\mathscr{D}$) correspond à l'ensemble de tous les sommets de départ (respectivement, de retour) des soignants. Quant à l'ensemble $\mathscr{B}$, il s'agit de l'ensemble de tous les sommets de pause des soignants. Chaque arc représente le plus court chemin pour rejoindre un sommet j en partant d'un sommet i auquel une estimation τ_{ij} du temps de trajet est associée.

Le tableau 2.1 récapitule l'ensemble de la notation introduite et définit brièvement chaque donnée. Nous conseillons au lecteur de se rapporter à ce tableau afin de faciliter la lecture des différentes formules qui suivront.

2.3 Programme Linéaire en Nombres Entiers

Cette section contient la formulation mathématique du problème. L'ensemble des variables de décisions est d'abord introduit tout en précisant le domaine de définition de chaque indice. Les contraintes du problème sont ensuite formulées et décrites succinctement. Cette section se clôt par la description des différentes fonctions économiques prises en compte.

2.3.1 Variables

La planification sera mise en place grâce aux variables de décisions suivantes :

— x_{ij}^{kd} : 1 si le soignant k visite le sommet i avant le sommet j le jour d ; 0 sinon
Ces variables correspondent au choix des arêtes à emprunter par le soignant et permettent ainsi la construction des tournées des soignants.

— st_i : 1 si le service i est sous-traité ; 0 sinon

Données	**Description**
$\mathscr{H}$	ensemble des jours de la planification
$\mathscr{W}$	ensemble des semaines
$\mathscr{K}$	ensemble des soignants (soignant fictif k_f inclus si la sous-traitance est autorisée)
$\mathscr{P}$	ensemble des patients
$\mathscr{S}$	ensemble des services
$\mathscr{O}$	ensemble des sommets de départ des soignants
$\mathscr{D}$	ensemble des sommets de retour des soignants
$\mathscr{B}$	ensemble des sommets de pause
$\mathscr{V}$	ensemble des sommets ; $\mathscr{V} = \mathscr{S} \cup \mathscr{O} \cup \mathscr{D} \cup \mathscr{B}$
$\mathscr{A}$	ensemble des arêtes du graphe ; $\mathscr{A} = \{(i,j) \mid i \in \mathscr{V}, j \in \mathscr{V}\}$
$\boldsymbol{\tau}_{ij}$	temps de trajet nécessaire pour traverser l'arc $(i,j), \forall (i,j) \in \mathscr{A}$
Services	
$\mathscr{H}_i$	jours durant lesquels le service i peut être réalisé ; $\mathscr{H}_i \subseteq \mathscr{H}$
π_i	temps nécessaire pour réaliser le service $i \in \mathscr{S}$
cs_i	coût de sous-traitance du service i
γ_{id}	nombre de fenêtres de temps durant lesquelles le service i peut être réalisé le jour d
$[e^n_{id}, l^n_{id}]$	n-ième fenêtre de temps pour le service i le jour d ; $n \in \{1, \ldots, \gamma_{id}\}$
$\delta^{min}_{ij}/\delta^{max}_{ij}$	délai min/max en heures entre les services i et j
$\Delta^{min}_{ij}/\Delta^{max}_{ij}$	délai min/max en jours entre les services i et j
$\mathscr{E}$	ensemble des ensembles de services ne devant pas être réalisés en même temps
Soignants	
k_f	soignant fictif qui n'est soumis à aucune contrainte
$\mathscr{H}_k$	jours pour lesquels le soignant k est disponible ; $\mathscr{H}_k \subseteq \mathscr{H}$
$\mathfrak{o}_k$	sommet de départ du soignant k ; $\forall k \in \mathscr{K},\ \exists! \mathfrak{o}_k \in \mathscr{O}$
$\mathfrak{d}_k$	sommet de retour du soignant k ; $\forall k \in \mathscr{K},\ \exists! \mathfrak{d}_k \in \mathscr{D}$
$\mathfrak{b}_{kd}$	sommet de pause du soignant k le jour d ; $\forall k \in \mathscr{K}, \forall d \in \mathscr{H},\ \exists! \mathfrak{b}_{kd} \in \mathscr{O}$
q_{ik}	égal à 1 si le soignant k dispose des qualifications suffisantes pour réaliser le service i
ρ_k	durée de la pause pour le soignant k
$[e_{kd}, l_{kd}]$	fenêtre de temps pour le début de la tournée du soignant k le jour d
ω_{kd}	temps de travail max pour le soignant k le jour d
ot^{jour}_k/ot^{heb}_k	nombre d'heures supplémentaires journalières/hebdomadaires autorisées
$\theta^{min}_k/\theta^{max}_k$	temps de travail min/max sans pause pour le soignant k
co_{kd}	coût unitaire des heures supplémentaires du soignant k le jour d
cp_{ikd}	coût pour réaliser le service i, le jour d par le soignant k
Patients	
$\mathscr{S}_p$	ensemble des services associés au patient p
$\mathscr{K}_p \subseteq \mathscr{K}$	ensemble des soignants ayant déjà soigné le patient p
$\overline{\mathscr{K}}_p$	ensemble des soignants n'ayant jamais soigné le patient p
ζ_p	nombre max de soignants autorisés à soigner le patient p et n'appartenant pas à $\mathscr{K}_p$
	$\zeta_p \leq \lvert\overline{\mathscr{K}}_p\rvert$

TABLE 2.1 – Notations

— y_{ij}^d : 1 si le service i est réalisé avant le service j le jour d ; 0 sinon
Les variables x_{ij}^{kd} permettent de déterminer la précédence entre deux services s'ils appartiennent à une même tournée. Cependant, si ces services sont réalisés par des soignants différents, les variables x_{ij}^{kd} ne suffisent pas pour déterminer la précédence entre ces services. Les variables y_{ij}^d ont ainsi été introduites pour combler cette lacune.
— u_{id}^n : 1 si le service i est effectué dans la n-ième fenêtre de temps du jour d ; 0 sinon.
— t_i : date de début du service i.
— oc : le coût total du nombre d'heures supplémentaires effectuées par l'ensemble des soignants
— $tard_i$: retard du service i.
— $tard^{max}$: retard maximal pour l'ensemble des services
— $tbreak_{kd}$: date de début de la pause du soignant k le jour d.
— $tstart_{kd}$: date de début de la tournée du soignant k le jour d.
— $tend_{kd}$: date de fin de la tournée du soignant k le jour d.
— z_k^p : 1 si le soignant k soigne le patient p ; 0 sinon.
— g^p : 1 si au moins un service du patient p est sous-traité ; 0 sinon.

2.3.2 Contraintes

Contraintes sur le choix des arêtes :

$$st_i + \sum_{k\in\mathscr{K}} \sum_{d\in\mathscr{H}_i\cap\mathscr{H}_k} \sum_{j\in\mathscr{S}\cup\{\mathfrak{d}_k\}\cup\{\mathfrak{b}_{kd}\}\setminus\{i\}} q_{ik}\cdot x_{ij}^{kd} = 1, \forall i \in \mathscr{S} \tag{2.1}$$

La réalisation de chaque service est garantie par les contraintes (2.1). Chaque service est réalisé une seule fois par un soignant ou est sous-traité.

$$\sum_{j\in\mathscr{S}\cup\{\mathfrak{d}_k\}} x_{\mathfrak{b}_{kd}j}^{kd} \leq 1, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}_k \tag{2.2}$$

Les contraintes (2.2) assurent à chaque soignant au plus une pause journalière.

$$\sum_{i\in\mathscr{S}\cup\{\mathfrak{b}_{kd}\}\cup\{\mathfrak{o}_k\}\setminus\{h\}} x_{ih}^{kd} = \sum_{j\in\mathscr{S}\cup\{\mathfrak{b}_{kd}\}\cup\{\mathfrak{d}_k\}\setminus\{h\}} x_{hj}^{kd}, \forall h \in \mathscr{S}, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}_h\cap\mathscr{H}_k \tag{2.3}$$

$$\sum_{i\in\mathscr{S}\cup\{\mathfrak{o}_k\}} x_{i\mathfrak{b}_{kd}}^{kd} = \sum_{j\in\mathscr{S}\cup\{\mathfrak{d}_k\}} x_{\mathfrak{b}_{kd}j}^{kd}, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}_k \tag{2.4}$$

Les contraintes (2.3) et (2.4) assurent la continuité des tournées. Ainsi, un soignant qui atteint un sommet du graphe (le domicile d'un patient ou le sommet de l'éventuelle pause journalière) devra ensuite le quitter.

$$\sum_{j\in\mathscr{S}\cup\{\mathfrak{b}_{kd}\}} x_{\mathfrak{o}_k j}^{kd} \leq 1, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}_k \tag{2.5}$$

$$\sum_{i \in \mathscr{S} \cup \{\mathfrak{b}_{kd}\}} x^{kd}_{i\mathfrak{d}_k} \leq 1, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}_k \tag{2.6}$$

$$\sum_{j \in \mathscr{S} \cup \{\mathfrak{b}_{kd}\}} x^{kd}_{\mathfrak{o}_k j} = \sum_{i \in \mathscr{S} \cup \{\mathfrak{b}_{kd}\}} x^{kd}_{i\mathfrak{d}_k}, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}_k \tag{2.7}$$

Les contraintes (2.5) et (2.6) garantissent la cohérence des sommets d'origine et de destination de chaque soignant. Un soignant ne pourra pas commencer (respectivement finir) sa tournée dans un sommet différent du sommet d'origine (respectivement de destination). Quant à la contrainte (2.7), elle force l'existence d'un retour au sommet $\mathfrak{d}_k$ si il y a un départ du sommet $\mathfrak{o}_k$.

Contraintes temporelles :

$$t_i + \pi_i + \tau_{ij} \leq t_j + M \cdot (1 - x^{kd}_{ij}), \forall i \in \mathscr{S}, \forall j \in \mathscr{S} \setminus \{i\}, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}_i \cap \mathscr{H}_j \cap \mathscr{H}_k \tag{2.8}$$

$$t_i + \pi_i + \tau_{i\mathfrak{b}_{kd}} \leq tbreak_{kd} + M \cdot (1 - x^{kd}_{i\mathfrak{b}_{kd}}), \forall i \in \mathscr{S}, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}_i \cap \mathscr{H}_k \tag{2.9}$$

$$tbreak_{kd} + \rho_k + \tau_{\mathfrak{b}_{kd}j} \leq t_j + M \cdot (1 - x^{kd}_{\mathfrak{b}_{kd}j}), \forall j \in \mathscr{S}, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}_j \cap \mathscr{H}_k \tag{2.10}$$

Les contraintes (2.8), (2.9) et (2.10) contraignent les dates de début des services et des pauses. Ainsi, on s'assure que la durée séparant la date de début de deux services est suffisante pour la réalisation du premier et pour rejoindre le second. Il en est de même pour les pauses lorsqu'elles précèdent un service. Ces contraintes jouent un rôle très important dans la résolution du problème car elles éliminent les sous-tours.

$$\sum_{n=1}^{\gamma_{id}} u^n_{id} \leq \sum_{k \in \mathscr{K}} \sum_{j \in \mathscr{S} \cup \{\mathfrak{b}_{kd}\} \cup \{\mathfrak{d}_k\} \setminus \{i\}} x^{kd}_{ij}, \forall i \in \mathscr{S}, \forall d \in \mathscr{H}_i \tag{2.11}$$

$$\sum_{d \in \mathscr{H}_i} \sum_{n=1}^{\gamma_{id}} u^n_{id} = 1 - st_i, \forall i \in \mathscr{S} \tag{2.12}$$

$$\sum_{d \in \mathscr{H}_i} \sum_{n=1}^{\gamma_{id}} e^n_{id} u^n_{id} \leq t_i, \forall i \in \mathscr{S} \tag{2.13}$$

$$M \cdot st_i + tard_i + \sum_{d \in \mathscr{H}_i} \sum_{n=1}^{\gamma_{id}} l^n_{id} u^n_{id} \geq t_i, \forall i \in \mathscr{S} \tag{2.14}$$

$$tard_i \leq tard^{max}, \forall i \in \mathscr{S} \tag{2.15}$$

Les contraintes (2.11), (2.12) et (2.13) réduisent le domaine de définition des dates de début des services en les incluant dans l'une des fenêtres temporelles définies. Par conséquent, la réalisation d'un service est ainsi liée à l'une des plages horaires auxquelles le patient concerné est disponible. La figure 2.1 illustre cette situation dans laquelle pour un jour donné, les fenêtres de temps sont coloriées en vert. Si un service est sous-traité, l'ensemble de ces contraintes sont levées ; sinon, un retard est calculé (2.14) et (2.15). Les termes concernant la sous-traitance n'apparaîtront uniquement si elle est autorisée ; de même, pour le retard.

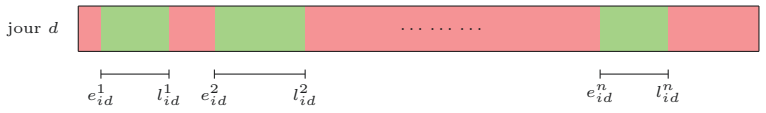


FIGURE 2.1 – Fenêtres de temps

$$e_{kd} \leq tstart_{kd} \leq l_{kd}, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}_k \tag{2.16}$$

$$tstart_{kd} + \tau_{\mathfrak{o}_k j} \leq t_j + M \cdot (1 - x^{kd}_{\mathfrak{o}_k j}), \forall k \in \mathscr{K}, \forall j \in \mathscr{S}, \forall d \in \mathscr{H}_k \cap \mathscr{H}_j \tag{2.17}$$

$$t_i + \pi_i + \tau_{i\mathfrak{d}_k} \leq tend_{kd} + M \cdot (1 - x^{kd}_{i\mathfrak{d}_k}), \forall k \in \mathscr{K}, \forall i \in \mathscr{S}, \forall d \in \mathscr{H}_k \cap \mathscr{H}_i \tag{2.18}$$

Les contraintes (2.16), (2.17) et (2.18) contraignent les dates de début et de fin de la tournée de chaque soignant. Les dates de début et de fin doivent correspondre aux horaires de travail du soignant. Les temps de trajet et les durées de réalisation ont été pris en compte pour garantir la cohérence de ces dates.

$$tstart_{kd} + \tau_{\mathfrak{o}_k \mathfrak{b}_{kd}} \leq tbreak_{kd} + M \cdot (1 - x^{kd}_{\mathfrak{o}_k \mathfrak{b}_{kd}}), \forall k \in \mathscr{K}, \forall d \in \mathscr{H}_k \tag{2.19}$$

$$tbreak_{kd} + \rho_k + \tau_{\mathfrak{b}_{kd}\mathfrak{d}_k} \leq tend_{kd} + M \cdot (1 - x^{kd}_{\mathfrak{b}_{kd}\mathfrak{d}_k}), \forall k \in \mathscr{K}, \forall d \in \mathscr{H}_k \tag{2.20}$$

$$tend_{kd} - tstart_{kd} \leq \theta^{max}_k + M \cdot \sum_{i \in \mathscr{S} \cup \{\mathfrak{o}_k\}} x^{kd}_{i\mathfrak{b}_{kd}}, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}_k \tag{2.21}$$

$$M \cdot (1 - \sum_{i \in \mathscr{S} \cup \{\mathfrak{o}_k\}} x^{kd}_{i\mathfrak{b}_{kd}}) + \theta^{max}_k \geq tbreak_{kd} - tstart_{kd}, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}_k \tag{2.22}$$

$$tbreak_{kd} - tstart_{kd} \geq \theta^{min}_k - M \cdot (1 - \sum_{i \in \mathscr{S} \cup \{\mathfrak{o}_k\}} x^{kd}_{i\mathfrak{b}_{kd}}), \forall k \in \mathscr{K}, \forall d \in \mathscr{H}_k \tag{2.23}$$

$$M \cdot (1 - \sum_{i \in \mathscr{S} \cup \{\mathfrak{o}_k\}} x^{kd}_{i\mathfrak{b}_{kd}}) + \theta^{max}_k \geq tend_{kd} - tbreak_{kd} - \rho_k, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}_k \tag{2.24}$$

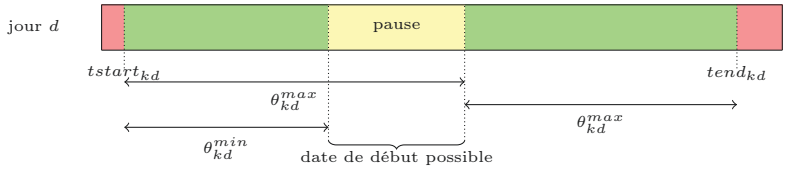


FIGURE 2.2 – Pause

Les contraintes (2.19) à (2.21) encadrent la date de début d'une éventuelle pause de chaque soignant. Les contraintes (2.22) à (2.24) organisent les pauses de chaque soignant. Si le soignant k prend une pause le jour d, elles le forcent à travailler au moins θ^{min}_k mais pas plus de θ^{max}_k avant de prendre son unique pause journalière. La figure 2.2 illustre le placement d'une éventuelle pause dans une journée.

$$t_j - t_i \leq \delta_{ij}^{max}, \forall i \in \mathscr{S}, \forall j \in \mathscr{S} \setminus \{i\}, \forall d \in \mathscr{H}_i \cap \mathscr{H}_j \tag{2.25}$$

$$t_j - t_i \geq \delta_{ij}^{min}, \forall i \in \mathscr{S}, \forall j \in \mathscr{S} \setminus \{i\}, \forall d \in \mathscr{H}_i \cap \mathscr{H}_j \tag{2.26}$$

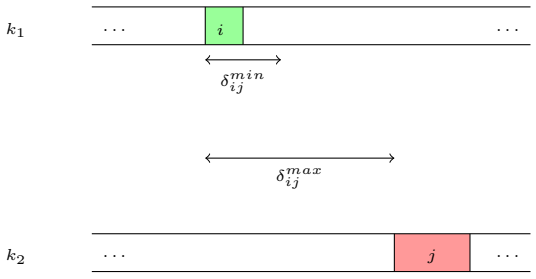


FIGURE 2.3 – Délais

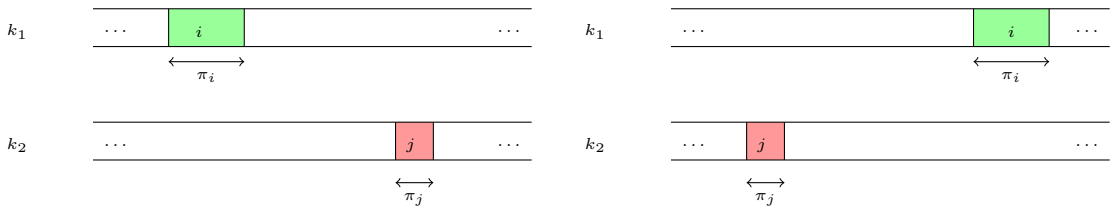
$$\sum_{k \in \mathscr{K}} \sum_{d \in \mathscr{H}} \sum_{h \in \mathscr{S} \cup \mathscr{D} \setminus \{j\}} d \cdot x_{jh}^{kd} - \sum_{k \in \mathscr{K}} \sum_{d \in \mathscr{H}} \sum_{h \in \mathscr{S} \cup \mathscr{O} \setminus \{i\}} d \cdot x_{ih}^{kd} \leq \Delta_{ij}^{max}, \forall i \in \mathscr{S}, \forall j \in \mathscr{S} \setminus \{i\} \tag{2.27}$$

$$\sum_{k \in \mathscr{K}} \sum_{d \in \mathscr{H}} \sum_{h \in \mathscr{S} \cup \mathscr{D} \setminus \{j\}} d \cdot x_{jh}^{kd} - \sum_{k \in \mathscr{K}} \sum_{d \in \mathscr{H}} \sum_{h \in \mathscr{S} \cup \mathscr{O} \setminus \{i\}} d \cdot x_{ih}^{kd} \geq \Delta_{ij}^{min}, \forall i \in \mathscr{S}, \forall j \in \mathscr{S} \setminus \{i\} \tag{2.28}$$

Les contraintes (2.25) à (2.28) permettent de tenir compte des délais en heures et jours entre les services. Elles répondent aux situations de synchronisation, de précédence et de dépendance temporelle. Dans la figure 2.3, les services i et j appartenant à des tournées distinctes sont séparés par des délais minimaux et maximaux. Ainsi, selon la valeur de ces délais, ces services peuvent commencer au moment ou la date de début du service j peut être différée après celle du service i.

$$y_{ij}^d = 1 - y_{ji}^d, \forall \epsilon \in \mathscr{E}, \forall i \in \epsilon, \forall j \in \epsilon \setminus \{i\}, \forall d \in \mathscr{H}_i \cap \mathscr{H}_j \tag{2.29}$$

$$t_i + \pi_i \leq t_j + M \cdot (1 - y_{ij}^d), \forall \epsilon \in \mathscr{E}, \forall i \in \epsilon, \forall j \in \epsilon \setminus \{i\}, \forall d \in \mathscr{H}_i \cap \mathscr{H}_j \tag{2.30}$$



(a) Le service i est réalisé avant le service j (b) Le service j est réalisé avant le service i

FIGURE 2.4 – Exclusion

L'exclusion entre les services correspond à une situation où deux services ne doivent pas être réalisés en même temps. Les contraintes (2.29) et (2.30) établissent une précédence

temporelle d'un service par rapport à un autre. Ne sachant pas à l'avance lequel des deux services précédera l'autre, l'introduction de la variable de décision y^d_{ji} nous permet de construire la relation de précédence. La figure 2.4 illustre les deux situations possibles qui peuvent exister.

$$tend_{kd} - tstart_{kd} - \omega_{kd} - \sum_{i \in \mathscr{S} \cup \{\mathfrak{o}_k\}} \rho_k \cdot x^{kd}_{i\mathfrak{b}_{kd}} \leq ot^{jour}_k, \forall k \in \mathscr{K}, \forall d \in \mathscr{H} \tag{2.31}$$

$$\sum_{d \in w} (tend_{kd} - tstart_{kd} - \omega_{kd} - \sum_{i \in \mathscr{S} \cup \{\mathfrak{o}_k\}} \rho_k \cdot x^{kd}_{i\mathfrak{b}_{kd}}) \leq ot^{heb}_k, \forall k \in \mathscr{K}, \forall w \in \mathscr{W} \tag{2.32}$$

$$\sum_{k \in \mathscr{K}} \sum_{d \in \mathscr{H}} co_{kd}(tend_{kd} - tstart_{kd} - \omega_{kd} - \sum_{i \in \mathscr{S} \cup \{\mathfrak{o}_k\}} \rho_k \cdot x^{kd}_{i\mathfrak{b}_{kd}}) \leq oc \tag{2.33}$$

Les heures supplémentaires journalières et hebdomadaires sont comptées à l'aide des contraintes (2.31) et (2.32). La première contrainte ((2.31)) impose de ne pas dépasser les heures supplémentaires journalières. Évidemment, si une pause est prise dans la tournée d'un soignant, la durée de la pause n'est pas prise en compte. La deuxième contrainte ((2.32)) limite le nombre d'heures supplémentaires hebdomadaires. Ces dernières sont calculées en cumulant le nombre d'heures supplémentaires journalières sur l'ensemble des jours d'une semaine. Enfin, la contrainte (2.32) calcule le coût des heures supplémentaires effectuées par l'ensemble des tournées.

Contraintes de continuité de soins :

$$\sum_{k \in \overline{\mathscr{K}}_p} z^p_k + g^p \leq \zeta_p, \ \forall p \in \mathscr{P} \tag{2.34}$$

$$\frac{\sum_{i \in \mathscr{S}_p} \sum_{j \in \mathscr{S} \cup \{\mathfrak{d}_k\} \cup \{\mathfrak{b}_{kd}\} \setminus \{i\}} \sum_{d \in \mathscr{H}_k \cap \mathscr{H}_i \cap \mathscr{H}_j} x^{kd}_{ij}}{|\mathscr{S}_p|} \leq z^p_k, \ \forall k \in \mathscr{K}, \ \forall p \in \mathscr{P} \tag{2.35}$$

$$z^p_k \leq \sum_{i \in \mathscr{S}_p} \sum_{j \in \mathscr{S} \cup \{\mathfrak{d}_k\} \cup \{\mathfrak{b}_{kd}\} \setminus \{i\}} \sum_{d \in \mathscr{H}_k \cap \mathscr{H}_i \cap \mathscr{H}_j} x^{kd}_{ij}, \ \forall k \in \mathscr{K}, \forall p \in \mathscr{P} \tag{2.36}$$

$$\frac{\sum_{i \in \mathscr{S}_p} st_i}{|\mathscr{S}_p|} \leq g^p, \ \forall p \in \mathscr{P} \tag{2.37}$$

$$g^p \leq \sum_{i \in \mathscr{S}_p} st_i, \ \forall p \in \mathscr{P} \tag{2.38}$$

Enfin, les contraintes (2.34), (2.35), (2.36) et (2.37) organisent la continuité des soins de chaque patient. En effet, la contrainte (2.34) compte le nombre de soignants différents n'ayant pas encore soigné un patient ; le soignant fictif y est inclus s'il prodigue des soins à ce patient. Ce nombre est restreint à une quantité ζ_p et est calculé grâce aux contraintes

(2.35) à (2.38). Les contraintes (2.35) et (2.36) permettent de déterminer si un soignant a prodigué des soins à un patient et fixe donc la valeur de la variable z_k^p. Le terme de droite de la contrainte (2.36) compte le nombre de services réalisés par un soignant k pour un patient p. Ce terme est également le numérateur de la fraction de gauche de la contrainte (2.35). Si ce terme nul, le soignant k ne réalise aucun des services que nécessite le patient p. Donc, la valeur de la variable z_k^p est également nulle. Par contre, si le soignant k réalise au moins un des services que nécessite le patient p, la fraction de gauche de la contrainte (2.35) a une valeur strictement supérieur à 0, ce qui force la valeur de la variable z_k^p à 1. Les contraintes (2.37) et (2.38) permettent de déterminer si le soignant fictif à prodiguer au moins un service que nécessite un patient donné.

Contraintes d'intégrité :

$$x_{ij}^{kd} \in \{0,1\}, \forall i \in \mathscr{V}, \forall j \in \mathscr{V}, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}$$

$$st_i \in \{0,1\}, \forall i \in \mathscr{S}$$

$$y_{ij}^{d} \in \{0,1\}, \forall i \in \mathscr{S}, \forall j \in \mathscr{S}, \forall d \in \mathscr{H}$$

$$u_{id}^{n} \in \{0,1\}, \forall i \in \mathscr{S}, \forall d \in \mathscr{H}, \forall n \in \{1,\ldots,\gamma_{id}\}$$

$$t_i \geq 0, \forall i \in \mathscr{S}$$

$$oc \geq 0$$

$$tard_i \geq 0, \forall i \in \mathscr{S}$$

$$tard^{max} \geq 0$$

$$tbreak_{kd} \geq 0, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}$$

$$tstart_{kd} \geq 0, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}$$

$$tend_{kd} \geq 0, \forall k \in \mathscr{K}, \forall d \in \mathscr{H}$$

$$z_k^p \in \{0,1\}, \forall p \in \mathscr{P}, \forall k \in \overline{\mathscr{K}}_p$$

$$g^p \in \{0,1\}, \forall p \in \mathscr{P}$$

Les contraintes d'intégrité définissent d'une part le domaine de définition de chaque variable et d'autre part l'intervalle de variation de chaque indice.

Afin de conserver la linéarité du modèle et tout en modélisant les situations de disjonction, nous avons introduit une constante M. Cette dernière est égale à la valeur maximale que peut prendre les variables $tstart_{kd}$. Cette valeur peut être affinée pour chaque contrainte afin d'améliorer la résolution de ce programme linéaire à variables mixtes à l'aide d'un solveur mathématique.

2.3.3 Fonctions économiques

Afin de rester le plus générique possible, plusieurs critères d'évaluation d'une solution ont été définis :

Coût de sous-traitance : le coût de sous-traitance représente les frais des services qui ne sont pas réalisés au sein de la structure de soins à domicile. Cette externalisation peut intervenir dans une situation d'absence de qualification pour réaliser ces services, de contraintes difficiles à satisfaire ou encore d'une charge de travail trop importante au regard des ressources humaines disponibles.

$$Minimiser \left(\sum_{i \in \mathscr{S}} co_i \cdot st_i \right) \tag{2.39}$$

Somme des retards de chaque service : la somme des retards correspond au cumul du temps écoulé entre la date de début au plus tard du service souhaité par le patient et la date de début effective du service pour les services en retard uniquement.

$$Minimiser \left(\sum_{i \in \mathscr{S}} tard_i \right) \tag{2.40}$$

Retard maximal : le retard maximal représente le plus grand retard associé à un seul service parmi l'ensemble de tous les services.

$$Minimiser \left(tard^{max} \right) \tag{2.41}$$

Temps total de trajet : le temps total de trajet représente le cumul du temps de trajet de l'ensemble des tournées nécessaires pour la réalisation des services.

$$Minimiser \left(\sum_{i \in \mathscr{V}} \sum_{j \in \mathscr{V} \setminus \{i\}} \tau_{ij} \sum_{k \in \mathscr{K}} \sum_{d \in \mathscr{H}_k} x_{ij}^{kd} \right) \tag{2.42}$$

Coût total des heures supplémentaires : le coût total des heures supplémentaires correspond au produit du coût unitaire des heures supplémentaires effectuées (dépendant du soignant et du jour) et du nombre total d'heures supplémentaires effectuées par les soignants de la structure de soins à domicile. Ce coût peut également permettre de calculer le nombre total d'heures supplémentaires effectuées si le coût unitaire des heures supplémentaires de tous les soignants est unitaire.

$$Minimiser \left(oc \right) \tag{2.43}$$

Continuité de soins : il s'agit minimiser le nombre de nouveaux soignants qu'un patient est amené à rencontrer tout au long de l'horizon de planification.

$$Minimiser \left(\sum_{p \in \mathscr{P}} g^p + \sum_{k \in \overline{\mathscr{K}}_p} z_k^p \right) \tag{2.44}$$

Coût de réalisation des services : le coût total de réalisation de l'ensemble des services hormis ceux sous-traités. Ce coût tient compte des spécificités contractuelles des soignants, du type de services et du jour de réalisation des services.

$$Minimiser \left(\sum_{k \in \mathscr{K}} \sum_{i \in \mathscr{S}} \sum_{d \in \mathscr{H}_k \cap \mathscr{H}_i} cp_{ikd} \sum_{j \in \mathscr{V} \setminus \{i\}} x_{ij}^{kd} \right) \tag{2.45}$$

Cette fonction économique peut également être vue comme un profit à maximiser si la sémantique associée à la constante cp_{ikd} change. Par exemple, on peut estimer que cette dernière correspond aux gains obtenus en réalisant un service. Cela a du sens si la sous-traitance est autorisée. Ainsi, le gain obtenu en sous-traitant devra être plus faible.

Les fonctions économiques ci-dessus peuvent être agrégées au sein d'une même fonction économique à l'aide d'une somme pondérée comme cela a été fait la plupart du temps dans la littérature. Cependant, les fonctions n'étant pas de même nature, il n'est pas aisé de déterminer de manière adéquate le poids associé à chaque fonction économique. Une méthode exacte de résolution multi-critère peut être également être envisagée comme ϵ-contrainte qui énumère l'ensemble du front de Pareto ou une méthode hiérarchisant les fonctions économiques comme l'approche lexicographique. Enfin, des méthodes approchées comme NSGA-II [Deb *et al.*, 2002] permettraient d'approximer le front de Pareto dans des temps de calcul raisonnable.

2.4 Limitations du modèle

D'abord, le modèle générique proposé est un modèle statique et déterministe. Les données sont supposées connues à l'avance et leurs valeurs ne varient pas au cours du temps. Par conséquent, le modèle obtenu peut être éloigné de la réalité où l'évolution des données reste dynamique. Cependant, il peut être facilement adaptable dans le cas dynamique en utilisant un horizon de planification glissant. La planification est reconstruite dès qu'un événement survient : l'arrivée d'un nouveau patient, l'annulation d'une consultation, etc.

Ensuite, en ce qui concerne la formulation, le modèle générique proposé ci-dessus n'ayant pas été formulé dans un souci d'efficacité, le nombre élevé de variables du modèle nuit à une résolution efficace à l'aide d'un solveur de programmation mathématique. De plus, l'utilisation d'une constante M dans la formulation pour prendre en compte les situations de disjonction tout en gardant la linéarité du modèle réduit l'efficacité de résolution. La valeur de la variable oc et celle de $tard^{max}$ ne sont pas calculées si les critères 2.41 et 2.43 ne sont pas minimisés.

Enfin, certains travaux [Rasmussen *et al.*, 2012] ont tenu compte de la dimension multimodale du réseau de transport multimodal, c'est-à-dire l'utilisation de différents moyens de transport par le personnel de soins. Dans un environnement urbain, la multi-modalité est un élément intéressant à prendre en compte avec l'existence de transport en commun et de transport privatif.

2.5 Génération d'instances

Dans le but de valider les approches proposées dans cette thèse, plusieurs instances construites à partir du modèle ci-dessus ont été générées. Ces instances nous permettent de conduire des expérimentations numériques afin de d'évaluer le modèle proposé. Elles seront largement utilisées afin d'identifier les caractéristiques complexifiant la résolution du problème, de distinguer les instances faciles à résoudre de celles qui ne le sont pas, de comparer les méthodes de résolution entre elles. Dans les lignes qui suivent, le processus de génération des données est décrit en détail. Nous commençons d'abord par définir les différents paramètres nécessaires à la génération des données et la manière dont ces données ont été générées. Ensuite, les différentes classes d'instances sont présentées avec leurs spécificités.

2.5.1 Paramétrage et données

2.5.1.1 Paramètres

Les paramètres qui suivent doivent être définies par le décideur pour moduler la taille des instances générées et de les configurer. L'utilité de chaque paramètre se révélera tout au long de la description du processus de génération des données. La valeur de tous ces paramètres a été fixée de manière à obtenir des instances difficile à résoudre et se rapprochant le plus possible de la réalité. La liste des paramètres est la suivante :

- l'horizon de planification
- le nombre de soignants disponibles dans la structure de soins à domicile, soit $|\mathscr{K}|$.
- un coefficient permettant d'augmenter le rapport entre le nombre de soignants et le nombre de services par jour, soit α. Il correspond au nombre moyen de services qu'un soignant peut délivrer chaque jour. Le nombre de total de services est calculé à partir de la formule suivante : $\alpha \cdot |\mathscr{H}| \cdot |\mathscr{K}|$. Ce paramètre permet de contrôler le rapport entre la charge de travail et les ressources disponibles et indirectement la difficulté de l'instance.
- un nombre de dépôts à partir desquels les soignants commencent et finissent leurs tournées. Ce nombre devra être supérieur ou égal à 1.
- un nombre de patients nécessitant des services périodiques, noté β
- pourcentage de patients parmi les β, qui nécessitent des services à délivrer tous les jours, noté a_1.
- pourcentage de patients parmi les β, qui nécessitent des services à délivrer tous les 2 jours, noté a_2.
- pourcentage de patients parmi les β, qui nécessitent des services à délivrer tous les 4 jours, noté a_3.
- pourcentage de patients parmi les β, qui nécessitent des services à délivrer toutes les semaines, noté a_4.
- le nombre maximum de services non périodiques que l'on pourra attribuer à un patient réclamant des services périodiques, m_1
- le nombre maximum de services non périodiques que l'on pourra attribuer à un patient ne réclamant que des services non périodiques, m_2

2.5.1.2 Génération des données concernant les patients et les services

Le nombre total de patients est déterminé de manière dynamique lors de la création des services. On procède comme suit en deux étapes en distinguant les patients nécessitant des services périodiques et ceux qui n'en nécessitent pas.

Une première étape consiste à créer les β patients nécessitant des services périodiques. Les soins administrés à ces patients suivent une régularité. Nous créons successivement les $a_i \cdot \beta$ patients nécessitant des services à délivrer selon la régularité imposée, $1 \leq i \leq 4$. Pour chaque patient nécessitant des services périodiques, autant de services sont créés qu'il y a de jours pour lesquels un service doit être réalisé sur l'horizon de planification. Par exemple, pour un horizon de planification d'une semaine, si le patient nécessite des soins tous les jours, sept services sont créés. Le jour pour lequel le premier service est assigné est choisi aléatoirement entre 1 et la fréquence en jours des soins. Pour chaque patient créé, on complète la liste de ses services périodiques, en lui affectant un nombre de services non périodiques tiré au hasard entre 0 et m_1.

La deuxième étape consiste à créer les patients nécessitant des services non périodiques. Les patients sont créés au fur et à mesure tout en leur affectant des services non périodiques. Tant que le nombre total de services à créer n'est pas atteint, un nouveau patient est créé et un nombre de services tiré au hasard entre 1 et m_2 lui est affecté. Le jour est choisi de manière aléatoire. Il est à noter que pour un horizon d'une seule journée, seuls des patients nécessitant des services non périodiques sont créés.

2.5.1.3 Données concernant la cartographie et le réseau routier

Toutes les coordonnées géographiques sont calculées sur une carte de taille 100×100 par défaut. Toutes les durées sont en minutes. La durée τ_{ij} séparant deux sommets quelconques du réseau routier est déterminée en utilisant la distance euclidienne.

2.5.2 Données concernant les services

La durée de réalisation de chaque service est choisie aléatoirement dans l'ensemble suivant : $\{10, 15, 20, 25, 30, 40, 50, 60\}$. Les services périodiques ne peuvent être réalisés que pendant un jour donné de l'horizon de planification. Ce jour est choisi aléatoirement. Concernant les services non périodiques, les jours d'indisponibilité pour les réaliser sont tirés aléatoirement et constituent entre 0 et 50% de l'horizon de planification sauf dans le cas d'un horizon d'une seule journée.

Le nombre de fenêtres de temps d'un service pour une même journée varie aléatoirement entre 1 et 4 et ces fenêtres ont une largeur de 60, 120, 180 et 240 min. Toutes ces fenêtres de temps sont disjointes et comprises entre 6 : 00 le matin et 20 : 00 le soir, par pas de 15 min pour l'heure de début.

Le coût de réalisation d'un service i le jour d par le soignant k est choisi aléatoirement dans l'intervalle $[10, 50]$.

Le coût de sous-traitance est plus élevé que le coût de réalisation d'un service par la structure de soins à domicile. Ce coût est choisi aléatoirement dans l'intervalle $[100, 200]$.

2.5.2.1 Données concernant les soignants

Les coordonnées des sommets de départ, d'arrivée et de pause du soignant sont choisies aléatoirement sur la carte. Le nombre total de sommets créés doit être inférieur ou égal au nombre de dépôts défini comme paramètre.

Afin de déterminer la fenêtre de temps du début de la tournée du soignant k le jour d, deux cas ont été envisagés :

— horaire fixe : l'heure de début de tournée est fixée, soit à 6 : 00, soit à 13 : 00, de manière à avoir autant de personnes disponibles le matin que l'après-midi.

— horaire variable : l'heure de début de tournée doit être déterminée et comprise entre 6 : 00 et 13 : 00.

Dans les deux cas, la valeur du temps de travail maximale pour le soignant k le jour d est tiré aléatoirement dans l'ensemble $\{360, 420, 480\}$. La durée de la pause pour chaque soignant est de 60 minutes. La durée de travail minimale (resp. maximale) sans pause pour le soignant k est égale à 180 (resp. 360). Le coût unitaire des heures supplémentaires d'un soignant k le jour d est choisi aléatoirement dans l'ensemble $\{20, 21, 22, \ldots, 100\}$. Enfin, les heures supplémentaires maximales par jour (resp. par semaine) prennent leurs valeurs dans l'ensemble $\{60, 120\}$ (resp. $\{300, 360, \cdots, 600\}$).

Dans le cas d'un horizon temporel de plusieurs jours, chaque soignant possède 2 jours successifs d'indisponibilité hebdomadaires qui sont identiques chaque semaine. Ces indisponibilités sont réparties équitablement en fonction des jours pour l'ensemble des soignants afin d'éviter une carence en soignant un jour donné. Enfin, afin de déterminer l'aptitude d'un soignant k à réaliser un service i donné, 75% des soignants sont tirés aléatoirement parmi l'ensemble des soignants pouvant traiter le service tel qu'il existe un jour $d \in \mathscr{H}_i \cap \mathscr{H}_k$.

2.5.2.2 Données concernant les patients

Les coordonnées géographiques de chaque patient sont déterminées aléatoirement sur la carte.

Pour déterminer le nombre de soignants n'étant pas intervenus chez le patient p mais pouvant le faire, la procédure suivante est appliquée. Soit $\mathscr{K}'_p$, l'ensemble des soignants pouvant réaliser au moins un des services attribués à p. Nous déterminons $\mathscr{K}_p$, le plus petit sous-ensemble de $\mathscr{K}'_p$ de soignants couvrant toutes les aptitudes nécessaires pour réaliser l'ensemble des services de p. $\mathscr{K}_p$ devient ainsi l'ensemble des soignants ayant déjà soigné le patient p et $|\overline{\mathscr{K}}_p|$ est l'ensemble $\mathscr{K}'_p \setminus \mathscr{K}_p$. La valeur ζ_p est égale à la moitié de $|\overline{\mathscr{K}}_p|$.

2.5.2.3 Mise en place de l'exclusion entre services

L'exclusion entre les services ne concerne que les services d'un même patient. Le nombre total de services ne devant pas être réalisés en même temps est égal à 15% du nombre de services. Afin de générer l'ensemble des services en exclusion, un patient nécessitant au moins 2 services non périodiques est sélectionné aléatoirement. Pour chacun de ces patients, 2 ou 3 de ces services ne devront pas être réalisés en même temps. La procédure est itérée tant que le pourcentage de services en exclusion n'est pas atteint.

2.5.2.4 Mise en place de la synchronisation (délais min/max) entre services

Deux cas sont à distinguer :

— le premier cas concerne la simultanéité. le processus est similaire à l'exclusion : un patient disposant d'au moins 2 services non périodiques est choisi aléatoirement, 2 ou 3 services de ce patient sont synchronisés de sorte à ce qu'ils doivent commencer en même temps. Cette procédure est itérée de manière à obtenir $0.15|\mathscr{S}|$ services synchronisés.

— pour le cas d'un délai non nul, 10% des services seront concernés par des délais en heures (5%) ou en jours (5%). Afin d'imposer des délais entre une paire de services, deux services i et j non périodiques d'un même patient sont choisis aléatoirement tels que : $\mathscr{H}_i \cap \mathscr{H}_j \neq \emptyset$ et que ces services ne soient pas soumis à une contrainte d'exclusion ou de simultanéité. Les délais en heures sont choisis aléatoirement dans l'ensemble $\{30, 60, 90\}$. Les délais en jours sont choisis aléatoirement entre 1 et la taille de l'horizon en garantissant qu'il existe au moins deux jours $d \in \mathscr{H}_i$ et $d' \in \mathscr{H}_j$ tels que le délai soit respecté.

2.5.3 Classes d'instances et instances de la littérature

Classes d'instances. Différentes classes d'instances sont proposées afin de couvrir la plupart des spécificités du problème. Elles peuvent être regroupées en deux catégories : les instances d'un jour et les instances de plusieurs jours. Pour chacune des catégories, un sous-ensemble de caractéristiques a été pris en compte pour créer une diversité parmi les instances.

Classe	1	2	3	4	5	6
Horizon	un jour	un jour	plusieurs jours	plusieurs jours	plusieurs jours	plusieurs jours
Continuité des soins	✗	✗	✗	✓	✓	✓
Sous-traitance	✓	✗	✓	✗	✓	✓
Dépendances temporelles	synchro exclusion délais	synchro exclusion	✗	délais jour	synchro exclusion délais jour/heure	✗
Disponibilité des services	FT_1	FT_s	FT_1	FT_1	FT_1	FT_s
Disponibilité du personnel	horaire fixe	horaire variable	horaire fixe	horaire fixe	horaire fixe	horaire variable

FT_1 : une seule fenêtre temporelle
FT_s : plusieurs fenêtres temporelles
horaire fixe : les dates de début des tournées des soignants sont fixées
horaire variable : les dates de début des tournées des soignants ne sont pas fixées

synchro : existence de services dont les dates de début doivent être identiques
exclusion : existence de services ne devant pas se chevaucher
délais : existence de services séparés d'une durée minimale et/ou maximale

Pour chaque classe d'instances, le nombre de soignants des instances générées prendra sa valeur dans l'ensemble $\{2, 5, 10, 15, 20\}$. Le ratio journalier entre les soignants et les services est de 1/8. Enfin, 25 instances sont générées pour chaque classe.

Instances de la littérature. Notre démarche nous a conduit à reprendre des instances de la littérature et à les convertir dans le format que nous avons créé pour la génération. Malheureusement, le format d'origine n'étant pas toujours documenté, toutes les instances obtenues n'ont pas été converties dans le format proposé. C'est le cas des instances de [Bard *et al.*, 2014] et [Liu *et al.*, 2013]. Les instances de la littérature que nous avons traduites sont celles de [Bredström et Rönnqvist, 2008], [Mankowska *et al.*, 2014] et [Rasmussen *et al.*, 2012]. Cependant, dans le cas [Rasmussen *et al.*, 2012], la multi-modalité ne nous permet pas de les utiliser puisqu'elle n'est pas prise en compte dans la formulation proposée.

2.6 Expérimentations numériques

Dans le souci de valider notre approche, des expérimentations numériques ont été réalisées sur les instances de la littérature. Ces expérimentations ont été conduites sur un ordinateur ayant la configuration suivante : Intel Xeon CPU E5-2620, 2.00GHz avec 16 Go de mémoire vive. Le modèle mathématique présenté ci-dessus a été implémenté en C++ à l'aide de la librairie Concert IBM ILOG CPLEX Optimization Studio 12.5.1. Pour les instances de la littérature, nous avons scrupuleusement respecté le protocole proposé par les auteurs afin de reproduire le plus fidèlement possible les résultats.

[Bredström et Rönnqvist, 2008] propose 30 instances où il s'agit d'optimiser séparément le temps total de trajet et les préférences. Ces instances représentent un VRPTW avec synchronisation. Une somme pondérée est mise en place et un poids unitaire est affecté à la fonction économique à optimiser. Les tableaux 2.2 et 2.3 représentent les résultats obtenus pour chaque fonction économique. Le temps de calcul maximum imparti est de 3600 secondes. Ces instances sont réparties en trois groupes distincts :

— 15 instances de petite taille composées de 4 soignants, 18 patients et 20 services
— 9 instances de taille moyenne composées de 10 soignants, 45 patients et 50 services
— 6 instances de grande taille composées de 16 soignants, 72 patients et 80 services

Comme nous pouvons le constater, la programmation mathématique résout toutes les instances de petite taille dans le temps imparti quelque soit la fonction économique. Malheureusement, comme nous pouvions nous y attendre, la programmation mathématique n'est plus adaptée pour des instances de taille plus importante. En effet, pour les instances de taille moyenne, quelques solutions réalisables sont obtenues sans pour autant prouver leur optimalité. Pour les instances de grande taille, aucune solution réalisable n'est obtenue. Ces résultats corroborent ceux obtenus dans [Bredström et Rönnqvist, 2008] quant à l'utilisation de la programmation mathématique pour ces instances.

Instance	Horizon	Soignants	Services	Patients	Temps (s)	Durée	Durée normalisée
01L	1	4	20	18	7.10	**262**	**3.39**
01M	1	4	20	18	1.60	**274**	**3.55**
01S	1	4	20	18	0.59	**274**	**3.55**
02L	1	4	20	18	21.68	**277**	**3.42**
02M	1	4	20	18	1.22	**290**	**3.58**
02S	1	4	20	18	0.19	**346**	**4.27**
03L	1	4	20	18	9.27	**257**	**3.29**
03M	1	4	20	18	1.64	**260**	**3.33**
03S	1	4	20	18	0.61	**283**	**3.63**
04L	1	4	20	18	2895.71	**316**	**5.13**
04M	1	4	20	18	22.56	**349**	**5.67**
04S	1	4	20	18	1.16	**378**	**6.14**
05L	1	4	20	18	1.74	**266**	**3.34**
05M	1	4	20	18	0.49	**281**	**3.53**
05S	1	4	20	18	0.45	**313**	**3.93**
06L	1	10	50	45	3656.58	-	-
06M	1	10	50	45	3707.58	605	8.10
06S	1	10	50	45	3670.13	608	8.14
07L	1	10	50	45	3605.11	-	-
07M	1	10	50	45	3603.96	-	-
07S	1	10	50	45	3647.23	607	8.39
08L	1	10	50	45	3604.41	-	-
08M	1	10	50	45	3601.67	-	-
08S	1	10	50	45	3602.30	887	11.34
09L	1	16	80	72	3616.12	-	-
09M	1	16	80	72	3616.31	-	-
09S	1	16	80	72	3600.30	-	-
10L	1	16	80	72	3657.72	-	-
10M	1	16	80	72	3618.48	-	-
10S	1	16	80	72	3613.17	-	-

TABLE 2.2 – Résultats pour les instances de [Bredström et Rönnqvist, 2008] en optimisant le temps total de trajet

Instance	Horizon	Soignants	Services	Patients	Temps (s)	Préference
01S	1	4	20	18	0.27	**-114.03**
01M	1	4	20	18	0.33	**-117.80**
01L	1	4	20	18	0.73	**-118.51**
02S	1	4	20	18	0.20	**-92.09**
02M	1	4	20	18	4.93	**-104.81**
02L	1	4	20	18	141.25	**-107.64**
03S	1	4	20	18	0.36	**-99.49**
03M	1	4	20	18	0.69	**-106.59**
03L	1	4	20	18	1.44	**-107.87**
04S	1	4	20	18	0.35	**-100.00**
04M	1	4	20	18	7.28	**-106.72**
04L	1	4	20	18	178.08	**-109.27**
05S	1	4	20	18	0.11	**-76.29**
05M	1	4	20	18	0.42	**-76.29**
05L	1	4	20	18	3.20	**-84.21**
06S	1	10	50	45	362.64	**-370.06**
06M	1	10	50	45	3601.08	-379.88
06L	1	10	50	45	3600.75	-
07S	1	10	50	45	813.57	**-401.11**
07M	1	10	50	45	3600.91	-
07L	1	10	50	45	3600.82	-
08S	1	10	50	45	3600.99	-351.99
08M	1	10	50	45	3600.88	-
08L	1	10	50	45	3600.88	-
09S	1	16	80	72	3600.32	-
09M	1	16	80	72	3600.82	-
09L	1	16	80	72	3601.77	-
10S	1	16	80	72	3601.49	-
10M	1	16	80	72	3602.20	-
10L	1	16	80	72	3600.11	-

TABLE 2.3 – Résultats pour les instances de [Bredström et Rönnqvist, 2008] en optimisant les préférences

[Mankowska *et al.*, 2014] propose 70 instances où il s'agit de minimiser une somme pondérée des fonctions économiques suivantes : la distance totale parcourue, la somme des retards des services et le retard maximal. Ces fonctions économiques ont le même poids ($\frac{1}{3}$) dans la somme pondérée. Le temps maximal imparti pour résoudre chaque instance est de 10 heures. Ces instances sont caractérisées par une fenêtre de temps pour chaque patient, des délais minimaux et maximaux séparant certains services et la contrainte de qualification des soignants pour réaliser les services. 7 groupes d'instances peuvent être aisément identifiés en fonction de la taille des instances. Les tableaux 2.4 et 2.5 récapitulent les résultats obtenus pour chaque groupe d'instances. Pour le premier groupe d'instances (A01-A10), la programmation mathématique résout et prouve l'optimalité des instances en à peine

une seconde. Pour le deuxième groupe d'instances (B01-B10), les temps de calcul sont plus importants. La programmation mathématique retourne une solution réalisable pour toutes les instances mais prouve l'optimalité de 5 d'entre elles dans le temps imparti. En ce qui concerne le troisième groupe d'instances (C01-C10), la programmation mathématique détermine une solution réalisable pour 3 instances sur 5 instances sans pour autant prouver leur optimalité. Enfin, pour les 4 groupes restants (D01-D10, E01-E10, F01-F10, G01-G10), comme nous pouvions nous y attendre, la programmation mathématique n'est pas adaptée pour les résoudre. En effet, à cause de la la taille des instances, le nombre de variables générées est trop important pour espérer obtenir ne serait-ce qu'une solution réalisable.

Instance	Horizon	Soignants	Services	Patients	Temps (s)	Objectif
A01	1	3	13	10	0.23	**218.20**
A02	1	3	13	10	0.25	**246.62**
A03	1	3	13	10	0.32	**305.86**
A04	1	3	13	10	0.92	**186.90**
A05	1	3	13	10	0.35	**189.54**
A06	1	3	13	10	0.26	**200.10**
A07	1	3	13	10	0.23	**225.37**
A08	1	3	13	10	0.18	**232.05**
A09	1	3	13	10	1.70	**222.29**
A10	1	3	13	10	0.05	**225.01**
B01	1	5	33	25	12711.60	**428.10**
B02	1	5	33	25	5944.16	**476.05**
B03	1	5	33	25	22530.20	**399.09**
B04	1	5	33	25	36001.90	411.30
B05	1	5	33	25	13114.60	**366.34**
B06	1	5	33	25	36000.70	440.93
B07	1	5	33	25	442.95	**328.67**
B08	1	5	33	25	575.17	**357.68**
B09	1	5	33	25	36001.10	402.67
B10	1	5	33	25	36001.40	469.58
C01	1	10	65	50	36000.90	-
C02	1	10	65	50	36000.70	-
C03	1	10	65	50	36029.40	547.54
C04	1	10	65	50	36013.40	-
C05	1	10	65	50	36004.00	691.56
C06	1	10	65	50	36001.20	-
C07	1	10	65	50	36002.00	-
C08	1	10	65	50	36005.50	469.49
C09	1	10	65	50	36001.10	-
C10	1	10	65	50	36001.50	-

TABLE 2.4 – Résultats pour les instances de [Mankowska *et al.*, 2014] en optimisation une somme pondérée des critères suivants : la distance totale parcourue, le retard maximal et la somme des retards. Le poids de chaque critère est $\frac{1}{3}$

Instance	Horizon	Soignants	Services	Patients	Temps (s)	Objectif
D01	1	15	98	75	36004.30	-
D02	1	15	98	75	36003.10	-
D03	1	15	98	75	36002.30	-
D04	1	15	98	75	36010.50	-
D05	1	15	98	75	36013.90	-
D06	1	15	98	75	36010.00	-
D07	1	15	98	75	36009.80	-
D08	1	15	98	75	36007.10	-
D09	1	15	98	75	36006.40	-
D10	1	15	98	75	36002.70	-
E01	1	20	130	100	36025.30	-
E02	1	20	130	100	36180.20	-
E03	1	20	130	100	36104.70	-
E04	1	20	130	100	36360.30	-
E05	1	20	130	100	36104.20	-
E06	1	20	130	100	36159.30	-
E07	1	20	130	100	36130.20	-
E08	1	20	130	100	36162.20	-
E09	1	20	130	100	36190.80	-
E10	1	20	130	100	36008.50	-
F01	1	30	260	200	36241.80	-
F02	1	30	260	200	36000.00	-
F03	1	30	260	200	36000.00	-
F04	1	30	260	200	36000.00	-
F05	1	30	260	200	36000.00	-
F06	1	30	260	200	36000.00	-
F07	1	30	260	200	36000.00	-
F08	1	30	260	200	36000.00	-
F09	1	30	260	200	36000.00	-
F10	1	30	260	200	36000.00	-
G01	1	40	400	300	36000.00	-
G02	1	40	400	300	36000.00	-
G03	1	40	400	300	36000.00	-
G04	1	40	400	300	36000.00	-
G05	1	40	400	300	36000.00	-
G06	1	40	400	300	36000.00	-
G07	1	40	400	300	36000.00	-
G08	1	40	400	300	36000.00	-
G09	1	40	400	300	36000.00	-
G10	1	40	400	300	36000.00	-

TABLE 2.5 – Résultats pour les instances de [Mankowska *et al.*, 2014] en optimisation une somme pondérée des critères suivants : la distance totale parcourue, le retard maximal et la somme des retards. Le poids de chaque critère est $\frac{1}{3}$ (suite)

2.7 Conclusion

À partir de la revue de la littérature établie au chapitre précédent, un modèle statique et déterministe a été proposé dans ce chapitre. Ce modèle est générique et prend en compte la plupart des caractéristiques rencontrées dans la littérature. Ce modèle comble un manque à savoir l'absence de modèle suffisamment général couvrant toute la diversité des structures de soins à domicile. Ce modèle constitue également un socle pour la construction de méthodes de résolution exactes et approchées. Les limites de notre approche ont été soulignées en mettant en avant le fait que les aspects dynamique et stochastique ont été négligés. Plusieurs classes d'instances ont été générées en combinant diverses spécificités du problème grâce à un processus de génération et un paramétrage décrit en amont. En plus des instances générées, les instances de la littérature ont été obtenues et converties dans le même format. Des expérimentations numériques ont été conduites pour tester la résolution du modèle à l'aide du solveur CPLEX.

Chapitre 3

Encodage et décodage d'une solution

Dans le chapitre précédent, une formulation générique du HCRSP a été proposée en prenant en compte la plupart des contraintes et des fonctions économiques rencontrées dans la littérature. Des instances de taille raisonnable ont été résolues à l'aide d'un solveur de programmation mathématique. Ce chapitre est dédié à l'élaboration de la représentation d'une solution et de son évaluation. À partir de ce socle, des méta-heuristiques sont proposées dans les chapitres qui suivent. Ce chapitre est organisé comme suit. Nous soulignons les contributions de ce chapitre dans la section 3.1. La section 3.2 est consacrée aux notations sur lesquelles les sections suivantes s'appuieront. La section 3.3 décrit un encodage indirect d'une solution au problème générique et l'algorithme de décodage correspondant. La section 3.4 présente un encodage direct et l'algorithme de décodage associé. Enfin, la section 3.5 est consacrée à des expérimentations numériques pour valider notre approche. Nous conclurons en récapitulant nos contributions dans cette section et en soulignant les perspectives qui s'offrent à nous.

3.1 Contributions

Les contributions de ce chapitre s'inscrivent dans la conception d'une méthode de résolution approchée du problème. Dans ce cadre, la représentation d'une solution et son évaluation y tiennent une place importante, tant par la qualité des solutions retournées que par le temps de résolution. Nos contributions sont les suivantes :

- l'adaptation de deux représentations d'une solution issues de la littérature au HCRSP
- l'extension d'algorithmes d'évaluation également issus de la littérature pour chaque représentation
- des expérimentations numériques testant différentes stratégies.

3.2 Préliminaires : notations

En plus de la notation introduite au chapitre précédent, cette section rassemble des notations utilisées dans les sections suivantes.

Notations.

σ *: séquence de tous les services* $\mathscr{S}$
σ^k *: sous-séquence de services assignés au soignant* k
$|\cdot|$ *: longueur de* $\cdot$
σ_i^k *: i-ème service de la sous-séquence* σ^k
$delay(i,j)$ *:* $\top$ *si* $\exists\ \delta_{ij}^{min} \geq 0 \vee \exists \delta_{ij}^{max} \geq 0$ *;* $\bot$ *sinon.*
$synchro(i,j)$ *:* $\top$ *si* $\exists\ \delta_{ij}^{max} = 0 \vee \exists\ \delta_{ji}^{max} = 0$ *;* $\bot$ *sinon*
$exclusion(i,j)$ *:* $\top$ *si* $\exists\ \epsilon \in \mathscr{E} \mid i,j \in \epsilon$ *;* $\bot$ *sinon*
$route(i,j)$ *:* $\top$ *si les services* i *et* j *appartiennent à la même tournée ;* $\bot$ *sinon*
$prec(i,j)$ *:* $\top$ *si* $route(i,j)$ *et le service* i *précède le service* j *dans la tournée ;* $\bot$ *sinon*

$delay(i,j)$, $synchro(i,j)$ et $exclusion(i,j)$ rendent compte respectivement de l'existence d'une contrainte de délai, de synchronisation et d'exclusion liant les services i et j.

3.3 Représentation indirecte d'une solution

Dans cette section, une représentation indirecte d'une solution du HCRSP est introduite en s'inspirant d'une représentation indirecte proposée dans la littérature des problèmes de tournées de véhicules. À partir de cette représentation, un algorithme de décodage est présenté afin d'évaluer la solution. Plusieurs améliorations sont introduites pour diminuer le temps de calcul de l'algorithme.

3.3.1 Encodage

Une solution ς du HCRSP est caractérisée par un ensemble de tournées $\mathscr{K} \times \mathscr{H}$. Chaque tournée (k,d) est réalisée par un soignant $k \in \mathscr{K}$ le jour $d \in \mathscr{H}$ de l'horizon de planification et se caractérise par une séquence de services. Les services sous-traités seront affectés au soignant fictif k_f. L'encodage indirect d'une solution est une permutation σ de tous les services à planifier. La permutation σ peut être considérée comme une tournée géante réalisant l'ensemble des services à planifier. La figure 3.1 illustre la représentation d'une solution du HCRSP pour un horizon de planification de deux jours. L'exemple en question porte sur sept patients $(p_1, \ldots, p_7)$ nécessitant en tout neuf services. Le patient p_7 nécessite la synchronisation des services σ_3 et σ_5 et le patient p_5 nécessite les services σ_1 et σ_7, un service par jour. L'encodage proposé est inspiré de celui introduit par [Prins, 2004] pour lequel une solution est représentée par une permutation de clients. Cette représentation est utilisée dans de nombreuses heuristiques dédiées au problème de tournées de véhicules.

L'encodage proposé étant une permutation de tous les services requis, un même patient p nécessitant n_p services sera présent dans la permutation n_p fois. Cette représentation devient une simple permutation de patients si chaque patient nécessite un unique service. Cette représentation prend en compte la périodicité du problème. Ainsi, deux décisions sont prises dans une tournée : le soignant qui la réalise et le jour pour lequel elle est réalisée. Notons que cet encodage diffère de ceux prenant en compte la périodicité du problème comme [Lacomme *et al.*, 2005]. En effet, ce dernier encode une solution par plusieurs tournées géantes, une par jour.

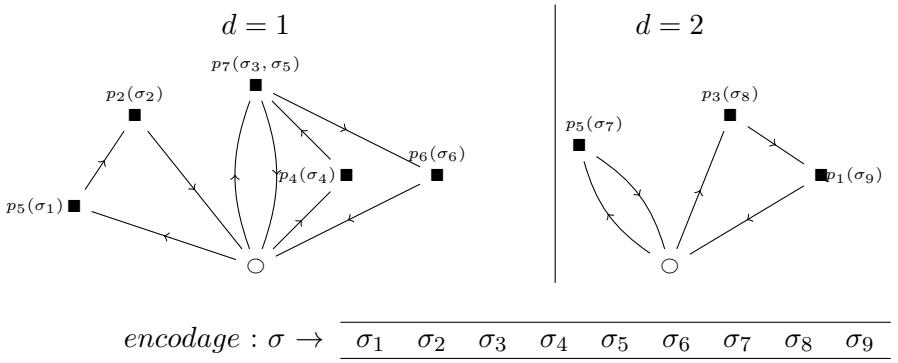


FIGURE 3.1 – Exemple d'une solution du HCRSP et de son encodage indirect

3.3.2 Décodage d'une solution

3.3.2.1 Algorithme Split pour HCRSP

L'algorithme Split introduit par [Prins, 2004] est un algorithme de plus court chemin appliqué à un graphe auxiliaire H. Ce graphe auxiliaire H est constitué de $n+1$ sommets indicés de 0 à n, n étant le nombre de clients, et il existe un arc (i, j), $i < j$ si les contraintes de capacité et de distance sont respectées. Cet algorithme est un algorithme d'étiquetage sans pour autant générer de manière explicite le graphe auxiliaire. Cependant, une seule étiquette est associée à chaque sommet du graphe. Cette étiquette correspond à la distance parcourue pour rejoindre le sommet étiqueté en empruntant le plus court chemin et en respectant les contraintes de capacité et de longueur maximale de chaque route. De nombreuses extensions [Prins *et al.*, 2014] de cet algorithme ont été proposées pour de nombreuses caractéristiques du problème de tournées de véhicules. Malheureusement, aucune de ces extensions ne peut être utilisée pour le HCRSP.

Un nouvel algorithme d'étiquetage a été mis en place afin de décoder une solution. Nous proposons ici une nouvelle définition du graphe auxiliaire H adaptée au HCRSP.

Définition 1. *Graphe auxiliaire H : soit un graphe H = (X,A) avec l'ensemble X des* $|\mathscr{S}| + 1$ *sommets, l'ensemble A des arcs* (i, j), $i < j$, *si la tournée réalisant les services* σ_{i+1} *à* σ_j *est réalisable au regard des contraintes du HCRSP. Le coût associé à l'arc* (i, j) *dépendra d'une somme pondérée des fonctions économiques à optimiser.*

L'algorithme doit choisir un soignant adéquat pour chaque arc du graphe auxiliaire H de tel sorte que pour un chemin donné, un soignant soit employé au plus une fois. Cet algorithme résout par conséquent un problème similaire au problème de plus court chemin avec contraintes de ressources qui est NP-difficile [Dror, 1994]. Le découpage d'une tournée géante encodant une solution du HCRSP est décrit dans l'algorithme 1. L'algorithme 1 introduit est une adaptation au HCRSP de l'algorithme Split proposé dans [Prins *et al.*, 2014] pour le problème de tournées de véhicules à flotte fixe hétérogène (HVRP). L'algorithme 1 associe plusieurs étiquettes à chaque sommet du graphe auxiliaire H. Soit Λ_i

Algorithm 1 Split pour HCRSP

1: $\Lambda_0 \leftarrow \{[-, 0, 0, \{\}]\}$
2: **for** $t \leftarrow 1$ **to** $|\mathscr{S}|$ **do**
3: $\quad \Lambda_i \leftarrow \emptyset$
4: **end for**
5: Calcul du tableau `tags`
6: $\lambda_{best} \leftarrow [-, \phi_{best}, -, -]$ ▷ $\phi_{best} = \infty$
7: **for** $t \leftarrow 0$ **to** $|\mathscr{S}| - 1$ **do**
8: $\quad$ **for each** $(k, d) \in \mathscr{K} \times \mathscr{H}$ **such that** $d \in \mathscr{H}_k$ **do**
9: $\quad\quad i \leftarrow t + 1$
10: $\quad\quad$ **while** $(i \leq |\mathscr{S}|) \wedge feasible1(k, d, t, i)$ **do**
11: $\quad\quad\quad$ **for each** $\lambda = [-, \phi_\lambda, t, \mathscr{R}_\lambda] \in \Lambda_t \mid feasible2(\lambda, t, i)$ **do**
12: $\quad\quad\quad\quad$ **if** `tags[i]` **then**
13: $\quad\quad\quad\quad\quad \mathscr{R}_{\lambda'} \leftarrow \mathscr{R}_\lambda \cup \{(k, d)\}$
14: $\quad\quad\quad\quad\quad \phi_{\lambda'} \leftarrow \phi_\lambda + evaluation(\lambda')$
15: $\quad\quad\quad\quad\quad \lambda' \leftarrow [\lambda, \phi_\lambda, i, \mathscr{R}_{\lambda'}]$
16: $\quad\quad\quad\quad\quad$ **if** $\nexists \lambda'' \in \Lambda_i$ **such that** $domine(\lambda'', \lambda')$ **then**
17: $\quad\quad\quad\quad\quad\quad$ Supprimer dans Λ_i tout λ'' tel que $domine(\lambda', \lambda'')$
18: $\quad\quad\quad\quad\quad\quad \Lambda_i \leftarrow \Lambda_i \cup \{\lambda'\}$
19: $\quad\quad\quad\quad\quad$ **end if**
20: $\quad\quad\quad\quad\quad$ **if** $(i = |\mathscr{S}|) \wedge (\phi_{best} < \phi_\lambda)$ **then**
21: $\quad\quad\quad\quad\quad\quad \lambda_{best} \leftarrow \lambda'$
22: $\quad\quad\quad\quad\quad$ **end if**
23: $\quad\quad\quad\quad$ **else**
24: $\quad\quad\quad\quad\quad \Lambda_i \leftarrow \Lambda_i \cup \{\lambda\}$
25: $\quad\quad\quad\quad$ **end if**
26: $\quad\quad\quad$ **end for**
27: $\quad\quad\quad i \leftarrow i + 1$
28: $\quad\quad$ **end while**
29: $\quad$ **end for**
30: **end for**

l'ensemble des étiquettes situé au sommet i. Chaque étiquette du sommet i représente une solution partielle, c'est-à-dire une combinaison de tournées. Une étiquette λ est un quadruplet défini par $[\lambda', \phi_\lambda, i, \mathscr{R}_\lambda]$ avec λ', l'étiquette dont elle est directement issue ; i, l'indice du sommet où λ se trouve ; ϕ_λ, son coût ; $\mathscr{R}_\lambda$, l'ensemble des tournées (k, d) dont elle est constituée. Le coût d'une étiquette λ représente la somme de l'évaluation de l'ensemble des tournées jusqu'au sommet où elle est située. L'évaluation d'une étiquette (ligne 14), que nous décrivons à la section 3.4, consiste en la satisfaction de l'ensemble des contraintes temporelles définies au chapitre 2. Malheureusement, il n'est pas toujours possible d'évaluer avec exactitude une étiquette à cause des contraintes de synchronisation, d'exclusion et de délai. En effet, lorsque des services appartenant à des tournées distinctes sont liés par ces contraintes, si une étiquette n'est constituée que d'une partie de ces tournées, l'évaluation de cette étiquette ne peut pas être réalisée. Le contenu du tableau `tags` (voir sous-section 3.3.3) permet de déterminer les sommets évaluables.

L'idée de l'algorithme est de propager l'ensemble des étiquettes d'un sommet t vers tous les sommets qui le suivent, en partant du sommet 0 (le dépôt) jusqu'au sommet $\mathscr{S} - 1$. À partir d'un sommet t, pour chaque étiquette $\lambda \in \Lambda_t$ et pour chaque tournée (k, d), une étiquette est générée pour chaque sommet de $t + 1$ à $\mathscr{S}$ en fonction des conditions de faisabilité $feasible1$ et $feasible2$ définies plus loin. Toutes les sous-séquences de la séquence σ des services sont énumérées au travers des boucles **for** (ligne 7) et **while** (ligne 10). La condition de la boucle **while** (ligne 10) permet de tester l'ensemble des contraintes de disponibilité journalière, de qualification et empêche la génération de sous-séquences réalisables, grâce à la fonction $feasible1(k, d, t, i)$ définie comme suit :

Définition 2. *$feasible1(k, d, t, i)$ retourne $\top$ si les conditions suivantes sont vérifiées :*

— *le soignant k est qualifié pour réaliser le service σ_i*

$$q_{\sigma_i k} = \top$$

— *le patient nécessitant le service σ_i est disponible le jour associé à la tournée (k, d)*

$$d \in \mathscr{H}_{\sigma_i}$$

L'ensemble Λ_i des étiquettes du sommet i du graphe est constitué uniquement d'étiquettes non dominées. La règle de dominance est définie comme suit :

Définition 3. *$domine(\lambda, \lambda')$: retourne $\top$ si l'étiquette $\lambda = [-, \phi_\lambda, i, \mathscr{R}_\lambda]$ domine l'étiquette $\lambda' = [-, \phi_{\lambda'}, i, \mathscr{R}_{\lambda'}]$, noté $domine(\lambda, \lambda')$ si :*

$$\mathscr{R}_\lambda \subseteq \mathscr{R}_{\lambda'} \wedge \phi_\lambda \leq \phi_{\lambda'}$$

Dans tous les autres cas, les étiquettes ne sont pas comparables. En éliminant certaines étiquettes et en se basant uniquement sur le critère du coût, les étiquettes gardées ne généreront peut-être pas de solutions réalisables ou elles généreront des étiquettes de moins bonne facture que celles éliminées. Les étiquettes dominées sont supprimées grâce à la condition **if** (lignes 16-19). La garantie de la cohérence de la solution et la vérification des contraintes de délai en jours et de continuité des soins sont assurées grâce à la fonction $feasible2(\lambda, t, i)$ défini comme suit :

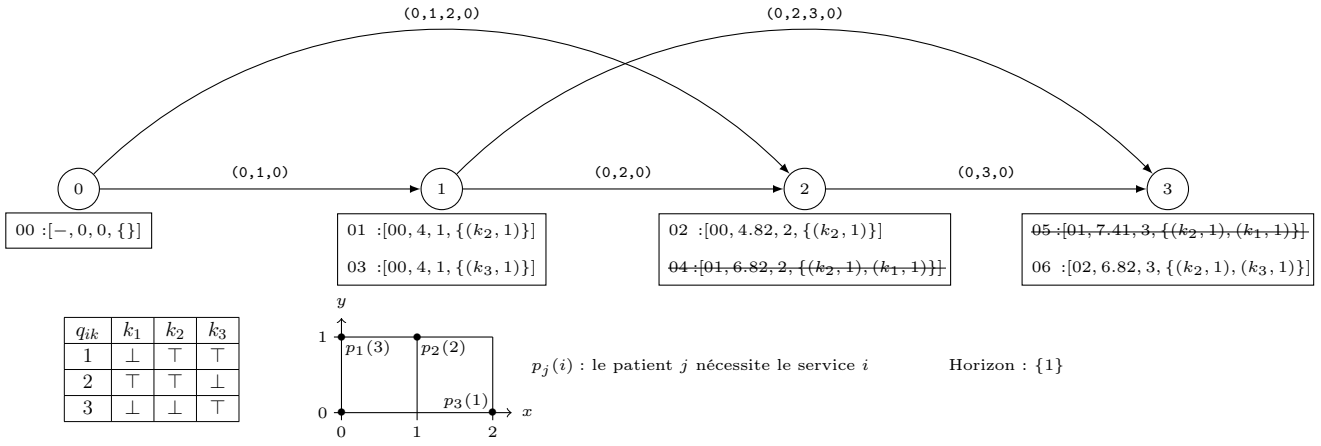


FIGURE 3.2 – Exemple de propagation d'étiquettes dans *Split* pour HCRSP

Définition 4. *$feasible2(\lambda, t, i)$ retourne $\top$ si :*

— *la tournée (k, d) n'a pas déjà été utilisée*

$$(k, d) \notin \mathscr{R}_\lambda$$

— *les délais en jours séparant les services sont respectés*

$$\forall j \in \{1, \ldots, i-1\} \; si \; \exists \; \Delta^{min}_{\sigma_i\sigma_j}, d_{\sigma_j} - d_{\sigma_i} \geq \Delta^{min}_{\sigma_i\sigma_j}$$

$$\forall j \in \{1, \ldots, i-1\} \; si \; \exists \; \Delta^{max}_{\sigma_i\sigma_j}, d_{\sigma_j} - d_{\sigma_i} \leq \Delta^{max}_{\sigma_i\sigma_j}$$

$$\forall j \in \{1, \ldots, i-1\} \; si \; \exists \; \Delta^{min}_{\sigma_j\sigma_i}, d_{\sigma_i} - d_{\sigma_j} \geq \Delta^{min}_{\sigma_j\sigma_i}$$

$$\forall j \in \{1, \ldots, i-1\} \; si \; \exists \; \Delta^{max}_{\sigma_j\sigma_i}, d_{\sigma_i} - d_{\sigma_j} \leq \Delta^{max}_{\sigma_j\sigma_i}$$

— *la contrainte de continuité des soins est respectée*

$$\forall p \in \mathscr{P}, |\mathscr{K}_\lambda \cap (\overline{\mathscr{K}}_p \cup \{k_f\})| \leq \zeta_p$$

avec $\mathscr{K}_\lambda = \{k \mid (k, d) \in \mathscr{R}_\lambda\}$

La condition **if** (ligne 12) qui teste la i-ème case du tableau `tags` nous permet de savoir si l'étiquette λ' est évaluable. L'algorithme qui détermine les valeurs du tableau `tags` est expliqué dans la sous-section 3.3.3. L'évaluation d'une étiquette λ à travers la fonction *evaluation*(λ) est décrite dans la section 3.4.

La figure 3.2 illustre un exemple de HCRSP impliquant trois patients p_1, p_2, p_3, situés sur un plan euclidien, nécessitant les services 3,2,1 respectivement, pour un horizon temporel d'un jour. Dans cet exemple, nous devons uniquement minimiser la distance totale parcourue. Trois soignants k_1, k_2, k_3 dont les qualifications sont données dans le tableau de la figure 3.2, sont disponibles pour réaliser ces services. Dans l'exemple, l'arc $(0, 3)$ n'est pas créé car aucun soignant n'est qualifié pour réaliser les services 1,2 et 3. Par conséquent, la tournée réalisant les services 1,2 et 3 ne peut être effectuée. À partir de l'étiquette fictive `00`,

l'étiquette `01` au sommet 1, l'étiquette `02` au sommet 2 et l'étiquette `03` au sommet 1 sont générées. Ces étiquettes représentent des solutions partielles et respectent les contraintes du problème qui sont dans cet exemple, uniquement les contraintes de qualification. La dominance ne peut être appliquée entre les étiquettes `01` et `03` car l'ensemble des tournées d'aucune des deux étiquettes n'est incluse dans l'autre. L'itération suivante génère l'étiquette `04` au sommet 2 et l'étiquette `05` au sommet 3 à partir l'étiquette `01`. L'étiquette barrée `04` est dominée par l'étiquette `02` car la tournée de l'étiquette `02` est un sous-ensemble des tournées de l'étiquette `04` et le coût de cette dernière est plus élevée. La propagation des étiquettes précédemment générées et l'application de dominance génère finalement une étiquette qui n'est dominée par aucune autre : `06:[02,6.82,{`$(k_2, 1)$`,`$(k_3, 1)$`}]`. Cette étiquette correspond à la solution optimale pour la séquence de services $\langle 1, 2, 3 \rangle$.

3.3.3 Calcul des sommets évaluables

Une étiquette représente une solution partielle, c'est-à-dire plusieurs tournées ne réalisant qu'une sous-séquence de la séquence fixée σ. Si les tournées sont indépendantes les unes des autres, c'est-à-dire qu'aucun service n'est lié à d'autres par une contrainte de délai, de synchronisation ou d'exclusion, alors chaque étiquette peut être évaluée quelque soit le sommet du graphe auxiliaire H. Par contre, si certains services sont liés à un autre par une des contraintes citées précédemment, il n'est pas toujours possible d'évaluer avec exactitude l'étiquette à chaque sommet du graphe auxiliaire H. En effet, pour que l'étiquette puisse être évaluée, elle doit disposer de toutes les tournées réalisant ces services.

Reprenons l'exemple de la figure 3.2. Si le service 1 et le service 3 sont liés par une contrainte de délai, alors les étiquettes `01` et `02` ne peuvent plus être évaluées car la tournée réalisant le service 1 est liée à une tournée qui n'est pas encore connue ; de même pour l'étiquette `02`. L'évaluation n'intervient qu'au sommet 3 où toutes les tournées liées sont connues. L'algorithme 2 détermine les sommets pour lesquels les étiquettes sont évaluables. Le résultat de l'algorithme est stocké dans le tableau `tags`. L'algorithme en question est composé de trois boucles : une boucle principale (ligne 3) parcourant la séquence à décoder, deux boucles secondaires (lignes 4 et 10). La première boucle secondaire marque toutes les cellules successives du tableau tant que le service correspondant n'est pas impliqué dans une contrainte d'exclusion, de synchronisation ou de délai. Dès la rencontre d'un service impliqué dans une contrainte de délai, de synchronisation ou d'exclusion, la deuxième boucle intervient. La deuxième boucle secondaire va déterminer les sous-séquences qui ne sont pas évaluables et marquer les cellules du tableau en conséquence. Les services succédant au service courant et qui lui sont liés sont stockés dans une liste $\mathscr{L}$ (lignes 11 et 16). L'appartenance du service courant à cette liste (ligne 19) traduit la détection de toutes les tournées ou seulement d'une partie d'entre elles. L'évaluation sera possible que si la liste $\mathscr{L}$ est vide (ligne 21), ce qui rend compte, pour les étiquettes stockées au sommet correspondant, de l'existence de toutes les tournées réalisant ces services liés.

Algorithm 2 Calcul du tableau `tags`

1: $\mathscr{L} \leftarrow \varnothing$
2: $i \leftarrow 1$
3: **while** $i \leq |\mathscr{S}|$ **do**
4: **while** $\nexists\, j \in \{i+1, \ldots, |\mathscr{S}|\} \mid exclusion(\sigma_i, \sigma_j) \vee delay(\sigma_i, \sigma_j)$ **do**
5: `tags[i]` $\leftarrow \top$
6: $i \leftarrow i + 1$
7: **end while**
8: $stop \leftarrow \bot$
9: $\mathscr{L} \leftarrow \varnothing$
10: **while** $(i \leq |\mathscr{S}|) \wedge \neg stop$ **do**
11: $\mathscr{M} \leftarrow j \in \{i+1, \ldots, |\mathscr{S}|\} \mid exclusion(\sigma_i, \sigma_j) \vee delay(\sigma_i, \sigma_j)\}$
12: **if** $\mathscr{M} \neq \varnothing$ **then**
13: **if** $i \in \mathscr{L}$ **then**
14: $\mathscr{L} \leftarrow \mathscr{L} \setminus \{i\}$
15: **end if**
16: $\mathscr{L} \leftarrow \mathscr{L} \cup \mathscr{M}$
17: `tags[i]` $\leftarrow \bot$
18: **else**
19: **if** $i \in \mathscr{L}$ **then**
20: $\mathscr{L} \leftarrow \mathscr{L} \setminus i$
21: **if** $\mathscr{L} = \varnothing$ **then**
22: `tags[i]` $\leftarrow \top$
23: $stop \leftarrow \top$
24: **else**
25: `tags[i]` $\leftarrow \bot$
26: **end if**
27: **else**
28: `tags[i]` $\leftarrow \bot$
29: **end if**
30: **end if**
31: $i \leftarrow i + 1$
32: **end while**
33: **end while**

3.3.4 Techniques d'amélioration et inégalités valides

L'algorithme 1 décrit ci-dessus permet de décoder des séquences de petite taille. Pour des séquences de plus grande taille, le nombre d'étiquettes est trop important pour pouvoir décoder la séquence en un temps de calcul raisonnable. Nous avons par conséquent introduit 6 techniques d'amélioration et inégalités valides pour réduire le nombre d'étiquettes générées et donc diminuer la complexité générale de l'algorithme.

Pour la suite de cette sous-section, on considère l'étiquette $\lambda' = [-, \phi_{\lambda'}, t, \mathscr{R}_{\lambda'}] \in \Lambda_t$ avec $t < |\mathscr{S}|$ dont on veut générer l'étiquette fille $\lambda = [\lambda', \phi_\lambda, i, \mathscr{R}_\lambda] \in \Lambda_i$ comme le montre la figure 3.3.

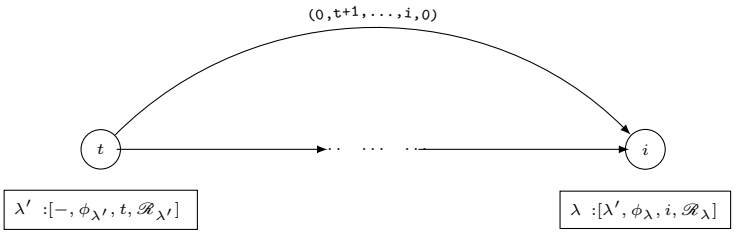


FIGURE 3.3 – Génération d'une étiquette λ à partir de λ'

3.3.4.1 Réduction du nombre d'arcs

Les conditions qui suivent peuvent être ajoutées à la fonction $feasible1$ pour éviter de construire certains arcs. $feasible1$ retourne $\perp$ si :

— il existe un délai horaire minimum entre le service σ_i et un autre service σ_j qui le précède dans la séquence de l'arc entre t et i

$$\exists j \in \{t+1, \ldots, i-1\} \mid \delta^{min}_{\sigma_i \sigma_j} > 0$$

— il existe une synchronisation entre le service σ_i et un autre service σ_j qui le précède dans la séquence de l'arc entre t et i

$$\exists j \in \{t+1, \ldots, i-1\} \mid synchro(\sigma_i, \sigma_j)$$

— le délai horaire maximum devant séparer un service précédant σ_i et le service σ_i est inférieur à la durée les séparant

$$\exists j \in \{t+1, \ldots, i-1\} \mid \delta^{max}_{\sigma_j \sigma_i} < \sum_{l=j}^{i-1} \pi_{\sigma_l} + \tau_{\sigma_l \sigma_{l+1}}$$

3.3.4.2 Nombre de tournées restantes

Proposition 1. *pour tout $t < |\mathscr{S}| - 1$ et $i < |\mathscr{S}|$, si $|\mathscr{R}_{\lambda'}| + 1 \geq |\{(k, d) \mid d \in \mathscr{H}_k\}|$ l'étiquette λ générée à partir de λ' n'est pas viable. Aucune étiquette générée à partir de λ ne peut atteindre le sommet $|\mathscr{S}|$ du graphe auxiliaire H.*

$|\mathscr{K} \times \mathscr{H}|$ représente le nombre maximum de tournées possibles. Par définition, au sommet $t < |\mathscr{S}| - 1$ du graphe auxiliaire H, $|\mathscr{R}_{\lambda'}|$ tournées ont été définies. Au sommet $i < |\mathscr{S}|$, $|\mathscr{K}_{\lambda'} \times \mathscr{H}_{\lambda'}|+1$ seront nécessaires. Si $|\mathscr{R}_{\lambda'}|+1 \geq |\mathscr{K} \times \mathscr{H}|$, l'ensemble des tournées possibles restantes est vide. Le sommet i n'étant pas le dernier sommet du graphe, les étiquettes générées au sommet i ne peuvent donc pas atteindre le dernier sommet $|\mathscr{S}|$du graphe auxiliaire H. La Proposition 1 permet de générer uniquement les étiquettes qui pourront potentiellement être propagées en tenant compte du nombre de tournées restantes.

3.3.4.3 Problème de recouvrement

Définition 5. *$bloc_{ab}^{kd}$ Une sous-séquence $\langle \sigma_a, \ldots, \sigma_b \rangle$, $1 \leq a \leq b \leq |\sigma|$ est un bloc de la tournée (k, d), noté $bloc_{ab}^{kd}$, si toutes les conditions suivantes sont satisfaites :*

- *le soignant k peut réaliser l'ensemble des services de la sous-séquence*

$$q_{\sigma_l k} = \top, a \leq l \leq b$$

- *les patients nécessitant les services de cette séquence sont disponibles le jour d*

$$d \in \mathscr{H}_{\sigma_l}, a \leq l \leq b$$

- *le premier service σ_a de cette sous-séquence doit vérifier au moins une des contraintes suivantes : σ_a suit le service σ_i ; le soignant k ne peut pas réaliser le service σ_{a-1}*

$$i + 1 = a \vee q_{\sigma_{a-1} k} = \bot \vee d \notin \mathscr{H}_{\sigma_{a-1}}$$

- *le dernier service σ_b de cette sous-séquence doit également vérifier une des contraintes suivantes : σ_b et $\sigma_{|\mathscr{S}|}$ sont identiques ; le soignant k ne peut pas réaliser le service σ_{b+1}*

$$j = |\mathscr{S}| \vee q_{\sigma_{b+1} k} = \bot \vee d \notin \mathscr{H}_{\sigma_{b+1}}$$

Une séquence σ peut ne pas être réalisable si la propagation des étiquettes n'atteint pas le dernier sommet. Cela peut être dû aux contraintes de qualification et de disponibilité journalière. D'autres contraintes telles que les contraintes de délai peuvent être impliquées dans ces situations d'infaisabilité. L'exemple 1 illustre un cas d'infaisabilité.

Exemple 1. *On considère une séquence σ de huit services. Pour chaque tournée, il est indiqué sur la figure 3.4 la liste des services pouvant être réalisée en tenant en compte des contraintes de qualification et de disponibilité journalière. Chaque bloc représente donc une sous-séquence de σ pouvant être réalisée par cette tournée. Les blocs présents sont : $bloc_{1,5}^{k_1,d_1}$, $bloc_{8,8}^{k_1,d_1}$, $bloc_{5,7}^{k_1,d_1}$*

L'application de l'algorithme 1 à la séquence σ ne générera pas d'étiquettes au dernier sommet du graphe H auxiliaire. En effet, sachant qu'un bloc représente une tournée de longueur maximale et assignable au soignant k le jour d, aucun choix de blocs ne permet de couvrir tous les services de la séquence. Si les blocs $bloc_{8,8}^{k_1,d_1}$, $bloc_{5,7}^{k_1,d_1}$ sont sélectionnés, cela traduit la création des tournées (k_1, d_1) et (k_2, d_1) réalisant respectivement les sous-séquences $\langle \sigma_8 \rangle$ et $\langle \sigma_5, \sigma_6, \sigma_7 \rangle$ les services $\sigma_1, \sigma_2, \sigma_3$ et σ_4 ne seront pas réalisés. Par contre, si ce sont les blocs $bloc_{1,5}^{k_1,d_1}$, $bloc_{5,7}^{k_1,d_1}$ qui sont sélectionnés, le service σ_8 ne sera pas réalisé. La séquence σ n'est par conséquent pas réalisable.

$\sigma \rightarrow$	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7	σ_8
(k_1, d_1)	σ_1	σ_2	σ_3	σ_4	σ_5			σ_8
(k_2, d_1)					σ_5	σ_6	σ_7	

FIGURE 3.4 – Exemple de blocs pour une séquence fixée σ

$\sigma' \rightarrow$	σ_1	σ_2	σ_3	σ_4	σ_5	σ_8	σ_7	σ_6
(k_1, d_1)	σ_1	σ_2	σ_3	σ_4	σ_5	σ_8		
(k_2, d_1)					σ_5		σ_7	σ_6

FIGURE 3.5 – Exemple de blocs pour une séquence fixée σ'

En échangeant les positions des services σ_6 et σ_8 dans la séquence σ, la nouvelle séquence σ' est potentiellement réalisable car les deux blocs $bloc_{1,6}^{k_1,d_1}$, $bloc_{7,8}^{k_2,d_1}$ recouvrent l'ensemble des services de la séquence.

Afin de vérifier si une solution réalisable peut être déterminée à partir de l'algorithme 1 et d'une séquence σ fixée, la faisabilité du programme de satisfaction défini ci-dessous est une condition nécessaire mais non suffisante.

Ce programme de satisfaction de contraintes est résolu pour vérifier cette condition. Avant la résolution de ce programme, nous vérifions au préalable qu'il existe au moins un bloc couvrant chaque service. Dans le cas contraire, il n'y aurait aucune solution.

`Données`

t	indice de l'étiquette λ'
i	indice de l'étiquette λ
$bloc_{ab}^{kd}$	$\forall (k,d) \in \mathscr{K} \times \mathscr{H} \setminus \mathscr{R}_\lambda,\ i < a \leq b \leq \lvert\mathscr{S}\rvert$

`Variables`

y_{ab}^{kd}	variable booléenne associée au bloc $bloc_{ab}^{kd}$ $\top$ si le bloc $bloc_{ab}^{kd}$ est choisi dans la solution ; $\bot$ sinon

`Contraintes`

$$\bigoplus_{i<a\leq b\leq|\mathscr{S}|} (y_{ab}^{kd} \mid \exists\ bloc_{ab}^{kd}),\ \forall (k,d) \in \mathscr{K} \times \mathscr{H} \setminus \mathscr{R}_\lambda$$

$$\bigwedge_{\forall l \in \{i+1,\ldots,|\mathscr{S}|\}} \ \bigvee_{\forall (k,d) \in \mathscr{K}\times\mathscr{H}\setminus\mathscr{R}_\lambda} (y_{ab}^{kd} \mid \exists\ bloc_{ab}^{kd},\ a \leq l \leq b)$$

Les contraintes définies dans le programme de satisfaction de contraintes vérifie respectivement que :

— pour chaque tournée, au plus un bloc est sélectionné

— pour chaque service, au moins un bloc contenant ce service est sélectionné

Ces contraintes peuvent être très simplement implémentées à l'aide des contraintes globales :

— `atmost`

— `atleast`

La structure de ce programme de satisfaction de contraintes montre qu'il s'agit d'un problème de recouvrement. Ici, seules les contraintes de qualification et disponibilité journalière ont été prises en compte. Prendre en compte d'autres contraintes augmenterait le nombre de blocs et donc le temps de résolution du programme de satisfaction de contraintes. Ce programme peut être résolu avant l'appel de l'algorithme 1 permettant ainsi d'éliminer certaines séquences qui ne sont pas réalisables, faisant ainsi office de filtrage. Il peut également être résolu pour chaque étiquette générée par l'algorithme 1 estimant la viabilité de cette étiquette.

Il est à remarquer que cette inégalité valide n'est utile que si la sous-traitance n'est pas autorisée. Dans le cas contraire, le soignant fictif étant capable de réaliser tous les services, le programme de satisfaction de contraintes est toujours réalisable.

3.3.4.4 Bornes

On suppose que l'on dispose :
— d'une borne inférieure LB_i au sommet i représentant le coût minimal nécessaire pour réaliser les services $\langle \sigma_{i+1}, \ldots, \sigma_{|\mathscr{S}|} \rangle$;
— d'une borne supérieure UB représentant le coût de la séquence σ. Cette borne peut être obtenue à la suite d'un décodage à l'aide d'une méthode heuristique.

L'idée est d'éviter la génération de l'étiquette λ si son coût est trop élevé. L'étiquette λ est générée si :

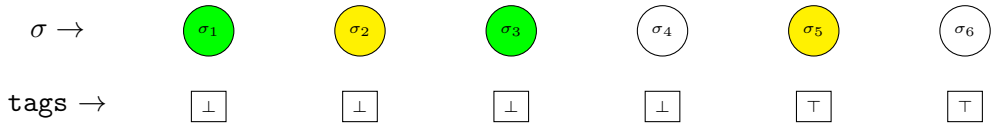
$$LS_i + \phi_\lambda < UB$$

Nous présentons dans les lignes qui suivent :
— l'estimation du coût ϕ_λ de l'étiquette λ car l'évaluation exacte des étiquettes à chaque sommet n'est pas toujours possible (voir sous-section 3.3.3). Il est donc nécessaire de calculer une estimation de la valeur ϕ_λ la plus réaliste possible.
— le calcul d'une borne inférieure pour chaque sommet i ;
— le calcul d'une borne supérieure de bonne facture respectant la séquence fixée σ

Estimation du coût d'une étiquette. Comme nous l'avons déjà souligné dans la sous-section 3.3.3, il n'est pas possible d'évaluer avec exactitude et d'appliquer la dominance à chaque sommet du graphe H à cause des contraintes de synchronisation, de délai et d'exclusion. Si la valeur de la cellule `tags[i]` vaut $\bot$, l'étiquette λ ne peut pas être évaluée. Le coût de l'étiquette λ correspond au coût de l'une des étiquettes dont elle est issue et pour laquelle la cellule du tableau `tags` correspondant vaut $\top$. Une estimation de l'étiquette λ est déterminée en respectant uniquement les contraintes de délai de synchronisation et d'exclusion pour les services dont l'ensemble des services avec lesquels ils sont liés appartiennent à la sous-séquence $\langle \sigma_1, \ldots, \sigma_i \rangle$.

Exemple 2. *La figure ci-dessous nous permet d'illustrer notre propos. Dans cette séquence de 6 services, les services σ_1 et σ_3 sont liés par une contrainte de délai, les services σ_2 et σ_5 sont également liés par une contrainte de délai. Supposons qu'il existe une étiquette générée au sommet σ_4. Cette étiquette ne peut être évaluée car la valeur de la cellule du tableau `tags` correspondante vaut $\bot$. Cela traduit simplement le fait que la tournée réalisant le service*

σ_5, qui est lié au service σ_2 n'est pas encore connue. L'estimation que nous proposons est la suivante : évaluer l'étiquette en tenant compte uniquement de la contrainte de délai qui lie les services σ_1 et σ_3. Le service σ_2 sera considéré pour cette étiquette comme un service qui n'est lié à aucun autre.



Borne inférieure. L'algorithme 1 est appliqué de gauche à droite sur la séquence σ, c'est-à-dire pour des valeurs de l'indice t allant croissant de 0 à $|\mathscr{S}| - 1$. Pour obtenir une borne inférieure pour chaque sommet du graphe auxiliaire H, l'algorithme 1 devra être appliqué de droite à gauche, pour des valeurs de l'indice t allant décroissant de $|\mathscr{S}| - 1$ à 0. Cependant, pour obtenir à la fois un temps de calcul raisonnable, la dominance s'effectue uniquement par valeur. Ainsi, la fonction *domine* de la Définition 3 est redéfinie en supprimant le premier terme de la conjonction et nous obtenons :

$$domine(\lambda, \lambda') \equiv \phi_\lambda \leq \phi_{\lambda'}$$

Borne supérieure. Nous proposons deux heuristiques pour obtenir une borne supérieure pour la séquence fixée σ.

La première heuristique est un algorithme de parcours en profondeur. L'algorithme 3 retourne une première solution réalisable. L'algorithme effectue un parcours en profondeur du graphe auxiliaire H. Pour mettre en place ce parcours, un pile est utilisée à cet effet. Une pile S vide est d'abord initialisée avec l'étiquette fictive λ_0 à la ligne 3. L'algorithme est construit sur le principe de génération d'étiquettes à partir de l'étiquette au sommet de la pile. La pile permet la mise en place d'un retour sur trace des chemins non explorés tout au long du parcours. Lorsqu'une étiquette est dépilée, nous disposons du sommet t où elle se situe dans le graphe auxiliaire H. De nouvelles étiquettes seront générées pour les sommets $t + 1$ à $|\mathscr{S}|$. Seules les nouvelle étiquettes réalisables et non dominées (lignes 11-29) sont stockées au i-ème sommet du graphe auxiliaire H. L'ensemble de ces étiquettes réalisables et non dominées est ensuite empilé dans la pile (lignes 30-32). L'ensemble du processus est réitéré jusqu'à ce que la pile soit vide. Toutes les inégalités valides de la section courante sont également valables pour cet algorithme.

Le deuxième algorithme est une heuristique de concaténation de sous-séquences. L'idée est de concaténer des sous-séquences successives. Dans l'algorithme 4, une sous-séquence est représentée par le triplet $[i, j, \mathscr{R}]$ avec : les indices i et j, respectivement l'indice du premier et du dernier service de la sous-séquence ; l'ensemble des tournées $(k, d) \in \mathscr{R}$ pouvant réaliser tous les services de la sous-séquence $\langle \sigma_i, \ldots, \sigma_j \rangle$. L'appel aux fonctions $start(\cdot)$, $end(\cdot)$ et $routes(\cdot)$ permet d'obtenir respectivement les valeurs de i, j et $\mathscr{R}$. Au début de l'algorithme, une sous-séquence est créée pour chaque service (lignes 2-4). Cette solution initiale est dans la plupart des cas irréalisable. Une des causes vient du nombre de tournées possibles qui est en général très inférieur au nombre de services. Seules les sous-séquences qui se succèdent vont être concaténée (lignes 11-13). La concaténation dont

Algorithm 3 Split parcours en profondeur pour HCRSP

$\lambda_0 \leftarrow [-, 0, 0,]$
$stop \leftarrow \bot$
$push(\lambda_0, S)$
while $\neg empty(S)$ **do**
    $\lambda \leftarrow top(S)$ ▷ $\lambda = [-, t, -, -]$
    $pop(S)$
    Supprimer λ de Λ_t
    $i \leftarrow t + 1$
    while $(i \leq |\mathscr{S}|)$ **do**
        for each $(k, d) \in \mathscr{K} \times \mathscr{H}$ **such that** $d \in \mathscr{H}_k$ **do**
            if $feasible1(k, d, t, i) \wedge feasible2(\lambda, t, i)$ **then**
                $\lambda' \leftarrow [\lambda, \phi_\lambda, i, \mathscr{R}_{\lambda'}]$
                if `tags[i]` **then**
                    $\phi_{\lambda'} \leftarrow \phi_\lambda + evaluation(\lambda')$
                    $\mathscr{R}_{\lambda'} \leftarrow \mathscr{R}_\lambda \cup \{(k, d)\}$
                    if λ' is feasible **then**
                        if $i = |\mathscr{S}|$ **then**
                            return λ
                        end if
                        if $\nexists \lambda'' \in \Lambda_i$ **such that** $domine(\lambda'', \lambda')$ **then**
                            Supprimer de Λ_i tout λ'' telle que $domine(\lambda', \lambda'')$
                            $\Lambda_i \leftarrow \Lambda_i \cup \{\lambda'\}$
                        end if
                    end if
                else
                    $\Lambda_i \leftarrow \Lambda_i \cup \{\lambda'\}$
                end if
            end if
        end for
        for each $\lambda'' \in \Lambda_i$ **do**
            $push(\lambda'', S)$
        end for
        $i \leftarrow i + 1$
    end while
end while

le coût est le plus faible est conservée pour la prochaine itération. L'algorithme s'arrête lorsque plus aucune concaténation n'est réalisable.

3.3.4.5 Symétrie

Soient deux soignants k_1 et k_2 tels que :
- leurs sommets de départ coïncident ;
  $\mathfrak{o}_{k_1} = \mathfrak{o}_{k_2}$
- leurs sommets de retour coïncident
  $\mathfrak{d}_{k_1} = \mathfrak{d}_{k_2}$
- le soignant k_2 est toujours disponible les jours pour lesquels le soignant k_1 est disponible
  $\mathscr{H}_{k_1} \subseteq \mathscr{H}_{k_2}$
- les horaires de travail de k_1 sont inclus dans ceux de k_2
  $\forall d \in \mathscr{H}_{k_1}, e_{k_1 d} \geq e_{k_2 d} \wedge l_{k_1 d} \leq l_{k_2 d}$
- pour un jour $d \in \mathscr{H}_{k_1}$ donné, leurs sommets de pause coïncident
  $\forall d \in \mathscr{H}_{k_1}, \mathfrak{b}_{k_1 d} = \mathfrak{b}_{k_2 d}$
- le soignant k_2 peut réaliser l'ensemble des services que le soignant k_1 peut réaliser
  $\forall i \in \mathscr{S}, q_{ik_1} \Rightarrow q_{ik_2}$
- les différents coûts qui leur sont associés sont identiques

On veut générer les étiquettes

$$\lambda_1 = [\lambda', \phi_{\lambda_1}, i, \mathscr{R}_{\lambda'} \cup \{(k_1, d)\}]$$

et

$$\lambda_2 = [\lambda', \phi_{\lambda_2}, i, \mathscr{R}_{\lambda'} \cup \{(k_2, d)\}]$$

à partir de l'étiquette λ' pour un jour d donné.

Proposition 2. *pour tout $d \in \mathscr{H}_{k_1}$, pour chaque étiquette λ'_1 générée à partir de λ_1 au sommet j du graphe auxiliaire H, il existe une étiquette λ'_2 générée à partir de λ_2 au sommet j, de coût identique et couvrant l'ensemble des qualifications et des disponibilités journalières de l'étiquette λ'_1, $i < j \leq |\mathscr{S}|$.*

L'ensemble des tournées restantes pouvant être utilisées par les étiquettes générées à partir de λ_1, respectivement λ_2 est :

$\mathscr{R}_{\lambda'_1} = \mathscr{K} \times \mathscr{H} \setminus (\mathscr{R}_{\lambda'} \cup \{(k_1, d)\})$
$\mathscr{R}_{\lambda'_2} = \mathscr{K} \times \mathscr{H} \setminus (\mathscr{R}_{\lambda'} \cup \{(k_2, d)\})$

Si on note $\mathscr{R}' = \mathscr{R}_{\lambda'_1} \cap \mathscr{R}_{\lambda'_2}$, on peut exprimer ces tournées de la manière suivante :

$\mathscr{R}_{\lambda'_1} = \mathscr{R}' \cup \{(k_2, d)\})$
$\mathscr{R}_{\lambda'_2} = \mathscr{R}' \cup \{(k_1, d)\})$

Toute étiquette λ'_1 générée à partir de λ_1 consommera une tournée de $\mathscr{R}_{\lambda'_1}$. De même, toute étiquette λ'_2 générée à partir de λ_2 consommera une tournée de $\mathscr{R}_{\lambda'_2}$. Il n'est par conséquent pas utile de générer l'étiquette λ_2 car les tournées possibles restantes de l'étiquette λ_1 couvrent toutes qualifications et les disponibilités journalières des tournées restantes de l'étiquette λ_2 et le coût de ϕ_{λ_1} est identique au coût de ϕ_{λ_2} du fait des hypothèses. Par

Algorithm 4 Split Heuristique pour HCRSP par concaténation de sous-séquences

1: $result \leftarrow$ `null`
2: **for** $t \leftarrow 1$ **to** $|\mathscr{S}|$ **do** ▷ construction d'un tableau dynamique Π de taille $|\mathscr{S}|$
3: $\quad \Pi[t] \leftarrow [t, t, \{(k,d) \in \mathscr{K} \times \mathscr{H} \mid q_{\sigma_t k} = \top \wedge d \in \mathscr{H}_{\sigma_t}\}]$
4: **end for**
5: $stop \leftarrow \bot$
6: **while** $\neg stop$ **do**
7: $\quad best \leftarrow$ `null`
8: $\quad index \leftarrow 0$
9: $\quad \phi_{best} \leftarrow \infty$
10: $\quad$ **for** $i \leftarrow 1$ **to** `size`(Π)-1 **do**
11: $\quad\quad$ start $\leftarrow$ `start`($\Pi[i]$)
12: $\quad\quad$ end $\leftarrow$ `end`($\Pi[i+1]$)
13: $\quad\quad$ tmp $\leftarrow$ [start, end, `routes`($\Pi[i]$) $\cap$ `routes`($\Pi[i+1]$)]
14: $\quad\quad$ **if** `routes`(tmp) $\neq \varnothing$ **then**
15: $\quad\quad\quad$ Créer une solution partielle ς avec la sous-séquence tmp
16: $\quad\quad\quad$ **if** ς *is feasible* $\wedge$ ($best =$ `null` $\vee \phi_\varsigma < \phi_{best}$) **then**
17: $\quad\quad\quad\quad best \leftarrow tmp$
18: $\quad\quad\quad\quad index \leftarrow i$
19: $\quad\quad\quad\quad \phi_\varsigma \leftarrow \phi_{best}$
20: $\quad\quad\quad$ **end if**
21: $\quad\quad$ **end if**
22: $\quad$ **end for**
23: $\quad$ **if** $best \neq$ `null` **then**
24: $\quad\quad$ Détruire $\Pi[index + 1]$
25: $\quad\quad \Pi[index] \leftarrow best$
26: $\quad\quad$ **if** $size(\Pi) \leq |\mathscr{K} \times \mathscr{H}|$ **then**
27: $\quad\quad\quad$ Créer une solution totale ς avec toutes les séquences de Π
28: $\quad\quad\quad$ **if** ς *is feasible* $\wedge$ ($result =$ `null` $\vee \phi_\varsigma < \phi_{result}$) **then**
29: $\quad\quad\quad\quad result \leftarrow \varsigma$
30: $\quad\quad\quad$ **else**
31: $\quad\quad\quad\quad stop \leftarrow \top$
32: $\quad\quad\quad$ **end if**
33: $\quad\quad$ **end if**
34: $\quad$ **else**
35: $\quad\quad stop \leftarrow \top$
36: $\quad$ **end if**
37: **end while**
38: **return** *result*

contre, générer uniquement l'étiquette λ_2 peut conduire à des étiquettes filles irréalisables car les tournées restantes de l'étiquette λ_2 ne couvrent pas forcément les qualifications et les disponibilités journalières des tournées restantes de l'étiquette λ_1.

3.3.4.6 Détection d'interblocage

Nous commençons par rappeler quelques résultats théoriques qui seront exploités par la suite.

Soit un programme linéaire dont le système de contraintes peut être noté $Ax \leq b$.

Définition 6. *([Cormen* et al.*, 2001]) Dans un* ***système de contraintes de différence****, chaque ligne de la matrice A du programme linéaire contient un 1 et un -1, et toutes les autres composantes de A valent 0. Les contraintes données par $Ax \leq b$ représentent donc un ensemble de m* ***contraintes de différence*** *à n inconnues, chaque contrainte étant une inégalité linéaire simple de la forme*

$$x_j - x_i \leq b_k, \text{ où } 1 \leq i, j \leq n, i \neq j \text{ et } 1 \leq k \leq m.$$

Définition 7. *([Cormen* et al.*, 2001]) Étant donné un système $Ax \leq b$ de contraintes de différence, le* ***graphe des contraintes*** *est un graphe orienté pondéré $G = (S, A)$, où*

$S = \{v_0, v_1, \ldots, v_n\}$

et

$A = \{(v_i, v_j) \mid x_j - x_i \leq b_k \text{ est une contrainte } \} \cup (v_0, v_1), (v_0, v_2), (v_0, v_3), \ldots, (v_0, v_n)$

Notons que v_0 est un sommet fictif et tous les arcs $(v_0, v_i), 1 \leq i \leq n$, ont un poids nul.

Théorème 1. *([Cormen* et al.*, 2001]) Étant donné un système $Ax \leq b$ de contraintes de différence, soit $G = (S, A)$ le graphe des contraintes correspondant. Si G contient un circuit de poids strictement négatif, il n'existe aucune solution réalisable pour le système.*

[Firat et Woeginger, 2011] a proposé de reformuler le test de faisabilité du problème de transport à la demande (Dial-A-Ride Problem) comme un système de contraintes de différence. Seules les contraintes de fenêtre de temps et de délai maximum ont été prises en compte. Dans un tel système, il a été prouvé (voir Théorème 1) qu'une solution est réalisable si et seulement si le graphe de contraintes correspondant ne contient aucun circuit de poids strictement négatif. De même, [Masson *et al.*, 2013] a proposé un test de faisabilité pour le PDPT (Pickup and Delivery Problem with Transfers). Les auteurs ont étendu le Forward Time Slack [Savelsbergh, 1992] au problème et vérifient la non-existence d'un cycle dans le graphe de précédence du problème. Ce cycle rend compte de la précédence d'un sommet par lui même ; ce qui conduit à une situation d'interblocage. Du fait de l'existence de contraintes de délai dans le HCRSP, une situation d'interblocage peut également intervenir. Nous nous proposons donc de détecter d'éventuels interblocages en prenant en compte uniquement les contraintes de précédence et de délai. Soit $V_\lambda = \{\sigma_l, 1 \leq l \leq i \mid \exists m,\ delay(\sigma_l, \sigma_m), 1 \leq m \leq i\}$, l'ensemble des services couverts par l'étiquette λ et impliqués dans une contrainte de délai.

Définition 8. *Le système de contraintes de différence associé à l'étiquette $\lambda = [\lambda', \phi_\lambda, i, \mathscr{R}_\lambda]$ en tenant compte uniquement des contraintes de délai et de précédence est défini comme suit :*

$$\begin{array}{lcll} x_u & - & x_v & \leq & -\sum_{p=u}^{v-1} \pi_{\sigma_p} - \tau_{\sigma_p\sigma_{p+1}}, & \forall \sigma_u, \sigma_v \in V_\lambda \wedge prec(\sigma_u, \sigma_v) \\ x_u & - & x_v & \leq & -\delta_{uv}^{min}, & \forall \sigma_u, \sigma_v \in V_\lambda \wedge \exists \delta_{\sigma_u,\sigma_v}^{min} \geq 0 \\ -x_u & + & x_v & \leq & \delta_{uv}^{max}, & \forall \sigma_u, \sigma_v \in V_\lambda \wedge \exists \delta_{\sigma_u,\sigma_v}^{max} \geq 0 \end{array}$$

Le Théorème 1 nous permet d'affirmer que l'existence d'un circuit strictement négatif dans le graphe de contraintes associé à ce système de contraintes de différence suffit à prouver l'irréalisabilité de l'étiquette λ.

Il est à remarquer, d'une part, que si les temps d'attente sont autorisés dans le HCRSP, ces derniers n'étant pas connus avant l'évaluation de l'étiquette, la durée séparant deux services appartenant à la même route est sous-estimée. Par conséquent, la détection d'interblocages proposée ne permet d'éliminer qu'une partie des interblocages existants dans la séquence fixée σ. D'autre part, dans une étiquette où tous les services liés à un autre sont uniquement liés par une synchronisation, $\delta_{ij}^{min} = \delta_{ij}^{max} = 0$, l'existence d'un cycle suffit à prouver la non-viabilité de l'étiquette en question.

3.4 Représentation directe d'une solution

Une représentation directe d'une solution du HCRSP est proposée dans cette section. Ensuite, un algorithme de décodage est décrit permettant d'obtenir une solution totale à partir de cette représentation directe.

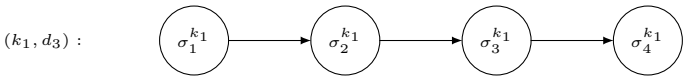
Définition 9. *Soient deux tournées (k, d) et (k', d') et l'ensemble des services réalisés par ces tournées respectivement $\mathscr{S}_{(k,d)}$ et $\mathscr{S}_{(k',d')}$. Les tournées (k, d) et (k', d') sont des tournées liées s'il existe deux services $i \in \mathscr{S}_{(k,d)}$ et $j \in \mathscr{S}_{(k',d')}$ impliqués dans une contrainte d'exclusion et de délai :*

$$\exists\ i \in \mathscr{S}_{(k,d)}, j \in \mathscr{S}_{(k',d')}\ :\ exclusion(i, j) \vee delay(i, j)$$

Définition 10. *Soient trois tournées (k, d), (k', d') et (k'', d''). On considère que les tournées (k, d) et (k', d') sont des tournées liées et que les tournées (k', d') et (k'', d'') sont également des tournées liées. Par transitivité, les tournées (k, d) et (k'', d'') sont des tournées liées.*

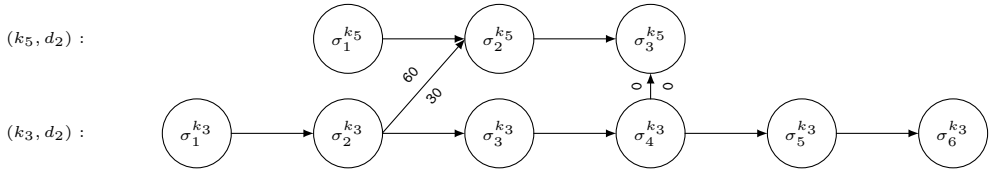
Définition 11. *Soient un ensemble des tournées noté $\mathscr{R}$ et l'ensemble des services réalisés par ces tournées noté $\mathscr{S}_{\mathscr{R}}$. $\mathscr{R}$ est un ensemble de tournées liées si : pour deux tournées distinctes (k, d) et (k', d') de $\mathscr{R}$, ces tournées sont des tournées liées.*

Deux exemples de tournées sont proposés ci-dessous : une tournée qui n'est pas liée et une tournée liée.



Exemple 3. *une tournée simple est donnée dans ce premier exemple. En dehors des contraintes de précédence, les services que réalisent la tournée* (k_1, d_3) *ne sont liés à aucun service réalisé par une autre tournée. Ce n'est pas une tournée liée.*

Exemple 4. *les tournées* (k_3, d_2) *et* (k_5, d_2) *appartiennent au même ensemble de tournées liées. En effet, le service* σ_2 *de la tournée* (k_5, d_2) *doit être réalisé au plus tôt 30 min et au plus tard 60 min après le service* σ_5 *de la tournée* (k_3, d_2)*. De même, le service* σ_3 *de la tournée* (k_5, d_2) *doit être réalisé en même temps que le service* σ_7 *de la tournée* (k_3, d_2)*.*



3.4.1 Encodage

L'encodage proposé pour représenter une solution est un encodage direct. Pour cet encodage, nous supposons que les contraintes suivantes sont satisfaites :

— contraintes de disponibilité journalière des soignants et des services
— contraintes de délai en jours
— contraintes d'affectation et de séquencement
— contraintes de continuité de soins

Chaque tournée (k, d) réalisée par un soignant k le jour d d'une solution ς partielle ou totale, est encodée par une séquence de services. Cette séquence correspond à l'ordre dans lequel le soignant k réalisera les services qui lui sont assignés. Cet encodage prend également en compte la sous-traitance en assignant au soignant fictif k_f l'ensemble des services qui n'appartiennent à aucune tournée. Cet encodage correspond à une liste de tournées simple et/ou liée. La Figure 3.6 illustre l'encodage d'une solution constitué de 5 tournées établies sur 2 jours. L'ensemble des services que nécessitent les 7 patients a été réalisé par ces tournées. Par conséquent, aucun service n'est affecté au soignant fictif k_f. Il est à noter que le premier jour, il existe une synchronisation liant les services σ_3 et σ_5. Les tournées $(k_1, 1)$ et $(k_2, 1)$ réalisant ces services sont donc des tournées liées.

3.4.2 Évaluation d'une solution

Une solution partielle ou totale est représentée par un ensemble de séquences de services qui est réalisé par des tournées simple et/ou liée. Pour évaluer une solution du HCRSP à partir de cette représentation directe, chaque tournée simple est évaluée indépendamment et chaque ensemble de tournées liées est évalué conjointement.

Dans l'exemple de la Figure 3.6, les tournées $(k_3, 1)$, $(k_1, 2)$ et $(k_3, 2)$ sont décodées indépendamment. Par contre, les tournées liées $(k_1, 1)$ et $(k_2, 1)$ sont décodées conjointement

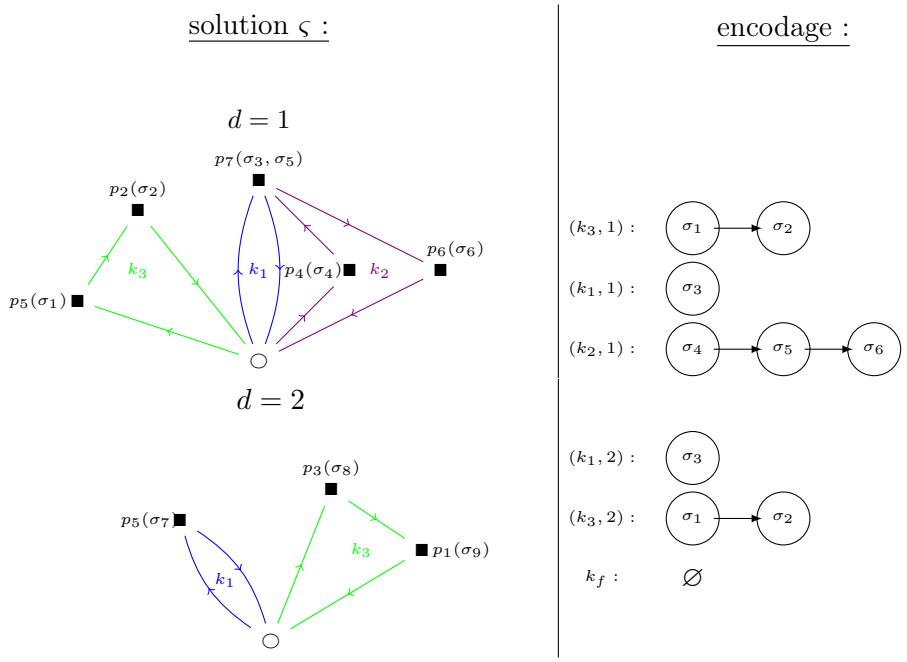


FIGURE 3.6 – Exemple d'une solution du HCRSP et de son encodage direct

à cause de la contrainte de synchronisation qui lie les services σ_3 et σ_5. Évaluer une solution consiste à déterminer les dates de début des services tout en respectant les contraintes du problème. Cette évaluation nous renvoie donc à la résolution d'un problème de *timing*. Une revue de la littérature a été proposée par [Vidal *et al.*, 2015]. Une étude des problèmes de *timing* y est proposée, soulignant les algorithmes les plus efficaces pour résoudre ces problèmes, et comment réaliser de manière efficace tout processus de ré-optimisation dans une situation de recherche locale par exemple. Les cas NP-difficiles sont mis en avant. Le lecteur est renvoyé à cette étude pour une quelconque évaluation des tournées simples. Le problème de *timing* général y est défini comme suit :

Définition 12. *([Vidal* et al.*, 2015]) Soit $A = (a_1, \ldots, a_n)$ une séquence de n tâches de durées d'exécution $p_1, \ldots, p_n$. Les dates d'exécution de ces tâches $\boldsymbol{t} = (t_1, \ldots, t_n)$ doivent pour respecter l'ordre total imposé par les indices, telles que $t_i + p_i \leq t_{i+1}$ pour $i \in \{1, \ldots, n-1\}$. Des caractéristiques additionnelles du problèmes F^x, pour $x \in \{1, \ldots, m\}$, permettent de prendre en compte des configurations où elles représentent des objectifs, $F^x \in \mathscr{F}^{obj}$, ou des contraintes $F^x \in \mathscr{F}^{cons}$. Chaque caractéristique F^x est représentée par un ensemble de m_x fonctions $f^x_y(\boldsymbol{t})$ pour $y \in \{1, \ldots, m_x\}$. Le* ***problème général de*** **timing** *consiste à trouver une solution réalisable $\boldsymbol{t}$, respectant les contraintes de séquencement, les contraintes des caractéristiques et minimisant la somme pondérée des contributions de toutes les caractéristiques du problème en tant que fonction objectif.*

$$\min_{\boldsymbol{t}=(t_1,\ldots,t_n)\in\mathbb{R}^{n+}} \quad \sum_{F^x\in\mathscr{F}^{obj}} \alpha_x \sum_{1\leq y\leq m_x} f_y^x(\boldsymbol{t}) \sum_{1\leq y\leq m_x} f_y^x(\boldsymbol{t}) \tag{3.1a}$$

$$s.c. \qquad t_i + p_i \leq t_{i+1} \qquad 1 \leq i < n \tag{3.1b}$$

$$f_y^x(\boldsymbol{t}) \leq 0 \quad F^x \in \mathscr{F}^{cons}, 1 \leq y \leq m_x \tag{3.1c}$$

Cette définition est limitée à une seule séquence de services. Pour plusieurs séquences de services qui sont liées par des contraintes de délai, de synchronisation, d'exclusion et de précédence, le problème de *timing* étant différent, la définition doit être étendue. La séquence A est remplacée par plusieurs séquences de services et plusieurs vecteurs de date d'exécution remplaceront le vecteur $\mathbf{t}$. Enfin, parmi les caractéristiques $\mathscr{F}^{cons}$, pour chaque séquence, il existe au moins une tâche qui est liée à une autre par une contrainte de délai et d'exclusion avec une autre tâche d'une autre séquence.

Nous proposons un programme linéaire en nombres entiers pour modéliser le problème de *timing* à résoudre. Ce programme linéaire en nombres entiers peut être résolu à l'aide d'un solveur mathématique. Ce programme linéaire reprend le modèle proposé au chapitre 2 pour lequel le séquencement issu de l'encodage a été respecté. Dans un souci d'allègement de la notation, les indices fixés sont supprimés, le jour d par exemple. Un algorithme polynomial est ensuite décrit couvrant de nombreux cas rencontrés dans la littérature du HCRSP. L'idée est d'utiliser une méthode spécifique en fonction des caractéristiques du problème en question. L'algorithme de décodage de l'encodage direct consiste donc en la résolution de cas particuliers d'un problème de *timing*.

3.4.2.1 Programme linéaire en nombres entiers

On dispose d'un ensemble K de soignants pour un même jour d donné. À chaque soignant k est affecté une séquence de services σ^k. Pour chaque séquence, il existe au moins un service lié à un autre service d'une séquence différente par une contrainte de délai ou d'exclusion. Le problème de *timing* auquel nous faisons face doit déterminer les dates d'exécution de tous les services de sorte à respecter toutes les contraintes prises en compte par le problème.

Données.

- K : l'ensemble des soignants réalisant les séquences de services. Pour chaque séquence, un seul soignant lui est affecté
- $\mathscr{S}_K$: l'ensemble de tous les services des séquences
- d : le jour pour lequel ces séquences sont réalisées
- σ^k : séquence de services réalisée par la soignant k
- σ_i^k : i-ème service de la séquence σ^k

Les autres données utilisées dans la formulation proviennent de la liste établie dans la définition du problème général.

Variables. La planification sera mise en place grâce aux variables de décisions suivantes :

— y_{ij} : si le service i est réalisé avant le service j ; 0 sinon
— t_i : date de début du service i
— $tard_i$: retard du service i.
— $tard^{max}$: retard maximal pour l'ensemble des services
— u_i^n : 1 si le service i est effectué dans la n-ième fenêtre de temps du jour d ; 0 sinon
— $tstart_k$: date de début de la tournée du soignant k
— $tend_k$: date de fin de la tournée du soignant k
— $tbreak_k$: date de début de la pause du soignant k
— p_{ij}^k : 1 si la pause du soignant k est prise entre le sommet i avant le sommet j ; 0 sinon
— oc : le coût total du nombre d'heures supplémentaires effectuées pour l'ensemble des soignants

Contraintes. Les inégalités ci-dessous sont des réécritures des inégalités présentées au chapitre 2 que nous avons adaptées au problème de *timing*.

$$e_k \leq tstart_k \leq l_k, \forall k \in K \tag{3.2}$$

Les inégalités (3.2) contraignent la date de départ de chaque soignant pour qu'elle respecte la fenêtre de temps qui lui est associée.

$$tstart_k + \tau_{\mathbf{o}_k \sigma_1^k} \leq t_{\sigma_1^k}, \forall k \in K \tag{3.3}$$

$$t_{\sigma_i^k} + \pi_{\sigma_i^k} + \tau_{\sigma_i^k \sigma_{i+1}^k} \leq t_{\sigma_{i+1}^k}, \forall i \in \{1, \ldots, |\sigma^k| - 1\}, \forall k \in K \tag{3.4}$$

$$t_{|\sigma^k|} + \pi_{|\sigma^k|} + \tau_{|\sigma^k| \mathbf{d}_k} \leq tend_k, \forall k \in K \tag{3.5}$$

Les inégalités (3.3), (3.4) et (3.5) forcent le respect du séquencement imposé par l'encodage. Ainsi, l'ensemble des contraintes de précédence pour une même tournée sont satisfaites. Deux services consécutifs sont séparés par la durée d'exécution du premier à laquelle s'ajoute la durée la durée minimale nécessaire pour joindre le second service en partant du premier.

$$\sum_{n=1}^{\gamma_{\sigma_i^k}} u_{\sigma_i^k}^n = 1, \forall i \in \{1, \ldots, |\sigma^k|\}, \forall k \in K \tag{3.6}$$

$$\sum_{n=1}^{\gamma_{\sigma_i^k}} e_{\sigma_i^k}^n u_{\sigma_i^k}^n \leq t_{\sigma_i^k}, \forall i \in \{1, \ldots, |\sigma^k|\}, \forall k \in K \tag{3.7}$$

$$tard_{\sigma_i^k} + \sum_{n=1}^{\gamma_{\sigma_i^k}} l_{\sigma_i^k}^n u_{\sigma_i^k}^n \geq t_{\sigma_i^k}, \forall i \in \{1, \ldots, |\sigma^k|\}, \forall k \in K \tag{3.8}$$

$$tard_{\sigma_i^k} \leq tard^{max}, \forall i \in \{1, \ldots, |\sigma^k|\}, \forall k \in K \tag{3.9}$$

Si le retard n'est pas autorisé, les inégalités (3.6), (3.7) et (3.8) suffisent à contraindre la date de début de chaque service à appartenir à une fenêtre de temps associée à ce service. Dans le cas contraire, la violation de ces fenêtres de temps est possible grâce à l'introduction de

variables d'écart qui comptent le retard induit. Le retard maximal est compté en ajoutant au modèle l'inégalité (3.9).

$$t_j - t_i \leq \delta_{ij}^{max}, \forall i \in \mathscr{S}_K, \forall j \in \mathscr{S}_K \setminus \{i\} \mid \exists \delta_{ij}^{max} \geq 0 \tag{3.10}$$

$$t_j - t_i \geq \delta_{ij}^{min}, \forall i \in \mathscr{S}_K, \forall j \in \mathscr{S}_K \setminus \{i\} \mid \exists \delta_{ij}^{min} \geq 0 \tag{3.11}$$

$$t_i + \pi_i \leq t_j + M \cdot (1 - y_{ij}), \forall \epsilon \in \mathscr{E}, \forall i \in \epsilon \cap \mathscr{S}_K, \forall j \in (\epsilon \cap \mathscr{S}_K) \setminus \{i\} \tag{3.12}$$

Les inégalités (3.10), (3.11) et (3.12) permettent de prendre en compte les contraintes de délai et d'exclusion entre services.

$$p^k_{\mathfrak{o}_k \sigma^k_1} + \sum_{i=1}^{|\sigma^k|-1} p^k_{\sigma^k_i \sigma^k_{i+1}} + p^k_{\sigma^k_{|\sigma^k|} \mathfrak{d}_k} \leq 1, \forall k \in K \tag{3.13}$$

$$tstart_k + \tau_{\mathfrak{o}_k \mathfrak{b}_k} \leq tbreak_k + M \cdot \left(1 - p^k_{\mathfrak{o}_k \sigma^k_1}\right), \forall k \in K \tag{3.14}$$

$$tbreak_k + \rho_k + \tau^k_{\mathfrak{b}_k \sigma^k_1} \leq t_{\sigma^k_1} + M \cdot \left(1 - p^k_{\mathfrak{o}_k \sigma^k_1}\right), \forall k \in K \tag{3.15}$$

$$t_{\sigma^k_i} + \pi_{\sigma^k_i} + \tau_{\sigma^k_i \mathfrak{b}_k} \leq tbreak_k + M \cdot \left(1 - p^k_{\sigma^k_i \sigma^k_{i+1}}\right), \forall i \in \{1, \ldots, |\sigma^k| - 1\}, \forall k \in K \tag{3.16}$$

$$tbreak_k + \rho_k + \tau_{\mathfrak{b}_k \sigma^k_j} \leq t_{\sigma^k_j} + M \cdot \left(1 - p^k_{\sigma^k_{j-1} \sigma^k_j}\right), \forall j \in \{2, \ldots, |\sigma^k|\}, \forall k \in K \tag{3.17}$$

$$t_{\sigma^k_{|\sigma^k|}} + \pi_{\sigma^k_{|\sigma^k|}} + \tau_{\sigma^k_{|\sigma^k|} \mathfrak{b}_k} \leq tbreak_k + M \cdot \left(1 - p^k_{\sigma^k_{|\sigma^k|} \mathfrak{d}_k}\right), \forall k \in K \tag{3.18}$$

$$tbreak_k + \rho_k + \tau_{\mathfrak{b}_k \mathfrak{d}_k} \leq tend_k + M \cdot \left(1 - p^k_{\sigma^k_{|\sigma^k|} \mathfrak{d}_k}\right), \forall k \in K \tag{3.19}$$

Si la pause est autorisé, les inégalités (3.13), (3.14), (3.15), (3.16), (3.17), (3.18) et (3.19) forcent chaque soignant à prendre au plus une pause et contraignent sa date de début.

$$tend_k - tstart_k \leq \theta^{max}_k + M \cdot \left(p^k_{\mathfrak{o}_k \sigma^k_1} + \sum_{i=1}^{|\sigma^k|-1} p^k_{\sigma^k_i \sigma^k_{i+1}} + p^k_{\sigma^k_{|\sigma^k|} \mathfrak{d}_k}\right), \forall k \in K \tag{3.20}$$

$$M \cdot \left[1 - \left(p^k_{\mathfrak{o}_k \sigma^k_1} + \sum_{i=1}^{|\sigma^k|-1} p^k_{\sigma^k_i \sigma^k_{i+1}} + p^k_{\sigma^k_{|\sigma^k|} \mathfrak{d}_k}\right)\right] + \theta^{max}_k \geq tbreak_k - tstart_k, \forall k \in K \tag{3.21}$$

$$tbreak_k - tstart_k \geq \theta^{min}_k - M \cdot \left[1 - \left(p^k_{\mathfrak{o}_k \sigma^k_1} + \sum_{i=1}^{|\sigma^k|-1} p^k_{\sigma^k_i \sigma^k_{i+1}} + p^k_{\sigma^k_{|\sigma^k|} \mathfrak{d}_k}\right)\right], \forall k \in K \tag{3.22}$$

$$M \cdot \left[1 - \left(p^k_{\mathfrak{o}_k \sigma^k_1} + \sum_{i=1}^{|\sigma^k|-1} p^k_{\sigma^k_i \sigma^k_{i+1}} + p^k_{\sigma^k_{|\sigma^k|} \mathfrak{d}_k}\right)\right] + \theta^{max}_k \geq tend_{kd} - tbreak_{kd} - \rho_k, \forall k \in K \tag{3.23}$$

Grâce aux inégalités (3.20), (3.21), (3.22) et (3.23), le début de la pause de chaque soignant interviendra entre son temps de travail minimal et maximal sans pause.

$$tend_k - tstart_k - \omega_k - \rho_k \cdot (p^k_{\mathfrak{o}_k \sigma^k_1} + \sum_{i=1}^{|\sigma^k|} p^k_{\sigma^k_i \sigma^k_{i+1}} + p^k_{\sigma^k_{|\sigma^k|} \mathfrak{d}_k}) \leq ot^{jour}_k, \forall k \in K \tag{3.24}$$

$$\sum_{k \in K} co_k(tend_k - tstart_k - \omega_k - \rho_k \cdot (p^k_{\mathfrak{o}_k \sigma^k_1} + \sum_{i=1}^{|\sigma^k|} p^k_{\sigma^k_i \sigma^k_{i+1}} + p^k_{\sigma^k_{|\sigma^k|} \mathfrak{d}_k}) \leq oc \tag{3.25}$$

Les inégalités (3.24) et (3.25) bornent à la fois les heures supplémentaires réalisées par chaque soignant k et comptent le coût total induit. Si une pause est autorisée, la durée de la pause n'est pas prise en compte dans le temps de travail journalier d'un soignant.

Contraintes d'intégrité :

$$tstart_k \geq 0, \forall k \in K$$

$$tend_k \geq 0, \forall k \in K$$

$$tbreak_k \geq 0, \forall k \in K$$

$$t_i \geq 0, \forall i \in \mathscr{S}_K$$

$$y_{ij} \in \{0,1\}, \forall i \in \mathscr{S}_K, \forall j \in \mathscr{S}_K$$

$$u^n_i \in \{0,1\}, \forall i \in \mathscr{S}_K, \forall n \in \{1, \ldots, \gamma_i\}$$

$$tard_i \geq 0, \forall i \in \mathscr{S}_K$$

$$tard^{max} \geq 0$$

$$oc \geq 0$$

Fonctions économiques. Les fonctions économiques sont identiques à celles formulées dans le chapitre 2. La seule fonction économique qui n'est pas prise en compte ici est le coût de sous-traitance. Le coût de sous-traitance est simplement déterminé grâce à l'ensemble des services qui sont affectés au soignant fictif k_f dans l'encodage.

3.4.2.2 Algorithme de résolution du problème de *timing* pour des cas particuliers

Nous présentons dans les lignes qui suivent un algorithme qui permet de déterminer les dates d'exécution des services d'un problème de *timing* avec les caractéristiques suivantes :

— une seule fenêtre de temps
— aucune pause n'est autorisée pour les soignant
— l'exclusion entre des services n'est pas prise en compte

Afin de déterminer ces dates d'exécution, nous commençons par définir un graphe de précédence qui constitue l'entrée de l'algorithme. Nous supposons qu'il n'existe pas d'interblocage dans le graphe.

Algorithm 5 Calcul des dates de début des services

Entrée : un graphe G_K orienté acyclique
Sortie : dates de début des services et faisabilité de la solution encodée
$tard^{max} \leftarrow 0$
for each $k \in K$ **do** ▷ prétraitement
  $tstart_k \leftarrow e_k$
  $t_{\sigma_1^k} \leftarrow \max(e^1_{\sigma_1^k}, tstart_k + \tau_{\mathfrak{o}_k \sigma_1^k})$
  for $i \leftarrow 2$ **to** $|\sigma^k|$ **do**
    $t_i \leftarrow \max(e^1_i, t_{\sigma^k_{i-1}} + \pi_{\sigma^k_i} + \tau_{\sigma^k_{i-1}\sigma^k_i})$
  end for
end for
$\mathscr{L} \leftarrow topologicalSort(G_K)$ ▷ $\mathscr{L}$: liste topologiquement triée des sommets de G_K
for each $s \in \mathscr{L}$ **do**
  Soit σ_i^k le service s situé à la i-ème position de la séquence σ^k
  if $i \geq 2$ **then**
    $t_s \leftarrow \max(e^1_{\sigma_i^k}, t_{\sigma^k_{i-1}} + \pi_{\sigma^k_i} + \tau_{\sigma^k_{i-1}\sigma^k_i})$
    if $\exists\, r \in \mathscr{S}_K \mid synchro(r, s)$ **then**
      ▷ si le service courant est un service synchronisé
      $\mathscr{M} \leftarrow \{s\} \cup \{s' \in \mathscr{S}_K \mid synchro(s, s')\}$
      for each $s' \in \mathscr{M}$ **do**
        $t_{s'} \leftarrow \max_{u \in \mathscr{M}} \{t_u\}$
      end for
    else
      if $\exists\, r \in \mathscr{S}_K \mid delay(r, s)$ **then**
        ▷ si le service courant est lié à un autre par un délai
        for each $s' \mid \exists \delta^{min}_{ss'}$ **do**
          $t_{s'} \leftarrow \max(t_s + \delta^{min}_{ss'}, l^1_{s'})$
        end for
        for each $s' \mid \exists \delta^{max}_{ss'}$ **do**
          if $t_s + \delta^{max}_{ss'} < t_{s'}$ **then**
            $t_s \leftarrow t_{s'} - \delta^{max}_{ss'}$
          end if
        end for
      end if
    end if
    if $TARD$ **then**
      $tard_s \leftarrow \max(t_s - l^1_j, 0)$
      $tard^{max} \leftarrow \max(tard^{max}, tard_{\sigma_i^k})$
    else
      if $t_s > l^1_j$ **then return** $\perp$ **end if**
    end if
  end if
end for
for each $k \in K$ **do**
  $tend_k \leftarrow t_{\sigma^k_{|\sigma^k|}} + \pi_{\sigma^k_{|\sigma^k|}} + \tau_{\sigma_1^k \mathfrak{d}_k}$
end for
return $\top$

Définition 13. *Soit un graphe de précédence acyclique $G_K = (V_K, A_K)$. L'ensemble des sommets de G_K est constitué de l'ensemble des services $\mathscr{S}_K$. L'ensemble des arcs du graphe G_K est composé :*

- *des arcs de précédence de chaque séquence σ_k*
  $(\sigma_i^k, \sigma_{i+1}^k)$, $\forall i \in \{1, \ldots, |\sigma^k| - 1\}, \forall k \in K$
- *des arcs liant deux services liés par une contrainte de délai minimum*
  (r, s), $\forall r \in \mathscr{S}_K, \forall s \in \mathscr{S}_K \mid \exists \delta_{rs}^{min} > 0$
- *des arcs liant deux services liés par une contrainte de délai maximum*
  (r, s), $\forall r \in \mathscr{S}_K, \forall s \in \mathscr{S}_K \mid \exists \delta_{rs}^{max} > 0$
- *des arcs liant le prédécesseur d'un service r au service s avec lequel il est synchronisé. Si on note ce prédécesseur r'*
  (r', s), $\forall r \in \mathscr{S}_K, \forall s \in \mathscr{S}_K \mid synchro(r', s)$
- *des arcs liant un service s synchronisé avec un service r au successeur de ce dernier. Si on note ce successeur s'*
  (r, s'), $\forall i \in \mathscr{S}_K, \forall j \in \mathscr{S}_K \mid synchro(i, j)$
- *des arcs liant le prédécesseur d'un service r au service s avec lequel il est lié par une contrainte de délai. Si on note ce prédécesseur r'*
  (r', s), $\forall r \in \mathscr{S}_K, \forall s \in \mathscr{S}_K \mid delay(r', s)$
  ces arcs ne sont pas créés s'il existe un cycle dans le graphe d'interblocage défini à la sous-section 3.3.4.6
- *des arcs liant un service s au successeur d'un service r avec lequel il est lié par une contrainte de délai. Si on note ce successeur s'*
  (r, s'), $\forall i \in \mathscr{S}_K, \forall j \in \mathscr{S}_K \mid delay(i, j)$
  ces arcs ne sont pas créés s'il existe un cycle dans le graphe d'interblocage défini à la sous-section 3.3.4.6

L'algorithme détermine des dates de début des services lorsqu'une seule fenêtre de temps est associée à chaque service, des contraintes de délai lient certains services et les heures supplémentaires ne sont pas autorisées. Cet algorithme couvre par conséquent les problèmes de *timing* pouvant exister dans de nombreux modèles de la littérature notamment ceux de [Bredström et Rönnqvist, 2008], [Mankowska *et al.*, 2014] et [Di Gaspero et Urli, 2014]. L'algorithme prend en entrée le graphe G_K et la variable booléenne $TARD$. Ces variables informent l'algorithme de l'autorisation ou non d'heures supplémentaires et du retard, respectivement. L'algorithme en question est simplement un algorithme de calcul de date d'exécution au plus tôt.

3.5 Expérimentations

Afin d'évaluer les algorithmes proposés dans ce chapitre, un échantillon de 100 instances a été choisi. Ces instances sont représentatives des instances dont nous disposions en tenant compte de leurs tailles et des contraintes du problème. Cet échantillon est constitué de deux groupes distincts :

- le premier groupe est composé de 30 instances pour lesquelles il n'y a pas de contraintes de qualification
- le deuxième groupe est composé de 70 instances pour lesquelles les qualifications

ont été prises en compte

Cette distinction a été mise en place car dans le premier groupe, la dominance entre deux étiquettes peut s'effectuer uniquement par valeur ; tandis que dans le deuxième groupe, la dominance doit également prendre en compte l'ensemble des tournées utilisées. Pour chaque instance de l'échantillon, 10 séquences réalisables ont été générées. Au total, 1000 séquences ont été générées. Ces séquences ont été obtenues à partir d'une heuristique de meilleure insertion randomisée, couplée à une recherche locale.

Les algorithmes présentés dans ce chapitre ont été implémentés en C++ et exécutés sur un processeur Xeon de 2.00GHz. Les problèmes de *timing* NP-difficile ont été résolus à l'aide de CPLEX (12.5.1). Ces algorithmes d'évaluation étant appelés plusieurs milliers de fois dans une méta-heuristique, un temps de calcul faible permet d'évaluer plus de solutions. Enfin, pour une séquence, au-delà de 60 secondes, l'algorithme est arrêté.

Séquences	$\lvert\mathscr{K} \times \mathscr{H}\rvert$	$\lvert\mathscr{S}\rvert$	Split sans inégalités valides		Split avec inégalités valides	
			Temps moyen (s)	Nombre moyen d'étiquettes	Temps moyen (s)	Nombre moyen d'étiquettes
150	4	20	0.002726	112	0.001760	43
90	10	50	0.042572	857	0.012883	152
60	16	80	0.143272	1943	0.036804	240

TABLE 3.1 – Tableau de résultats : Split avec inégalités valides sans bornes (Algorithme 1)

Le tableau 3.1 présente les résultats expérimentaux pour le premier groupe d'instances. Les trois premières colonnes correspondent respectivement au nombre de séquences évaluées, au nombre de tournées possibles pour ces séquences et à la taille de ces séquences. Les quatre dernières colonnes représentent pour chaque algorithme testé, le temps moyen nécessaire pour décoder une séquence et le nombre moyen d'étiquettes générées. Deux versions de l'algorithme 1 ont été utilisées pour décoder ces séquences :

— avec techniques d'amélioration et avec inégalités valides (3.3.4.1, 3.3.4.2, 3.3.4.3, 3.3.4.5, 3.3.4.6)

— sans techniques d'amélioration et sans inégalités valides

Cependant, nous n'avons pris en compte les bornes pour la première version car elles n'ont aucun impact dans la réduction du temps de calcul. Les algorithmes 4 et 3 qui sont des heuristiques n'ont pas été utilisés pour décoder ces séquences. En effet, l'algorithme 1 qui est un algorithme exact décode ces séquences très rapidement. Comme on pouvait s'y attendre, le nombre d'étiquettes générées et le temps de calcul sont plus importants pour l'algorithme 1 sans inégalités valides qu'avec inégalités valides. Cela est illustré dans la figure 3.7 pour les instances 10L, 10M et 10S, qui montre l'évolution du nombre d'étiquettes générées pour chaque sommet du graphe auxiliaire H.

Les tableaux 3.2, 3.3, 3.4 et 3.5 présentent les résultats expérimentaux pour le deuxième groupe d'instances en utilisant respectivement l'algorithme 1 sans les inégalités valides, l'algorithme 1 avec les inégalités valides, l'algorithme 3 et l'algorithme 4. Les trois premières colonnes correspondent respectivement au nombre de séquences évaluées, au nombre de tournées possibles pour ces séquences et à la taille de ces séquences. Nous présentons dans ces tableaux le nombre de séquences décodées, le temps moyen de décodage (pour les

séquences décodées), le nombre moyen d'étiquettes générées (pour les séquences décodées) et le nombre de fois où l'algorithme a obtenu le meilleur décodage. Pour les séquences générées pour les instances du deuxième groupe, il est beaucoup plus difficile de les décoder à cause de la dominance qui tient en plus compte des tournées utilisées. Pour de petites séquences avec des tailles inférieures à 33, l'algorithme 1 (tableaux 3.2 et 3.3) réussit à les décoder. Malheureusement, au-delà, le nombre d'étiquettes générées est trop important pour décoder une séquence dans le temps imparti. La figure 3.8 illustre cette explosion pour les instances B01 et C01. Dans le premier graphique, grâce aux inégalités valides, l'explosion du nombre d'étiquettes est contrôlée. Par contre dans le deuxième graphique, à partir du 4-ème sommet (sans inégalités valides) et du 35-ème sommet (avec inégalités valides), le nombre d'étiquettes croît de façon exponentielle. Pour de grandes séquences dont la taille est supérieure à 33, seuls les algorithmes 3 et 4 réussissent à décoder les séquences dans le temps imparti. L'algorithme 3 est bien meilleur que l'algorithme 4 sauf pour le nombre de séquences décodées sur de petites instances. Pour l'algorithme 3, plus de la moitié du nombre de séquences est décodé dans des temps de calcul de quelques secondes.

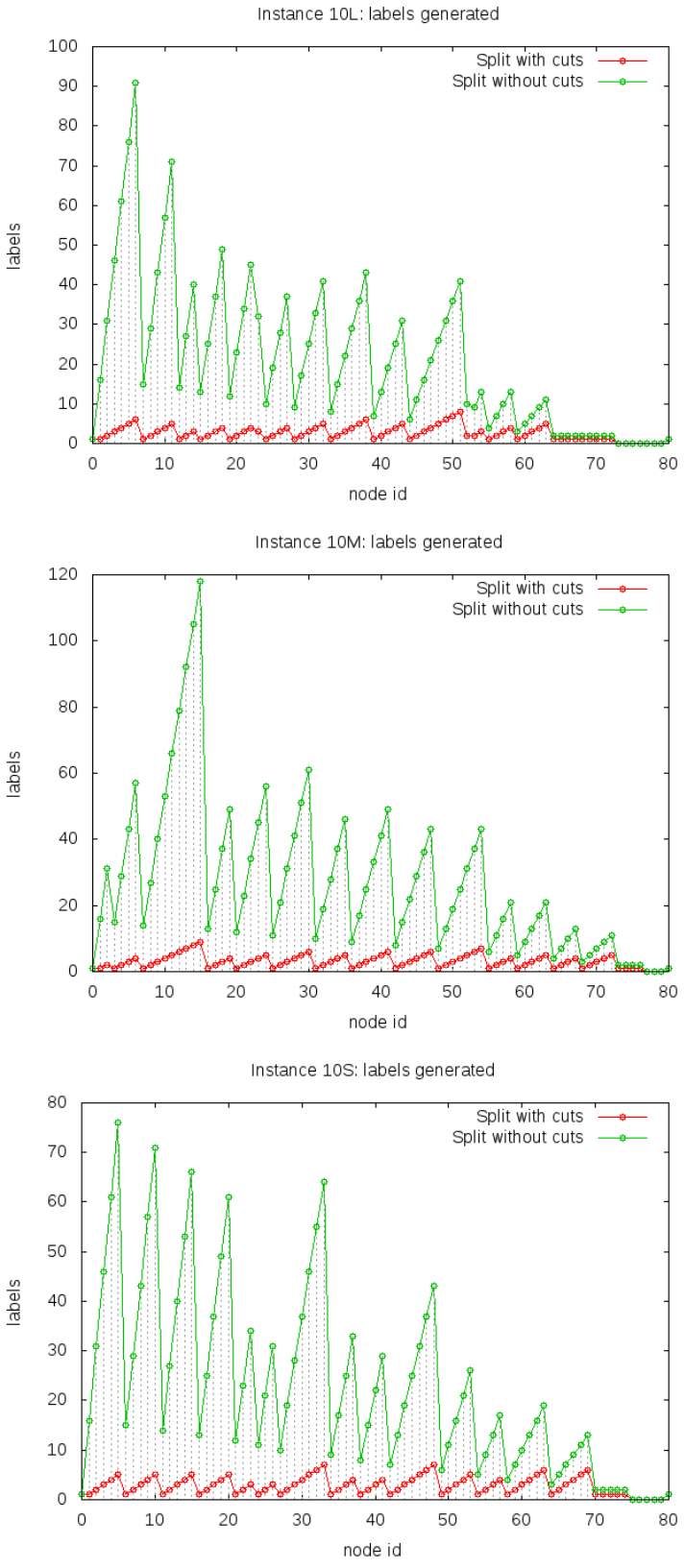


FIGURE 3.7 – Nombre d'étiquettes générées par sommet pour les instances 10S, 10L, 10M

Séquences	$\lvert\mathscr{K} \times \mathscr{H}\rvert$	$\lvert\mathscr{S}\rvert$	Split sans inégalités valides			
			Nombre de séquences décodées	Temps moyen (s)	Nombre moyen d'étiquettes	Nombre de séquences décodées Best
100	3	13	100	0.0847513	36	100
100	5	33	43	25.9049	3611	43
100	10	65	-	-	-	-
100	15	98	-	-	-	-
100	20	130	-	-	-	-
100	30	260	-	-	-	-
100	40	400	-	-	-	-

TABLE 3.2 – Tableau de résultats : Split sans inégalités valides (Algorithme 1)

Séquences	$\lvert\mathscr{K} \times \mathscr{H}\rvert$	$\lvert\mathscr{S}\rvert$	Split avec inégalités valides			
			Nombre de séquences décodées	Temps moyen (s)	Nombre moyen d'étiquettes	Nombre de séquences décodées Best
100	3	13	100	0.00082914	15	100
100	5	33	100	0.0421067	594	100
100	10	65	-	-	-	-
100	15	98	-	-	-	-
100	20	130	-	-	-	-
100	30	260	-	-	-	-
100	40	400	-	-	-	-

TABLE 3.3 – Tableau de résultats : Split avec inégalités valides sans bornes (Algorithme 1)

Séquences	$\lvert\mathscr{K} \times \mathscr{H}\rvert$	$\lvert\mathscr{S}\rvert$	Split parcours en profondeur			
			Nombre de séquences décodées	Temps moyen (s)	Nombre moyen d'étiquettes	Nombre de séquences décodées Best
100	3	13	100	0.0140281	5	73
100	5	33	70	0.090503	19	48
100	10	65	55	4.00779	279	55
100	15	98	49	0.954538	108	49
100	20	130	51	0.989222	93	51
100	30	260	83	4.37774	224	83
100	40	400	82	10.8825	397	82

TABLE 3.4 – Tableau de résultats : Split parcours en profondeur (Algorithme 3)

Séquences	$\lvert\mathscr{K} \times \mathscr{H}\rvert$	$\lvert\mathscr{S}\rvert$	Split heuristique			
			Nombre de séquences décodées	Temps moyen (s)	Nombre moyen d'étiquettes	Nombre de séquences décodées Best
100	3	13	100	0.411103	*	0
100	5	33	100	3.88797	*	0
100	10	65	100	22.5896	*	0
100	15	98	15	41.2093	*	0
100	20	130	-	-	*	-
100	30	260	-	-	*	-
100	40	400	-	-	*	-

TABLE 3.5 – Tableau de résultats : Split heuristique (Algorithme 4)

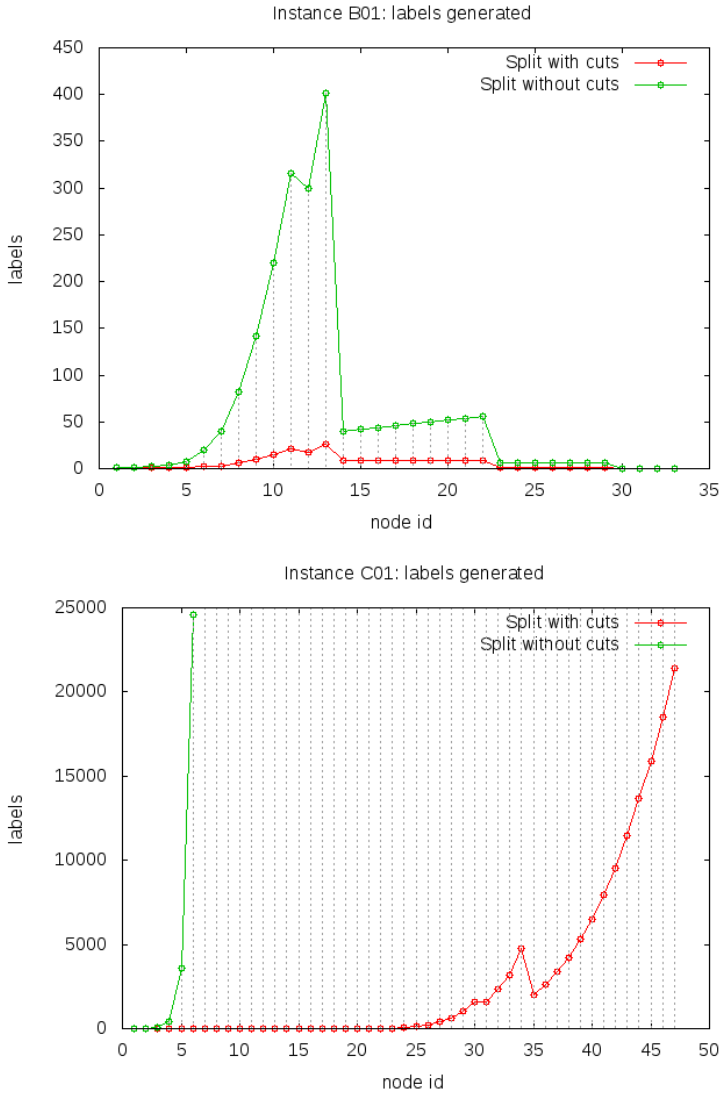


FIGURE 3.8 – Nombre d'étiquettes générées par sommet pour les instances B01 et C01

3.6 Conclusion

Dans ce chapitre, deux encodages ont été proposés. Le premier est un encodage direct représentant une solution comme un tour géant sans délimiteurs. L'algorithme de décodage associé est une extension au HCRSP de l'algorithme Split. Des inégalités valides sont présentées pour réduire la complexité de l'algorithme. Le deuxième encodage correspond à un encodage direct et l'algorithme de décodage consiste en la résolution d'un problème de *timing*. Enfin, des expérimentations numériques ont été conduites pour valider l'approche proposée. L'ensemble des contributions de ce chapitre s'inscrivent dans le processus d'une méthode de résolution générique pour le HCRSP.

Chapitre 4

Méthode par décomposition

Dans le chapitre précédent, deux encodages et plusieurs algorithmes de décodage ont été proposés. Ces propositions sont des extensions d'encodages et d'algorithmes issus de la littérature du VRP. Ce chapitre introduit une première méthode de résolution approchée du HCRSP. Cette méthode de résolution est un algorithme de décomposition accompagné d'un processus d'amélioration. Elle utilise l'encodage direct et les algorithmes de décodage du chapitre précédent.

Ce chapitre s'organise comme suit : la section 4.1 met en évidence l'ensemble des contributions. La section 4.2 décrit l'architecture générale de la méthode en l'illustrant à l'aide d'un diagramme ; la section 4.3 est consacrée à la description des différentes phases qui conduisent à une solution initiale ; la section 4.4 décrit un processus d'amélioration mêlant à la fois intensification et diversification ; la section 4.5 rassemble les expérimentations numériques et la dernière section résume le travail effectué dans ce chapitre et le clôture.

4.1 Contributions

Les contributions qui suivent constituent une première méthode de résolution pour le HCRSP. Nos contributions sont les suivantes :

- une méthode décomposant le problème en sous-problèmes en les résolvant séparément ;
- des procédures de retour sur trace faisant office de processus de diversification ;
- une recherche locale permettant d'explorer le voisinage d'une solution et faisant office de processus d'intensification ;
- des expérimentations numériques validant notre approche en comparant les résultats obtenus à ceux de la littérature

4.2 Architecture générale de la méthode

La méthode de résolution du HCRSP que nous proposons se décompose en trois phases distinctes. Chaque phase sera résolue de manière exacte ou approchée en fonction de sa difficulté. L'idée est de décomposer le problème en plusieurs sous-problèmes résolus de

manière indépendante. Nous essayons par ce biais d'amoindrir la difficulté de résolution du problème. Les trois phases de la méthode sont :

— la répartition de la charge de travail sur l'horizon de planification ;
— l'affectation des services aux soignants ;
— la planification horaire des soignants et des services.

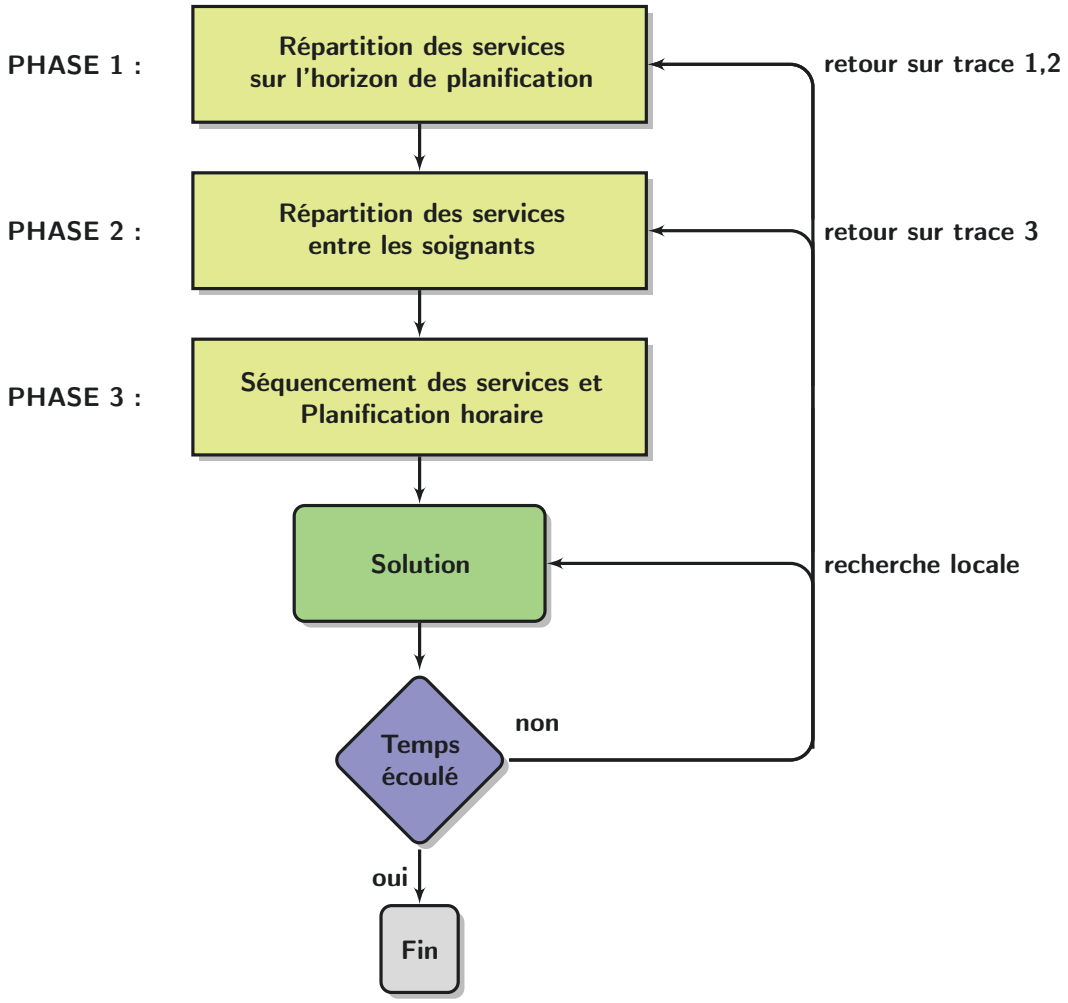


FIGURE 4.1 – Diagramme de la méthode en trois phases

La figure 4.1 illustre l'architecture générale de la méthode. Les trois phases y sont séparées de sorte à les distinguer aisément et le processus conduisant à l'amélioration de la solution courante y est également schématisé. Le résultat de chaque phase de la méthode correspond aux données d'entrée de la phase qui suivra jusqu'à obtenir une solution initiale. À partir de cette dernière, trois procédures de retour sur trace et un ensemble d'opérateurs de voisinage pour la recherche locale ont été proposées pour l'améliorer. Ces procédures de retour sur trace reviennent sur les décisions prises au préalable dans les phases ayant conduit à la construction d'une solution et permettent ainsi d'explorer d'autres zones de l'espace de solutions. Quant à la recherche locale, elle permet d'explorer l'espace proche de la solution courante à l'aide d'opérateurs classiques issus de la littérature. L'ensemble

du processus est réitéré tant que le temps total imparti n'est pas écoulé ou que le nombre d'itérations n'est pas épuisé. À l'issue de ce processus, la meilleure solution obtenue est renvoyée.

4.3 Construction d'une solution initiale

Dans cette section, les trois phases de la méthode sont décrites. Pour chaque phase, un ensemble de décisions est pris sans remettre en cause celles prises à la phase précédente. Ces trois phases se décomposent comme suit :

- la répartition des services sur l'horizon de planification : elle permet de déterminer le jour pour lequel chaque service est effectué.
- la répartition des services entre les soignants : elle permet de déterminer un unique soignant par service à effectuer ;
- le séquencement des services et la planification horaire : enfin, cette phase permet d'une part de déterminer l'ordre dans lequel les services seront réalisés et d'autre part, de résoudre le problème de *timing* induit.

4.3.1 PHASE 1 : Répartition des services sur l'horizon de planification

Dans cette première phase, il s'agit de déterminer le jour de réalisation de chaque service tout en respectant l'ensemble des contraintes liées à la planification journalière. Cette phase n'est donc pas nécessaire pour un horizon de planification d'une seule journée. Une description formelle de cette phase est donnée par l'algorithme 6. Cette première phase se décompose en deux étapes :

- la première étape consiste à trier les services en favorisant les services *critique*s. La notion de service *critique* sera précisée au fil de la description de cette étape.
- à partir du résultat de la première étape, la deuxième étape détermine le jour le plus adéquat pour réaliser chaque service.

Nous commençons par introduire quelques définitions qui sont réutilisées tout au long de cette section.

Définition 14. *Le nombre d'heures qualifiées disponibles pour la réalisation d'un service correspond à la somme des durées réglementaires de travail des soignants habilités à réaliser ce service pour tous les jours d où le service i peut être réalisé par le soignant k :*

$$QAH(i) = \sum_{d \in \mathscr{H}_k \cap \mathscr{H}_i} \sum_{k \in \mathscr{K} \wedge\ q_{ik} = \top} \omega_{kd}$$

Définition 15. *Le nombre de dépendances sur l'horizon de planification d'un service correspond au nombre de services qui devront être réalisés Δ_{ij}^{min} et/ou Δ_{ij}^{max} jours après le service i :*

$$NDT(i) = |\{j \in \mathscr{S} \mid \Delta_{ij}^{min} > 0 \vee \Delta_{ij}^{max} > 0\}|$$

4.3.1.1 Étape 1

Le nombre d'heures qualifiées disponible permet de distinguer les services *critique*s de ceux qui ne le sont pas. En effet, les services pour lesquels le nombre d'heures qualifiées dis-

ponibles est faible représentent donc les services disposant de peu de soignants disponibles pour les réaliser et peuvent donc être considérés comme *critique*s. Il est donc préférable de les planifier en priorité. Par contre, les services disposant de nombreux soignants disponibles pour les réaliser sont planifiés après car la planification de ces services est moins contraignante. Lorsque deux services disposent du même nombre d'heures qualifiées disponibles, un deuxième critère est utilisé pour les hiérarchiser : le nombre de dépendances sur l'horizon. En utilisant ce critère, il s'agit de favoriser les services dont de nombreux autres dépendent de sa planification. Si les deux premiers critères décrits ne suffisent pas à hiérarchiser deux services, la durée de réalisation sera utilisée. La procédure de tri lexicographique des services selon ces 3 critères est définie à la ligne 6 de l'algorithme 6.

4.3.1.2 Étape 2

À partir de l'ordre obtenu dans la première étape, l'étape courante consiste à déterminer pour chaque service le jour de planification le plus adapté. Pour les services liés à d'autres par une contrainte de synchronisation ou de délai en heures, le jour de réalisation de tous ces services est déterminé en même temps. L'ensemble constitué du service i et de tous les services devant être planifiés le même jour est noté $\mathscr{S}'$ comme indiqué (ligne 7 de l'algorithme 6). Ce jour devra être déterminé dans l'ensemble $\mathscr{H}_{\mathscr{S}'}$ des jours de disponibilité. Pour planifier ces services, un graphe de planification est construit pour chaque jour $d \in \mathscr{H}_{\mathscr{S}'}$. Le graphe de planification est conçu comme suit.

Définition 16. *(Graphe de planification). Soient un ensemble $\mathscr{S}' \subset \mathscr{S}$ de services devant être réalisés le même jour et l'ensemble $\mathscr{H}_{\mathscr{S}'}$ des jours possibles de réalisation de ces services. Pour chaque jour $d \in \mathscr{H}_{\mathscr{S}'}$, un graphe $G_d^{\mathscr{S}'} = (V, A)$ graphe orienté est construit avec l'ensemble de ses sommets noté V et l'ensemble de ses arcs noté A. Chaque arc est caractérisé par une capacité et un coût. Ce graphe constitue le graphe de planification pour un jour donné.*

L'ensemble des sommets du graphe est constitué de :

- *un sommet source s ;*
- *un sommet puits t ;*
- *un sommet pour chaque service dans l'ensemble $\mathscr{S}_d$ des services dont le jour de réalisation a déjà été déterminé au préalable le jour d ;*
- *un sommet pour chaque service dans l'ensemble $\mathscr{S}'$;*
- *un sommet pour chaque soignant de l'ensemble $\mathscr{K}_d$. L'ensemble $\mathscr{K}_d$ étant l'ensemble des soignants disponibles le jour d et habilité à réaliser au moins un service de l'ensemble $\mathscr{S}_d \cup \mathscr{S}'$;*
- *k_f, un soignant fictif ;*

L'ensemble des arcs de $G_d^{\mathscr{S}'}$ est constitué :

- *d'arcs reliant le sommet source à tous les services : (s, v), $\forall v \in \mathscr{S}_d \cup \mathscr{S}'$. La capacité de chaque arc correspond à la durée nécessaire pour réaliser le service concerné. Ces arcs ont un coût nul ;*
- *d'arcs reliant chaque soignant (y compris le soignant fictif) au puits : (u, t), $\forall u \in \mathscr{K}_d \cup \{k_f\}, v \in \{t\}$. À chaque arc est associé une capacité correspondant à la durée légale de travail ω_{kd} du soignant concerné. Le coût sera identique pour tous ces arcs et égal à 1. En ce qui concerne l'arc reliant le soignant fictif k_f au puits, une capacité*

infinie et un coût M (une valeur arbitraire très grande) lui sont associés ;
— *d'arcs reliant chaque service aux soignants disposant des qualifications suffisantes pour le réaliser :* $(u, v) \in \mathscr{S}_d \cup \mathscr{S}' \times \mathscr{K}_d \cup \{k_f\} \mid q_{uv} = \top$. *Le soignant fictif étant capable de réaliser tous les services, il existera un arc reliant chaque service au soignant fictif ;*

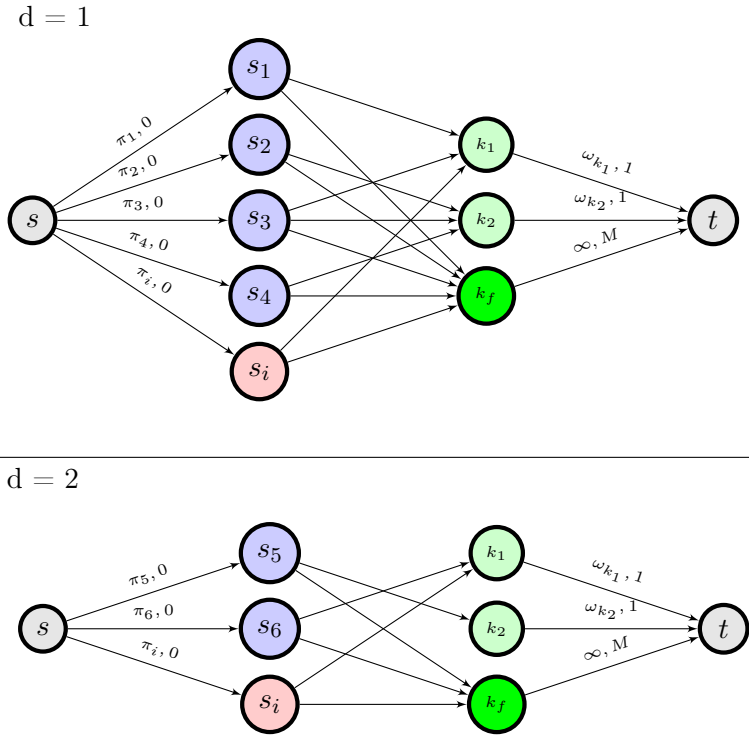


FIGURE 4.2 – Graphes de planification du service i pour les jours 1 et 2

Pour un jour donné, ce graphe de planification représente la répartition de la charge de travail. En fonction des disponibilités journalières du personnel, l'idée est de répartir la charge de travail équitablement selon la capacité de chaque jour. Pour cela, un problème de flot maximum à coût minimal est résolu chaque jour. Étant donné le graphe $G_d^{\mathscr{S}'}$, $min \sum_{(u,v)\in A} c(u, v) \cdot f(u, v)$ correspond au coût minimal du problème de flot maximum depuis la source s vers le puits t sous contrainte de capacité de chaque arc, avec $c(u, v)$, étant le coût associé à l'arc (u, v). Ce coût minimal sera égal au temps total des services si aucun service n'est sous-traité. Le jour de réalisation de l'ensemble des services $\mathscr{S}'$ correspond au jour $d \in \mathscr{H}_{\mathscr{S}'}$ dont le coût minimal du flot maximum dans le graphe $G_d^{\mathscr{S}'}$ est le plus faible. Cela traduit que planifier l'ensemble des services $\mathscr{S}'$ est plus faible pour ce jour, en tenant compte à la fois des services déjà planifiés à ce jour et de la capacité de travail disponible selon les qualifications. L'algorithme 6 met à jour (lignes 18-25) l'ensemble des jours de disponibilité des services exécutés après les services appartenant à l'ensemble $\mathscr{S}'$. Un service ayant des contraintes de précédence journalière avec d'autres services sera

Algorithm 6 Répartition des services sur l'horizon de planification

for $i \in \mathscr{S}$ **do**
  $d_i \leftarrow \varnothing$ ▷ initialisation
end for
$\mathscr{L} \leftarrow \mathscr{S} \setminus \{j \in \mathscr{S} \mid \exists i \in \mathscr{S}, \Delta_{ij}^{min} > 0 \vee \Delta_{ij}^{max} > 0\}$
while $|\mathscr{L}| \neq 0$ **do**
  $i \leftarrow lexmin_{j \in \mathscr{L}}(QAH(j), -NDT(j), p_j)$ ▷ tri lexicographique
  $\mathscr{S}' \leftarrow \{i\} \cup \{j \in \mathscr{L} \mid \Delta_{ij}^{min} = \Delta_{ij}^{max} = 0\}$ ▷ ensemble des services à planifier
  $\mathscr{L} \leftarrow \mathscr{L} \setminus \mathscr{S}'$
  $\phi_{best} \leftarrow +\infty$
  for $d \in \mathscr{H}_{\mathscr{S}'}$ **do**
    Construire le graphe $G_d^{\mathscr{S}'} = (V, A)$
    $\phi \leftarrow min \sum_{(u,v) \in A} c(u,v) \cdot f(s,v)$ ▷ calcul du coût min du graphe $G_d^{\mathscr{S}'}$
    if $\phi < \phi_{best}$ **then**
      $d_h \leftarrow d, \forall h \in \mathscr{S}'$
      $\phi_{best} \leftarrow \phi$
    end if
  end for
  for $h \in \mathscr{S}'$ **do**
    for $j \in \mathscr{S} \mid \Delta_{h,j}^{min} > 0$ **do**
      $\mathscr{H}_j \leftarrow \{d \in \mathscr{H}_j \mid d \geq d_h + \Delta_{hj}^{min}\}$ ▷ mise à jour des jours de disponibilité
    end for
    for $j \in \mathscr{S} \mid \Delta_{h,j}^{max} > 0$ **do**
      $\mathscr{H}_j \leftarrow \{d \in \mathscr{H}_j \mid d \leq d_h + \Delta_{hj}^{max}\}$ ▷ mise à jour des jours de disponibilité
    end for
  end for
  $\mathscr{L} \leftarrow \mathscr{L} \cup \{j \in \mathscr{S} \mid d_j = \varnothing \wedge \nexists h \in \mathscr{L}, \Delta_{hj}^{min} > 0 \ \vee \ \Delta_{hj}^{max} > 0\}$
end while

planifié lorsque ces derniers auront été planifiés. Ainsi, certains jours qui conduiraient à une solution irréalisable sont supprimés.

La figure 4.2 illustre un exemple. Pour un horizon de planification de deux jours, quatre services $\{s_1, s_2, s_3, s_4\}$ ont déjà été planifiés le premier jour et deux autres $\{s_5, s_6\}$ le deuxième jour. Les deux soignants sont disponibles tous les jours de l'horizon de planification. Évidemment, si l'un des deux soignants n'était pas disponible un jour donné, il n'apparaîtrait pas dans le graphe correspondant. Enfin, les arcs reliant les services et les soignants modélisent les contraintes de qualification. Comme nous pouvons le constater, le soignant k_1 est habilité à réaliser les services $\{s_1, s_3, s_6, s_i\}$. Quant au soignant k_2, il est habilité à réaliser les services $\{s_2, s_3, s_4, s_5, s_6, s_i\}$. Le jour de planification du service s_i doit être déterminé. Pour cela, on choisit le jour correspondant au graphe de coût minimum le plus faible.

4.3.2 PHASE 2 : Répartition des services entre les soignants

Dans la section précédente, le jour de réalisation de chaque service a été déterminé. Pour un service i donné, ce jour de planification est noté d_i. La deuxième phase a pour objectif de déterminer l'ensemble des services que chaque soignant réalisera en tenant compte des contraintes du problème. Tout comme la phase précédente, cette phase est décomposée en deux étapes. Une première étape consiste à satisfaire les contraintes de continuité de soins de chaque patient. Elle détermine par conséquent pour un patient donné, l'ensemble des soignants éligibles pour réaliser les services que nécessite ce patient. La deuxième étape permettra d'obtenir le soignant adéquat qui réalisera chaque service. Ce soignant est choisi parmi l'ensemble des soignants assurant la continuité des soins du patient nécessitant ce service.

4.3.2.1 Étape 1 : Prise en compte de la continuité des soins.

Avant de déterminer le soignant qui réalisera un service, nous devons nous assurer que ce soignant satisfait les contraintes de continuité des soins du patient qui nécessite ce service. L'étape 1 a pour but de déterminer un ensemble de soignants satisfaisant les contraintes de continuité des soins de chaque patient. Cette étape est réalisée en utilisant un programme linéaire en nombres entiers. Pour un patient donné, l'idée est de choisir parmi les soignants ayant les qualifications pour réaliser au moins un service de ce patient, la combinaison de soignants dont le cumul du temps de travail maximum légal est le plus élevé. Bien évidemment, cette étape n'est pas nécessaire si la continuité des soins n'est pas prise en compte dans le problème.

Avant de proposer une formulation, nous commençons par décrire le problème. Certaines données décrites sont un rappel de celles introduites au chapitre 2.

Soit un patient p nécessitant des services regroupés dans l'ensemble $\mathscr{S}_p$. Chaque service i doit être réalisé un jour $d_i \in \mathscr{H}_k$ de l'horizon de planification. Durant la première phase, certains services de l'ensemble $\mathscr{S}_p$ ont pu être sous-traités. Si pour un patient donné, il existe au moins un service sous-traité dans la phase 1, la donnée $P1$ prendra la valeur 1. Sinon, $P1$ vaudra 0. Ces services seront réalisés par l'ensemble des soignants dont le temps

de travail maximum légal du soignant k le jour d est noté ω_{kd}. Enfin, si un soignant k donné est qualifié pour réaliser un service i, la donnée q_{ik} vaudra 1 ; sinon 0.

Afin de déterminer l'ensemble des soignants satisfaisant la continuité des soins de chaque patient, nous introduisons les variables de décision suivantes :

$$x_k = \begin{cases} 1 & \text{si le soignant } k \text{ est sélectionné pour appartenir l'ensemble des} \\ & \text{soignants satisfaisant la continuité des soins du patient } p \\ 0 & \text{sinon} \end{cases}$$

Pour chaque patient p, nous cherchons à maximiser le cumul du temps de travail maximum légal des soignants pouvant réaliser au moins un service de ce patient. Donc, pour un service $i \in \mathscr{S}_p$, ces soignants doivent être disponibles et qualifiés (contraintes 4.2) pour réaliser ce service. Il faut en plus s'assurer que parmi l'ensemble des soignants choisis, le nombre maximum de nouveaux soignants autorisés à soigner le patient en question soit inférieur à ζ_p (contraintes 4.3).

$$\max \sum_{i \in \mathscr{S}_p} \sum_{k \in \mathscr{K} | d_i \in \mathscr{H}_k} \omega_{kd_i} \cdot q_{ik} \cdot x_k \tag{4.1}$$

$$\text{s.t.} \quad \sum_{k \in \mathscr{K} | d_i \in \mathscr{H}_k} q_{ik} \cdot x_k \geq 1 \qquad \forall i \in \mathscr{S}_p \tag{4.2}$$

$$P1 + \sum_{k \in \overline{\mathscr{K}}_p} x_k \leq \zeta_p \tag{4.3}$$

$$x_k \in \{0,1\} \qquad \forall k \in \mathscr{K} \tag{4.4}$$

Pour chaque patient p nécessitant de plus d'un service, le programme linéaire en nombre entiers ci-dessus est résolu à l'aide d'un solveur de programmation mathématique. Dans cette étude, le solveur choisi est CPLEX. Comme nous pouvons le constater, la formulation de ce problème est très proche du problème de couverture par ensembles qui est NP-difficile.

4.3.2.2 Étape 2 : Affectation d'un soignant à la réalisation d'un service

L'étape précédente a déterminé l'ensemble des soignants qui seront amenés à réaliser les services pour un patient tout au long de l'horizon de planification en respectant les contraintes de continuité de soins. Pour un patient p donné, nous noterons cet ensemble de soignants Φ_p. Cette deuxième étape a pour objectif d'identifier pour chaque service le soignant le plus adéquat pour le réaliser. Nous proposons ici de répartir les services parmi les soignants en tenant à la fois compte du temps de travail légal maximum autorisé et du temps de trajet nécessaire pour rejoindre les patients. Ainsi, une répartition équitable de la charge travail est recherchée.

Pour réaliser cette répartition, deux programmes linéaires en nombres entiers sont proposés. Le premier programme est résolu et si la solution obtenue est irréalisable, le second programme est résolu. Nous commençons ici par définir l'ensemble des données et variables qui sont utilisées dans la formulation des deux programmes.

Soit un jour donné $d \in \mathscr{H}$ pour lequel un ensemble de services a été planifié. Cet ensemble de services que nous notons $\mathscr{S}_d$, a été déterminé dans la première phase. Pour ce jour, des soignants sont disponibles pour réaliser ces services. Pour chaque soignant k, son temps de travail maximum légal est ajusté en lui retirant le temps nécessaire pour se rendre chez le patient le plus proche. Ce nouveau temps de travail maximum est noté ω'_{kd} et vaut :

$$\omega'_{kd} = \omega_{kd} - \min_{i \in \mathscr{S}_d} \left\{ \tau_{\mathfrak{o}_k i} \mid q_{ik} = \top \right\}$$

De même, le temps π_i nécessaire pour la réalisation d'un service i est actualisé en tenant compte du temps de trajet nécessaire pour rendre au domicile du patient nécessitant ce service par un soignant k donné. Ainsi, nous obtenons :

$$\pi_i'^k = \pi_i + \min \left(\tau_{\mathfrak{b}_{kd} i}, \min_{j \in \mathscr{S}_d} \left\{ \tau_{ij} \mid i \neq j \ \wedge \ q_{ik} = q_{jk} = \top \right\} \right)$$

Les variables de décision sont les suivantes :

$$x_{ki} = \begin{cases} 1 & \text{si le soignant } k \text{ est choisi pour réaliser le service } i \\ 0 & \text{sinon} \end{cases}$$

Pour ces variables de décisions, seuls les soignants satisfaisant la continuité des soins d'un patient p nécessitant le service i sont pris en compte.

Dans le premier programme $PLNE1$, pour un jour donné d, il s'agit de maximiser pour chaque soignant, le plus petit écart séparant son temps de travail maximum légal ajusté et le temps minimal nécessaire pour réaliser tous les services qui lui seront assignés. Ainsi une variable A, représentant cet écart est introduite. La fonction économique (4.5) consiste à maximiser A. La deuxième inéquation (4.7) contraint le temps minimal nécessaire pour réaliser les services qui sont assignés à un soignant donné. Il s'agit dans ce cas d'éviter d'effectuer d'éventuelles heures supplémentaires. Enfin, l'inéquation (4.8) force le respect des contraintes de qualification.

$PLNE1(d)$:

$$\max \quad A \tag{4.5}$$

$$\text{s.t.} \quad \omega'_{kd} - \sum_{i \in \mathscr{S}_d} \pi_i'^k \cdot x_{ki} \geq A \qquad \forall k \in \mathscr{K} \tag{4.6}$$

$$\sum_{i \in \mathscr{S}_d} \pi_i'^k \cdot x_{ki} \leq \omega'_{kd} \qquad \forall k \in \mathscr{K} \tag{4.7}$$

$$\sum_{k \in \mathscr{K}} q_{ik} \cdot x_{ki} = 1 \qquad \forall i \in \mathscr{S}_d \tag{4.8}$$

$$x_{ki} \in \{0,1\} \qquad \forall k \in \mathscr{K}, \forall i \in \mathscr{S}_d \tag{4.9}$$

$PLNE2(d)$:

$$\min \quad W \tag{4.10}$$

$$\text{s.t.} \quad \sum_{i \in \mathscr{S}_d} \pi_i'^k \cdot x_{ki} \leq \omega'_{kd} + W \qquad \forall k \in \mathscr{K} \tag{4.11}$$

$$\sum_{k \in \mathscr{K}} q_{ik} \cdot x_{ki} = 1 \qquad \forall i \in \mathscr{S}_d \tag{4.12}$$

$$x_{ki} \in \{0,1\} \qquad \forall k \in \mathscr{K},\ \forall i \in \mathscr{S}_d \tag{4.13}$$

Le deuxième programme est résolu uniquement en cas d'irréalisabilité du premier. Cela traduit simplement des heures supplémentaires de travail ou une éventuelle sous-traitance. L'inéquation (4.7) peut conduire à cela si la charge de travail est trop importante au regard du temps de travail maximum légal. Pour éviter cet écueil, le deuxième programme mis en place est moins contraint car il s'agit de minimiser le dépassement du temps de travail maximum légal. Ainsi, les services qu'un soignant sera amené à réaliser seront ceux nécessitant le moins d'heures supplémentaires.

La contrainte suivante est ajoutée à chacun de ces programmes pour éviter d'assigner un même soignant à la réalisation de deux services devant commencer en même temps.

$$x_{ki} + x_{kj} \leq 1, \qquad \forall k \in \mathscr{K},\ \forall i \in \mathscr{S}_d,\ \forall j \in \mathscr{S}_d \mid synchro(i,j) = \top \tag{4.14}$$

4.3.3 PHASE 3 : Séquencement des services et Planification horaire

Dans les phases qui précèdent, pour chaque service que nécessitent les patients, deux décisions ont été prises. D'une part, le jour où un service donné sera réalisé. D'autre part, le soignant qui sera amené à réaliser ce service. Ainsi, pour un soignant et un jour donnés, un ensemble de services a été assigné. La phase courante a deux objectifs : déterminer le séquencement de cet ensemble de services et résoudre le problème de *timing* induit. L'issue de cette phase nous conduit à l'obtention d'une solution initiale ς. Dans les lignes qui suivent, nous séparons ces deux objectifs.

4.3.3.1 Séquencement des services

Pour construire les séquences de services pour chaque soignant et pour chaque jour, plusieurs procédures de tri ont été proposées :

- tri croissant en fonction de la date au plus tôt de la première fenêtre de temps : les patients étant disponibles sur plusieurs créneaux horaires, il n'est pas aisé de discriminer des services les uns par rapport aux autres en tenant compte de toutes ces plages horaires. Nous proposons d'ordonner d'abord les services au plus tôt en tenant compte uniquement de la date au plus tôt de la première fenêtre de temps du patient nécessitant un service. Nous espérons par ce processus éviter d'éventuels retards dans la planification ;
- tri croissant en fonction du milieu de la première fenêtre de temps : en tenant compte uniquement de la date au plus tôt de la première fenêtre de temps, la largeur des

fenêtres n'est pas prise en compte. Cependant, cette largeur a un impact important lorsqu'il s'agit de minimiser d'éventuels retards ;
— tri croissant en fonction de la date de fin de la première fenêtre de temps : la troisième procédure de tri est similaire à la première à la seule différence que nous considérons la date de fin de la fenêtre de temps.
— séquencement basé sur la meilleure insertion possible : l'idée envisagée est de partir de l'une des trois procédures de tri précédemment décrites et d'insérer chaque service dans la séquence courante en minimisant à la fois la durée totale de trajet et la somme des retards. Le poids affecté à ces deux critères est identique.

Le séquencement impose de facto une précédence parmi les services. Comme nous l'avons montré dans la sous-section 3.3.4.6 du chapitre 3, ces précédences peuvent générer des interblocages et par conséquent conduire à une solution irréalisable. Afin d'éviter cet écueil, dès qu'un séquencement est fixé, nous vérifions qu'il n'existe pas d'interblocage dans la solution partielle correspondante. Si un interblocage est détecté, une procédure de réparation est appelée. Dès qu'un interblocage est détecté, deux services de la séquence courante, liés à deux autres par des contraintes de délai, sont inter-changés. La place des autres services dans la séquence reste identique, ils ne sont pas déplacés.

4.3.3.2 Planification horaire

La planification horaire correspond à la résolution d'un problème de *timing*. Les décisions qui ont été prises en amont conduisent à la création d'une solution encodée comme dans la section 3.4 du chapitre 3. En effet, dans cette représentation, une solution correspond à une séquence de services affectée à un soignant pour un jour donné. Ce jour a été déterminé durant la première phase de cette méthode par décomposition et le soignant durant la deuxième phase. Enfin, le séquencement a été construit ci-dessus. Le décodage de la solution encodée consiste simplement en l'appel des algorithmes décrits dans la sous-section 3.4.2 du chapitre 3. Dans le cas d'infaisabilité, les services sont sous-traités.

4.4 Amélioration de la solution : retour sur trace et recherche locale

Dans la section précédente, trois phases ont été décrites générant une solution ς initiale. Cette dernière devient également la solution courante qui sera utilisée dans les procédures qui suivent. Cette section a pour but de définir les différentes procédures d'amélioration de cette solution. Ces procédures permettent à la fois une diversification et une intensification de la recherche. La diversification s'effectue grâce à des retours sur trace remettant en cause des décisions prises au préalable, tandis que l'intensification explorera le voisinage proche de la solution courante à partir d'opérateurs de recherche locale. Quatre procédures de retours sur trace sont proposées remettant en cause totalement ou partiellement l'affectation établie aux phases 1 et 2 de la méthode par décomposition. Enfin, des opérateurs classiques de recherche locale sont introduits afin d'améliorer le séquencement et la planification déterminée en amont.

4.4.1 RETOUR SUR TRACE 1 : réaffectation totale des jours de réalisation des services

La phase 1 a permis de déterminer le jour de réalisation de chaque service. Dans le but d'explorer d'autres espaces de solution, l'idée de ce retour sur trace est de remettre en cause l'ensemble des affectations services/jours afin de recalculer une nouvelle solution. Notons d_i le jour de réalisation du service i dans la solution ς courante. Afin de générer une nouvelle solution dont le jour de réalisation de chaque service est distinct de d_i, l'ensemble $\mathscr{H}_i$ des jours de disponibilité de chaque service est redéfini comme suit :

$$\mathscr{H}_i \leftarrow \mathscr{H}_i \setminus \left\{ d_i \text{ si } \left| \mathscr{H}_i \cap \{d \in \mathscr{H}_k \mid k \in \mathscr{K} \wedge q_{ik} = \top\} \right| > 1 \right\}$$

L'ensemble des jours de disponibilité est privé de la valeur d_i uniquement si cette opération n'entraîne pas un ensemble vide. À partir de cette nouvelle définition de $\mathscr{H}_i$, ce retour sur trace consiste à relancer la phase 1 en tenant compte de ces nouvelles valeurs. Chaque valeur d_i est gardée en mémoire. Ainsi lors de la prochaine itération de ce retour sur trace, les valeurs d_i précédemment obtenues sont à nouveaux interdites. Nous réinitialisons l'ensemble $\mathscr{H}_i$ lorsque toutes les valeurs d_i sont interdites.

4.4.2 RETOUR SUR TRACE 2 : réaffectation partielle des jours de réalisation des services

Ce retour sur trace a pour but de remettre en cause partiellement la répartition de la charge de travail tout au long de l'horizon temporel. À l'étape 1 de la phase 2, l'ensemble des services planifiés un jour d donné a été noté $\mathscr{S}_d$. L'idée ici est de changer partiellement l'ensemble $\mathscr{S}_d$ et donc re-planifier un autre jour certains services appartenant à cet ensemble. Pour chaque jour d, un nombre μ_d de services est déterminé. Ce nombre correspond au nombre maximum de nouveaux services qui seront affectés à ce jour d ou au nombre maximum de services qui seront retirés de ce jour. Dans le premier cas, la valeur μ_d est positive et dans le deuxième cas, elle est négative. Pour assurer une cohérence de ces mouvements de service, le nombre total de services qui sont retirés de certains jours est égal au nombre de service que d'autres reçoivent. Ainsi, pour tout l'horizon $\mathscr{H}$ de planification, la somme des μ_d est nulle : $\sum_{d \in \mathscr{H}} \mu_d = 0$.

La valeur de μ_d peut-être calculée de manière aléatoire. Cependant, ce mode de calcul ne nous permet pas de garantir la stabilité de la méthode. Nous proposons donc un calcul heuristique de la valeur de μ_d en prenant en considérant un certain nombre d'information sur la charge de travail de la journée d (i.e. la sous-traitance, le retard des services, le temps d'avance, les heures supplémentaires et le temps total de trajet). Cette valeur est obtenue comme suit :

$$\mu_d = \alpha_d + \beta_d + \gamma_d + \delta_d + \epsilon_d$$

avec :

$$\alpha_d = \begin{cases} \lceil SUB_d \rceil \text{ si } SUB_d > 0 \\ \lfloor SUB_d \rfloor \text{ si } SUB_d < 0 \\ 0 \text{ sinon} \end{cases}$$

SUB_d correspond à la différence entre le nombre moyen de services planifiés par jour et le nombre de services planifiés le jour d.

$$\beta_d = \begin{cases} -3 \text{ si } TARD_d \text{ est la plus élevée} \\ -2 \text{ si } TARD_d \text{ est la } 2^{\text{ème}} \text{ valeur la plus élevée} \\ -1 \text{ si } TARD_d \text{ est la } 3^{\text{ème}} \text{ valeur la plus élevée} \end{cases}$$

$TARD_d$ correspond à la somme des retards des services le jour d.

$$\gamma_d = \begin{cases} +3 \text{ si } EARL_d \text{ est la plus élevée} \\ +2 \text{ si } EARL_d \text{ est la } 2^{\text{ème}} \text{ valeur la plus élevée} \\ +1 \text{ si } EARL_d \text{ est la } 3^{\text{ème}} \text{ valeur la plus élevée} \end{cases}$$

$EARL_d$ correspond à la somme du temps d'avance des services le jour d.

$$\delta_d = \begin{cases} -3 \text{ si } HS_d \text{ est la plus élevée} \\ -2 \text{ si } HS_d \text{ est la } 2^{\text{ème}} \text{ valeur la plus élevée} \\ -1 \text{ si } HS_d \text{ est la } 3^{\text{ème}} \text{ valeur la plus élevée} \\ +3 \text{ si } HS_d \text{ est la moins élevée} \\ +2 \text{ si } HS_d \text{ est la } 2^{\text{ème}} \text{ valeur la moins élevée} \\ +1 \text{ si } HS_d \text{ est la } 3^{\text{ème}} \text{ valeur la moins élevée} \end{cases}$$

HS_d correspond à la somme du nombre d'heures supplémentaires le jour d.

$$\epsilon_d = \begin{cases} -3 \text{ si } TRAVEL_d \text{ est la plus élevée} \\ -2 \text{ si } TRAVEL_d \text{ est la } 2^{\text{ème}} \text{ valeur la plus élevée} \\ -1 \text{ si } TRAVEL_d \text{ est la } 3^{\text{ème}} \text{ valeur la plus élevée} \\ +3 \text{ si } TRAVEL_d \text{ est la moins élevée} \\ +2 \text{ si } TRAVEL_d \text{ est la } 2^{\text{ème}} \text{ valeur la moins élevée} \\ +1 \text{ si } TRAVEL_d \text{ est la } 3^{\text{ème}} \text{ valeur la moins élevée} \end{cases}$$

$TRAVEL_d$ correspond au temps total de trajet le jour d.

Les valeurs de μ_d étant calculée de manière heuristique, si $\sum_{d \in \mathscr{H}} \mu_d \neq 0$, nous incrémentons aléatoirement certaines valeurs de μ_d afin d'obtenir $\sum_{d \in \mathscr{H}} \mu_d = 0$.

La nouvelle répartition des services tout au long de l'horizon est réalisée à l'aide d'un graphe de re-planification qui dépend de la solution courante ς.

Définition 17. *(Graphe de re-planification).*

L'ensemble des sommets du graphe de planification est constitué de :

- *un sommet source s ;*
- *un sommet puits t ;*
- *un sommet pour chaque jour d dont $\mu_d < 0$;*
- *un sommet pour chaque service $i \in \mathscr{S}$*
- *un sommet pour chaque jour d dont $\mu_d > 0$;*

L'ensemble des arcs du graphe est constitué :

- *d'arcs reliant le sommet source aux jours dont la valeur de μ_d est négative. La capacité de chaque arc vaut : $-\mu_d$. Quant au coût, il sera nul pour tous les arcs ;*
- *d'arcs reliant chaque jour d dont la valeur de μ_d est positive au sommet puits. À chacun de ces arcs est associé une capacité μ_d calculée au préalable. Quant au coût,*

il sera nul pour tous les arcs ;

— *d'arcs reliant chaque jour d dont la valeur de μ_d est négative aux services. À chacun de ces arcs est associé une capacité de 1. Le coût associé à ces arcs dont le service en jeu est planifié ce jour dans la solution ς est $M - \Delta$. La valeur M étant une valeur arbitraire élevée et la valeur de Δ est calculée comme suit :*

$$\Delta = \left\{ \begin{array}{l} \left\{ \begin{array}{l} 1 \textit{ si le service correspondant est sous-traité dans la solution } \varsigma \\ 0 \textit{ sinon} \end{array} \right. \\ + \left\{ \begin{array}{l} 1 \textit{ si le service correspondant est en retard} \\ 0 \textit{ sinon} \end{array} \right. \\ + \left\{ \begin{array}{l} 1 \textit{ si le temps minimum nécessaire pour réaliser le service} \\ \quad \textit{correspondant est le plus élevé} \\ 0 \textit{ sinon} \end{array} \right. \end{array} \right.$$

Le coût de tous autres arcs vaut M ;

— *d'arcs reliant chaque service aux jours dont la valeur μ_d est positive si le patient nécessitant ce service est disponible le jour en question. À chacun de ces arcs est associé une capacité 1. Quand au coût, il sera nul pour tous ces arcs ;*

Le flot maximum à coût minimum est calculé pour le graphe de re-planification qui vient d'être défini afin de déterminer la nouvelle répartition de la nouvelle charge de travail tout au long de l'horizon. Si le flot passant sur l'arc (i, d) est égal à 1 (i représentant un service quelconque et d, un jour pour lequel μ_d est positif) alors le jour d sera le nouveau jour de réalisation du service i dans la nouvelle solution.

4.4.3 RETOUR SUR TRACE 3 : réaffectation totale des services entre soignants

Dans la phase 2, la répartition des services entre les soignants a été déterminé. Cette répartition satisfait la contrainte de continuité des soins du patient et assure un partage équitable de la charge de travail. L'objet de ce retour sur trace consiste à explorer d'autres espaces de solutions en remettant en cause totalement l'affectation des soignants. L'idée est donc de revoir cette répartition en essayant d'éviter au maximum les mêmes affectations soignants/services de la solution précédente et en respectant toujours l'ensemble des contraintes liées aux affectations. Notons k_i le soignant qui réalise le service i dans la solution ς. Ce retour sur trace vise à calculer cette répartition en interdisant la réalisation du service i au soignant k_i s'il existe un autre soignant k qui :

— satisfait la continuité des soins du patient p nécessitant le service i
— est qualifié à réaliser le service i, donc $q_{ik} = \top$
— est disponible le jour d_i, c'est-à-dire $d_i \in \mathscr{H}_k$

L'ensemble des soignants satisfaisant la continuité des soins du patient p nécessitant le service i a été noté Φ_p à l'étape 1 de la phase 2 et la variable de décision x_{ki} représente le choix du soignant k pour la réalisation du service i. Afin d'interdire le choix de k_i de nouveau, nous ajoutons aux deux programmes linéaires en nombres entiers $PLNE1(d)$ et

$PLNE2(d)$ de l'étape 2 de la phase 2, la contrainte suivante :

$$\begin{array}{rcl} si \quad \exists k \in \Phi_p \quad \wedge \quad q_{ik} = \top \quad \wedge \quad k_i \neq k \quad \wedge \quad d_i \in \mathscr{H}_k : \\ x_{k_i i} \quad = \quad 0 \qquad\qquad\qquad\qquad\qquad\qquad \forall i \in \mathscr{S}_d \end{array}$$

Cette contrainte est ajoutée à ces programmes et gardée en mémoire afin d'éviter de réaffecter le même soignant à la réalisation d'un service. Lorsque toutes possibilités d'affectation ont été envisagées, une réinitialisation est effectuée.

4.4.4 RECHERCHE LOCALE

À l'issue de la phase 3 établissant un séquencement des services et une planification horaire, une solution ς réalisable est produite. Afin d'explorer l'espace de solutions proche de celle-ci, nous proposons une recherche locale ayant pour rôle d'intensifier la recherche. Cette recherche locale est composée d'un ensemble de mouvements altérant la solution ς. Cela permet de changer le séquencement des services, de réaffecter des services à un autre soignant et aussi de réaffecter des services à un autre jour. Après chaque mouvement, l'algorithme de décodage associé à l'encodage concerné est appliqué pour obtenir une nouvelle solution. La stratégie d'amélioration implémentée consiste à appliquer un mouvement dès la première amélioration de la solution courante.

4.4.4.1 Encodage direct

Appliquer la recherche locale sur l'encodage indirect peut s'avérer coûteux, car, à chaque mouvement, l'algorithme de décodage est appelé. Cet algorithme étant de complexité exponentielle dans certains cas, le temps de calcul induit serait trop important. Nous avons choisi de réaliser la recherche locale sur l'encodage direct. Pour rappel, une solution dans l'encodage direct est représentée par un ensemble de tournées. Chaque tournée (k, d) réalisée par un soignant k le jour d est constituée d'une séquence de services. Cette dernière étant l'ordre dans lequel le soignant k réalisera ces services le jour d. Cet encodage dispose également d'un soignant fictif k_f auquel sont affectés les services qui sont sous-traités.

La recherche locale sur l'encodage direct consiste dans le déplacement d'un ou plusieurs services d'une tournée vers une autre tournée. Ainsi, ces déplacements peuvent changer, d'une part, le séquencement des services, et/ou, d'autre part, le soignant et le jour auxquels les services déplacés ont été préalablement affectés. La recherche locale que nous proposons est basée sur un ensemble d'opérateurs issus de la littérature du VRP. Les opérateurs implémentés sont les suivants :

- **Relocate**
- **2-Opt**
- **2-Opt***
- **Or-Opt**
- **Exchange**
- **CROSS-Exchange**

Un septième opérateur a été joint à ces 6 opérateurs pour prendre en compte la sous-traitance. Cet opérateur que nous dénommerons **Relocate*** est une variante de l'opérateur **Relocate**. Ce dernier déplace un service d'une tournée d'origine vers une tournée de

destination. Cependant, dans le cas de **Relocate***, ce sont uniquement les services affectés au soignant fictif k_f, c'est-à-dire ceux qui sont sous-traités qui sont déplacés dans les autres tournées.

Un certain nombre d'améliorations a été mis en place pour améliorer la recherche locale. D'abord, tout mouvement d'un service impliqué dans une contrainte de synchronisation ou de délai peut entraîner l'introduction d'un interblocage. Comme nous l'avons déjà montré, cela conduit nécessairement à une solution irréalisable. Ainsi, à chaque mouvement d'un service impliqué dans une contrainte de synchronisation ou de délai, une détection d'interblocage est effectuée et permet de savoir si la solution représentée par l'encodage direct est potentiellement réalisable. Ensuite, avant chaque mouvement effectué, nous vérifions que les contraintes de qualification et de disponibilité journalière sont satisfaites. Cette vérification s'effectue en temps constant.

4.5 Expérimentations numériques

Les tableaux 4.1, 4.2, 4.3, 4.4 présentent les résultats des expérimentations numériques réalisées sur les instances de [Bredström et Rönnqvist, 2008] et les tableaux 2.4, 2.5, ceux sur les instances de [Mankowska *et al.*, 2014]. Ces résultats sont comparées aux résultats de la littérature. Pour chaque tableau, l'instance, les résultats de la programmation mathématique (CPLEX), les résultats d'une méthode de résolution de la littérature à laquelle nous nous comparons et les résultats de la méthode par décomposition (DECOMP.) décrite ci-dessus. Les valeurs mises en gras correspondent aux meilleures solutions ; dans le cas de CPLEX, ces solutions sont optimales. Le gap est toujours calculé par rapport à la meilleure solution obtenue entre la programmation mathématique et la méthode de résolution à laquelle nous nous comparons. La version de la méthode par décomposition retenue utilise une recherche locale appliquée l'encodage direct. L'utilisation de l'encodage indirect nécessite un temps de calcul plus important pour des résultats de moindre qualité.

Les tableaux 4.1, 4.2 rendent compte des résultats obtenus et de leurs comparaisons à ceux de l'heuristique (Bredstrom H) proposée par [Bredström et Rönnqvist, 2008] en optimisant respectivement le temps total de trajet et les préférences. Dans le premier tableau où il s'agit de minimiser le temps total de trajet, nous pouvons constater que la méthode par décomposition retrouve ou améliore les meilleures solutions de 25 sur 30 instances obtenues à partir de CPLEX ou de l'heuristique de [Bredström et Rönnqvist, 2008]. Ces performance sont en plus réalisées avec des temps de calcul plus faible. Les 5 instances pour lesquelles la méthode par décomposition n'est pas meilleure, le gap induit est d'au plus 3.43 %. Dans le deuxième tableau où il s'agit d'optimiser les préférences, les résultats sont mitigés. En effet, la méthode par décomposition retrouve ou améliore les meilleures solutions de 14 sur 30 instances obtenues à partir de CPLEX ou de l'heuristique de [Bredström et Rönnqvist, 2008]. Les performances de la méthode par décomposition sur les instances moyennes (06 à 08) et grandes (09 à 10) sont correctes voire bonnes. Paradoxalement, sur les petites instances, les résultats sont de qualité moindre.

Dans les tableaux 4.1 et 4.2, nous comparons la méthode par décomposition à l'algorithme de recuit simulé hybridé à une recherche locale itérée (Afifi SA-ILS) proposée [Afifi *et al.*, 2016]. Ce dernier est dédié à la résolution de ces instances et à notre connaissance,

constitue l'état de l'art. Dans le tableau4.1 où l'objectif consiste à optimiser le temps total de trajet, nous nous apercevons que la méthode par décomposition est dominée tant par la qualité des solutions que par le temps de calcul nécessaire pour les obtenir. Il semblerait que la généricité nuise aux performances. Cependant, le gap reste relativement faible puisque nous obtenons dans le pire des cas un gap de 11.12% et un gap moyen de 2.69%. Ces résultats peuvent être également constatés dans le tableau 4.4 où il s'agit d'optimiser les préférences. En effet, en dehors de l'instance 09S, la méthode par décomposition est dominée par SA-ILS avec un gap moyen de 7.51%.

Les tableaux 4.5, 4.6 rassemblent les résultats expérimentaux réalisées sur les instances de [Mankowska *et al.*, 2014] à partir de la méthode par décomposition. Il est difficile de dégager des conclusions pour l'ensemble des instances. Cependant, il est plus aisé de faire des comparaisons en tenant compte des 7 groupes (A,B,C,D,E,F,G) d'instances. Pour le premier groupe (A01-A10) d'instances, nous pouvons relever que la méthode par décomposition retrouve toutes les solutions optimales déterminées par CPLEX sauf pour l'instance A02. La méthode par décomposition obtient des résultats équivalents à ceux de la méthode AVNS proposée par [Mankowska *et al.*, 2014]. Concernant le deuxième groupe (B01-B10) d'instances, la programmation mathématique domine la méthode par décomposition pour la qualité des solutions mais nécessite des temps de calcul plus élevés. La méthode par décomposition ne réussit pas à retrouver les meilleures solutions. Un gap moyen 9.28% est observé. Cependant, nous pouvons constater que la méthode par décomposition domine la méthode AVNS pour 7 instances sur 10 avec des temps de calcul plus importants. Pour le troisième groupe (C01-C10) d'instances, les résultats s'inversent. En effet, la méthode par décomposition est dominée malgré des temps de calcul plus important sur 8 des 10 instances par la méthode AVNS. La méthode par décomposition réussit tout de même à améliorer 2 des meilleures solutions connues. Les instances des groupes (D,E,F,G) sont de grande taille puisque la programmation mathématique n'arrive à prouver la réalisabilité de ces instances après 36000 secondes. Le groupe composé des instances D01 à D10, la méthode par décomposition améliore les meilleures solution connues de 5 d'entre elles un gap minimum de 7.74%. Cependant, elle peine à trouver de bonnes solutions pour les autres instances de ce groupe. Nous pouvons observer par exemple un gap de 138.29 % pour l'instance D04. Les résultats expérimentaux des trois derniers groupes (E,F,G) sont homogènes puisque la méthode par décomposition améliore systématiquement les meilleures solutions connue sauf pour les instances F10 et G01. Cependant, ces améliorations sont réalisées avec des temps de calcul plus importants.

Instance	CPLEX		Bredstrom H		DECOMP.		
	Z	CPU(s)	Z	CPU(s)	Z	GAP(%)	CPU(s)
01S	**3.55**	0.59	**3.55**	120.03	**3.55**	0.0	8.77
01M	**3.55**	1.60	**3.55**	120.25	**3.55**	0.0	11.38
01L	**3.39**	7.10	**3.39**	120.48	**3.39**	0.0	7.35
02S	**4.27**	0.19	**4.27**	120.07	**4.27**	0.0	13.11
02M	**3.58**	1.22	**3.58**	120.11	3.6	0.55	5.44
02L	**3.42**	21.68	**3.42**	120.95	**3.42**	0.0	5.61
03S	**3.63**	0.61	**3.63**	120.26	**3.63**	0.0	9.19
03M	**3.33**	1.64	**3.33**	120.19	**3.53**	0.0	9.69
03L	**3.29**	9.27	**3.29**	120.6	**3.29**	0.0	10.29
04S	**6.14**	1.16	**6.14**	120.16	6.19	0.81	24.22
04M	**5.67**	22.56	**5.67**	120.15	**5.67**	0.0	17.01
04L	**5.13**	2895.71	5.3	120.04	**5.13**	0.0	8.47
05S	**3.93**	0.45	**3.93**	120.13	3.94	0.25	9.07
05M	**3.53**	0.49	**3.53**	120.09	**3.53**	0.0	10.27
05L	**3.34**	1.74	**3.34**	120.85	**3.34**	0.0	5.91
06S	**8.14**	3670.13	**8.14**	600.94	8.42	3.43	74.95
06M	8.10	3707.58	11.63	609.58	**8.07**	-0.37	69.54
06L	-	3656.58	-	624.06	**7.43**	-	72.29
07S	**8.39**	3647.23	8.97	603.97	8.64	2.97	156.91
07M	-	3603.96	-	648.02	**7.82**	-	79.76
07L	-	3605.11	-	645.33	**7.21**	-	85.48
08S	11.34	3602.30	-	657.03	**10.32**	-8.99	225.61
08M	-	3601.67	-	632.61	**9.49**	-	394.99
08L	-	3604.41	-	618.63	**8.28**	-	108.77
09S	-	3600.30	-	626.26	**12.71**	-	1373.66
09M	-	3616.31	-	612.19	**11.39**	-	477.59
09L	-	3616.12	-	607.36	**11.01**	-	475.09
10S	-	3613.17	-	604.46	**9**	-	341.11
10M	-	3618.48	-	705.2	**8.38**	-	239.42
10L	-	3657.72	-	631.39	**7.86**	-	279.27

TABLE 4.1 – Résultats pour les instances de [Bredström et Rönnqvist, 2008] en optimisant le temps total de trajet et comparaison avec l'heuristique proposée par [Bredström et Rönnqvist, 2008]

Instance	CPLEX		Bredstrom H		DECOMP.		
	Z	CPU(s)	Z	CPU(s)	Z	GAP(%)	CPU(s)
01S	**-114.03**	0.27	**-114.03**	3600	-111.52	2.20	23.57
01M	**-117.80**	0.33	**-117.80**	3600	**-117.80**	0.0	13.84
01L	**-118.51**	0.73	**-118.51**	3600	-108.26	8.64	15.62
02S	**-92.09**	0.20	**-92.09**	3600	**-92.09**	0.0	29.30
02M	**-104.81**	4.93	**-104.81**	3600	-100.05	4.54	12.13
02L	**-107.64**	141.25	**-107.64**	3600	-102.51	4.76	16.93
03S	**-99.49**	0.36	**-99.49**	3600	-92.23	7.29	21.38
03M	**-106.59**	0.69	**-106.59**	3600	-97.41	8.61	22.86
03L	**-107.87**	1.44	**-107.87**	3600	-96.09	10.92	30.42
04S	**-100.00**	0.35	**-100.00**	3600	**-100.00**	0.0	40.95
04M	**-106.72**	7.28	**-106.72**	3600	-47.02	55.94	18.43
04L	**-109.27**	178.08	**-109.27**	3600	-95.35	12.55	20.97
05S	**-76.29**	0.11	**-76.29**	3600	-72.74	4.65	25.00
05M	**-76.29**	0.42	**-76.29**	3600	-73.52	3.63	15.25
05L	**-84.21**	3.20	**-84.21**	3600	-76.45	9.21	18.62
06S	**-370.06**	362.64	**-370.06**	3600	-364.07	1.61	286.30
06M	-379.88	3601.08	-374.257	3600	-365.246	2.65	266.04
06L	-	3600.75	-368.876	3600	**-369.79**	-0.24	282.51
07S	**-401.11**	813.57	-296.725	3600	-379.34	5.42	331.48
07M	-	3600.91	-368.565	3600	**-378.61**	-2.72	560.90
07L	-	3600.82	-355.716	3600	**-404.16**	-13.6	173.74
08S	-351.99	3600.99	-	3600	-	-	437.35
08M	-	3600.88	-	3600	**-337.72**	-	348.31
08L	-	3600.88	-	3600	**-396.58**	-	396.68
09S	-	3600.32	-	3600	**-593.44**	-	1317.93
09M	-	3600.82	-	3600	**-594.62**	-	1423.17
09L	-	3601.77	-	3600	**-518.21**	-	1846.5
10S	-	3601.49	-	3600	**-667.05**	-	1011.69
10M	-	3602.20	-	3600	**-658.55**	-	854.35
10L	-	3600.11	-445.027	3600	**-635.90**	-42.89	813.62

TABLE 4.2 – Résultats pour les instances de [Bredström et Rönnqvist, 2008] en optimisant les préférences et comparaison avec l'heuristique proposée par [Bredström et Rönnqvist, 2008]

Instance	CPLEX		Afifi SA-ILS		DECOMP.		
	Z	CPU(s)	Z	CPU(s)	Z	GAP(%)	CPU(s)
01S	**3.55**	0.59	**3.55**	0.02	**3.55**	0.0	8.77
01M	**3.55**	1.60	**3.55**	0.02	**3.55**	0.0	11.38
01L	**3.39**	7.10	**3.39**	0.03	**3.39**	0.0	7.35
02S	**4.27**	0.19	**4.27**	0.02	**4.27**	0.0	13.11
02M	**3.58**	1.22	**3.58**	0.03	3.6	0.55	5.44
02L	**3.42**	21.68	**3.42**	0.03	**3.42**	0.0	5.61
03S	**3.63**	0.61	**3.63**	0.02	**3.63**	0.0	9.19
03M	**3.33**	1.64	**3.33**	0.03	**3.53**	0.0	9.69
03L	**3.29**	9.27	**3.29**	0.02	**3.29**	0.0	10.29
04S	**6.14**	1.16	**6.14**	0.02	6.19	0.81	24.22
04M	**5.67**	22.56	**5.67**	0.05	**5.67**	0.0	17.01
04L	**5.13**	2895.71	**5.13**	0.09	**5.13**	0.0	8.47
05S	**3.93**	0.45	**3.93**	0.03	3.94	0.25	9.07
05M	**3.53**	0.49	**3.53**	0.03	**3.53**	0.0	10.27
05L	**3.34**	1.74	**3.34**	0.03	**3.34**	0.0	5.91
06S	**8.14**	3670.13	**8.14**	13.97	8.42	3.43	74.95
06M	8.10	3707.58	**7.7**	26.68	8.07	4.80	69.54
06L	-	3656.58	**7.14**	15.86	7.43	4.06	72.29
07S	**8.39**	3647.23	**8.39**	15.08	8.64	2.97	156.91
07M	-	3603.96	**7.48**	18.34	7.82	4.54	79.76
07L	-	3605.11	**6.88**	15.92	7.21	4.79	85.48
08S	11.34	3602.30	**9.54**	25.13	10.32	8.17	225.61
08M	-	3601.67	**8.54**	15.01	9.49	11.12	394.99
08L	-	3604.41	**8**	24.51	8.28	3.49	108.77
09S	-	3600.30	**11.93**	150.52	12.71	6.53	1373.66
09M	-	3616.31	**10.92**	292.17	11.39	4.30	477.59
09L	-	3616.12	**10.49**	207.17	11.01	4.95	475.09
10S	-	3613.17	**8.60**	16.10	9	4.65	341.11
10M	-	3618.48	**7.62**	52.75	8.38	9.97	239.42
10L	-	3657.72	**7.75**	51.89	7.86	1.41	279.27

TABLE 4.3 – Résultats pour les instances de [Bredström et Rönnqvist, 2008] en optimisant le temps total de trajet et comparaison avec l'heuristique proposée par [Afifi *et al.*, 2016]

Instance	CPLEX		Afifi SA-ILS		DECOMP.		
	Z	CPU(s)	Z	CPU(s)	Z	GAP(%)	CPU(s)
01S	**-114.03**	0.27	**-114.03**	0.03	-111.52	2.20	23.57
01M	**-117.80**	0.33	**-117.80**	0.02	**-117.80**	0.0	13.84
01L	**-118.51**	0.73	**-118.51**	0.04	-108.26	8.64	15.62
02S	**-92.09**	0.20	**-92.09**	0.05	**-92.09**	0.0	29.30
02M	**-104.81**	4.93	**-104.81**	0.04	-100.05	4.54	12.13
02L	**-107.64**	141.25	**-107.64**	0.38	-102.51	4.76	16.93
03S	**-99.49**	0.36	**-99.49**	0.02	-92.23	7.29	21.38
03M	**-106.59**	0.69	**-106.59**	0.07	-97.41	8.61	22.86
03L	**-107.87**	1.44	**-107.87**	0.14	-96.09	10.92	30.42
04S	**-100.00**	0.35	**-100.00**	0.03	**-100.00**	0.0	40.95
04M	**-106.72**	7.28	**-106.72**	0.07	-47.02	55.94	18.43
04L	**-109.27**	178.08	**-109.27**	0.13	-95.35	12.55	20.97
05S	**-76.29**	0.11	**-76.29**	0.02	-72.74	4.65	25.00
05M	**-76.29**	0.42	**-76.29**	0.03	-73.52	3.63	15.25
05L	**-84.21**	3.20	**-84.21**	0.04	-76.45	9.21	18.62
06S	**-370.06**	362.64	**-370.06**	0.7	-364.07	1.61	286.30
06M	-379.88	3601.08	**-379.88**	25.26	-365.246	3.85	266.04
06L	-	3600.75	**-387.2**	16.33	-369.79	4.49	282.51
07S	**-401.11**	813.57	**-401.11**	0.58	-379.34	5.42	331.48
07M	-	3600.91	**-406.17**	6.48	-378.61	6.78	560.90
07L	-	3600.82	**-407.48**	2.53	-404.16	0.81	173.74
08S	-351.99	3600.99	**-380.76**	26.12	-	-	437.35
08M	-	3600.88	**-403.57**	59.34	-337.72	16.31	348.31
08L	-	3600.88	**-407.48**	20.51	-396.58	2.67	396.68
09S	-	3600.32	-581.12	117.4	**-593.44**	-2.12	1317.93
09M	-	3600.82	**-656.5**	10.90	-594.62	9.42	1423.17
09L	-	3601.77	**-666.5**	17.81	-518.21	22.24	1846.5
10S	-	3601.49	**-675.81**	162.42	-667.05	1.29	1011.69
10M	-	3602.20	**-686.75**	150.63	-658.55	4.10	854.35
10L	-	3600.11	**-691.48**	270.26	-635.90	8.03	813.62

TABLE 4.4 – Résultats pour les instances de [Bredström et Rönnqvist, 2008] en optimisant les préférences et comparaison avec l'heuristique proposée par [Afifi *et al.*, 2016]

Instance	CPLEX		Mankowska AVNS		DECOMP.		
	Z	CPU(s)	Z	CPU(s)	Z	GAP(%)	CPU(s)
A01	**218.20**	0.23	**218.2**	1	**218.199**	0.0	0.55
A02	**246.62**	0.25	248.1	1	246.626	2.97	0.0
A03	**305.86**	0.32	**305.9**	1	**305.86**	0.0	0.66
A04	**186.90**	0.92	**186.9**	1	**186.897**	0.0	0.81
A05	**189.54**	0.35	192.0	1	**189.543**	0.0	0.54
A06	**200.10**	0.26	**200.1**	1	**200.099**	0.0	0.68
A07	**225.37**	0.23	**225.4**	1	**225.369**	0.0	0.71
A08	**232.05**	0.18	**232.0**	1	**232.049**	0.0	0.71
A09	**222.29**	1.70	**222.3**	1	**222.295**	0.0	0.53
A10	**225.01**	0.05	**225.0**	1	**225.006**	0.0	0.18
B01	**428.10**	12711.60	458.9	1	450.967	5.34	7.23
B02	**476.05**	5944.16	580.9	1	502.919	5.64	3.91
B03	**399.09**	22530.20	431.4	1	480.916	20.50	5.89
B04	411.30	36001.90	587.3	1	445.446	8.30	3.85
B05	**366.34**	13114.60	391.1	1	392.759	7.21	3.46
B06	440.93	36000.70	545.9	1	495.632	12.40	11.41
B07	**328.67**	442.95	356.6	1	395.107	20.21	6.18
B08	**357.68**	575.17	410.9	1	364.418	1.88	7.90
B09	402.67	36001.10	487.9	1	440.658	9.43	5.14
B10	469.58	36001.40	500.4	1	478.911	1.98	6.93
C01	-	36000.90	**1123.6**	1	1306.71	16.29	64.64
C02	-	36000.70	**677.0**	1	891.528	31.68	38.76
C03	547.54	36029.40	642.4	1	658.51	20.26	37.06
C04	-	36013.40	**580.4**	1	844.925	45.57	25.49
C05	691.56	36004.00	754.6	1	931.038	34.62	28.10
C06	-	36001.20	**951.6**	1	1063.78	11.78	42.78
C07	-	36002.00	577.4	1	**530.78**	-8.07	33.42
C08	469.49	36005.50	540.6	1	556.21	2.88	22.65
C09	-	36001.10	**608.7**	1	612.204	0.54	41.99
C10	-	36001.50	679.3	1	**643.95**	-5.20	23.85

TABLE 4.5 – Résultats pour les instances de [Mankowska *et al.*, 2014] en optimisation une somme pondérée des critères suivants : la distance totale parcourue, le retard maximal et la somme des retards. Le poids de chaque critère est $\frac{1}{3}$. Comparaison avec la méthode de résolution AVNS proposée par [Mankowska *et al.*, 2014]

Instance	CPLEX		Mankowska AVNS		DECOMP.		
	Z	CPU(s)	Z	CPU(s)	Z	GAP(%)	CPU(s)
D01	36004.30	-	**1321.8**	5	1486.61	12.46	94.34
D02	36003.10	-	892.7	4	**823.529**	-7.74	69.34
D03	36002.30	-	819.4	6	**719.08**	-12.13	67.25
D04	36010.50	-	**877.4**	4	2090.84	138.29	211.55
D05	36013.90	-	872.1	7	**801.549**	-8.08	84.44
D06	36010.00	-	**835.2**	8	951.808	13.96	86.30
D07	36009.80	-	706.3	7	**645.322**	-8.63	63.74
D08	36007.10	-	811.4	7	**732.973**	-9.66	37.88
D09	36006.40	-	**860.3**	7	881.672	2.48	86.77
D10	36002.70	-	**1306.6**	4	1448.82	10.88	82.09
E01	36025.30	-	1604.9	31	**1388.44**	-13.48	199.59
E02	36180.20	-	1159.7	27	**868.307**	-25.12	146.36
E03	36104.70	-	986.4	28	**935.212**	-5.18	121.28
E04	36360.30	-	871.0	37	**731.129**	-16.05	166.72
E05	36104.20	-	1018.0	29	**879.461**	-13.60	144.75
E06	36159.30	-	1003.0	36	**840.93**	-16.15	134.28
E07	36130.20	-	921.1	38	**783.794**	-14.90	145.69
E08	36162.20	-	884.6	36	**821.426**	-7.14	103.80
E09	36190.80	-	1131.7	34	**1004.47**	-11.24	196.64
E10	36008.50	-	1066.6	31	**901.183**	-15.50	216.41
F01	36241.80	-	1721.4	889	**1585.17**	-7.91	1810.95
F02	36000.00	-	1763.8	909	**1488.94**	-15.58	1577.21
F03	36000.00	-	1549.6	868	**1330.83**	-14.11	1885.41
F04	36000.00	-	1420.4	1321	**1349.05**	-5.02	1553.97
F05	36000.00	-	1701.9	1145	**1507.19**	-11.44	1815.47
F06	36000.00	-	1639.7	836	**1553.21**	-5.27	2210.09
F07	36000.00	-	1384.3	1294	**1333.92**	-3.63	1716.53
F08	36000.00	-	1544.6	924	**1291.56**	-16.38	2564.28
F09	36000.00	-	1572.9	1642	**1531.63**	-2.62	2207.57
F10	36000.00	-	**1581.0**	1326	2141.3	35.43	3579.55
G01	36000.00	-	**2248.0**	7200	2602.22	15.75	18562.5
G02	36000.00	-	2316.1	7200	**2122.86**	-8.34	13197.2
G03	36000.00	-	1885.3	7147	**1727.32**	-8.37	10526.6
G04	36000.00	-	2023.2	7200	**1845.7**	-8.77	13109.1
G05	36000.00	-	2247.6	7200	**2067.27**	-8.02	17093.6
G06	36000.00	-	2144.4	7200	**2127.78**	-0.77	14310.5
G07	36000.00	-	1971.5	6934	**1737.69**	-11.85	12461.5
G08	36000.00	-	1987.4	7200	**1738.89**	-12.50	14526.1
G09	36000.00	-	2415.5	7023	**2263.24**	-6.30	18729.5
G10	36000.00	-	2373.4	7003	**2049.63**	-13.64	17318.9

TABLE 4.6 – Résultats pour les instances de [Mankowska *et al.*, 2014] en optimisation une somme pondérée des critères suivants : la distance totale parcourue, le retard maximal et la somme des retards. Le poids de chaque critère est $\frac{1}{3}$. Comparaison avec la méthode de résolution AVNS proposée par [Mankowska *et al.*, 2014] (suite)

4.6 Conclusion

Dans ce chapitre, une méthode par décomposition a été proposée. Cette méthode est composée de trois phases distinctes. La première phase détermine les jours de planification des services. La deuxième établit pour chaque soignant l'ensemble des services à réaliser. La troisième phase consiste à séquencer des services pour chaque tournée à construire d'une planification horaire. La solution générée à l'issu de ces phases est améliorée grâce à des procédures de retour sur trace et une recherche locale. Les procédures de retour sur trace, par la diversification, permettent d'explorer d'autres espaces de solution. La recherche locale, par l'intensification, sonde le voisinage de la solution courante. Ce processus d'amélioration est itéré tant que le temps imparti n'est pas écoulé ou qu'au bout d'un certain nombre d'itérations, aucune solution améliorante n'a pas trouvée. Cette méthode par décomposition est la première méthode de résolution que nous avons envisagée.

Chapitre 5

Un algorithme hybride génétique avec gestion de la population

Après avoir proposé un modèle générique pour le HCRSP, des encodages et des algorithmes de décodage d'une solution, et une méthode de résolution par décomposition du problème, nous proposons dans ce chapitre un algorithme hybride génétique avec gestion de la population.

Ces dernières années, l'algorithme mémétique ou l'algorithme hybride génétique [Prins, 2004, Nagata et Bräysy, 2009] a prouvé son efficacité quant à la résolution du VRP. Plus récemment, l'algorithme hybride génétique avec une gestion avancée de la population [Vidal *et al.*, 2014] a obtenu les meilleurs résultats connus dans la résolution du VRP multi-attributs. Cet algorithme combine l'algorithme génétique avec une représentation astucieuse d'une solution, un algorithme d'évaluation efficace, un processus d'intensification à l'aide d'une recherche locale et une gestion avancée de la population. Tous ces succès nous ont amené à proposer aussi un algorithme hybride génétique avec gestion de la population pour le HCRSP.

L'algorithme proposé présente des similitudes avec celui proposé par [Sörensen et Sevaux, 2006] où la gestion de la population s'effectue en utilisant une mesure de distance des solutions. Afin de se prémunir d'une convergence précoce, la population est en partie réinitialisée lorsqu'un seuil de ressemblance des solutions est atteint.

Ce chapitre est organisé comme suit. Nous soulignons nos contributions dans la section 5.1. La section 5.2 décrit l'architecture générale de l'algorithme tout en rappelant les travaux sur lesquels nos propositions sont basées. Les sections 5.3 à 5.9 sont consacrées à la description de l'algorithme ainsi qu'aux justifications des choix qui ont été effectués. La section 5.10 présente les expérimentations numériques menées et la section 5.11 fait office de conclusion en résumant le travail réalisé.

5.1 Contributions

Les contributions de ce chapitre s'inscrivent dans la conception d'une méta-heuristique pour la résolution du problème. Nos contributions sont les suivantes :

— un algorithme hybride génétique avec gestion de la population pour la résolution du HCRSP générique. Cet algorithme ré-utilise les notions du chapitre 3 sur la représentation d'une solution et son évaluation.
— des expérimentations numériques ayant pour objectif d'évaluer l'algorithme en comparant les résultats à ceux de la littérature

5.2 Principes et architecture générale de l'algorithme hybride génétique avec gestion de la population

L'algorithme génétique a été proposé entre autres par [Holland, 1975]. Il appartient à la famille des algorithmes évolutionnaires. Il s'agit de faire évoluer une population d'individus, ici représentant des solutions du HCRSP, afin d'améliorer d'une génération à l'autre, la qualité de la population. Pour ce faire, un certain nombre d'opérations est appliqué à un sous-ensemble d'individus générant de nouveaux individus. Ces nouveaux individus héritent des caractéristiques de leurs parents en espérant que d'une génération à l'autre, les caractéristiques les plus avantageuses soient transmises à la génération suivante. Au cours de ces dernières décennies, l'algorithme génétique a été utilisé pour la résolution de problèmes combinatoires avec plus ou moins de succès. Ses principaux défauts sont un temps de calcul qui peut être élevé, une éventuelle importante consommation de la mémoire et une composante aléatoire non négligeable. Des extensions ont vu le jour afin d'améliorer ses performances. L'algorithme mémétique est une extension de l'algorithme génétique en lui associant une recherche locale afin d'intensifier la recherche. Évidemment, en intégrant une recherche locale à l'algorithme génétique, les temps de calcul sont plus importants mais au bénéfice de meilleurs résultats. À notre connaissance, l'idée de l'algorithme mémétique ou algorithme hybride génétique a été formulée pour la première fois par [Moscato, 1989]. Une des faiblesses de l'algorithme mémétique est la convergence rapide voire précoce de la population. Cette convergence se caractérise par le fait que les individus se ressemblent de plus en plus d'une génération à l'autre et qu'à terme, les nouvelles générations ne diffèrent guère des précédentes. Afin de se prémunir de cette convergence rapide et garantir une diversification de la recherche, il faut garantir une diversité des individus tout au long du déroulement de l'algorithme. Cette diversité peut être réalisée en s'assurant qu'à chaque génération deux individus aient un coût différent. Cette technique a été utilisée avec succès par [Prins, 2004] conduisant à un algorithme hybride génétique pour la résolution du CVRP. Une stratégie plus avancée a été proposée par [Sörensen et Sevaux, 2006]. Il propose de maintenir la diversité au sein de la population grâce à une mesure de distance entre deux individus. Elle évalue la ressemblance ou la dissemblance entre deux solutions à partir de leurs représentations. Ainsi, au-delà de la différence de coût, deux solutions se différencient par leur position dans l'espace de recherche. Cette dernière stratégie a été mise en place avec succès par [Vidal *et al.*, 2014] pour le VRP multi-attributs. La stratégie proposée par [Vidal *et al.*, 2014] diffère légèrement puisqu'elle affecte le coût d'une solution en lui ajoutant un terme représentant la contribution de celle-ci à la diversité de la population. Cet algorithme retrouve ou améliore les meilleures solutions connues de 1045 instances sur 1099 de la littérature, ce qui en fait le meilleur algorithme pour le VRP multi-attributs.

Les succès de l'algorithme hybride génétique nous a donc amené à proposer une méthode

similaire pour la résolution du HCRSP. L'algorithme que nous proposons est basé sur les idées suivantes :

- une solution est représentée par une permutation de services sans délimiteurs (cf. 3.3.1),
- un algorithme de décodage pouvant être exact ou approché selon les contraintes prises en compte (cf. 3.3.2),
- une population restreinte de solutions distinctes
- une recherche locale faisant office de processus d'intensification
- une gestion de la population basée sur une mesure de distance en terme d'espace de recherche

Les différentes étapes de l'algorithme hybride génétique sont synthétisées dans l'algorithme 7. L'algorithme commence par une phase d'initialisation (ligne 1) d'une population P constituée de solutions réalisables et irréalisables du HCRSP. Cette population est construite à partir d'heuristiques et d'une génération aléatoire. La boucle principale (lignes 2-14) de l'algorithme va faire évoluer la population P tant qu'un nombre maximum d'itérations I_{max} sans améliorations de la meilleure solution n'est pas atteint ou que le temps imparti T_{max} n'est pas écoulé. À chaque itération de la boucle principale, deux solutions sont sélectionnées (ligne 3) et croisées (ligne 4). La solution ς obtenue à l'issue de ce processus est "éduquée". L'éducation de la solution ς consiste simplement à explorer son proche voisinage. Ainsi, si la solution est réalisable, l'éducation permet d'améliorer le coût de la solution ς. Par contre, si elle est irréalisable, l'éducation fait office de procédure de réparation. Cette solution est ensuite ajoutée à la population si elle répond à des critères d'ajout (lignes 6-8). Ces derniers tiennent compte du coût de la solution ς et de sa contribution à la diversité des solutions de la population. La solution de la population à remplacer est choisie aléatoirement dans la moitié basse de la population, c'est-à-dire les solutions de coût médiocre. Une diversification de la population est organisée lorsque I_{div} itérations ont été réalisées sans améliorer la meilleure solution ς^* connue. Enfin, dans le but de contrôler la diversité des solutions dans la population, le paramètre de diversification Δ est mis à jour à chaque itération. Les sections qui suivent décrivent plus en détails chaque étape de l'algorithme 7.

Algorithm 7 Algorithme hybride génétique avec gestion de la population pour le HCRSP

```
Initialisation de la population P
while i < I_max ∧ time < T_max do
    Sélection des solutions p_1 et p_2 dans P
    Croisement ⊗ des solutions p_1 et p_2 générant la solution ς
    Éducation de la solution ς
    if ς satisfait les critères d'ajout then
        Remplacement d'une solution de la population P par la solution ς
    end if
    if ς* n'est pas améliorée au bout de I_div itérations then
        Diversification de la population P selon le paramètre Δ
    end if
    Mise à jour du paramètre de diversification Δ
    i ← i + 1
end while
return ς*
```

5.3 Représentation d'une solution et évaluation

La représentation indirecte d'une solution proposée au chapitre 3 est parfaitement adaptée ici. Une solution est donc représentée par un chromosome qui est une séquence sans délimiteurs de tournées. L'avantage de cette représentation est qu'elle nous permet de réutiliser les opérateurs habituellement utilisés dans l'algorithme génétique sans pour autant les adapter aux contraintes du HCRSP.

Concernant l'évaluation d'une solution, l'algorithme 1 du chapitre 3 avec les techniques d'améliorations et les inégalités valides proposées est utilisé en guise d'algorithme d'évaluation pour des cas sans contraintes de qualification et pour de petites instances avec contraintes de qualification. En effet, lorsque les contraintes de qualification sont prises en compte, l'algorithme 1 nécessite un temps de calcul très élevé. Par contre, l'algorithme 3 du chapitre 3 est utilisé pour décoder tous les autres cas. Ce dernier algorithme est un algorithme approché par conséquent, la solution décodée n'est pas forcément la meilleure solution pour le chromosome correspondant. Cette méthode d'évaluation a été mise en place à partir des expérimentations numériques du chapitre 3.

Enfin, il a été montré [Glover et Laguna, 2000, Nagata *et al.*, 2010] que l'oscillation stratégique, c'est-à-dire l'exploitation des solutions irréalisables durant la recherche, permettait d'obtenir de meilleurs résultats. En effet, les solutions situées à la frontière des espaces réalisables et irréalisables sont les plus prometteuses. L'oscillation stratégique est habituellement prise en compte dans le problème de tournées de véhicules en relaxant les fenêtres de temps. Nous avons choisi de mettre en place l'oscillation stratégique par le biais de la sous-traitance. Ainsi, les services qui ne peuvent être planifiés sans violer les contraintes sont assignés au soignant k_f.

5.4 Mesure de distance

De nombreuses mesures de distance existent pour des permutations [Sörensen, 2007]. Cependant, nous devons faire ici un compromis entre le temps de calcul et la qualité de la mesure de distance. En effet, la mesure de distance choisie doit être calculée entre la solution ς et toutes les solutions de la population P. La complexité de la mesure de distance choisie doit être dans le pire cas linéaire ; autrement le temps de calcul sera trop important. Pour toutes ces raisons, nous avons choisi une extension de la distance de distance de Hamming [Ronald, 1998] au détriment de la distance de Levenshtein et de la Broken Pair distance qui nécessitent un temps de calcul plus élevé.

Notons $\sigma(\varsigma)$ l'encodage de la solution ς et $\sigma_i(\varsigma)$ le i-ème élément de cet encodage.

$$D_{\varsigma,\varsigma'} = \sum_{i=1}^{|\mathscr{S}|} f(\varsigma, \varsigma', i)$$

avec

$$f(\varsigma, \varsigma', i) = \begin{cases} 1 \text{ si } \sigma_i(\varsigma) \neq \sigma_i(\varsigma') \\ 0 \text{ sinon} \end{cases}$$

Pour toute la population P de solutions, le calcul de la mesure de distance ci-dessus nécessite une complexité de : $O(|\mathscr{S}| \cdot |P|)$

5.5 Initialisation de la population

La population initiale P est générée à partir d'une heuristique HRandom de meilleure insertion avec une composante aléatoire. Le reste de la population est constitué de solutions obtenues en générant des séquences à partir de l'algorithme 8.

L'heuristique HRandom commence par initialiser un ensemble de tournées possibles vides en tenant compte des disponibilités journalières des soignants. À partir de cet ensemble de tournées, les services sont choisis aléatoirement pour être insérés dans la tournée minimisant le coût total de la solution. Les services ne pouvant être ajoutés sont affectés au soignant fictif k_f. Tout au long du processus, l'ensemble des contraintes du problème est respecté. Trois solutions sont ainsi calculées pour faire partie de la population. Ce nombre de solutions a été choisi arbitrairement.

Une génération totalement aléatoire des séquences peut produire de nombreuses séquences irréalisables à cause des contraintes de qualification et de disponibilité journalière. La plupart des séquences générées ainsi ne satisfait pas la condition nécessaire (mais pas suffisante) de faisabilité décrite à la sous-section 3.3.4.3. Pour rappel, cette condition a été formulée à partir d'un programme de satisfaction de contraintes. Pour sortir de cet écueil, deux stratégies peuvent être envisagées :

— une oscillation stratégique en autorisant la sous-traitance si elle n'est pas autorisée dans le problème d'origine
— une génération astucieuse des séquences de sorte à maximiser le nombre de séquences qui satisfont cette condition nécessaire de faisabilité

L'algorithme 8 a finalement été proposé pour pallier ce problème. Il permet de générer des séquences satisfaisant cette condition. L'idée est de créer des séquences aléatoires à partir d'une pré-affectation partielle. Au début, chaque tournée $(k, d) \in \mathscr{K} \times \mathscr{H}$ est représentée par une séquence vide. Chaque service est ensuite affecté aléatoirement à une tournée si les contraintes de qualification et de disponibilité journalière sont respectées :

$$q_{ik} = \top \wedge d \in \mathscr{H}_i$$

Nous devons cependant éviter d'affecter deux services liés par une contrainte de synchronisation ou de délai d'être affectés à la même tournée. La condition de la ligne 3 permet de traiter cette situation. Enfin, pour chaque tournée, l'ensemble des services qui lui sont affectés est trié selon le milieu de la première fenêtre de temps. Les séquences σ' obtenues sont concaténées (lignes 12-16) pour obtenir une séquence finale σ.

Pour étayer notre propos, 1000000 séquences ont été générées pour chaque méthode de génération de séquences sur des instances présentant des contraintes de qualification et de disponibilités journalières. Les instances choisies sont de taille raisonnable pour que l'algorithme de décodage puisse les décoder en quelques millisecondes. La génération totalement aléatoire ne génère que 718 séquences réalisables si la sous-traitance n'est pas autorisée. Par contre, la génération réalisée à partir de l'algorithme 8 permet d'augmenter de façon substantielle le nombre de séquences réalisables puisque 634763 séquences obtenues sont réalisables. Cette augmentation du nombre de solutions réalisables vient essentiellement de la satisfaction de la condition nécessaire de faisabilité.

Enfin, afin de maintenir la diversité au sein de la population initiale, chaque nouvelle solution ς générée est acceptée si elle satisfait à la condition de distance :

$$D_{\varsigma,\varsigma'} \geq \Delta, \forall \varsigma' \in P$$

Le paramètre de diversification Δ représente la distance minimale autorisée entre deux solutions distinctes de la population.

L'ensemble du processus est itéré tant que la taille définie $|P|$ de la population initiale n'est pas atteinte. Cette méthode d'initialisation de la population permet d'éviter l'existence de solutions trop proches au sein de la population et d'essayer de représenter de façon uniforme l'espace de recherche.

Algorithm 8 Génération de séquences pour le HCRSP

$\sigma \leftarrow \langle\rangle$ ▷ création d'une séquence σ vide
for $i \in \mathscr{S}$ **do**
  if $\exists j \mid \delta_{ij}^{min} \geq 0 \vee \delta_{ij}^{max} \geq 0$ **then**
    Choisir aléatoirement une tournée (k, d) telle que $q_{ik} = \top \wedge d \in \mathscr{H}_i$ et le
    service j ne soit pas affecté à la tournée (k, d)
    Affecter le service i à la tournée (k, d)
  else
    Choisir aléatoirement une tournée (k, d) telle que $q_{ik} = \top \wedge d \in \mathscr{H}_i$
    Affecter le service i à la tournée (k, d)
  end if
end for
for $(k, d) \in \mathscr{K} \times \mathscr{H}$ **do**
  Trier les services affectés à la tournée (k, d) par ordre croissant du milieu de la
  première fenêtre de temps : soit σ' la séquence obtenue
  $\sigma \leftarrow \sigma \oplus \sigma'$ ▷ concaténation
end for
return σ

5.6 Sélection

Afin de déterminer les solutions p_1 et p_2 qui seront croisées, deux tournois binaires ont été mis en place. Deux solutions sont choisies aléatoirement, grâce à une loi de probabilité uniforme, dans l'ensemble la population P. La solution de coût moindre est celle qui remporte le tournoi. Dans ces tournois, nous ne séparons pas les solutions réalisables des solutions irréalisables. Ainsi, les solutions p_1 et p_2 peuvent être une solution réalisable et une autre irréalisable. Leur croisement favorise l'approche de la frontière séparant l'espace réalisable et irréalisable où se situent les solutions les plus prometteuses.

5.7 Croisement

À partir des solutions p_1 et p_2 issues du processus de sélection décrit ci-dessus, des solutions o_1 et o_2 vont être générées en les croisant. Plusieurs opérateurs de croisement issus de la littérature ont été envisagés et comparés. La diversité des contraintes du problème nous oblige à avoir cette démarche. [Puljić et Manger, 2013] a implémenté plusieurs opérateurs de croisement et comparé leurs performances. Dans les lignes qui suivent, nous décrivons les opérateurs de croisement implémentés. La séquence obtenue à l'issue du croisement est décodée pour être la solution ς. Dans les sous-sections qui suivent, nous décrivons les trois opérateurs de croisement qui ont été envisagés : ORDER CROSSOVER (OX), PARTIALLY MAPPED CROSSOVER (PMX), ALTERNATING EDGES CROSSOVER (AEX).

5.7.1 ORDER CROSSOVER - OX

Deux nombres i et j, $i \leq j$ sont tirés aléatoirement les sous-séquences $\langle p_1(i) \dots p_1(j) \rangle$ et $\langle p_2(i) \dots p_2(j) \rangle$ sont copiées respectivement dans les sous-séquences $\langle o_1(i) \dots o_1(j) \rangle$ et $\langle o_2(i) \dots o_2(j) \rangle$. À partir de $j+1$, en parcourant de façon circulaire l'autre parent, le chromosome courant est complété en évitant de dupliquer les valeurs déjà présentes. La figure 5.1 illustre la construction de deux chromosomes o_1 et o_2 à partir des chromosomes p_1 et p_2. Dans cet exemple, les valeurs de i et j sont respectivement 3 et 5.

$p_1 \rightarrow$	5	4	8	9	1	6	3	2	7
$p_2 \rightarrow$	9	1	2	3	7	8	6	4	5
$o_1 \rightarrow$	3	7	8	9	1	6	4	5	2
$o_2 \rightarrow$	9	1	2	3	7	6	5	4	8

FIGURE 5.1 – Exemple de l'opérateur de croisement OX – CROSSOVER

5.7.2 PARTIALLY MAPPED CROSSOVER - PMX

Tout comme pour l'opérateur de croisement OX, deux nombres i et j, $i \leq j$ sont également tirés aléatoirement. Les sous-séquences $\langle p_1(i) \dots p_1(j) \rangle$ et $\langle p_2(i) \dots p_2(j) \rangle$ sont copiées respectivement dans les sous-séquences $\langle o_1(i) \dots o_1(j) \rangle$ et $\langle o_2(i) \dots o_2(j) \rangle$. Ces copies nous permettent d'établir une correspondance entre $o_1(k)$ et $o_2(k)$, $i \leq k \leq j$. Le contenu des cellules de l'autre parent est recopié dans chaque chromosome si cela est possible. Enfin, à partir de la correspondance pré-établie, chaque chromosome o_1 et o_2 est complété de sorte à éviter de dupliquer les valeurs dans les séquences. La figure 5.2 illustre la construction de deux séquences o_1 et o_2 à partir des séquences p_1 et p_2. Dans cet exemple, les valeurs de i et j valent 3 et 5. Nous obtenons les correspondances suivantes : $8 \leftrightarrow 2$, $9 \leftrightarrow 3$, $1 \leftrightarrow 7$. Trois étapes distinctes de construction des chromosomes o_1 et o_2 sont détaillées : la recopie des sous-séquences $\langle 8\ 9\ 1 \rangle$ $\langle 2\ 3\ 7 \rangle$; la complétion du chromosome o_1 à partir p_2 et de o_2 à partir de p_1 si cela est possible. Enfin, à partir des correspondances pré-établies, la complétion des chromosomes o_1 et o_2.

$p_1 \rightarrow$	5	4	8	9	1	6	3	2	7
$p_2 \rightarrow$	9	1	2	3	7	8	6	4	5
$o_1 \rightarrow$	–	–	8	9	1	–	–	–	–
$o_2 \rightarrow$	–	–	2	3	7	–	–	–	–
$o_1 \rightarrow$	–	–	8	9	1	–	6	4	5
$o_2 \rightarrow$	5	4	2	3	7	6	–	–	–
$o_1 \rightarrow$	3	7	8	9	1	2	6	4	5
$o_2 \rightarrow$	5	4	2	3	7	6	9	8	1

FIGURE 5.2 – Exemple de l'opérateur de croisement PMX

5.7.3 ALTERNATING EDGES CROSSOVER - AEX

L'opérateur de croisement AEX prend alternativement des sous-séquences de taille 2 des parents pour créer un nouveau chromosome en choisissant alternativement une sous-séquence de taille 2 d'un des deux parents p_1 et p_2. Lorsqu'un choix n'est pas possible parce que l'on se trouve en fin de séquence ou qu'une valeur existe déjà dans le chromosome à générer, une position est tirée aléatoirement pour compléter ce chromosome. La figure 5.3 illustre notre propos. En partant d'un chromosome p_1, l'algorithme commence par choisir la sous-séquence $5-4$ qui est de taille 2. Cette sous-séquence va constituer le début de la séquence du nouveau chromosome o_1. On bascule ensuite au chromosome p_2 pour trouver la sous-séquence suivante. La prochaine sous-séquence qui devrait être ajoutée est $4-5$ car 4 est le dernier sommet ajouté au chromosome o_1. Malheureusement, 5 existe déjà. On tire aléatoirement le prochain sommet ; ici 6. On bascule au chromosome p_1. La prochaine sous-séquence est $6-3$. Ainsi de suite. Les positions tirées aléatoirement ont été entourées. Le processus est identique pour le chromosome o_2 à la seule différence que nous commençons par le chromosome p_2.

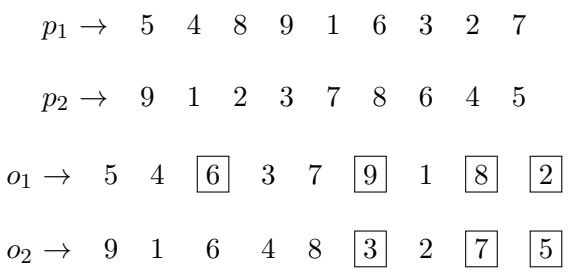


FIGURE 5.3 – Exemple de l'opérateur de croisement AEX

5.8 Éducation

L'algorithme hybride génétique se distingue de l'algorithme génétique par une procédure de recherche locale. Dans l'algorithme 7, la recherche locale fait office d'opérateur d'éducation. Le voisinage de la solution ς issue du croisement des solutions p_1 et p_2 est exploré avec une probabilité $prob_e$. Cette étape de l'algorithme correspond donc à l'intensification de la recherche en réparant la solution ς si elle est irréalisable ou en l'améliorant. Le processus de gestion de la population prémuni l'algorithme hybride génétique d'une convergence précoce.

Après quelques expérimentations, il apparaît plus efficace de réaliser la recherche locale sur l'encodage direct car cela reste le meilleur compromis entre le temps de calcul et la qualité des solutions produites. En effet, selon les caractéristiques du problème en entrée, l'appel de l'algorithme de décodage *Split* peut s'avérer coûteux en temps de calcul. Nous proposons une procédure de recherche locale combinant l'application à la solution ς de la première amélioration rencontrée et l'arrêt à l'épuisement de tous les mouvements améliorants possibles. À partir d'un ensemble de mouvements prédéfinis, un mouvement est sélectionné aléatoirement puis testé à la solution courante ς. Si ce mouvement peut améliorer cette solution ς alors il est appliqué. Ce processus est itéré tant qu'il existe des mouvements pouvant améliorer la solution.

Les mouvements proposés ici sont habituellement utilisés dans les méthodes de résolution traitant du VRP. Comme cela a été mis en exergue par [Kindervater et Martin W.P. Salvesbergh, 1997], ces mouvements peuvent être considérés comme des concaténations de séquences. Les mouvements qui suivent ont été proposés par [Prins, 2004]. Nous les reprenons pour notre approche.

Soient deux tournées (k, d) et (k', d') réalisant respectivement les séquences de services $\sigma = \langle \sigma_0, \sigma_1, \ldots, \sigma_{|\sigma|-1} \rangle$ et $\sigma' = \langle \sigma'_0, \sigma'_1, \ldots, \sigma'_{|\sigma'|-1} \rangle$. Les services σ_0 et σ'_0 sont des services fictifs et représentent le sommet de départ d'un soignant. Soient les services j et v les successeurs respectifs des services i et u dans les tournées (k, d) et (k', d').

— **mouvement 1** : déplacer i après u, $i \neq \sigma_0$,
— **mouvement 2** : déplacer (i, j) après u, $i \neq \sigma_0$,
— **mouvement 3** : déplacer (j, i) après u, $i \neq \sigma_0$,
— **mouvement 4** : échanger i et u, $i \neq \sigma_0$ et $u \neq \sigma'_0$,
— **mouvement 5** : échanger (i, j) et u, $i \neq \sigma_0$ et $u \neq \sigma'_0$,

— **mouvement 6** : échanger (i, j) et (u, v), $i \neq \sigma_0$ et $u \neq \sigma'_0$,
— **mouvement 7** : remplacer (i, j) et (u, v) par (i, u) et (j, v) si $(k, d) = (k', d')$,
— **mouvement 8** : remplacer (i, j) et (u, v) par (i, u) et (j, v) si $(k, d) \neq (k', d')$,
— **mouvement 9** : remplacer (i, j) et (u, v) par (i, v) et (j, u) si $(k, d) \neq (k', d')$,

5.9 Gestion de la population

Stratégie de gestion de la population. Plusieurs stratégies de gestion de la population ont été proposées par [Sörensen et Sevaux, 2006]. Ces stratégies font évoluer différemment la valeur du paramètre de diversification Δ. Deux écueils doivent être évités : une valeur trop élevée conduirait au rejet de nombreuses solutions et donc à une trop grande diversification de la recherche. Par contre, une valeur trop faible conduirait à une convergence précoce. Les stratégies de gestion proposées sont les suivantes :

— la valeur de Δ est constante tout au long de la recherche. Cette stratégie assure la diversité de la population si la valeur de Δ n'est pas nulle. Aucune solution ne peut être dupliquée dans la population P. Cependant, une valeur de Δ mal calibrée, trop élevée par exemple, conduirait au rejet de nombreuses solutions issues du croisement.
— la valeur de Δ est décroissante tout au long de la recherche. Cette stratégie assure la diversité de la population de l'algorithme au début de la recherche et fait converger la population pas à pas.
— la valeur de Δ est croissante au début de la recherche puis devient constante lorsqu'un seuil est atteint. Cette stratégie est par conséquent l'inverse de la stratégie précédente puisqu'elle diversifie la population à la fin de la recherche.
— la valeur de Δ est adaptative c'est-à-dire alterne des périodes croissante et décroissante. Les périodes pour lesquelles la valeur de Δ est croissante permettent de diversifier la population. Par contre, les périodes pour lesquelles la valeur de Δ est décroissante tendent à faire converger la population. Cette stratégie est plus complexe que les stratégies précédentes et nécessite une calibration plus importante.

La stratégie (Figure 5.4) que nous proposons est une variante de la stratégie adaptative. Le paramètre de diversification Δ est d'abord initialisé à sa valeur maximale et décrémente au fur et à mesure en cas d'amélioration. Lorsqu'un nombre d'itérations sans amélioration de la meilleure solution connue est atteint, la population est diversifiée et le paramètre Δ ré-initialisé à sa valeur de début. En cas de convergence prématurée, la ré-initialisation du paramètre Δ et la diversification de la population permet d'atteindre d'autres zones de l'espace de solutions.

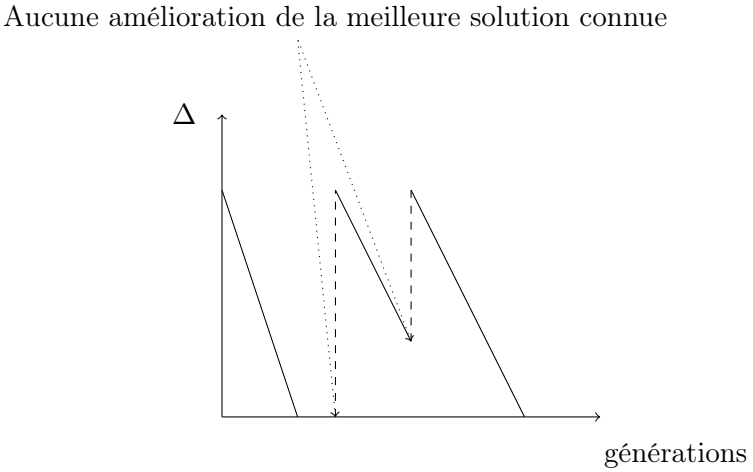


FIGURE 5.4 – Stratégie de gestion de la population mise en place

Diversification de la population. Au fil des itérations, l'éducation conduit à faire converger les solutions puisque le paramètre Δ décroît. Il faut donc un mécanisme permettant de s'en échapper. Un mécanisme de diversification de la population a été mis en place à cet effet. La diversification intervient lorsque la meilleure solution connue n'est pas améliorée au bout de I_{div} itérations. Elle intègre dans la population des solutions appartenant à de nouveaux espaces de recherche. Elle est composée de deux étapes :

— la première étape consiste à purger d'éventuels clones de la population P. Un clone est une solution disposant de la même représentation ou du même coût qu'une autre. Ainsi, toute solution de la population de distance de Hamming nulle ou de coût identique à une autre est supprimée.

— la deuxième étape consiste à intégrer de nouvelles solutions dans la population. Pour ce faire, le processus d'initialisation de la population décrit dans l'algorithme 8 est appelé à cet effet. Cette étape s'achève dès que la population atteint une taille prédéfinie.

5.10 Expérimentations numériques

Les tableaux 5.1, 5.2, 5.3, 5.4 présentent les résultats des expérimentations numériques réalisées sur les instances de [Bredström et Rönnqvist, 2008] et les tableaux 5.5, 5.6 ceux sur les instances de [Mankowska *et al.*, 2014]. Ces résultats sont comparés aux résultats de la littérature. Comme pour les expérimentations numériques réalisées au chapitre précédent, les colonnes de chaque tableau représentent respectivement le nom de l'instance, les résultats de la programmation mathématique (CPLEX), les résultats d'une méthode de résolution de la littérature à laquelle nous nous comparons et les résultats de l'algorithme hybride génétique (HGA) décrit ci-dessus. Les valeurs mises en gras correspondent aux meilleures solutions. Le gap est toujours calculé par rapport à la meilleure solution obtenue entre la programmation mathématique et la méthode de résolution à laquelle nous nous comparons. Après différentes expérimentations, nous avons retenu le calibrage suivant :

— la taille de la population $|P|$: 23
— la mesure de distance Δ : $\frac{|\mathscr{S}|}{2}$
— le nombre d'itérations sans améliorations I_{max} : 1000
— l'opérateur de croisement : OX
— le nombre d'itérations avant diversification I_{div} : $0.3 \cdot I_{max}$
— la probabilité d'éducation : 0.5

Les tableaux 5.1, 5.2 rendent compte des résultats obtenus et de leurs comparaisons à ceux de l'heuristique (Bredstrom H) proposée par [Bredström et Rönnqvist, 2008] en optimisant respectivement le temps total de trajet et les préférences. Dans le premier tableau où il s'agit de minimiser le temps total de trajet, l'algorithme hybride génétique (HGA) domine l'heuristique proposée par [Bredström et Rönnqvist, 2008] en terme de qualité des solutions retournées. En effet, l'algorithme hybride génétique (HGA) retrouve ou améliore 29 des 30 instances. En plus, de nouvelles solutions sont obtenues pour les grandes instances. La seule instance pour laquelle l'heuristique de [Bredström et Rönnqvist, 2008] est meilleure, le gap obtenu n'est que de 1.47%. Cependant, l'algorithme hybride génétique nécessite des temps de calcul plus importants. Comme nous l'avions souligné dans la description de l'algorithme, c'est un des défauts de l'algorithme. Ces temps de calcul s'expliquent également par le fait que la recherche locale n'exploite pas la structure du problème en entrée. Cela permettrait de réduire le nombre d'évaluations grâce à des tests de faisabilité qui seraient en temps constant. Dans le deuxième tableau où il s'agit d'optimiser les préférences, les résultats vont globalement dans le même sens que dans le tableau précédent. En effet, l'algorithme hybride génétique retrouve ou améliore 23 des 30 solutions si il est comparée à l'heuristique de [Bredström et Rönnqvist, 2008]. Pour les instances où l'heuristique de [Bredström et Rönnqvist, 2008] est meilleure que l'algorithme hybride génétique, le gap le plus élevé est de 9.47%. Nous observons également ici des temps de calcul plus élevé.

Dans les tableaux 5.3 et 5.4, l'algorithme hybride génétique est comparée à l'algorithme de recuit simulé hybridé à une recherche locale itérée (Afifi SA-ILS) proposée par [Afifi *et al.*, 2016]. Cet algorithme de recuit simulé reste le meilleur algorithme connu à ce jour pour le problème modélisé par [Bredström et Rönnqvist, 2008]. Dans ces deux tableaux, l'algorithme hybride génétique est dominé par l'algorithme de recuit simulé hybridé à une recherche locale itérée tant par la qualité des solutions obtenues que par le temps de calcul. Malgré ces résultats, nous obtenons dans le pire des cas, pour le tableau 5.3 un gap de 9.25%, et pour le tableau 5.4, un gap de 11.06%. Ces résultats s'expliquent par l'exploitation de la structure du problème par [Afifi *et al.*, 2016]. Ainsi, les tests de faisabilité s'effectuent en temps constant. Par ce biais, un plus grand nombre de solutions est testé.

Les tableaux 5.5, 5.6 rassemblent les résultats expérimentaux réalisés sur les instances de [Mankowska *et al.*, 2014] en utilisant l'algorithme hybride génétique. Dans le tableau 5.5 traitant des instances des groupes A,B et C, l'algorithme hybride génétique retrouve ou améliore les meilleures solutions connues de 19 sur 30 instances. Nous nous apercevons que l'algorithme hybride génétique domine la méthode AVNS proposée par [Mankowska *et al.*, 2014] en ce qui concerne les solutions obtenues malgré des temps de calcul plus importants. En effet, pour toutes les instances des groupes A, B et C, l'algorithme hybride améliore ou retrouve les solutions obtenues par la méthode AVNS. Nous pensons que le processus de diversification mis en place dans l'algorithme hybride génétique permet d'avoir accès

à des zones de l'espace de recherche auxquelles la méthode AVNS accède difficilement. Les conclusions que nous venons de formuler restent valables pour les instances de taille moyenne et grande. En effet, dans le tableau 4.6 rassemblant les instances des groupes D, E, F et G, l'algorithme hybride améliore systématiquement la meilleure solution connue, sauf pour l'instance D10, avec un gain allant jusqu'à 19.32%. Comme nous pouvions nous y attendre, l'algorithme hybride génétique nécessite ici aussi des temps de calcul plus élevés et pour les très grandes instances (le groupe G), consomme tout le temps de calcul autorisé.

Instance	CPLEX		Bredstrom H		HGA		
	Z	CPU(s)	Z	CPU(s)	Z	GAP(%)	CPU(s)
01S	**3.55**	0.59	**3.55**	120.03	**3.55**	0.0	15.13
01M	**3.55**	1.60	**3.55**	120.25	**3.55**	0.0	15.78
01L	**3.39**	7.10	**3.39**	120.48	**3.39**	0.0	17.77
02S	**4.27**	0.19	**4.27**	120.07	**4.27**	0.0	14.38
02M	**3.58**	1.22	**3.58**	120.11	**3.58**	0.0	10.96
02L	**3.42**	21.68	**3.42**	120.95	**3.42**	0.0	12.77
03S	**3.63**	0.61	**3.63**	120.26	**3.63**	0.0	14.30
03M	**3.33**	1.64	**3.33**	120.19	**3.33**	0.0	15.10
03L	**3.29**	9.27	**3.29**	120.6	**3.29**	0.0	11.49
04S	**6.14**	1.16	**6.14**	120.16	**6.14**	0.0	17.05
04M	**5.67**	22.56	**5.67**	120.15	**5.67**	0.0	28.16
04L	**5.13**	2895.71	5.3	120.04	5.29	3.11	22.94
05S	**3.93**	0.45	**3.93**	120.13	**3.93**	0.0	18.02
05M	**3.53**	0.49	**3.53**	120.09	**3.53**	0.0	14.02
05L	**3.34**	1.74	**3.34**	120.85	**3.34**	0.0	15.86
06S	**8.14**	3670.13	**8.14**	600.94	8.26	1.47	234.57
06M	8.10	3707.58	11.63	609.58	**7.99**	1.37	385.054
06L	-	3656.58	-	624.06	**7.56**	-	165.13
07S	**8.39**	3647.23	8.97	603.97	8.75	4.29	235.02
07M	-	3603.96	-	648.02	**7.97**	-	371.13
07L	-	3605.11	-	645.33	**7.13**	-	162.55
08S	11.34	3602.30	-	657.03	**9.90**	-12.69	955.55
08M	-	3601.67	-	632.61	**9.33**	-	551.60
08L	-	3604.41	-	618.63	**8.25**	-	486.80
09S	-	3600.30	-	626.26	**12.58**	-	1642.56
09M	-	3616.31	-	612.19	**11.64**	-	2656.28
09L	-	3616.12	-	607.36	**11.05**	-	2052.13
10S	-	3613.17	-	604.46	**8.84**	-	943.92
10M	-	3618.48	-	705.2	**8.11**	-	733.56
10L	-	3657.72	-	631.39	**8.02**	-	761.75

TABLE 5.1 – Résultats pour les instances de [Bredström et Rönnqvist, 2008] en optimisant le temps total de trajet et comparaison avec l'heuristique proposée par [Bredström et Rönnqvist, 2008]

Instance	CPLEX		Bredstrom H		HGA		
	Z	CPU(s)	Z	CPU(s)	Z	GAP(%)	CPU(s)
01S	**-114.03**	0.27	**-114.03**	3600	**-114.03**	0.0	7.52
01M	**-117.80**	0.33	**-117.80**	3600	**-117.80**	0.0	9.37
01L	**-118.51**	0.73	**-118.51**	3600	**-118.51**	0.0	7.76
02S	**-92.09**	0.20	**-92.09**	3600	**-92.09**	0.0	8.62
02M	**-104.81**	4.93	**-104.81**	3600	**-104.81**	0.0	9.07
02L	**-107.64**	141.25	**-107.64**	3600	-102.51	4.75	6.07
03S	**-99.49**	0.36	**-99.49**	3600	**-99.49**	0.0	12.56
03M	**-106.59**	0.69	**-106.59**	3600	-102.45	3.88	8.75
03L	**-107.87**	1.44	**-107.87**	3600	-106.59	1.18	9.83
04S	**-100.00**	0.35	**-100.00**	3600	**-100.00**	0.0	10.53
04M	**-106.72**	7.28	**-106.72**	3600	-96.616	9.47	14.01
04L	**-109.27**	178.08	**-109.27**	3600	-105.27	3.66	15.95
05S	**-76.29**	0.11	**-76.29**	3600	**-76.29**	0.0	9.62
05M	**-76.29**	0.42	**-76.29**	3600	**-76.29**	0.0	10.81
05L	**-84.21**	3.20	**-84.21**	3600	**-84.21**	0.0	8.70
06S	**-370.06**	362.64	**-370.06**	3600	-360.88	2.48	133.57
06M	**-379.88**	3601.08	-374.257	3600	-372.172	2.02	118.74
06L	-	3600.75	-368.876	3600	**-384.34**	-4.19	143.46
07S	**-401.11**	813.57	-296.725	3600	-397.52	0.89	116.45
07M	-	3600.91	-368.565	3600	**-403.04**	-9.35	130.69
07L	-	3600.82	-355.716	3600	**-407.09**	-14.44	120.69
08S	**-351.99**	3600.99	-	3600	-338.63	3.79	352.82
08M	-	3600.88	-	3600	**-386.87**	-	195.11
08L	-	3600.88	-	3600	**-392.53**	-	177.64
09S	-	3600.32	-	3600	**-547.95**	-	1290.92
09M	-	3600.82	-	3600	**-625.66**	-	1021.03
09L	-	3601.77	-	3600	**-635.37**	-	1153.88
10S	-	3601.49	-	3600	**-656.70**	-	701.88
10M	-	3602.20	-	3600	**-673.51**	-	988.43
10L	-	3600.11	-445.027	3600	**-679.63**	-52.71	681.10

TABLE 5.2 – Résultats pour les instances de [Bredström et Rönnqvist, 2008] en optimisant les préférences et comparaison avec l'heuristique proposée par [Bredström et Rönnqvist, 2008]

Instance	CPLEX		Afifi SA-ILS		HGA		
	Z	CPU(s)	Z	CPU(s)	Z	GAP(%)	CPU(s)
01S	**3.55**	0.59	**3.55**	0.02	**3.55**	0.0	15.13
01M	**3.55**	1.60	**3.55**	0.02	**3.55**	0.0	15.78
01L	**3.39**	7.10	**3.39**	0.03	**3.39**	0.0	17.77
02S	**4.27**	0.19	**4.27**	0.02	**4.27**	0.0	14.38
02M	**3.58**	1.22	**3.58**	0.03	**3.58**	0.0	10.96
02L	**3.42**	21.68	**3.42**	0.03	**3.42**	0.0	12.77
03S	**3.63**	0.61	**3.63**	0.02	**3.63**	0.0	14.30
03M	**3.33**	1.64	**3.33**	0.03	**3.33**	0.0	15.10
03L	**3.29**	9.27	**3.29**	0.02	**3.29**	0.0	11.49
04S	**6.14**	1.16	**6.14**	0.02	**6.14**	0.0	17.05
04M	**5.67**	22.56	**5.67**	0.05	**5.67**	0.0	28.16
04L	**5.13**	2895.71	**5.13**	0.09	5.29	3.11	22.94
05S	**3.93**	0.45	**3.93**	0.03	**3.93**	0.0	18.02
05M	**3.53**	0.49	**3.53**	0.03	**3.53**	0.0	14.02
05L	**3.34**	1.74	**3.34**	0.03	**3.34**	0.0	15.86
06S	**8.14**	3670.13	**8.14**	13.97	8.26	1.47	234.57
06M	8.10	3707.58	**7.7**	26.68	7.99	3.76	385.054
06L	-	3656.58	**7.14**	15.86	7.56	5.88	165.13
07S	**8.39**	3647.23	**8.39**	15.08	8.75	4.29	235.02
07M	-	3603.96	**7.48**	18.34	7.97	6.55	371.13
07L	-	3605.11	**6.88**	15.92	7.13	3.63	162.55
08S	11.34	3602.30	**9.54**	25.13	9.90	3.77	955.55
08M	-	3601.67	**8.54**	15.01	9.33	9.25	551.60
08L	-	3604.41	**8**	24.51	8.25	3.12	486.80
09S	-	3600.30	**11.93**	150.52	12.58	5.44	1642.56
09M	-	3616.31	**10.92**	292.17	11.64	6.59	2656.28
09L	-	3616.12	**10.49**	207.17	11.05	5.33	2052.13
10S	-	3613.17	**8.60**	16.10	8.84	2.79	943.92
10M	-	3618.48	**7.62**	52.75	8.11	6.43	733.56
10L	-	3657.72	**7.75**	51.89	8.02	3.48	761.75

TABLE 5.3 – Résultats pour les instances de [Bredström et Rönnqvist, 2008] en optimisant le temps total de trajet et comparaison avec l'heuristique proposée par [Afifi *et al.*, 2016]

Instance	CPLEX		Afifi SA-ILS		HGA		
	Z	CPU(s)	Z	CPU(s)	Z	GAP(%)	CPU(s)
01S	**-114.03**	0.27	**-114.03**	0.03	**-114.03**	0.0	7.52
01M	**-117.80**	0.33	**-117.80**	0.02	**-117.80**	0.0	9.37
01L	**-118.51**	0.73	**-118.51**	0.04	**-118.51**	0.0	7.76
02S	**-92.09**	0.20	**-92.09**	0.05	**-92.09**	0.0	8.62
02M	**-104.81**	4.93	**-104.81**	0.04	**-104.81**	0.0	9.07
02L	**-107.64**	141.25	**-107.64**	0.38	-102.51	4.76	6.07
03S	**-99.49**	0.36	**-99.49**	0.02	**-99.49**	0.0	12.56
03M	**-106.59**	0.69	**-106.59**	0.07	-102.45	3.88	8.75
03L	**-107.87**	1.44	**-107.87**	0.14	-106.59	1.18	9.83
04S	**-100.00**	0.35	**-100.00**	0.03	**-100.00**	0.0	10.53
04M	**-106.72**	7.28	**-106.72**	0.07	-96.61	9.47	14.01
04L	**-109.27**	178.08	**-109.27**	0.13	-105.27	4.05	15.95
05S	**-76.29**	0.11	**-76.29**	0.02	**-76.29**	0.0	9.62
05M	**-76.29**	0.42	**-76.29**	0.03	**-76.29**	0.0	10.81
05L	**-84.21**	3.20	**-84.21**	0.04	**-84.21**	0.0	8.70
06S	**-370.06**	362.64	**-370.06**	0.7	-360.888	2.48	133.57
06M	**-379.88**	3601.08	**-379.88**	25.26	-372.172	2.02	118.74
06L	-	3600.75	**-387.2**	16.33	-384.34	0.74	143.46
07S	**-401.11**	813.57	**-401.11**	0.58	-397.522	0.89	116.45
07M	-	3600.91	**-406.17**	6.48	-403.045	0.76	130.69
07L	-	3600.82	**-407.48**	2.53	-407.09	0.09	120.69
08S	-351.99	3600.99	**-380.76**	26.12	-338.633	11.06	352.82
08M	-	3600.88	**-403.57**	59.34	-386.879	4.13	195.11
08L	-	3600.88	**-407.48**	20.51	-392.533	3.66	177.64
09S	-	3600.32	**-581.12**	117.4	-547.959	5.70	1290.92
09M	-	3600.82	**-656.5**	10.90	-625.669	4.69	1021.03
09L	-	3601.77	**-666.5**	17.81	-635.37	4.67	1153.88
10S	-	3601.49	**-675.81**	162.42	-656.707	2.82	701.88
10M	-	3602.20	**-686.75**	150.63	-673.511	1.92	988.43
10L	-	3600.11	**-691.48**	270.26	-679.631	1.71	681.10

TABLE 5.4 – Résultats pour les instances de [Bredström et Rönnqvist, 2008] en optimisant les préférences et comparaison avec l'heuristique proposée par [Afifi *et al.*, 2016]

Instance	CPLEX		Mankowska AVNS		HGA		
	Z	CPU(s)	Z	CPU(s)	Z	GAP(%)	CPU(s)
A01	**218.20**	0.23	**218.2**	1	**218.19**	0.0	13.25
A02	**246.62**	0.25	248.1	1	**246.62**	0.0	14.70
A03	**305.86**	0.32	**305.9**	1	**305.86**	0.0	13.78
A04	**186.90**	0.92	**186.9**	1	**186.89**	0.0	12.87
A05	**189.54**	0.35	192.0	1	**189.54**	0.0	12.15
A06	**200.10**	0.26	**200.1**	1	**200.09**	0.0	11.35
A07	**225.37**	0.23	**225.4**	1	**225.36**	0.0	15.00
A08	**232.05**	0.18	**232.0**	1	**232.04**	0.0	11.55
A09	**222.29**	1.70	**222.3**	1	**222.29**	0.0	14.59
A10	**225.01**	0.05	**225.0**	1	**225.00**	0.0	6.83
B01	**428.10**	12711.60	458.9	1	442.34	3.32	135.15
B02	**476.05**	5944.16	580.9	1	**476.05**	0.0	83.29
B03	**399.09**	22530.20	431.4	1	418.01	4.74	142.82
B04	411.30	36001.90	587.3	1	439.72	6.90	82.59
B05	**366.34**	13114.60	391.1	1	385.17	5.14	107.69
B06	440.93	36000.70	545.9	1	495.90	12.46	142.81
B07	**328.67**	442.95	356.6	1	329.05	0.11	157.62
B08	**357.68**	575.17	410.9	1	**357.68**	0.0	101.55
B09	402.67	36001.10	487.9	1	407.64	1.23	241.30
B10	469.58	36001.40	500.4	1	484.27	3.12	122.79
C01	-	36000.90	1123.6	1	**969.23**	-13.73	550.311
C02	-	36000.70	677.0	1	**606.52**	-10.41	797.77
C03	547.54	36029.40	642.4	1	582.38	6.36	921.291
C04	-	36013.40	580.4	1	**537.00**	-7.47	775.965
C05	691.56	36004.00	754.6	1	712.04	2.96	454.685
C06	-	36001.20	951.6	1	**894.19**	-6.03	865.126
C07	-	36002.00	577.4	1	**542.14**	-6.10	1695.52
C08	469.49	36005.50	540.6	1	491.08	4.59	712.943
C09	-	36001.10	608.7	1	**604.41**	-0.70	850.141
C10	-	36001.50	679.3	1	**604.55**	-11.00	705.396

TABLE 5.5 – Résultats pour les instances de [Mankowska *et al.*, 2014] en optimisation une somme pondérée des critères suivants : la distance totale parcourue, le retard maximal et la somme des retards. Le poids de chaque critère est $\frac{1}{3}$. Comparaison avec la méthode de résolution AVNS proposée par [Mankowska *et al.*, 2014]

Instance	CPLEX		Mankowska AVNS		HGA		
	Z	CPU(s)	Z	CPU(s)	Z	GAP(%)	CPU(s)
D01	-	36004.30	1321.8	5	**1256.42**	-4.94	3481.78
D02	-	36003.10	892.7	4	**793.93**	-11.06	3588.56
D03	-	36002.30	819.4	6	**703.02**	-14.20	3478.51
D04	-	36010.50	877.4	4	**803.57**	-8.41	3215.44
D05	-	36013.90	872.1	7	**743.23**	-14.77	3538.24
D06	-	36010.00	835.2	8	**756.23**	-9.45	3443.31
D07	-	36009.80	706.3	7	**623.44**	-11.73	4826.47
D08	-	36007.10	811.4	7	**706.84**	-12.88	4167.94
D09	-	36006.40	860.3	7	**742.37**	-13.70	4030.75
D10	-	36002.70	**1306.6**	4	1441.11	10.29	2977.86
E01	-	36025.30	1604.9	31	**1407.87**	-12.27	2651.22
E02	-	36180.20	1159.7	27	**894.271**	-22.88	4176.65
E03	-	36104.70	986.4	28	**860.406**	-12.77	2859.89
E04	-	36360.30	871.0	37	**743.28**	-14.66	3550.01
E05	-	36104.20	1018.0	29	**827.122**	-18.75	2997.7
E06	-	36159.30	1003.0	36	**853.422**	-14.91	3354.84
E07	-	36130.20	921.1	38	**751.321**	-18.43	3118.11
E08	-	36162.20	884.6	36	**810.415**	-8.38	3516.41
E09	-	36190.80	1131.7	34	**1033.71**	-8.65	3413
E10	-	36008.50	1066.6	31	**860.433**	-19.32	2010.92
F01	-	36241.80	1721.4	889	**1489.05**	-13.49	19785.8
F02	-	36000.00	1763.8	909	**1471.67**	-16.56	19482.2
F03	-	36000.00	1549.6	868	**1269.7**	-18.06	11664.1
F04	-	36000.00	1420.4	1321	**1165.89**	-17.91	13669.5
F05	-	36000.00	1701.9	1145	**1461.54**	-14.12	12456.7
F06	-	36000.00	1639.7	836	**1462.7**	-10.794	15237.5
F07	-	36000.00	1384.3	1294	**1222.78**	-11.66	12720.4
F08	-	36000.00	1544.6	924	**1246.81**	-19.27	13711.8
F09	-	36000.00	1572.9	1642	**1352.12**	-14.03	22387.4
F10	-	36000.00	1581.0	1326	**1403.37**	-11.23	15808.6
G01	-	36000.00	2248.0	7200	**2192.91**	-2.45	36000.00
G02	-	36000.00	2316.1	7200	**2115.44**	-8.66	36000.00
G03	-	36000.00	1885.3	7147	**1676.58**	-11.07	36000.00
G04	-	36000.00	2023.2	7200	**1743.15**	-13.84	36000.00
G05	-	36000.00	2247.6	7200	**2118.35**	-5.75	36000.00
G06	-	36000.00	2144.4	7200	**2092.43**	-2.42	36000.00
G07	-	36000.00	1971.5	6934	**1609.25**	-18.37	36000.00
G08	-	36000.00	1987.4	7200	**1731.28**	-12.88	36000.00
G09	-	36000.00	2415.5	7023	**2189.67**	-9.34	36000.00
G10	-	36000.00	2373.4	7003	**2172.28**	-8.47	36000.00

TABLE 5.6 – Résultats pour les instances de [Mankowska *et al.*, 2014] en optimisation une somme pondérée des critères suivants : la distance totale parcourue, le retard maximal et la somme des retards. Le poids de chaque critère est $\frac{1}{3}$. Comparaison avec la méthode de résolution AVNS proposée par [Mankowska *et al.*, 2014] (suite)

5.11 Conclusion

Dans ce chapitre, un algorithme hybride génétique avec gestion de la population pour le HCRSP a été proposé. Cette méta-heuristique s'inspire de méta-heuristiques issues de la littérature du VRP. Elle combine une représentation des solutions consistant en une séquence de services sans délimiteurs, des algorithmes d'évaluation, une procédure d'intensification et une gestion avancée de la population. Trois opérateurs de croisement ont été implémentés afin de déterminer le plus adapté au problème. Notre approche a été validée par des expérimentations numériques où l'algorithme hybride génétique est comparé à l'algorithme du chapitre précédent. Ces expérimentations nous ont conduit à tester plusieurs variantes de la méta-heuristique pour déterminer une calibration qui soit la plus performante. L'algorithme hybride génétique retrouve et améliore les meilleures solutions connues pour la plupart des instances et domine la méthode AVNS proposée par [Mankowska *et al.*, 2014] malgré des temps de calcul plus importants.

Conclusion générale

Les travaux de recherche présentés dans cette thèse s'inscrivent dans le cadre du projet "Planification et Optimisation au Niveau Opérationnel des Soins à Domicile" (PONOSAD) financé par l'Agence Nationale de la Recherche. Ce projet vise à développer des outils d'aide à la décision pour les professionnels de soins à domicile. Ces travaux traitent donc de problèmes se rattachant à la gestion des soins à domicile. Les problèmes étudiés interviennent au niveau opérationnel et sont connexes aux problèmes de transports et de tournées de véhicules. Comme nous l'avons montré dans la revue de la littérature, plusieurs études ont précédé nos travaux. Cependant, ces études étaient dédiées à une structure de soins à domicile spécifique et les résultats (modèles et algorithmes) ne pouvaient pas, ou difficilement, être transposées à une autre structure. Notre approche a donc consisté à avoir une démarche unificatrice de ces problèmes en couvrant la plupart des caractéristiques de ces problèmes, tant du point de vue de la modélisation que de la conception de méthodes de résolution.

Dans le premier chapitre, nous avons réalisé un état de l'art des travaux réalisés. Cet état de l'art nous a permis d'identifier les caractéristiques communément prises en compte dans ce type de problèmes. Nous avons proposé une classification des travaux de la littérature basée sur trois acteurs : le personnel de soins, la structure de soins à domicile et les patients. De plus, les fonctions économiques les plus fréquemment utilisées dans la littérature ont été décrites. Une étude des méthodes de résolution a été réalisée en tenant compte de l'horizon de planification et des techniques mises en œuvre. Enfin, ce premier chapitre nous a également permis de souligner les pistes de recherche qui n'avaient pas encore été envisagées. La généricité et la prise en compte de l'aspect dynamique et stochastique représentent les principales pistes de recherche mises en exergue.

Le deuxième chapitre traite d'une de ces pistes de recherche. En effet, après avoir décrit un problème où la plupart des caractéristiques citées précédemment ont été prises en compte, nous avons formulé un programme linéaire en nombre entiers. Nous avons tenu à proposer un modèle aussi générique que possible. Nous avons également souligné les limites de la formulation proposée à savoir l'absence de l'aspect dynamique et stochastique. À partir de ce modèle, une génération d'instances a été réalisée et plusieurs instances de la littérature ont été converties au même format. Enfin, des expérimentations numériques ont été réalisées pour valider notre approche en implémentant le modèle grâce au solveur de programmation mathématique CPLEX.

Dans le troisième chapitre, nous avons présenté deux encodages d'une solution et des algorithmes de décodage. Le premier encodage est une extension d'un encodage bien connu

dans la littérature des problèmes de tournées de véhicules : un tour géant sans délimiteurs. L'encodage que nous avons proposé prend en compte les spécificités du problème défini au chapitre 2. Nous avons présenté ensuite des algorithmes exact et approché de décodage. Ces algorithmes sont des algorithmes d'étiquetage pour le calcul de plus court chemin dans un graphe spécifique. Des techniques d'amélioration sont également introduites permettant de réduire de façon substantielle le temps de calcul. Ces techniques exploitent les symétries du problème, les bornes et les situations d'interblocage pour réduire le nombre d'étiquettes générées. Le deuxième encodage est un encodage direct représentant une solution comme plusieurs séquences de services. Chaque séquence est associée à une unique tournée. Des algorithmes d'évaluation sont décrits qui déterminent la réalisabilité d'une solution mais également son coût. Ces algorithmes sont testés et comparés pour identifier l'approche la plus adaptée.

Le quatrième chapitre traite d'une méthode par décomposition. Cette dernière décompose le problème en sous-problèmes. Séquentiellement, chaque sous-problème est résolu de manière exacte ou approchée. La décomposition retenue consiste à d'abord déterminer le jour de planification des services, ensuite les tournées auxquelles ces services seront affectées, enfin le séquencement des services au sein d'une même tournée. Dans le but d'explorer d'autres parties de l'espace de recherche, trois procédures de retour sur trace ont été envisagées. Ces procédures remettent en cause partiellement ou totalement les décisions prises au préalable. De plus, un processus d'intensification a été implémenté à l'aide d'une recherche locale. Enfin, des expérimentations ont été réalisées sur des instances de la littérature. Les résultats obtenus sont comparés et commentés.

Enfin, dans le cinquième chapitre, nous avons présenté un algorithme hybride génétique avec gestion de la population. Ce choix a été motivé par les succès de cet algorithme quant à la résolution du problème de tournées de véhicules multi-attributs. L'algorithme que nous avons proposé combine plusieurs contributions de la littérature. Une solution est encodée grâce à l'encodage indirect du chapitre 2. Une petite population de solutions, séparées les unes des autres par une mesure de distance, va évoluer au fil des générations. Grâce à une mesure de distance, la convergence précoce de l'algorithme est évitée. Lorsque les solutions sont trop proches, la population est réinitialisée pour y inclure une certaine diversité. Un processus d'intensification a été implémenté afin d'explorer le voisinage proche d'une solution. Des expérimentions numériques nous ont permis de valider notre approche et de comparer l'algorithme proposé à l'état de l'art.

Plusieurs perspectives de recherche s'offrent à nous à la suite de ces travaux. Tout d'abord, cette thèse a uniquement porté sur des problèmes de tournées apparaissant au niveau opérationnel d'une structure de soins à domicile. D'autres problèmes forts intéressants existent au niveau opérationnel et que nous souhaitons étudier. Il s'agit notamment du Home Care Assignment Problem. Il s'agit de déterminer uniquement le soignant ou le sous-ensemble de soignants le plus adéquat pour prodiguer les soins que nécessitent un patient. Aux niveaux tactique et stratégique, d'autres problèmes existent. Nous pensons au Districting Problem et au Ressource Dimensioning Problem qui ont été brièvement décrits dans le premier chapitre. Ces problèmes sont actuellement résolus séparément et les décisions prises à un niveau sont transmises à d'autres niveaux. Ce cloisonnement de ces problèmes ne permet pas une vison globale. Une approche intégrée serait une perspective très intéressante et permettrait d'améliorer la prise de décision en sortant de l'approche

myopique actuellement adoptée. Dans cette thèse d'ailleurs, nous pouvons considérer le travail réalisé comme une approche intégrée car nous traitons à la fois de l'affectation et du routage.

Comme nous l'avons souligné, le modèle formulé au chapitre 2 est un modèle statique et déterministe. Par conséquent, ce modèle ne peut pas capter certains aspects de la réalité ; d'une part, l'aspect dynamique, car les données évolues au cours du temps ; d'autre part, l'aspect stochastique, car les données ne sont pas certaines. Une extension du modèle proposé serait nécessaire pour prendre en compte ces aspects.

En ce qui concerne la représentation et son évaluation, l'algorithme d'étiquetage proposé à une complexité exponentielle dans le pire des cas. Il serait intéressant de trouver de nouvelles techniques d'amélioration et inégalités valides pour réduire le temps de calcul. Ces algorithmes d'évaluation sont appelés des milliers de fois dans une méta-heuristique donc chaque gain aussi minime soit-il permettrait de baisser le temps de calcul global.

Nous avons étendu le problème général de *timing* au cas avec plusieurs séquences liées par des contraintes de synchronisation et de délais minimal et maximal. Une revue de la littérature de problèmes connexes devra être réaliser. Ainsi, ce travail permettra de distinguer les cas polynomiaux de ceux qui ne le sont pas et d'éviter d'utiliser la programmation mathématique. Nous avons proposé l'algorithme 5 pour prendre en compte les situations de synchronisation, de délais minimal et maximal. Il est important de trouver d'autres algorithmes pour prendre en compte de nouvelles situations qui traiteront de l'exclusion, des fenêtres de temps multiples par exemple.

Dans un souci de généricité, les procédures de recherche locale mises en place n'exploitent pas la structure du problème en entrée. L'exploitation de la structure du problème réduirait le nombre d'opérations en évitant de réaliser certains mouvements irréalisables ou non profitables. L'une des clés serait la mise en place de test de réalisabilité en temps constant. Une esquisse de ce travail a été initiée par [Afifi *et al.*, 2016]. Cependant, l'étude proposée par ces auteurs n'est dédiée qu'au problème formulé par [Bredström et Rönnqvist, 2008].

Bibliographie

[Afifi *et al.*, 2016] Afifi, S., Dang, D.-C. et Moukrim, A. (2016). Heuristic solutions for the vehicle routing problem with time windows and synchronized visits. *Optimization Letters*, 10(3):511–525.

[Akjiratikarl *et al.*, 2007] Akjiratikarl, C., Yenradee, P. et Drake, P. R. (2007). PSO-based algorithm for home care worker scheduling in the UK. *Computers and Industrial Engineering*, 53(4):559–583.

[Allaoua *et al.*, 2013] Allaoua, H., Borne, S., Létocart, L. et Wolfler Calvo, R. (2013). A matheuristic approach for solving a home health care problem. *Electronic Notes in Discrete Mathematics*, 41:471–478.

[Amorim et Almada-Lobo, 2014] Amorim, P. et Almada-Lobo, B. (2014). The impact of food perishability issues in the vehicle routing problem. *Computers and Industrial Engineering*, 67(1):223–233.

[B. Bashir, M. Chabrol, 2012] B. Bashir, M. Chabrol, C. C. (2012). LITERATURE REVIEW IN HOME CARE. *In 9th International Conference of Modeling, Optimization and Simulation - MOSIM'12*, volume 2005, Bordeaux.

[Baptista *et al.*, 2002] Baptista, S., Oliveira, R. C. et Z ? ?quete, E. (2002). A period vehicle routing case study. *European Journal of Operational Research*, 139(2):220–229.

[Bard *et al.*, 2014] Bard, J. F., Shao, Y. et Jarrah, A. I. (2014). A sequential GRASP for the therapist routing and scheduling problem. *Journal of Scheduling*, 17(2):109–133.

[Bard *et al.*, 2013] Bard, J. F., Shao, Y. et Wang, H. (2013). Weekly scheduling models for traveling therapists. *Socio-Economic Planning Sciences*, 47(3):191–204.

[Begur *et al.*, 1997] Begur, S. V., Miller, D. M. et Weaver, J. R. (1997). An integrated spatial DSS for scheduling and routing home-health-care nurses. *Interfaces*, 27(4):35–48.

[Bektas, 2006] Bektas, T. (2006). The multiple traveling salesman problem : an overview of formulations and solution procedures. *Omega*, 34(3):209–219.

[Bektas et Laporte, 2011] Bektas, T. et Laporte, G. (2011). The Pollution-Routing Problem. *Transportation Research Part B : Methodological*, 45(8):1232–1250.

[Bennett, 2010] Bennett, A. R. (2010). HOME HEALTH CARE LOGISTICS PLANNING, Doctoral Dissertation. (May).

[Bennett et Erera, 2011] Bennett, A. R. et Erera, A. L. (2011). Dynamic periodic fixed appointment scheduling for home health. *IIE Transactions on Healthcare Systems Engineering*, 1(1):6–19.

[Bennett Milburn, 2012] Bennett Milburn, A. R. (2012). Operations Research Applications in Home Healthcare. *In* Hall, R., éditeur : *Handbook of Healthcare System Scheduling*, volume 168 de *International Series in Operations Research & Management Science*, chapitre 11, pages 281–302. Springer US, Boston, MA.

[Bertels et Fahle, 2006] Bertels, S. et Fahle, T. (2006). A hybrid setup for a hybrid scenario : Combining heuristics for the home health care problem. *Computers and Operations Research*, 33(10):2866–2890.

[Blais *et al.*, 2003] Blais, M., Lapierre, S. D. et Laporte, G. (2003). Solving a home-care districting problem in an urban setting. *Journal of the Operational Research Society*, 54(11):1141–1147.

[Bowers *et al.*, 2014] Bowers, J., Cheyne, H., Mould, G. et Page, M. (2014). Continuity of care in community midwifery. *Health Care Management Science*, pages 195–204.

[Braekers *et al.*, 2016] Braekers, K., Hartl, R. F., Parragh, S. N. et Tricoire, F. (2016). A bi-objective home care scheduling problem : Analyzing the trade-off between costs and client inconvenience. *European Journal of Operational Research*, 248(2):428–443.

[Bräysy et Gendreau, 2005a] Bräysy, O. et Gendreau, M. (2005a). Vehicle Routing Problem with Time Windows, Part I : Route Construction and Local Search Algorithms. *Transportation Science*, 39(1):104–118.

[Bräysy et Gendreau, 2005b] Bräysy, O. et Gendreau, M. (2005b). Vehicle Routing Problem with Time Windows, Part II : Metaheuristics. *Transportation Science*, 39(1): 119–139.

[Bräysy *et al.*, 2009] Bräysy, O., Nakari, P., Dullaert, W. et Neittaanmäki, P. (2009). An optimization approach for communal home meal delivery service : A case study. *Journal of Computational and Applied Mathematics*, 232(1):46–53.

[Bredström et Rönnqvist, 2008] Bredström, D. et Rönnqvist, M. (2008). Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research*, 191(1):19–29.

[Busby et Carter, 2006] Busby, C. R. et Carter, M. W. (2006). A Decision Tool for Negotiating Home Care Funding Levels in Ontario. *Home Health Care Services Quarterly*, 25(3-4):91–106.

[Cappanera *et al.*, 2011] Cappanera, P., Gouveia, L. et Scutellà, M. G. (2011). The skill vehicle routing problem. *Network Optimization*, pages 354–364.

[Cappanera et Scutellà, 2015] Cappanera, P. et Scutellà, M. G. (2015). Joint Assignment, Scheduling, and Routing Models to Home Care Optimization : A Pattern-Based Approach. *Tranportation Science*, 49(4):830–852.

[Chahed *et al.*, 2009] Chahed, S., Marcon, E., Sahin, E., Feillet, D. et Dallery, Y. (2009). Exploring new operational research opportunities within the Home Care context : the chemotherapy at home. *Health Care Manag Sci*, 12(2):179–191.

[Cheng et Rich, 1998] Cheng, E. et Rich, J. (1998). A home health care routing and scheduling problem. *Rice University, Texas, Tech. Rep. TR98-04*, pages 1–16.

[Cire et Hooker, 2012] Cire, A. A. et Hooker, J. N. (2012). A Heuristic Logic-Based Benders Method for the Home Health Care Problem A Heuristic Logic-Based Benders Method for the Home Health Care Problem. *Tepper School of Business*, pages 1–12.

[Cordeau *et al.*, 1997] Cordeau, J.-F., Gendreau, M. et Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2): 105–119.

[Cormen *et al.*, 2001] Cormen, T. H., Leiserson, C. E., Rivest, R. L. et Stein, C. (2001). *Introduction to Algorithms*, volume 7.

[Dantzig, 1954] Dantzig, G. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the operations ...*, 1934.

[Deb *et al.*, 2002] Deb, K., Pratap, A., Agarwal, S. et Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

[Di Gaspero et Urli, 2014] Di Gaspero, L. et Urli, T. (2014). A CP/LNS approach for multi-day homecare scheduling problems. *In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8457 LNCS, pages 1–15.

[Di Mascolo *et al.*, 2014] Di Mascolo, M., Espinouse, M.-l. et Erdem Ozkan, C. (2014). Synchronization between human resources in Home Health Care context. *In* Matta, A., Li, J., Sahin, E., Lanzarone, E. et Fowler, J., éditeurs : *Proceedings of the International Conference on Health Care Systems Engineering*, volume 61 de *Springer Proceedings in Mathematics & Statistics*, pages 73–86, Cham. Springer International Publishing.

[Drexl, 2012] Drexl, M. (2012). Synchronization in Vehicle Routing – A Survey of VRPs with Multiple Synchronization Constraints. *Transportation Science*, 46(3):297–316.

[Dror, 1994] Dror, M. (1994). Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW. *Operations Research*, 42(5):977–978.

[Eveborn *et al.*, 2006] Eveborn, P., Flisberg, P. et Rönnqvist, M. (2006). Laps Care-an operational system for staff planning of home care. *In European Journal of Operational Research*, volume 171, pages 962–976.

[Eveborn *et al.*, 2009] Eveborn, P., Ronnqvist, M., Einarsdottir, H., Eklund, M., Liden, K. et Almroth, M. (2009). Operations Research Improves Quality and Efficiency in Home Care. *Interfaces*, 39(1):18–34.

[Firat et Woeginger, 2011] Firat, M. et Woeginger, G. J. (2011). Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. *Operations Research Letters*, 39(1):32–35.

[Francis *et al.*, 2008] Francis, P., Smilovitz, K. et Tzur, M. (2008). The Period Vehicle Routing Problem and its extensions. *In The Vehicle Routing Problem : Latest Advances and New Challenges*, pages 73–102.

[Freeman, 2010] Freeman, H. (2010). Continuity of care and the patient experience. Rapport technique.

[Gehring et Homberger, 1999] Gehring, H. et Homberger, J. (1999). A Parallel Hybrid Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows. *In Proceedings of EUROGEN'99*, pages 57–64.

[Glover et Laguna, 2000] Glover, F. et Laguna, M. (2000). Fundamentals of Scatter Search and Path Relinking. (42743).

[Gunawan *et al.*, 2016] Gunawan, A., Lau, H. C. et Vansteenwegen, P. (2016). Orienteering Problem : A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255:–.

[Gutiérrez et Vidal, 2013] Gutiérrez, E. V. et Vidal, C. J. (2013). Home Health Care Logistics Management Problems : A critical review of models and methods. *Rev. Fac. Ingeniería. Universidad de Antioquia*, 68:160–175.

[Hasama, 1998] Hasama, T. K. (1998). A heuristic approach based on the string model to solve vehicle routing problem with backhauls. *In Proceedings of the 5th World Congress on Intelligent Transport Systems (ITS)*, Seoul. TOWARDS THE NEW HORIZON TOGETHER.

[Hiermann *et al.*, 2013] Hiermann, G., Prandtstetter, M., Rendl, A., Puchinger, J. et Raidl, G. R. (2013). Metaheuristics for solving a multimodal home-healthcare scheduling problem.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.*

[Ikegami et Uno, 2007] Ikegami, A. et Uno, A. (2007). Bounds for staff size in home help staff scheduling. *Journal of the Operations Research Society of Japan*, 50(4):563–575.

[Issaoui *et al.*, 2015] Issaoui, B., Zidi, I., Marcon, E. et Ghedira, K. (2015). New multiobjective approach for the home care service problem based on scheduling algorithms and variable neighborhood descent. *Electronic Notes in Discrete Mathematics*, 47:181–188.

[Jemai *et al.*, 2013] Jemai, J., Chaieb, M. et Mellouli, K. (2013). The Home Care Scheduling Problem : A modeling and solving issue. *In 2013 5th International Conference on Modeling, Simulation and Applied Optimization, ICMSAO 2013*, pages 1–6. Ieee.

[Kergosien *et al.*, 2009] Kergosien, Y., Lente, C. et Billaut, J.-C. (2009). Home health care problem : An extended multiple Traveling Salesman Problem. *In Proceedings of the 4th Multidisciplinary International Scheduling Conference : Theory and Applications (MISTA 2009), 10-12 Aug 2009, Dublin, Ireland*, pages 85–92.

[Kergosien *et al.*, 2014] Kergosien, Y., Ruiz, A. et Soriano, P. (2014). Solving a routing problem for medical test sample collection in home health care. *In* Matta, A., Li, J., Sahin, E. et Lanzarone, E., éditeurs : *Proceedings of the International Conference on Health Care Systems Engineering*, pages 29–46.

[Kindervater et Martin W.P. Salvesbergh, 1997] Kindervater, G. A. et Martin W.P. Salvesbergh (1997). Vehicle routing : handling edge exchanges. *In Local Search in Combinatorial optimization*, pages 337–360.

[Kovacs *et al.*, 2014] Kovacs, A. A., Golden, B. L., Hartl, R. F. et Parragh, S. N. (2014). The Generalized Consistent Vehicle Routing Problem. *Transportation Science*, XXXX(XX):1–21.

[Lacomme *et al.*, 2005] Lacomme, P., Prins, C. et Ramdane-Chérif, W. (2005). Evolutionary algorithms for periodic arc routing problems. *European Journal of Operational Research*, 165(2):535–553.

[Lanzarone *et al.*, 2012] Lanzarone, E., Matta, A. et Sahin, E. (2012). Operations management applied to home care services : The problem of assigning human resources to patients. *IEEE Transactions on Systems, Man, and Cybernetics Part A :Systems and Humans*, 42(6):1346–1363.

[Liu *et al.*, 2013] Liu, R., Xie, X., Augusto, V. et Rodriguez, C. (2013). Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. *European Journal of Operational Research*, 230(3):475–486.

[Liu *et al.*, 2014] Liu, R., Xie, X. et Garaix, T. (2014). Hybridization of tabu search with feasible and infeasible local searches for periodic home health care logistics. *Omega*, 47:17–32.

[Macdonald, 2010] Macdonald, T. (2010). *Metaheuristics for the consistent nurse scheduling and routing problem.* Master's thesis, Université de Nantes.

[Mankowska *et al.*, 2014] Mankowska, D. S., Meisel, F. et Bierwirth, C. (2014). The home health care routing and scheduling problem with interdependent services. *Health Care Management Science*, 17(1):15–30.

[Masson *et al.*, 2013] Masson, R., Lehuédé, F. et Péton, O. (2013). Efficient feasibility testing for request insertion in the pickup and delivery problem with transfers. *Operations Research Letters*, 41(3):211–215.

[Maya Duque *et al.*, 2015] Maya Duque, P. A., Castro, M., Sörensen, K. et Goos, P. (2015). Home care service planning. the case of Landelijke Thuiszorg. *European Journal of Operational Research*, 243(1):292–301.

[Moscato, 1989] Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts : Towards memetic algorithms. *Caltech Concurrent Computation Program, C3P Report*, 826:1989.

[Nagata et Bräysy, 2009] Nagata, Y. et Bräysy, O. (2009). Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. *Networks*, 54(4):205–215.

[Nagata *et al.*, 2010] Nagata, Y., Bräysy, O. et Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows.

[Nickel *et al.*, 2012] Nickel, S., Schröder, M. et Steeg, J. (2012). Mid-term and short-term planning support for home health care services. *European Journal of Operational Research*, 219(3):574–587.

[Pillac *et al.*, 2012] Pillac, V., Guéret, C. et Medaglia, a. L. (2012). A parallel matheuristic for the technician routing and scheduling problem. *Optimization Letters*, 7(7):1525–1535.

[Prins, 2004] Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002.

[Prins *et al.*, 2014] Prins, C., Lacomme, P. et Prodhon, C. (2014). Order-first split-second methods for vehicle routing problems : A review. *Transportation Research Part C : Emerging Technologies*, 40:179–200.

[Puljić et Manger, 2013] Puljić, K. et Manger, R. (2013). Comparison of eight evolutionary crossover operators for the vehicle routing problem. *Mathematical Communications*, 18(2):359–375.

[Rasmussen *et al.*, 2012] Rasmussen, M. S., Justesen, T., Dohn, A. et Larsen, J. (2012). The Home Care Crew Scheduling Problem : Preference-based visit clustering and temporal dependencies. *European Journal of Operational Research*, 219(3):598–610.

[Redjem, 2013] Redjem, R. (2013). *Aide à la décision pour la planififcation des activités et des ressources humaines en hospitalisation à domicile.* Thèse de doctorat, Université Jean Monnet de Saint Etienne.

[Rendl, 2011] Rendl, A. (2011). Multimodal home healthcare scheduling using a novel CP-VND-DP approach. *Cpaior 2011*, (1):1–3.

[Rendl *et al.*, 2012] Rendl, A., Prandtstetter, M., Hiermann, G., Puchinger, J. et Raidl, G. (2012). Hybrid heuristics for multimodal homecare scheduling. *In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7298 LNCS, pages 339–355, Nantes.

[Rest et Hirsch, 2013] Rest, K.-d. et Hirsch, P. (2013). Time-dependent travel times and multi-modal transport for daily home health care planning. *In TRISTAN VIII - The Eight Triennial Symposium on Transportation Analysis*, San Pedro de Atacama, Chile.

[Rest et Hirsch, 2015] Rest, K.-d. et Hirsch, P. (2015). Daily scheduling of home health care services using time-dependent public transport. *Flexible Services and Manufacturing Journal*, pages 1–31.

[Rest *et al.*, 2012] Rest, K.-D., Trautsamwieser, A. et Hirsch, P. (2012). Trends and risks in home health care. *Journal of Humanitarian Logistics and Supply Chain Management*, 2(1):34–53.

[Ronald, 1998] Ronald, S. (1998). More distance functions for order-based encodings. *In IEEE Conference on Evo- lutionary Computation*, pages 558–563.

[Russell, 1995] Russell, R. a. (1995). Hybrid Heuristics for the Vehicle Routing Problem with Time Windows.

[Savelsbergh, 1992] Savelsbergh, M. W. P. (1992). The Vehicle Routing Problem with Time Windows : Minimizing Route Duration. *ORSA Journal on Computing*, 4(2):146–154.

[Solomon, 1987] Solomon, M. M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints.

[Sörensen, 2007] Sörensen, K. (2007). Distance measures based on the edit distance for permutation-type representations. *Journal of Heuristics*, 13(1):35–47.

[Sörensen et Sevaux, 2006] Sörensen, K. et Sevaux, M. (2006). MAPM : memetic algorithms with population management. *Computers & Operations Research*, 33(5):1214–1225.

[Steeg et Schröder, 2008] Steeg, J. et Schröder, M. (2008). A Hybrid Approach to Solve the Periodic Home Health Care Problem. *Operations Research Proceedings 2007 Selected Papers of the Annual International Conference of the German Operations Research Society (GOR) Saarbrücken, September 5-7, 2007*, 2007:297–302.

[Thomsen, 2006] Thomsen, K. (2006). Optimization on Home Care. pages 1–182.

[Toth et Vigo, 2001] Toth, P. et Vigo, D., éditeurs (2001). *The vehicle routing problem.* Siam Monographs on Discrete Mathematics and Applications.

[Trautsamwieser *et al.*, 2011] Trautsamwieser, A., Gronalt, M. et Hirsch, P. (2011). Securing home health care in times of natural disasters. *OR Spectrum*, 33(3):787–813.

[Trautsamwieser et Hirsch, 2011] Trautsamwieser, A. et Hirsch, P. (2011). Optimization of daily scheduling for home health care services. *Journal of Applied Operational Research*, 3(3):124–136.

[Trautsamwieser et Hirsch, 2014] Trautsamwieser, A. et Hirsch, P. (2014). A branch-price-and-cut approach for solving the medium-term home health care planning problem. *Networks*, 64(3):143–159.

[Vidal *et al.*, 2013] Vidal, T., Crainic, T. G., Gendreau, M. et Prins, C. (2013). Heuristics for multi-attribute vehicle routing problems : A survey and synthesis. *European Journal of Operational Research*, 231(1):1–21.

[Vidal *et al.*, 2014] Vidal, T., Crainic, T. G., Gendreau, M. et Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234(3):658–673.

[Vidal *et al.*, 2015] Vidal, T., Crainic, T. G., Gendreau, M. et Prins, C. (2015). Timing problems and algorithms : Time decisions for sequences of activities. *Networks*, 65(2):102–128.

[Weigel et Cao, 1999] Weigel, D. et Cao, B. (1999). Applying GIS and OR techniques to solve Sears technician-dispatching and home-delivery problems. *Interfaces*, 29(1):112–130.

Résumé :

Le soin à domicile est un secteur en plein essor ces dernières années. Cela est dû au vieillissement de la population, à la volonté de réduire les coûts hospitaliers et d'assurer le bien-être du patient en le gardant dans son cadre familial tout en maintenant la qualité des soins. L'organisation de ces soins nécessite une prise de décisions aux niveaux stratégique, tactique et opérationnel. Cette thèse s'articule autour de l'étude de problèmes apparaissant uniquement au niveau opérationnel. Ces problèmes traitent de la planification des tournées du personnel de soins à domicile. La première étape de cette étude a consisté à faire une revue de la littérature. De nombreux modèles mathématiques ont été formulés dans la littérature. Cependant, ces modèles étaient dédiés à une structure de soins à domicile spécifique et pouvaient être difficilement transposés. Nous proposons ici une approche générique tant du point de vue de la modélisation que des méthodes de résolutions. À cet effet, nous avons identifié les caractéristiques fréquemment rencontrées dans la littérature à travers cette revue de la littérature. Un modèle générique a été proposé prenant en compte la plupart des caractéristiques. Ce modèle générique constitue le socle pour la construction de méthodes de résolution. Deux méthodes de résolution ont été conçues. La première méthode est une méthode par décomposition et la deuxième méthode est un algorithme hybride génétique avec gestion de la population. Ces deux méthodes utilisent des représentations d'une solution issues de la littérature et adaptées aux caractéristiques du problème. Des expérimentations numériques ont été réalisées dans le but d'évaluer les méthodes proposées et de se comparer à la littérature.

Mots clés :

Recherche Opérationnelle, Santé, Tournées de véhicules, Soins à domicile, Méta-heuristique