Titre: Title:	Résolution en temps réel du problème de la prise de rendez-vous dans des tournées de véhicules par apprentissage automatique
Auteur: Author:	Théau Lepouttre
Date:	2025
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Lepouttre, T. (2025). Résolution en temps réel du problème de la prise de rendez-vous dans des tournées de véhicules par apprentissage automatique [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/64755/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/64755/
Directeurs de recherche: Advisors:	Quentin Cappart
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Résolution en temps réel du problème de la prise de rendez-vous dans des tournées de véhicules par apprentissage automatique

THÉAU LEPOUTTRE
Département de génie informatique et génie logiciel

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

Avril 2025

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Résolution en temps réel du problème de la prise de rendez-vous dans des tournées de véhicules par apprentissage automatique

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

Amine MHEDHBI, président
Quentin CAPPART, membre et directeur de recherche
Martin TRÉPANIER, membre

DÉDICACE

À mes parents et grands-parents,
Je vous aime !

REMERCIEMENTS

Je tiens à remercier mon directeur de recherche et professeur, Quentin Cappart, pour m'avoir proposé ce projet ainsi que pour son soutien, son écoute et sa disponibilité sans faille. Je lui suis reconnaissant pour l'opportunité offerte à travers ce mémoire, ainsi que pour les charges de cours et de laboratoire que j'ai eu l'occasion d'effectuer. Merci à Fastercom pour le partenariat enrichissant et les réunions hebdomadaires constructives, ainsi qu'à Compute Canada et au CIRRELT pour la mise à disposition de leurs clusters de calcul, sans lesquels l'obtention des résultats de cette recherche aurait été bien plus complexe. Je remercie également Martin Trépanier et Amine Mhedhbi d'avoir accepté de faire partie de mon jury de maîtrise.

J'adresse mes remerciements aux différents professeurs rencontrés au cours de mon parcours académique, avec une mention spéciale à Marianne Renaut. Merci également à l'UCLouvain pour avoir rendu possible ce double diplôme, qui m'a permis d'acquérir de nombreuses connaissances et d'affiner mes projets pour l'avenir.

Je souhaite également remercier les nombreux amis rencontrés durant ces deux années passées au Canada. Je pense particulièrement à Romane, Gilles, Rebecca, Marianna, Flora et Axel, pour ne citer qu'eux, ainsi qu'à toutes les autres personnes, au Canada, en Belgique ou ailleurs, qui se reconnaîtront.

Enfin, un immense merci à mes parents, sans qui ce double diplôme et l'ensemble de mon parcours académique auraient été bien différents. Merci pour leur confiance et leur soutien tout au long de mes études.

RÉSUMÉ

La construction de planning en temps réel pour des tournées de véhicules est une problématique présente dans de multiples domaines industriels. Ce mémoire se focalise sur le secteur des services de soins à domicile, où le planning hebdomadaire des tournées du personnel soignant est construit dynamiquement en ajoutant les patients demandant à être servis un à un.

Ce problème est activement considéré dans la littérature, et notre travail se focalise sur l'insertion de patients, nécessitant un algorithme pouvant délivrer une réponse en temps réel. Dans le cadre de l'optimisation dynamique des tournées de véhicules pour l'insertion de patients, ce mémoire propose une méthode innovante basée sur l'apprentissage automatique afin d'éviter la résolution répétée d'instances du problème de tournées et de planification des soins à domicile (*Home Health Care Routing and Scheduling Problem*, HHCRSP). Le problème est d'abord formulé comme un programme linéaire représentant une variante du problème de tournées de véhicules multi-dépôt avec capacités et fenêtres de temps (*Multi-Depot Capacitated Vehicle Routing Problem with Time Windows*, MDCVRPTW). Le problème intègre une double contrainte de compatibilité, les fenêtres de temps pour les patients et soignants, les stocks de pansements et de seringues, en plus de points de départ distincts pour chaque membre du personnel soignant, assurant ainsi la cohérence des plannings créés par l'algorithme.

Pour maintenir l'expressivité nécessaire à la résolution du problème, nous représentons les solutions de ce dernier en les modélisant sous forme de graphes hétérogènes. Sur cette base, un modèle d'apprentissage en cascade combinant un réseau de neurones pour graphes (*Graph Neural Network*, GNN) et un perceptron multi-couches (*Multi-Layer Perceptron*, MLP) est développé pour encoder et décoder les informations issues des graphes et prédire la faisabilité des insertions.

Des expériences sont réalisées sur des données réelles et historiques venant d'une entreprise partenaire, en simulant un cadre d'opération réel. Les résultats indiquent que la méthode développée permet de réduire de moitié le nombre de requêtes faites à un solveur externe et d'assurer un temps de réponse inférieur à 20 secondes, tout en limitant la perte de patients servis à 1.79 %. Une analyse critique des résultats propose des possibilités d'amélioration pour des travaux futurs portant sur l'enrichissement et la diversification des données d'entraînement, l'ajustement des critères de sélection des insertions et l'exploration de modèles hybrides afin d'accroître la robustesse et la généralisation de la méthode face à des scénarios plus complexes.

ABSTRACT

The construction of real-time schedules for vehicle rounds is a problem that is present in many industrial domains. This thesis focuses on the homecare sector, where the weekly schedule of nursing staff rounds is dynamically constructed by adding patients requesting service one by one.

This problem is actively considered in the literature, and our work focuses on patient insertion, requiring an algorithm that can deliver a real-time response. In the context of dynamic optimization of vehicle routes for patient insertion, this dissertation proposes an innovative machine learning-based method to avoid repeated solving of instances of the Home Health Care Routing and Scheduling Problem (HHCRSP). The problem is first formulated as a linear program representing a variant of the Multi-Depot Capacitated Vehicle Routing Problem with Time Windows (MDCVRPTW). The problem incorporates a double compatibility constraint, time windows for patients and caregivers, and stocks of dressings and syringes, in addition to distinct starting points for each member of the nursing staff, thus ensuring the consistency of the schedules created by the algorithm.

To maintain the expressiveness required to solve the problem, we represent its solutions by modeling them as heterogeneous graphs. On this basis, a cascade learning model combining a graph neural network (GNN) and a multi-layer perceptron (MLP) is developed to encode and decode graph information and predict the feasibility of insertions.

Experiments are carried out on real and historical data from a partner company, simulating a real operating environment. The results show that the method developed can halve the number of requests made to an external solver and ensure a response time of less than 20 seconds, while limiting the loss of patients served to 1.79 %. A critical analysis of the results suggests possible improvements for future work on enriching and diversifying training data, adjusting insertion selection criteria and exploring hybrid models to increase the robustness and generalizability of the method in the face of more complex scenarios.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

TSP	Problème du voyageur de commerce (*Travelling salesman problem*)
VRP	Problème de tournées de véhicules (*Vehicle Routing Problem*)
CVRP	VRP avec contrainte de demande et de capacité (*Capacitated VRP*)
VRPTW	VRP avec contrainte de fenêtre de temps (*VRP with Time Window*)
MDVRP	VRP multi-dépôt (*Multi Depot VRP*)
HHCRSP	Problème d'acheminement et de programmation des soins à domicile (*Home health care routing and scheduling problem*)
MDVRPTW	VRP multi-dépôt avec de fenêtre de temps (*Multi Depot VRP with Time Windows*)
MDCVRPTW	VRP multi-dépôt avec contrainte de capacité et de fenêtre de temps (*Multi Depot Capacitated VRP with Time Windows*)
BB	Branch & Bound
BP	Branch & Price
GC	Génération de colonnes
BC	Branch & Cut
ML	Apprentissage machine (*Machine learning*)
MLP	Perceptron multicouche (*Multilayer perceptron*)
RL	Apprentissage par renforcement (*Reinforcement learning*)
DRL	Apprentissage par renforcement profond (*Deep Reinforcement learning*)
LP	Programmation linéaire
CP	Programmation par contraintes
MILP	Programmation linéaire en nombres entiers mixtes
MIQP	Programmation quadratique en nombres entiers mixtes
GNN	Réseau de neurones pour graphe (*Graph Neural Network*)
GAT	Réseaux de neurones attentionnels pour graphes (*Graph Attention Networks*)
GCN	Réseau convolutionnel pour graphe (*Graph Convolutional Network*)
DGL	Librairie python pour le développement de GNN (*Deep Graph Library*)
BS	Recherche en faisceau (*Beam Search*)
PSO	Optimisation par essaims de particules

LISTE DES NOTATIONS

Formulation mathématique du problème P

n	Nombre de patients à servir
m	Nombre d'infirmières disponibles
$A = \{1, 2, \ldots, n\}$	Ensemble des patients
$R = \{1, 2, \ldots, m\}$	Ensemble des infirmières
$V = R \cup A$	Ensemble de tous les nœuds (ordonné avec les infirmières, puis les patients)
$\mathcal{F}$	Ensemble des compétences possibles (chaque compétence $\in \mathbb{N}$)
$\mathcal{S}$	Ensemble des affinités possibles (chaque affinité $\in \mathbb{N}$)
$t_{i,j}$	Temps de trajet entre les nœuds $i, j \in V$ (en minutes)
C_k^1, C_k^2	Capacités en pansements et en seringues pour l'infirmière $k \in R$
d_i^1, d_i^2	Demandes en pansements et en seringues du patient $i \in A$
l_i	Temps de service au nœud $i \in V$ (0 pour les noeuds des infirmières)
$[a_i, b_i]$	Fenêtre temporelle du nœud $i \in V$
$[a_k, b_k]$	Fenêtre temporelle de l'infirmière $k \in R$
B	Pénalité de non-service d'un patient
$F_i \subseteq \mathcal{F}$	Compétences requises par le patient $i \in A$
$S_i \subseteq \mathcal{S}$	Affinités requises par le patient $i \in A$
$F_k \subseteq \mathcal{F}$	Compétences disponibles pour l'infirmière $k \in R$
$S_k \subseteq \mathcal{S}$	Secteurs possibles pour l'infirmière $k \in R$
$\rho_{i,k}$	Paramètre binaire de compatibilité entre le nœud i et l'infirmière k
M	Paramètre constant suffisamment grand (Big-M)
$x_{i,j,k}$	Variable de décision binaire : 1 si l'arc (i, j) est emprunté par l'infirmière k, 0 sinon
$y_{i,k}$	Variable de décision binaire : 1 si l'infirmière k dessert le patient i, 0 sinon
z_i	Variable de décision binaire : 1 si le patient i n'est pas servi, 0 sinon
s_i	Temps de début du service du nœud $i \in V$

Cadre opérationnel et fonction objectif

objective	Fonction objectif évaluant la qualité d'une insertion
T_i	Temps de travail individuel de l'infirmière i

$T_{\max}$	Temps de travail total maximal possible
$\sigma(T_i)$	Écart-type des temps de travail individuels des infirmières
$\text{écart}_{\max}$	Valeur maximale de l'écart-type, définie par $\frac{T_{\max}}{2}$
a	Poids associé à la minimisation du temps total de travail des infirmières ($a = 0.999$)
b	Poids associé à l'équilibre du temps de travail entre infirmières ($b = 0.001$)

Modélisation graphique et création de données

$\mathbf{p}$	Une instance du problème $\mathbf{P}$
$G_{\mathbf{p}} = (V, E)$	Graphe hétérogène représentant la solution d'une instance $\mathbf{p}$
V	Ensemble des nœuds du graphe
E	Ensemble des arêtes du graphe
A	Ensemble des patients
R	Ensemble des infirmières
T	Ensemble des étapes temporelles discrètes
E_{A-A}	Ensemble des arêtes entre patients
E_{R-R}	Ensemble des arêtes entre infirmières
E_{T-T}	Ensemble des arêtes entre étapes temporelles
E_{A-R}, E_{R-A}	Ensembles des arêtes entre patients et infirmières
E_{A-T}, E_{T-A}	Ensembles des arêtes entre patients et étapes temporelles
E_{R-T}, E_{T-R}	Ensembles des arêtes entre infirmières et étapes temporelles
$h_v \in \mathbb{R}^{d_1}$	Vecteur des caractéristiques associées au nœud $v \in V$
$h_e \in \mathbb{R}^{d_2}$	Vecteur des caractéristiques associées à l'arête $e \in E$
h_{v-i}	i-ème caractéristique du nœud v
`ins`	Insertion possible d'un patient dans un planning optimisé
$V_{G_{\mathbf{p},\texttt{ins}}}$	Ensemble des nœuds du graphe d'insertion
$E_{G_{\mathbf{p},\texttt{ins}}}$	Ensemble des arêtes du graphe d'insertion
$Q_{\texttt{ins}}$	Booléen indiquant si l'insertion `ins` est faisable

Modélisation du modèle d'apprentissage automatique

K	Nombre de têtes d'attention du GNN
$\sum_{p=1}^{3} n_{f_p}$	Nombre total de caractéristiques latentes associées aux trois types de nœuds
$\mathbf{\Theta}$	Tenseur de poids apprenables du GNN

$\|$	Opérateur de concaténation utilisé pour fusionner les représentations latentes
$\alpha_{j,r,i}^{k}$	Poids d'attention pour une relation r et une tête k entre les nœuds j et i
W_k, b_k	Poids et biais des couches du MLP
BN_k	Normalisation de lot (batch norm) pour la couche k du MLP
σ	Fonction d'activation appliquée dans le MLP
a_k, z_k	Sorties latentes avant et après activation pour la couche k
$\sigma_s(z_i)$	Fonction sigmoïde appliquée aux logits z_i
w^+, w^-	Poids attribués aux échantillons positifs et négatifs dans la fonction de perte
$\mathcal{L}$	Fonction de perte d'entropie croisée binaire avec logit
w_t	Poids d'un neurone à l'itération t lors de l'optimisation
η	Taux d'apprentissage utilisé dans la mise à jour des poids
λ	Paramètre de décroissance des poids pour la régularisation
d_k	Masque binaire utilisé pour le *dropout* dans la couche k
$\odot$	Produit d'Hadamard utilisé dans l'application du *dropout*

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Le problème des tournées de véhicules (*Vehicle Routing Problem*, VRP) est un problème d'optimisation combinatoire visant à déterminer les itinéraires optimaux pour une flotte de véhicules chargée de desservir un ensemble de clients. Depuis sa première formalisation en 1959 par Dantzig et Ramser [1], le VRP a fait l'objet de nombreuses recherches [2, 3], aboutissant à la création de diverses variantes, adaptées à des contextes spécifiques, telles que le VRP avec contraintes de capacité [1] (CVRP) ou le VRP avec fenêtres de temps (VRPTW) [4].

Cette problématique se retrouve dans de nombreux secteurs, tels que, par exemple, la logistique [5] ou la collecte de déchets [6]. Dans le secteur des soins à domicile, de nombreuses entreprises doivent planifier efficacement les déplacements des membres du personnel soignant afin d'assurer des interventions successives chez différents patients. Lorsqu'une entreprise souhaite ajouter le rendez-vous d'un nouveau patient dans un planning basé sur les tournées de véhicules, il est essentiel de trouver l'insertion optimale pour ce patient. Pour ce faire, il est nécessaire de tester chaque insertion possible afin de déterminer la meilleure.

Le partenaire industriel de ce projet, Fastercom, distribue un algorithme auprès d'entreprises, leur permettant de prévoir leur prochaine semaine d'opération au moyen d'un outil hebdomadaire. Dans ce travail, l'algorithme distribué sert à créer des plannings pour le service de soins à domicile de patients par des soignants véhiculés.

Ce mémoire s'appuie sur des données réelles historiques provenant du secteur des soins à domicile, fournies dans le cadre d'une collaboration avec un partenaire industriel. Fastercom est spécialisé dans la logistique du transport pour des secteurs critiques, tels que les soins médicaux et les services aux entreprises. Ce partenariat permet d'accéder à un ensemble de données réelles et garantit que le cadre proposé est conforme aux exigences pratiques dans des applications concrètes. Le contexte industriel souligne l'importance de l'extensibilité, de l'interprétabilité et de la fiabilité opérationnelle, éléments placés au cœur de la conception de l'approche choisie.

1.1 Contexte du problème

Le problème que nous abordons dans cette recherche est le problème d'acheminement et de programmation des soins à domicile (*Home Health Care Routing and Scheduling Problem*, HHCRSP) [7]. Ce type de problème représente, plus généralement, une variante du VRP avec

des fenêtres temporelles, plusieurs dépôts et des contraintes spécifiques au problème étudié. Sans ces contraintes spécifiques, le problème à résoudre est nommé problème de tournées de véhicules multi-dépôts avec contraintes de fenêtres de temps (MDVRPTW). Dans notre cas, les contraintes spécifiques sont des contraintes de capacité et de compatibilité. Les contraintes de capacité font du problème une instance du problème de tournées de véhicules multi-dépôts avec contraintes de fenêtres de temps et de capacité (MDCVRPTW).

La variante classique du HHCRSP nécessite une résolution statique où l'ensemble des patients est connu à l'avance. Il est important de noter que le problème décrit dans cette recherche va au-delà de la simple résolution d'une instance de HHCRSP. Le planning du personnel soignant est créé dynamiquement en ajoutant un à un les patients demandant à être servis. Ce planning est élaboré durant la semaine précédant le départ des membres du personnel soignant pour leurs tournées hebdomadaires. À chaque nouvelle demande d'insertion d'un patient, l'algorithme doit résoudre plusieurs instances du HHCRSP afin de déterminer s'il peut être inséré dans le planning, et à quel moment. Le cadre opérationnel est donc semi-dynamique : la phase de création du planning est effectuée dynamiquement en amont pour chaque ajout de patient, mais aucun changement n'est effectué lors des tournées de véhicules.

Un planning est constitué de l'ensemble des patients déjà ajoutés et des insertions qui leur ont été communiqués afin d'obtenir la route optimale empruntée par les membres du personnel soignant pour effectuer les services. Une insertion constitue une plage horaire promise pour le service du patient. L'algorithme utilisé par Fastercom fonctionne dans un cadre continu en temps réel, où des patients peuvent demander à être ajoutés au planning à tout moment lors de sa phase de création. L'algorithme doit alors résoudre des instances du HHCRSP à chaque nouveau patient à insérer. Une demande d'ajout se fait généralement par téléphone, et il est crucial de fournir une réponse au patient le plus rapidement possible, par exemple dans un délai de quinze à vingt secondes. Le cadre opérationnel s'inscrit donc dans un contexte continu en temps réel, illustré à la figure 1.1.

Lorsqu'une entreprise cherche à intégrer le rendez-vous d'un nouveau patient dans un planning basé sur les tournées de véhicules, il est essentiel d'identifier l'insertion optimale le plus rapidement possible afin de minimiser le temps d'attente du patient au téléphone. Cependant, toutes les insertions possibles doivent être testées afin de sélectionner la meilleure d'entre elles. L'algorithme de Fastercom a pour but d'indiquer si l'insertion d'un patient dans le planning actuel est possible et, le cas échéant, à quel moment. Il ajuste ensuite le planning en conséquence et reste en attente jusqu'au prochain appel téléphonique. Le problème est complexe pour trois raisons. Premièrement, il faut résoudre une multitude de nouvelles instances du problème à chaque ajout d'un patient pour tester chaque insertion possible

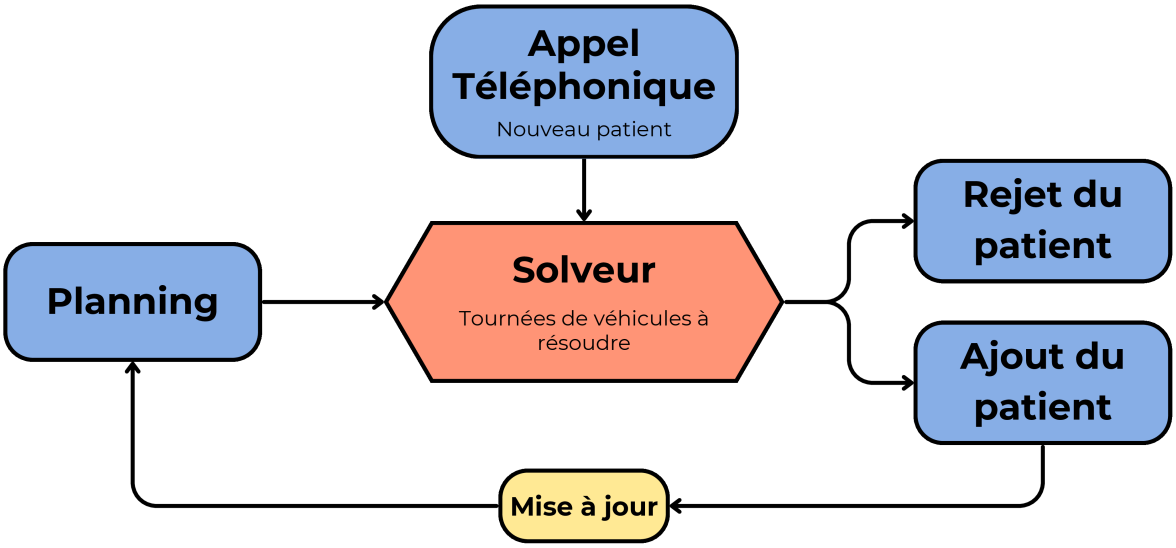


FIGURE 1.1 Cadre continu de résolution

dans le planning actuel. Deuxièmement, de nouveaux patients peuvent demander un ajout à tout moment, nécessitant de multiples résolutions. Troisièmement, l'insertion d'un patient consiste en une promesse de service durant un intervalle de temps. Cette nouvelle promesse et toutes celles effectuées aux autres patients constituent des contraintes supplémentaires pour la future re-résolution du problème. Il est donc nécessaire de résoudre plusieurs instances du problème de tournées de véhicules rapidement afin de trouver la meilleure insertion, et ce, à de nombreuses reprises, étant donné l'ajout continu de nouveaux patients dans un planning en constante mise à jour. Nous définissons dans la section 3.3 la notion de « meilleure insertion ».

1.2 Cadre de la recherche

Notre recherche s'inscrit dans le cadre spécifique de notre partenaire Fastercom, qui diffère des cadres théoriques du VRP classique. Dans ce travail, le problème étudié est une variante du MDCVRPTW [8–10], avec une contrainte de double compatibilité entre patients et soignants. Ce problème est décomposé en cinq parties :

— **Les demandes :** Le problème présente deux contraintes liées aux demandes et aux capacités. Dans notre contexte, les patients ont deux types de demandes : en *pansement* et en *seringue*. Chaque demande doit être satisfaite par le membre du personnel soignant qui la prend en charge, sans que la somme des demandes servies ne dépasse le stock total pour chaque type de demande.

— **Les compatibilités :** Le problème présente une contrainte de double compatibilité. La compatibilité est déterminée par deux types de propriétés appelées « compétences » et « affinités ». La compatibilité de *compétence* fait référence aux qualifications requises pour soigner le patient, tandis que l'*affinité* concerne les préférences du patient pour son service, comme la langue parlée. La compatibilité est définie ainsi : le service d'un patient est possible si, et seulement si, l'ensemble des compétences nécessaires au soin du patient est inclus dans celui du membre du personnel soignant le servant, et que ce soit le cas pour les affinités du patient également.
— **Les points de départ de la tournée :** Chaque membre du personnel soignant a un point de départ associé, correspondant à son domicile ou à son hôpital de travail, par exemple.
— **La cohérence des routes :** Le problème intègre toutes les contraintes associées à la forme théorique connue du problème de tournées de véhicules [1].
— **Fenêtre temporelle :** Le problème présente une contrainte de fenêtre temporelle. Chaque membre du personnel soignant et chaque patient ne peut travailler ou être servi que durant une certaine période appelée fenêtre de temps.

1.3 Objectif de la recherche

La difficulté rencontrée par l'algorithme de Fastercom, qui constitue un défi global dans la résolution des tournées de véhicules, réside dans le temps nécessaire pour résoudre une instance du problème, celui-ci étant *NP-difficile* [11, 12], et, par conséquent, pour répondre au patient. Lorsqu'un nouveau patient se manifeste, il est nécessaire de résoudre à nouveau le problème pour chaque insertion possible, en tenant compte de ses contraintes spécifiques, afin de déterminer si, et quand, ce patient doit être servi.

En plus de la complexité combinatoire du problème, les techniques de résolution et les avancées scientifiques sont rarement transférables entre les différentes variantes du VRP. Ainsi, chaque variante impose une conception d'algorithmes spécifiquement adaptée à ces contraintes particulières, ce qui limite fortement la généralisation des résultats obtenus.

Les méthodes exactes comme le *Branch and Bound* (BB) [13] ou le *Branch and Price* (BP) via génération de colonnes (GC) [14] sont souvent envisagées, mais leur applicabilité est généralement limitée aux instances de petite taille, en raison de la complexité computationnelle de ce genre de méthodes. Actuellement, l'algorithme de Fastercom utilise un solveur basé sur la recherche locale [15, 16], une méthode incomplète. C'est l'une des approches classiques pour la résolution de ce genre de problème. Cependant, même si cette méthode fonctionne bien en pratique, elle ne respecte pas la contrainte de réponse en temps réel, la qualité de la

solution trouvée dépendant souvent du temps alloué à la recherche.

C'est dans ce cadre que se situe notre recherche, motivant l'utilisation de l'apprentissage automatique. Notre hypothèse de recherche est de remplacer les résolutions du problème par de l'apprentissage automatique en prédisant la faisabilité d'une insertion. On fournit les informations du problème et de l'insertion sous forme de graphe en entrée de notre modèle, et on infère une prédiction de sortie sur la faisabilité de l'insertion associée. Grâce à ce remplacement, notre modèle réduit le nombre de résolutions nécessaires pour répondre au patient et respecte la contrainte de temps réel introduite. Cette contrainte de temps réel sur le temps de réponse a été fixée à un maximum de vingt secondes, correspondant à un temps d'attente raisonnable en pratique pour un patient au téléphone.

L'objectif principal de ce travail est de démontrer l'intérêt des techniques d'apprentissage automatique pour permettre une prise de décision en temps réel lors d'ajouts séquentiels de patients pour la construction d'un planning basé sur les tournées de véhicules. Plus précisément, lorsqu'une nouvelle demande de patient est reçue, le modèle intègre les possibilités d'insertion en générant les multiples prédictions correspondantes. Pour chaque insertion, le modèle prédit sa faisabilité. Ensuite, sur la base de ses prédictions, le modèle sélectionne l'insertion optimale à communiquer au patient ou communique l'impossibilité de service.

Cette approche incrémentale permet au système de maintenir une efficacité élevée tout en s'adaptant dynamiquement aux nouvelles demandes de service, répondant ainsi aux exigences pratiques de la programmation en temps réel.

1.4 Plan du mémoire

Le chapitre 1 introduit la recherche et le contexte du problème à résoudre. Le chapitre 2 propose une revue de la littérature, partant des prémices de la création du problème de tournées de véhicules jusqu'aux avancées récentes. Il liste les différentes méthodes utilisées pour la résolution du VRP et de ses variantes, l'utilisation de l'apprentissage automatique pour la résolution de problèmes combinatoires, et enfin l'application de l'apprentissage automatique dans le cadre des tournées de véhicules. Le chapitre 3 décrit le problème avec une formulation mathématique et pose les bases pour la compréhension du cadre opérationnel ainsi que de l'approche proposée. Le chapitre 4 développe en profondeur la méthodologie utilisée. Le chapitre 5 présente les résultats obtenus et discute de leur pertinence par rapport aux objectifs de l'étude. Finalement, le chapitre 6 conclut le mémoire.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre présente une revue de littérature portant sur l'historique du problème de tournées de véhicules, ses méthodes de résolution, ainsi que l'impact des approches d'apprentissage sur ce type de problème. Notre recherche s'intéressant à l'insertion de nouveaux patients dans un planning de soins à domicile déjà optimisé, nous détaillons les méthodes historiques ainsi que les approches pertinentes pour la résolution du problème étudié. Puisque l'insertion doit être effectuée en temps réel, les méthodes utilisées doivent permettre de résoudre l'instance associée en quelques secondes. Dans cette optique, nous nous intéressons aux réseaux de neurones pour graphes (*Graph Neural Network*, GNN), capables d'exploiter la structure relationnelle du problème pour estimer la faisabilité d'une insertion sans recourir à une résolution complète, tout en respectant les contraintes opérationnelles.

La section 2.1 présente les prémices du VRP ainsi que les méthodes classiques de résolution. La section 2.2 traite de l'utilisation de l'apprentissage automatique pour résoudre des problèmes combinatoires, en particulier les premières applications au VRP. La section 2.3 est consacrée au VRP avec fenêtres de temps, au problème des soins à domicile (*Home Health Care Routing and Scheduling Problem*, HHCRSP) et aux méthodes adaptées à leur résolution. Enfin, la section 2.4 explore plus en détail les avancées récentes en apprentissage automatique appliquées au problème des tournées de véhicules.

2.1 Historique et résolution du VRP

Comme mentionné précédemment, le VRP, introduit en 1959 par Dantzig et Ramser [1], a suscité un grand nombre de recherches depuis. Cet article modélisait pour la première fois le ravitaillement à l'aide d'une flotte de véhicules partant d'un dépôt unique. Cinq années plus tard, Clark et Wright [17] ont généralisé ce problème sous forme d'un programme linéaire, ce qui leur a permis de proposer l'heuristique « *savings* », qui a posé les bases de la construction d'heuristiques pour de nombreuses variantes du VRP. En 1981, le problème a été démontré comme étant *NP-difficile* [11]. En raison de sa présence dans de nombreux domaines industriels, il est rapidement devenu l'un des problèmes scientifiques les plus étudiés. Une synthèse des méthodes fondamentales ayant marqué son évolution est présentée dans [18].

Les méthodes de résolution du VRP sont classiquement regroupées en trois grandes catégories, en fonction de leur nature algorithmique et de leur capacité à garantir ou non l'optimalité des solutions obtenues.

La première catégorie regroupe les méthodes heuristiques, qui ont pour objectif de produire rapidement des solutions de qualité acceptable, sans garantie d'optimalité. Parmi les travaux fondateurs de cette approche, on peut citer ceux de Dantzig et Ramser [1], notamment en lien avec l'utilisation de l'algorithme de *matching*. Ces méthodes se subdivisent généralement en deux sous-classes : d'une part, les heuristiques de construction, qui génèrent une solution initiale en assemblant progressivement des tournées à l'image de l'heuristique dite « *savings* » proposée par Clarke et Wright [17] ; d'autre part, les heuristiques d'amélioration, qui visent à optimiser une solution préexistante par des modifications locales, comme l'algorithme « *k-opt* » introduit dans [19].

La deuxième catégorie correspond aux métaheuristiques, qui étendent les principes des heuristiques à travers des schémas de recherche plus globaux, souvent inspirés de phénomènes naturels ou de stratégies d'exploration-intensification. Parmi les approches les plus étudiées dans ce cadre, on retrouve la *recherche tabou*, développée par Glover [20], ainsi que le *recuit simulé*, proposé initialement dans [21], qui repose sur une analogie avec le processus de refroidissement des métaux.

Enfin, la troisième catégorie regroupe les méthodes exactes, dont l'objectif est de garantir la résolution optimale du problème. On distingue principalement quatre grandes approches dans cette famille : la programmation dynamique [22], la relaxation lagrangienne [23], le Branch and Bound (BB) [24], ainsi que le Branch and Cut (BC), qui combine les principes de séparation de contraintes et d'élagage. Ce dernier, ainsi que le Branch and Price (BP) [14], se sont imposés comme les méthodes exactes les plus performantes pour le VRP jusqu'au début des années 2000, en particulier pour la résolution d'instances de petite à moyenne taille, comptant jusqu'à une cinquantaine de clients.

L'approche BC est fondée sur les travaux de Padberg et Rinaldi [25], ainsi que ceux de Grötschel et Holland [26]. Elle est exploitée dans [27], se basant sur une reformulation du CVRP proposée par Baldacci et al. [28]. Cette reformulation repose sur une modélisation en programmation linéaire en nombres entiers selon une approche de flux sur un réseau à deux commodités. L'algorithme associe énumération implicite et plans de coupe, produisant à chaque nœud de l'arbre de recherche des coupes valides renforçant l'enveloppe convexe sans exclure de solutions entières. L'un des éléments clés réside dans la recherche de solutions réalisables de qualité ou de bornes supérieures resserrées.

Les modèles récents du VRP s'écartent toutefois du cadre théorique initial en raison des contraintes issues du monde réel. Diverses variantes, de plus en plus complexes, intègrent ces contraintes : fenêtres temporelles (VRPTW), capacités limitées (CVRP), compatibilités patient–personnel soignant, etc. Bien que Dantzig et Ramser [1] n'aient pas introduit ex-

plicitement le terme *Capacitated Vehicle Routing Problem*, leur étude constituait déjà une première formalisation de la problématique des capacités ; ils ont donc jeté les bases du CVRP, dont la terminologie est apparue par la suite.

Le livre « *Vehicle Routing : Problems, Methods, and Applications, Second Edition* » [18] fournit une revue complète des variantes du VRP et de leurs méthodes de résolution, couvrant les principales avancées jusqu'en 2014. On y apprend que, durant la décennie 2004–2014, les recherches se sont majoritairement concentrées sur l'amélioration des métaheuristiques, en particulier par hybridation. En 2012, cette approche est mise en œuvre par Vidal et al. [29] au moyen d'une métaheuristique hybride combinant la capacité d'exploration étendue des algorithmes évolutifs à base de population, les mécanismes d'amélioration agressive des méthodes de voisinage, et des stratégies avancées de gestion de la diversité. Leur méthode surpasse l'ensemble des métaheuristiques contemporaines pour chaque type de problème considéré. En 2021, une version améliorée de cet algorithme est proposée dans [30], spécialisée pour le CVRP. Elle introduit la structure de voisinage SWAP*, qui consiste à échanger deux clients entre différentes tournées sans imposer de position fixe, ce qui accroît la diversité des solutions explorées et renforce l'efficacité de la recherche locale. Cette version permet à l'algorithme de conserver son statut de métaheuristique de référence.

2.2 Apprentissage automatique pour les tournées de véhicules

En parallèle de l'évolution du VRP, les techniques d'apprentissage automatique ont connu un développement remarquable depuis leurs débuts. Moins de deux années avant l'apparition du VRP dans la littérature scientifique, Rosenblatt [31] introduit la notion de *perceptron*, visant à améliorer la précision des prédictions effectuées par une machine, et constituant l'une des premières briques théoriques des réseaux neuronaux. Au fil des décennies, l'apprentissage automatique est devenu un outil incontournable dans de nombreux problèmes scientifiques contemporains.

L'application des modèles d'apprentissage à l'optimisation combinatoire s'est développée progressivement sous différentes formes. Bengio et al. [32] reviennent sur l'utilisation de l'apprentissage automatique dans la résolution de problèmes combinatoires, incluant le VRP et ses variantes complexes. Les méthodes de l'état de l'art reposent souvent sur des choix heuristiques remplaçant des décisions computationnellement coûteuses ; l'apprentissage automatique (*Machine Learning*, ML) apparaît ainsi comme un candidat naturel pour apprendre ces choix.

L'utilisation du ML pour la résolution de problèmes combinatoires peut prendre plusieurs

formes décrites dans [32]. Il peut être utilisé pour configurer des algorithmes de résolution en prédisant, par exemple, si l'application d'une décomposition permettrait de résoudre le problème plus efficacement. Kruber et al. [33] l'emploient ainsi pour estimer l'intérêt d'une décomposition de Dantzig-Wolfe dans des instances de programmation linéaire en nombres entiers mixtes (MILP), tandis que Bonami et al. [34] prédisent l'utilité d'une linéarisation dans des problèmes de programmation quadratiques en nombres entiers mixtes (MIQP).

Le ML peut également être utilisé en parallèle de l'exécution des algorithmes de résolution. Dans ce contexte, Lodi et Zarpellon [35] explorent le remplacement des heuristiques par apprentissage automatique pour la sélection de la variable de branchement et la sélection de nœuds dans le *Branch and Bound* pour la résolution de problèmes MILP.

Enfin, le ML peut permettre de générer directement une solution à un problème d'optimisation combinatoire, sans recours à d'autres algorithmes. Vinyals et al. [36] introduisent pour cela le *Pointer Network*, un modèle visant à résoudre le problème du voyageur de commerce euclidien à l'aide de l'apprentissage profond. Le modèle repose sur un encodeur et un décodeur, tous deux construits sur des réseaux neuronaux récurrents. L'encodeur traite les nœuds du graphe d'entrée et génère des vecteurs d'activation pour chacun, tandis que le décodeur utilise un mécanisme d'attention pour produire une distribution de probabilité sur ces nœuds, facilitant ainsi la sélection de l'ordre optimal de visite. Ce mécanisme d'attention, introduit dans [37], constitue la base des *Graph Attention Networks* (GAT).

Les GAT représentent une classe particulière de réseaux de neurones pour graphes. Ces derniers sont largement utilisés pour la résolution de problèmes combinatoires. Cappart et al. [38] répertorient un large éventail d'applications actuelles des GNN. Initialement introduits par Scarselli [39], les GNN sont conçus pour traiter des données structurées en graphes. L'article fondateur propose un cadre formel pour l'apprentissage automatique pour des structures relationnelles. Dans les GNN, chaque nœud agrège les informations de ses voisins. Ce processus est itéré afin d'exploiter des dépendances à plus longue portée. Les GAT ajoutent à ce mécanisme une pondération adaptative sur l'importance des voisins à l'aide d'un mécanisme d'attention.

Pour le VRP et le CVRP, les premières recherches se concentrent sur l'apprentissage par renforcement (*Reinforcement Learning*, RL) [40, 41]. Dans [41], l'approche proposée surpasse certaines heuristiques classiques, comme par exemple « savings », ainsi que l'outil OR-Tools de Google [42], sur des instances de 50 à 100 clients, en termes de qualité de solution, avec des temps de calcul comparables après entraînement. Bien que leur méthode gloutonne soit efficace, les meilleurs résultats sont obtenus à l'aide d'un décodeur reposant sur une stratégie de recherche en faisceau (*Beam Search*, BS), conservant les dix trajectoires les plus probables

sans augmentation significative du temps de calcul. Par ailleurs, cette méthode s'adapte efficacement à l'augmentation de la taille des instances, tout en maintenant des performances compétitives.

En 2018, Kool et al. [43] proposent un modèle fondé sur les mécanismes d'attention pour résoudre des variantes complexes du VRP. En s'appuyant sur les jeux de données présentés dans [41], ils démontrent que leur approche dépasse les méthodes fondées sur l'apprentissage par renforcement et les meilleures références disponibles pour le décodage glouton. Leur stratégie gloutonne surpasse également la recherche en faisceau décrite par Nazari, tout en se rapprochant des performances de LKH3 [44], un algorithme de référence pour les benchmarks du CVRP, avec un temps de calcul inférieur à une seconde par instance.

Plus récemment, Morabit et al. [45] proposent d'utiliser l'apprentissage automatique pour résoudre de manière successive des instances du CVRP, dans lesquelles seules de légères modifications sont apportées à partir d'une solution précédente. Les avancées récentes concernant les GNN appliqués aux tournées de véhicules sont détaillées dans la section finale de cette revue de littérature.

2.3 Problème d'acheminement des soins de santé à domicile

En 1987, Solomon introduit dans [46] la notion de problème de tournées de véhicules avec contrainte de fenêtres de temps. Ce problème est central dans le cadre de notre recherche, cette contrainte étant l'une des plus déterminantes dans la construction d'une solution au problème présenté dans la section 1.1. De plus, elle modifie la structure du problème de manière significative, rendant complexe le transfert d'idées de conception entre le VRP classique et le VRPTW, et nécessitant une réflexion spécifique sur les méthodes de résolution. En 2005, une revue de littérature en deux parties est proposée par Bräysy et Gendreau [47,48], retraçant les différentes heuristiques et métaheuristiques appliquées à la résolution du VRPTW. En 2012, Baldacci et al. présentent une autre revue portant cette fois sur les méthodes exactes [49]. Ils y soulignent que la qualité des bornes inférieures, désormais proche des valeurs optimales, s'est fortement améliorée grâce à la combinaison d'une formulation par partition et d'algorithmes fondés sur la génération de colonnes (GC). Toutefois, ces méthodes exactes restent limitées par leur complexité temporelle. Dans le cadre de notre recherche, les algorithmes doivent être exécutés dans un contexte continu en temps réel, afin de répondre rapidement aux patients en attente. Ce besoin opérationnel impose le recours à des méthodes heuristiques, métaheuristiques, ou fondées sur l'apprentissage automatique. Nous nous concentrons donc, dans la suite de cette revue de littérature, sur les avancées de ces méthodes non exactes pour les différentes variantes du VRP.

En parallèle, le secteur des soins à domicile connaît depuis plusieurs décennies une croissance soutenue, portée par une demande accrue de services personnalisés et flexibles. Cette évolution génère des défis logistiques importants, notamment en matière de planification quotidienne des interventions et d'optimisation des tournées. Le problème associé à ce contexte est celui de l'acheminement et de la programmation des soins à domicile, appelé HHCRSP.

En 1998, Cheng et Rich [50] définissent le HHCRSP comme une variante du VRP intégrant des fenêtres temporelles, plusieurs dépôts, et des contraintes spécifiques au domaine considéré, comme détaillées en section 1.2. Leur article propose une première formulation mathématique du problème sous forme de MILP, intégrant les contraintes de temps de trajet et de durée de travail des soignants. À l'aide du solveur CPLEX [51], les auteurs résolvent exactement de petites instances (4 membres du personnel soignant et 10 patients) et ouvrent la voie à l'utilisation d'approches heuristiques pour les cas de plus grande taille.

La formulation du HHCRSP repose sur celle du MDVRPTW, à laquelle viennent s'ajouter les contraintes spécifiques au contexte médical lorsque nécessaire. Pour la suite de cette section, nous concentrons l'analyse sur le MDVRPTW et ses variantes plus complexes, telles que le MDCVRPTW, intégrant des contraintes de capacité, de compatibilité, etc.

Dans les sous-sections suivantes, nous décrivons les contributions récentes relatives au HHCRSP ainsi qu'à ses variantes dynamiques, plus proches du cadre du problème étudié. Nous ne détaillons que les approches compatibles avec une résolution rapide, en lien avec les contraintes de temps réel imposées dans ce travail. Ainsi, nous nous concentrons sur les méthodes heuristiques, métaheuristiques et issues de l'apprentissage automatique, en écartant les méthodes exactes trop coûteuses en temps de calcul.

L'algorithme de résolution utilisé dans cette recherche a pour objectif de résoudre rapidement des instances du HHCRSP, avec un niveau de qualité proche de l'optimal. Notre problème repose sur un MDCVRPTW avec contrainte de double compatibilité. L'algorithme traite plusieurs instances statiques du problème afin de comparer la qualité des insertions possibles pour un nouveau patient dans un planning existant. L'algorithme ne distingue pas la résolution de plusieurs instances similaires, modifiées uniquement par la fenêtre temporelle d'insertion du patient. L'aspect dynamique de cette procédure n'est perçu que par l'opérateur humain.

2.3.1 Résolution du HHCRSP

Trois revues de littérature présentent les travaux effectués sur les méthodes de résolution pour le HHCRSP. En 2017, Fikar et Hirsch [52] se concentrent sur les contributions parues

dans des journaux scientifiques avant 2016. La même année, Cissé et al. [53] étendent cette analyse à d'autres contributions. Plus récemment, en 2021, Di Mascolo et al. [54] retracent les approches utilisées au fil du temps pour la résolution du HHCRSP.

Parmi les méthodes recensées, plusieurs reposent sur les métaheuristiques. Les trois travaux suivants traitent le HHCRSP en intégrant différentes considérations dans la fonction objectif et les contraintes. Dans [55], Bertels et Fahle étudient une version du HHCRSP intégrant des contraintes faibles de préférence. L'article propose un modèle combinant programmation linéaire (LP), programmation par contraintes (CP) et métaheuristiques, afin de minimiser les coûts de transport tout en maximisant la satisfaction des patients et du personnel soignant. Ils montrent que leur approche hybride LP-CP-métaheuristique, notamment la combinaison CP et recherche tabou, permet d'obtenir de meilleures solutions pour chaque type d'instance sans accroître le temps de calcul.

En 2007, une version simplifiée, avec un dépôt unique et sans contraintes de compatibilité, est présentée dans [56]. Les auteurs y proposent une méthode de résolution fondée sur l'optimisation par essaims de particules (PSO). L'intérêt principal de cette méthode, dans le contexte de notre recherche, réside dans l'usage des heuristiques. L'article introduit une stratégie appelée *Earliest Start Time Priority with Minimum Distance Assignment* (ESTPMDA), permettant de générer une solution initiale en priorisant les soins selon leur heure de début, tout en minimisant la distance parcourue. À cette heuristique de construction s'ajoutent des opérations d'amélioration locale par insertion et échange, intégrées à l'algorithme PSO. La méthode permet une réduction significative des distances parcourues (jusqu'à 31 %) par rapport aux solutions manuelles et à l'outil ILOG Dispatcher, sur des cas réels. Elle démontre également que la combinaison des métaheuristiques améliore les performances de l'algorithme face à d'autres variantes de celui-ci. Toutefois, le temps de calcul reste supérieur à trois minutes, ce qui dépasse les seuils acceptables pour le contexte temps réel de notre étude.

En 2015, Hiermann et al. [57] proposent un modèle multi-modal complet, intégrant notamment des contraintes de fenêtres temporelles et de compatibilité. Le modèle est conçu pour être facilement adaptable à d'autres instances du HHCRSP, en intégrant les spécificités du problème directement dans la fonction objectif. La méthode comporte deux phases : une phase de construction, via CP ou aléatoire, suivie d'une phase d'amélioration appliquant une métaheuristique. Quatre métaheuristiques sont considérées : la recherche de voisinage variable, un algorithme mémétique, la recherche par dispersion (*Scatter Search*), et une hyper-heuristique de recuit simulé. Les résultats montrent que l'algorithme mémétique fournit les meilleures solutions parmi les approches testées. Toutefois, l'obtention de solutions de bonne qualité nécessite à nouveau un temps de calcul de 100 à 300 secondes inadapté au contexte de temps

réel dans lequel notre travail se situe.

Ces différents travaux visent à réduire le temps nécessaire à la résolution tout en maintenant une qualité de solution satisfaisante. À notre connaissance, peu de contributions se concentrent sur la réduction du temps de calcul pour le HHCRSP et les variantes du VRP afin d'atteindre des délais compatibles avec les exigences du temps réel. La plupart des recherches privilégient d'autres axes d'amélioration ou des variantes plus complexes du problème. Nous citons quelques travaux récents permettant une réduction significative du temps de résolution pour des instances du CVRP, VRPTW et MDVRPTW.

Pour le VRP, en 2023, Hou et al. [58] proposent la méthode TAM (*Two-stage Divide Method*) pour résoudre efficacement le VRP pour des tailles d'instances contenant plus de 5000 nœuds en temps réel. L'approche propose une fonction de masque global permettant de garantir le respect des contraintes globales lors de la décomposition du VRP en plusieurs sous-problèmes du problème du voyageur de commerce (TSP) de petite taille. Ils utilisent ensuite une résolution en parallèle plus rapide tout en facilitant ainsi la généralisation des heuristiques. La méthode permet la résolution d'instances du VRP à plus de 1000 nœuds en restant sous les 2 secondes tout en se rapprochant des performances de LKH3 [44] pour les instances de grandes tailles. Pour les instances un peu plus petites, de 100 à 400 nœuds, ces performances sont un peu moins bonnes que LKH3 mais avec un temps d'exécution 10 fois plus faible pour la version de base de leur approche.

Pour le CVRP, l'apprentissage par renforcement profond est utilisé pour améliorer les performances de [43]. Dans [59], les auteurs étendent la méthode basée sur l'attention de [43] en y ajoutant deux politiques collaboratives d'apprentissage par renforcement profond (*Deep Reinforcement Learning*, DRL) : le « *seeder* » et le « *reviser* ». Le premier génère des solutions candidates, tandis que le second les modifie en divisant le trajet complet en sous-tours. Le *reviser* optimise simultanément chaque sous-tour pour réduire la distance totale parcourue, en se concentrant sur un espace de recherche restreint facilitant l'exploitation des meilleures solutions. Cette approche améliore la qualité des résultats sans accroissement significatif du temps d'exécution, permettant de résoudre des instances de 100 clients en moins de deux secondes.

Pour le VRPTW, une approche hybride est proposée dans [60]. Celle-ci est constituée de deux phases : la première applique une heuristique fondée sur le recuit simulé, tandis que la seconde sélectionne une fenêtre temporelle pour chaque patient afin de définir les horaires de visite. Les résultats montrent que cette approche est plus rapide et plus performante que les méthodes heuristiques et exactes de l'état de l'art, divisant par dix le temps de calcul pour les instances les plus larges. En 2024, Zong et al. [61] proposent une approche basée

sur l'apprentissage par renforcement intégrant un modèle d'agent encodant les contraintes du VRPTW. Cette approche utilise une récompense augmentée par une pénalité temporelle pour modéliser les limites des fenêtres temporelles et un gestionnaire de tâches pour faciliter la coopération entre les véhicules. Les expériences menées sur des ensembles de données réels et un benchmark public montrent une amélioration des performances allant jusqu'à 11,7 % par rapport aux autres approches basées sur l'apprentissage par renforcement, tout en résolvant des instances contenant jusqu'à 100 clients en restant proche d'une seconde d'exécution. Cependant, la qualité des solutions peut être affectée par la taille et la complexité des instances, et l'entraînement du modèle nécessite des ressources computationnelles significatives.

Enfin, pour le MDVRPTW, l'apprentissage par renforcement profond est à nouveau exploité dans [62], couplé à la recherche locale. L'algorithme proposé est multi-agents, utilisant un cadre encodeur-décodeur avec une stratégie d'attention multi-têtes pour générer des itinéraires optimisés. La qualité des solutions est ensuite renforcée par l'application d'opérations de recherche locale. L'approche surpasse l'outil OR-Tools de Google [42], utilisé comme référence dans cette étude. Elle permet de produire des solutions de meilleur coût pour l'ensemble des instances testées, tout en divisant par 20 le temps de calcul, y compris en présence d'incertitudes.

2.3.2 Variante dynamique

Contrairement aux problèmes d'acheminement et de planification des soins à domicile abordés dans la littérature, cette recherche se concentre sur la résolution d'un problème soumis à une contrainte de temps. Lorsqu'un nouveau patient doit être inséré, l'algorithme doit résoudre plusieurs instances du problème, chacune correspondant à une insertion possible, et ce, en moins de vingt secondes. Pour répondre à cette exigence, les avancées récentes en matière de résolution dynamique sont particulièrement pertinentes.

Nous nous intéressons aux approches dynamiques qui traitent l'arrivée progressive de nouveaux patients à insérer dans un planning. Le premier travail structurant ce domaine est proposé par Bennett et Erera [63], qui visent à maximiser le nombre de clients servis en fixant des rendez-vous à partir d'un inventaire fini de créneaux disponibles. L'approche repose sur une planification dynamique en horizon roulant, intégrant continuellement les nouvelles demandes dans un planning existant. Elle améliore la gestion des rendez-vous dans un environnement dynamique.

Sur la base de ce travail fondateur, Demirbilek et al. proposent deux contributions successives [64,65] introduisant une méthode heuristique basée sur la construction incrémentale. Celle-ci prend en compte le planning en cours, la nouvelle demande, ainsi que des demandes futures

générées de manière aléatoire afin d'anticiper les évolutions probables. Ces demandes simulées permettent de mieux préparer l'insertion de patients à venir. Dans des cas à forte densité de demandes, cette approche permet d'augmenter d'environ 20 % le nombre de patients pris en charge, avec un temps d'insertion inférieur à une minute par demande.

Enfin, l'apprentissage par renforcement est également exploré dans ce contexte dynamique. Dans [66], la résolution de la variante dynamique suppose qu'une agence de soins à domicile doit décider, en temps réel, d'accepter ou de rejeter chaque nouvelle demande. Le problème est modélisé comme un processus de décision markovien, et il est résolu à l'aide de techniques d'apprentissage par renforcement. Les résultats expérimentaux montrent que l'approche proposée surpasse les performances des algorithmes existants en termes de qualité de solution. Elle permet une réduction significative du temps d'exécution ainsi qu'une augmentation du nombre de patients pris en charge, rendant la méthode applicable dans un cadre opérationnel de prise de décision en temps réel.

Cependant, comme pour toute méthode d'apprentissage par renforcement, cette approche nécessite un temps d'entraînement élevé. De plus, ses performances dépendent fortement de la qualité de la fonction de récompense, de la représentativité des environnements simulés utilisés lors de l'apprentissage, ainsi que du calibrage des hyperparamètres du modèle, rendant son déploiement complexe dans un contexte réel.

2.4 Les GNN pour les tournées de véhicules

À ce jour, l'application des réseaux de neurones pour graphes à la résolution des problèmes de tournées de véhicules, et en particulier au HHCRSP, reste limitée. Cette recherche vise à combler ce manque, à approfondir les résultats prometteurs déjà observés pour les problèmes combinatoires, et à étudier l'apport des réseaux neuronaux dans l'efficacité des algorithmes de résolution. Actuellement, la majorité des travaux utilisant les GNN pour résoudre le VRP et ses variantes reposent sur leur couplage avec l'apprentissage par renforcement. Cette tendance est confirmée par les revues de littérature sur le sujet [67,68]. Cette approche hybride obtient de bons résultats en pratique lorsqu'elle est comparée à des méthodes de référence telles qu'OR-Tools [42] ou à l'approche proposée par Nazari [41], discutée précédemment. Dans [69], la méthode combine un *Graph Convolutional Network* (GCN) [70] avec le RL, tandis que dans [71], ce sont les GAT qui sont utilisés. Dans les deux cas, les auteurs montrent que leur méthode surpasse les approches existantes tout en conservant un temps d'exécution de quelques secondes.

À l'inverse, les résultats obtenus par l'utilisation des GNN sans apprentissage par renforce-

ment sont moins satisfaisants. Ammon et al. [72] proposent une approche fondée sur l'apprentissage supervisé pour prédire la probabilité qu'une arête appartienne à la tournée optimale. Le modèle génère une matrice d'adjacence attribuant à chaque arête (i, j) une probabilité p_{ij} d'être incluse dans la solution. À partir de cette matrice, une tournée est reconstruite via BS. Bien que cette méthode surpasse les stratégies aléatoires de référence, ses performances restent inférieures à celles d'OR-Tools et des approches basées sur le RL, avec un écart de 16 à 20 % sur le coût total de la tournée. L'article met en évidence le potentiel des GNN pour la résolution de tels problèmes, tout en soulignant que l'apprentissage supervisé requiert de grandes quantités de données d'entraînement, notamment pour les instances de VRP de grande taille, et que le RL reste pour le moment plus performant dans ce contexte.

Peu de travaux se concentrent sur les variantes du VRP avec fenêtres temporelles, qui sont plus complexes et donc moins abordées dans la littérature. Parmi ces rares contributions, Zhang et al. [73] s'intéressent au MDVRPSTW, une extension du MDVRPTW dans laquelle la contrainte de fenêtre de temps est dite « molle » : le non-respect de la contrainte induit une pénalité au lieu de l'infaisabilité de la solution. L'architecture utilisée repose sur un GAT couplé à du RL. Les auteurs montrent, à l'aide d'un benchmark, que leur approche est supérieure en termes de qualité de solution et de temps de calcul, avec une résolution d'instances contenant 100 clients en moins de deux secondes. Par ailleurs, Mukherjee et al. [74] proposent d'utiliser les GNN pour améliorer la métaheuristique des algorithmes évolutionnistes, en prédisant l'aptitude des solutions candidates afin d'identifier les individus les plus prometteurs. Toutefois, à notre connaissance, ce travail ne présente aucun résultat expérimental.

Dans notre travail, les GAT sont utilisés pour prédire la faisabilité d'une insertion de patient dans un planning déjà optimisé, sans nécessiter la résolution complète de l'instance correspondante. Ce travail s'appuie directement sur les fondations posées dans [75], qui constituent le point de départ méthodologique de notre approche. L'adoption des GAT y est motivée par leur capacité à pondérer efficacement l'importance des informations issues des nœuds voisins lors de l'agrégation des caractéristiques. Cette propriété est particulièrement adaptée à la modélisation des relations complexes et des contraintes spécifiques au HHCRSP, telles que les contraintes de compatibilité, de fenêtre temporelle ou encore de capacité.

CHAPITRE 3 DESCRIPTION DU PROBLÈME ET DE L'APPROCHE

3.1 Description du problème

L'objectif du problème est que l'ensemble des soignants $R = \{1, 2, \ldots, m\}$ serve l'intégralité des patients contenus dans l'ensemble $A = \{1, 2, \ldots, n\}$. Chaque membre du personnel soignant effectue exactement une *tournée* durant la journée. On définit une *tournée* comme le parcours nœud à nœud des arcs (i, j) de distance temporelle $t_{i,j}$ exprimée en minutes, reliant les différents nœuds patients et points de départ. Pour que la *tournée* soit valide, le membre du personnel soignant doit partir de son point de départ (dépôt), représenté par un nœud, et y revenir. Les soignants doivent partir et revenir durant leur fenêtre temporelle, $[a_k, b_k]$ pour le membre du personnel soignant k par exemple, et servir les patients pendant leur propre fenêtre, $[a_i, b_i]$ pour le patient i par exemple, en tenant compte du temps nécessaire pour effectuer le service l_i.

En plus de servir les patients, les soignants doivent veiller à ce que la somme des demandes en pansements d_i^1 et en seringues d_i^2 de chaque patient ne dépasse pas le nombre de pansements C_k^1 et de seringues C_k^2 disponibles dans leur véhicule. Le problème est formulé comme une tournée de véhicules où les m soignants doivent servir les patients compatibles parmi les n patients, sans dépasser leurs capacités en pansements et en seringues. L'ensemble global des nœuds est noté $V = R \cup A = \{1, 2, \ldots, m + n\}$, contenant les soignants $\{1, 2, \ldots, m\}$ et les patients $\{m + 1, m + 2, \ldots, m + n\}$.

Les ensembles des compétences et affinités possibles pour chaque patient et chaque membre du personnel soignant sont respectivement notés $\mathcal{F}$ et $\mathcal{S}$. Les compétences requises pour soigner le patient i sont notées F_i, et celles maîtrisées par le membre du personnel soignant k sont notées F_k. Les affinités utilisent une notation similaire avec S_i et S_k. On introduit également le paramètre $\rho_{i,k} \in \{0, 1\}$ qui assure la compatibilité entre le patient i et le membre du personnel soignant k, avec $\rho_{k,k} = 1$ et $\rho_{j,k} = 0$ $\forall k \in R$, $\forall j \in R \setminus \{k\}$. On définit sa valeur par l'équation suivante :

$$\rho_{i,k} = \begin{cases} 1, & \text{si } (F_i \subseteq F_k) \wedge (S_i \subseteq S_k) \\ 0, & \text{sinon} \end{cases}$$

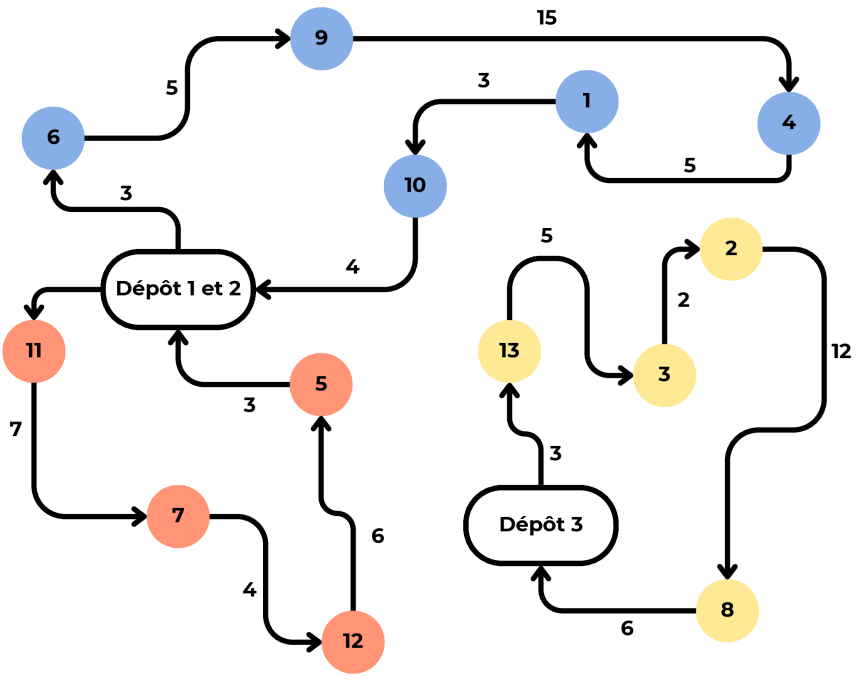


FIGURE 3.1 Illustration du service de 13 patients par 3 soignants

3.2 Formulation mathématique

Les variables de décision sont définies comme suit : $x_{i,j,k} \in \{0,1\}$ qui vaut 1 si l'arc (i,j) est traversé par le membre du personnel soignant k, $y_{i,k} \in \{0,1\}$ qui vaut 1 si le service du nœud i est effectué par k et $z_i \in \{0,1\}$, qui vaut 1 si le nœud i n'a pas été servi. Enfin, on définit $s_i \geq 0$ la variable déterminant le temps de début du service du nœud i. On définit le problème **P** comme le problème d'optimisation linéaire décrit ci-dessous.

$$\min \sum_{k \in R} \sum_{i,j \in V} t_{i,j} x_{i,j,k} + B \sum_{i \in A} z_i \tag{3.1}$$

$$\text{sujet à :} \quad \sum_{k \in R} y_{i,k} + z_i = 1 \qquad \forall i \in A \tag{3.2}$$

$$\sum_{j \in V} x_{k,j,k} = 1 \qquad \forall k \in R \tag{3.3}$$

$$\sum_{i \in V} x_{i,k,k} = 1 \qquad \forall k \in R \tag{3.4}$$

$$\sum_{h \in V} x_{i,h,k} = \sum_{h \in V} x_{h,i,k} \qquad \forall i \in A,\ \forall k \in R \tag{3.5}$$

$$\sum_{k \in R} \sum_{j \in V} x_{j,i,k} \leq 1 \qquad \forall i \in V \tag{3.6}$$

$$y_{i,k} = \sum_{j \in V} x_{j,i,k} \qquad \forall i \in A,\ \forall k \in R \tag{3.7}$$

$$s_j \geq s_i + l_i + t_{i,j} - M(1 - x_{i,j,k}) \qquad \forall i, j \in V,\ \forall k \in R \tag{3.8}$$

$$a_i \leq s_i \leq b_i - l_i \qquad \forall i \in V \tag{3.9}$$

$$\sum_{i \in A} d_i^1 y_{i,k} \leq C_k^1 \qquad \forall k \in R \tag{3.10}$$

$$\sum_{i \in A} d_i^2 y_{i,k} \leq C_k^2 \qquad \forall k \in R \tag{3.11}$$

$$y_{i,k} \leq \rho_{i,k} \qquad \forall i \in A,\ \forall k \in R \tag{3.12}$$

$$x_{i,j,k} \leq \rho_{i,k} \cdot \rho_{j,k} \qquad \forall i \in A,\ j \in V,\ \forall k \in R \tag{3.13}$$

L'objectif 3.1 est de minimiser le temps de trajet des soignants tout en servant le maximum de patients, en tenant compte du terme de pénalisation avec B suffisamment grand. La contrainte 3.2 sert à fixer les valeurs de $y_{i,k}$ et z_i en fonction du service du patient i. Les contraintes 3.3 et 3.4 garantissent le départ et le retour du membre du personnel soignant k à son dépôt. La contrainte 3.5 est celle de conservation de flux, et 3.6 est celle de cohérence de service, garantissant qu'un nœud ne soit servi qu'une seule fois. La contrainte 3.7 assure que les valeurs des booléens $y_{i,k}$ et $x_{i,j,k}$ soient correctement reliées. Les contraintes 3.8 et 3.9 garantissent le respect des temporalités et des fenêtres temporelles. Plus précisément, la contrainte 3.8 utilise la méthode du Big-M [76], nécessitant un paramètre constant M suffisamment grand. Les contraintes 3.10 et 3.11 concernent les demandes et les capacités. Enfin, les contraintes 3.12 et 3.13 gèrent la compatibilité entre patients et soignants.

3.3 Description du cadre opérationnel

Fastercom propose à ses clients un algorithme permettant la prise de rendez-vous dans des tournées de véhicules. Un cadre typique d'opération est illustré à la figure 3.2. Ce cadre est caractérisé par quatre étapes clés : le stockage des informations déjà transmises et du planning associé, la prise en temps réel d'informations sur un nouveau patient, la réalisation de requêtes vers un solveur externe afin de déterminer la meilleure insertion, et enfin la transmission des informations au patient ainsi que la mise à jour du planning.

3.3.1 Description des insertions

L'algorithme définit les insertions à partir d'une discrétisation temporelle. Dans ce contexte, une insertion ne correspond pas à une heure précise, mais à un intervalle de temps dans la semaine à venir. Fastercom discrétise les semaines en divisant chaque journée en trois parties :

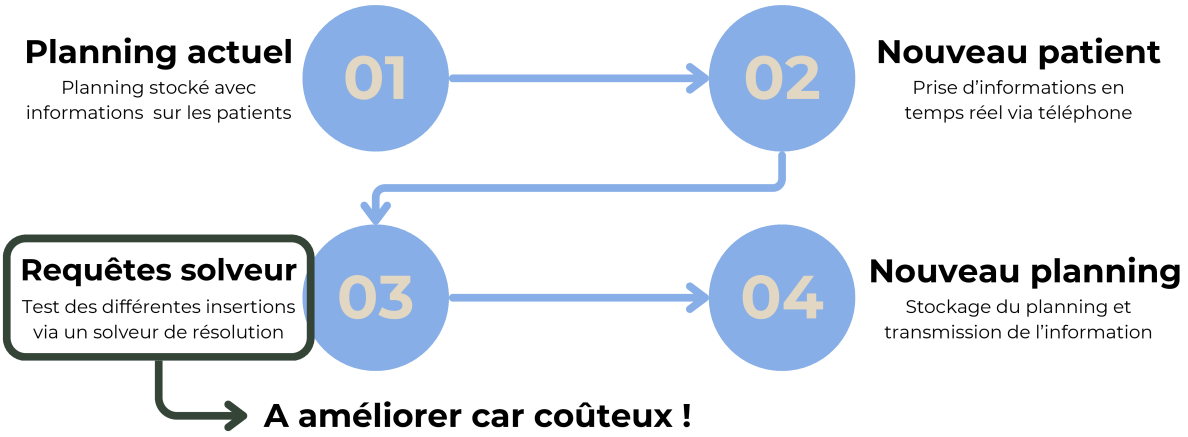


FIGURE 3.2 Cadre opérationnel actuel

AM, PM et soirée. La combinaison des jours et de cette division AM/PM/soirée définit les insertions possibles pour un patient. Dans le cas où un patient appelle et indique être disponible le lundi, mardi et mercredi toute la journée, il possède neuf insertions possibles, correspondant aux différents créneaux AM, PM et soirée de chaque journée. Lorsque ce patient communique ses disponibilités, l'algorithme détermine la meilleure insertion et la lui transmet, par exemple, mardi soir.

Cette information, « mardi soir », a deux objectifs :

- Elle est communiquée au patient, qui peut ainsi prévoir la venue d'un membre du personnel soignant à son domicile durant une plage horaire plus restreinte que ses disponibilités initiales.
- Elle devient une nouvelle contrainte du modèle. La fenêtre temporelle $[a_i, b_i]$ du patient est mise à jour pour correspondre à l'insertion choisie. Dans notre exemple, la fenêtre de temps passerait de [lundi, mercredi] à [mardi 16h, mardi 20h].

Il est important de noter que le service du patient n'est pas fixé à une heure précise, mais peut se dérouler à tout moment durant une certaine fenêtre temporelle, restreinte par rapport aux disponibilités initiales du patient. C'est cette fenêtre que nous appelons « insertion ». Par ailleurs, nous faisons l'hypothèse que le solveur utilisé par l'algorithme de Fastercom est *OR-Tools* [42]. Ce dernier se base sur une recherche locale [16], et la qualité des solutions retournées dépend souvent du temps alloué à la recherche. C'est ce solveur que nous utilisons pour créer les données, comparer les méthodes et effectuer toutes les résolutions nécessaires à l'avancement de la recherche. L'outil développé par Fastercom est utilisé par d'autres entreprises pour construire leur prochain planning hebdomadaire. Lorsqu'un nouveau patient appelle et donne ses disponibilités, ses demandes et ses compatibilités, l'outil cherche à le

placer dans le planning à l'endroit idéal.

Cet endroit idéal, appelé « meilleure insertion », désigne celle dont la qualité calculée est la plus élevée. C'est cette insertion qui est retenue et communiquée au patient. En pratique, il s'agit de l'insertion ayant la valeur objectif la plus haute. Nous définissons une fonction objectif représentant la qualité d'une insertion. Cette fonction évalue l'horaire de travail en prenant en compte le temps total de travail des soignants et l'équilibre du temps de travail entre eux. La fonction retourne un score compris entre 0 et 1, où 1 est optimal. Ce choix est motivé par l'importance de laisser du temps pour de futurs patients potentiels et d'éviter qu'un membre du personnel soignant, pouvant être le seul compatible avec un futur patient, ne puisse pas effectuer le service en raison d'un déséquilibre important dans les temps de travail. La formule de la fonction objectif est donnée par :

$$\textit{objective} = \min\left(1,\ \max\left(0,\ a \cdot \left(1 - \frac{\sum_{i=1}^{m} T_i}{T_{\max}}\right) + b \cdot \left(1 - \frac{\sigma_T}{\text{écart}_{\max}}\right)\right)\right) \tag{3.14}$$

où m représente le nombre total de soignants, T_i désigne le temps de travail du membre du personnel soignant i (pour $i = 1, \ldots, m$), et σ_T correspond à l'écart-type calculé sur l'ensemble des temps de travail $\{T_1, \ldots, T_m\}$. La quantité $T_{\max}$ représente le temps de travail total maximal possible pour l'ensemble des soignants, et le terme $\text{écart}_{\max}$ est défini comme $\frac{T_{\max}}{2}$. Les coefficients $a = 0.999$ et $b = 0.001$ sont utilisés pour pondérer respectivement la minimisation du temps de travail global et l'équilibrage de la charge entre les soignants. Cette formulation permet de favoriser les solutions minimisant le temps de travail tout en évitant une répartition trop déséquilibrée du travail.

Si l'algorithme de Fastercom trouve au moins une insertion où tous les patients, y compris le nouveau, sont servis, il communique celle qui maximise la valeur objective. Dans le cas contraire, l'algorithme communique l'impossibilité d'ajouter le nouveau patient.

3.3.2 Planification journalière et données historiques

Les données fournies, représentant 1137 journées indépendantes d'opérations, proviennent de l'optimisation effectuée par l'algorithme de Fastercom. Cependant, ces données ont pu être ajustées par l'entreprise utilisant l'algorithme après leur optimisation. Ces données, dites « post-optimisation », sont potentiellement insolubles, car des informations telles que le nombre de patients, les demandes et les compétences peuvent avoir été modifiées après l'exécution de l'algorithme, entraînant l'insatisfaction de certaines contraintes du problème. Pour remédier à cela, lors de l'extraction des données, une première passe est effectuée afin de s'assurer qu'aucune infaisabilité n'est présente. Si le problème est faisable, nous le conservons

tel quel. En revanche, s'il est infaisable, nous retirons les patients problématiques.

De plus, l'ensemble des données est constitué de journées indépendantes, alors que l'outil souhaité est un outil hebdomadaire. Pour répondre à cette problématique, un compromis est effectué en simulant les semaines à l'intérieur des journées d'opérations. En pratique, la discrétisation hebdomadaire en AM/PM/Soirée utilisée par Fastercom est remplacée par une discrétisation journalière en intervalles de temps de 30 minutes. L'idée étant qu'en discrétisant le temps d'une journée de 8 heures en intervalles de 30 minutes, comme le fait Fastercom en AM/PM/Soirée pour les jours de la semaine, l'ordre de grandeur du nombre d'insertions possibles dans le planning est préservé.

Ainsi, pour une journée de 8 heures, 16 insertions sont obtenues, et pour une semaine de 5 jours (excluant les fins de semaine), on obtiendrait 15 insertions possibles. L'objectif devient donc de déterminer quel intervalle de 30 minutes constitue la meilleure insertion pour le patient dans une journée d'exécution. Notre méthode communiquera alors cette insertion et restreindra la fenêtre du patient pour les ajouts subséquents.

3.4 Prérequis techniques et hypothèses approfondies

Cette section présente les notions essentielles à la compréhension de l'approche proposée, en précisant rigoureusement les concepts mathématiques employés. Les notions abordées dans ce travail se situent dans la théorie des graphes et dans celle de l'intelligence artificielle, plus précisément de l'apprentissage profond [77, 78]. Les différentes notions mathématiques utiles à la compréhension sont reprises ci-dessous :

— **Graphe orienté** : Un graphe orienté $G = (V, E)$ représente un ensemble de sommets V (également appelés nœuds) reliés par des arêtes orientées $E \subseteq V \times V$. Chaque arête $(u, v) \in E$ décrit une relation asymétrique, allant d'un sommet source u vers un sommet cible v, représentant un sens précis dans les relations ou flux d'informations modélisés.

— **Graphe homogène** : Un graphe homogène est un graphe orienté $G = (V, E)$ dans lequel tous les sommets et toutes les arêtes appartiennent à un seul et même type. Formellement, il existe une unique fonction de typage :

$$\tau : V \cup E \to \{\text{type unique}\}$$

Ce type de graphe est simple à manipuler mais ne permet pas de représenter des interactions complexes entre entités différentes.

— **Graphe hétérogène** : Un graphe hétérogène, ou hétérographe, est un graphe orienté

$G = (V, E)$ dans lequel les sommets et les arêtes sont répartis en différentes catégories ou types. Formellement, il existe des partitions pour les ensembles de sommets et d'arêtes en sous-ensembles distincts. Par exemple, dans notre recherche, les nœuds sont répartis en trois sous-ensembles :

$$V = A \cup R \cup T$$

où chaque sous-ensemble représente un type distinct de sommets, ici A l'ensemble des patients, R l'ensemble des soignants et T l'ensemble des intervalles de 30 minutes. Le même type de partition s'applique à l'ensemble des arêtes. L'utilisation des graphes hétérogènes dans ce mémoire est justifiée par la possibilité de représenter explicitement les interactions variées et complexes entre différents types d'entités, cruciales pour encoder fidèlement les instances d'un problème comme celui décrit à la section 3.2. Les graphes homogènes sont, eux, incapables d'apporter l'expressivité nécessaire.

— **Perceptron multicouche** [79] : Un MLP est une classe de réseau neuronal artificiel constitué de couches successives entièrement connectées permettant d'apprendre des représentations complexes à partir de données initialement simples. Mathématiquement, chaque couche l calcule :

$$h^{(l)} = \sigma(W^{(l)} h^{(l-1)} + b^{(l)})$$

où $W^{(l)}$ et $b^{(l)}$ sont les paramètres à apprendre (poids et biais), $h^{(l-1)}$ correspond aux entrées issues de la couche précédente, et σ est une fonction d'activation non linéaire.

— **Réseau de neurones pour graphes** [80, 81] : Les GNN étendent les réseaux neuronaux classiques pour traiter les données structurées sous forme de graphe, en capturant les informations relationnelles. Un GNN produit, pour chaque sommet v, une représentation latente en agrégeant ses propres caractéristiques et celles de ses voisins directs :

$$h_v^{(l)} = \gamma^{(l)} \left(h_v^{(l-1)}, \square_{u \in \mathcal{N}(v)} \, \phi^{(l)}(h_v^{(l-1)}, h_u^{(l-1)}, e_{u,v}) \right)$$

où $h_v^{(l)}$ est la représentation du sommet v à la couche l, $\mathcal{N}(v)$ représente l'ensemble des voisins de v, $e_{u,v}$ désigne les caractéristiques éventuelles associées à l'arête reliant u et v, $\phi^{(l)}$ est une fonction de propagation, $\gamma^{(l)}$ une fonction combinant les informations collectées, et $\square$ une opération d'agrégation symétrique telle que la somme, la moyenne ou le maximum.

Le terme $e_{u,v}$ est inclus dans la définition du GNN afin de considérer explicitement les caractéristiques des arêtes. Ce travail utilise spécifiquement des réseaux de neurones

pour graphes capables d'exploiter ces caractéristiques, puisque celles-ci encodent les informations décrivant les relations entre nos différents agents. Ainsi, les GNN permettent d'intégrer à la fois des informations locales (relations immédiates) et globales (structure complète du graphe) dans le processus d'apprentissage.

3.5 Objectif de la recherche et aperçu de l'approche

L'objectif de ce travail étant de proposer une méthode de résolution permettant de réduire le nombre d'appels vers un solveur et de répondre à la contrainte de temps réel introduite, notre étude se concentre sur la troisième étape du cadre opérationnel illustré à la figure 3.2, car c'est celle qui engendre les temps de calcul les plus élevés et celle qui doit être modifiée pour répondre en temps réel au patient.

L'utilisation de requêtes vers un solveur externe nécessitant plus de cinq secondes pour trouver une solution à chaque résolution du problème implique qu'il est souvent possible de ne tester que trois insertions parmi celles permises par le patient afin de respecter le délai de vingt secondes. À partir de ces différentes hypothèses et observations, nous modifions légèrement le cadre opérationnel et l'illustrons à la figure 3.3.

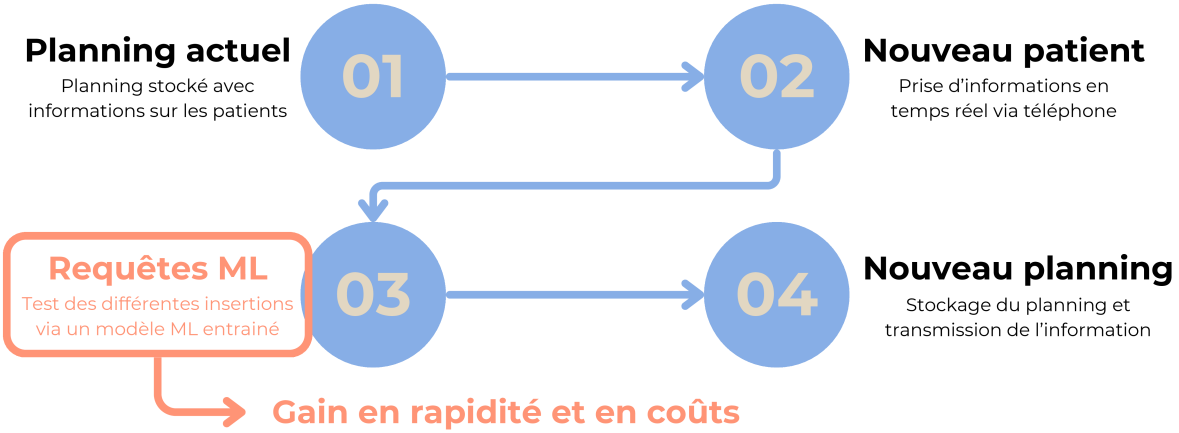


FIGURE 3.3 Nouveau cadre opérationnel proposé

Le nouveau cadre conserve la même structure avec une modification clé : l'utilisation d'un modèle basé sur l'apprentissage automatique pour déterminer la meilleure insertion, remplaçant les multiples appels à un solveur. Ce remplacement permet de réduire les coûts temporels en utilisant l'inférence du modèle ML pour prédire la meilleure insertion au lieu d'une résolution complète du problème.

Alors que la formulation traditionnelle du problème se concentre sur l'optimisation sans restrictions temporelles, ce mémoire traite d'une variante dynamique et plus exigeante où les décisions doivent être prises en temps réel, par exemple lors d'un appel téléphonique. Dans ce travail, nous présentons un cadre d'apprentissage automatique conçu pour répondre efficacement à ces demandes en temps réel. Notre approche modélise l'instance du problème, définie à la section 3.2, sous la forme d'un graphe hétérogène qui préserve à la fois la structure et les données du problème original. Cette modélisation vise à intégrer des relations différenciées selon la nature des entités et des connexions, permettant ainsi de modéliser des systèmes complexes avec des interactions variées. L'idée est de représenter les différents agents, tels que les patients, les soignants, ainsi que les intervalles de 30 minutes représentant les insertions possibles, tout en conservant la nature distincte des relations entre eux.

À l'aide de cette représentation graphique, nous utilisons un GNN et un MLP en cascade pour effectuer les prédictions. Ce modèle permet de prendre en entrée le graphe encodant l'instance associée à une insertion et de prédire si l'ajout d'un patient au plan d'acheminement est possible. Le GNN crée une représentation latente des nœuds constituant l'entrée du MLP, dont la tâche est une classification binaire déterminant si l'insertion associée au graphe d'entrée est possible ou non. Lorsque les prédictions de chacune des insertions ont été obtenues, on détermine, si cela est possible, l'insertion optimale pour le service parmi les différentes prédictions.

La figure 3.4 représente, de façon abstraite, le cheminement du graphe encodant une insertion vers la prédiction associée. Ce mécanisme prédictif permet de fournir des réponses rapides, réduisant ainsi la dépendance vis-à-vis des solveurs d'optimisation traditionnels. Il est cependant crucial d'entraîner ce modèle de manière adéquate pour maintenir la qualité des insertions communiquées au patient, sous peine de diminuer le nombre de patients servis.

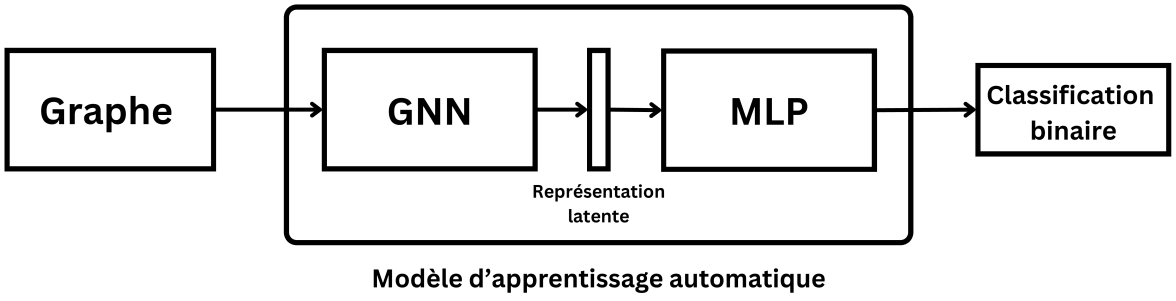


FIGURE 3.4 Visualisation abstraite du modèle d'apprentissage automatique

CHAPITRE 4 DESCRIPTION DE LA MÉTHODOLOGIE

Les sections suivantes approfondissent la méthodologie, en commençant par la modélisation du problème sous forme de graphe. Nous décrivons ensuite le modèle d'apprentissage automatique et son processus d'entraînement. Rappelons que nous nous concentrons sur la résolution du problème **P** défini à la section 3.2. L'intégration de l'apprentissage automatique dans ce type de problème présente des défis uniques, car l'espace de solution est à haute dimension et les dépendances structurelles peuvent entraver l'efficacité de l'apprentissage.

La méthodologie utilisée dans cette recherche est inspirée de [82], qui propose de modéliser des scènes de trafic routier en utilisant un graphe hétérogène intégrant des agents dynamiques et des informations statiques à l'aide de couches de convolution de graphes en cascade pour encoder la scène. Cette modélisation utilise ensuite un GNN comme encodeur et un autre modèle d'apprentissage automatique, spécifique à la tâche demandée, comme décodeur. Cette idée est particulièrement adaptée aux problèmes de tournées de véhicules, car elle permet de modéliser les relations entre les différents éléments du réseau et de prendre en compte les caractéristiques des arêtes. Cette capacité en fait un outil pertinent pour le contexte de cette recherche.

Dans ce travail, nous utilisons les GNN, une classe de modèles intrinsèquement bien adaptés aux données pouvant être modélisées sous forme de graphe hétérogène. En représentant les instances de notre problème sous forme d'hétérographe, où les nœuds correspondent aux agents et les arêtes représentent les relations entre ces derniers, notre modèle est capable d'apprendre les dépendances relationnelles au sein de la structure du problème **P**. Ces dépendances sont importantes, car elles représentent soit des contraintes à respecter, soit des informations clés pour la résolution du problème.

4.1 Modélisation sous forme de graphe

La solution d'une instance $\mathbf{p}$ du problème **P** est modélisée sous la forme d'un graphe hétérogène $G_{\mathbf{p}} = (V, E)$, où les sommets V et les arêtes E encodent les éléments essentiels et les relations inhérentes à la structure du problème. Cette section définit les différents ensembles utiles à la modélisation.

L'hétérographe $G_{\mathbf{p}} = (V, E)$ est constitué de l'ensemble des nœuds V et de l'ensemble des arêtes E. Les ensembles V et E se décomposent en sous-ensembles distincts, chacun caractérisant les différents types de nœuds et d'arêtes :

— **Ensemble des nœuds** V : Les nœuds représentent les différents agents du problème, définis dans des sous-ensembles distincts. Soit A l'ensemble des patients ayant des rendez-vous programmés à honorer au cours de la journée, R l'ensemble des soignants se déplaçant pour servir les patients, et T l'ensemble des étapes temporelles discrètes, chacune représentant un intervalle de 30 minutes au cours de la journée de service, simulant la discrétisation AM/PM/Soirée utilisée dans un outil hebdomadaire.

— **Ensemble des arêtes** E : Les arêtes représentent les différentes relations entre nos agents. Le type de relation d'une arête $(u, v) \in E$ est déterminé à partir du type du nœud source u et du type du nœud cible v. Les différents sous-ensembles sont définis plus précisément dans la section suivante.

4.1.1 Structure du graphe hétérogène

La figure 4.1 présente un exemple d'encodage de la solution d'une instance distinguant les nœuds (agents) et les arêtes (relations). L'ensemble des nœuds $V = \{(v_a)_{a\in A}, (v_r)_{r\in R}, (v_t)_{t\in T}\}$ se compose donc de trois types de nœuds distincts :

— v_a pour chaque patient a dans le sous-ensemble A

— v_r pour chaque membre du personnel soignant r dans le sous-ensemble R

— v_t pour chaque étape temporelle de 30 minutes t dans le sous-ensemble T, représentant la discrétisation temporelle de la disponibilité des services et donc les insertions possibles.

L'importance de différencier les types de nœuds réside dans les informations associées à chacun d'eux. Par exemple, les intervalles de temps sont associés au nombre de services réalisés sur une période de 30 minutes, sans être liés aux contraintes de demande et de capacité. Ces contraintes concernent spécifiquement les patients et les soignants. Les caractéristiques d'entrée du GNN propres à chaque nœud diffèrent donc selon son type.

Les arêtes sont également séparées en différents sous-ensembles, chacun ayant des propriétés distinctes : $E = \{E_{A-A}, E_{R-R}, E_{T-T}, E_{A-R}, E_{R-A}, E_{A-T}, E_{T-A}, E_{R-T}, E_{T-R}\}$, définissant les relations entre les agents de chaque type de nœud.

— E_{A-A} : Relations entre les patients, exprimées en distance temporelle (en minutes plutôt qu'en mètres), en plus du temps de service du patient source.

— E_{R-R} : Relations entre les soignants, représentant le nombre de patients compatibles en commun.

— E_{T-T} : Relations entre les étapes temporelles, préservant l'ordre temporel et représentant le temps écoulé entre les différents intervalles.

— E_{A-R} et E_{R-A} : Relations entre les patients et les soignants, encodant leur compati-

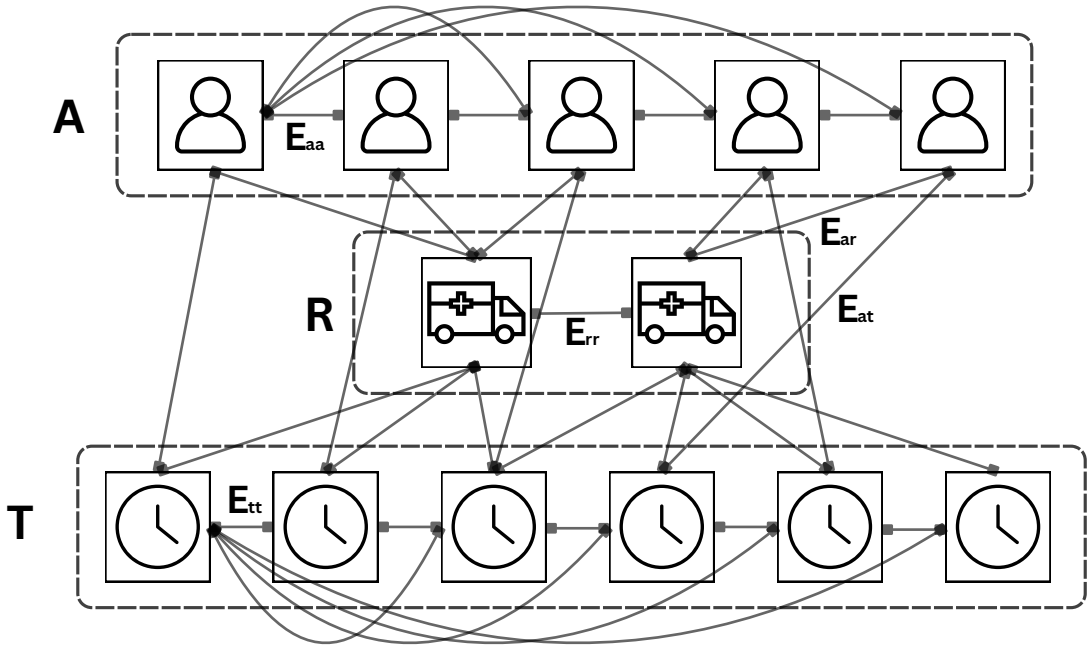


FIGURE 4.1 Encodage de la solution d'une instance sous forme de graphe

bilité ainsi que leur distance temporelle.

— E_{A-T} et E_{T-A} : Relations entre les patients et les intervalles de temps, représentant l'insertion choisie et les patients pris en charge durant un certain intervalle.

— E_{R-T} et E_{T-R} : Relations entre les soignants et les intervalles de temps, modélisant la contrainte de disponibilité, les temps de travail et les services effectués.

Comme pour les nœuds, chaque type d'arête est associé à des caractéristiques décrivant la nature spécifique de la relation qu'il encode. Un aspect essentiel de cette représentation graphique est que la présence d'une arête transmet également des informations. Par exemple, la présence d'une arête E_{A-R} et d'une arête E_{R-A} indique qu'un patient est compatible avec un membre du personnel soignant ; sinon, l'arête ne serait pas présente. Cela permet d'encoder explicitement certaines contraintes via les caractéristiques, comme les contraintes de demande et de capacité, mais aussi implicitement certaines contraintes, comme ici pour la contrainte de compatibilité.

4.1.2 Caractéristiques de l'hétérographe

Lorsqu'un graphe est introduit dans notre modèle d'apprentissage automatique, ce dernier exploite deux éléments : sa structure, qui fournit des informations implicites, et ses caractéristiques, qui apportent des informations explicites. Les caractéristiques correspondent aux

valeurs associées à un nœud ou à une arête, permettant de reconstruire les contraintes ou de transmettre les informations utiles à la résolution de l'instance $\mathbf{p}$. Par exemple, pour un patient, une information importante pourrait être sa demande, et pour un membre du personnel soignant, sa capacité.

Étant donné que nous utilisons des graphes hétérogènes, le nombre de caractéristiques accompagnant un nœud ou une arête dépend directement du type de l'élément associé. Les différentes caractéristiques du graphe sont définies comme suit : soit $h_v \in \mathbb{R}^{d_1}, v \in V$ et $h_e \in \mathbb{R}^{d_2}, e \in E$, les caractéristiques associées aux nœuds et aux arêtes du graphe, avec d_1 et d_2 les tailles des vecteurs, dépendant du type de nœud et d'arête considéré. Nous définissons h_{v-i} la i-ème caractéristique associée au nœud v. Toutes les caractéristiques associées aux éléments du graphe sont listées dans le tableau 4.1.

En suivant la convention, h_{v_t-3} avec $v_t \in T$ correspond au nombre de patients servis durant v_t dans la solution actuelle de l'instance du problème. En encodant la solution de l'instance $\mathbf{p}$ sous forme de graphe hétérogène $G_{\mathbf{p}}$, il est possible de reconstruire chaque contrainte à partir des caractéristiques et de la structure de ce dernier. La contrainte de double compatibilité se reconstruit explicitement avec h_{v_a-3}, h_{v_a-4}, h_{v_r-3}, h_{v_r-4} où $v_a \in A$ et $v_r \in R$, et se complète implicitement via la présence d'une arête entre un patient et un membre du personnel soignant si ces deux agents sont compatibles. Cette modélisation permet à notre modèle de préserver la structure du problème, lui permettant ainsi d'apprendre à le résoudre par la suite.

4.1.3 Normalisation des caractéristiques

Un dernier point clé concernant la création des caractéristiques est leur normalisation. Des données normalisées permettent à un modèle d'apprentissage automatique d'apprendre de manière plus efficace et plus stable [83].

Nos contraintes, et donc nos caractéristiques, utilisent des unités distinctes. Une normalisation « naïve », où toutes les caractéristiques sont divisées par la valeur maximale parmi elles, n'est pas applicable dans notre cas. Les caractéristiques sont donc normalisées « par groupes », avec plusieurs groupes différents regroupant les caractéristiques partageant une même unité ou appartenant à une contrainte commune. Soit $\mathcal{G}$ un groupe de caractéristiques devant être normalisées ensemble, et soit h_i une caractéristique appartenant à ce groupe. La normalisation s'effectue comme suit :

$$h_i^{\text{norm}} = \frac{h_i}{\max\limits_{h_j \in \mathcal{G}} h_j} \qquad \forall h_i \in \mathcal{G}$$

où $\max\limits_{h_j \in \mathcal{G}} h_j$ représente la plus grande valeur parmi toutes les caractéristiques du groupe $\mathcal{G}$.

TABLEAU 4.1 Caractéristiques de l'hétérographe

$v \in \ldots$	h_v inclut
A	Demande de type 1 (d_i^1) Demande de type 2 (d_i^2) # compétences (f_i^1) # affinités (f_i^2) Taille de la fenêtre temporelle
R	Capacité de type 1 (C_i^1) Capacité de type 2 (C_i^2) # compétences (f_i^1) # affinités (f_i^2) Taille de la fenêtre temporelle
T	# patients disponibles # soignants disponibles # patients servis Progrès dans la journée $\in [0, 1]$

$e \in \ldots$	h_e inclut
E_{A-A}	Temps de trajet ($t_{i,j}$) Temps de service (l_i)
E_{R-R}	# patients communs
E_{T-T}	Différence de temps entre t_1 et t_2
E_{R-A}	Temps de trajet ($t_{i,j}$)
E_{A-R}	Temps de trajet ($t_{i,j}$) Temps de service (l_i)
E_{A-T} E_{T-A}	a est servi dans t (booléen) Taille de la fenêtre durant t
E_{R-T} E_{T-R}	r travaille durant t (booléen) Taille de la fenêtre durant t

Cette normalisation garantit que toutes les caractéristiques d'un même groupe sont mises à l'échelle selon leur groupe respectif et que les dépendances entre ces caractéristiques sont conservées. Ainsi, la caractéristique ayant la plus grande valeur du groupe est ramenée à 1, et toutes les autres sont comprises entre 0 et 1. Cette méthode permet d'éviter les déséquilibres causés par des unités ayant des échelles différentes.

Notons v_a un nœud appartenant au sous-ensemble A, v_r pour le sous-ensemble R et v_t pour T. De même, pour les arêtes, nous notons e_{a-a} pour le sous-ensemble E_{A-A}, e_{r-t} pour E_{R-T}, et réciproquement pour les autres sous-ensembles. Nous définissons ci-dessous les trois groupes de normalisation :

- Les contraintes de demande, de capacité et de compatibilité étant liées entre patients et soignants, les caractéristiques h_{v_a-i} et h_{v_r-i} $\forall i \in \{1, 2, 3, 4\}$ forment des paires interdépendantes, car elles sont contraintes ensemble. Elles constituent ainsi quatre groupes indépendants de deux caractéristiques.
- La dépendance de nombre concerne toutes les caractéristiques représentant un nombre d'agents, qu'il s'agisse de patients ou de soignants. Les caractéristiques h_{v_t-i} $\forall i \in \{1, 2, 3\}$ et $h_{e_{r-r}-1}$ forment un groupe de quatre caractéristiques normalisées ensemble.
- La dépendance temporelle concerne toutes les caractéristiques dont l'unité de mesure est la minute. Ces caractéristiques apparaissent dans différentes contraintes et parmi les informations utiles à la résolution du problème. Les caractéristiques h_{v_a-5}, h_{v_r-5}, $h_{e_{a-a}-1}$, $h_{e_{a-a}-2}$, $h_{e_{t-t}-1}$, $h_{e_{r-a}-1}$, $h_{e_{a-r}-1}$, $h_{e_{a-r}-2}$, $h_{e_{t-a}-2}$, $h_{e_{a-t}-2}$, $h_{e_{t-r}-2}$ et $h_{e_{r-t}-2}$ forment le dernier groupe de normalisation.

Toutes les autres caractéristiques non mentionnées dans ces trois groupes sont indépendantes et ne nécessitent pas de normalisation. Cela s'explique par leur nature, étant déjà comprises entre 0 et 1 ou étant booléennes. Ces caractéristiques incluent h_{v_t-4}, qui correspond à l'avancement dans la journée, compris entre 0 et 1, ainsi que $h_{e_{a-t}-1}$ et $h_{e_{t-a}-1}$, qui valent 1 si le patient a est pris en charge durant l'intervalle de temps t, et 0 sinon. De même, $h_{e_{r-t}-1}$ et $h_{e_{t-r}-1}$ valent 1 si le membre du personnel soignant r travaille durant l'intervalle t, et 0 sinon.

4.1.4 Graphe d'insertion

La notion de graphe d'insertion représente le type de graphe sur lequel le modèle s'entraîne et prédit la faisabilité des possibilités d'insertion. Une insertion est définie comme un intervalle de 30 minutes durant lequel un nouveau patient peut être inséré dans un planning déjà optimisé. Ce planning optimisé représente la solution de $\mathbf{p}$, une instance du problème $\mathbf{P}$. La modélisation sous forme de graphe hétérogène $G_{\mathbf{p}}$ de ce planning optimisé sert de base à la création du graphe d'insertion. Le graphe $G_{\mathbf{p}}$ contient toutes les informations de l'instance $\mathbf{p}$ résolue avant l'ajout du nouveau patient.

Plusieurs graphes d'insertion $G_{\mathbf{p},\texttt{ins}}$ sont construits à partir de $G_{\mathbf{p}}$ et à partir des informations du nouveau patient à ajouter au planning pour chaque insertion `ins` possible. Si le nouveau patient possède trois insertions possibles, notées $\texttt{ins}_1$, $\texttt{ins}_2$ et $\texttt{ins}_3$, alors trois graphes d'insertion, $G_{\mathbf{p},\texttt{ins}_1}$, $G_{\mathbf{p},\texttt{ins}_2}$ et $G_{\mathbf{p},\texttt{ins}_3}$, sont créés. Les informations liées à la résolution du planning actuel étant déjà encodées dans $G_{\mathbf{p}}$, le nouveau patient est ajouté au graphe en imposant son service à un certain intervalle de temps pour créer un graphe d'insertion. Chaque insertion génère un graphe d'insertion distinct représentant l'ajout du nouveau patient à l'intervalle de temps associé à l'insertion considérée.

Cette méthodologie permet de conserver les informations du planning actuel tout en ajoutant celles du nouveau patient. L'ajout d'un nouveau nœud patient à $G_{\mathbf{p}}$ et l'adaptation des caractéristiques du graphe permettent d'obtenir le graphe d'insertion associé. La figure 4.2 met en évidence le greffage du nouveau patient sur la base du schéma présenté dans la figure 4.1 représentant $G_{\mathbf{p}}$, pour obtenir $G_{\mathbf{p},\texttt{ins}}$. En rouge, $v_{a,new} \in A$ illustre le nouveau nœud patient. En vert, certaines des nouvelles arêtes ont été ajoutées. En violet, $v_{t,new} \in T$ illustre une situation où l'insertion du patient se fait dans un intervalle de temps qui n'est pas encore utilisé par les autres patients. Dans ce cas, il est nécessaire d'ajouter un nouveau nœud $v_{t,new} \in T$ ainsi que les arêtes associées. Si l'intervalle de temps considéré est déjà utilisé par un des patients du planning, l'ajout d'un nœud $v_{t,new}$ n'est pas nécessaire, et $v_{a,new}$ est directement relié au nœud correspondant au nœud intervalle déjà présent dans $G_{\mathbf{p}}$.

L'ensemble des nœuds V du graphe d'insertion $G_{\mathbf{p},\texttt{ins}}$ est défini par l'union des nœuds du

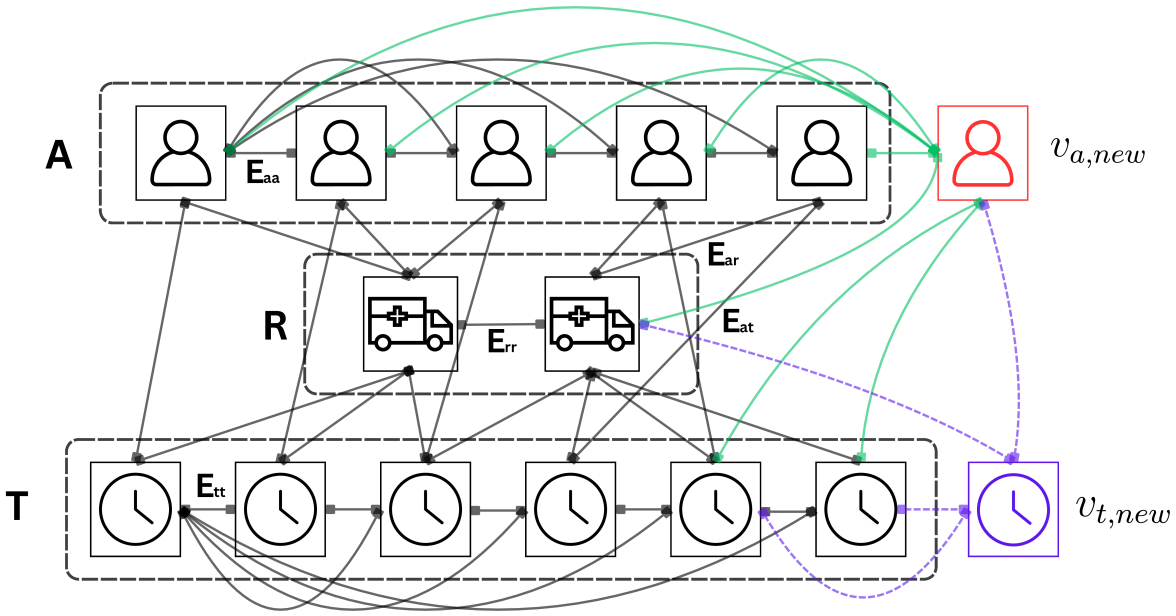


FIGURE 4.2 Encodage d'un graphe d'insertion

graphe $G_{\mathbf{p}}$, du nœud représentant le nouveau patient $v_{a,new}$ et, si nécessaire, du nœud associé à l'intervalle de temps inutilisé $v_{t,new}$.

Mathématiquement, l'ensemble des nœuds du graphe d'insertion est défini par :

$$V_{G_{\mathbf{p},\mathtt{ins}}} = V_{G_{\mathbf{p}}} \cup \{v_{a,new}\} \cup \{v_{t,new}\}$$

avec $v_{a,new} \in A$ le nœud représentant le nouveau patient et $v_{t,new} \in T$ le potentiel nœud représentant l'intervalle de temps encore non utilisé, comme expliqué ci-dessus.

L'ensemble des arêtes E suit une logique identique, où $E_{G_{\mathbf{p}}} \subset E_{G_{\mathbf{p},\mathtt{ins}}}$. Les nouvelles arêtes associées au nouveau patient et à son intervalle de temps sont ajoutées à $E_{G_{\mathbf{p}}}$ pour obtenir $E_{G_{\mathbf{p},\mathtt{ins}}}$.

Pour déterminer l'insertion d'un nouveau patient dans le planning actuel, le modèle d'apprentissage automatique prédit la faisabilité de l'insertion `ins` associée au graphe $G_{\mathbf{p},\mathtt{ins}}$. L'algorithme génère les graphes d'insertion pour chaque insertion possible du nouveau patient et infère leur prédiction respective. À partir de ces prédictions, notre algorithme est alors capable de communiquer une réponse au patient.

4.2 Création des données d'apprentissage

Nous utilisons les concepts expliqués précédemment pour créer les graphes d'insertion ainsi que les sorties associées devant être apprises par le modèle. Pour ce faire, nous exploitons les 1137 instances contenues dans les données historiques de Fastercom et recréons les insertions possibles au sein de plannings déjà optimisés. L'objectif est de reproduire au mieux les situations réelles auxquelles le modèle sera confronté.

4.2.1 Algorithme de création des données

Dans un premier temps, nous considérons une journée à moitié remplie comme point de départ. Ce choix est fait car l'insertion des 50 premiers pour cent des patients dans un planning quasiment vide constitue une situation non contrainte et facile à résoudre.

Ensuite, les 50 autres pour cent de patients sont ajoutés un à un dans le planning en testant individuellement leurs insertions possibles. Pour chaque insertion, l'instance est résolue avec *OR-Tools* afin d'en mesurer la qualité et la faisabilité. À chaque résolution, un booléen $Q_{\texttt{ins}}$ est associé à l'insertion `ins` testée, défini par l'équation :

$$Q_{\texttt{ins}} = \begin{cases} 1, & \text{si tous les patients sont pris en charge} \\ 0, & \text{sinon} \end{cases}$$

Un problème fréquent est que l'ajout du nouveau patient empêche le service d'un autre patient déjà planifié. Si tel est le cas, l'insertion est considérée comme infaisable et $Q_{\texttt{ins}} = 0$.

En parallèle, les graphes d'insertion sont construits afin de les faire correspondre à la faisabilité de l'insertion. Premièrement, nous créons le graphe $G_{\mathbf{p}}$ lié au planning actuel, servant de base à la création des graphes d'insertion. Ensuite, la fenêtre de temps du nouveau patient est décomposée en intervalles de 30 minutes afin d'obtenir toutes les insertions possibles. À partir de ces intervalles, les différents graphes d'insertion $G_{\mathbf{p},\texttt{ins}}$ sont générés. Nous obtenons finalement une multitude de couples $(G_{\mathbf{p},\texttt{ins}}, Q_{\texttt{ins}})$. Ces couples sont ensuite utilisés pour l'apprentissage du modèle, qui doit prédire si l'insertion est possible ou non à partir du graphe d'insertion associé. Le processus de création des données est décrit dans le pseudo-code défini dans l'algorithme 1. L'algorithme suit le processus décrit en commençant par créer un planning initial contenant 50% des patients de l'instance. Ce planning est généré en utilisant la méthode classique de Fastercom, qui consiste à tester chaque insertion possible via un solveur pour chaque nouveau patient et à sélectionner la meilleure insertion.

Ensuite, l'algorithme ajoute les patients restants un par un, en créant pour chacun d'eux les

Algorithm 1 Génération des données

Require: $I = \{i_1, i_2, \ldots, i_{|H|}\}$: Ensemble des instances historiques (H)
Require: VRP_solver : Solveur utilisé par la compagnie (ici OR-Tools par hypothèse)
Require: $data\_ML$: Dictionnaire vide pour stocker les données
for chaque instance i appartenant à I **do**
    $C_1 \leftarrow$ Sélection aléatoire de 50% des patients de l'instance i
    $C_2 \leftarrow$ Les autres 50 pour cent des patients
    $i_{current} \leftarrow$ Génération du planning des patients dans C_1 à l'aide d'appels à VRP_solver
    for chaque patient c appartenant à C_2 **do**
        Résolution du problème de routage associé à $i_{current}$ via VRP_solver
        $G_{i_{current}} \leftarrow$ Génération du graphe hétérogène correspondant à la solution de $i_{current}$
        $timesteps \leftarrow$ Identification des insertions possibles dans la fenêtre du patient c
        for chaque insertion t dans $timesteps$ **do**
            $G_{i_{current},t} \leftarrow$ Création du graphe d'insertion associé à t à partir de $G_{i_{current}}$
            $i_{new} \leftarrow$ Ajout du patient c à l'insertion t dans $i_{current}$
            $Q_t \leftarrow$ Evaluation du booléen à partir de la résolution de i_{new} via VRP_solver
            Ajout du couple ($G_{i_{current},t}$, Q_t) à data_ML
        end for
        $i_{current} \leftarrow$ Mise à jour du planning en intégrant le patient c à la meilleure insertion
    end for
end for
return $data\_ML$: l'ensemble des données

couples de données graphe/booléen. À mesure que des patients sont ajoutés, les insertions deviennent plus complexes car les degrés de liberté diminuent. L'objectif est que le modèle soit capable de répondre à des situations où l'insertion est facile, avec une journée peu remplie, mais qu'il soit également capable de répondre à des situations où l'insertion est complexe, voire même impossible, avec une journée presque pleine.

4.2.2 Répartition des données

Nous utilisons un ensemble de données divisé en trois parties non égales selon une répartition 70/15/15 : 70% pour l'apprentissage, 15% pour la validation et 15% pour le test. La répartition est effectuée en découpant les données à partir des instances d'origine et non pas au niveau des couples graphe/booléen. Ce choix permet d'éviter les biais. Si la séparation avait été effectuée après la génération des données, les informations de certains patients auraient pu apparaître à la fois dans les ensembles de validation/test et dans l'ensemble d'apprentissage. En segmentant au niveau des instances et étant donné que les journées d'opération sont indépendantes, nous garantissons qu'aucune donnée ne soit partagée entre les différents ensembles.

4.3 Formulation du modèle d'apprentissage automatique

Le modèle proposé est une architecture hybride conçue pour traiter des données graphiques hétérogènes en entrée et résoudre une tâche de classification en sortie. Cette approche est inspirée de [82], utilisant un GNN comme encodeur et un autre modèle d'apprentissage automatique, spécifique à la tâche demandée, comme décodeur. Dans notre cas, ce décodeur est un MLP. Nous utilisons ainsi un modèle combinant ces deux types d'architectures en cascade. L'objectif du modèle est de prédire si une insertion est possible à partir d'un graphe d'insertion reçu en entrée. Le GNN traite le graphe d'insertion $G_{\mathbf{p},\texttt{ins}}$ en entrée en générant une représentation latente servant d'entrée à un MLP, lequel produit une valeur de sortie indiquant si l'insertion `ins` est faisable ou non.

Ces deux éléments ont chacun leur spécificité. Le réseau de neurones pour graphes est capable d'apprendre à partir des relations et ajuste ses poids en fonction de chaque type de nœud et d'arête, l'hétérogénéité du graphe permettant l'apprentissage de poids distincts et indépendants pour chacun des types présents. En sortie du GNN, une représentation latente est calculée en appliquant une agrégation de type « moyenne » sur les trois types de nœuds. Le MLP utilise cette représentation et applique plusieurs couches d'apprentissage profond pour produire la valeur de prédiction.

4.3.1 Formalisation du GNN

Pour le GNN, nous utilisons les *EdgeGATConv*, une classe définie dans la librairie DGL [84]. L'*EdgeGATConv* est une couche de graphe attentionnelle qui permet d'intégrer les caractéristiques des arêtes dans le modèle. Cette couche, issue de [82], est particulièrement adaptée aux problèmes de tournées de véhicules, car elle permet de modéliser les relations entre les différents agents tout en prenant en compte les caractéristiques des arêtes.

La notion d'attention dans les GNN repose sur un mécanisme permettant de pondérer dynamiquement l'importance des voisins d'un nœud lors de l'agrégation des valeurs latentes du modèle. Cette approche est introduite dans les *Graph Attention Networks* (GAT) [85, 86], où l'attention apprend à ajuster les poids des connexions entre les nœuds en fonction de leur pertinence, renforçant ainsi les relations les plus significatives pour l'apprentissage. Les GAT emploient une attention multi-têtes afin de capturer les relations entre les nœuds, permettant ainsi une meilleure prise en compte de la structure locale du graphe sans nécessiter de connaissance a priori sur celle-ci.

Le GNN de notre modèle traite les caractéristiques des nœuds et des arêtes de nos graphes d'insertion, à l'aide de deux couches superposées d'*EdgeGATConv* appliquées à chaque type

de relation. Pour agréger les caractéristiques latentes issues des différentes relations entre chaque couche, l'opération de somme a été choisie ($\square = \sum$ dans la section 3.4) afin de garantir que l'influence de chaque type de nœud et de relation soit préservée. En sortie, le GNN agrège les représentations latentes de chaque type de nœud et les concatène en un tenseur en effectuant une moyenne sur les nœuds de chaque type. Ce tenseur d'entrée du MLP est de taille $K \sum_{p=1}^{3} n_{f_p}$, où K est le nombre de têtes d'attention utilisées et $\sum_{p=1}^{3} n_{f_p}$ le nombre de caractéristiques latentes associées aux trois types de nœuds.

La formulation mathématique des *EdgeGATConv* est résumée par les équations suivantes. Nous définissons le tenseur $\mathbf{\Theta}_{\mathbf{s},r}$ désignant les matrices de poids apprenables permettant d'obtenir les caractéristiques latentes du nœud à mettre à jour, à partir du nœud source s, de ses voisins r et notons les caractéristiques des arêtes e. La mise à jour du nœud v_i en tenant compte des nœuds voisins connectés par le type de relation r est donnée par :

$$\mathbf{v}'_{i,r} = \ EdgeGAT_{\ r}(\mathbf{v}_i) = \mathbf{\Theta}_{\mathbf{s},r} \cdot \mathbf{v}_i + \Big\|_{k=1}^{K} \left(\sum_{j \in \mathcal{N}_r(v_i)} \alpha^k_{j,r,i} \left(\mathbf{\Theta}^k_{\mathrm{n},r} \cdot \mathbf{v}_j + \mathbf{\Theta}^k_{\mathrm{e},r} \cdot \mathbf{e}_{j,r,i} \right) \right)$$

Le nombre de têtes d'attention est donné par K et $\|$ est l'opérateur de concaténation. De plus, une connexion résiduelle $\mathbf{\Theta}_{\mathrm{s},r} \cdot \mathbf{v}_i$ est ajoutée pour améliorer la propagation du gradient et éviter un lissage excessif. La fonction d'activation utilisée entre les couches est une $ReLU$ [87]. Les poids d'attention sont obtenus par :

$$\alpha^k_{j,r,i} = softmax_{r,i}(LeakyReLU(\ \mathbf{a}_r^{k^T} \left[\mathbf{\Theta}^k_{\mathrm{n},r} \cdot \mathbf{v}_i \Big\| \mathbf{\Theta}^k_{\mathrm{n},r} \cdot \mathbf{v}_j \Big\| \mathbf{\Theta}^k_{\mathrm{e},r} \cdot \mathbf{e}_{j,r,i} \right]))$$

avec $\mathbf{a}_r$ un vecteur apprenable et $softmax_{\mathrm{r,i}}$ représentant la normalisation sur toutes les arêtes entrantes du nœud i connectées par la relation r.

En utilisant les *EdgeGATConv* plutôt que les *EGATConv* introduites dans [88], seules les caractéristiques nodales sont mises à jour. Ainsi, il n'y a pas de représentation latente pour les arêtes, ce qui implique que les caractéristiques initiales des arêtes restent inchangées dans la deuxième couche du GNN, contrairement aux *EGATConv* qui génèrent des caractéristiques latentes pour les nœuds et les arêtes.

Les deux raisons principales justifiant ce choix sont les suivantes. Premièrement, les caractéristiques des arêtes représentent des informations statiques du modèle, telles que les distances, l'avancement dans la journée ou les services effectués. Il est donc pertinent de conserver ces informations inchangées pour les deux entrées des couches du GNN. Deuxièmement, en pratique, les résultats obtenus avec les *EdgeGATConv* se sont révélés légèrement supérieurs à ceux obtenus avec les *EGATConv*, tout en étant plus simples à implémenter.

4.3.2 Formalisation du MLP

Le réseau neuronal profond proposé est un perceptron multicouche composé de sept couches denses successives. L'architecture suit un motif d'expansion-contraction, visible à la figure 4.3, qui permet une transformation progressive des caractéristiques d'entrée vers la prédiction finale.

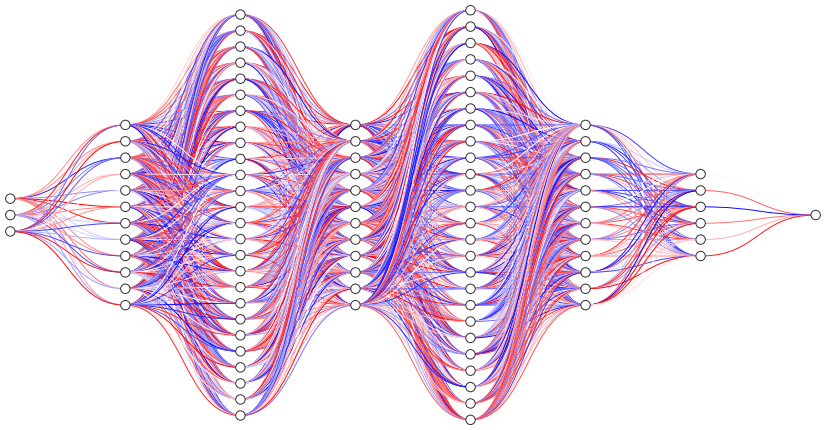


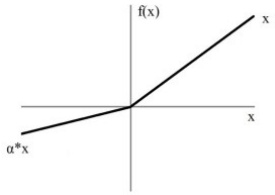
FIGURE 4.3 Visualisation du MLP (échelle 1 :10)

Chacune des couches cachées est composée d'une transformation linéaire suivie d'une normalisation et d'une fonction d'activation. Soient W_k et b_k respectivement les poids et les biais, BN_k l'opération de normalisation (*batch norm* [89]) associée à la couche cachée k, et σ la fonction d'activation. En définissant les sorties latentes de la couche k comme z_k et les sorties avant activation comme a_k, les couches cachées sont décrites à l'aide des équations suivantes :

$$a_k = \mathrm{BN}_k(W_k h_{k-1} + b_k)$$
$$z_k = \sigma(a_k)$$

La fonction d'activation utilisée est la *Leaky Rectified Linear Unit*, introduite dans [90], et elle est définie par :

$$\sigma(x) = \begin{cases} x, & \text{si } x \geq 0 \\ \alpha x, & \text{si } x < 0 \end{cases}$$



où le paramètre α est fixé à 0.1 dans notre modèle. Cette fonction est utilisée dans toutes les couches, sauf dans la dernière, où la fonction identité $\sigma(x) = x$ est appliquée. La fonction identité est choisie afin d'éviter une double application de la sigmoïde, dans le cadre de

l'utilisation de la fonction de perte d'entropie croisée binaire avec logit. Pour obtenir les prédictions, une sigmoïde doit néanmoins être appliquée a posteriori sur la valeur de sortie, afin de pouvoir l'interpréter comme un score de confiance relatif à la faisabilité de l'insertion du patient dans l'intervalle de temps correspondant. Si cette valeur est inférieure à 0.5, le modèle prédit que l'insertion est impossible. Plus la valeur est proche de 1, plus le modèle est confiant quant à la faisabilité de l'insertion associée. Un résumé du modèle d'apprentissage automatique est présenté dans le tableau 4.2.

Tableau 4.2 Résumé du modèle d'apprentissage automatique

Partie	Couches cachées	Nombre de caractéristiques/neurones	Activation
GNN (3 têtes)	2	{5, 5, 4}, {5, 5, 5}, {3, 3, 3}	ReLU
MLP	4	{27, 128, 256, 128, 256, 128, 64, 1}	Leaky-ReLU

Il est à noter que les termes « petite » et « grande » architecture seront utilisés dans la section 4.4. La « petite » architecture correspond à celle décrite dans le tableau, avec trois têtes d'attention. Pour la « grande » architecture, les tailles de toutes les couches cachées du MLP sont multipliées par deux et nous considérons l'utilisation de quatre têtes d'attention.

4.4 Entraînement du modèle

L'apprentissage de ce réseau de neurones est réalisé par lots sur l'ensemble d'entraînement décrit dans la section 4.2.1. L'ensemble d'apprentissage est utilisé pour la mise à jour des poids et la phase d'apprentissage est contrôlée à l'aide de l'ensemble de validation. Pour valider la capacité du modèle à apprendre la tâche demandée, nous avons procédé à une validation progressive de son fonctionnement : d'abord en vérifiant qu'il pouvait apprendre sur un petit ensemble de données, puis en testant sa capacité à mémoriser l'intégralité de l'ensemble de données.

4.4.1 Algorithme d'apprentissage

Après validation de son fonctionnement, nous remarquons que le modèle présente une difficulté à généraliser et montre des signes de surapprentissage (*overfitting*). Plusieurs mécanismes sont alors introduits pour stabiliser et améliorer l'apprentissage. Ces mécanismes incluent l'initialisation des poids, un ordonnanceur de taux d'apprentissage, l'identification des plateaux, du *dropout*, etc.

Les fonctions d'initialisation testées sont Xavier [91] et Kaiming He [92].

$$\text{Xavier} : W \sim U\left(-\frac{\sqrt{6}}{\sqrt{n_{\text{in}}+n_{\text{out}}}}, \frac{\sqrt{6}}{\sqrt{n_{\text{in}}+n_{\text{out}}}}\right) \qquad \text{Kaiming He} : W \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}}}\right)$$

où n_{in} et n_{out} désignent respectivement le nombre de neurones entrants (couche précédente) et le nombre de neurones sortants (couche actuelle).

La fonction de perte employée pour l'entraînement est l'entropie croisée binaire avec logit, pondérée différemment pour les classes positives et négatives. Elle est définie par l'équation suivante :

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left[w^{+} y_i \log(\sigma_s(z_i)) + w^{-}(1-y_i)\log(1-\sigma_s(z_i))\right]$$

où : w^{+} est le poids attribué aux échantillons positifs ($y_i = 1$), w^{-} est le poids attribué aux échantillons négatifs ($y_i = 0$) et $\sigma_s(z_i) = \frac{1}{1+e^{-z_i}}$ est la fonction sigmoïde appliquée aux logits z_i. En fusionnant la sigmoïde et la fonction de perte en une seule couche, nous exploitons l'astuce du *log-sum-exp* [93] afin d'améliorer la stabilité numérique. Cette technique permet de réécrire des expressions du type $\log(e^x + e^y)$ pour éviter les erreurs d'*overflow*, en utilisant la forme $\max(x, y) + \log(1 + e^{-|x-y|})$. Notre algorithme d'apprentissage est décrit sous forme de pseudo-code dans l'algorithme 2.

Algorithm 2 Algorithme d'apprentissage

Require: M : Modèle à entraîner
Initialisation des poids et des biais du modèle M
for chaque époque de 1 à 5000 **do**
    Mélange des données d'entraînement et division en lots $\{B_i\}$
    for chaque lot B_i **do**
        out ← Propagation avant du modèle M sur le lot B_i
        loss ← Calcul de la perte entre *out* et les booléens associées à B_i
        M ← Rétro-propagation de *loss* et mise à jour des poids et biais de M
    end for
    Évaluation du modèle M sur les données de validation sans mise à jour des poids
    Ajustements dynamiques :
      Réduction du taux d'apprentissage lors de la détection d'un plateau
      Application d'un mécanisme d'arrêt anticipé si aucune amélioration n'est détectée
end for
return Modèle entraîné M

4.4.2 Calibrage des hyper-paramètres

Pour accompagner l'algorithme d'apprentissage, nous calibrons les hyperparamètres utilisés. Une approche classique consiste à tester exhaustivement toutes les combinaisons possibles de valeurs pour les différents hyperparamètres. Le calibrage s'est concentré sur six aspects de l'apprentissage : la taille de l'architecture, l'initialisation des poids, le taux d'apprentissage, la décroissance des poids, le *dropout* et la taille des lots. Passons en revue ces six éléments :

- **La taille de l'architecture** : comme expliqué dans la section 4.3.2, il existe deux configurations, « petite » ou « grande », permettant d'ajuster l'expressivité du modèle.
- **L'initialisation** : varie entre Xavier [91] et Kaiming He [92].
- **Le taux d'apprentissage** (η) : contrôle la vitesse de mise à jour des poids lors de l'optimisation, selon :

$$w_{t+1} = w_t - \eta \nabla L(w_t)$$

  où $\nabla L(w_t)$ est le gradient de la fonction de perte L.
- **La décroissance des poids** [94] (λ) : régularisation visant à pénaliser les poids trop grands afin d'éviter le surapprentissage (*overfitting*). Elle est intégrée à la mise à jour des poids selon :

$$w_{t+1} = w_t - \eta(\nabla L(w_t) + \lambda w_t)$$

  où λ contrôle l'intensité de la pénalisation.
- **Le *dropout*** [95] : consiste à multiplier l'activation z_k par un masque binaire d_k, où chaque élément est tiré indépendamment d'une loi de Bernoulli de paramètre p (probabilité de conserver un neurone). Le *dropout* est appliqué entre les couches cachées du modèle lors de la phase d'apprentissage (avec $p = 1$ lors de l'évaluation). Il est représenté par les équations suivantes :

$$\begin{aligned} a_k &= \mathrm{BN}_k(W_k h_{k-1} + b_k) \\ z_k &= \sigma_k(a_k) \\ d_k &\sim \mathrm{Bernoulli}(p) \\ h_k &= d_k \odot z_k \end{aligned}$$

  où h_k représente le tenseur des caractéristiques latentes de la couche k, et $\odot$ désigne le produit d'Hadamard élément par élément.
- **La taille des lots** : correspond au nombre d'échantillons propagés dans le réseau à chaque itération. Le choix de cette taille influence à la fois la rapidité d'exécution et la vitesse de convergence de l'apprentissage.

Les valeurs explorées dans cette recherche exhaustive sont résumées dans le tableau 4.3.

TABLEAU 4.3 Résumé des valeurs utilisées dans la recherche exhaustive

Architecture	Initialisation	η	λ	*Dropout*	Taille de lot
petite	Xavier	$1e^{-3}$, $8e^{-4}$	$1e^{-4}$	0.2, 0.25, 0.3	32, 64, 128
grande	Kaiming He	$4e^{-4}$, $2e^{-4}$, $1e^{-4}$	$1e^{-5}$	0.35, 0.4, 0.5	256, 512

4.5 Formalisation de l'approche

Cette dernière section revient sur les différentes notions discutées et définies dans les sections précédentes afin d'expliquer, de manière globale, l'approche utilisée pour intégrer l'apprentissage automatique dans l'objectif de réduire les coûts temporels et budgétaires du cadre opérationnel utilisé par des entreprises comme Fastercom.

L'objectif de la méthode est de trouver rapidement la meilleure insertion d'un nouveau patient dans un planning existant, lorsque celle-ci est possible. Pour ce faire, une approche en plusieurs étapes est utilisée pour l'insertion d'un nouveau patient dans le planning optimisé correspondant à la solution de l'instance $\mathbf{p}$. La création du graphe $G_{\mathbf{p}}$ contenant les informations du planning actuel, la prise d'informations en temps réel sur le nouveau patient et la création des graphes d'insertion $G_{\mathbf{p},\texttt{ins}}$ pour chaque insertion `ins` possible à l'intérieur de la fenêtre temporelle du patient, le calcul des prédictions via notre modèle entraîné pour chaque insertion et l'utilisation de ces prédictions pour insérer le patient au moment optimal. Ces étapes sont illustrées à la figure 4.4. Étant donné que nous simulons une semaine dans le cadre des journées d'opérations, notre modèle permet d'identifier le meilleur intervalle de 30 minutes pour le nouveau patient, restreignant ainsi sa fenêtre temporelle à cet intervalle.

Le principe d'utilisation du modèle, qui pourrait être appliqué dans le futur, peut être résumé en trois étapes :

- **Instance vers graphes :** Le graphe $G_{\mathbf{p}}$ est obtenu lors de l'ajout du patient précédent. Il contient les informations des patients et les décisions prises. Ensuite, lors d'une nouvelle demande, l'agent réceptionnant l'appel encode les informations du nouveau patient et l'algorithme décompose sa fenêtre temporelle en un ensemble d'intervalles de 30 minutes. Pour chaque intervalle, un graphe d'insertion est construit en greffant le patient à $G_{\mathbf{p}}$ pour l'insertion considérée. Si la fenêtre de temps couvre x intervalles de 30 minutes, alors x graphes d'insertion sont générés. Par exemple, un patient demandant à être servi entre 10h15 et 11h40 permet quatre insertions : 10h15-10h29, 10h30-10h59, 11h00-11h29 et 11h30-11h40.
- **Graphes vers prédictions :** Les x graphes d'insertion créés sont injectés dans le modèle entraîné, qui génère une prédiction pour chacune des x insertions possibles.

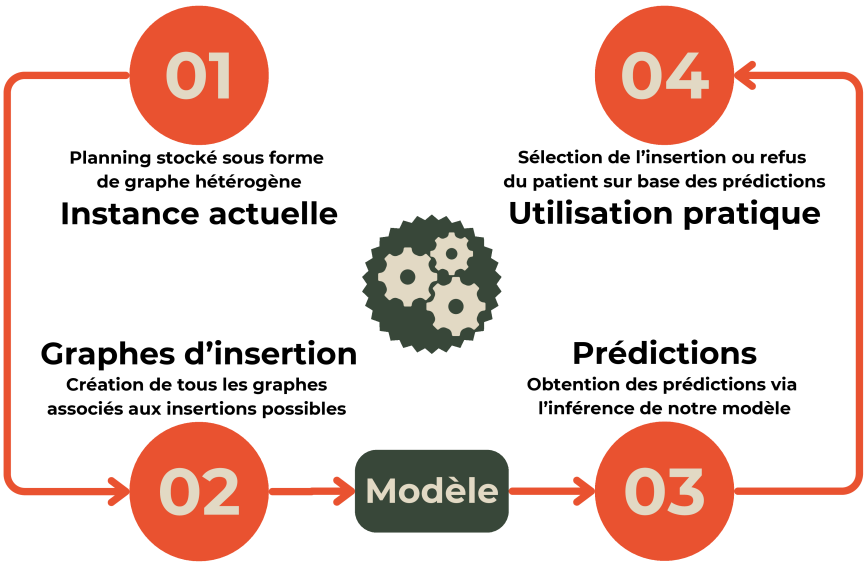


FIGURE 4.4 Utilisation pratique du modèle

- **Prédictions vers utilisation :** Les prédictions permettent de déterminer l'insertion à communiquer au patient par téléphone. Le choix basé sur les prédictions est flexible et on peut accroître la robustesse du modèle si nécessaire. L'utilisation d'appels à un solveur comme *OR-Tools* pour renforcer la robustesse réduit légèrement le gain de temps, mais augmente le nombre de patients servis.

Enfin, l'outil est résumé par le pseudo-code de l'algorithme 3.

Algorithm 3 Outil final basé sur l'apprentissage automatique

Require: M : Modèle entraîné
Require: c : Nouveau patient à insérer
Require: VRP_solver : Solveur utilisé par la compagnie (ici OR-Tools par hypothèse)
Require: ML_choice : Méthode déterminant le choix optimal à partir des prédictions
Require: $G_{i_{current}}$: Graphe hétérogène encodant le planning actuel

timesteps ← Identification des insertions possibles dans la fenêtre du patient c
pred ← liste vide des prédictions
for chaque insertion `t` dans *timesteps* **do**
    $G_{i_{current},\mathtt{t}}$ ← Création du graphe d'insertion associé à `t` à partir de $G_{i_{current}}$
    out ← Prédiction inférée par M à partir $G_{i_{current},\mathtt{t}}$
    pred ← Ajout de *out* à la liste des prédictions
end for
choice ← Sélection de la meilleure via ML_choice appliquée sur *pred*
return Décision d'insertion du patient ou impossibilité de service

CHAPITRE 5 RÉSULTATS ET DISCUSSION

Dans ce chapitre, nous présentons les résultats obtenus par la méthode proposée dans différents contextes d'évaluation. La section 5.1 définit les métriques utilisées pour sélectionner le modèle déployé. La section 5.2 décrit les méthodes de comparaison ainsi que les métriques d'évaluation retenues. La section 5.3 présente les performances obtenues sur les données historiques de Fastercom. Enfin, la section 5.4 étudie une variante relaxée du problème.

5.1 Sélection du modèle

Cette section définit le modèle retenu parmi l'ensemble des modèles entraînés, en vue d'une évaluation dans un contexte opérationnel. Le choix repose sur les métriques classiques d'évaluation associées à une tâche de classification en apprentissage automatique. Nous notons TP les vrais positifs (cas positifs correctement prédits), TN les vrais négatifs, FP les faux positifs (négatifs prédits comme positifs) et FN les faux négatifs. Les métriques considérées sont la précision, le rappel, l'*accuracy*, le score F1 et l'aire sous la courbe précision-rappel. Elles sont brièvement décrites ci-dessous :

— **Précision** : proportion d'instances prédites positives qui sont effectivement positives :

$$\frac{TP}{TP + FP}$$

Elle mesure la fiabilité des prédictions positives.

— **Rappel** : proportion d'instances réellement positives qui sont correctement identifiées :

$$\frac{TP}{TP + FN}$$

Il évalue la capacité du modèle à détecter les cas positifs.

— **Accuracy** : proportion d'instances correctement classées parmi l'ensemble des prédictions :

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Elle fournit une mesure globale de la performance du modèle.

— **Score F1** : moyenne harmonique entre la précision et le rappel :

$$2 \cdot \frac{\text{Précision} \cdot \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

Cette métrique équilibre les deux précédentes et est particulièrement pertinente en cas de déséquilibre entre les classes.

— **Aire sous la courbe précision-rappel** : cette mesure évalue l'aire sous la courbe précision-rappel. Elle est particulièrement adaptée aux problèmes de classification déséquilibrée, car elle accorde davantage de poids à la performance sur la classe minoritaire.

En s'appuyant sur ces critères, les modèles ayant obtenu les meilleures performances globales ont été sélectionnés, en mettant l'accent sur la précision et le rappel. L'objectif principal est de minimiser les erreurs de type faux positifs, c'est-à-dire les cas où le modèle prédit à tort qu'une insertion est possible.

Cette priorité repose sur l'idée qu'une prédiction négative erronée (prédire qu'une insertion est impossible alors qu'elle l'est) est moins critique qu'une prédiction positive erronée, laquelle pourrait aboutir à une promesse de service non tenue. Pour pallier les promesses non tenues, l'algorithme utilisant le modèle doit résoudre l'instance associée à l'insertion sélectionnée en utilisant une requête vers un solveur. Un des objectifs de la recherche étant de minimiser le nombre de requêtes, il est crucial de réduire les requêtes inutiles.

Le modèle choisi repose sur une architecture "grande", une initialisation de Xavier, et un dropout de 0.3. Il a été entraîné avec un taux d'apprentissage de $8e^{-4}$, une décroissance des poids de $1e^{-5}$ et une taille des lots de 64. Ce modèle est utilisé pour tous les résultats sur les données historiques de Fastercom de la section 5.3.

5.2 Évaluation de la qualité de l'approche

Pour déterminer si l'approche utilisée dans ce travail est prometteuse, il est nécessaire d'en évaluer la qualité et de la comparer à d'autres méthodes de référence. Dans cette section, nous détaillons les différentes méthodes testées ainsi que les métriques utilisées en vue d'une comparaison globale.

5.2.1 Méthodes considérées

Sept méthodes différentes sont comparées, en s'appuyant sur les mêmes instances pour chacune. Nous les classons selon différents types : trois méthodes sont basées sur l'apprentissage, deux méthodes sont gloutonnes, une repose sur l'aléatoire, et enfin la méthode de ***Fastercom***, décrite dans le chapitre précédent, est également incluse. Pour chaque méthode, nous précisons le critère déterminant le choix de l'insertion. Certaines méthodes s'arrêtent à la première insertion faisable, tandis que d'autres sélectionnent la meilleure parmi celles faisables.

Nous listons ci-dessous les différentes méthodes pour plus de clarté :

— Méthode classique :

1. ***Fastercom*** : Cette méthode teste toutes les insertions possibles via des requêtes vers un solveur. Pour chacune d'elles, elle récupère deux informations distinctes : sa faisabilité, déterminée via l'appel au solveur, et sa « qualité », définie comme la valeur attribuée par la fonction objectif (3.14). Si au moins une insertion est jugée faisable, la méthode sélectionne celle maximisant cette valeur. En l'absence d'insertion faisable, elle signale l'infaisabilité et n'insère pas le patient.

— Méthodes gloutonnes :

1. ***G_first*** : Cette méthode repose sur la logique de ***Fastercom***, mais limite le nombre d'insertions testées. Elle sélectionne le patient déjà servi le plus proche du nouveau patient à insérer, et teste les trois insertions centrées sur l'horaire de ce patient. Par exemple, si le patient le plus proche est servi entre 10h et 10h29, les insertions testées pour le nouveau patient sont [9h30, 9h59], [10h, 10h29], et [10h30, 10h59], si elles sont compatibles avec la fenêtre temporelle de ce dernier. La meilleure insertion faisable est choisie, sinon la méthode élargit les tests à l'ensemble des insertions possibles comme ***Fastercom***. En l'absence d'insertion faisable, le patient n'est pas inséré.
2. ***G_3*** : Cette méthode fonctionne comme ***G_first*** mais teste un plus grand ensemble d'insertions, correspondant à l'union des insertions associées aux trois patients les plus proches déjà insérés dans le planning.

— Méthode aléatoire :

1. ***Random*** : Cette méthode ne calcule pas la qualité des insertions. Elle teste aléatoirement les insertions possibles et sélectionne la première insertion faisable identifiée via un solveur. Dans ce travail, la méthode est exécutée dix fois pour améliorer la fiabilité des résultats. La moyenne et l'écart-type sont ensuite calculés, et les résultats sont agrégés pour former une unique méthode de référence.

— Méthodes basées sur l'apprentissage :

1. ***ML_top1*** : Cette méthode sélectionne l'insertion ayant la plus haute valeur de prédiction. Si celle-ci est supérieure à 0.5, elle est prédite comme faisable, et une requête au solveur est lancée pour vérification afin d'éviter une promesse non tenue. Si la faisabilité de l'insertion est confirmée, le patient est inséré. En cas d'échec ou dans le cas où aucune prédiction n'est supérieure à 0.5, la stratégie adoptée reprend celle de ***Fastercom*** en évaluant successivement toutes les insertions.

2. ***ML_top1_rd*** : Variante de ***ML_top1***. Elle sélectionne la prédiction la plus haute et vérifie sa faisabilité. Cependant, lorsque l'insertion n'est pas faisable ou que la prédiction est inférieure à 0.5, la stratégie adoptée reprend celle de ***Random*** en testant aléatoirement les insertions jusqu'à en trouver une faisable.
3. ***ML_sorted*** : Cette méthode utilise ses prédictions pour trier les insertions à tester. Les insertions sont testées dans l'ordre décroissant des prédictions, et la première insertion faisable est retenue.

5.2.2 Métriques de comparaison

L'évaluation des résultats de chaque méthode repose sur deux métriques : le nombre de patients servis et le nombre de requêtes effectuées, qui permettent d'estimer conjointement la performance d'une méthode.

Le nombre de patients servis doit être maximisé. Chaque patient non servi représente une perte pour l'entreprise, ce qui rend cette métrique essentielle. Pour faciliter les comparaisons, nous utilisons un pourcentage de perte ou de gain par rapport à la méthode ***Fastercom***. Une perte de 50 % rendrait une méthode inutilisable en pratique, tandis qu'une légère perte peut être tolérée si elle s'accompagne d'une réduction des requêtes vers un solveur externe. Le nombre de requêtes vers un solveur externe mesure la capacité du modèle à répondre rapidement au patient et à limiter les résolutions effectuées. Toutefois, une réduction excessive du nombre de requêtes ne doit pas compromettre le nombre de patients servis.

Dans la section précédente, nous précisons l'utilisation de requêtes vers un solveur pour valider les insertions choisies. Cette approche présente un avantage fondamental : elle garantit que le patient ajouté, ainsi que tous ceux initialement présents dans le planning, sont effectivement servis. Ce choix élimine le risque de promesse non tenue, améliorant la robustesse des différentes méthodes. Un compromis important entre la robustesse de la méthode et son gain de temps est à prendre en compte.

5.3 Résultats sur les données historiques de Fastercom

Dans cette section, nous décrivons les instances utilisées et testons notre méthode sur les données historiques de l'entreprise. Deux environnements ont été utilisés dans ce travail. L'entraînement des modèles a été réalisé en partie grâce au soutien fourni par le Cirrelt [96] sur un serveur distant équipé d'un processeur Intel Xeon Silver 4116 fonctionnant à 2,10 GHz avec 16 Go de mémoire vive, et d'une carte graphique NVIDIA Tesla V100-PCIE-32GB. Le système utilise la version 5.14.0 de Linux et le code est implémenté en Python 3.10.12, avec

la version 2.0.0+cu121 de la bibliothèque DGL [84]. La génération des données et l'exécution des expériences ont été effectuées en partie grâce au soutien fourni par l'Alliance [97] sur un second serveur équipé d'un processeur Intel Xeon E5-2683 v4 fonctionnant à 2,10 GHz avec 16 Go de mémoire vive. Ce serveur fonctionne avec la version 5.14.0 de Linux et le code est également implémenté avec les mêmes versions de Python et DGL.

5.3.1 Description des instances

Pour produire des résultats représentatifs, il est nécessaire de placer le modèle dans des situations proches des conditions d'exploitation réelles. L'algorithme 1 servant à la création des données est légèrement modifié afin d'obtenir un cadre similaire au fonctionnement opérationnel des méthodes. Les étapes de génération des couples graphe/booléen et leur stockage sont remplacées par une sélection d'insertion (dépendante de la méthode utilisée) communiquée au patient, comme dans l'algorithme 3. Ce cadre d'évaluation est formalisé dans l'algorithme 4 pour une méthode arbitraire $\boldsymbol{X}$. Pour représenter au mieux les conditions réelles, cette

Algorithm 4 Génération des plannings pour la méthode $\boldsymbol{X}$

Require: $I = \{i_1, i_2, \ldots, i_{172}\}$: Ensemble de test des instances historiques
1: **for** chaque instance i appartenant à I **do**
2: $C_1 \leftarrow$ Sélection aléatoire de 50% des patients de l'instance i
3: $C_2 \leftarrow$ Les autres 50 pour cent des patients
4: $i_{current} \leftarrow$ Génération du planning des patients dans C_1 avec la méthode ***Fastercom***
5: **for** chaque patient c appartenant à C_2 **do**
6: `ins` $\leftarrow$ Choix de l'insertion via $\boldsymbol{X}$
7: $i_{current} \leftarrow$ Mise à jour du planning en intégrant le patient c à l'insertion `ins`
8: **end for**
9: **end for**
10: **return** Ensemble des plannings construits

méthodologie est appliquée uniquement sur les données qui n'ont jamais été vues ou utilisées pour l'entraînement du modèle. Cela constitue notre ensemble de test, composé de 172 instances indépendantes. Pour chaque journée, un cadre typique d'exécution est appliqué, où différents patients sont ajoutés un à un après avoir préalablement rempli la journée avec 50 % des patients en utilisant la méthode ***Fastercom***. Les plannings de chaque méthode sont générés à partir des mêmes instances, permettant une comparaison cohérente.

5.3.2 Résultats des méthodes

Le tableau 5.1 présente le nombre de patients servis ainsi que le nombre de requêtes nécessaires pour les 172 plannings construits par chaque méthode. Les résultats sont présentés en

comparaison de ceux obtenus avec la méthode de référence ***Fastercom***.

L'avantage des méthodes basées sur l'apprentissage réside dans le fait qu'elles ne testent qu'un sous-ensemble d'insertions, en utilisant les prédictions comme critère de tri préalable. On observe que la méthode ***ML_top1_rd*** surpasse l'ensemble des autres approches en réduisant de plus de 55 % le nombre de requêtes vers un solveur externe effectuées, tout en servant 1.1 % de patients en plus que la méthode ***Fastercom***. De manière générale, nos méthodes performent mieux que la méthode classique de l'entreprise. Elles permettent de servir un plus grand nombre de patients tout en réduisant le temps d'exécution associé.

Cependant, les résultats sont mitigés : bien que supérieurs à ceux de la méthode ***Fastercom***, les résultats des méthodes d'apprentissage automatique ne sont que légèrement supérieurs à ceux obtenus avec la méthode ***Random***. Les bonnes performances de la méthode aléatoire peuvent être attribuées à deux facteurs principaux : la simplicité des instances considérées et le caractère glouton des autres approches. Les méthodes ***Fastercom***, **G_best** et ***G_3*** utilisent la fonction objectif (3.14) visant à minimiser le temps de travail des soignants, ce qui conduit à favoriser systématiquement les insertions en début de journée lorsque cela est possible. Appliquée sans connaissance des patients futurs, cette stratégie engendre un effet de saturation des premiers créneaux et réduit les opportunités d'insertion ultérieures.

En revanche, la méthode ***Random***, en explorant l'espace des créneaux disponibles de manière non biaisée, évite cet effet de congestion et parvient à insérer un plus grand nombre de patients. Par ailleurs, la méthode ***ML_top1***, bien qu'inspirée de ***Fastercom***, repose sur une approche de classification moins directement dépendante de la fonction objectif, ce qui lui permet d'obtenir des performances comparables. Ces résultats, ainsi que les perspectives d'amélioration de l'approche proposée, sont discutés plus en détail dans le chapitre final du document.

Au-delà du nombre de patients servis, une métrique essentielle pour comparer les méthodes est

Tableau 5.1 Résultats des méthodes ML sur les données historiques

	Nombre de patients servis	**Gain de patients (%)**	**Nombre de requêtes**	**Gain de requêtes (%)**
Fastercom	1730	0	7089	0
ML_top1_rd	**1749**	**1.1**	**3136**	**55.76**
ML_sorted	1747	0.98	3263	53.97
Random	1745.4 ± 6.71	0.89 ± 0.39	3141.5 ± 27.63	55.69 ± 0.39
ML_top1	1743	0.75	5219	26.38
G_best	1739	0.52	4604	35.05
G_3	1735	0.29	6092	14.06

la qualité des plannings générés. Un planning est considéré comme de bonne qualité lorsque le temps de travail des soignants est à la fois réduit et équitablement réparti entre eux. Pour évaluer cette qualité, nous utilisons la fonction objectif (3.14). Pour chaque instance, lorsque les plannings générés par les méthodes ***ML_top1*** et ***Random*** permettent de servir un nombre équivalent de patients, nous comparons les méthodes via la fonction (3.14).

L'apport de ***ML_top1***, méthode fondée sur l'apprentissage, réside dans sa capacité à minimiser le temps de travail tout en favorisant les insertions dont la faisabilité est la plus probable. Parmi les 105 instances pour lesquelles ***ML_top1*** et ***Random*** servent le même nombre de patients, la première obtient un meilleur score objectif dans 87 cas. Ainsi, dans 83 % des cas, la méthode fondée sur le ML génère un planning de coût plus faible, réduisant davantage le temps de travail total des soignants.

Les méthodes ***ML_top1_rd*** et ***ML_sorted*** n'utilisent pas la fonction objectif dans le processus de sélection de l'insertion ; le résultat discuté ci-dessus pour ***ML_top1*** ne leur est donc pas applicable. Ces deux méthodes ne prennent pas en compte le temps de travail pour effectuer leur choix, et produisent des plannings dont la qualité est équivalente à celle obtenue avec la méthode ***Random***.

Notons que toutes les méthodes servent les patients auxquels elles ont promis un service. Pour obtenir ce résultat, les vérifications par requêtes vers un solveur externe sont nécessaires. Elles augmentent considérablement la robustesse des méthodes, au prix d'une réduction du gain de temps.

5.3.3 Limitation du temps utilisable

La section précédente analysait les gains de temps et de patients des méthodes proposées dans un contexte sans contrainte temporelle. Rappelons qu'un patient disposant de 10 possibilités d'insertion nécessite, avec la méthode ***Fastercom***, 10 appels à un solveur, ce qui conduit à un temps de réponse proche d'une minute.

Nous allons plus loin en appliquant un cadre réaliste et en temps réel, exploitable en pratique. Nous étendons les méthodes en leur imposant un seuil temporel afin qu'elles respectent la contrainte de temps réel, en ne disposant que de 20 secondes pour répondre au patient. Dans ce laps de temps, chaque méthode peut effectuer au maximum trois requêtes vers un solveur externe, ainsi que les calculs supplémentaires nécessaires. Nous listons ci-dessous les adaptations spécifiques pour chaque méthode.

— Méthode classique :

1. ***Fastercom_seuil*** : La méthode sélectionne aléatoirement trois insertions parmi

celles possibles, puis les teste afin de déterminer la meilleure.

— Méthodes gloutonnes :

1. ***G_first_seuil*** : Pas de changement. La méthode applique sa logique initiale, limitée à trois insertions.
2. ***G_3_seuil*** : La méthode sélectionne aléatoirement trois insertions parmi l'union des insertions à tester pour les trois patients déjà servis les plus proches du nouveau patient à insérer.

— Méthode aléatoire :

1. ***Random_seuil*** : La méthode s'arrête après trois insertions testées, même si aucune n'est faisable.

— Méthodes basées sur l'apprentissage :

1. ***ML_top1_seuil*** : Aucun changement, hormis une limitation à trois requêtes lorsque l'insertion prédite s'avère infaisable.
2. ***ML_top1_rd_seuil*** : Aucun changement, hormis une limitation à trois requêtes lorsque l'insertion prédite s'avère infaisable.
3. ***ML_sorted_seuil*** : Comme pour ***Random_seuil***, l'algorithme s'arrête après trois insertions testées.

Le tableau 5.2 présente les résultats de toutes les méthodes respectant la contrainte de temps réel, comparées à la méthode de référence ***Fastercom*** ayant un temps illimité pour effectuer sa réponse. La méthode ***Fastercom_seuil*** présentée dans le tableau permet une comparaison à ce que ferait potentiellement une entreprise pour respecter la contrainte de temps réel. Le seuil temporel est fixé à 20 secondes, afin de garantir un temps d'attente acceptable pour un patient au téléphone.

Tableau 5.2 Résultats des méthodes avec seuil de temps

	Nombre de patients servis	**Gain de patients (%)**	**Nombre de requêtes**	**Gain de requêtes (%)**
Fastercom	1730	0	7089	0
ML_top1_seuil	**1699**	**-1.79**	3478	50.94
Random_seuil	1698.90 ± 7.48	-1.80 ± 0.43	2860.50 ± 19.36	59.65 ± 0.27
ML_top1_rd_seuil	1691	-2.25	**2836**	**59.99**
Fastercom_seuil	1688	-2.43	4430	37.51
ML_sorted_seuil	1668	-3.58	2963	58.2
G_3_seuil	1609	-6.99	3852	45.66
G_best_seuil	1520	-12.14	3175	55.21

Comme précédemment, les méthodes basées sur l'apprentissage et l'aléatoire surpassent les autres méthodes pour les deux métriques considérées. Par exemple ***ML_top1_seuil*** permet

de répondre à la contrainte de temps réel tout en réduisant le nombre de requêtes effectuées de plus de 50 % en ne perdant que 1.79 % de patients servis. Les méthodes ML performent mieux que la méthode ***Fastercom_seuil*** permettant de servir plus de patients tout en réduisant le nombre de requêtes nécessaires. Seule ***ML_sorted_seuil*** performe moins bien que ***Fastercom_seuil*** en termes de patients servis, mais permet une plus grande réduction du nombre de requêtes. Les méthodes gloutonnes sont de loin moins bonnes que les autres. Cette sous-performance est liée au phénomène de congestion discuté dans la section précédente qui s'accentue quand le nombre d'insertions à tester se réduit. L'apport de ***ML_top1_seuil*** réside à nouveau dans la qualité des plannings créés.Parmi les 95 instances où **ML_top1_seuil** et ***Random_seuil*** servent le même nombre de patients, dans 66 d'entre elles, c'est **ML_top1_seuil** dont le planning trouvé a la meilleure valeur objectif. La méthode basée sur le ML crée donc, dans 69% des cas, un planning de coût plus faible qui minimise le travail des soignants.

5.4 Variante relaxée du problème

En observant les données, nous remarquons que les contraintes de capacité sont peu serrées et ne contraignent pas fortement notre problème. Ces contraintes peuvent freiner l'apprentissage du modèle, sans pour autant représenter des éléments centraux dans la façon dont les instances sont résolues. L'apprentissage de nos modèles étant basé sur ces données, il est intéressant d'étudier les performances des méthodes lorsque ces contraintes sont retirées. Le problème à résoudre devient une instance du MDVRPTW avec une contrainte de double compatibilité. Suite à ce retrait, nous ajustons les différentes caractéristiques d'entrée du modèle en ne gardant que celles jugées essentielles.

Nous décrivons comment le retrait des contraintes de capacité a été effectué dans le but de respecter la cohérence des données et du cadre de travail associé. En retirant ces contraintes, les instances du problème deviennent potentiellement un peu plus simples à résoudre. Cependant, les informations transmises lors de l'entraînement du modèle sont moins redondantes. Nous détaillons l'architecture du nouveau modèle, sa sélection, et finalement ses résultats.

5.4.1 Architecture du nouveau modèle

Nous simplifions le modèle en retirant les informations redondantes et non essentielles. L'architecture est modifiée sur deux aspects : le retrait des contraintes de capacité et la suppression de certaines caractéristiques jugées superflues.

Premièrement, les caractéristiques associées aux contraintes de capacité h_{v_a-1}, h_{v_a-2}, h_{v_r-1}

et h_{v_r-2} avec $v_a \in A$ et $v_r \in R$ sont retirées. Deuxièmement, les caractéristiques associées aux nombres de compétences et d'affinités de chaque patient et de chaque membre du personnel soignant h_{v_a-3}, h_{v_a-4}, h_{v_r-3} et h_{v_r-4} avec $v_a \in A$ et $v_r \in R$ sont également retirées. Comme expliqué à la section 4.1.2, la contrainte de compatibilité est encodée à la fois de manière explicite et implicite dans la représentation graphique du problème. Ici, nous retirons l'encodage explicite et ne conservons que l'encodage implicite. Troisièmement, les caractéristiques h_{v_t-1} et h_{v_t-2}, avec $v_t \in T$, sont également retirées car elles sont déjà représentées implicitement par les arêtes de types E_{T-A} et E_{T-R} sortantes du nœud v_t. Pour le reste, la nouvelle architecture est similaire à l'ancienne.

5.4.2 Sélection du modèle et description des instances

Dans cette section, nous appliquons le même principe que celui décrit dans la section 5.1 pour choisir le modèle à utiliser. Les instances utilisées sont légèrement modifiées par rapport à la description faite à la section 5.3.1. Pour chaque patient et chaque membre du personnel soignant, les demandes et capacités associées sont retirées. Les instances sont donc identiques à celles utilisées précédemment, à l'exception du fait que les deux contraintes de capacité sont désormais relaxées.

5.4.3 Résultats des méthodes

Dans cette section, nous déterminons si les performances de notre modèle sont différentes sur la version relaxée du problème. La méthode utilisée pour obtenir les résultats est identique à celle expliquée en début de section 5.3 et présentée dans l'algorithme 4. Les résultats obtenus, après la création des 172 plannings pour chaque méthode, sont présentés dans le tableau 5.3. Nous observons que les résultats restent similaires à la version précédente du

Tableau 5.3 Résultats sans limites de temps du nouveau modèle

	Nombre de patients servis	**Gain de patients (%)**	**Nombre de requêtes**	**Gain de requêtes (%)**
Fastercom	1740	0	7131	0
ML_top1_rd	**1757**	**0.98**	3293	53.82
ML_sorted	1756	0.92	3325	53.37
Random	1752 ± 7.18	0.69 ± 0.41	**3158.8 ± 26.17**	**55.70 ± 0.37**
ML_top1	1745	0.29	4827	32.31
G_best	1743	0.17	3175	34.13
G_3	1736	-0.23	3898	12.52

modèle. Le retrait des contraintes et des caractéristiques ne permet pas d'accroître fortement les résultats des méthodes ML. Dans cette version relaxée, toutes les méthodes sont capables de servir plus de patients, mais les méthodes basées sur l'apprentissage ne se démarquent pas plus qu'avant. Au contraire, parmi les 115 instances où **ML_top1** et ***Random*** servent le même nombre de patients, **ML_top1** n'est meilleur que dans 75 d'entre elles représentant plus de 65% des cas contre plus de 80% auparavant.

5.4.4 Limitation du temps utilisable

Nous appliquons la contrainte de temps réel au nouveau problème afin d'analyser les performances des différentes méthodes. Les résultats obtenus pour les méthodes limitées par un seuil temporel pour la variante relaxée du problème sont présentés dans le tableau 5.4.

Nous obtenons des résultats légèrement moins bons qu'avant et la méthode ***Random*** performe mieux que les autres méthodes. De plus, la méthode ***ML_top1_seuil*** performe moins bien que ***Fastercom_seuil*** sur ces instances du problème et elle ne génère des plannings de meilleure qualité que 60 % du temps. Nous discutons de ces observations plus en profondeur dans la section 6.1.

Tableau 5.4 Résultats avec seuil de temps pour le problème relaxé

	Nombre de patients servis	**Gain de patients (%)**	**Nombre de requêtes**	**Gain de requêtes (%)**
Fastercom	1740	0	7131	0
Random_seuil	**1704.80 ± 5.11**	**-2.02 ± 0.29**	**2857.10 ± 23.96**	**59.93 ± 0.34**
ML_top1_rd_seuil	1695	-2.59	2940	58.77
ML_sorted_seuil	1690	-2.87	2980	58.21
Fastercom_seuil	1687	-3.05	4421	38
ML_top1_seuil	1684	-3.22	3209	55
G_3_seuil	1616	-7.13	3898	45.34
G_best_seuil	1507	-13.39	3175	55.11

CHAPITRE 6 CONCLUSION

Ce mémoire propose une méthode de résolution basée sur l'apprentissage automatique pour l'ajout dynamique de patients dans des plannings déjà optimisés de tournées de véhicules. La méthode utilise les GNN pour éviter les multiples résolutions successives d'instances du HHCRSP.

Premièrement, nous formulons le problème comme un problème d'optimisation linéaire, représentant les instances de notre problème. Le problème que nous considérons est une variante d'un MDCVRPTW avec une double contrainte de compatibilité. La formulation exprime la présence des contraintes de fenêtres de temps, des contraintes de capacités en pansements et en seringues, des points de départ distincts pour chaque membre du personnel soignant, ainsi que toutes les autres contraintes liées à la cohérence des routes et aux compatibilités entre patients et soignants.

Ensuite, nous proposons une modélisation des solutions (plannings) du problème sous forme de graphes hétérogènes. Cette modélisation est suffisamment expressive pour contenir, de façon implicite et explicite, toutes les informations nécessaires à la reconstruction de la solution associée. Nous précisons les caractéristiques contenues dans le graphe et leur normalisation pour une création cohérente des données. La création des données est détaillée par l'introduction de la notion de graphe d'insertion, modélisant l'insertion d'un nouveau patient dans un planning déjà optimisé à un intervalle de temps précis. Nous décrivons une construction intelligente du graphe d'insertion à partir du graphe hétérogène associé au planning avant l'insertion et détaillons l'algorithme de création des données et leur répartition.

Ensuite, nous détaillons la formulation du modèle d'apprentissage automatique, composé d'un GNN et d'un MLP en cascade, permettant l'encodage et le décodage nécessaires pour réaliser la tâche de classification demandée. Pour chaque graphe d'insertion, le modèle prédit si l'insertion associée est faisable ou non. En parallèle, nous décrivons l'algorithme d'apprentissage, le calibrage des hyperparamètres ainsi que son utilisation en pratique.

Les expériences sont effectués sur les 172 instances de test issues des données historiques de Fastercom. Pour chaque instance, nous créons un chemin d'exécution similaire à un cadre pratique et insérons un à un les patients dans le planning. Les résultats agrégés démontrent que les méthodes ML et aléatoires performent mieux que les méthodes gloutonnes et celle de l'entreprise. La méthode ***ML_top1_seuil*** réduit de moitié le nombre de requêtes utilisées et répond au cadre de réponse en temps réel recherché, tout en ne concédant qu'une perte de 1.79 % de patients servis en moins. La méthode proposée permet donc d'accélérer le temps de

résolution et de répondre à la contrainte de temps réel en délivrant une réponse en moins de 20 secondes, sans que le nombre de patients servis en moins ne soit trop important. Les tests ont montré que, bien que les performances relatives des algorithmes ML soient satisfaisantes, leurs performances globales restent limitées, n'étant que légèrement supérieures à celles d'un algorithme choisissant aléatoirement les insertions à effectuer.

Nous proposons également une variante relaxée du problème visant à affiner les caractéristiques utilisées pour l'apprentissage et à retirer les contraintes non nécessaires du modèle. Nous obtenons cependant des résultats légèrement moins bons que pour la variante classique du problème considéré.

6.1 Limitations et améliorations futures

Dans cette section, nous discutons des résultats obtenus en vue de proposer des pistes d'amélioration et de justifier l'apport de l'apprentissage automatique dans le contexte de cette recherche. Nous abordons également l'intérêt de l'utilisation de l'apprentissage supervisé et des GNN dans la résolution successive d'instances du HHCRSP.

Les résultats obtenus sont mitigés. Bien que les méthodes ML soient capables de servir plus de patients que la méthode ***Fastercom***, tout en réduisant drastiquement le nombre de requêtes, cela ne suffit pas, dans la mesure où la méthode ***Random*** obtient des résultats similaires. Pour comprendre ces performances, il est essentiel de considérer les instances sur lesquelles les méthodes sont appliquées. Celles-ci sont, pour la plupart, simples à résoudre. Dans ce contexte, une instance simple correspond à une instance peu contrainte, pour laquelle il n'est pas difficile de servir la grande majorité, voire la totalité, des patients. La faible complexité de ces instances permet à la plupart des méthodes de servir un grand nombre de patients. Pour expliquer les performances des méthodes ML et de la méthode ***Random***, il convient d'examiner plus en profondeur la manière dont les patients sont insérés.

D'un côté, les insertions effectuées par les méthodes ***Fastercom***, ***G__best*** et ***G__3*** sont déterminées à partir de la valeur objectif donnée par la fonction de l'équation (3.14). Cette fonction, cherchant à minimiser le temps de travail des soignants, attribue souvent la valeur maximale aux insertions effectuées en début de planning. Ce placement systématique des rendez-vous en début de journée crée une congestion qui empêche l'insertion de patients futurs disposant de fenêtres temporelles restreintes à des périodes déjà saturées.

De l'autre, l'impact de la fonction objectif sur les méthodes ML et sur la méthode ***Random*** est plus faible. Les insertions sont soit aléatoires, soit effectuées à l'intervalle temporel jugé le plus probable pour que l'insertion soit faisable. En plaçant les patients plus librement

dans la journée, et compte tenu de la simplicité des instances considérées, la congestion est moins prononcée pour ces méthodes, leur permettant ainsi de servir un plus grand nombre de patients. Pour obtenir des résultats plus significatifs, l'obtention ou la création de données plus complexes constitue une piste pour les travaux futurs.

Pour améliorer les performances de nos méthodes, rappelons que celles-ci reposent sur le paradigme de l'apprentissage supervisé, où les données d'entraînement déterminent la qualité du modèle final. Si les données utilisées ne représentent pas fidèlement celles rencontrées en pratique, les performances du modèle seront faibles en raison d'une généralisation limitée. En plus du type de données, le nombre de données utilisées pour l'entraînement est un facteur clé. Un modèle d'apprentissage supervisé généralise mieux une tâche lorsqu'il est entraîné sur un ensemble plus large. Nous identifions une source d'inefficacité à la ligne 15 de l'algorithme 1, qui détermine la manière dont sont construits les plannings d'entraînement. En insérant systématiquement les patients à la « meilleure » insertion, le modèle est rarement confronté à des configurations alternatives. Pourtant, lors de la création des plannings, il est peu probable que le modèle insère systématiquement le patient à l'insertion qui minimise la fonction objectif (3.14), cette information lui étant inconnue. Pour pallier ce problème, diversifier et élargir l'ensemble d'entraînement est nécessaire. Nous proposons, pour de futures recherches, une modification de l'algorithme 1, en particulier de sa ligne 15. En complément de l'insertion optimale, nous suggérons de générer des plannings où les insertions se font de manière aléatoire. Cette approche permet de générer un ensemble d'entraînement plus large et plus diversifié, rendant le modèle capable de répondre à un éventail plus large de situations.

En complément, il est important de rappeler que les données utilisées simulent un cadre hebdomadaire à l'intérieur d'une seule journée d'exécution. Cette simplification modifie l'ordre de grandeur du nombre d'agents dans les plannings. Il serait pertinent de collecter ou de construire des données hebdomadaires pour évaluer l'outil dans son contexte d'exécution réel. Cela permettrait une mise à l'échelle du nombre d'agents et accroîtrait également la complexité du problème. Il conviendrait également de modifier la fonction objectif utilisée afin de limiter la congestion. Une fonction plus expressive pourrait évaluer les plannings de manière plus fine, en intégrant, par exemple, le coût horaire d'un véhicule, le coût d'un trajet ou le coût d'utilisation d'un véhicule.

Pour finir, nous discutons de l'intérêt des GNN face à une méthode aléatoire pour la résolution successive d'instances du HHCRSP. Indépendamment des résultats obtenus jusqu'ici, l'utilisation de l'apprentissage automatique est justifiée par sa capacité à sélectionner directement des insertions optimisées, maximisant ainsi l'espace disponible pour les patients futurs, même dans des contextes complexes où une stratégie aléatoire causerait des pénalités

importantes et des solutions de moins bonne qualité. Pour cela, il conviendrait d'adapter légèrement l'approche d'entraînement afin d'améliorer la prise de décision, comme évoqué précédemment. Les méthodes basées sur l'apprentissage apparaissent comme des outils plus robustes pour la gestion de scénarios complexes, tout en restant suffisamment flexibles pour s'adapter aux besoins de l'outil. Pour des travaux futurs, une architecture à deux têtes pourrait être envisagée, permettant de réaliser simultanément une tâche de classification et une tâche de régression. La régression sur une nouvelle fonction objectif permettrait d'évaluer la qualité des insertions faisables et ainsi de mieux prendre en compte l'incertitude liée aux demandes dynamiques, ce qu'un algorithme aléatoire ne peut faire.

RÉFÉRENCES

[1] G. B. Dantzig et J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, n°. 1, p. 80–91, Oct. 1959.

[2] R. Bellman, "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, n°. 1, p. 87–90, Apr. 1958.

[3] B. Eksioglu, A. V. Vural et A. Reisman, "The vehicle routing problem : A taxonomic review," *Computers & Industrial Engineering*, vol. 57, n°. 4, p. 1472–1483, May 2009.

[4] M. Desrochers, J. Desrosiers et M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows," *Operations Research*, vol. 40, n°. 2, p. 342–354, Apr. 1992.

[5] G. D. Konstantakopoulos, S. P. Gayialis et E. P. Kechagias, "Vehicle routing problem and related algorithms for logistics distribution : A literature review and classification," *Operational research*, vol. 22, n°. 3, p. 2033–2062, 2022.

[6] H. Han et E. Ponce Cueto, "Waste collection vehicle routing problem : literature review," *PROMET-Traffic&Transportation*, vol. 27, n°. 4, p. 345–358, 2015.

[7] E. Cheng et J. Rich, "A home health care routing and scheduling problem," /, 10 1998.

[8] H. Bae et I. Moon, "Multi-depot vehicle routing problem with time windows considering delivery and installation vehicles," *Applied Mathematical Modelling*, vol. 40, n°. 13–14, p. 6536–6549, Feb. 2016.

[9] H. Shankar, G. Mani et K. Pandey, "Gis based solution of multi-depot capacitated vehicle routing problem with time window using tabu search algorithm," *International Journal of Traffic and Transportation Engineering*, vol. 3, n°. 2, p. 83–100, 2025.

[10] S. Kadambi et P. Rao, "Constrained quadratic model formulations for mdcvrptw : Quantum vs classical," dans *Proceedings of COMSNETS 2024*, Jan. 2024, p. 258–263.

[11] J. K. Lenstra et A. R. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, n°. 2, p. 221–227, 1981.

[12] P. Toth et D. Vigo, *The Vehicle Routing Problem*. Philadelphia : SIAM, 2002.

[13] G. Laporte et Y. Nobert, "A branch and bound algorithm for the capacitated vehicle routing problem," *Operations-Research-Spektrum*, vol. 5, n°. 2, p. 77–85, Jun. 1983.

[14] C. Barnhart *et al.*, "Branch-and-price : Column generation for solving huge integer programs," *Operations research*, vol. 46, n°. 3, p. 316–329, 1998.

[15] M. W. P. Savelsbergh, "Local search in routing problems with time windows," *Annals of Operations Research*, vol. 4, n°. 1, p. 285–305, Dec. 1985.

[16] E. D. Taillard, “Local search,” dans *Graduate Texts in Operations Research*. Springer, Aug. 2022, p. 103–129.

[17] G. Clarke et J. W. Wright, “Scheduling of vehicles from a central depot to a number of delivery points,” *Operations Research*, vol. 12, p. 568–581, 1964.

[18] P. Toth et D. Vigo, *Vehicle routing : problems, methods, and applications*. SIAM, 2014.

[19] S. Lin, “Computer solutions of the traveling salesman problem,” *Bell System Technical Journal*, vol. 44, n°. 10, p. 2245–2269, Dec. 1965. [En ligne]. Disponible : https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1965.tb04146.x

[20] F. Glover et M. Laguna, “Tabu search,” dans *Handbook of Combinatorial Optimization*, 1998, p. 2093–2229. [En ligne]. Disponible : https://link.springer.com/chapter/10.1007/978-1-4613-0303-9_33

[21] S. Kirkpatrick, C. D. Gelatt Jr et M. P. Vecchi, “Optimization by simulated annealing,” *science*, vol. 220, n°. 4598, p. 671–680, 1983.

[22] R. P. Jefferis et K. A. Fegley, “Application of dynamic programming to routing problems,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 1, n°. 1, p. 21–26, 1965.

[23] A. M. Geoffrion, “Lagrangean relaxation for integer programming,” dans *Approaches to integer programming*. Springer, 2009, p. 82–114.

[24] A. H. Land et A. G. Doig, *An automatic method for solving discrete programming problems*. Springer, 2010.

[25] M. Padberg et G. Rinaldi, “A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems,” *SIAM review*, vol. 33, n°. 1, p. 60–100, 1991.

[26] M. Grötschel et O. Holland, “Solution of large-scale symmetric travelling salesman problems,” *Mathematical Programming*, vol. 51, n°. 1, p. 141–202, 1991.

[27] D. Naddef et G. Rinaldi, “Branch-and-cut algorithms for the capacitated vrp,” dans *The vehicle routing problem*. SIAM, 2002, p. 53–84.

[28] R. Baldacci, E. Hadjiconstantinou et A. Mingozzi, “An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation,” *Operations research*, vol. 52, n°. 5, p. 723–738, 2004.

[29] T. Vidal *et al.*, “A hybrid genetic algorithm for multidepot and periodic vehicle routing problems,” *Operations Research*, vol. 60, n°. 3, p. 611–624, Jun. 2012.

[30] T. Vidal, “Hybrid genetic search for the cvrp : Open-source implementation and swap* neighborhood,” *Computers & Operations Research*, vol. 140, p. 105 643–105 643, Nov. 2021.

[31] F. Rosenblatt, “The perceptron : a probabilistic model for information storage and organization in the brain.” *Psychological review*, vol. 65, n°. 6, p. 386, 1958.

[32] Y. Bengio, A. Lodi et A. Prouvost, “Machine learning for combinatorial optimization : a methodological tour d’horizon,” *European Journal of Operational Research*, vol. 290, n°. 2, p. 405–421, 2021.

[33] M. Kruber, M. E. Lübbecke et A. Parmentier, “Learning when to use a decomposition,” dans *Lecture Notes in Computer Science*. Springer, Jan. 2017, p. 202–210.

[34] P. Bonami, A. Lodi et G. Zarpellon, “Learning a classification of mixed-integer quadratic programming problems,” dans *Lecture Notes in Computer Science*. Springer, 2018, p. 595–604.

[35] A. Lodi et G. Zarpellon, “On learning and branching : a survey,” *TOP*, vol. 25, n°. 2, p. 207–236, Jun. 2017.

[36] O. Vinyals, M. Fortunato et N. Jaitly, “Pointer networks,” dans *Advances in Neural Information Processing Systems 28*, C. Cortes *et al.*, édit. Curran Associates, Inc., 2015, p. 2692–2700.

[37] D. Bahdanau, K. Cho et Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv :1409.0473*, 2014.

[38] Q. Cappart *et al.*, “Combinatorial optimization and reasoning with graph neural networks,” *Journal of Machine Learning Research*, vol. 24, n°. 130, p. 1–61, 2023.

[39] F. Scarselli *et al.*, “The graph neural network model,” *IEEE transactions on neural networks*, vol. 20, n°. 1, p. 61–80, 2008.

[40] I. Bello *et al.*, “Neural combinatorial optimization with reinforcement learning,” *arXiv preprint arXiv :1611.09940*, 2016.

[41] M. Nazari *et al.*, “Reinforcement learning for solving the vehicle routing problem,” *Advances in neural information processing systems*, vol. 31, 2018.

[42] L. Perron et V. Furnon, “Or-tools,” Google, [En ligne]. Disponible : https://developers.google.com/optimization/.

[43] W. Kool, H. Van Hoof et M. Welling, “Attention, learn to solve routing problems !” *arXiv preprint arXiv :1803.08475*, 2018.

[44] K. Helsgaun, “An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems,” *Roskilde : Roskilde University*, vol. 12, p. 966–980, 2017.

[45] M. Morabit *et al.*, “Learning to repeatedly solve routing problems,” *Networks*, vol. 83, n°. 3, p. 503–526, Dec. 2023. [En ligne]. Disponible : https://onlinelibrary.wiley.com/doi/epdf/10.1002/net.22200

[46] 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.

[47] O. Bräysy et M. Gendreau, “Vehicle routing problem with time windows, part i : Route construction and local search algorithms,” *Transportation science*, vol. 39, n°. 1, p. 104–118, 2005.

[48] ——, “Vehicle routing problem with time windows, part ii : Metaheuristics,” *Transportation science*, vol. 39, n°. 1, p. 119–139, 2005.

[49] R. Baldacci, A. Mingozzi et R. Roberti, “Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints,” *European Journal of Operational Research*, vol. 218, n°. 1, p. 1–6, 2012.

[50] E. Cheng et J. L. Rich, “A home health care routing and scheduling problem,” Technical Report. Oakland University (Michigan) and Rice University (Texas), USA, Rapport technique, 1998.

[51] IBM, *IBM ILOG CPLEX 12.7 User’s Manual*, IBM ILOG CPLEX Division, Incline Village, NV, 2017.

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

[53] M. Cissé *et al.*, “Or problems related to home health care : A review of relevant routing and scheduling problems,” *Operations Research for Health Care*, vol. 13–14, p. 1–22, Jun. 2017.

[54] M. Di Mascolo, C. Martinez et M.-L. Espinouse, “Routing and scheduling in home health care : A literature survey and bibliometric analysis,” *Computers & Industrial Engineering*, vol. 158, p. 107255, 2021.

[55] S. Bertels et T. Fahle, “A hybrid setup for a hybrid scenario : combining heuristics for the home health care problem,” *Computers & Operations Research*, vol. 33, n°. 10, p. 2866–2890, Oct. 2006.

[56] C. Akjiratikarl, P. Yenradee et P. R. Drake, “Pso-based algorithm for home care worker scheduling in the uk,” *Computers & Industrial Engineering*, vol. 53, n°. 4, p. 559–583, Jun. 2007.

[57] G. Hiermann *et al.*, “Metaheuristics for solving a multimodal home-healthcare scheduling problem,” *Central European Journal of Operations Research*, vol. 23, p. 89–113, 2015.

[58] Q. Hou *et al.*, “Generalize learned heuristics to solve large-scale vehicle routing problems in real-time,” dans *The Eleventh International Conference on Learning Representations*, 2023.

[59] M. Kim, J. Park et J. Kim, “Learning collaborative policies to solve np-hard routing problems,” dans *Advances in Neural Information Processing Systems*, vol. 34, Dec. 2021, p. 10 418–10 430. [En ligne]. Disponible : https://proceedings.neurips.cc/paper/2021/hash/564127c03caab942e503ee6f810f54fd-Abstract.html

[60] M. Bazirha, R. Benmansour et A. Kadrani, “An efficient two-phase heuristic for the home care routing and scheduling problem,” *Computers & Industrial Engineering*, vol. 181, p. 109329, Jul. 2023.

[61] Z. Zong *et al.*, “Reinforcement learning for solving multiple vehicle routing problem with time window,” *ACM Transactions on Intelligent Systems and Technology*, vol. 15, n°. 2, p. 1–19, 2024.

[62] F. Yu *et al.*, “Logistics distribution route optimization with time windows based on multi-agent deep reinforcement learning,” *International Journal of Information Technologies and Systems Approach*, vol. 17, n°. 1, p. 1–23, Apr. 2024.

[63] A. R. Bennett et A. L. Erera, “Dynamic periodic fixed appointment scheduling for home health,” *IIE Transactions on Healthcare Systems Engineering*, vol. 1, n°. 1, p. 6–19, Mar. 2011.

[64] M. Demirbilek, J. Branke et A. Strauss, “Dynamically accepting and scheduling patients for home healthcare,” *Health Care Management Science*, vol. 22, n°. 1, p. 140–155, Jan. 2018.

[65] M. Demirbilek, J. Branke et A. K. Strauss, “Home healthcare routing and scheduling of multiple nurses in a dynamic environment,” *Flexible Services and Manufacturing Journal*, vol. 33, n°. 1, p. 253–280, Apr. 2019.

[66] Q. Ta-Dinh *et al.*, “A reinforcement learning approach for the online dynamic home health care scheduling problem,” *Health Care Management Science*, vol. 27, n°. 4, p. 650–664, Nov. 2024.

[67] A. Bogyrbayeva *et al.*, “Learning to solve vehicle routing problems : A survey,” *arXiv preprint arXiv :2205.02453*, 2022.

[68] R. Shahbazian *et al.*, “Integrating machine learning into vehicle routing problem : Methods and applications,” *IEEE Access*, 2024.

[69] L. Duan *et al.*, “Efficiently solving the practical vehicle routing problem,” dans *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Aug. 2020.

[70] T. N. Kipf et M. Welling, “Semi-supervised classification with graph convolutional networks,” *CoRR*, vol. abs/1609.02907, 2016. [En ligne]. Disponible : http://arxiv.org/abs/1609.02907

[71] H. Cai *et al.*, “Solving the vehicle routing problem with stochastic travel cost using deep reinforcement learning,” *Electronics*, vol. 13, n°. 16, p. 3242, Aug. 2024.

[72] S. Ammon, F. Phillipson et R. Almeida, “A supervised machine learning approach for the vehicle routing problem,” dans *Proceedings of the 13th International Conference on Operations Research and Enterprise Systems*, 2024, p. 364–371.

[73] K. Zhang, X. Lin et M. Li, “Graph attention reinforcement learning with flexible matching policies for multi-depot vehicle routing problems,” *Physica A : Statistical Mechanics and its Applications*, vol. 611, p. 128451, Feb. 2023.

[74] P. Mukherjee et S. Dey, “Efficient vehicle routing problem : A machine learning and evolutionary computation approach,” dans *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023, p. 3–4.

[75] A. P. de Larivière, “Apprentissage profond en optimisation combinatoire : Apprentissage de bornes et résolution de problèmes de transport,” Thèse de doctorat, Polytechnique Montréal, affiliée à l’Université de Montréal, Jul. 2023, voir chapitre 6, pp. 40–48 : Projet 3 – Apprentissage Automatique pour l’Approximation et la Résolution de Problèmes de Planification de Rendez-vous dans des Tournées de Véhicules en Temps Réel.

[76] M. Cococcioni et L. Fiaschi, “The big-m method with the numerical infinite m,” *Optimization Letters*, vol. 15, n°. 7, p. 2455–2468, Sep. 2020.

[77] J. Schmidhuber, “Deep learning in neural networks : An overview,” *Neural Networks*, vol. 61, p. 85–117, Oct. 2014.

[78] L. Alzubaidi *et al.*, “Review of deep learning : concepts, cnn architectures, challenges, applications, future directions,” *Journal of Big Data*, vol. 8, n°. 1, Mar. 2021.

[79] D. E. Rumelhart, G. E. Hinton et R. J. Williams, “Learning internal representations by error propagation,” dans *Parallel Distributed Processing : Explorations in the Microstructure of Cognition, Volume 1 : Foundations*, D. E. Rumelhart, J. L. McClelland et the PDP Research Group, édit. Cambridge, MA : MIT Press, 1986, p. 318–362.

[80] F. Scarselli *et al.*, “The graph neural network model,” *IEEE Transactions on Neural Networks*, vol. 20, n°. 1, p. 61–80, Dec. 2008.

[81] Z. Wu *et al.*, “A comprehensive survey on graph neural networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, n°. 1, p. 4–24, Mar. 2020.

[82] T. Monninger *et al.*, “Scene : Reasoning about traffic scenes using heterogeneous graph neural networks,” *IEEE Robotics and Automation Letters*, vol. 8, n°. 3, p. 1531–1538, Jan. 2023.

[83] K. Cabello-Solorzano *et al.*, “The impact of data normalization on the accuracy of machine learning algorithms : A comparative analysis,” dans *Lecture Notes in Networks*

and Systems. Springer, 2023, p. 344–353.

[84] M. Wang *et al.*, "Deep graph library : A graph-centric, highly-performant package for graph neural networks," *arXiv*, 2019. [En ligne]. Disponible : https://arxiv.org/abs/1909.01315

[85] P. Veličković *et al.*, "Graph attention networks," *arXiv preprint arXiv :1710.10903*, 2017.

[86] X. Wang *et al.*, "Heterogeneous graph attention network," dans *The world wide web conference*, 2019, p. 2022–2032.

[87] X. Glorot, A. Bordes et Y. Bengio, "Deep sparse rectifier neural networks," dans *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, p. 315–323.

[88] K. Kamiński *et al.*, "Rossmann-toolbox : a deep learning-based protocol for the prediction and design of cofactor specificity in rossmann fold proteins," *Briefings in Bioinformatics*, vol. 23, n°. 1, Sep. 2021.

[89] L. Huang *et al.*, "Normalization techniques in training dnns : Methodology, analysis and application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, n°. 8, p. 10 173–10 196, Feb. 2023.

[90] A. L. Maas *et al.*, "Rectifier nonlinearities improve neural network acoustic models," dans *Proc. icml*, vol. 30, n°. 1. Atlanta, GA, 2013, p. 3.

[91] X. Glorot et Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," dans *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, p. 249–256.

[92] K. He *et al.*, "Delving deep into rectifiers : Surpassing human-level performance on imagenet classification," dans *Proceedings of the IEEE international conference on computer vision*, 2015, p. 1026–1034.

[93] P. Blanchard, D. J. Higham et N. J. Higham, "Accurate computation of the log-sum-exp and softmax functions," *arXiv*, 2019. [En ligne]. Disponible : https://arxiv.org/abs/1909.03469

[94] A. Krogh et J. Hertz, "A simple weight decay can improve generalization," *Advances in neural information processing systems*, vol. 4, 1991.

[95] N. Srivastava *et al.*, "Dropout : A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, n°. 56, p. 1929–1958, 2014. [En ligne]. Disponible : https://jmlr.org/papers/v15/srivastava14a.html

[96] Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, "Site officiel du cirrelt," consulté le 5 avril 2025. [En ligne]. Disponible : https://www.cirrelt.ca/

[97] Alliance de recherche numérique du Canada, "Site officiel de l'alliance de recherche numérique du canada," consulté le 5 avril 2025. [En ligne]. Disponible : https://alliancecan.ca/fr

ANNEXE A LECTURE COMPLÉMENTAIRE - RÉSULTATS OBTENUS À PARTIR DE LA GÉNÉRATION ALÉATOIRE

Nous proposons en complément des résultats obtenus sur les données historiques de Fastercom, une analyse des forces et faiblesses de nos méthodes d'apprentissage automatique dans différents cas de figure. Cette analyse repose sur une catégorisation des situations, obtenue par la génération aléatoire de neuf catégories d'instances distinctes sur lesquelles les méthodes sont testées. Nous commençons par expliquer comment la génération de données est effectuée. Ensuite, nous présentons les résultats obtenus pour chaque type d'instance par les méthodes basées sur l'apprentissage automatique.

Génération aléatoire :

Nous calculons les différentes métriques qui régissent la distribution des données historiques à notre disposition. En prenant en compte cette distribution lors de la création des données aléatoires, nous nous assurons que la distribution de nos données générées se rapproche de celle des données historiques, améliorant également la cohérence de la structure des données. Nous listons ci-dessous la méthodologie utilisée pour la génération des différents éléments du problème :

- **Nombre de patients :** Valeur aléatoire (uniforme discrète) entre un minimum et un maximum défini.
- **Nombre de soignants :** Valeur aléatoire (uniforme discrète) entre un minimum et un maximum défini.
- **Temps de service :** Multiple de 5 compris entre 0 et 30, tiré aléatoirement (uniforme discrète).
- **Demandes :** Valeur aléatoire tirée suivant une distribution pondérée d'après la fréquence des valeurs dans les données historiques.
- **Capacités :** La capacité en seringues est fixée à 20 pour toutes les infirmières. La capacité en pansements est tirée selon une loi uniforme discrète entre 150 et 250.
- **Compatibilité :** Les compétences et affinités sont générées à partir d'une distribution pondérée d'après la fréquence des valeurs dans les données historiques.
- **Fenêtre temporelle — patient :** Déterminée à partir de la génération de deux éléments : la taille et l'heure de début.

  1) La taille des fenêtres est générée à partir d'une distribution gaussienne similaire à celle observée dans les données historiques.

2) Le placement temporel des fenêtres est généré à partir d'une distribution pondérée, favorisant les trois premiers quarts de la journée afin d'éviter les dépassements hors journée de travail. Cette approche permet un meilleur remplissage des journées et un positionnement plus réaliste des fenêtres.

— **Fenêtre temporelle — infirmière :** Définie à partir de la durée de la journée, elle-même tirée selon une loi normale. Le début et la fin de la fenêtre temporelle sont ajustés en ajoutant ou en retirant une valeur tirée d'une loi uniforme discrète entre 0 et 90 minutes avec une probabilité de 25 % pour chaque extrémité de la journée.

Les données sont générées aléatoirement suivant deux axes, nous permettant d'étudier le comportement des algorithmes face à des types de données spécifiques. Ces deux axes sont le nombre de patients à servir et la taille des fenêtres temporelles de ces patients. Nous utilisons trois catégories différentes pour chacun des axes, créant ainsi neuf classes distinctes de données. Premièrement, les trois catégories de l'axe « nombre de patients » correspondent à celles des données historiques : 10 à 19 patients, 20 à 29 patients, et 30 à 40 patients. Deuxièmement, les trois catégories de l'axe « taille des fenêtres temporelles » sont : « Petit », « Moyen », et « Grand ». Les neuf ensembles de test correspondant aux différentes combinaisons sont présentés dans le tableau A.1.

Tableau A.1 Catégorisation des ensembles d'instances aléatoire

	Petit	**Moyen**	**Grand**
10 à 19	s_1	s_2	s_3
20 à 29	s_4	s_5	s_6
30 to 40	s_7	s_8	s_9

La taille des fenêtres temporelles des patients est ajustée en décalant la médiane et en divisant l'écart-type utilisés dans la loi normale. Pour les petites fenêtres, l'écart-type est divisé par 2 et la moyenne est décalée à gauche de ((médiane − min)/2). Pour les grandes fenêtres, l'écart-type est divisé par 2 et la moyenne est décalée à droite de ((max − médiane)/2). Pour la taille moyenne, aucun changement n'est fait.

Résultats des méthodes :

Nous évaluons les performances du modèle d'apprentissage sur les neuf ensembles d'entraînement distincts. Le modèle utilisé a été entraîné sur des données aléatoires de chaque catégorie, en suivant la même méthodologie de génération que pour les instances historiques de Fastercom. Seules les performances de la meilleure méthode ML avec le seuil temporel pour chaque catégorie sont présentées. A part pour la catégorie « 10 à 19 / Petit », la meilleure méthode est ***ML_top1_seuil***. Les valeurs présentées correspondent au pourcentage de patients sup-

plémentaires servis et au pourcentage de requêtes économisées par rapport à la méthode de référence ***Fastercom***. Chaque case du tableau A.2 correspond aux résultats obtenus pour une catégorie d'instances, ordonnée comme dans le tableau A.1.

Tableau A.2 Résultats des méthodes ML sur la génération aléatoire

	Petit	**Moyen**	**Grand**
10 à 19	0.69 / 56.15	-1.32 / 43.2	-2.47 / 59.89
20 à 29	0.79 / 36.81	-0.21 / 36.45	-3.55 / 63.15
30 à 40	-0.88 / 32.55	-0.97 / 39.02	-3.09 / 71.30

Nous observons que nos méthodes performent correctement pour toutes les catégories, à l'exception de celles présentant des fenêtres temporelles de grande taille. Lorsque la majorité des fenêtres temporelles sont larges, les hallucinations sont fréquentes. Nous observons également que les résultats des méthodes sont plus faibles pour les trois catégories devant servir de 30 à 40 patients. Cette réduction est possiblement causée par un appauvrissement des données d'entraînement pour ce type de cas. Lors de la génération, nous vérifions qu'il est possible de créer un planning servant au moins 30 patients à partir des patients de base. Cependant, la création d'un planning n'étant pas déterministe, il arrive fréquemment que le planning construit ne contienne pas plus de 30 patients. Ce phénomène réduit le nombre de couples graphe/booléen incluant plus de 30 patients. Le modèle, disposant de moins de données pour s'entraîner sur ce type de configuration, obtient des résultats moins bons sur les catégories associées. Ce phénomène illustre une difficulté bien connue en apprentissage supervisé, dont nous discutons dans la section 6.1 consacrée aux limites du modèle.