POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Développement d'une méthode de réoptimisation progressive appliquée à un problème de planification de soins à domicile

AURIANE PETER–HEMON
Département de mathématiques et de génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Mathématiques

Septembre 2025

© Auriane Peter–Hemon, 2025.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Développement d'une méthode de réoptimisation progressive appliquée à un problème de planification de soins à domicile

présenté par **Auriane PETER–HEMON**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Michel GENDREAU, président
Louis-Martin ROUSSEAU, membre et directeur de recherche
Nadia LAHRICHI, membre et codirectrice de recherche
Antoine LEGRAIN, membre

REMERCIEMENTS

Je tiens en premier lieu à remercier mon directeur de recherche, M. Louis-Martin Rousseau, ainsi que ma codirectrice, Mme. Nadia Lahrichi, pour leur soutien et leurs conseils tout au long de ce projet.

Je remercie également les autres étudiant.es du laboratoire, dont la présence et les échanges ont été sources de motivation.

Je souhaite également remercier mes ami.es en dehors du laboratoire, qui ont rendu cette expérience à Montréal si enrichissante. En particulier, je remercie mes colocataires, Côme, Emma et Oriane, pour leur présence durant ces derniers mois.

Merci à Louise d'avoir été un soutien constant tout au long de ces deux ans.

Enfin, merci à ma famille de m'avoir toujours soutenue, pour cette maîtrise comme pour tout le reste.

RÉSUMÉ

La planification dans le domaine médical constitue un enjeu majeur pour assurer la qualité des soins tout en optimisant les ressources disponibles. De nombreuses méthodes opérationnelles permettent d'obtenir des plannings optimaux ou très performants. Cependant, dans la pratique, il est rarement possible d'utiliser directement le planning optimal, car il peut différer fortement du calendrier en vigueur. Ces différences pourraient entraîner de trop fortes contraintes logistiques ou pourraient nuire à la continuité des soins et à la satisfaction des patients et du personnel soignant.

Nous proposons donc dans ce travail une méthode générale pour planifier la transition progressive entre une planification existante et une planification optimale, et ce en limitant à chaque étape le nombre de modifications apportées. Cette approche vise à concilier efficacité et stabilité de la planification.

Dans un premier temps, nous formulons ce problème sous la forme d'un problème linéaire en nombres entiers (PLNE) afin d'identifier, à chaque itération, la meilleure modification possible dans la limite du nombre de changements autorisés. Pour nous adapter aux contextes dynamiques où les situations évoluent en permanence (nouveaux patients, imprévus...), nous développons ensuite une méthode de recherche arborescente qui permet d'anticiper plus efficacement l'impact des décisions sur l'avenir et d'adapter la planification de manière plus robuste.

Pour valider notre approche, nous appliquons notre méthode à un problème de planification de soins à domicile. Nous résolvons ce problème en deux phases : d'abord en résolvant le problème d'affectation (statique puis dynamique), puis en effectuant le routage pour chaque jour et chaque soignant. Ces phases nous permettent de tester nos méthodes sur des cas de complexité croissante.

Les résultats expérimentaux montrent la capacité de notre méthode à réparer efficacement la planification initiale. Nous proposons également des recommandations pratiques à destination des décideurs.

ABSTRACT

Effective planning in healthcare is crucial to ensuring high-quality care while making the best use of available resources. Although many operational methods exist to generate optimal or near-optimal schedules, directly implementing these schedules is often impractical, as they can differ substantially from the current ones. Such differences may cause significant logistical challenges and negatively affect continuity of care, as well as the satisfaction of both patients and healthcare staff.

In this work, we propose a general approach to manage the gradual transition from an existing schedule to an improved, optimal one, by limiting the number of changes introduced at each step. This strategy seeks to strike a balance between improving efficiency and maintaining schedule stability.

Our approach begins by formulating the problem as an integer linear program (ILP), allowing us to identify, at each iteration, the best possible modifications within a predefined limit on the number of changes. To better accommodate dynamic environments, where new patients arrive and unexpected events occur, we develop a tree search method that anticipates the future impact of decisions and adapts the schedule more effectively and robustly.

We validate our method on a home healthcare scheduling problem. The problem is tackled in two stages: first, by solving the assignment problem (both in static and dynamic settings), and then by incorporating routing for each day and caregiver. This stepwise progression enables us to evaluate our approach on problems of increasing complexity.

Experimental results demonstrate that our method successfully improves the initial schedule while respecting operational constraints. We also offer practical recommendations to assist decision-makers in implementing these strategies.

TABLE DES MATIÈRES

REMERCIEMENTS iii

RÉSUMÉ iv

ABSTRACT v

LISTE DES TABLEAUX viii

LISTE DES FIGURES ix

LISTE DES SIGLES ET ABRÉVIATIONS x

CHAPITRE 1 INTRODUCTION 1

CHAPITRE 2 REVUE DE LITTÉRATURE 4
- 2.1 Méthodes de replanification et notion de stabilité 4
- 2.2 Le problème de soins à domicile 6
- 2.3 La planification dynamique dans un contexte de soins à domicile 8
- 2.4 Conclusion 9

CHAPITRE 3 MÉTHODOLOGIE 10
- 3.1 Méthode initiale 10
  - 3.1.1 Formulation sous forme d'un PLNE 10
  - 3.1.2 Application à un cas statique 12
  - 3.1.3 Application à un cas dynamique 13
  - 3.1.4 Limites 15
- 3.2 Méthode arborescente 15
  - 3.2.1 Lien avec le Monte-Carlo Tree Search 16
  - 3.2.2 Principe de la méthode 17

CHAPITRE 4 MODÉLISATIONS DU PROBLÈME DE PLANIFICATION DES SOINS À DOMICILE 20
- 4.1 Problème d'affectation 21
- 4.2 Problème de routing 23
- 4.3 Planification initiale 25
  - 4.3.1 Cas statique 25
  - 4.3.2 Cas dynamique 26

4.4 Application de la méthode arborescente 26

CHAPITRE 5 EXPÉRIMENTATIONS ET RÉSULTATS 27
5.1 Génération des instances 27
5.1.1 Création des soignants 27
5.1.2 Création des patients 28
5.1.3 Organisations en instances 29
5.2 Résultats pour le problème d'affectation statique 30
5.3 Résultats et discussion pour le problème d'affectation dynamique 34
5.3.1 Méthode PLNE 34
5.3.2 Méthode arborescente 36
5.3.3 Décisions opérationnelles 42
5.4 Résultats et discussion pour le problème complet 44
5.4.1 Décisions opérationnelles 49

CHAPITRE 6 CONCLUSION 50

RÉFÉRENCES 52

LISTE DES TABLEAUX

Tableau 3.1 Paramètres du modèle de réparation 11
Tableau 4.1 Paramètres du modèle d'affectation 21
Tableau 4.2 Paramètres du modèle de routing 24
Tableau 5.1 Caractéristiques des instances dynamiques 30
Tableau 5.2 Écart initial pour chaque instance 31
Tableau 5.3 Matrice de recommandation pour le choix du paramètre *max_changes* 43

LISTE DES FIGURES

Figure 5.1 Nombre d'étapes nécessaires pour atteindre l'optimal en fonction du nombre de changements autorisés 32
Figure 5.2 Nombre de changements effectués à chaque étape pour l'instance 1 . . 33
Figure 5.3 Évolution des affectations des patients selon max_changes pour l'instance 2 34
Figure 5.4 Écart relatif à l'optimal au cours du temps selon différents niveaux de *max_changes* (méthode myope) 35
Figure 5.5 Évolution de l'écart relatif entre la solution courante et la solution optimale pour les instances 1 à 4 (méthode arborescente) 37
Figure 5.6 Évolution de l'écart relatif entre la solution courante et la solution optimale pour les instances 5 et 6 (méthode arborescente) 38
Figure 5.7 Évolution de l'écart relatif entre la solution courante et la solution optimale pour les instances 7 et 8 (méthode arborescente) 39
Figure 5.8 Comparaison du nombre de patients modifiés à deux semaines de la fin de leur traitement en fonction de la méthode utilisée pour l'instance 4 40
Figure 5.9 Nombre de changements effectués à chaque étape pour l'instance 3 (méthode arborescente) 41
Figure 5.10 Évolution du nombre de patients refusés en fonction du nombre de changements autorisés (méthode arborescente) 42
Figure 5.11 Évolution de l'écart combiné en fonction du paramètre max_changes 46
Figure 5.12 Évolution du pourcentage de patients mal affectés en fonction du paramètre max_changes 48

LISTE DES SIGLES ET ABRÉVIATIONS

PLNE	Programmation Linéaire en Nombre Entiers
TSP	Traveling Salesman Problem
VRP	Vehicle Routing Problem
PVRP	Periodic Vehicle Routing Problem
CVRP	Capacitated Vehicle Routing Problem
MCTS	Monte-Carlo Tree Search

CHAPITRE 1 INTRODUCTION

Dans le domaine médical, la planification constitue un enjeu central pour la qualité des soins procurés. Partout dans le monde, les systèmes de santé doivent répondre à des défis d'une ampleur et d'une complexité croissantes. Dans un contexte marqué par des contraintes budgétaires, l'optimisation de la planification est devenue un impératif. Dans de nombreux domaines, tels que l'organisation des soins à domicile, la gestion des blocs opératoires ou la confection des horaires du personnel infirmier, la planification influence en effet directement la qualité des soins fournis et l'efficacité des équipes. Le vieillissement démographique accentue cette pression. Selon les projections de l'Organisation Mondiale de la Santé (OMS), la proportion de personnes âgées de plus de 60 ans passera de 12% en 2015 à 22% en 2050, avec un nombre croissant de patients atteints de pathologies chroniques nécessitant un suivi régulier. [1] Cette évolution entraîne une demande accrue de services de soin. Dans ce contexte, les soins à domicile occupent une place croissante dans les politiques de santé. En Europe, ils représentent déjà entre 1% et 5% du budget total de la santé public [2]. Ces soins ont plusieurs avantages : ils permettent de réduire le nombre de patients admis à l'hôpital, de réduire la durée de séjour à l'hôpital pour les patients hospitalisés. De plus, les patients concernés ont le confort de rester chez eux, ce qui contribue à leur bien-être. Compte tenu des tendances démographiques, les soins à domicile devraient continuer à gagner en importance dans les prochaines années.

Compte tenu de l'importance de la planification dans le secteur de la santé, les gestionnaires et les chercheurs se sont intéressés depuis plusieurs décennies à la question de l'optimisation de la planification dans le domaine médical. Des approches variées ont été proposées. La Programmation Linéaire en Nombres Entiers (PLNE) a ainsi été largement utilisée, car elle permet de générer des plannings optimaux sous un ensemble de contraintes. Les problèmes liés au milieu médical étant souvent des problèmes de grande taille, de nombreuses heuristiques et métaheuristiques ont été développées pour permettre d'obtenir des solutions satisfaisantes en un temps raisonnable aux problèmes rencontrés [3, 4]. Récemment, des méthodes utilisant l'apprentissage machine et l'apprentissage par renforcement ont également été utilisées pour améliorer la planification [5, 6].

Ces méthodes ont montré leur efficacité dans de nombreux contextes. Cependant, leur application pratique peut rencontrer des obstacles. En particulier, elles supposent généralement

que la planification peut être entièrement recalculée. Or, dans la réalité, les gestionnaires disposent déjà d'un planning opérationnel, qui ne peut être remplacé instantanément par une solution calculée avec un outil opérationnel. Cela est dû entre autres à des contraintes logistiques, mais aussi à un autre facteur : la continuité des soins. En effet, il est généralement souhaité qu'un patient conserve le même soignant pour toute la durée de son traitement. Cela permet d'améliorer la qualité du suivi et la relation de confiance entre patient et soignant [7,8]. Pour améliorer la planification, il faudrait donc pouvoir modifier progressivement la planification actuelle pour se rapprocher de la solution optimale.

Cela soulève la question suivante : comment passer progressivement d'une planification existante à une planification optimale, en minimisant les perturbations et en respectant les contraintes médicales et logistiques ?

Plus précisément, il s'agit de définir un processus de transition permettant de modifier, à chaque étape, seulement un nombre limité d'éléments du planning, tout en s'approchant de la solution optimale visée. Ce problème est particulièrement pertinent dans les contextes dynamiques, où la planification doit évoluer régulièrement (imprévus, patients dont le traitement se termine, nouveaux patients...). Cette problématique est différente de celles habituellement étudiées dans la littérature : nous voulons faire évoluer une planification courante vers une cible optimale prédéterminée et connue.

L'objectif principal de cette maîtrise a été de concevoir et d'évaluer une méthode permettant la transition progressive entre un planning existant et un planning optimal. Cette méthode doit respecter les contraintes suivantes :

- limiter les changements à chaque itération, pour assurer la continuité des soins et limiter les perturbations
- assurer à chaque étape la faisabilité de la solution (respect des contraintes)
- s'adapter à des contextes statiques et dynamiques

Pour atteindre cet objectif, nous proposons une approche en deux étapes. Dans un premier temps, nous formulons notre problème comme un problème de PLNE. Ce problème nous permettra d'identifier à chaque itération la meilleure modification possible dans la limite du nombre maximal de changements autorisés. Dans un second temps, nous intégrerons une méthode de recherche arborescente. Cette méthode est particulièrement adaptée aux cas dynamiques car elle permet de mieux anticiper l'impact de nos choix sur le futur.

Nous avons choisi d'appliquer notre méthode à un problème de soins à domicile. En effet,

il s'agit d'un problème complexe qui combine des contraintes logistiques strictes. De plus, la demande de continuité des soins et les contraintes logistiques y sont particulièrement présentes, renforçant le besoin d'une réoptimisation incrémentale. Enfin, ce problème peut assez facilement être simplifié, ce qui nous permettra de tester notre méthode sur des cas de difficulté croissante.

En premier lieu, nous nous sommes intéressés à un problème statique, comportant uniquement de la planification, et où la liste des patients traités est fixe. Ensuite, nous avons étudié un cas dynamique, en continuant à nous intéresser uniquement à la partie planification. Dans un problème statique, l'ensemble des patients et des contraintes est fixé au départ et ne varie pas avec le temps. À l'inverse, dans un problème dynamique, des événements (arrivée ou départs de patients) surviennent et nécessitent une mise à jour régulière du planning. Enfin, nous avons ajouté la dimension de *routing* pour nous rapprocher du problème dans son entièreté.

Ce mémoire débutera par une revue de littérature au chapitre 2. Dans le chapitre 3, nous présenterons notre méthode ainsi que les cas d'application auxquels nous allons nous intéresser. Le chapitre 4 sera consacré à la phase d'expérimentation, aux résultats obtenus et à leur analyse. Enfin, le chapitre 5 fera la synthèse de ce travail.

CHAPITRE 2 REVUE DE LITTÉRATURE

Dans ce projet, nous développons une méthode de replanification progressive qui permette de passer d'une planification courante utilisée par un gestionnaire à une planification optimale obtenue par une méthode d'optimisation. Nous appliquons ensuite cette méthode à un problème de soins à domicile. Notre objectif n'est pas de créer une nouvelle planification, mais de faire évoluer celle existante vers une cible optimale prédéterminée. En cela, notre objectif diffère de ce qui est habituellement étudié dans la littérature

Dans cette revue de littérature, nous nous intéresserons donc à trois points : les méthodes de replanification existantes, les méthodes de résolution de problèmes de soins à domicile et les méthodes mélangeant ces deux domaines.

2.1 Méthodes de replanification et notion de stabilité

La replanification, ou *rescheduling*, est une pratique qui a d'abord été théorisée dans le domaine de la planification de la production industrielle. En effet, dans l'industrie, il est courant que les plannings ne puissent être effectués comme prévus, et ce pour diverses raisons. Ces raisons peuvent être liées aux ressources (casse d'une machine, absence d'un opérateur, retard de livraison pour un matériau) ou aux tâches elles-même (annulation, date limite modifiée, changement de priorités) [9]. Dans ce cas, il est nécessaire de mettre à jour le planning existant pour réagir à ces changements. Vieira et al. [10] sont les premiers à discuter d'un cadre théorique pour la replanification. L'étude souligne que la replanification s'inscrit dans un cadre prédictif-réactif : un planning initial est construit, puis il est ajusté lorsqu'un imprévu survient. L'étude menée par Ouelhadj et al. [9] précise que le problème de replanification est associé à deux décisions opérationnelles : quand replanifier, et avec quelle méthode. Concernant le moment de replanifier, il existe trois possibilités : replanification périodique, replanification lorsqu'un imprévu survient, ou méthode hybride. Concernant la méthode de replanification, deux stratégies sont possibles : réparer le planning perturbé, ou replanifier complètement la production. Ce choix est déterminant, car il introduit la question de la stabilité d'un planning, c'est-à-dire le nombre de modifications apportées au calendrier initial lors de sa révision. Une autre notion associée est la robustesse, qui mesure dans quelle mesure ces modifications dégradent les performances du système. Les deux concepts sont liés mais distincts : un planning peut rester performant mais très différent de l'initial (robuste sans être stable), ou au contraire peu modifié mais avec une forte baisse de performance (stable sans être robuste). Cette notion de stabilité fait écho au problème résolu dans ce mémoire :

en effet, nous souhaitons réoptimiser progressivement une planification actuelle en limitant le nombre de changements effectués à chaque étape.

Lorsque la méthode de replanification choisie est la réparation du planning perturbé, il existe trois méthodes courantes [11]. La première est le "décalage par la droite" (*Right-Shift Rescheduling (RSR)*). Cette méthode consiste à décaler vers la droite les heures de début de toutes les opérations restantes du temps nécessaire à la gestion de la perturbation. Cette méthode est facile à mettre en place et assure la stabilité du calendrier. Cependant, elle risque de dégrader fortement les performances du système lors de perturbations prolongées. La seconde méthode est la "génération partielle" (*Partial Generation (PG)*) et c'est la méthode qui cherche le plus à garantir la stabilité du calendrier. Elle ne modifie que les opérations affectées par la perturbation. Cependant, son coût de calcul est plus élevé que la méthode précédente. Enfin, la méthode de "génération complète" (*Complete Generation (CG)*) replanifie toutes les tâches qui n'avaient pas été exécutées avant la perturbation.

Plusieurs travaux ont intégré la notion de stabilité dans leurs approches de replanification. Cependant, il existe un conflit entre stabilité et efficacité [12]. Pour pallier à cela, des fonctions objectif intégrant ces deux critères ont été utilisées avec succès [13]. Leus et al. [14] proposent un modèle de replanification pour une machine en intégrant la stabilité dans la fonction objectif. Ils développent ensuite un algorithme de type *branch-and-bound* pour le résoudre. Wallace et al. [15] proposent une approche de programmation par contraintes combinée avec une méthode de recherche locale pour trouver un planning modifié qui minimise la perturbation par rapport au planning initial. Plus récemment, Valledor et al. [16] proposent une métaheurisitque basée sur un algorithme génétique dans le but de résoudre un problème de replanification pour plusieurs machines. Leur modèle cherche à optimiser trois fonctions objectif : le délai d'exécution, le retard total et la stabilité.

Si le rescheduling a été particulièrement étudié dans le domaine de la production industrielle, il a également été abordé dans d'autres domaines tels que le transport ou la logistique. Cependant, la majorité de ces travaux n'incluent pas de notion pouvant se rapprocher de la notion de stabilité. Une exception est Sato et al [17] qui développent une approche de replanification pour le transport ferroviaire fondée sur la formulation PLNE. Leur approche cherche à minimiser l'inconfort supplémentaire subi par les passagers, pris en compte via les temps de trajet, d'attente et de correspondance, ce qui peut s'apparenter à la recherche d'une nouvelle planification la plus proche possible de l'ancienne planification. En revanche,

plusieurs études incluent une notion de robustesse dans leurs travaux. C'est par exemple le cas de Soltani et al. [18] qui proposent une méthode de replanification pour une entreprise de transport maritime. L'intégration de la notion de robustesse lors de la replanification permet à l'entreprise de réduire le coût des imprévus lors de ses prochains voyages.

Ainsi, la replanification est un concept qui a été largement étudié dans le domaine de la production industrielle. Est notamment théorisée la notion de stabilité, qui correspond à la minimisation de la différence entre la planification perturbée et la nouvelle planification, et qui a été intégrée à plusieurs travaux comme mesure de performance. Si la replanification existe également dans d'autres domaines comme le transport ou la logistique, la notion de stabilité y est peu étudiée. Dans tous les travaux abordant les replanifications, ces dernières ont lieu suite à une perturbation du planning initial. Suite à cet imprévu, on cherche une nouvelle planification pour se rapprocher d'un optimal inconnu. Notre problématique est différente : notre objectif est de faire évoluer une planification courante vers une cible optimale prédéterminée et connue.

2.2 Le problème de soins à domicile

Le problème de planification des soins à domicile peut se formuler ainsi : étant donné un ensemble de patients et un ensemble de soignants, il s'agit de déterminer la séquence optimale des visites. Plus précisément, sur un horizon temporel donné, il faut décider quel soignant rend visite à quels patients, quels jours et dans quel ordre.

Ce problème est une variation du problème de tournées de véhicules (Vehicle Routing Problem (VRP)) avec des contraintes additionnelles liées au contexte médical. Ce problème étant NP-difficile, le problème de planification de soins à domicile l'est également [19]. Ces contraintes variant fortement selon les contextes, il n'existe pas de définition unique du problème de planification des soins à domicile dans la littérature.

Le problème peut être modélisé sur différentes échelles de temps. La plupart des travaux considèrent que le problème doit être résolu pour un jour donné [20–24]. D'autres travaux étudient le problème sur une période de plusieurs jours, par exemple sur une semaine. Dans ce cas, de nouvelles contraintes apparaissent.

Les contraintes les plus fréquentes liées aux patients sont liées à leur plan de soin. Elles concernent la durée de traitement des patients, mais aussi le nombre de visites nécessaires sur une période donnée. Le problème est alors une variation du problème de tournées de

véhicules périodique, Periodic Vehicle Routing Problem (PVRP), dans lequel chaque patient doit être visité une ou plusieurs fois dans un horizon de plusieurs périodes [25]. Elles peuvent également intégrer les disponibilités des patients, par exemple sous forme d'un ensemble de jours disponibles par semaine. Cependant, un grand nombre de travaux font l'hypothèse que les patients sont disponibles toute la semaine. Les disponibilitées peuvent également être exprimées sous forme de fenêtre de temps [26].

Les contraintes peuvent aussi concerner l'affectation des patients aux soignants. Les soins dispensés à un patient peuvent requérir un certain niveau de compétence qui n'est pas partagé par tous les soignants, limitant ainsi les possibilités d'affectation pour ce patient [22–24, 27].

La continuité des soins peut également être exprimée sous forme de contrainte souple ou dure. Lorsque c'est une contrainte dure, le problème se rapproche d'une variante du VRP nommée *consistent vehicle routing problem* [26, 28]. Couplée avec la contrainte de périodicité des visites, on se rapproche d'un problème qui a été nommé *periodic vehicle routing problem with driver consistency* [29].

Les contraintes liées aux soignants peuvent concerner leur contrat de travail, en limitant par exemple le nombre d'heures travaillées sur la période.

Différents objectifs peuvent être poursuivis. Le problème étant une extension du VRP, la majorité des travaux considèrent des objectifs liés au transport, comme la minimisation de la distance totale parcourue par les soignants ou la minimisation du temps de trajet total des soignants. On retrouve également des objectifs liés à la minimisation des coûts, comme la minimisation des temps d'attente. La majorité des travaux ont recours à des fonctions objectifs qui incluent plusieurs de ces éléments avec une certaine pondération [26, 30].

Au vu de la complexité du problème de planification des soins à domicile, en particulier pour des problèmes de grande taille, des méthodes de résolution approchées ont donc été développées. Parmi les heuristiques, plusieurs auteurs ont proposé des approches en deux phases [20, 21, 31, 32]. De nombreuses métaheuristiques ont également été explorées, ces dernières étant couramment utilisées pour résoudre le problème de tournées de véhicules. On trouve notamment des approches basées sur les algorithmes génétiques [22, 27, 33], le recuit simulé [23, 24], la recherche taboue [34, 35], ou encore l'optimisation par colonies de fourmis [21, 36, 37]. Pour un panorama détaillé des méthodes et variantes employées, on pourra se référer à la revue récente proposée par Fu et al. [38].

2.3 La planification dynamique dans un contexte de soins à domicile

L'immense majorité des travaux consacrés à la planification des soins à domicile (ou de manière plus générale au VRP) considèrent que la planification est faite à partir de zéro, sans prendre en compte les plannings existants [31]. Seuls quelques travaux abordent le problème de planification dynamique dans le contexte des soins à domicile. Cependant, ce problème de planification dynamique est pertinent : dans de nombreux cas, les plannings créés sont valables jusqu'à ce qu'un changement survienne (par exemple l'arrivée ou le départ d'un patient). Il est alors intéressant de modifier la planification actuelle sans tout recommencer.

Parmi les travaux abordant le *rescheduling* du problème de tournées de véhicules, seuls quelques uns y intègrent la notion de stabilité.

Gomes et al. [31] s'intéressent à un problème de planification dynamique de services de soins à domicile. Le problème est formulé comme un PLNE. Trois fonctions objectifs sont modélisées, dont une minimisant l'écart total de début de visite induite par la replanification. Cet objectif permet de limiter la variation du planning par rapport au calendrier initial. De plus, les patients sont séparés en deux groupes (flexibles et non flexibles) auxquels sont associés une déviation maximale autorisée de l'heure de début de visite. Cela permet de contrôler davantage que les deux plannings resteront proches. Ce problème est ensuite résolu en le divisant en deux sous-problèmes. Le problème est d'abord résolu en regroupant les soignants sous forme d'équipes. Par la suite, les patients sont assignés à un unique patient membre de l'équipe à laquelle ils ont été assignés dans la première phase.

Martinez et al. [39] proposent une méthode de replanification des tournées lors d'un changement dans le pool de patients ou de personnel soignant. La continuité est prise en compte de deux manières via la continuité des soins, mais aussi via une continuité temporelle qui garantit que les horaires des visites soient les mêmes entre la planification d'origine et la nouvelle planification. Les autrices souhaitent ainsi assurer que la nouvelle planification ne diffère pas trop du calendrier initial. Le problème de réoptimisation est ensuite découpé en deux phases : dans un premier temps, toutes les tournées quotidiennes respectant les contraintes légales et de continuité sont générées. Dans un second temps, le meilleur ensemble de tournées est sélectionné.

2.4 Conclusion

Nous avons vu que la replanification est une problématique largement étudiée dans le domaine de la production industrielle, où la notion de stabilité est régulièrement intégrée comme un critère de performance. Cette stabilité permet d'assurer un équilibre entre la qualité du planning et la continuité des opérations. Dans d'autres secteurs où le *rescheduling* est exploré, comme le transport ou la logistique, la prise en compte explicite de la stabilité est encore relativement rare, même si des notions proches comme la robustesse, sont parfois étudiées.

Nous avons ensuite étudié le problème de la planification des soins à domicile. Ce problème, souvent modélisé comme une variante du problème de tournées de véhicules, a été largement étudié dans la littérature. Il est rendu particulièrement difficile à résoudre par les nombreuses contraintes spécifiques au domaine médical qui lui sont associées. De nombreuses méthodes, souvent approchées, ont été développées pour le résoudre.

Cependant, l'étude de la replanification pour ce problème reste limitée, et presque aucun des travaux n'intègre la notion de stabilité.

Que ce soit dans le contexte de production industrielle ou dans un autre contexte, la plupart des approches se concentrent sur la replanification après un imprévu, en cherchant la meilleure replanification possible.

Or, notre problématique est différente : nous voulons développer une méthode de replanification progressive permettant de passer d'une planification courante à une planification optimale connue, en limitant les changements à chaque étape afin de préserver la stabilité. Ce besoin spécifique n'est, à notre connaissance, pas étudié, ni dans le domaine des soins à domicile, ni plus largement dans les autres secteurs étudiés.

CHAPITRE 3 MÉTHODOLOGIE

Dans ce chapitre, nous présentons la méthodologie de réoptimisation développée pour améliorer progressivement une planification existante. Nous proposons d'abord une formulation générique sous forme de PLNE, que l'on applique à un cadre statique puis dynamique. Pour dépasser les limites de cette première méthode, nous présentons une deuxième méthode fondée sur une recherche arborescente guidée par simulation qui permet de prendre en compte les effets à moyen terme de notre replanification.

3.1 Méthode initiale

Comme illustré précédemment, dans un contexte de planification opérationnelle, il n'est la plupart du temps pas envisageable de remplacer brutalement la planification actuelle par la solution optimale.

L'objectif général de notre travail est de développer une méthode de réparation ou de replanification capable d'améliorer progressivement une planification existante, en la rapprochant de l'optimum, tout en limitant les changements appliqués à chaque étape.

Nous cherchons à formuler cette méthode de manière générale, afin qu'elle puisse être appliquée à de nombreux problèmes de planification. L'idée est la suivante : étant donnée une planification courante X, ainsi qu'une planification optimale de référence X^{opti}, nous voulons déterminer une nouvelle planification X^{new} qui :

- reste réalisable en respectant les contraintes du problème ;
- se rapproche au maximum de X^{opti} ;
- diffère de la planification courante X de manière contrôlée.

3.1.1 Formulation sous forme d'un PLNE

Nous modélisons ce problème sous la forme d'un PLNE. Pour que cette méthode soit applicable au plus grand nombre de problèmes possible, nous le décrivons de la manière la plus générale possible. Les données du problème sont réprésentées dans le tableau 3.1.

Tableau 3.1 Paramètres du modèle de réparation

Symbole	Description
Ensembles	
$\mathcal{F}$	Ensemble des solutions réalisables pour le problème considéré
Données	
X	Planification courante utilisée
X^{opti}	Planification optimale
$max_changes$	Nombre maximum de changements autorisés entre deux planifications
Variables de décision	
X^{new}	Nouvelle planification

L'objectif et les contraintes sont les suivants :

Objectif :

$$\min d(X^{opti}, X^{new}) \tag{3.1}$$

Contraintes :

$$X^{new} \in \mathcal{F} \tag{3.2}$$

$$d(X, X^{new}) \leq max_changes \tag{3.3}$$

L'équation (3.1) représente l'objectif de notre modèle : minimiser l'écart entre la nouvelle planification X^{new} et la planification optimale X^{opti}. Pour cela, on utilise une fonction de distance d. Dans notre implémentation, $d(X, Y)$ représentera le nombre d'affectations différentes entre les solutions X et Y. L'équation (3.2) assure que la solution X^{new} soit une solution réalisable du problème considéré. Enfin, l'équation (3.3) représente la contrainte de stabilité qui limite le nombre de changements autorisés. Le paramètre $max_changes$ représente le nombre de changements maximum autorisés pour la réparation. Il permet de contrôler le compromis entre qualité de solution et stabilité opérationnelle.

En appliquant itérativement le modèle, la planification courante X converge vers la planification optimale X^{opti}. Le nombre minimal de périodes k nécessaires pour atteindre l'optimum

est borné par

$$k \geq \left\lceil \frac{max_changes}{d(X, X^{opti})} \right\rceil$$

Ainsi, k représente la durée de transition entre la situation initiale et l'optimum, c'est-à-dire le nombre de périodes au cours desquelles des réparations successives doivent être effectuées avant de parvenir à la solution cible.

Cette formulation reste volontairement abstraite : l'ensemble $\mathcal{F}$, la fonction de distance d et la nature des solutions dépendent du problème considéré.

3.1.2 Application à un cas statique

Dans un cas statique, les données du problème de planification considéré ne changent pas. Nous disposons de la planification actuelle utilisée ainsi que de la solution optimale que l'on souhaite atteindre. Pour cela, nous appliquons le modèle PLNE de manière itérative, en rapprochant à chaque étape la solution courante de la solution optimale jusqu'à atteindre cette dernière.
En théorie, il n'est pas garanti que cette application itérative nous mène systématiquement à la solution optimale. En effet, il faut pour cela qu'il existe un chemin de solutions réalisables successives allant de X à X^{opti}. Il faut donc vérifier que c'est le cas avant d'appliquer cette méthode, au risque de se retrouver dans une boucle infinie. Pour éviter cela, on ajoute une condition de sortie de la boucle : après un certain nombre d'itérations, le processus s'arrête.

Le processus est décrit dans le pseudo-code suivant :

Algorithm 1: Réoptimisation statique

Input: X^{init} : planification initiale
X^{opti} : planification optimale
$max_changes$
$unmovable_duration$
max_steps

Output: nb_steps, solutions intermédiaires

```
nb_steps ← 0 ;
while X ≠ X^opti or nb_steps ≤ max_steps do
    X ← repair(X, X^opti, max_changes) ;
    nb_steps ← nb_steps + 1 ;
end
```

La fonction `repair(`X`, `X^{opti}`, `$max_changes$`)` résout le modèle PLNE décrit dans la section

3.1.1, en fixant X comme solution courante et X^{opti} comme référence. Elle produit une nouvelle solution réalisable X^{new} respectant la contrainte de stabilité.

3.1.3 Application à un cas dynamique

Dans un cadre dynamique, le problème change périodiquement. Par exemple, dans le cas de la planification des soins à domicile, cela correspond au départ des patients dont le traitement est terminé et à l'arrivée de nouveaux patients. Dans ce cadre, à chaque période t, la solution optimale évolue. Il est donc nécessaire de la recalculer à chaque étape. Pour éviter des ruptures brutales, nous cherchons à sélectionner, parmi les solutions optimales possibles, celle qui est la plus proche de la solution optimale précédente.

Le processus se déroule ainsi :

1. Mise à jour de la solution courante X
2. Calcul de la nouvelle solution optimale X^{opti}
3. Réparation progressive de la solution courante pour se rapprocher de la nouvelle solution optimale

Le pseudo-code de cette procédure est le suivant :

Algorithm 2: Réoptimisation dynamique avec méthode PLNE

Input: X^{init} : solution initiale
$X^{opti,init}$: solution optimale initiale
$max_changes$

Output: nb_steps, solutions intermédiaires

```
nb_steps ← 0 ;
X ← X^init ;
X^opti ← X^opti, init ;
while horizon non terminé do
    X ← update(X) ;
    X^opti ← get_new_optimal(X^opti) ;
    X ← repair(X, X^opti, max_changes) ;
    nb_steps ← nb_steps + 1 ;
end
```

Nous détaillons à présent les différentes parties.

Mise à jour de la solution courante : fonction `update(X)`

L'objectif de cette fonction est de garantir une dynamique de roulement réaliste tout en préservant la faisabilité.

Elle prend en entrée la solution courante X ainsi que les données modifiées depuis la dernière période (par exemple, des clients retirés ou des tâches annulées, ajoutées ou modifiées).

Elle procède ensuite aux étapes suivantes :

1. Supprimer les entités qui ne sont plus valides (client retiré, tâche terminée...)
2. Ajouter les nouvelles entités dans la solution, avec une méthode simple (par exemple une méthode gloutonne)
3. Mettre à jour les paramètres du problème (coûts, contraintes...)
4. Vérifier que la solution obtenue reste réalisable, si besoin appliquer une réparation minimale

La fonction renvoie la nouvelle solution X mise à jour.

Sélection de la solution optimale : fonction `get_new_optimal(`X^{opti}`)`

La sélection de la nouvelle solution optimale est à effectuer avec précaution. En effet, si les solutions optimales successives diffèrent trop, cela compromet la convergence du processus de réoptimisation : se rapprocher de la solution optimale à la période t ne garantira pas que un rapprochement de celle de la période $t+1$. Pour limiter ce phénomène, nous introduisons un critère de stabilité dans la sélection de la solution optimale. Pour cela, nous introduisons une seconde fonction objectif dans notre modèle d'optimisation. Deux objectifs sont ainsi définis :

— Objectif 1 (prioritaire) : objectif du problème que l'on souhaite résoudre
— Objectif 2 : minimiser l'écart par rapport à la solution optimale précédente

Les deux objectifs sont traités par ordre hiérarchique. Si plusieurs solutions minimisent l'objectif 1, on sélectionnera celle qui minimise la distance d par rapport à la solution optimale de la période précédente.

Cette sélection avec deux objectifs pourrait également être mise en place dès le début, afin de sélectionner une première solution optimale la plus proche de notre affectation initiale. Cela pourrait également être utilisé dans le cas statique pour déterminer la solution optimale de référence.

La fonction `get_new_optimal(`$Xopti$`)` prend en entrée la solution optimale précedente, résout le problème considéré avec les objectifs décrits précédemment et renvoie la nouvelle solution optimale référence.

Réparation de la solution courante : fonction `repair(`X`, `X^{opti}`, `$max_changes$`)`

La fonction `repair` permet d'ajuster la solution courante X pour la rapprocher de la solution optimale cible X^{opti}. Le principe reste le même que dans le cas statique.

La fonction prend en entrée la solution courante X, la solution optimale référence X^{opti} et le nombre maximum de changements autorisés $max_changes$.

Pour réparer, elle utilise la formulation PLNE du modèle de réparation décrite dans la section 3.1.1. La solution courante X est utilisée comme état de départ et on résout le modèle pour obtenir la solution réparée X^{new}, qui est finalement renvoyée par la fonction.

Contrairement au cas statique, cette réparation n'est appliquée qu'une fois par période.

Ainsi, à chaque période, la solution courante est mise à jour (`update`), puis on recalcule une cible optimale (`get_new_optimal`). Enfin, on effectue une étape de réparation (`repair`). Ce processus est répété jusqu'à la fin de l'horizon considéré.

3.1.4 Limites

Cette méthode présente un comportement **myope** : à chaque période, la réparation est effectuée en cherchant à se rapprocher de la solution optimale de la période t, sans considération pour l'évolution future du système. Or, l'absence d'anticipation empêche toute stratégie à long terme : une réparation jugée pertinente à la période t peut devenir inutile ou sous-optimale dès la période $t + 1$.

Par conséquent, pour dépasser ces limites, nous souhaitons introduire une forme d'anticipation dans le processus de décision. L'objectif est de sélectionner les réparations les plus bénéfiques à moyen terme. C'est dans cette optique que nous introduisons, dans la section suivante, une stratégie de réoptimisation qui prend en compte le futur : la recherche arborescente simulée.

3.2 Méthode arborescente

Pour pallier à ces limites, nous voulons intégrer un mécanisme permettant d'évaluer l'impact à moyen terme des décisions prises. Pour cela, nous implémentons une procédure de recherche arborescente guidée par simulation. L'idée est d'explorer plusieurs réparations candidates à l'instant t, puis d'évaluer leur impact sur un horizon de h semaines futures, en simulant l'évolution du système.

3.2.1 Lien avec le Monte-Carlo Tree Search

Le Monte-Carlo Tree Search (MCTS) est une famille d'algorithmes de recherche qui combine la recherche arborescente et l'utilisation d'échantillons aléatoires [40]. Dans cette méthode, un arbre de décision est construit et étendu de manière incrémentale. À chaque itération, quatre étapes sont effectuées :

1. **Sélection** d'un nœud à explorer dans l'arbre
2. **Expansion :** à partir de ce nœud, ajout d'un nouveau nœud enfant (sauf dans le cas d'un noeud terminal)
3. **Simulation (*rollout*) :** à partir du nouvel enfant, une simulation aléatoire est effectuée jusqu'à un état terminal ou jusqu'à un horizon fixé. Le résultat fournit une estimation de la qualité de ce chemin.
4. **Rétropropagation (*backpropagation*) :** le résultat de la simulation est rétropropagé en remontant l'arbre jusqu'à la racine et en mettant à jour la récompense associée à chaque nœud

À la fin de la procédure, le nœud enfant de la racine ayant la récompense la plus élevée est sélectionné.

Notre méthode s'inspire de ce principe, car nous utilisons également un arbre de décisions et une simulation pour évaluer nos actions. Cependant, notre approche diffère, principalement en ce qui concerne la phase de sélection. En effet, dans un MCTS classique, la sélection du nœud à visiter est guidée par une politique qui cherche un compromis entre exploitation des chemins prometteurs et exploration de nouveaux chemins. Dans notre version, nous générons directement un ensemble de chemins complets depuis la racine. Chaque chemin correspond à une séquence de changements de longueur au plus $max_changes$), puis nous évaluons chacun de ces chemins par une simulation déterministe sur un horizon h.

Nous choisissons cette approche pour plusieurs raisons. Tout d'abord, elle permet une exploration limitée des actions possibles. En effet, selon le problème étudié, certaines actions ont plus de potentiel que d'autres. Par exemple, dans un contexte dynamique, modifier la planification d'une tâche qui s'arrête à la période suivante est moins pertinent que modifier la planification d'une tâche qui demeurera dans le planning pour de nombreuses périodes. Certaines branches de l'arbre devenant rapidement obsolètes, on ne les explore pas. De plus, cette limitation du nombre de chemins explorés permet de garantir un temps de calcul raisonnable.

3.2.2 Principe de la méthode

Notre méthode se présente donc de la manière suivante : en partant de l'état courant du système, nous construisons un ensemble de nœuds enfant. Un nœud enfant correspond à un sous-ensemble de patients à réparer (dans la limite du budget *max_changes*). Chacune de ces réparations candidates est ensuite évaluée à l'aide d'une simulation (ou *rollout*) de l'évolution du système sur un horizon de h périodes. La réparation ayant généré le coût global le plus faible à l'issue de la simulation est retenue. Cette approche permet de tester plusieurs choix de modifications possibles, et de privilégier l'impact à long terme plutôt que leur effet immédiat.

Le pseudo-code général est le suivant :

Algorithm 3: Recherche arborescente avec simulation

Input: *root_node* : état actuel du système
max_changes : nombre maximum de changements autorisés
nb_sim : nombre de simulations
h : horizon

Output: *best_child*

```
1 children_nodes ← generate_children(root_node, max_changes, nb_sim) ;
2 for child_node in children_nodes do
3     reward ← simulate_rollout(child_node, h) ;
4     backpropagate(child_node, reward) ;
5 end
6 best_child ← compute_best_child(root_node) ;
```

Dans cet algorithme, *root_node* représente l'état actuel de notre planification, qui sera utilisé comme racine de l'arbre. Le paramètre *max_change* est le même que précédemment : le nombre maximum de changements autorisés chaque semaine. Les paramètres *nb_sim* et h sont spécifiques à la recherche arborescente : le premier représente le nombre de simulations effectuées (donc le nombre de réparations testées), le second représente le nombre de semaines qui seront simulées pour évaluer l'impact des changements faits.

Nous détaillons à présent chaque phase.

Génération des nœuds enfants : fonction `generate_children(root_node, max_changes, nb_sim)`

À partir de l'affectation actuelle (*root_node*), nous générons un ensemble de réparations

candidates. Chacune de ces réparations correspond à une modification possible du planning actuel, dans la limite de *max_changes*. Le nombre d'enfants générés est fixé par le paramètre *nb_sim* et correspond au nombre de trajectoires qui seront testées.

La fonction renvoie l'ensemble des réparations candidates, qui seront ensuite testées.

Simulation : fonction `simulate_rollout(child_node, horizon)`

Chaque nœud enfant est ensuite évalué par une simulation sur un horizon de h semaines. Durant cette simulation, la planification est mise à jour. Aucune réparation supplémentaire n'est effectuée pendant la simulation : l'idée est d'évaluer les effets durables d'un ensemble de modifications effectuées à t comme s'il s'agissait des dernières autorisées.

La récompense *reward* associée à chaque enfant correspond au coût total de la solution après l'ensemble des semaines simulées. C'est cette récompense qui est renvoyée par la fonction.

Backpropagation et choix du meilleur enfant : focntions `backpropagate(child_node, reward)` et `compute_best_child(root_node)`

À la fin de chaque simulation, la récompense obtenue est mémorisée pour le nœud correspondant avec la fonction `backpropagate`.
Une fois toutes les simulations terminées, le meilleur enfant est sélectionné selon le critère souhaité : le coût total après h semaines, le nombre de différences entre la planification enfant et la planification optimale après h semaines ou un hybride des deux. La fonction `best_child` renvoie ce meilleur enfant sous forme d'un ensemble de changements à effectuer sur la planification initiale.

Finalement, le pseudo code de notre méthode de réparation avec recherche arborescente est le suivant :

Algorithm 4: Réoptimisation dynamique avec méthode arborescente

Input: X^{init} : solution initiale

$X^{opti,init}$: solution optimale initiale

$max_changes$

Output: nb_steps, solutions intermédiaires

```
nb_steps ← 0 ;
X ← X^init ;
X^opti ← X^opti, init ;
while horizon non terminé do
    X ← update(X) ;
    X^opti ← get_new_optimal(data) ;
    best_changes ← tree_search(X, X^opti, max_changes)
      X ← apply_changes(X, best_changes) ;
    nb_steps ← nb_steps + 1 ;
end
```

La fonction `tree_search` applique la procédure décrite dans cette section et renvoie les meilleurs changements à effectuer à l'issue de celle-ci. La fonction `apply_changes` applique ensuite ces changements à la solution courante et renvoie la solution réparée.

Les fonctions `update` et `get_new_optimal` sont les mêmes que dans la section précédente.

CHAPITRE 4 MODÉLISATIONS DU PROBLÈME DE PLANIFICATION DES SOINS À DOMICILE

Le problème de planification des soins à domicile est un problème NP-difficile comprenant un grand nombre de contraintes. Nous souhaitons tester notre méthode de réoptimisation sur ce problème. Nous avons décidé de prendre en compte un certain nombre de contraintes régulièrement présentes dans les problèmes de planification des soins à domicile.

Pour chaque patient, un nombre de visites nécessaires par semaine doit être respecté. Nous faisons néanmoins l'hypothèse que les patients sont disponibles tous les jours de la semaine. Nous intégrons la continuité des soins comme une contrainte dure : le patient doit être visité par un unique soignant sur toute la période. Nous intégrons également une contrainte de compétence : certains patients nécessitent que le soignant qui les visite possède un certain *skill*.

Concernant les soignants, ils sont contraints par une limite maximale de temps de travail par jour ainsi qu'un nombre maximal de patients qui peuvent leur être attribués simultanément.

L'objectif de notre modèle sera de minimiser la distance totale parcourue par les soignants. Pour cela, nous utiliserons une estimation de cette dernière.

Pour tester notre méthode de replanification dans un contexte dynamique, nous allons avoir besoin de recalculer à chaque période la planification optimale. Il est donc nécessaire de pouvoir obtenir une telle planification en un temps raisonnable. Pour cela, nous décidons de le décomposer en deux phases distinctes :

1. **Phase d'affectation et de scheduling** : déterminer, pour chaque patient, le soignant responsable ainsi que les jours de visite.
2. **Phase de *routing*** : établir, pour chaque journée et pour chaque soignant, l'ordre des visites.

Dans notre approche, la première phase est formulée comme un problème de programmation linéaire en nombres entiers, que nous résolvons de façon exacte pour obtenir une affectation optimale. La seconde phase correspond à un problème de *routing* similaire à un problème de voyageur de commerce. Nous le résolvons avec une métaheuristique, qui ne nous garantit donc pas l'optimalité. Cependant, pour plus de lisibilité, nous utiliserons le terme « solution optimale » pour désigner la solution obtenue à l'issue des deux phases.

4.1 Problème d'affectation

Dans un premier temps, nous résolvons le problème d'affectation et de *scheduling*. Pour chaque patient, nous déterminons quel soignant lui est attribué et quels seront ses jours de visite. Les données et variables sont représentées dans le tableau 4.1.

TABLEAU 4.1 Paramètres du modèle d'affectation

Symbole	Description
Ensembles	
I	Ensemble des patients
J	Ensemble des jours considérés (jours de la semaine travaillés)
K	Ensemble des soignants
Données	
c_{ik}	Coût d'assigner le patient i au soignant k
c_{max}	Nombre maximum de patients assignables à un soignant
d_i	Durée d'une visite pour le patient i
sr_i	1 si le patient i nécessite que le soignant associé possède une certaine compétence
s_k	1 si le soignant k possède la compétence nécessaire
Variables de décision	
x_{ijk}	1 si le patient i est assigné au soignant k le jour j
y_{ij}	1 si le patient i a un rendez-vous le jour j
z_{ik}	1 si le patient i est affecté au soignant k

La fonction objectif et les contraintes du modèles sont les suivantes :

Objectif

$$\min \sum_{i,k} c_{ik} z_{ik} \tag{4.1}$$

Contraintes

$$\sum_k z_{ik} = 1 \qquad \forall i \in I \tag{4.2}$$

$$\sum_j y_{ij} = nv_i \qquad \forall i \in I \tag{4.3}$$

$$x_{ijk} = y_{ij} z_{ik} \qquad \forall i \in I, \forall k \in K, \forall j \in J \tag{4.4}$$

$$\sum_i z_{ik} \leq c_{max} \qquad \forall k \in K \tag{4.5}$$

$$\sum_i d_i x_{ik} \leq 8, \qquad \forall j \in J, \forall k \in K \tag{4.6}$$

$$sr_i \times z_{ik} \leq s_k \qquad \forall i \in I, \forall k \in K \tag{4.7}$$

$$x_{ijk} \geq 0 \qquad \forall i \in I, \forall k \in K \tag{4.8}$$

$$y_{ij} \geq 0 \qquad \forall j \in J \tag{4.9}$$

$$z_{ik} \geq 0 \qquad \forall k \in K \tag{4.10}$$

L'équation (4.1) est l'objectif de notre modèle : minimiser le coût d'affectation total des patients aux soignants. La contrainte (4.2) assure que chaque patient est assigné à exactement un soignant. La contrainte (4.3) assure quant à elle que le patient est visité exactement le nombre de fois nécessaires sur une semaine. La contrainte (4.4) assure le lien entre les variables x_{ijk}, y_{ij} et z_{ik}. Les contraintes (4.5) et (4.6) assurent le respect des limites liées aux soignants : chaque soignant travaille moins de 8h par jour et est affecté à au plus c_{max} patients. L'équation (4.7) assure que si un patient nécessite une certaine compétence, il est assigné à un soignant qui la possède. Enfin, les contraintes (4.8), (4.9) et (4.10) assurent le respect du domaine des variables.

Coût d'affectation

Pour déterminer le coût d'affectation d'un patient à un soignant, on souhaite se baser sur la distance parcourue par le soignant : ajouter un patient proche des patients déjà affectés à ce soignant devrait être peu coûteux, tandis qu'ajouter un patient éloigné devrait être coûteux. Pour refléter cela, on désigne pour chaque soignant un patient référence. Le coût d'affectation d'un nouveau patient à ce soignant sera alors la distance entre ce patient référence et le nouveau patient. Lorsque le patient référence termine son traitement, on désigne un nouveau patient référence parmi ceux actuellement assignés au soignant. Pour cela, on choisit le patient

le plus proche de tous les autres patients affectés à ce soignant. Plus formellement, cela revient à résoudre pour le soignant k le problème d'optimisation suivant :

$$\min_{i \in I_k} \sum_{i' \in I_k} d_{ii'}$$

avec I_k l'ensemble des patients assignés au soignant k et en notant $d_{ii'}$ la distance entre les patients i et i'.

Réparation

Dans le problème d'affectation, les mouvements de réparation possible sont définis par rapport aux patients. Ces deux mouvements sont :

- modifier le soignant associé au patient
- modifier les jours de visite du patient

Le premier changement implique souvent le deuxième. En effet, on considère que si le soignant est modifié, il faut s'assurer que les jours de visite qui étaient prévus avec l'ancien soignant restent compatibles avec l'emploi du temps du nouveau. Dans le cas contraire, les jours sont modifiés et on considère que l'on a effectué deux changements. Ainsi, en autorisant un seul changement, on pourrait se retrouver dans une situation bloquante où tous les patients mal affectés doivent être changés à la fois de soignant et de jours et donc aucun ne pourrait être modifié sans violer la contrainte sur le nombre de changements. Pour éviter ce cas, on autorise toujours au moins deux changements dans nos tests.

Pour éviter de se retrouver dans une situation où aucun changement n'est possible, on autorise au minimum deux changements dans nos tests.

4.2 Problème de routing

Une fois l'affectation et le *scheduling* optimaux obtenus, nous résolvons le problème de *routing*. Pour chaque soignant et pour chaque jour, nous disposons d'une liste de patients à visiter. Il reste alors à déterminer dans quel ordre effectuer ces visites afin de minimiser la distance totale parcourue par le soignant.

Ce sous-problème correspond à un cas particulier du VRP : le problème du voyageur de commerce (Travelling Salesman Problem (TSP)), dans lequel un seul véhicule (ici, le soignant) doit visiter un ensemble de clients (ici, les patients).

Mathématiquement, on le modélise de la manière suivante :

Tableau 4.2 Paramètres du modèle de routing

Symbole	Description
Ensembles	
J	Ensemble des jours considérés (jours de la semaine travaillés)
K	Ensemble des soignants
I_{jk}	Ensemble des patients affectés au soignant k le jour j
Données	
$d_{ii'}$	Distance entre les patients i et i'
Variables de décision	
$x_{ii'}$	1 si le patient i' est visité immédiatement après le patient i

Objectif

$$\min \sum_{i,i' \in I_{jk}} c_{ii'} d_{ii'} \tag{4.11}$$

Contraintes

$$\sum_{i' \in I_{jk}} x_{ii'} = 1 \qquad \forall i \in I_{jk} \tag{4.12}$$

$$\sum_{i \in I_{jk}} x_{ii'} = 1 \qquad \forall i' \in I_{jk} \tag{4.13}$$

$$\sum_{i \in \overline{S}} \sum_{i' \in \overline{S}} x_{ii'} \geq 1 \qquad \forall S \subset I_{jk}, S \neq \emptyset \tag{4.14}$$

$$x_{ii'} \geq 0 \qquad \forall i, i' \in I_{jk}, \tag{4.15}$$

L'équation (4.11) représente notre objectif : minimiser la distance totale parcourue par le soignant. Les équation (4.12) et (4.13) assurent que chaque patient n'a qu'un seul successeur et un seul prédecesseur. La contrainte (4.14) assure qu'une seule route est construite et non plusieurs cycles. Enfin, l'équation (4.15) détermine le domaine des variables.

Pour résoudre ce modèle, nous allons utiliser une méthode développée par Vidal et al. [41,42] et implémentée dans un solveur open-source. Cette méthode a été généralement développée pour résoudre le problème de tournées de véhicules avec contraintes de capacité Capacitated Vehicle Routing Problem (CVRP). Cette variante du VRP classique ajoute une contrainte

selon laquelle chaque véhicule ne peut accepter qu'un certain nombre de clients (contrainte de capacité). La métaheurisitque développée par Vidal et al. pour résoudre ce problème est un algorithme génétique hybride comprenant également une exploration de différents voisinages.

Réparation

Pour la phase de routing, la réparation associée est la modification des routes. Quand nous intégrerons un horizon dynamique, les patients qui quittent le système laisseront dans les routes des trous qui ne seront pas systématiquement comblés par les nouveaux patients arrivant. On autorisera alors deux types de réparation :

— déplacer un patient vers une position vide de la route. Cela compte comme un changement.

— échanger les positions de deux patients. Cela compte comme deux changements.

4.3 Planification initiale

4.3.1 Cas statique

Afin de tester notre méthode sur un problème de planification de soins à domicile statique, nous devons créer une planification initiale qui représente la planification actuelle. Cette planification doit être une solution au problème admissible mais non optimale, qui servira de point de départ pour le processus de réoptimisation. Nous la construisons selon une heuristique gloutonne basée sur le regroupement géographique. Le processus se déroule en deux temps :

— **Sélection des patients de référence :** les patients de référence sont sélectionnés de manière itérative en maximisant leur éloignement géographique mutuel. Pour cela, un premier patient est d'abord choisi aléatoirement. Cette stratégie vise à créer des regroupements de patients spatialement cohérents, chaque groupe étant destiné à un soignant. Idéalement, les patients attribués à un même soignant doivent être proches les uns des autres et éloignés des patients attribués aux autres soignants. Choisir des patients référence éloignés les uns des autres permet de faciliter cela. Le nombre de patients référence est fixé au nombre de soignants disponibles. Ces patients sont ensuite affectés aléatoirement aux soignants, sous réserve de compatibilité avec leurs compétences. Plus précisément, le nombre de patients nécessitant une compétence particulière ne doit pas excéder le nombre de soignants en disposant.

— **Affectation des patients restants :** les patients non sélectionnés sont ensuite affectés un par un à l'aide d'un critère glouton : pour chaque patient, on choisit le soignant

dont l'affectation entraîne le plus faible coût, sous réserve de faisabilité. Le coût est ici la distance entre le patient et le patient référence associé au soignant. Comme nous n'avons aucune préférence à prendre en compte sur les jours de visite, ceux-ci sont choisis aléatoirement parmi les jours autorisés, en tenant compte de la contrainte de durée maximale de travail quotidien des soignants.

4.3.2 Cas dynamique

Pour le cas dynamique, nous débutons avec la solution optimale comme planification initiale. Le système est ensuite simulé sur une période de 10 itérations (correspondant à 10 semaines), au cours desquelles la planification évolue de manière gloutonne sans processus de réparation.

Pendant cette période, les patients dont le traitement est terminé sont retirés de la planification et les nouveaux patients sont intégrés selon un processus glouton : ils sont affectés aux soignants disponibles en minimisant le coût d'attribution, tout en respectant les contraintes de faisabilité.

Au terme des 10 semaines, on obtient ainsi une planification toujours réalisable mais sous-optimale, qui servira de point de départ représentatif pour évaluer notre méthode de réoptimisation

4.4 Application de la méthode arborescente

Enfin, nous donnons quelques détails sur la manière dont nous appliquons notre méthode arborescente à ce cas d'étude :

— La génération de noeuds enfants correspond à la génération de combinaisons de patients à modifier. Pour identifier les réparations les plus prometteuses, on calcule pour chaque patient un indice de priorité basé sur le produit $coût \times durée\ restante$ est calculé. $coût$ représente le coût d'affectation du patient à son soignant actuel. Le but est de sélectionner les patients restant longtemps dans le système et ceux qui entraînent le plus fort surcoût.Les patients dont le score est le plus élevé sont sélectionnés. Pour générer une réparation candidate, des patients sélectionnés sont choisis aléatoirement jusqu'à atteindre le budget de $max_changes$ changements à chaque itération

— Lors de la simulation, le noeud enfant est évalué sur un horizon de h semaines. Pour chaque semaine, les patients dont le traitement se termine sont retirés et l'arrivée de nouveaux patients est simulée. Comme nous n'avons pas d'accès aux patients futurs, nous simulons ces nouveaux patients en sélectionnant d'anciens patients dont le traitement est terminé et qui ne sont plus dans le système.

CHAPITRE 5 EXPÉRIMENTATIONS ET RÉSULTATS

Cette partie est dédiée au test des méthodes proposées et à l'analyse des résultats obtenus. Comme développé précédemment, nous effectuons ces tests sur un problème de planification des soins à domicile. Afin de valider notre approche, nous avons d'abord créé un ensemble de patients que nous avons ensuite répartis dans différentes instances.

L'expérimentation s'articule en trois étapes principales, correspondant à des cas d'étude de complexité croissante. Dans un premier temps, nous testons notre méthode sur un problème d'affectation statique, où la liste des patients est fixe et la planification ne subit pas de modifications, afin de valider le bon fonctionnement de la méthode proposée. Dans un second temps, nous étendons l'approche à un contexte dynamique, avec l'ajout d'événements tels que l'arrivée ou le départ de patients, ce qui nécessite une réorganisation progressive du planning. Enfin, nous abordons un problème combinant assignation dynamique et routing, afin de se rapprocher d'un problème pouvant être rencontré dans des cas réels.

Pour chaque cas d'étude, nous présentons les résultats obtenus ainsi que leur analyse. Nous proposons ensuite des recommandations concrètes destinées aux décideurs.

5.1 Génération des instances

Pour tester notre méthode, nous créons des instances de test. Pour cela, nous commençons par créer un certain nombre de patients et de soignants, puis nous les regroupons en instances qui seront ensuite utilisées dans nos algorithmes de réoptimisation.

5.1.1 Création des soignants

Pour nos expérimentations, nous considérerons trois ensembles de soignants, constitués respectivement de quatre, huit et douze soignants.

Ces soignants ont tous les mêmes contraintes : leur temps de travail journalier ne peut pas excéder huit heures et ils ont tous le même nombre maximal de patients qui peuvent leur être attribués simultanément.

Nous avons intégré dans notre modèle une contrainte de compétence. Certains patients ne

peuvent être attribués qu'à des soignants possédant un certain *skill*. Dans chaque ensemble, nous considérons que la moitié des soignants possède cette compétence. Une seule compétence est représentée.

5.1.2 Création des patients

Pour tester notre méthode, nous avons besoin de générer un ensemble de patients.

Pour chaque patient, les données dont nous avons besoin sont les suivantes :

- sa position (sous forme de coordonnées (x,y)) ;
- le nombre de fois qu'il doit être vu par semaine ;
- la durée de visite nécessaire ;
- s'il est nécessaire qu'il soit vu par un soignant possédant un certain niveau de compétences ;
- dans un contexte dynamique, sa durée de traitement.

Données communes aux cas statiques et dynamiques

Le problème de soins à domicile se rapprochant d'un VRP, nous utilisons pour les coordonnées des patients des jeux de données classiques utilisés pour ces problèmes.
En particulier, nous utilisons les instances de Solomon [43]. Ces instances se divisent en 3 catégories : R, C et RC. Les instances R contiennent des coordonnées générées de manière aléatoires, les instances C contiennent des coordonnées réparties dans des clusters et les instances RC contiennent un mélange des deux : des clusters mais aussi des points plus aléatoires. Nous avons choisi d'utiliser les instances RC.

Concernant les autres données, le nombre de visites nécessaires par semaine est généré uniformément entre 1 et 5. La durée d'une visite est de 30, 45 ou 60 minutes, également choisi de manière uniforme. Un patient nécessitera que le soignant ait un certain niveau de compétences avec une probabilité de 0.25. On s'assure expérimentalement que le nombre de soignants est suffisant pour que le nombre de patients refusés reste limité.

Donnée spécifique au cas dynamique : la durée de traitement

Dans le cas dynamique, une durée de traitement est attribuée à chaque patient. Cette durée, comptée en nombre de semaines, correspond à la période durant laquelle le patient doit être vu par un soignant. Cela permet de créer un système de roulement : chaque semaine, les patients dont le traitement est terminé sont retirés du système et de nouveaux patients sont susceptibles d'être admis. Le nombre de nouveaux patients proposés chaque semaine

est proportionnel au nombre de patients sortants, afin de maintenir une charge suffisante dans le système. Nous faisons l'hypothèse suivante : un patient entrant n'est accepté que si il est possible de l'affecter à un soignant et à des jours de visite tout en respectant toutes les contraintes. Dans le cas contraire, le patient n'est pas accepté. Le nombre de patients refusés devient donc un indicateur de performance : une solution permettant d'accepter plus de patients reflète une meilleure gestion du planning.

Chaque patient se voit ensuite attribuer une durée de traitement. Nous souhaitons tester l'impact du **turnover** sur la performance de notre réoptimisation. En effet, si les patients ont des durées de traitement rapide, le turnover sera important, ce qui entraîne une forte variabilité de la solution optimale au fil des semaines. À l'inverse, si les durées de traitement sont longues, le turnover sera assez faible, ce qui permettra la stabilisation de la solution optimale. Pour contrôler ce paramètre, nous tirons la durée de traitement d'un patient aléatoirement selon une loi bêta dont nous ajustons les paramètres afin d'en contrôler la moyenne.

La loi bêta est une distribution de probabilité définie sur l'intervalle $[0, 1]$ et caractérisée par deux paramètres α et β. Cette distribution est particulièrement flexible :

- si $\alpha = \beta = 1$, elle est uniforme sur $[0, 1]$;
- si $\alpha > \beta$, elle est biaisée vers les valeurs proches de 1 ;
- si $\alpha < \beta$, elle est biaisée vers les valeurs proches de 0.

Dans notre cas, les valeurs générées par la loi bêta sont ensuite mises à l'échelle linéairement pour correspondre à des durées de traitement comprises entre 1 et 20 semaines. Concrètement, une valeur x issue de la loi bêta est transformée en $\lfloor 1 + 19x \rfloor$, assurant ainsi que la durée de traitement soit bien dans l'intervalle souhaité. Ainsi, une valeur $\alpha > \beta$ correspondra à de longues durées de traitement (turnover faible), tandis que des valeurs $\alpha < \beta$ correspondront à de faibles durées de traitement (turnover élevé).

5.1.3 Organisations en instances

Instances statiques

Pour tester notre approche sur un cas statique, nous générons 5 instances, chacune composée de 30 patients choisis aléatoirement parmi l'ensemble de patients que nous avons créé. Chaque patient est défini par sa localisation géographique, le nombre de jours où il doit être visité, la durée des visites et le besoin éventuel de soins nécessitant une compétence spécifique (appelée *skill*).

Concernant les soignants, nous utilisons pour chaque instance quatre soignants créés comme décrit précédemment. La contrainte limitant le nombre d'heures travaillées par jour pour un

soignant permet d'assurer que la charge de travail des soignants reste réaliste.

Instances dynamiques

Par la suite, nous testons notre modèle sur des cas dynamiques. Pour cela, nous fusionnons le nombre souhaité d'instances de Solomon comprenant chacune 101 patients. L'objectif est que le nombre de patients offre un bon compromis en permettant l'évolution suffisante du système pour évaluer les effets de la réoptimisation tout en conservant un temps d'exécution raisonnable.

Le choix de la durée de traitement des patients a son importance. Pour pouvoir tester l'influence du turnover sur notre modèle, nous créons les instances de manière à contrôler la moyenne sur l'instance de la durée de traitement des patients, à l'exception de la dernière pour laquelle nous choisissons les durées de traitement aléatoirement selon une loi uniforme.

Pour les loi bêta, les valeurs de α et β ont été choisies expérimentalement pour avoir des durées moyennes de traitement correspondantes à celles souhaitées.

Nous créons quatre instances comprenant 4 soignants et 303 patients, deux instances comprenant 8 soignants et 505 patients et deux instances comprenant 12 soignants et 808 patients. Un résumé des caractéristiques des instances est trouvable dans le tableau 5.1.

Tableau 5.1 Caractéristiques des instances dynamiques

Instance	Nombre de patients	Nombre de soignants	Durée moyenne de traitement (semaines)	Turnover
1	303	4	4.64	Rapide
2	303	4	8.78	Moyen
3	303	4	13.20	Lent
4	303	4	9.43	Aléatoire
5	505	8	10.36	Aléatoire
6	505	8	9.83	Aléatoire
7	808	12	10.16	Aléatoire
9	808	12	9.94	Aléatoire

5.2 Résultats pour le problème d'affectation statique

Le cas statique étant assez simple, il nous sert à valider notre méthode. Pour cela, on observe le nombre d'itérations nécessaires pour atteindre la solution optimale selon différentes valeurs

du paramètre *max_changes*. Ce paramètre représente le nombre maximum de changements que l'on autorise à chaque étape.

Le tableau 5.2 présente les écarts initiaux entre la solution heuristique et la solution optimale. Cet écart est compté de la manière suivante : pour chaque patient, l'écart est de 0 si il est affecté aux mêmes jours et au même soignant dans les deux solutions, de 1 si un des deux diffère et de 2 si les deux diffèrent.

Nous n'avons pas cherché à choisir la solution optimale la plus proche de la solution heuristique, et ce pour avoir des écarts suffisamment grands pour que nos tests soient intéressants. Cependant, si cette méthode venait à être appliquée dans un cas réel, il serait pertinent de choisir la solution optimale la plus proche de la solution heuristique. Cela peut être fait en utilisant une méthode avec deux objectifs telle que décrite dans la section 3.1.3.

Tableau 5.2 Écart initial pour chaque instance

Instance	Écart
instance 1	22
instance 2	30
instance 3	36
instance 4	25
instance 5	19

La figure 5.1 illustre le nombre d'étapes nécessaires pour atteindre la solution optimale en fonction de la valeur de *max_changes* pour chaque instance.

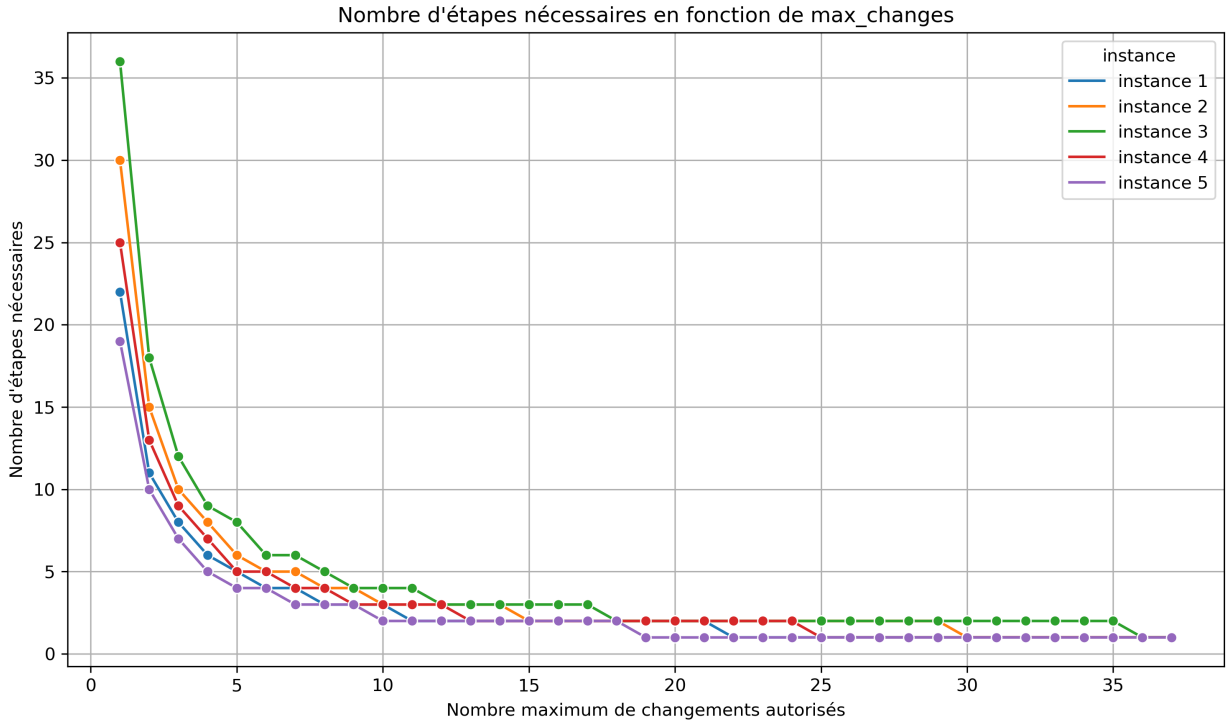

FIGURE 5.1 Nombre d'étapes nécessaires pour atteindre l'optimal en fonction du nombre de changements autorisés

Il apparaît que plus le nombre de changements autorisés est élevé, plus le nombre d'étapes nécessaires pour atteindre la solution optimale est faible. Ces résultats sont cohérents : autoriser davantage de modifications permet de s'approcher plus rapidement de la solution optimale. Plus précisément, le nombre d'étapes nécessaires pour atteindre la solution optimale correspond systématiquement au nombre minimal requis compte tenu de l'écart initial et du nombre maximal de changements autorisés par itération. Par exemple, si l'écart initial est de 44 et que le nombre de changements autorisés est de 12, alors $\lceil \frac{44}{12} \rceil = 4$ étapes seront nécessaires. Cela peut être le cas dans notre exemple car tous les échanges sont réalisables. La capacité de réparation à chaque étape est donc utilisée au maximum.

C'est confirmé par la figure 5.2 qui montre, pour chaque valeur de *max_changes*, le nombre de changements effectivement effectués à chaque étape pour l'instance 1. Pour chaque étape, le nombre maximum de changements possibles est effectué, sauf dans le cas où le nombre de changements nécessaires est inférieur à *max_changes*.

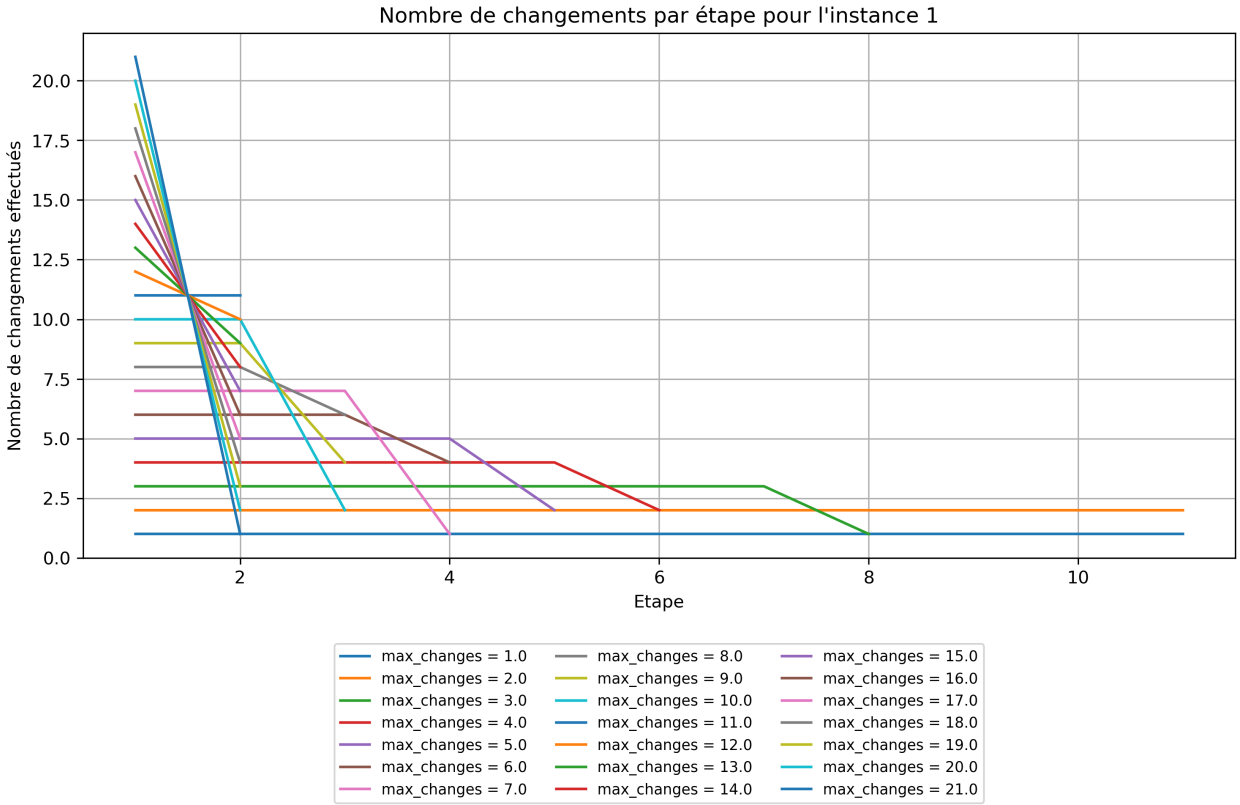

FIGURE 5.2 Nombre de changements effectués à chaque étape pour l'instance 1

On s'intéresse plus précisément aux changements effectués à chaque étape. La figure 5.3 montre, pour l'instance 2, les changements effectués à chaque étape. On remarque que chaque patient n'est bien déplacé qu'une fois : il est modifié de son affectation initiale à son affectation optimale. Cependant, l'ordre de modification des patients diffère selon $max_changes$, ce qui laisse penser à une certaine myopie du modèle, comme expliqué dans la section 3.1.5.

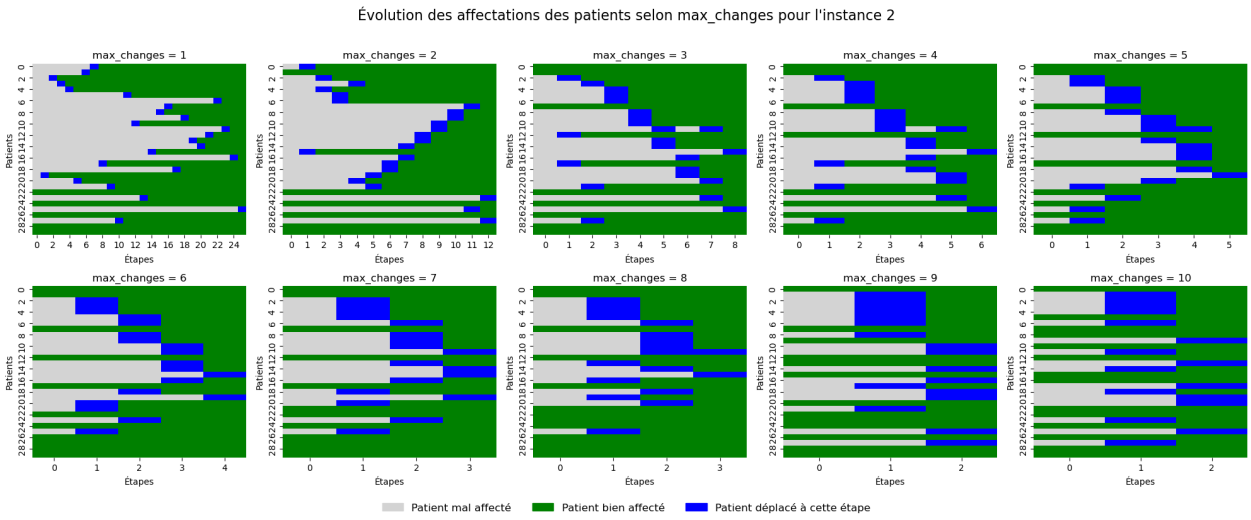

FIGURE 5.3 Évolution des affectations des patients selon max_changes pour l'instance 2

Ainsi, pour un décideur qui a une affectation initiale avec un écart à l'optimal de $ecart_initial$, le paramétrage du processus de réoptimisation dépend de ses objectifs. Deux cas de figure se présentent :

— Si le décideur souhaite atteindre la solution optimale en un nombre de semaines $nb_semaines$ fixé, il lui faudra autoriser $\left\lceil \frac{ecart_initial}{nb_semaines} \right\rceil$ changements par semaine.

— À l'inverse, si il souhaite autoriser $max_changes$ par semaine, il faudra prévoir au moins $\left\lceil \frac{ecart_initial}{max_changes} \right\rceil$ pour atteindre la solution optimale.

5.3 Résultats et discussion pour le problème d'affectation dynamique

Notre méthode ayant été validée sur un cas statique simple, nous nous intéressons désormais à sa version dynamique. Pour cela, nous commençons par tester la méthode initiale basée sur un PLNE, puis nous testerons la méthode arborescente.

5.3.1 Méthode PLNE

La figure 5.4 présente l'évolution de l'écart relatif entre la solution courante et la solution optimale, en fonction du nombre de changements hebdomadaires autorisés ($max_changes$), sur l'ensemble des semaines simulées.

Les résultats mettent en évidence plusieurs limites importantes de la méthode :

— Autoriser des changements tend à réduire l'écart à l'optimal, mais la relation entre

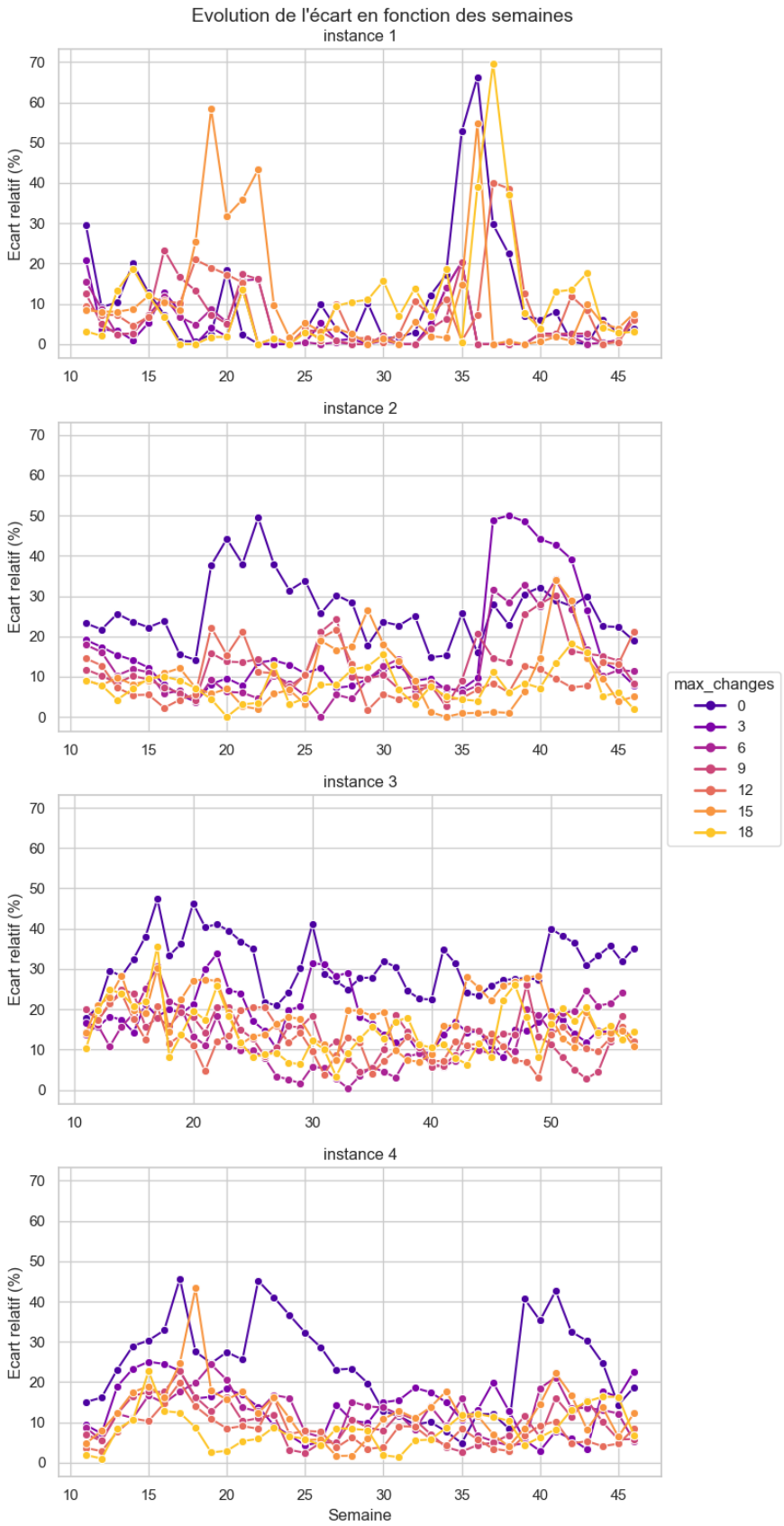

FIGURE 5.4 Écart relatif à l'optimal au cours du temps selon différents niveaux de *max_changes* (méthode myope)

max_changes et l'écart n'est ni linéaire ni systématique.

— Aucune convergence stable vers la solution optimale n'est observée : même avec un nombre important de changements autorisés, l'écart reste fluctuant et persiste au fil des semaines.

Par exemple, dans l'instance 4, (semaine 36, 6 changements autorisés), la décision de modifier le patient 233 apparaît discutable. En effet, bien que ce patient soit vu 4 fois par semaine, il n'avait plus qu'une semaine de traitement. En parallèle, plusieurs patients (233, 237, 241, 248 et 257) disposant encore de 11 à 16 semaines restantes n'ont pas été modifiés, bien que tous soient affectés au mauvais soigant et les mauvais jours. Des situations semblables se retrouvent régulièrement pour toutes les instances, et ce peu importe le nombre de changements autorisés.

Ces résultats étaient attendus et sont dûs à la myopie de la méthode utilisée, comme discuté dans la partie 3.1.5.

5.3.2 Méthode arborescente

Encore une fois, nous cherchons à évaluer la dynamique de convergence de la solution gloutonne vers la solution optimale en fonction du nombre de changements autorisés à chaque étape de la procédure de réoptimisation.

Pour évaluer l'impact de notre méthode, nous observons la différence de coût relative entre la solution optimale et la solution réparée pour différentes valeurs du paramètre *max_changes*.

La figure 5.5 donne nos résultats pour les instances 1 à 4 (4 soignants). La figure 5.6 les présente pour les instances 5 et 6 (8 soignants) et la figure 5.7 pour les instances 7 et 8 (12 soignants). Les figure montrent que cette fois-ci, plus le nombre de changements autorisés est élevé, plus la solution réparée tend à se rapprocher de la solution optimale. Plus précisément :

— à partir de 9 changements autorisés, l'écart avec la solution optimale est systématiquement inférieur à 10% ;

— à partir de 12 changements autorisés, la solution optimale est atteinte plus de la moitié du temps.

On remarque que la taille du système n'a pas de réel impact sur nos résultats. Cela peut s'expliquer par le fait que notre méthode gloutonne nous permet de maintenir un écart à l'optimal raisonnable, ce qui permet à la méthode arborescente de la "rattraper" peu importe la taille du système.

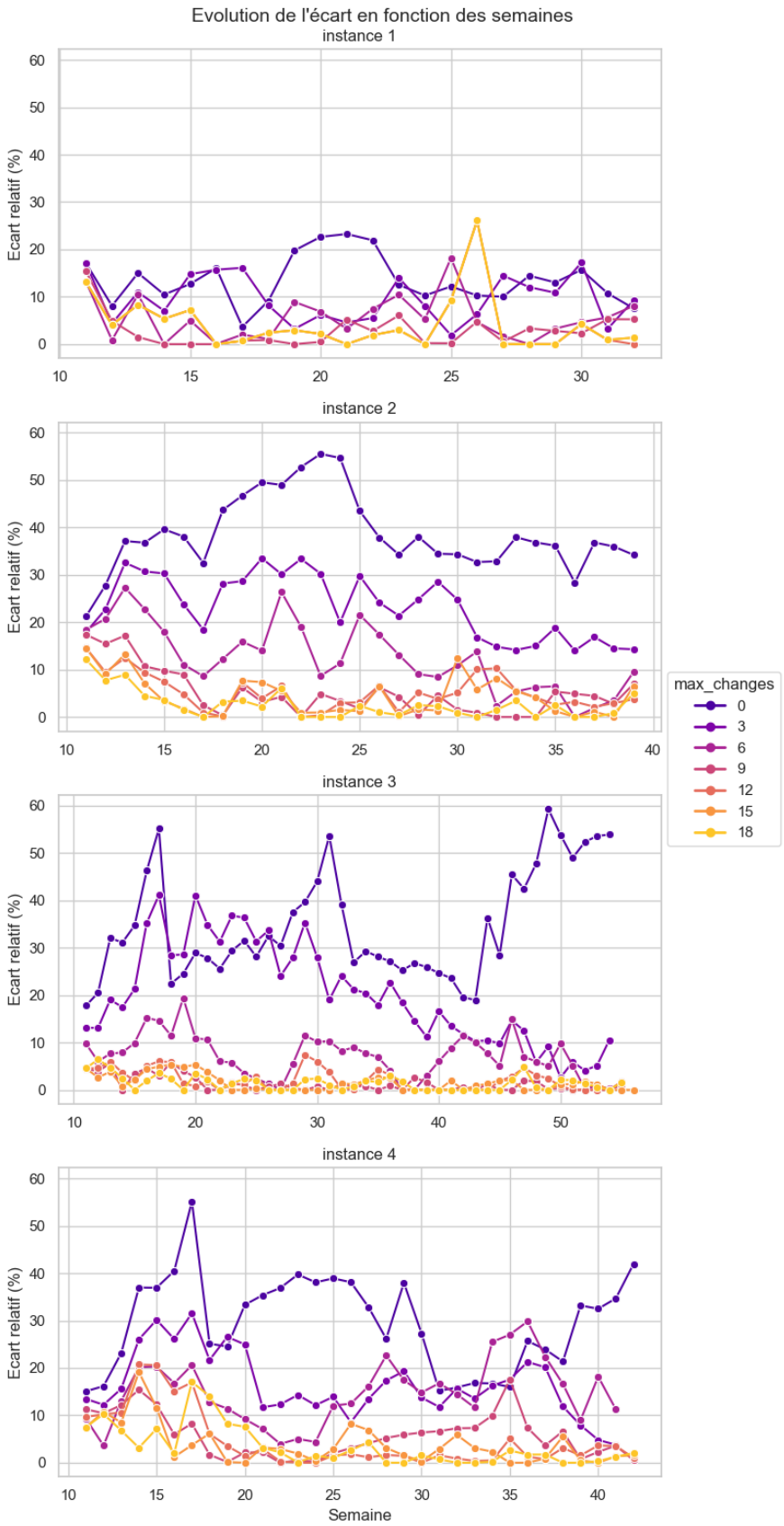

FIGURE 5.5 Évolution de l'écart relatif entre la solution courante et la solution optimale pour les instances 1 à 4 (méthode arborescente)

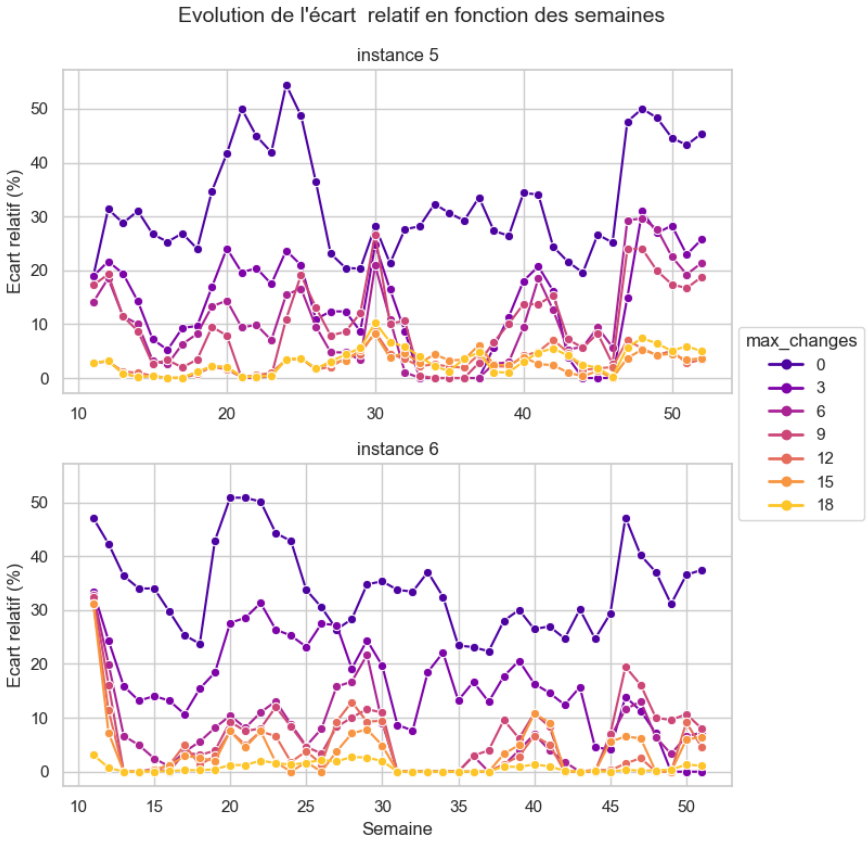

FIGURE 5.6 Évolution de l'écart relatif entre la solution courante et la solution optimale pour les instances 5 et 6 (méthode arborescente)

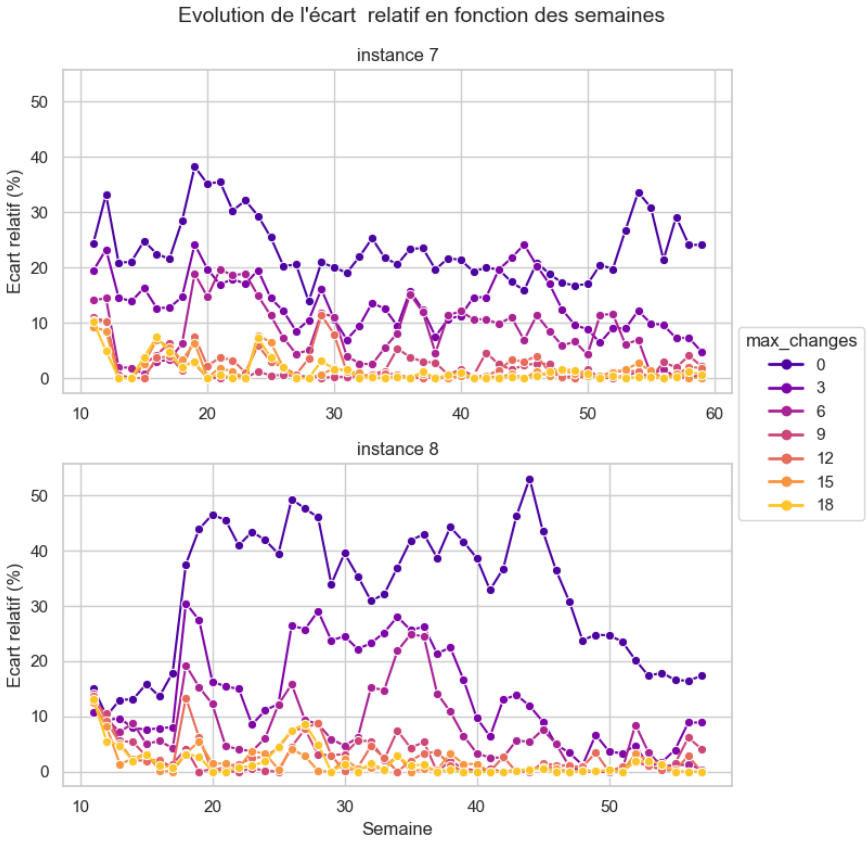

FIGURE 5.7 Évolution de l'écart relatif entre la solution courante et la solution optimale pour les instances 7 et 8 (méthode arborescente)

Cette méthode nous permet bien de mieux prendre en compte le futur. Les patients dont le traitement se termine bientôt sont moins modifiés que lorsque l'on utilise la méthode PLNE. Ainsi, la figure 5.8 montre le nombre de patients modifiés alors qu'il leur restait deux semaines de traitement ou moins pour l'instance 4. Nous prenons ces résultats pour un nombre de changements inférieur à 12. En effet, au delà, la solution optimale est presque systématiquement atteinte par la méthode arborescente, ce qui fausse les résultats. Il est clair sur la figure que la méthode PLNE modifie beaucoup plus de patients en fin de traitement que la méthode MCTS, ce qui est un choix sous-optimal.

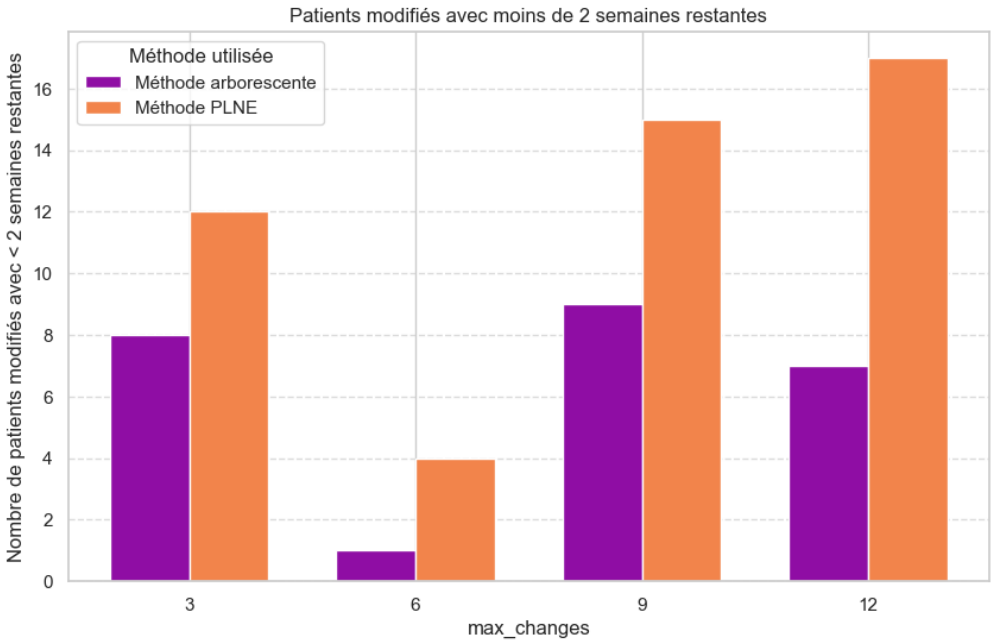

FIGURE 5.8 Comparaison du nombre de patients modifiés à deux semaines de la fin de leur traitement en fonction de la méthode utilisée pour l'instance 4

Importance du turnover

Le turnover des patients joue un rôle important dans les résultats :

— Pour l'instance possédant le turnover le plus rapide (instance 1), l'écart entre la solution courante et la solution optimale est relativement faible, même en l'absence de réparation ($max_changes = 0$). Cela s'explique par le fait que de nombreux patients quittent le système chaque semaine, libérant des créneaux dans les plannings : la solution gloutonne peut alors effectuer des affectations acceptables dès l'arrivée des nouveaux patients.

— Pour les autres instances, l'écart en l'absence de réparation peut atteindre 50%. Cela souligne l'intérêt de mettre en place un mécanisme de réparation.
— Par ailleurs, on observe que la réparation est d'autant plus efficace que le turnover est faible. En effet, un patient réparé restant longtemps dans le système, on finit par pouvoir réparer presque tous les patients avant qu'ils ne quittent le système.

Utilisation réelle des changements autorisés

On observe également un phénomène de saturation : au-delà d'un certain seuil, augmenter *max_changes* n'améliore plus significativement la qualité des solutions. Cela suggère que tous les changements autorisés ne sont pas toujours exploités par le solveur. Autrement dit, autoriser un grand nombre de modifications ne garantit pas qu'elles seront toutes utilisées ou pertinentes.

Pour illustrer ce phénomène, la figure 5.9 présente, pour l'instance 3, le nombre de changements effectivement réalisés à chaque itération, en fonction de la valeur de *max_changes*. On y observe que, passé un certain seuil, le nombre de changements effectifs plafonne.

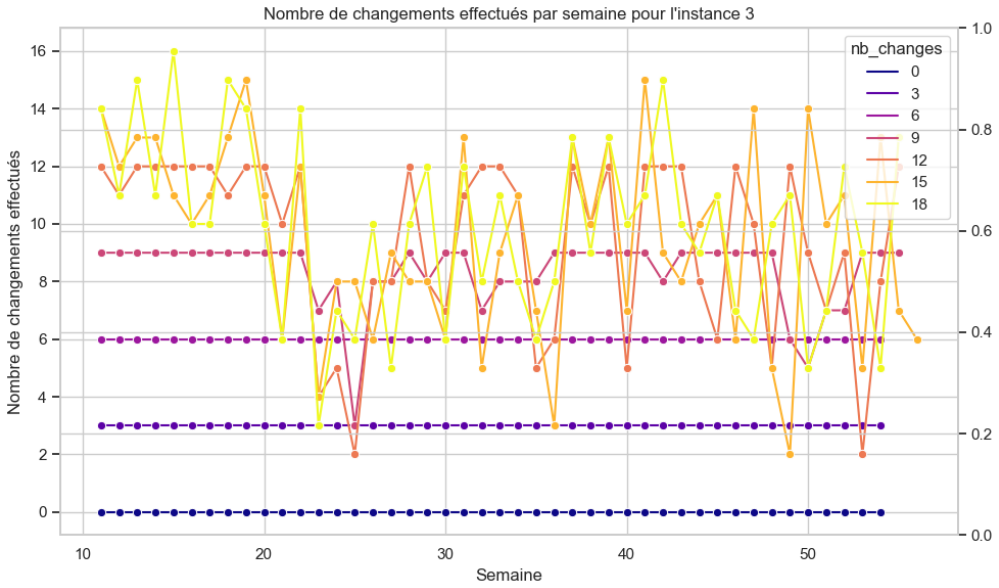

FIGURE 5.9 Nombre de changements effectués à chaque étape pour l'instance 3 (méthode arborescente)

Impact sur le nombre de patients refusés

La figure 5.10 montre, pour les instances 1 à 4, le nombre de patients refusés en fonction du nombre de changements hebdomadaires autorisés. Cette figure confirme nos résultats précédents, notamment concernant l'importance du turnover :

— Pour l'instance 1 (turnover rapide), très peu de patients sont refusés (moins de 5). Cela confirme le fait que l'affectation gloutonne donne de bons résultats.

— Pour les autres instances, plus le turnover est faible, plus le nombre de patients refusés est important.

— Augmenter le nombre de changements permet de réduire légèrement le nombre de patients refusés. C'est particulièrement le cas pour l'instance 3, où l'on passe de 39 patients refusés à 28.

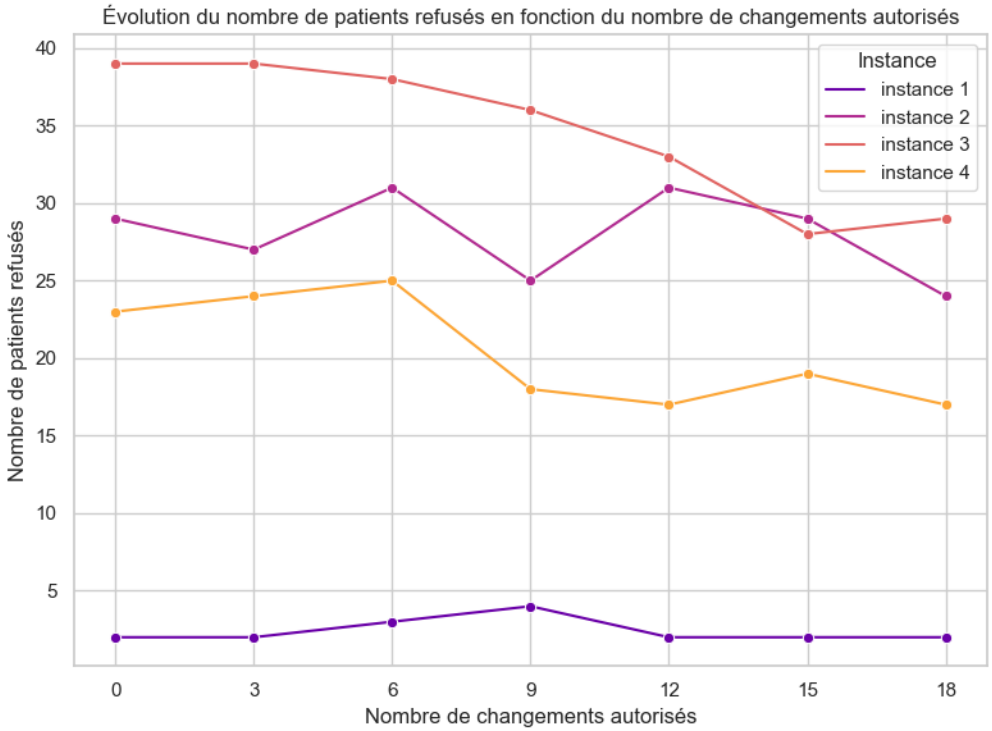

FIGURE 5.10 Évolution du nombre de patients refusés en fonction du nombre de changements autorisés (méthode arborescente)

5.3.3 Décisions opérationnelles

Les résultats précédents permettent de dégager plusieurs enseignements opérationnels sur la manière de piloter la réoptimisation d'un planning dynamique. Pour un décideur en charge de la gestion des tournées de soins, le paramétrage du nombre de changements autorisés par

semaine constitue un levier stratégique permettant de trouver un équilibre entre stabilité organisationnelle et performance globale.

Deux facteurs principaux doivent être pris en compte pour choisir le bon niveau de réoptimisation :

1. **Le turnover des patients** : plus le turnover est élevé (entrées/sorties fréquentes), moins la réoptimisation est nécessaire, car les plannings sont régulièrement "rafraîchis" naturellement. À l'inverse, un faible turnover signifie que les patients restent longtemps, ce qui justifie la nécessité d'une politique de réparation.
2. **La saturation du système** : les résultats montrent qu'au-delà d'un certain seuil, autoriser davantage de changements n'améliore plus significativement la performance. Cela invite à éviter les niveaux de réparation excessifs, coûteux à mettre en œuvre pour un gain marginal.

Recommandation synthétique

Nous proposons ci-dessous une matrice de décision qualitative à l'intention d'un décideur. Elle associe le niveau de turnover observé dans le système à un niveau de recommandation pour le paramètre *max_changes*. Nous considérons deux objectifs possibles :

— objectif 1 : stabilité : on souhaite garantir un écart entre la solution courante et la solution optimale autour de 10% tout en limitant le nombre de changements (quitte à parfois dépasser ce seuil) ;

— objectif 2 : performance : on souhaite garantir un écart maximal de 10% entre la solution courante et la solution optimale.

Tableau 5.3 Matrice de recommandation pour le choix du paramètre *max_changes*

Turnover	**Objectif : stabilité**	**Objectif : performance**
Rapide	3 à 6	6 ou plus
Moyen	6 à 9	9 ou plus
Lent	6 à 9	9 ou plus

Les résultats empiriques montrent que :

— À partir de 9 changements hebdomadaires, l'écart avec l'optimal devient très faible (souvent $< 10\%$) ;

— Au-delà de 12, les gains marginaux deviennent négligeables, tandis que la complexité opérationnelle augmentent.

Un décideur peut donc fixer *max_changes* entre 6 et 9 dans la plupart des cas pour obtenir un bon compromis entre performance et stabilité.

5.4 Résultats et discussion pour le problème complet

Enfin, nous nous intéressons au problème complet, qui comprend une phase d'affectation des patients aux soignants et aux jours de visite puis une phase de création des tournées. Ce problème étant plus complexe, il est d'autant plus nécessaire d'intégrer une prise en compte du futur dans nos décisions de replanification. Nous utilisons donc uniquement la méthode arborescente présentée précédemment.

Dans notre implémentation, nous effectuons la phase d'affectation avant la phase de routage. la qualité du routage dépend donc directement de la qualité de l'affectation. Pour obtenir des solutions proches de la solution optimale, nous devons donc nous approcher au mieux de l'affectation optimale.

Cette nécessité va orienter l'implémentation de notre méthode arborescente. Ainsi, nous allons prioriser pour la réoptimisation les patients qui sont affectés aux mauvais soignants et/ou aux mauvais jours.ment de notre méthode arborescente, qui privilégie lors de la réoptimisation les patients mal assignés, c'est-à-dire ceux affectés à des soignants inadaptés ou à des jours non optimaux.

Pour évaluer la qualité des solutions obtenues, nous combinons deux écarts :

- Le gap relatif de coût de routing entre la solution courante et la solution optimale,
- La proportion de patients mal assignés dans la solution courante.

Nous définissons ainsi la mesure d'écart globale à la fin de chaque simulation par la formule suivante :

$$\text{Écart} = 100 \times \left(\frac{\text{coût_courant} - \text{coût_optimal}}{\text{coût_optimal}} + \frac{\text{nombre de patients mal assignés}}{\text{nombre total de patients actifs}} \right)$$

Cet écart permet de quantifier à la fois la dégradation dûe à un routage sous-optimal et la dégradation dûe à une affectation inadéquate. C'est l'indicateur principal que nous utiliserons pour évaluer les performances de notre méthode arborescente sur le problème complet. Tant que l'affectation est mauvaise, le deuxième terme a tendance à avoir une importance plus importante que le premier. Comme nous souhaitons en premier lieu avoir une bonne affectation pour ensuite permettre un bon routing, cela met mieux en évidence la part de cette mauvaise affectation.

La figure 5.11 montre que, cette fois encore, plus le nombre de changements autorisés est élevé, plus la solution réparée tend à se rapprocher de la solution optimale. Plus précisément :

— à partir de 12 changements autorisés, l'écart combiné le plus souvent inférieur à 50 ;
— à partir de 18 changements autorisés, l'écart combiné est presque systématiquement inférieur à 25.

Ici, nous avons testé nos résultats sur les instances de 4 soignants et 300 patients. Cela rend l'interprétation des résultats plus délicate, car si notre méthode gloutonne devient moins performante lorsque l'instance grandit, les performances de notre méthode de réparation seront affectées.

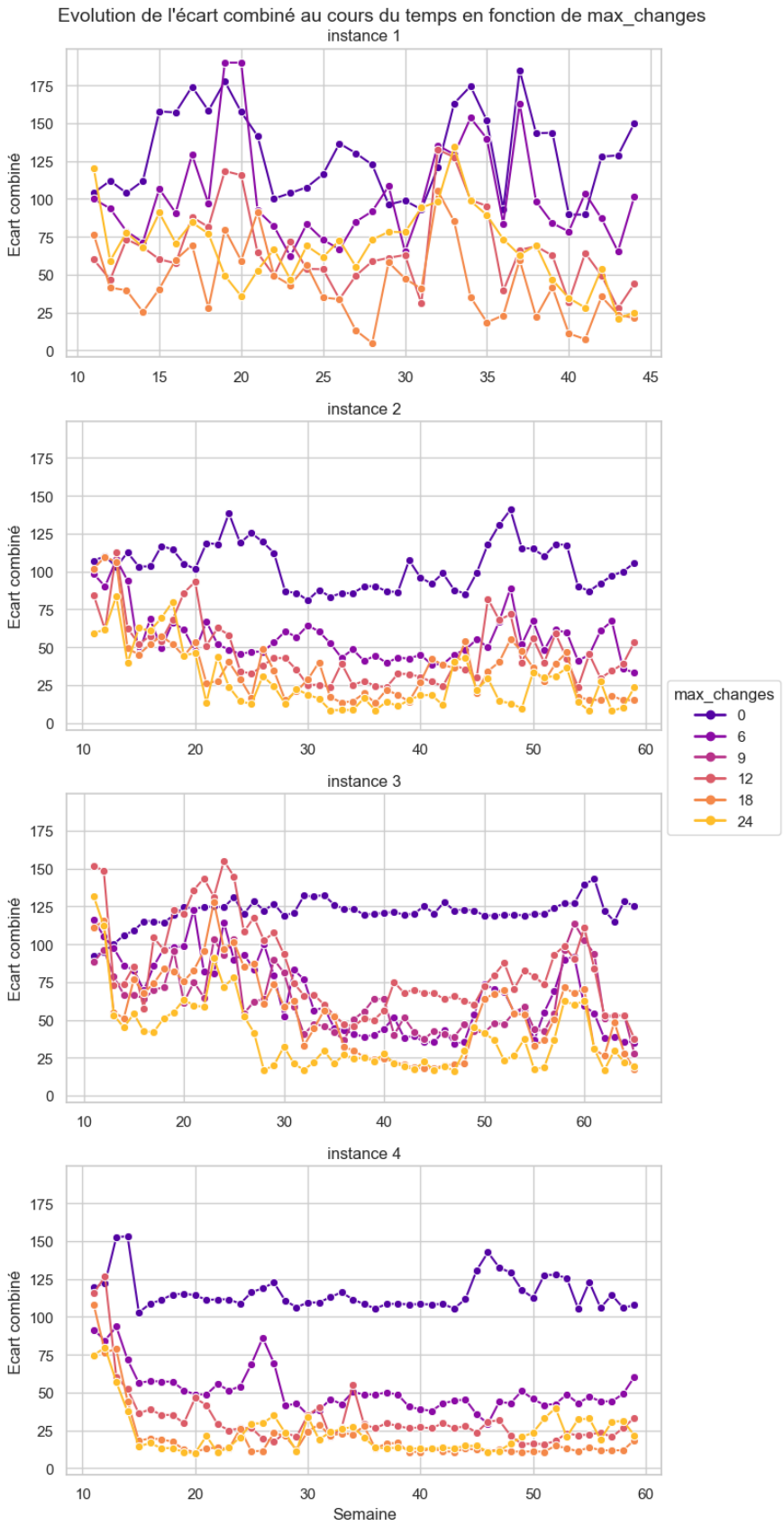

FIGURE 5.11 Évolution de l'écart combiné en fonction du paramètre max_changes

Importance du turnover

Ici encore, le turnover joue un rôle important dans les résultats. En effet, les résultats cités précédemment sont vérifiés pour toutes les instances à l'exception de la première. Cela nous confirme que lorsque le turnover est trop rapide, le processus de replanification ne peut pas être mené à bien car un certain nombre de patients quittent le système avant d'avoir été réassignés optimalement.

Évolution du nombre de patients bien affectés

Nous souhaitons mesurer à quel point notre approche priorisant la bonne affectation des patients a été efficace. La figure 5.12 montre l'évolution du pourcentage de patients mal affectés en fonction du paramètre $max_changes$.

Cette priorisation se montre particulièrement efficace. En effet, on observe clairement que plus le nombre de changements autorisés est élevé, plus le pourcentage de patients mal affectés est faible. Cependant, contrairement au cas précédent où l'affectation optimale pouvait être atteinte, celle-ci n'est jamais obtenue ici. Cette limitation s'explique probablement par la complexité additionnelle introduite par la phase de routage, qui contraint la méthode gloutonne à produire des affectations moins pertinentes. Atteindre l'optimal nécessiterait donc d'autoriser un volume de changements significativement plus important.

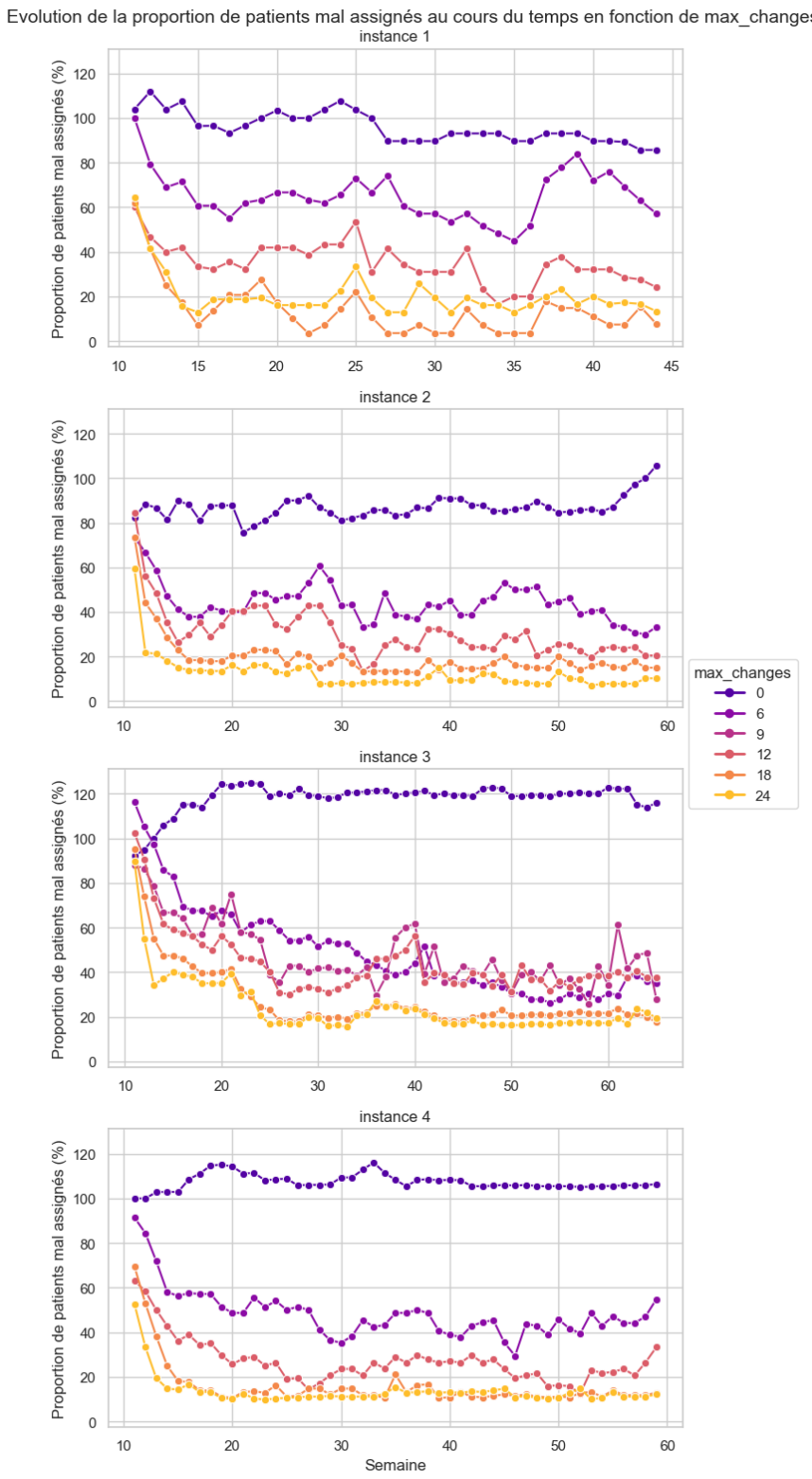

FIGURE 5.12 Évolution du pourcentage de patients mal affectés en fonction du paramètre max_changes

5.4.1 Décisions opérationnelles

Nous souhaitons de nouveau dégager des enseignements opérationnels à partir de nos résultats.

Ici encore, le turnover joue un rôle important. Pour des cas de figure où il est élevé, la méthode de réoptimisation ne semble pas adaptée : la planification change trop à chaque étape. Dans ce contexte, la solution optimale varie rapidement et la procédure n'a pas le temps de s'en rapprocher avant qu'elle ne soit de nouveau profondément modifiée. À partir d'un certain seuil, il devient pertinent d'utiliser une méthode de réoptimisation. Ce seuil est à déterminer expérimentalement selon le problème considéré. Dans notre cas, une durée de traitement moyenne de 8 semaines est suffisante pour que le processus de réoptimisation soit intéressant à mettre en place.

Ce cas met également en évidence que lorsque le problème est complexe, il ne sera pas possible d'atteindre la solution optimale sans autoriser un nombre de changements très important. Cependant, notre processus améliore tout de même significativement les performances du système.

En particulier, il est efficace pour corriger l'affectation des patients. Dans un contexte où les tournées pourraient être recalculées périodiquement sans modifier l'affectation (comme c'est parfois le cas dans la planification des soins à domicile) notre méthode présenterait un intérêt opérationnel marqué.

Recommandations synthétiques

Finalement, nous faisons les recommandations suivantes :

- pour un turnover trop élevé (durée moyenne de traitement inférieure à 8 semaines), le processus de réoptimisation n'a qu'un intérêt très limité et ne permettra pas d'améliorer significativement les résultats ;
- lorsque le turnover est suffisamment faible, autoriser 12 changements ou plus permet d'améliorer significativement les performances du système ;
- le nombre de changements nécessaires pour atteindre l'optimal semble se rapprocher du nombre de patients actifs dans le système. Dans ce cas, une réoptimisation progressive ne semble plus pertinente et on pourra s'intéresser à des méthodes de régénération complète du planning.

CHAPITRE 6 CONCLUSION

Dans ce travail, nous avons développé et évalué une méthode permettant de passer progressivement d'un planning existant à un planning optimal, tout en limitant les perturbations organisationnelles et en respectant les contraintes médicales et logistiques. L'enjeu central était de concilier deux impératifs souvent antagonistes : la performance de la planification et la stabilité nécessaire à la continuité des soins.

Pour répondre à cette problématique, nous avons proposé deux approches. La première, fondée sur un modèle de programmation linéaire en nombres entiers (PLNE), permet de déterminer à chaque itération la meilleure modification possible dans la limite d'un nombre de changements fixé. La seconde, basée sur une méthode de recherche arborescente, introduit une capacité d'anticipation particulièrement adaptée aux contextes dynamiques, en tenant compte de l'impact futur des décisions. Ce faisant, nous avons formalisé un cadre général de réoptimisation progressive applicable à de nombreux contextes.

L'application de ces méthodes au problème de planification des soins à domicile a permis d'évaluer leurs performances dans des contextes statiques, dynamiques, puis complets (avec routage). Les résultats montrent que :

- dans le cas statique, la méthode atteint systématiquement la solution optimale, le nombre d'itérations nécessaires étant directement lié à l'écart initial et au nombre de changements autorisés ;
- dans les contextes dynamiques, la méthode arborescente surpasse nettement la version myope et parvient à réduire significativement l'écart à l'optimal, particulièrement lorsque le turnover des patients est faible à moyen ;
- dans le problème complet, bien que l'optimal ne soit pas toujours atteint, la réoptimisation progressive améliore nettement la qualité des affectations et, indirectement, celle des tournées.

De plus, nous donnons des recommandations chiffrées pour guider les décideurs dans le choix des paramètres de réoptimisation.

Nous avons montré qu'une réoptimisation progressive constitue une approche pertinente pour améliorer la planification dans le domaine médical, tout en préservant la stabilité indispensable au maintien de la qualité des soins.

Plusieurs perspectives de recherche peuvent prolonger ce travail. D'abord, l'intégration de contraintes supplémentaires liées par exemple aux préférences du personnel permettrait de se rapprocher d'un cas réel. Ensuite, l'exploration d'approches hybrides combinant réoptimisation progressive et recalcul complet périodique pourrait améliorer la performance dans les contextes à forte variabilité. Enfin, appliquer cette méthode sur un cas réel du secteur médical permettrait de valider nos recommandations théoriques.

RÉFÉRENCES

[1] Organisation mondiale de la Santé, “Vieillissement et santé,” 2024, consulté en août 2025. [En ligne]. Disponible : https://www.who.int/fr/news-room/fact-sheets/detail/ageing-and-health

[2] N. Genet *et al.*, “Home care across Europe : current structure and future challenges,” dans *Home care across Europe : current structure and future challenges*, 2012. [En ligne]. Disponible : https://iris.who.int/handle/10665/327948

[3] Z. A. Abdalkareem *et al.*, “Healthcare scheduling in optimization context : a review,” *Health and Technology*, vol. 11, n°. 3, p. 445–469, 2021. [En ligne]. Disponible : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8035616/

[4] C.-W. Tsai *et al.*, “Metaheuristic Algorithms for Healthcare : Open Issues and Challenges,” *Computers & Electrical Engineering*, vol. 53, p. 421–434, juill. 2016. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0045790616300532

[5] A. Rajagopal *et al.*, “Machine learning operations in health care : A scoping review,” *Mayo Clinic Proceedings : Digital Health*, vol. 2, n°. 3, p. 421–437, 2024. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S2949761224000701

[6] C. Yu, J. Liu et S. Nemati, “Reinforcement Learning in Healthcare : A Survey,” avr. 2020. [En ligne]. Disponible : http://arxiv.org/abs/1908.08796

[7] C. Van Walraven *et al.*, “The association between continuity of care and outcomes : a systematic and critical review,” *Journal of Evaluation in Clinical Practice*, vol. 16, n°. 5, p. 947–956, 2010. [En ligne]. Disponible : https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2753.2009.01235.x

[8] J. M. Reckrey *et al.*, “Home care worker continuity in home-based long-term care : Associated factors and relationships with client health and well-being,” *Innovation in Aging*, vol. 8, n°. 3, p. igae024, 02 2024. [En ligne]. Disponible : https://doi.org/10.1093/geroni/igae024

[9] D. Ouelhadj et S. Petrovic, “A survey of dynamic scheduling in manufacturing systems,” *J Sched*, 2009.

[10] G. E. Vieira, J. W. Herrmann et E. Lin, “Rescheduling Manufacturing Systems : A Framework of Strategies, Policies, and Methods,” *Journal of Scheduling*, vol. 6, n°. 1, p. 39–62, janv. 2003. [En ligne]. Disponible : https://doi.org/10.1023/A:1022235519958

[11] S. Holguin Jimenez, W. Trabelsi et C. Sauvey, “Multi-objective production rescheduling : A systematic literature review,” *Mathematics*, vol. 12, n°. 20, p. 3176, 2024.

[12] A. Pfeiffer, B. Kádár et L. Monostori, “Stability-oriented evaluation of rescheduling strategies, by using simulation,” *Computers in Industry*, vol. 58, n°. 7, p. 630–643, 2007.

[13] S. D. Wu, R. H. Storer et C. Pei-Chann, “One-machine rescheduling heuristics with efficiency and stability as criteria,” *Computers & Operations Research*, vol. 20, n°. 1, p. 1–14, 1993.

[14] R. Leus et W. Herroelen, “Scheduling for stability in single-machine production systems,” *Journal of Scheduling*, vol. 10, n°. 3, p. 223–235, juin 2007. [En ligne]. Disponible : http://link.springer.com/10.1007/s10951-007-0014-z

[15] H. El Sakkout, T. Richards et M. Wallace, “Minimal perturbation in dynamic scheduling,” dans *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98). John Wiley & Sons*, 1998.

[16] P. Valledor *et al.*, “Solving rescheduling problems in dynamic permutation flow shop environments with multiple objectives using the hybrid dynamic non-dominated sorting genetic ii algorithm,” *Mathematics*, vol. 10, n°. 14, p. 2395, 2022.

[17] K. Sato, K. Tamura et N. Tomii, “A mip-based timetable rescheduling formulation and algorithm minimizing further inconvenience to passengers,” *Journal of Rail Transport Planning & Management*, vol. 3, n°. 3, p. 38–53, 2013.

[18] H. Soltani, S. M. J. M. Al-e *et al.*, “Robust maritime disruption management with a combination of speedup, skip, and port swap strategies,” *Transportation Research Part C : Emerging Technologies*, vol. 153, p. 104146, 2023.

[19] S. Nickel, M. Schröder et J. Steeg, “Mid-term and short-term planning support for home health care services,” *European Journal of Operational Research*, vol. 219, n°. 3, p. 574–587, 2012.

[20] M. Erdem et S. Bulkan, “A two-stage solution approach for the large-scale home healthcare routeing and scheduling problem,” *South African Journal of Industrial Engineering*, vol. 28, n°. 4, p. 133–149, nov. 2017, publisher : South African Institute of Industrial Engineers (SAIIE). [En ligne]. Disponible : https://journals.co.za/doi/abs/10.7166/28-4-1754

[21] J. Decerle *et al.*, “A two-phases matheuristic for the home care routing and scheduling problem,” *IFAC-PapersOnLine*, vol. 49, n°. 12, janv. 2016. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S240589631631062X

[22] A. F. Kummer *et al.*, “A biased random-key genetic algorithm for the home health care problem,” *International Transactions in Operational Research*, vol. 31, n°. 3, p.

1859–1889, 2024.

[23] S. Shahnejat-Bushehri *et al.*, “A robust home health care routing-scheduling problem with temporal dependencies under uncertainty,” *Expert Systems with Applications*, vol. 182, p. 115209, 2021.

[24] Z. Dai, Z. Zhang et M. Chen, “The home health care location-routing problem with a mixed fleet and battery swapping stations using a competitive simulated annealing algorithm,” *Expert Systems with Applications*, vol. 228, p. 120374, 2023.

[25] M. Gaudioso et G. Paletta, “A heuristic for the periodic vehicle routing problem,” *Transportation Science*, vol. 26, n°. 2, p. 86–92, 1992.

[26] F. Grenouilleau, *Méthodes exactes et approchées pour le problème de planification des soins à domicile.* Ecole Polytechnique, Montreal (Canada), 2020.

[27] Y. Li, T. Xiang et W. Y. Szeto, “Home health care routing and scheduling problem with the consideration of outpatient services,” *Transportation Research Part E : Logistics and Transportation Review*, vol. 152, p. 102420, 2021.

[28] C. Groër, B. Golden et E. Wasil, “The consistent vehicle routing problem,” *Manufacturing & service operations management*, vol. 11, n°. 4, p. 630–643, 2009.

[29] I. Rodríguez-Martín, J.-J. Salazar-González et H. Yaman, “The periodic vehicle routing problem with driver consistency,” *European Journal of Operational Research*, vol. 273, n°. 2, p. 575–584, mars 2019. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0377221718307276

[30] C. Fikar et P. Hirsch, “Home health care routing and scheduling : A review,” *Computers & Operations Research*, vol. 77, p. 86–95, 2017.

[31] M. I. Gomes et T. R. P. Ramos, “Modelling and (re-) planning periodic home social care services with loyalty and non-loyalty features,” *European Journal of Operational Research*, vol. 277, n°. 1, p. 284–299, 2019.

[32] N. Lahrichi, E. Lanzarone et S. Yalçındağ, “A first route second assign decomposition to enforce continuity of care in home health care,” *Expert Systems with Applications*, vol. 193, p. 116442, 2022.

[33] G. Du, X. Liang et C. Sun, “Scheduling optimization of home health care service considering patients’ priorities and time windows,” *Sustainability*, vol. 9, n°. 2, p. 253, 2017.

[34] R. Liu, B. Yuan et Z. Jiang, “The large-scale periodic home health care server assignment problem : A region-partition-based algorithm,” *IEEE Transactions on Automation Science and Engineering*, vol. 17, n°. 3, p. 1543–1554, 2020.

[35] M. Chaieb et D. B. Sassi, “Measuring and evaluating the home health care scheduling problem with simultaneous pick-up and delivery with time window using a tabu search metaheuristic solution,” *Applied Soft Computing*, vol. 113, p. 107957, 2021.

[36] E. Martin *et al.*, “Iacs-hcsp : Improved ant colony optimization for large-scale home care scheduling problems,” *Expert systems with applications*, vol. 142, p. 112994, 2020.

[37] T. Zhang *et al.*, “Home health care routing and scheduling in densely populated communities considering complex human behaviours,” *Computers & Industrial Engineering*, vol. 182, p. 109332, 2023.

[38] Y. Fu *et al.*, “A review on metaheuristics for solving home health care routing and scheduling problems,” *Applied Soft Computing*, p. 113560, 2025.

[39] C. Martinez, M.-L. Espinouse et M. Di Mascolo, “An exact two-phase approach to re-optimize tours in home care planning,” *Computers & Operations Research*, vol. 161, p. 106408, 2024.

[40] C. B. Browne *et al.*, “A survey of monte carlo tree search methods,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, n°. 1, p. 1–43, 2012.

[41] T. Vidal *et al.*, “A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems,” *Operations Research*, vol. 60, n°. 3, p. 611–624, juin 2012, publisher : INFORMS. [En ligne]. Disponible : https://pubsonline.informs.org/doi/10.1287/opre.1120.1048

[42] T. Vidal, “Hybrid genetic search for the CVRP : Open-source implementation and SWAP* neighborhood,” *Computers & Operations Research*, vol. 140, p. 105643, avr. 2022. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S030505482100349X

[43] M. M. Solomon, “Algorithms for the vehicle routing and scheduling problems with time window constraints,” *Operations research*, vol. 35, n°. 2, p. 254–265, 1987.