Titre: Title:	Résolution heuristique par génération de colonnes et apprentissage automatique du problème d'horaires d'autobus électriques
Auteur: Author:	Juliette Gerbaux
Date:	2023
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Gerbaux, J. (2023). Résolution heuristique par génération de colonnes et apprentissage automatique du problème d'horaires d'autobus électriques [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/53409/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/53409/
Directeurs de recherche: Advisors:	Guy Desaulniers, & Quentin Cappart
Programme: Program:	Maitrise recherche en mathématiques appliquées

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Résolution heuristique par génération de colonnes et apprentissage automatique du problème d'horaires d'autobus électriques

JULIETTE GERBAUX
Département de mathématiques et de génie industriel

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

Mai 2023

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Résolution heuristique par génération de colonnes et apprentissage automatique du problème d'horaires d'autobus électriques

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

Nadia LAHRICHI, présidente
Guy DESAULNIERS, membre et directeur de recherche
Quentin CAPPART, membre et codirecteur de recherche
Daniel ALOISE, membre

REMERCIEMENTS

Ma maîtrise recherche à Polytechnique Montréal m'a apporté de nombreuses connaissances et a renforcé mon intérêt pour la recherche opérationnelle. En particulier, ce projet de recherche m'a permis de développer mes compétences et mes méthodes en recherche et en mathématiques appliquées. Je tiens donc à remercier les différentes personnes qui ont contribué, de près ou de loin, au bon déroulement de cette formation.

Mes remerciements s'adressent tout particulièrement à mes directeurs de recherche, Guy Desaulniers et Quentin Cappart, pour m'avoir proposé ce projet de recherche et pour m'avoir accompagnée tout au long de la réalisation de ma maîtrise par leur encadrement, leurs conseils, leurs encouragements, leur temps, leur flexibilité et leur réactivité.

Je remercie aussi Nadia Lahrichi et Daniel Aloise pour avoir accepté de faire partie de mon jury de maîtrise. Je remercie également l'équipe du Gerad, en particulier François Lessard, Benoit Rochefort et Pierre Girard, qui m'ont permis d'avoir les outils adaptés pour mener à bien ce travail de recherche. Je remercie en outre Jonathan Brasseur qui m'a partagé son travail et m'a permis d'avoir des données pour mes expériences. Je remercie aussi l'équipe Giro de Véronique Bouffard pour son financement et leurs précieux conseils sur ce projet.

Je remercie ma famille et mes amis pour leur soutien et leur présence ainsi que toutes les personnes que j'ai rencontrées à Montréal, en particulier Pauline et Zoé, qui ont permis que cette maîtrise à Polytechnique Montréal soit une si belle expérience.

Enfin, je souhaite remercier les différents professeurs que j'ai eus au cours de ma scolarité qui m'ont fait découvrir et aimer les mathématiques appliquées.

RÉSUMÉ

De plus en plus d'autobus électriques sont utilisés dans les transports en commun pour les bénéfices environnementaux qu'ils apportent par rapport à un autobus classique. Cependant, utiliser des autobus électriques demande de repenser le processus de planification et d'opération des autobus. En particulier le problème d'horaires de véhicules (*Vehicle Scheduling Problem*) (VSP) et sa résolution doivent être revus. En effet, ce problème qui vise à affecter les autobus aux différents trajets de bus est complexe et crucial pour minimiser les coûts d'opération liés au nombre d'autobus utilisés et à la distance qu'ils parcourent. Dans le cas de véhicules électriques, la minimisation de la distance est essentielle et de nouvelles contraintes liées à l'autonomie et à la recharge des véhicules doivent être considérées.

De nombreux travaux ont été menés sur le problème d'horaires de véhicules électriques (*Electric Vehicle Scheduling Problem*) (EVSP). Différentes formulations et méthodes de résolution ont été proposées. Le problème résolu dans ce travail est le problème d'horaires de véhicules électriques multi-dépôt (*Multi Depot Electric Vehicle Scheduling Problem*) (MDEVSP) avec recharge partielle et linéaire par morceaux, un seul type de véhicules électriques et un seul type de chargeur. La modélisation choisie discrétise les temps de recharge et permet de tenir compte du nombre de chargeurs disponibles aux différentes stations de recharge. Ce modèle se veut proche de la réalité et permet ainsi d'être utilisée en pratique. À notre connaissance, peu de méthodes de résolution ont été proposées pour résoudre cette variante du MDEVSP.

Une des méthodes habituellement utilisée et très performante pour résoudre le problème d'horaires de véhicules multi-dépôt (*Multi Depot Vehicle Scheduling Problem*) (MDVSP) (qui est un problème NP-difficile) est la génération de colonnes (GC) qui se base à la fois sur un modèle de partitionnement d'ensemble et sur un modèle de réseau dans lequel on cherche des plus courts chemins. Cette méthode, même en utilisant des accélérations heuristiques, ne permet pas de résoudre assez rapidement le EVSP. On propose donc une nouvelle méthode pour accélérer la GC qui s'appuie sur une réduction heuristique du nombre de variables dans le problème. La réduction est faite par élimination d'arcs dans le graphe qui modélise le problème. Pour cela, on combine deux méthodes : un algorithme glouton et de l'apprentissage supervisé basé sur un réseau de neurones pour graphe (*Graph Neural Network*) (GNN). Le GNN est une architecture neuronale qui est entrainée sur des instances résolues par GC heuristique.

La méthode proposée est validée sur des instances comprenant environ 700 trajets et 2500 trajets générées aléatoirement à partir de données de la ville de Montréal. Les tests montrent

que la combinaison des deux méthodes apporte un gain significatif par rapport à l'utilisation séparée des méthodes. En divisant le nombre d'arcs dans la modélisation par 5, elle permet de réduire, pour les plus petites instances, entre 65% et 87% le temps de résolution avec un écart du coût de la solution à la valeur optimale d'au plus 3,7%. Avec une division par 10 du nombre d'arcs, la méthode permet aussi un gain de temps de 49% avec un écart de 4,2% sur les grandes instances grâce à un apprentissage par transfert du GNN. Malheureusement, les tests font apparaitre que les modèles appris généralisent mal à des instances ayant une distribution différente de celles utilisées lors de l'entrainement.

ABSTRACT

More and more electric buses are used in public transport, thanks to the environmental benefits they bring compared to traditional buses. However, using electric buses requires rethinking the bus operational planning process. In particular, the Vehicle Scheduling Problem (VSP) and its resolution must be adapted. Indeed, the resolution of this problem, which consists in assigning vehicles to different bus trips, is major to minimize the operating costs related to the fleet size and the travelling distance. In the case of electric vehicles, distance minimization is critical and new constraints related to vehicle autonomy and charging process must be considered.

Many works have studied the Electric Vehicle Scheduling Problem (EVSP). Different formulations and approaches have been proposed. The problem solved in this work is the Multi Depot Electric Vehicle Scheduling Problem (MDEVSP) with partial and piecewise linear charging, only one type of electric vehicles and one type of chargers. The model discretizes the charging periods and takes into account the number of chargers available at the different charging stations. It is designed to be realistic and, therefore, to be used in practice. To the best of our knowledge, only few solution methods have been proposed to solve this variant of the MDEVSP.

One of the approaches usually very efficient to solve the Multi Depot Vehicle Scheduling Problem (MDVSP) is column generation (GC) which is based on a set partitioning model and a network model to generate resource-constraint shortest paths. This approach does not solve the EVSP fast enough, even with heuristic speed-up strategies. Consequently, we propose a new strategy to accelerate the GC based on a heuristic reduction of the number of variables in the problem. The reduction is done by eliminating arcs in the graph that models the problem. For this, we combine two techniques: a greedy algorithm and supervised learning based on a graph neural network (GNN). The GNN is a neural architecture that is trained on instances solved by GC.

The proposed approach is validated on instances involving around 700 and 2500 randomly generated trips from Montreal real-life data. The experiments show that the combination of the two techniques provides a significant gain compared to the separate use of the techniques. For the smallest instances, it allows reducing between 65% and 87% of the computational time, with a gap to the optimal value less than 3.7%. For the large instances, the solution method also provides a time saving of 49% with a gap of 4,2% thanks to a transfer learning approach. The number of arcs is divided by 5 and 10 for the small and large instances

respectively. Unfortunately, the learned GNN generalizes poorly to instances with a different distribution than the one used during training.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

VSP	Problème d'horaires de véhicules (*Vehicle Scheduling Problem*)
SDVSP	Problème d'horaires de véhicules à un seul dépôt (*Single Depot Vehicle Scheduling Problem*)
MDVSP	Problème d'horaires de véhicules multi-dépôt (*Multi Depot Vehicle Scheduling Problem*)
EVSP	Problème d'horaires de véhicules électriques (*Electric Vehicle Scheduling Problem*)
SDEVSP	Problème d'horaires de véhicules électriques à un seul dépôt (*Single Depot Electric Vehicle Scheduling Problem*)
MDEVSP	Problème d'horaires de véhicules électriques multi-dépôt (*Multi Depot Electric Vehicle Scheduling Problem*)
GC	Génération de colonnes
BP	Branch & Price
BC	Branch & Cut
BB	Branch & Bound
GNN	Réseau de neurones pour graphe (*Graph Neural Network*)
LNS	Recherche locale à grand voisinage (*Large Neighbourhood Search*)
ALNS	Recherche locale à grand voisinage adaptatif (*Adaptive Large Neighbourhood Search*)
PLNE	Problème linéaire en nombres entiers
MP	Problème maitre
SP	Sous-problème

LISTE DES NOTATIONS

Modélisation

n	Nombre de trajets à desservir
$T = \{t_1, t_2, ..., t_n\}$	Ensemble des trajets
q_t^s	Temps de départ du trajet t
q_t^e	Temps d'arrivée du trajet t
ρ_t	Temps du trajet à vide sur le parcours du trajet t
α	Vitesse des véhicules
σ_t	Consommation d'énergie sur le trajet t
e_{mouv}	Consommation d'énergie par unité de distance parcourue
e_{arr}	Consommation d'énergie à l'arrêt par unité de temps
m	Nombre de dépôts
$D = \{d_1, d_2, ..., d_m\}$	Ensemble des dépôts
v^d	Capacité du dépôt d
r	Nombre de stations de recharge
$H = \{h_1, h_2, ..., h_r\}$	Ensemble des stations de recharge
u^h	Nombre de bornes de recharge à la station de recharge h
f	Fonction de recharge
δ	Période de la discrétisation pour les stations de recharge
N	Nombre de périodes de la discrétisation
$P = \{p_1, p_2, ..., p_N\}$	Périodes de temps de la discrétisation
τ_p^s, τ_p^e	Début et fin de la période p, respectivement
σ_{max}	Capacité des batteries
σ_{min}	État de charge minimal à maintenir
β	Temps maximal entre la fin d'un trajet et le début d'un autre trajet desservis consécutivement
γ	Temps minimum d'attente au dépôt
c_f	Coût fixe d'utilisation d'un véhicule
c_a	Coût d'attente par unité de temps à un arrêt
c_v	Coût de voyage à vide par unité de distance
c_d^f	Coût fixe de retourner au dépôt pendant la journée
c_r^f	Coût fixe d'aller à la recharge

c_r^v	Coût d'attente à la recharge par unité de temps passée à la recharge

Graphe

$\mathcal{G} = (\mathcal{N}, \mathcal{A})$	Graphe complet regroupant tous les dépôts
$\mathcal{G}^d$	Graphe modélisant les routes de dépôt d
o_d	nœud origine
k_d	nœud destination
$s_p^{d'}$	nœud de retour au dépôt d' à la période de temps p
t_i	nœud du trajet t_i
w_p^h	nœud d'attente à la station de recharge h à la période p
r_p^h	nœud de recharge à la station de recharge h à la période p
c_a	Coût de l'arc a
σ_a	Consommation d'énergie sur l'arc a
$O = \bigcup_{d \in D} o^d$	Ensemble des nœuds origine
$T = \bigcup_{i=1}^{n} t_i$	Ensemble des nœuds trajet que l'on confond avec l'ensemble des trajets
$S = \bigcup_{d \in D} \bigcup_{p \in P} s_p^d$	Ensemble des nœuds d'attente aux dépôts
$R = \left(\bigcup_{h \in H} \bigcup_{p \in P} w_p^h\right) \cup \left(\bigcup_{h \in H} \bigcup_{p \in P} r_p^h\right)$	Ensemble des nœuds de recharge
$K = \bigcup_{d \in D} k^d$	Ensemble des nœuds destination
IT	Ensemble des arcs qui relient deux nœuds de T (les arcs rouges)
TS	Ensemble des arcs qui relient un nœud de T et un nœud de S (les arcs verts)
TR	Ensemble des arcs qui relient un nœud de T et un nœud de R (les arcs bleus)
B	Ensemble des arcs qui regroupent les arcs dépôt-trajet, les arcs trajet-dépôt, les arcs de la recharge et les arcs de l'attente (les arcs noirs)

Algorithme glouton

$\tilde{T} = (\tilde{t_i})_{i \in \{1,..,n\}}$	Liste des trajets ordonnés par heure de départ, du plus tôt au plus tard
L^d	Liste des arcs composant les routes en construction du dépôt d
Σ^d	Liste des états de charge des routes du dépôt d
y^d	Nombre de véhicules utilisés au dépôt d
τ	Ensemble des derniers trajets des routes en construction
c_{tot}	Coût total de la solution en construction
$\tilde{A}$	Ensemble des arcs valides dans la solution en construction

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

De plus en plus de villes souhaitent utiliser des autobus électriques pour le transport en commun. Par exemple, le regroupement C40 de grandes villes à travers le monde s'est engagé à proposer une offre de transport en commun avec zéro-émissions d'ici à 2025 [1]. En effet, l'utilisation de véhicules électriques dans le transport public permet de réduire les émissions de gaz à effet de serre ainsi que la pollution et le bruit.

Cependant, le déploiement d'autobus électriques demande de repenser le processus de planification et d'opération des autobus et par conséquent les méthodes de résolution des problèmes mathématiques associés à cause de nouvelles contraintes imposées par les batteries des véhicules. Les changements nécessaires pour prendre en compte les véhicules électriques dans ce processus suivant les technologies de recharge choisies sont décrits dans [2].

Le processus de planification et d'opération est très important pour les entreprises de service de transport public, car il détermine leur offre et leur rentabilité. Des entreprises comme Giro proposent alors des logiciels d'optimisation pour trouver des solutions aux différents problèmes de ce processus. Il est important que les méthodes proposées donnent des solutions de grande qualité en des temps courts pour qu'elles soient utilisables en pratique. Le travail présenté est financé par cette entreprise et s'intéresse au problème d'horaires de véhicules électriques (*Electric Vehicle Scheduling Problem*) (EVSP), qui est une des étapes importantes de ce processus, et à sa résolution que l'on souhaite rapide et efficace.

1.1 Définition et contexte du problème d'horaires de véhicules électriques

Le processus de planification et d'opération des autobus a été largement étudié dans la littérature [3] et comporte plusieurs étapes importantes : la planification stratégique qui consiste à créer le réseau de transport et les différentes lignes, la planification tactique qui consiste à définir les fréquences et les horaires des lignes de transport et la planification opérationnelle qui consiste à définir les horaires des bus, les horaires des chauffeurs de bus, les plannings de maintenance et à assigner les autobus aux parkings. Chacune de ces étapes peut être formulée comme un problème d'optimisation combinatoire, de minimisation des coûts et/ou de maximisation de la satisfaction des passagers. Les contraintes peuvent être des contraintes d'infrastructures (par exemple l'emplacement des dépôts) ou des contraintes budgétaires (par exemple le nombre maximum d'autobus) ou des contraintes opérationnelles (par exemple le fait qu'un autobus doive retourner au même dépôt le soir). Les différents

problèmes sont généralement résolus de manière séquentielle, mais il est possible de combiner certains problèmes entre eux. Une des étapes essentielles de ce processus est le problème d'horaires de véhicules (*Vehicle Scheduling Problem*) (VSP). Connaissant la position et la capacité des dépôts, les horaires des différentes lignes, le VSP consiste à trouver pour chaque autobus de la flotte une suite de trajets à desservir formant un horaire réalisable de sorte que les coûts d'opération soient minimisés et que tous les trajets planifiés soient desservis. Les coûts d'opération comprennent par exemple le nombre d'autobus utilisés, les temps d'attente et les temps de trajet à vide. Dans le cas d'un dépôt simple, ce problème est nommé problème d'horaires de véhicules à un seul dépôt (*Single Depot Vehicle Scheduling Problem*) (SDVSP) et dans le cas de plusieurs dépôts, il est appelé problème d'horaires de véhicules multi-dépôt (*Multi Depot Vehicle Scheduling Problem*) (MDVSP). Dans le cas de plusieurs dépôts, c'est un problème NP-difficile [4] qui peut être formulé comme un problème linéaire en nombres entiers (PLNE) et qui est habituellement résolu à l'aide des différents outils de la recherche opérationnelle, notamment par génération de colonnes.

Les méthodes développées pour résoudre le VSP ne permettent pas de résoudre directement sans adaptations le EVSP, qui est aussi un problème NP-difficile [5], même pour le problème d'horaires de véhicules électriques à un seul dépôt (*Single Depot Electric Vehicle Scheduling Problem*) (SDEVSP). En effet, l'utilisation de véhicules électriques pose certaines contraintes supplémentaires comme l'autonomie des véhicules du fait de la capacité limitée des batteries qui ne permet de parcourir qu'une centaine de kilomètres sans être rechargées. Il faut donc prendre en compte la capacité limitée des batteries dans la formulation mathématique ainsi que le processus de recharge des batteries. La recharge des batteries est un processus non linéaire qu'il faut modéliser [6]. De plus, il faut tenir compte et/ou décider de l'emplacement des bornes de recharge et de leur nombre dans la résolution du problème. Ces contraintes qui complexifient le problème demandent de repenser la formulation mathématique et la résolution du VSP.

1.2 Éléments à prendre en compte pour la résolution du EVSP

La modélisation choisie pour le EVSP joue un rôle important dans sa résolution. Habituellement, il est formulé comme un PLNE. Différentes modélisations sont possibles, qui influencent la complexité du problème mathématique. Les éléments cités plus hauts, par exemple le temps de recharge non linéaire par rapport à l'état initial de charge, peuvent être pris en considération ou non. En particulier, il existe de nombreuses manières de modéliser le processus de recharge : états de charge et périodes de recharge discrétisées ou continues, temps de recharge linéaire, linéaire par morceaux ou non linéaire, possibilité de recharge partielle, etc. D'autres

éléments peuvent être modélisés comme différents types de véhicules électriques avec différentes autonomies ou différents types de chargeur (lent ou rapide). Les choix de modélisation reflètent plus ou moins la réalité des problèmes rencontrés en pratique.

Un autre élément important à prendre en compte dans la résolution du EVSP est la taille des instances que l'on souhaite résoudre. La taille d'une instance est définie par le nombre de trajets que l'on veut pouvoir desservir, le nombre de dépôts considérés et le nombre de stations de recharge. La taille des instances que l'on peut résoudre dépend de la modélisation retenue, de la méthode de résolution choisie, de la qualité de la solution souhaitée, c'est-à-dire l'écart au coût optimal si l'on utilise une heuristique et du temps imparti pour aboutir à une solution. Dans la littérature, les plus grosses instances du problème d'horaires de véhicules électriques multi-dépôt (*Multi Depot Electric Vehicle Scheduling Problem*) (MDEVSP) résolues de façon quasi optimale, c'est-à-dire en contrôlant l'écart du coût de la solution trouvée à la valeur optimale, comptent un millier de trajets. Des instances de taille plus grande sont résolues par des heuristiques, mais sans garantie sur la qualité de la solution. Or dans une situation réaliste, à Montréal par exemple, il peut y avoir jusqu'à environ 20000 trajets par jour dans le problème d'horaires. En pratique, ces instances sont découpées en sous-problèmes, suivant la position géographique des trajets, de 2000 à 3000 trajets par sous-problème.

Une des méthodes classiques pour résoudre le MDVSP [7] est l'algorithme de Branch & Price (BP) qui combine un arbre de branchement et la génération de colonnes (GC). La GC est une méthode pour résoudre des problèmes avec beaucoup de variables. Elle s'appuie sur la résolution d'un problème maitre avec un sous-ensemble des variables et la résolution d'un sous-problème pour générer de nouvelles variables intéressantes. Dans le cas du MDVSP, lorsque la GC est utilisée, la modélisation utilise un graphe pour résoudre le sous-problème. Cependant, le BP a ses limites, en partie à cause de la dégénérescence de la GC. Différentes techniques, qui le rendent heuristique, sont utilisées pour l'accélérer : branchement heuristique, troncature de la GC, etc. Une autre manière d'accélérer la résolution par BP est de réduire a priori la taille des problèmes par des heuristiques. Une façon particulière de faire cela est d'utiliser l'apprentissage automatique pour prédire quels sont les arcs du graphe à conserver et ceux à éliminer dans la modélisation.

1.3 Objectifs de recherche

L'objectif de ce projet de recherche est de développer une méthode de résolution heuristique du EVSP rapide et efficace. Les instances que l'on veut pouvoir résoudre sont des instances multi-dépôts, avec plusieurs stations de recharge, un seul type de véhicules électriques, un processus de recharge non linéaire et la possibilité de recharger partiellement. À notre connais-

sance, peu de méthodes permettent de résoudre ce problème avec une bonne qualité de la solution en un temps raisonnable [6, 8–10]. La méthode développée devra être capable de résoudre des instances comportant jusqu'à 2500 trajets en moins de 50% du temps et avec un écart d'au plus 5% comparé à une résolution classique par BP.

La méthode proposée s'appuie sur une résolution heuristique par BP et une réduction de la taille du modèle du problème par un algorithme glouton et par apprentissage automatique. La réduction de la taille consiste en l'élimination d'arcs dans le graphe qui modélise le sous-problème de la GC. Pour la partie apprentissage automatique, on utilise un réseau de neurones pour graphe (*Graph Neural Network*) (GNN), entrainé de manière supervisée sur des instances résolues au préalable par BP, pour prédire quels seront les arcs utilisés dans la solution optimale.

1.4 Plan du mémoire

Le présent chapitre 1 introduit le travail présenté dans ce mémoire. Le chapitre 2 propose une revue de littérature des différentes méthodes pour résoudre le VSP et le EVSP ainsi que les différentes manières de combiner l'optimisation et l'apprentissage machine. Le chapitre 3 décrit en détails le problème résolu ainsi que la modélisation choisie. Ensuite, le chapitre 4 présente les différents éléments sur lesquels s'appuie la méthode de résolution développée. Le chapitre 5 présente les expériences menées et leurs résultats. Enfin, le chapitre 6 conclut ce mémoire.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre présente une revue de littérature des différentes problématiques dans lesquelles s'inscrit la résolution du problème d'horaires de véhicules électriques (*Electric Vehicle Scheduling Problem*) par GC et apprentissage automatique présentée dans ce travail. Tout d'abord, la section 2.1 présente les différentes méthodes pour résoudre le problème d'horaires de véhicules multi-dépôt (*Multi Depot Vehicle Scheduling Problem*). Ensuite, la section 2.2 présente l'évolution des méthodes et des formulations du EVSP. Enfin, la section 2.3 présente les manières possibles de combiner l'apprentissage automatique et la résolution de problème d'optimisation combinatoire.

2.1 Résolution du MDVSP

Le MDVSP a fait l'objet de nombreuses études depuis plusieurs décennies. Ces études proposent des méthodes de différentes natures comme la résolution exacte par solveur de PLNE, les méthodes génétiques, le BP avec la GC, les méthodes de recherche dans un voisinage ou des heuristiques gloutonnes. Une revue de littérature des différentes méthodes ainsi que la description du processus complet de planification du service par autobus peuvent être trouvées dans [3]. Une autre revue de littérature légèrement plus récente est proposée dans [11].

Parmi les méthodes recensées, on retrouve notamment la GC [12] combinée à un algorithme de Branch & Bound (BB) [13] avec un modèle de partitionnement d'ensembles. L'intérêt de la GC est démontrée sur des instances générées aléatoirement comme dans [13] : elle permet de résoudre de manière exacte des instances de 300 trajets et 6 dépôts en une heure, ce qui était très rapide en 1989. Dans [14], un modèle de PLNE avec des commodités multiples est utilisé et la relaxation linéaire du problème est résolue par GC. La méthode proposée permet de résoudre à l'optimalité des instances réelles de 8563 trajets en une dizaine d'heures. Avec une modélisation similaire, les auteurs de [15] résolvent le MDVSP par Branch & Cut (BC), c'est-à-dire en utilisant un arbre de branchement avec une stratégie d'exploration innovante et des plans coupants. Cette méthode permet d'accélérer l'algorithme de BB : il peut résoudre des instances réelles de 630 trajets en une vingtaine de minutes.

Une autre modélisation est possible pour le MDVSP, basée sur un réseau espace-temps [16]. La modélisation par un réseau espace-temps permet d'agréger les arcs dans un réseau de connexion classique via des arcs d'attente à des arrêts de bus pour réduire le nombre d'arcs et donc de variables. Il faut, dans un second temps, désagréger le flot sur les arcs d'attente

via une méthode de décomposition de flot. Cette modélisation permet de réduire considérablement la taille du problème jusqu'à diminuer de 99% le nombre de variables, ce qui permet de résoudre avec un solveur commercial des instances de 7068 trajets en 3 heures. Les auteurs de [17] utilisent également la modélisation espace temps et résolvent le problème grâce à la fixation des variables en utilisant le résultat du SDVSP associé au MDVSP. Des instances de plus de 2000 trajets sont résolues avec un grand nombre de dépôts en quelques heures (entre 1 et 5), mais les solutions peuvent ne pas couvrir tous les trajets. La même modélisation par un réseau espace-temps est utilisé dans [18], le MDVSP avec des types de véhicules multiples est résolu par une heuristique basée sur la GC après avoir réduit la taille du réseau espace-temps qui permet de résoudre des problèmes de 3000 trajets et 16 dépôts en moins d'une heure avec un écart de 1% à la valeur optimale.

Enfin, Pepin et al. [7] proposent 5 heuristiques différentes de résolution : "un algorithme de BC tronqué, une résolution par relaxation lagrangienne, un algorithme de GC tronqué, un algorithme de recherche locale par voisinage et un algorithme tabou". Parmi ces méthodes, la génération de colonnes tronquée performe le mieux. Elle permet de résoudre des instances de 1500 trajets et 8 dépôts avec un écart à la valeur optimale de moins de 1% en moins d'une heure. L'algorithme de recherche locale à grand voisinage (*Large Neighbourhood Search*) (LNS) donne aussi de bons résultats, il permet aussi de résoudre ces instances en moins d'une heure, mais avec un écart de 3,7% à la valeur optimale.

2.2 Résolution du EVSP

Cette section retrace l'histoire du EVSP, les différentes méthodes de résolution récentes du EVSP et les variantes du EVSP. Comparé au VSP, le EVSP doit prendre en compte l'autonomie de la batterie des autobus et la recharge des autobus. Le temps de recharge est un processus non linéaire dans la vie réelle. Il faut aussi prendre en compte le nombre de bornes de recharge disponible à chaque instant. De manière générale, tout le processus de planification aux échelles stratégique, tactique et opérationnelle doit être revu lorsque des autobus électriques sont employés à la place d'autobus traditionnels. Un aperçu des différents articles résolvant ces problèmes peut être trouvé dans [19]. En particulier, l'optimisation du plan de transformation de la flotte des autobus conventionnels en autobus électriques est décrite dans [20]. L'effet de l'électrification des véhicules pour les opérateurs de transports est analysé à l'aide de programmation mathématique par [21].

2.2.1 Historique

Le EVSP sous la forme actuelle qu'on lui connait, avec des autobus électriques nécessitant d'être rechargés régulièrement de manière partielle en un temps dépendant non linéairement de l'état de charge, s'est construit progressivement.

Autonomie de la batterie et temps de recharge constant

Tout d'abord, seulement l'autonomie et la recharge en un temps constant des batteries ont été pris en compte. En 2002, Haghani et Banihashemi [22] proposent de résoudre le VSP en rajoutant des contraintes de temps maximal en dehors du dépôt, qui est l'ébauche du EVSP. Une nouvelle formulation de ce problème ainsi que deux méthodes qui permettent de réduire la taille du problème d'en moyenne 78% sont données. Le problème est à la fois résolu de manière exacte en générant des contraintes et de manière heuristique. Des instances réelles de 5650 trajets sont résolues en un jour de temps de calcul.

En 2013, une première version du EVSP est réellement proposée par Chao et Xiaohong [23]. Les véhicules sont équipés de batteries électriques qui peuvent être échangées au cours de la journée en un temps fixe de 15 minutes. Celles-ci sont ensuite rechargées en un temps linéaire. La formulation est multiobjectif dans le but de minimiser le nombre de véhicules, le nombre de batteries de rechange et la demande en recharge. Le problème est résolu par un algorithme génétique. Les tests sont effectués sur des petites instances de 59 trajets.

Dans [24], on retrouve deux formulations différentes du EVSP, une qui prend en compte un échange de batterie ou une recharge rapide de durée fixe et l'autre pour des véhicules avec une distance maximale qu'ils peuvent parcourir. Des modèles de PLNE sont proposés. Des algorithmes basés sur la GC sont développés avec troncature de la GC et branchement heuristique pour les plus grandes instances. Les plus grandes instances résolues comptent 947 trajets et sont résolues en des temps variants entre 8 et 32 heures.

Adler et Mirchandani [25] en 2017 reprennent l'idée de la thèse d'Adler de 2014 de résoudre le problème d'horaires pour véhicules à carburant alternatif qui nécessitent d'être rechargés en temps constant dans des stations de recharge par une heuristique et par BP exact. L'heuristique permet de résoudre des problèmes de 4373 trajets en moins de 5 minutes. Sur des instances plus petites (jusqu'à 50 trajets), l'heuristique permet de trouver un nombre presque optimal de véhicules (moins de 3% d'écart à la valeur optimale). Dans [26], deux problèmes différents sont résolus : le problème d'horaires pour véhicules électriques seuls dont les batteries peuvent être chargées ou remplacées en 10 minutes et le problème pour véhicules mixtes avec des véhicules traditionnels et électriques. La modélisation est faite sur un réseau espace-

temps. Un algorithme exact et deux heuristiques inspirées de [25] sont proposés. Des instances jusqu'à 10000 trajets sont résolues. Toutefois, l'heuristique ne permet pas toujours de trouver une solution proche de la solution optimale suivant les instances considérées.

Temps de recharge linéaire

Les premiers modèles qui prennent en compte le temps de recharge non constant par rapport à l'état de charge des véhicules apparaissent ensuite. Dans [27] est développé un algorithme k-greedy pour résoudre un problème d'horaires pour flotte mixte (autobus électriques et autobus traditionnels) où la priorité est donnée aux véhicules électriques. Les instances testées sont de petite taille : 189 trajets, mais un temps de recharge linéaire est pris en compte. Dans le travail de Wen et al. [28], les instances générées sont résolues par un algorithme de recherche locale à grand voisinage adaptatif (*Adaptive Large Neighbourhood Search*) (ALNS) qui utilise différentes méthodes de destruction et une heuristique de regret pour la reconstruction des solutions. Un temps de recharge linéaire et la recharge partielle sont utilisés. Les plus grandes instances résolues comptent 500 trajets, 8 dépôts et 16 stations de recharge en une vingtaine de minutes. Sur des petites instances, l'heuristique permet de trouver des solutions à moins de 0,1% de la valeur optimale. En plus de montrer que le EVSP est NP-difficile, le travail de Sassi et Oulamara [5] modélise le temps de recharge à l'aide d'une discrétisation de la recharge en tranches de 15 minutes. Les autobus conventionnels et électriques sont considérés. Deux heuristiques sont proposées pour résoudre les plus grandes instances qui vont jusqu'à 120 trajets. L'heuristique globale proposée est très rapide (quelques minutes) et donne de bons résultats (20% des instances résolues à l'optimalité).

Temps de recharge non linéaire

Après, des modèles modélisant le temps de recharge non linéaire par rapport à l'état de charge sont apparus. La recharge non linéaire pour les véhicules électriques a d'abord été introduite par Montoya [29] pour le problème des tournées de véhicules électriques. La prise en compte de la non-linéarité de la recharge pour le EVSP est apparue avec van Kooten Niekerk et al. [30]. Les auteurs proposent plusieurs modèles de PLNE et méthodes de résolution. Le modèle le plus complexe permet de prendre en compte le coût de l'électricité, la dépréciation de la batterie et un processus de recharge non linéaire. La charge est discrétisée pour simplifier le modèle. Les méthodes de résolution sont : la résolution exacte par un solveur commercial, la GC et une heuristique de relaxation lagrangienne ajoutée à la GC. En pratique, les tests sont faits avec un modèle de recharge linéaire. Les plus grandes instances résolues ont 543 trajets et sont résolues en une quinzaine de minute avec l'heuristique.

Olsen et Kliewer [31] étudient les simplifications et approximations sur les modèles de recharge et montrent qu'une simplification du temps de recharge par un temps constant amène à une solution sous-optimale, tandis qu'une simplification par un temps linéaire peut mener à une solution non réalisable. Ils utilisent l'heuristique d'Adler [25] et ajoutent des recharges partielles par rétro-inspection. Le nombre de chargeurs nécessaires est également analysé. Les instances résolues sont très grandes (10710 trajets). Zhang et al. [6] résolvent le MDEVSP avec recharge partielle possible en ajoutant une contrainte de nombre de changements de ligne maximum par un algorithme de ALNS avec insertion par regret. Ils utilisent une fonction de recharge linéaire par morceaux et montrent qu'une approximation linéaire peut causer des problèmes en service. Les plus grandes instances résolues ont 518 trajets.

Recharge partielle des batteries

Certains modèles récents prennent aussi en considération la possibilité de recharger partiellement les batteries. Par exemple, [32] traite du EVSP avec des contraintes de relocation et prise en compte de la recharge partielle. La résolution est faite avec un algorithme de LNS avec de nouveaux opérateurs de destruction/construction. Les tests, réalisés sur des instances d'une centaine de trajets, montrent que l'heuristique est très rapide et donne des solutions presque optimales (moins de 0,1%). Zhang [33] s'intéresse aussi à la recharge partielle. Il résout le problème par ALNS. L'intérêt de la recharge partielle pour la qualité de la solution est démontré. Les tests comportent 200 trajets et prouvent la rapidité et l'efficacité de l'heuristique proposée. Dans [34], de petites instances (160 trajets) sont résolues par solveur en plusieurs heures. La particularité de la recharge est, qu'en plus d'être partielle, celle-ci est en continu au lieu d'être discrète et que le nombre de bornes de recharge est pris en compte.

2.2.2 Travaux récents

Des revues de littérature complètes sur le EVSP sont disponibles dans [35] et [36]. On liste ici les travaux les plus pertinents par rapport au EVSP étudié dans ce travail par type de méthode de résolution employée.

Résolution exacte par solveur PLNE

Le travail de Zhou et al. [37] modélise le SDEVSP avec dégradation de la batterie, modèle de recharge non linéaire, recharge partielle possible et 2 types de chargeurs différents comme un PLNE. Les instances, de petite taille (30 trajets), sont résolues en 5 heures, à l'aide d'un solveur commercial et de 3 types d'inégalité valides. La modélisation sur un réseau espace-

temps est employée dans [38] pour modéliser le SDEVSP, un temps de recharge linéaire et recharge partielle possible. Leur objectif étant de maximiser le nombre de routes réalisables pour les véhicules électriques, ils modifient la phase de décomposition de flot après avoir construit le réseau et résolu un problème de flot maximal par solveur commercial sur le réseau espace-temps pour obtenir une solution. La recharge est introduite a posteriori. Ils peuvent résoudre des instances jusqu'à 10710 trajets en 30 secondes. Suivant la distribution des trajets dans les instances considérées, la proportion de véhicules électriques varie entre 30% et 70%.

Branch & Price

De leur côté, Jiang et al. [39] résolvent des instances jusqu'à 400 trajets en moins de 72 minutes grâce à une méthode de BP initialisée par une heuristique, utilisant des heuristiques dans les sous-problèmes et avec un branchement sur le flot dans les arcs. Dans la résolution des sous-problèmes, le temps de charge est pris en compte, mais la recharge n'est pas planifiée à proprement parler à ce moment-là, elle est planifiée à la fin de la résolution du BP via la résolution d'un problème de planification de recharge électrique pour route fixe pour chaque route de la solution optimale. Pour des instances de cette taille, le BP donne de meilleures solutions que des heuristiques comme ALNS mais est plus long. La particularité de la modélisation est que les trajets ont une fenêtre de temps pour être desservis et non un horaire fixe. Zhang et al. [8] utilisent aussi un algorithme de BP pour résoudre le EVSP. Leur modélisation prend en compte la recharge non linéaire, le nombre de chargeurs à une station de recharge, la recharge partielle et la dégradation des batteries dans leur fonction objectif, mais ne considère qu'un seul dépôt et qu'une seule borne de recharge. Ils emploient un algorithme spécial dans les sous-problèmes, la stabilisation duale, un branchement sur les inter-tâches ainsi qu'un branchement heuristique pour le nombre de chargeurs utilisé. Les plus grandes instances comptent 160 trajets et sont résolues en environ 2 heures. De Vos et al. [40] résolvent le MDEVSP en prenant en compte un temps de recharge linéaire, la recharge partielle et le nombre de stations de recharge. Les états de recharge sont discrétisés dans la modélisation. La résolution du problème est faite par GC heuristique et résout des instances de 800 trajets en 28 heures avec un écart d'optimalité de moins de 3,5%. Yildirim et al. [41] résolvent le MDEVSP avec plusieurs types de véhicules électriques, plusieurs types de recharge (dont une recharge sans fil) et recharge non linéaire partielle par GC. Ils ont développé un nouvel algorithme pour les plus courts chemins avec ressources pour la GC reposant sur la programmation dynamique. Des instances de 800 trajets sont résolues en 1,7 heure en moyenne.

Décomposition de Benders

Le problème d'horaires à flotte mixte pour un seul dépôt avec recharge partielle linéaire en temps continu et prise en compte du nombre de chargeurs est résolu à l'aide d'une décomposition de Benders dans [42]. Les instances de 250 trajets sont résolues en moins d'une heure.

Algorithme génétique

Wang et al. [43] résolvent le MDEVSP avec un temps de recharge linéaire par un algorithme génétique qui utilise les colonnes de la GC de la relaxation linéaire du problème. Leur méthode sur des instances allant jusqu'à 510 trajets est 40 fois plus rapide que le BP si l'on autorise la couverture en double ou la non-couverture de certains trajets dans la solution finale.

Autres heuristiques

Comme cité dans la section précédente, il est possible de résoudre le EVSP par des algorithmes de ALNS. Il existe aussi d'autres heuristiques pour résoudre ce problème. Tout d'abord, Jovanovic et al. [44] résolvent le SDEVSP et considèrent un temps de recharge linéaire. Ils ont créé un nouvel algorithme glouton auquel est ajouté de l'aléatoire et une recherche locale qui permet de résoudre en moins d'une heure des instances de 2000 trajets. Le problème d'horaires pour flotte mixte avec un seul dépôt est résolu dans [45] par un algorithme heuristique itératif à deux niveaux : le premier par recuit simulé pour résoudre le problème d'horaires et le deuxième par programmation dynamique pour placer les évènements de recharge en prenant en compte le coût variable de l'électricité. Deux fonctions objectif différentes sont proposées : une qui minimise les coûts et une qui minimise les émissions de gaz à effet de serre. Des instances de 575 trajets sont résolues en une dizaine de minutes.

2.2.3 Problèmes reliés au EVSP

Autour du EVSP, on retrouve de nombreux problèmes qui prennent en compte le EVSP ou des variantes du EVSP. On en décrit une partie dans cette section.

De nombreux articles cherchent à résoudre, en plus du EVSP, la phase tactique du nombre et/ou de la localisation des bornes de recharge comme [9, 46–49]. Dans cette même démarche d'optimisation conjointe, certains travaux s'intéressent aussi à la résolution du problème de chauffeurs, qui cherche à trouver des horaires pour les chauffeurs d'autobus [10, 50], aux demandes origines-destination des passagers [51] ou à la création des horaires [52]. Le décalage

des horaires est aussi autorisé dans certains travaux comme dans [53].

Certaines études considèrent également que les temps de trajets ne sont pas fixes et sont aléatoires comme [54]. D'autres, comme [55] et [56] s'intéressent aussi à l'impact sur le réseau électrique du problème d'horaires.

2.3 Combiner apprentissage automatique et optimisation combinatoire

Combiner apprentissage automatique et optimisation combinatoire est une pratique populaire actuellement et peut se faire de diverses manières. Bengio et al. [57] expliquent les différentes manières d'utiliser l'apprentissage automatique pour résoudre des problèmes d'optimisation et ils listent des exemples d'application. D'après cet article, un modèle d'apprentissage automatique peut être utilisé pour trouver une solution à un problème d'optimisation directement sans utiliser d'autre algorithme d'optimisation. Il peut aussi être utilisé au préalable d'un algorithme d'optimisation pour : simplifier le problème en approximant l'objectif ou les contraintes [58] ou bien en réduisant le nombre de variables, choisir des paramètres de résolution ou simplifier des problèmes complexes en problèmes plus simples à résoudre [59]. Une dernière manière de combiner apprentissage automatique et optimisation combinatoire est de faire appel à l'apprentissage automatique au cours de la résolution pour prendre des décisions qu'il serait coûteux d'évaluer dans un algorithme classique de résolution. Deux méthodes d'apprentissage sont envisagées dans [57] pour l'optimisation combinatoire : l'apprentissage par imitation où l'on cherche à reproduire le plus fidèlement possible les décisions d'un expert et l'apprentissage par renforcement où l'on cherche à trouver une politique d'action dans un environnement donné qui permet de maximiser des gains prédéfinis.

Par exemple, dans le cas de problèmes résolus par GC, on peut prédire quelles sont les colonnes les plus pertinentes à ajouter au problème maitre à chaque itération [60]. Une autre façon d'utiliser l'apprentissage automatique dans la GC est de prédire, à chaque itération de GC, quels arcs doivent être considérés pour la recherche de plus courts chemins dans la résolution du sous-problème [61]. Une autre façon d'utiliser l'apprentissage automatique dans le BP est d'avoir des méthodes de branchement (sélection de variables et des nœuds) qui prennent en compte des modèles d'apprentissage [62]. Ces exemples utilisent l'apprentissage automatique au sein d'un algorithme d'optimisation pour prendre des décisions rapidement, voire pour trouver de meilleures solutions.

Dans le cas du MDVSP, on peut chercher à réduire le nombre de variables pour accélérer la résolution du problème comme expliqué plus haut dans [17]. Il est alors possible d'utiliser de l'apprentissage automatique pour cela comme exploré dans [63] sur un réseau espace-temps.

Dans cet article, une méthode utilisant l'apprentissage supervisé est proposée pour éliminer des arcs avant la résolution en nombres entiers du problème. Cet exemple illustre l'utilisation de l'apprentissage automatique pour simplifier un problème d'optimisation.

Le problème du voyageur de commerce ou le cas plus général du problème de tournées de véhicules sont des applications importantes de l'apprentissage automatique pour résoudre des problèmes combinatoires. Les travaux [64] et [65] passent en revue les différentes méthodes d'apprentissage automatique qui peuvent être utilisées pour accélérer la résolution de problèmes combinatoires sur des graphes comme le problème de tournées de véhicules. Dans la plupart des cas, pour ces problèmes-là, l'apprentissage automatique est utilisé pour trouver directement une solution. Des modèles d'apprentissage profond (qui sont des réseaux de neurones avec un nombre important de couches cachées) sont souvent employés, car ils ont des meilleures performances. Les réseaux de neurones pour graphe (GNN) y sont notamment mentionnés.

En effet, les GNN sont très prometteurs pour la combinaison intelligence artificielle et optimisation de problèmes modélisés sur un graphe. Cappart et al. [66] en recensent plusieurs applications. On peut notamment citer [67] et [68] qui résolvent le problème du voyageur de commerce avec des GNN et [69] qui est la première utilisation de l'attention des réseaux de neurones dans de l'apprentissage pour un problème de graphes pour résoudre entre autres le problème de tournées de véhicules. Les GNN sont une forme d'apprentissage profond pour des données d'entrée structurées sur un graphe. Ils ont été introduits par Scarselli [70] qui explique qu'ils permettent de généraliser les réseaux de neurones récurrents et les chaines de Markov sur n'importe quelle structure de graphe. Dans un GNN, chaque nœud a un état, actualisé en fonction de relations (arêtes) qu'il a avec ses nœuds voisins. L'article décrit toute la théorie des GNN, sur comment entrainer ces réseaux de neurones, la complexité des réseaux, puis fournit des expériences très probantes pour imiter des fonctions basées sur les graphes ou sur certains nœuds. Pour des revues de littérature sur l'utilisation de ce type de réseaux de neurones, on peut se référer à [71] ou [72].

Dans le présent travail, les GNN seront adoptés pour simplifier le MDEVSP en réduisant le nombre de variables considérées dans la modélisation du problème avec la même idée que [63].

CHAPITRE 3 DESCRIPTION ET MODÉLISATION DU PROBLÈME

Dans ce chapitre, la modélisation du MDEVSP étudié est détaillée. Tout d'abord, le problème est introduit dans la section 3.1 puis la formulation mathématique est présentée dans la section 3.2. Enfin, le graphe utilisé dans la résolution par GC est décrit dans la section 3.3.

3.1 Description

Le MDEVSP est défini pour une seule journée à la fois. Un ensemble de n trajets à couvrir $T = \{t_1, t_2, ..., t_n\}$, dont le point de départ, le point d'arrivée et les horaires sont connus, est considéré. Pour chaque trajet t, on note q_t^s le temps de départ du trajet t et q_t^e le temps d'arrivée du trajet t. On connait pour chaque paire de points du réseau de transport le temps de trajet à vide entre ces points. En particulier, pour un trajet t, on note ρ_t le temps de trajet à vide pour relier le départ à l'arrivée du trajet en question. On suppose que tous les véhicules à vide roulent à une vitesse constante notée α. Pour chaque trajet t, on connait également la consommation d'énergie $\sigma_t = e_{mouv}\rho_t\alpha + e_{arr}(q_t^e - q_t^s - \rho_t)$ où e_{mouv} est la consommation d'énergie par unité de distance parcourue et e_{arr} est la consommation d'énergie à l'arrêt (avec le moteur en marche, par exemple lorsque les passagers montent dans le bus) par unité de temps. On calcule la consommation d'énergie des voyages à vide de manière similaire sans prendre en compte la consommation à l'arrêt. On considère également l'ensemble des m dépôts $D = \{d_1, d_2, ..., d_m\}$. Chaque dépôt d a une capacité de véhicules v^d. Enfin, on a l'ensemble des r stations de recharge $H = \{h_1, h_2, ..., h_r\}$ et à chaque station de recharge h, on note u^h le nombre de bornes de recharge.

On utilise un modèle linéaire par morceaux pour le modèle de recharge. On note f la fonction de recharge qui, à partir de l'état de la batterie σ_s et du temps de recharge Δ, donne l'état de charge final $\sigma_e = f(\sigma_s, \Delta)$. Cette fonction est illustrée dans la figure 3.1. Pour suivre le nombre de véhicules présents simultanément à une station de recharge, on discrétise le temps en intervalles de δ minutes et l'on note $P = \{p_1, p_2, ..., p_N\}$ l'ensemble des N périodes de temps créées permettant de couvrir la journée étudiée. Chaque période de temps p a une heure de début τ_p^s et une heure de fin τ_p^e. On suppose que le temps de recharge est toujours un multiple entier de δ. Cette hypothèse permet de simplifier la formulation du problème. De plus, il est raisonnable d'imaginer qu'en pratique on demanderait à un chauffeur de recharger son véhicule un nombre rond de minutes (15 minutes plutôt que 17,5).

On considère que tous les véhicules sont identiques avec une capacité de batterie notée σ_{max}

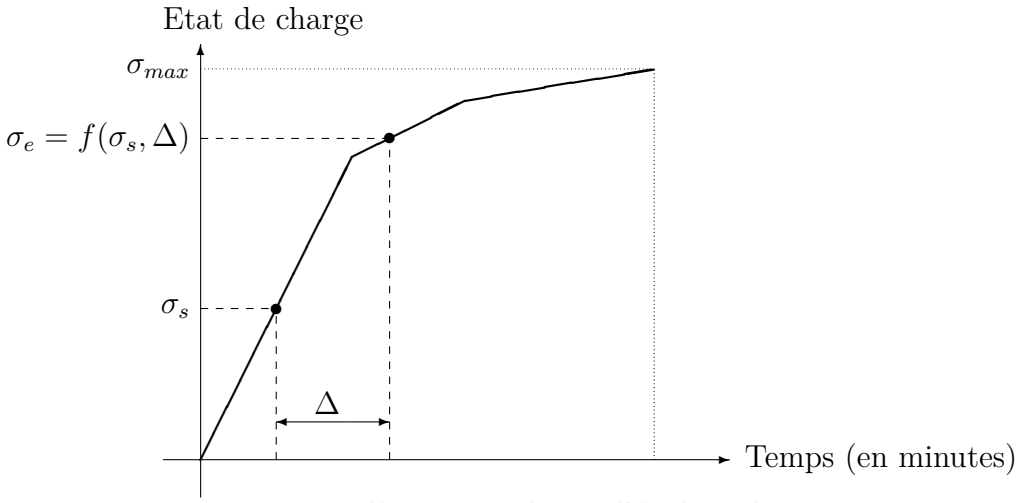

FIGURE 3.1 Illustration du modèle de recharge

et qu'ils partent du dépôt le matin chargés au maximum [1].

On appelle une *route* un enchaînement de voyages qui peuvent à la fois être des *trajets à desservir* et des *trajets à vide* entre deux trajets à desservir, des trajets vers ou depuis une station de recharge ou un dépôt. Une route est valide si elle commence et finit au même dépôt, si le niveau de charge de la batterie reste entre un état de charge minimal prédéterminé σ_{min} (ici 0) et la capacité maximale σ_{max}, s'il y a toujours une recharge effectuée si l'on se rend à une station de recharge et s'il n'y a pas plus de β minutes (ici 45 minutes) entre 2 trajets à desservir consécutivement. Il est possible de retourner dans un des dépôts pendant la journée, mais si c'est le cas, l'autobus doit stationner au dépôt pendant au moins γ minutes (ici 30 minutes). On appelle *horaire* un ensemble de routes.

Le coût d'une route peut être séparé en six parties : le coût fixe d'utilisation d'un véhicule noté c_f, le coût d'attente par minute à un arrêt noté c_a, le coût de voyage à vide par kilomètre noté c_v, le coût fixe de retourner au dépôt pendant la journée c_d^f, le coût fixe d'aller à la recharge c_r^f et le coût d'attente à la recharge par unité de temps passée à la recharge c_r^v. On ne paie pas de coût d'attente aux dépôts, car l'on suppose que le chauffeur peut quitter son véhicule et laisser quelqu'un d'autre s'en occuper.

L'objectif du MDEVSP est de trouver un ensemble de routes valides de coût minimum qui permet de desservir tous les trajets tout en respectant la capacité des dépôts et des stations de recharge.

1. Les véhicules ne se rechargeront pas nécessairement tous au même moment car ils ne partent pas tous à la même heure du dépôt et peuvent se recharger à partir de n'importe quel état de charge

3.2 Formulation mathématique

Le MDEVSP peut se formuler comme un problème de partitionnement d'ensemble. On note S^d l'ensemble des routes réalisables à partir du dépôt d et pour une route s, on note c_s son coût et x_s la variable binaire qui vaut 1 si la route est utilisée et 0 sinon. On note également a_s^t le paramètre binaire qui vaut 1 si trajet t est couvert par la route s et 0 sinon et b_s^{ph} le paramètre binaire qui vaut 1 si la route s prévoit une recharge à la station h à la période de temps p et 0 sinon. Le problème s'exprime alors comme le PLNE ci-dessous :

$$\min \sum_{d \in D} \sum_{s \in S^d} c_s x_s \tag{3.1}$$

$$\text{sujet à : } \sum_{d \in D} \sum_{s \in S^d} a_s^t x_s = 1, \ \forall t \in T, \tag{3.2}$$

$$\sum_{d \in D} \sum_{s \in S^d} b_s^{ph} x_s \leq u^h, \ \forall p \in P, \forall h \in H, \tag{3.3}$$

$$\sum_{s \in S^d} x_s \leq v^d, \ \forall d \in D, \tag{3.4}$$

$$x_s \in \{0, 1\}, \ \forall d \in D, \forall s \in S^d. \tag{3.5}$$

L'objectif 3.1 est de minimiser le coût total des routes. Les contraintes 3.2 assurent que chaque trajet soit desservi une et une seule fois. Les contraintes 3.3 modélisent la capacité des bornes de recharge à chaque période de temps. Les contraintes 3.4 permettent de respecter la capacité des dépôts. Enfin, les contraintes 3.5 assurent la binarité des variables.

3.3 Graphe de construction de routes

Trouver des horaires réalisables à partir d'un dépôt d peut se reformuler comme un problème de plus court chemin avec contraintes de ressources dans un graphe $\mathcal{G}^d$ que l'on va décrire dans cette partie. La modélisation utilisée est un graphe de connexions avec un réseau espace-temps pour les stations de recharge et le retour au dépôt comme proposé dans [73]. Le graphe complet qui regroupe tous les dépôts D est noté $\mathcal{G} = (\mathcal{N}, \mathcal{A})$.

Comme illustré dans la figure 3.2, on crée un nœud origine o_d et un nœud destination k_d qui représentent le dépôt d en début et fin de journée, respectivement. On crée également pour chaque période de temps p et chaque dépôt d' un nœud de retour au dépôt $s_p^{d'}$. On crée pour chaque trajet t_i un nœud associé t_i. On crée pour chaque station de recharge h et chaque période de temps p, un nœud d'attente à la station de recharge w_p^h et un nœud de recharge

r_p^h. Dans la figure 3.2 ne sont illustrés les noeuds que pour une station de recharge et un dépôt d' pour le retour au dépôt.

Pour chaque trajet t_i, on crée un arc entre l'origine o^d du graphe $\mathcal{G}^d$ et le trajet t_i et un arc entre le trajet t_i et la destination k^d du graphe $\mathcal{G}^d$. Pour chaque trajet et pour chaque station de recharge h, on crée un arc entre le dernier nœud d'attente w_p^h qui permet d'atteindre à temps le début du trajet t_i. De même, on crée un arc entre le nœud t_i et le premier nœud d'attente w_p^h qui puisse être atteint après avoir effectué le trajet t_i. Les arcs bleus sont les arcs reliant les trajets aux stations de recharge dans la figure. Puis, pour chaque trajet et pour chaque dépôt, on crée un arc entre le dernier nœud de retour au dépôt $s_p^{d'}$ qui permet d'atteindre à temps le début du trajet t_i. De plus, on crée un arc entre le nœud t_i et le nœud de retour au dépôt $s_p^{d'}$ de telle sorte que $s_{p'}^{d'}$ puisse être atteint après avoir effectué le trajet t_i et $\tau_p^s - \tau_{p'}^s \geq \gamma$. Cette dernière contrainte permet de modéliser le fait qu'un véhicule est obligé de rester au dépôt au moins γ minutes s'il s'y rend. Les arcs verts sont les arcs reliant les trajets au dépôt dans la figure. Ensuite, on crée un arc entre les trajets t_i et t_j s'il est possible de desservir le trajet t_j après avoir desservi le trajet t_i et si le temps entre les deux trajets est inférieur à β minutes. Ce sont les arcs rouges, dans la figure, qui relient les trajets entre eux. Enfin, pour chaque station de recharge h et chaque période de temps p_i, on crée un arc d'attente entre les nœuds $w_{p_i}^h$ et $w_{p_{i+1}}^h$ et entre les nœuds $r_{p_i}^h$ et $r_{p_{i+1}}^h$ et des arcs de recharge entre les nœuds $w_{p_i}^h$ et $r_{p_{i+1}}^h$. De même, pour chaque dépôt d' et chaque période de temps p_i, on crée un arc d'attente entre les nœuds $s_{p_i}^{d'}$ et $s_{p_{i+1}}^{d'}$.

Chaque arc a un coût associé calculé à partir de la structure de coût expliquée précédemment.

Pour modéliser la recharge, on utilise une ressource déchargement de la batterie dans ce graphe. À chaque arc, on associe une valeur (positive) de cette ressource qui correspond à la consommation d'énergie sur cet arc. La valeur (négative) de la ressource sur les arcs entre deux nœuds recharge n'a pas une valeur fixe, elle dépend du chemin considéré dans le graphe. Pour un chemin donné partant de l'origine, la valeur de la ressource est calculée à partir de la valeur de la ressource déchargement à l'origine de l'arc en utilisant la fonction de recharge f décrite à la section 3.1 et la durée de l'arc de recharge. Enfin, pour chaque nœud, on associe une fenêtre de validité de cette ressource déchargement qui est que la ressource doit être tout le temps entre $\sigma_{min} = 0$ et σ_{max}, où σ_{min} est l'état de charge minimum à respecter. On rajoute deux ressources en plus pour modéliser la contrainte de la recharge obligatoire dans le cas où l'on va à une station de recharge. La première ressource modélise le fait qu'un véhicule soit allé se charger et la deuxième ressource modélise le fait qu'un véhicule soit allé dans une station de recharge. Les valeurs de consommation et les fenêtres de valeur de ces ressources font en sorte qu'un véhicule qui va dans une station de recharge est obligé de s'y

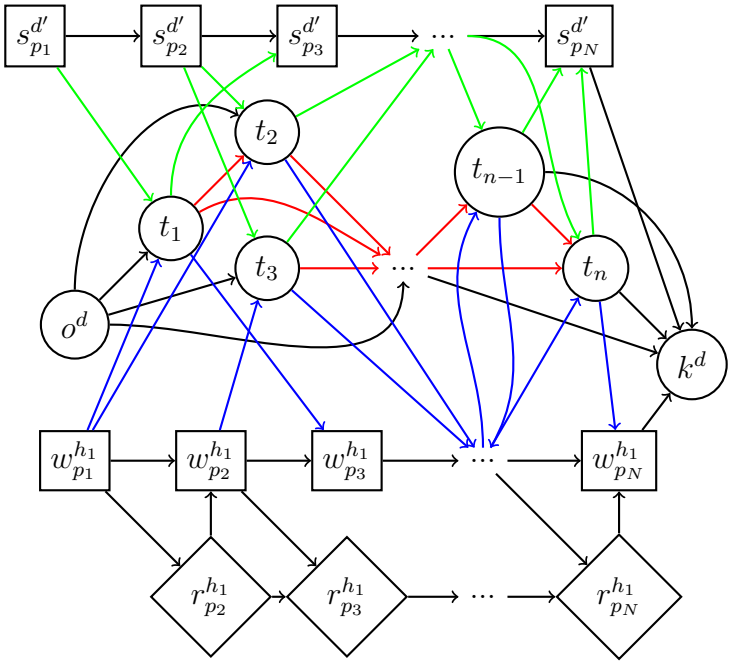

FIGURE 3.2 Graphe $\mathcal{G}^d$ pour le dépôt d

charger avant de la quitter. Des détails sur les ressources sont donnés en annexe A.

Un chemin de o^d vers k^d qui respecte les contraintes sur les ressources est une route réalisable. Son coût est la somme des coûts des arcs qui le composent. Cependant, lors de la recherche de plus courts chemins dans ce graphe, les coûts sur les arcs seront modifiés par les variables duales du problème en cours de résolution comme expliqué dans la section 4.1.1.

CHAPITRE 4 RÉSOLUTION DU PROBLÈME

Dans ce chapitre, les différentes méthodes développées pour résoudre le MDEVSP sont présentées. Ces méthodes reposant toutes sur la méthode du BP, celle-ci est présentée dans la section 4.1. La section 4.2 détaille les différentes méthodes de résolution proposées combinant un algorithme glouton et une partie apprentissage basée sur les réseaux de neurones pour graphes.

4.1 Branch & Price

La méthode du BP [74] est une méthode permettant de résoudre des problèmes mathématiques en nombres entiers de grande taille. Elle utilise le principe du BB où chaque nœud de relaxation linéaire est résolu par la méthode de la GC, qui est présentée dans la section 4.1.1. Les différentes manières de brancher qui sont spécifiques au BP sont expliquées dans la section 4.1.2. Enfin, des stratégies d'accélération de la résolution du MDEVSP par BP sont étudiées dans la section 4.1.3.

4.1.1 Génération de colonnes

La génération de colonnes est une méthode de résolution de problèmes d'optimisation proposée la première fois pour résoudre le MDVSP par Ribeiro et Soumis [12] en 1994. La GC permet de résoudre un problème de programmation linéaire avec beaucoup de variables que l'on appelle problème maitre (MP). On suppose que ces variables peuvent être générées facilement par un problème auxiliaire appelé sous-problème (SP). Souvent, le problème mathématique demande à être réécrit pour pouvoir être résolu par GC. Cette reformulation peut être faite par la décomposition de Dantzig-Wolfe [75].

La GC s'applique à toute sorte de problèmes. Cependant, le fonctionnement de la GC pour résoudre la relaxation linéaire du modèle (3.1) à (3.5) du MDEVSP est expliqué dans ce qui suit.

À chaque itération i de la GC, le problème (4.1) à (4.5) est résolu sur un sous-ensemble $\Omega_i \subset \bigcup_{d \in D} S^d$ des routes possibles, comme accentué en bleu dans les équations. On appelle ce problème le problème maitre restreint (RMP_i). Il correspond au modèle suivant (où le fait

que l'on résout la relaxation linéaire est accentué en orange) :

$$\min \sum_{s \in \Omega_i} c_s x_s \tag{4.1}$$

$$\text{sujet à : } \sum_{s \in \Omega_i} a_s^t x_s = 1, \ \forall t \in T, \tag{4.2}$$

$$\sum_{s \in \Omega_i} b_s^{ph} x_s \leq u^h, \ \forall p \in P, \forall h \in H, \tag{4.3}$$

$$\sum_{s \in S^d \cap \Omega_i} x_s \leq v^d, \ \forall d \in D, \tag{4.4}$$

$$x_s \geq 0, \ \forall s \in \Omega_i. \tag{4.5}$$

On résout ensuite le SP en utilisant les variables duales (π^i, σ^i, μ^i) obtenues en résolvant le (RMP$_i$). En effet, la solution trouvée par le MP à l'étape i est optimale si aucune variable $s \in \bigcup_{d \in D} S^d$ n'a un coût réduit négatif, c'est-à-dire, qu'aucune variable qui n'avait pas été prise en compte gagnerait à être ajoutée au problème. Le SP consiste alors à trouver des colonnes (des variables) x_s de coût réduit négatif où le coût réduit $\bar{c}_s$ d'une variable x_s est calculée par la formule : $\bar{c}_s = c_s - \sum_{t \in T} \pi_t^i a_s^t - \sum_{p \in P} \sum_{h \in H} \sigma_{ph}^i b_s^{ph} - \sum_{d \in D} \mu_d^i \eta_s^d$, où η_s^d vaut 1 si la route s est associée au dépôt d et 0 sinon. S'il n'existe aucune colonne de coût réduit négatif, la solution optimale a été trouvée. Autrement, on peut sélectionner un certain nombre de colonnes de coût réduit négatif et les ajouter à Ω_i pour former Ω_{i+1}. On peut en effet, choisir d'ajouter une seule colonne ou bien un nombre maximal de colonnes ou bien toutes les colonnes. Le nombre de colonnes ajoutées à chaque itération influence le temps pour résoudre une itération et le nombre d'itérations pour atteindre la solution optimale. Ensuite, une nouvelle itération $i + 1$ de GC est faite. On peut montrer que la GC converge en un nombre d'étapes fini vers la solution optimale si elle existe puisque l'on ajoute des nouvelles variables à chaque itération où la solution du (RMP$_i$) n'est pas optimale (les variables déjà générées ne peuvent pas avoir un coût réduit négatif, donc ne peuvent pas être générées plus d'une fois) et puisqu'il y a un nombre fini de variables.

À la première itération de GC, on peut utiliser une solution heuristique, une solution précédente dans l'arbre de branchement ou bien des variables artificielles.

Pour que la GC soit efficace, il faut que le SP associé soit facile et rapide à résoudre puisque l'on va le résoudre beaucoup de fois. Les détails sur la résolution du SP dans le cas du MDEVSP sont décrits ci-après. Tout d'abord, puisque chaque colonne est spécifique à un dépôt donné, on peut résoudre un SP par dépôt. Cela permet d'ailleurs de sélectionner plus d'une colonne à ajouter au MP à chaque itération puisqu'il y a plusieurs SPs. À l'itération i, le SP pour le dépôt d est noté (SP$_d^i$) et correspond à un plus court chemin dans le graphe

présenté en section 3.3 entre o^d et k^d. Le coût réduit d'un chemin est $c_s - \sum_{t \in T} \pi_t^i a_s^t - \sum_{p \in P} \sum_{h \in H} \sigma_{ph}^i b_s^{ph} - \mu_d^i$. Il faut alors modifier le coût des arcs dans le graphe de la façon suivante pour que le coût réduit d'un chemin donné soit bien la somme des arcs qui le constituent :

- Le coût $c_{(o^d,t_j)}$ des arcs de la forme (o^d, t_j) devient $c_{(o^d,t_j)} - \mu_d^i$.
- Le coût $c_{(n,t)}$ des arcs de la forme (n, t), où n est un nœud quelconque, devient $c_{(n,t)} - \pi_t^i$.
- Le coût $c_{(n,r_p^h)}$ des arcs de la forme (n, r_p^h), où n est un nœud recharge (attente ou charge), devient $c_{(n,r_p^h)} - \sigma_{ph}^i$.

En effet, pour une route donnée partant du dépôt d, il faut prendre une fois en compte la variable duale μ_d^i. On le fait donc sur le premier arc. Il faut aussi prendre en compte pour chaque trajet t la variable π_t^i ce qu'on fait sur chaque arc entrant dans le noeud trajet t. Enfin, pour chaque station de recharge h et chaque pas de temps p, il faut prendre en compte la variable duale σ_{ph}^i associée au nombre de véhicules se rechargeant en même temps. On la prend donc en compte dans tous les arcs entrant dans le noeud r_p^h.

De plus, pour qu'une route soit réalisable, il faut que celle-ci respecte les ressources introduites dans la section 3.3. Formellement, une ressource est une valeur prise à chaque nœud dans un chemin qui est fonction de la valeur de la consommation de la ressource sur l'arc entrant dans le nœud et de la valeur de la ressource au dernier nœud dans le chemin. On a en plus une fenêtre de ressource pour chaque nœud indiquant la faisabilité d'un chemin. Pour résoudre ce type de problème de plus court chemin avec des contraintes de ressources, on utilise la programmation dynamique [73] [76]. L'idée est d'étendre itérativement le chemin minimal, qui est juste le nœud source, en des chemins réalisables et optimaux jusqu'à atteindre le nœud destination. Les différents chemins partiels sont stockés à l'aide d'étiquettes (labels). On appelle ce genre d'algorithme un algorithme d'étiquetage. À chaque itération, on regarde les derniers nœuds atteints par les différents chemins et on étend les chemins aux nœuds réalisables suivants. On utilise des règles de dominance pour éliminer des chemins partiels sous-optimaux. Des détails sur le fonctionnement de l'algorithme d'étiquetage sont donnés en annexe A.

La GC permet donc de résoudre rapidement un problème de programmation linéaire avec beaucoup de colonnes.

4.1.2 Méthodes de branchement

La méthode BP permet de résoudre un problème linéaire en nombres entiers en faisant appel à un arbre de branchement comme dans l'algorithme de BB en résolvant à chaque nœud de l'arbre, la relaxation linéaire par GC. Les règles de branchement pour trouver une solution

entière sont spécifiques au BP et dépendent fortement du problème étudié.

Dans le cas du MDEVSP, voici une liste des différentes méthodes de branchement qui permettent une exploration exacte de l'arbre de branchement que l'on peut appliquer :

- Branchement sur le nombre de véhicules utilisés pour un dépôt.
- Branchement sur les *inter-tâches*. Une inter-tâche est une succession de deux tâches, ici les trajets. Ainsi, la décision sur un des noeuds fils sera que le trajet i est desservi directement après le trajet j tandis que l'autre sera que le trajet i n'est pas desservi après le trajet i. Cette décision s'applique aussi bien si les deux trajets sont directement connectés ou bien s'ils sont connectés en passant par une station de recharge ou un dépôt.
- Branchement sur le flot d'un arc. Par exemple, on peut choisir que le flot sur un arc $(w^h_{p_i}, r^h_{p_{i+1}})$ soit supérieur ou égal à 1 ou bien nul.

Pour accélérer l'algorithme de BP, on peut aussi utiliser des règles de branchement plus agressives qui ne permettent pas une exploration complète de l'arbre de branchement comme :

- La fixation de colonnes. On peut, par exemple, fixer un seuil sur la valeur des variables au-delà duquel toutes les variables associées aux colonnes seront fixées définitivement dans tous les nœuds-fils à 1.
- La fixation d'inter-tâches à 1. On peut par exemple, fixer toutes les inter-tâches à 1 au-dessus d'une valeur seuil.

Ces derniers branchements ne permettent pas de trouver la solution optimale à coup sûr, mais permettent de trouver une solution réalisable très rapidement. Dans le problème étudié, les solutions heuristiques trouvées sont de bonne qualité.

Les méthodes de branchement choisies peuvent être implémentées au niveau du MP comme le branchement sur le nombre de véhicules, tandis que d'autres comme le branchement sur le flot d'un arc entre deux trajets sont implémentées au niveau du SP en éliminant du graphe, dans lequel les plus courts chemins sont cherchés, les arcs qui conviennent. Par exemple, la condition que le flot sur l'arc $(w^h_{p_i}, r^h_{p_{i+1}})$ soit supérieur ou égal à 1 est implémentée au niveau du problème maître en rajoutant une nouvelle contrainte tandis que la condition que le flot soit nul est implémentée au niveau du sous-problème en éliminant l'arc en question.

Plusieurs méthodes de branchement peuvent être combinées en favorisant certaines méthodes. Dans le présent travail, la fixation de colonnes est privilégiée puis la fixation heuristique d'inter-tâches et enfin la méthode dichotomique de flot sur un arc pour les derniers nœuds de l'arbre de branchement.

4.1.3 Stratégies d'accélération

Malgré l'utilisation de la GC et le branchement heuristique, pour les grandes instances, les temps de convergence peuvent devenir très longs, en partie à cause de la dégénérescence. La dégénérescence est un phénomène qui apparait lorsqu'une même solution peut avoir différentes bases. Il peut y avoir alors beaucoup d'itérations du simplexe lors de la résolution d'une itération de GC et, quand de nouvelles colonnes sont ajoutées, celles-ci ne modifient pas la solution courante et restent à une valeur de zéro. La résolution du MP devient donc plus longue à cause de la dégénérescence. La résolution du SP peut aussi être plus compliquée pour les grandes instances. Plusieurs solutions sont utilisées pour trouver une solution presque optimale plus rapidement.

Tout d'abord, on peut utiliser la méthode des perturbations fixes des contraintes de couverture des trajets comme dans [77] pour accélérer sa résolution. Les perturbations sont alors enlevées sur la fin de l'arbre de branchement pour trouver une solution entière non perturbée.

Ensuite, on peut tronquer la GC à chaque résolution du MP lorsque le coût de la solution ne décroît pas plus d'un certain seuil sur un nombre d'itérations de GC données. Cela permet aussi d'accélérer l'algorithme de BP et d'éviter la dégénérescence.

Enfin, pour les très gros problèmes où la résolution du SP devient, elle aussi, longue, on peut résoudre chaque nœud de l'arbre de branchement sans utiliser les règles de dominance sur les ressources dans la résolution du SP dans un premier temps, jusqu'à que le coût de la solution ne décroît plus suffisamment, puis on prend en compte les règles de dominance sur toutes les autres ressources dans une deuxième partie. Il est aussi possible dans la deuxième partie d'utiliser les règles de dominance uniquement pour la ressource d'état de la batterie. Des tests réalisés avec et sans ces dominances heuristiques ont montré qu'elles permettent d'accélérer la résolution du SP.

Malheureusement, malgré toutes ses solutions, la résolution d'instances peut être lente et on cherche une manière d'accélérer la résolution. L'idée développée dans cette étude est de n'utiliser qu'un sous-ensemble d'arcs du graphe décrit dans la figure 3.2 sur lequel résoudre le SP associé à chaque dépôt. Le SP serait alors beaucoup plus rapide à résoudre et même le MP serait plus rapide à résoudre, car on réduit le nombre de colonnes possible et leur diversité. En effet, l'idée est qu'à la sortie d'un nœud trajet, il y ait un nombre limité (entre 2 et 5 typiquement) d'arcs possibles. La réduction du nombre d'arcs se fait principalement à l'aide d'apprentissage supervisé en utilisant des réseaux de neurones pour graphes décrit dans la section suivante (section 4.2.2).

Pour tester cette idée, des expériences peuvent être réalisées en résolvant par BP des ins-

tances sur lesquelles on a diminué le nombre d'arcs et gardé les arcs apparaissant dans la solution optimale. Ces tests préliminaires ont permis de montrer qu'il y a en effet un potentiel important de réduction des temps de calcul. Celui-ci devient moins intéressant sur les grandes instances si l'on ne réduit pas assez le nombre d'arcs. Les méthodes développées visent donc à choisir un sous-ensemble d'arcs le plus réduit possible permettant d'aboutir à une solution de coût presque minimal.

4.2 Méthodes proposées

Dans cette section, les méthodes proposées pour résoudre le MDEVSP sont détaillées. Tout d'abord, un algorithme glouton est présenté dans la section 4.2.1 puis les méthodes basées sur les prédictions d'un GNN sont listées dans la section 4.2.2. Ensuite, la combinaison de l'algorithme glouton et des prédictions est expliquée dans la section 4.2.3. Enfin, des méthodes explorées et moins concluantes sont présentées dans la section 4.2.4. L'ensemble des méthodes proposées est récapitulé dans le tableau 4.1.

Notations

Les différentes méthodes de résolution proposées consistent toutes à sélectionner un sous-ensemble d'arcs parmi l'ensemble $\mathcal{A}$ des arcs du modèle de la section 3.3. Pour faciliter la lecture, on utilise les notations des sous-ensembles de nœuds et des sous-ensembles d'arcs définies dans la liste des notations. En particulier, on note B, les arcs qui regroupent l'ensemble des arcs dépôt-trajet, trajet-dépôt, arcs de recharge et arcs d'attente au dépôt (les arcs noirs dans la figure 3.2).

4.2.1 Heuristique gloutonne

Une première méthode simple de résolution du MDEVSP est d'utiliser une heuristique de construction de solutions inspirée de [25]. L'idée est de construire une solution en ajoutant un à un les trajets à un horaire incomplet au moindre coût en commençant par le trajet qui commence le plus tôt. Quand on ajoute un trajet, on cherche d'abord à l'ajouter directement à la suite d'un trajet, si ce n'est pas possible, on cherche une station de recharge ou un dépôt qui permettrait de partir d'un des derniers trajets desservis par les routes en construction, d'aller à la station de recharge ou au dépôt puis de desservir le trajet voulu. Si ce n'est toujours pas possible, on rajoute un nouveau véhicule. Le fonctionnement de l'algorithme est illustré dans la figure 4.1. L'algorithme détaillé est présenté dans l'algorithme 1. Dans cet algorithme, $\tilde{T} = (\tilde{t}_i)_{i\in\{1,..,n\}}$ est la liste des trajets ordonnés par heure de départ, du

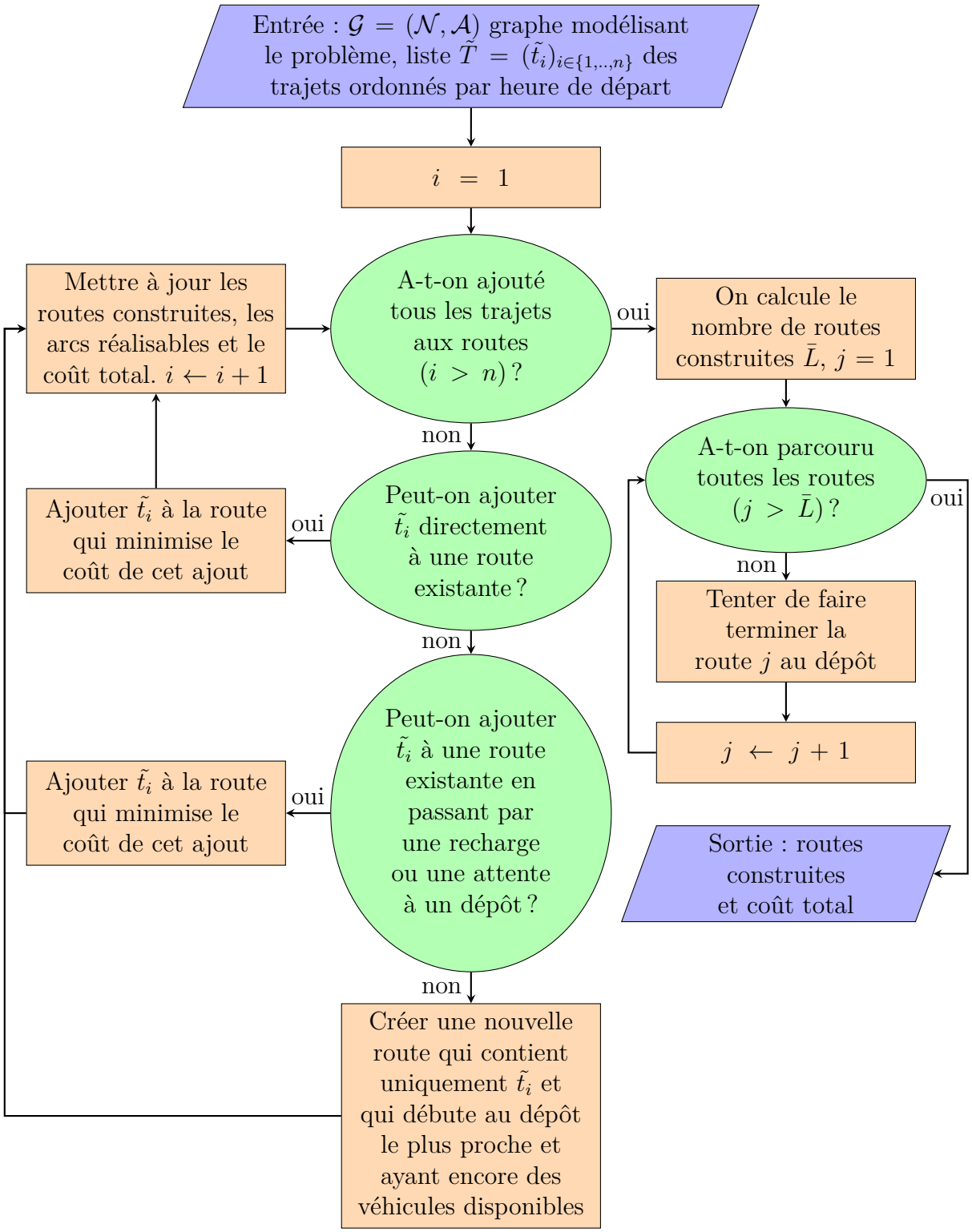

FIGURE 4.1 Fonctionnement de l'algorithme glouton

plus tôt au plus tard. Pour un dépôt d, L^d est la liste des arcs composant les routes en construction du dépôt d et Σ^d est la liste des états de charge des routes du dépôt d. On note y^d le nombre de véhicules utilisés au dépôt d. τ est l'ensemble des derniers trajets des routes en construction. c_{tot} est le coût total de la solution en construction. $\tilde{A}$ est l'ensemble des arcs du graphe modélisant le problème, valides dans la solution en construction. Les sous-algorithmes 3 à 6 utilisés dans l'algorithme principal sont en annexe B. L'algorithme 3 permet de construire les ensembles E_r et E_s pour identifier les stations de recharge et les attentes au dépôt, respectivement, qui peuvent être atteints par une des routes en construction, puis qui permettent de desservir à temps le trajet que l'on cherche à ajouter. L'algorithme 4 permet de choisir parmi les stations de recharge et les dépôts recensés celui qui permet l'ajout du trajet que l'on cherche à ajouter au moindre coût. L'algorithme 5 permet d'essayer de faire terminer les routes construites jusqu'au dépôt de départ. Enfin, l'algorithme 6 permet de mettre à jour les différents ensembles et listes qui permettent de garder en mémoire les routes construites et d'assurer la faisabilité des routes quand on ajoute de nouveaux trajets.

Algorithme 1 : Algorithme glouton

Données : $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ graphe modélisant le problème, $\tilde{T} = (\tilde{t_i})_{i \in \{1,..,n\}}$

début

Pour chaque dépôt d, initialiser L^d à la liste vide et Σ^d à la liste vide;

Initialiser $y^d = 0$;

Initialiser $\tau = \emptyset$;

Initialiser $c_{tot} = 0$;

Initialiser $\tilde{A}$ à $\mathcal{A}$;

pour $i = 1$ *à* n *;* `/* Ajouter les trajets un à un aux routes en construction */`

faire

$E_t = \left\{ t_j | t_j \in \tau, (t_j, \tilde{t_i}) \in \tilde{A} \right\}$;

si $E_t \neq \emptyset$; `/* Essayer d'ajouter le trajet à la suite d'un autre */`

alors

$t_{j_0} = \arg\min \left\{ c_{(t_j, \tilde{t_i})} | t_j \in E_t \right\}$; `/* Minimiser le coût */`

Mettre à jour $(L^d)_{d \in D}$, $(\Sigma^d)_{d \in D}$, c_{tot}, $\tilde{A}$, τ avec l'arc $(t_{j_0}, \tilde{t_i})$ (alg. 6)

sinon

Construire les ensembles E_r et E_s (alg. 3) ; `/* Essayer de passer par un dépôt ou une station de recharge */`

si $E_r \bigcup E_s \neq \emptyset$ **alors**

Trouver les arcs qui permettent de le faire au moindre coût (alg. 4);

Mettre à jour $(L^d)_{d \in D}$, $(\Sigma^d)_{d \in D}$, c_{tot}, $\tilde{A}$, τ avec les arcs qui conviennent (alg. 6);

sinon

$d_0 = \arg\min \left\{ c_{(o^d, \tilde{t_i})} | d \in D, y^d < v^d \right\}$; `/* Rajouter un véhicule au moindre coût */`

$y^{d_0} \leftarrow y^{d_0} + 1$;

Mettre à jour $(L^d)_{d \in D}$, $(\Sigma^d)_{d \in D}$, c_{tot}, $\tilde{A}$, τ avec l'arc $(o^{d_0}, \tilde{t_i})$ (alg. 6);

$\bar{L} = \sum_{d \in D} y^d$; `/* Nombre de routes construites */`

pour $j = 1$ *à* $\bar{L}$ **faire**

Tenter de faire terminer la route j au dépôt (alg. 5);

retourner $(L^d)_{d \in D}$, c_{tot}

Il peut arriver que la solution construite par l'algorithme ne soit pas réalisable pour trois raisons. La première raison est qu'on ne tient pas compte du nombre de bornes de recharge

disponibles à une station de recharge dans l'algorithme. La deuxième raison est qu'il peut arriver qu'après avoir ajouté un trajet, l'état de charge de la batterie soit insuffisant pour se déplacer vers n'importe quel autre point et que l'on ne puisse pas finir la route à la fin de l'algorithme. La troisième raison est que l'on peut avoir utilisé tous les véhicules disponibles sans avoir desservi tous les trajets. En pratique, ce dernier problème ne survient pas si l'on considère un nombre de véhicules assez grand.

Deux solutions sont alors utilisées pour résoudre ces problèmes. Une première méthode pour réduire le nombre de routes irréalisables dans la solution, à cause de la capacité de la batterie, est de fixer un seuil $\Delta\sigma$ sur l'état de charge et d'empêcher d'ajouter un trajet à une route si jamais cela impliquerait que l'état de charge passe sous le seuil. Cette solution s'implémente facilement dans l'algorithme 6 au moment où l'on retire les arcs irréalisables en modifiant artificiellement la valeur de la charge maximale σ_{max} par $\sigma_{max} - \Delta\sigma$. La deuxième méthode consiste à récupérer la liste des arcs utilisés $(L^d)_{d\in D}$ par la solution construite par l'algorithme, ajouter à cet ensemble tous les arcs B. On résout ensuite le problème sur ce sous-ensemble d'arcs par BP. Si le nombre de véhicules maximal n'est pas trop restrictif, l'algorithme BP parviendra à trouver une solution réalisable. En particulier, il prendra en compte la capacité des stations de recharge, ce que cette heuristique ne fait pas.

Cette heuristique, avec les deux améliorations proposées, a l'avantage d'être rapide et de fournir des solutions réalisables à un coût relativement faible, mais qui ne sont pas optimales. On note `initH` cette méthode de résolution du EVSP.

Ajout d'aléatoire dans l'heuristique

En s'inspirant de [44], on peut espérer obtenir de meilleures solutions si on ajoute un peu d'aléatoire dans l'heuristique, puis si l'on génère s solutions qui vont être différentes. On peut ensuite récupérer tous les arcs utilisés dans les différentes solutions et résoudre le problème par BP sur ce sous-ensemble d'arcs auxquels on aura ajouté les arcs B.

Pour ajouter de l'aléatoire dans l'heuristique, on peut fixer un paramètre m ; à chaque fois que l'on doit choisir un arc ou un ensemble d'arcs, au lieu de prendre l'arc ou l'ensemble d'arcs de coût minimum, on choisit l'arc ou l'ensemble d'arcs aléatoirement parmi les m arcs ou ensemble d'arcs de coût minimum en utilisant une distribution uniforme.

Cette variante permet d'obtenir de meilleures solutions comme on le verra dans le chapitre 5 mais est plus coûteuse en temps de calcul. On note `initSsMm` cette méthode de résolution du EVSP où s et m sont les valeurs choisies des paramètres.

4.2.2 Méthodes basées sur les prédictions d'un réseau de neurones pour graphes

Une solution plus poussée consiste à identifier un sous-ensemble d'arcs pouvant apparaitre dans la solution optimale et à résoudre par GC le MDEVSP sur un sous-ensemble d'arcs. Ce sous-ensemble d'arcs peut être choisi à partir de prédictions d'un modèle d'apprentissage basé sur un GNN. En effet, le GNN est une architecture neuronale qui a fait ses preuves dans l'apprentissage automatique sur des graphes. Plusieurs modèles de GNN ont été testés, celui qui est présenté ici est celui qui a donné les meilleurs résultats.

Modèle d'apprentissage

Comme sera détaillé en section 5.3.1, l'apprentissage est supervisé sur des instances préalablement résolues par BP. On détaille d'abord les caractéristiques d'entrée utilisées puis le modèle utilisé.

Les caractéristiques d'entrée utilisées pour les nœuds sont :

- `nbTrajet` : paramètre identique pour une même instance qui donne le nombre de trajets dans cette instance divisé par 1000
- 6 paramètres binaires donnant le type de nœud : `o, k, t, w, r, s`
- `ts, te` : l'heure de départ et l'heure de fin du nœud normalisées entre 0 et 1
- `duree` : la durée du nœud normalisée entre 0 et 1
- 2 paramètres binaires par nœud du réseau routier. Le premier vaut 1 si le départ du nœud (un trajet par exemple) du graphe est à ce nœud du réseau. Le deuxième correspond à l'arrivée du nœud du graphe.
- `nbDep10` : pour un nœud trajet, donne le nombre de trajets qui partent dans les 10 minutes suivant le départ de ce trajet du même point de départ
- `nbDepId10` : pour un nœud trajet, donne le nombre de trajets qui partent dans les 10 minutes suivant le départ de ce trajet du même point de départ pour le même point d'arrivée
- `nbFin10` : pour un nœud trajet, donne le nombre de trajets qui arrivent dans les 10 minutes précédant l'arrivée de ce trajet au même point d'arrivée
- `nbFinId10` : pour un nœud trajet, donne le nombre de trajets qui arrivent dans les 10 minutes précédant l'arrivée de ce trajet du même point de départ au même point d'arrivée

Les caractéristiques d'entrée utilisées pour les arcs sont :

- `c` : le coût de l'arc normalisé entre 0 et 1 (sans prendre en compte le coût fixe d'utilisation d'un véhicule)
- `r1` : la consommation d'énergie sur cet arc normalisée entre 0 et 1. Pour les arcs de

recharge, celle-ci vaut 0 même si, en pratique, elle est négative et dépendante de l'état de charge du véhicule.
- `td` : le temps de déplacement entre la fin du nœud origine de l'arc et le début du nœud destination de l'arc normalisé entre 0 et 1
- `tw` : le temps d'attente (payant) entre la fin du nœud origine de l'arc et le début du nœud destination de l'arc normalisé entre 0 et 1
- `delta` : la durée entre la fin du nœud origine de l'arc et le début du nœud destination de l'arc normalisée entre 0 et 1. Dans le cas d'un arc entre 2 trajets, elle correspond à la somme des temps précédents.
- `rg` : dans le cas d'un arc entre deux trajets, en regardant à partir du trajet origine, le rang du temps de départ du trajet destination
- `rgId` : dans le cas d'un arc entre deux trajets, en regardant à partir du trajet origine les arcs qui vont au même nœud dans le réseau routier, le rang du temps de départ du trajet destination
- `rgC` : dans le cas d'un arc entre deux trajets, en regardant à partir du trajet origine le rang du coût de cet arc

Pour la première couche du réseau de neurones, h_i^0 et e_{ij}^0 sont des projections linéaires des caractéristiques d'entrée du réseau de neurones h_i^{-1} et e_{ij}^{-1} via deux matrices qui peuvent être apprises M_n et M_e respectivement de taille $d_h \times d_n$ et $d_h \times d_e$ où d_n est la dimension des caractéristiques d'entrée des nœuds et d_e est la dimension des caractéristiques d'entrée des arcs.

Le modèle choisi est un réseau de neurones pour graphes avec L couches de neurones. Chaque couche l correspond à l'actualisation des caractéristiques h_j^l d'un nœud j en fonction de ses voisins N_j et des caractéristiques e_{ij}^l des arcs ij suivant la formule suivante [78] :

$$h_j^{l+1} = h_j^l + ReLU\left(Norm\left(U^l h_j^l + \max_{i \in N_j}\left(\sigma\left(e_{ij}^l\right) V^l h_i^l\right)\right)\right),\ \forall l \in \{0, ..., L-1\}, \forall j \in \mathcal{N} \quad (4.6)$$

$$e_{ij}^{l+1} = e_{ij}^l + ReLU\left(Norm\left(A^l e_{ij}^l + B^l h_i^l + C^l h_j^l\right)\right),\ \forall l \in \{0, ..., L-1\}, \forall (i,j) \in \mathcal{A} \quad (4.7)$$

h_i^l et e_{ij}^l sont des vecteurs de dimension d_h. U^l, V^l, A^l, B^l et C^l sont des matrices que l'on peut apprendre de taille $d_h \times d_h$. $Norm$ correspond à une normalisation par lot des vecteurs. σ est la fonction sigmoïde. $ReLU$ est la fonction d'activation telle que

$$ReLU(x) = \begin{cases} x & \text{si } x \geq 0 \\ 0 & \text{sinon.} \end{cases} \quad (4.8)$$

La sortie du réseau de neurones est un score p_{ij} calculé pour chaque arc ij à partir des formules suivantes [78] :

$$p_{ij} = \sigma\left(W_2^T\left(ReLU\left(W_1\left([h_G, h_i^L, h_j^L]\right)\right)\right)\right), \text{ où } h_G = \frac{1}{n}\sum_{i=0}^{n} h_i^L,\ \forall (i,j) \in \mathcal{A} \tag{4.9}$$

[] représente la concaténation des vecteurs et W_1 et W_2 sont deux matrices que l'on peut apprendre de taille $d_h \times 3d_h$ et $1 \times d_h$, respectivement. h_G est la moyenne des caractéristiques des nœuds sur tout le lot.

L'apprentissage de ce réseau de neurones est fait par lots sur un ensemble de graphes complets comme décrit dans la section 3. Cependant, la fonction de perte est calculée uniquement sur les arcs IT, TS et TR (tous les arcs rouges, bleus et verts dans la figure 3.2). La fonction de perte est l'entropie croisée binaire avec un poids différencié pour les classes positives et les classes négatives.

Le fonctionnement global du GNN est récapitulé dans l'algorithme 2.

Algorithme 2 : Fonctionnement du GNN

Données : $(h_i^{-1})_{(i\in\mathcal{N})} \in \mathbb{R}^{d_n}$ et $(e_{ij}^{-1})_{((i,j)\in\mathcal{A})} \in \mathbb{R}^{d_e}$ les caractéristiques d'entrée,
$M_n \in \mathbb{R}^{d_h \times d_n}$, $M_e \in \mathbb{R}^{d_h \times d_e}$, $(U^l)_{(l\in\{1,\dots,L\})} \in \mathbb{R}^{d_h \times d_h}$,
$(V^l)_{(l\in\{1,\dots,L\})} \in \mathbb{R}^{d_h \times d_h}$, $(A^l)_{(l\in\{1,\dots,L\})} \in \mathbb{R}^{d_h \times d_h}$, $(B^l)_{(l\in\{1,\dots,L\})} \in \mathbb{R}^{d_h \times d_h}$,
$(C^l)_{(l\in\{1,\dots,L\})} \in \mathbb{R}^{d_h \times d_h}$, $W_1 \in \mathbb{R}^{d_h \times 3d_h}$, $W_2 \in \mathbb{R}^{d_h}$ les matrices constituant le GNN

début

- **pour tous** $j \in \mathcal{N}$ **faire** $h_j^0 = M_n \times h_j^{-1}$;
- **pour tous** $(i,j) \in \mathcal{A}$ **faire** $e_{ij}^0 = M_e \times e_{ij}^{-1}$;
- **pour tous** $l = 0$ *à* $L-1$ **faire**
  - **pour tous** $j \in \mathcal{N}$ **faire**
    - $h_j^{l+1} = h_j^l + ReLU\left(Norm\left(U^l h_j^l + \max_{i\in N_j}\left(\sigma\left(e_{ij}^l\right) V^l h_i^l\right)\right)\right)$;
  - **pour tous** $(i,j) \in \mathcal{A}$ **faire**
    - $e_{ij}^{l+1} = e_{ij}^l + ReLU\left(Norm\left(A^l e_{ij}^l + B^l h_i^l + C^l h_j^l\right)\right)$
- $h_G = \frac{1}{n}\sum_{i=0}^{n} h_i^L$;
- **pour tous** $(i,j) \in \mathcal{A}$ **faire**
  - $p_{ij} = \sigma\left(W_2^T\left(ReLU\left(W_1\left([h_G, h_i^L, h_j^L]\right)\right)\right)\right)$

retourner $(p_{ij})_{((i,j)\in\mathcal{A})}$

Choix d'arcs pour la résolution

À partir des scores prédits par l'apprentissage sur les arcs, on cherche à sélectionner un sous-ensemble d'arcs sur lequel utiliser la GC dans le but d'avoir une résolution beaucoup plus rapide et quasiment optimale si les arcs sont bien choisis.

Les arcs qui partent ou arrivent au dépôt sont toujours gardés ainsi que les arcs qui composent la recharge et l'attente au dépôt (tous les arcs noirs dans la figure 3.2). On garde tous les arcs liés au dépôt dans le but d'avoir toujours une solution réalisable (si le nombre de véhicules disponibles est assez grand [1]). On garde les arcs de recharge et d'attente au dépôt, car il y en a relativement peu et c'est plus simple de les garder pour être sûr de ne pas trop restreindre les possibilités de recharge.

Pour le reste des arcs (entre les trajets ou entre les trajets et la recharge ou entre les trajets et l'attente au dépôt), il est possible d'utiliser différentes manières de les sélectionner à partir des scores :

- À partir d'un nœud, en fonction du rang des scores des arcs sortants, on peut choisir de garder tous les arcs dont le rang est inférieur à 1, 2 ou 3 par exemple.
- À partir d'un nœud, en fonction du rang des scores des arcs entrants, on peut choisir de garder tous les arcs dont le rang est inférieur à 1, 2 ou 3 par exemple.
- En fonction des scores directement, on peut choisir tous les arcs dont les scores sont supérieurs à 0,1 par exemple.
- À partir d'un nœud, on peut aussi choisir de garder une proportion d'arcs entrants et une portion d'arcs sortants, 20% par exemple, en gardant les arcs avec les scores les plus élevés.

Dans notre étude, les différentes méthodes de sélection d'arcs sont tous les arcs dont le rang entrant ou sortant est inférieur ou égal soit à 1, soit à 2, soit à 3. On appelle ces méthodes respectivement `rg1`, `rg2` et `rg3`. Si l'on choisit de garder 20% des meilleurs arcs entrants et sortants d'un nœud en gardant au moins un arc entrant et un arc sortant, on nomme cette méthode `f20`. Si l'on choisit de garder tous les arcs dont le rang entrant ou sortant est inférieur ou égal à 1 plus tous les arcs dont le score est supérieur ou égal soit à $0,1$, soit à $0,2$, on appelle ces méthodes respectivement `s01` et `s02`.

4.2.3 Méthodes basées sur les prédictions et l'heuristique

Il est possible de combiner l'algorithme heuristique et les prédictions de deux manières différentes.

1. Dans le cas contraire, on ne peut être surs de trouver une solution réalisable.

La première manière de combiner les deux méthodes est d'ajouter tous les arcs utilisés par l'algorithme heuristique `initH` à ceux sélectionnés à partir des prédictions. On a alors un nouveau sous-ensemble d'arcs que l'on peut utiliser pour résoudre le problème par GC. Si l'on combine les arcs choisis par `rg2` et ceux de l'heuristique, on nomme cette méthode `rg2_initH`.

La deuxième manière est d'utiliser les scores donnés par les réseaux de neurones directement dans l'heuristique pour générer s solutions aléatoires. De la même manière que lorsque l'on ajoute de l'aléatoire dans l'heuristique, au moment où l'on doit choisir des arcs, au lieu de prendre l'arc de coût minimum, on choisit un arc à partir des scores que l'on normalise de sorte que la somme des probabilités vaille 1. On appelle cette méthode `initPredSs`.

Tableau 4.1 Méthodes proposées

Nom de la méthode	Description rapide	Section
`initH`	Algorithme glouton	4.2.1
`initSsMm`	Algorithme glouton avec aléatoire	4.2.1
`rgX`	Sélection d'arcs à chaque nœud dont le rang du score prédit par le GNN est inférieur à `X`	4.2.2
`fX`	Sélection d'arcs à chaque nœud des `X`% meilleurs scores prédits par le GNN	4.2.2
`sX`	Sélection d'arcs à chaque nœud des arcs dont le score prédit par le GNN est supérieur à `X`	4.2.2
`pred_initH`	Combinaison d'une méthode `pred` basée sur les prédictions (par exemple `rg2`) et de la méthode `initH`	4.2.3
`initPredSs`	Utilisation des scores prédits par le GNN dans l'algorithme glouton avec aléatoire	4.2.3
`rg2Rand`	Méthode `rg2` avec des scores aléatoires	5.3.2
`rg2IT`	`rg2` sur les arcs IT uniquement	5.3.2
`initRandS3`	Utilisation de scores aléatoires dans l'algorithme glouton avec aléatoire	5.3.2

4.2.4 Autres méthodes explorées

Dans cette section, on liste d'autres pistes explorées dont les résultats ne sont pas reportés dans la section 5.

Autres réseaux de neurones explorés

Au cours du travail présenté, d'autres réseaux de neurones ont été testés. Une première manière de modifier le réseau de neurones utilisé est d'utiliser la fonction softmax en sortie

du réseau de neurones plutôt que la fonction sigmoïde de sorte que la somme des scores des arcs sortants d'un nœud soit égale à 1. Une deuxième manière de modifier le réseau de neurones est d'utiliser K couches de neurones à partir de la couche $ReLU(W_1([h_G, h_i^L, h_j^L]))$ avec d_h unités cachées et $ReLu$ comme fonction d'activation (au lieu d'une seule couche). La sortie est calculée avec la fonction sigmoïde ou la fonction softmax. Une troisième possibilité est d'ajouter de l'attention dans les couches du réseau de neurones [69].

Ces réseaux de neurones donnent des résultats semblables ou moins bons. Les instances résolues à l'aide de ces réseaux ne sont donc pas présentées.

Il est aussi possible de changer l'étiquette d'apprentissage. En effet, au lieu d'apprendre si un arc apparait dans la solution optimale ou non, on peut choisir d'apprendre si un arc apparait dans une solution d'une relaxation linéaire dans l'arbre de branchement ou seulement dans la relaxation linéaire du problème (premier nœud de l'arbre). Apprendre les arcs qui apparaissent dans la relaxation linéaire donne des résultats légèrement plus mauvais.

L'apprentissage peut aussi être fait en considérant qu'une instance correspond seulement à un nœud trajet et soit les arcs qui sortent du nœud associé soit les arcs qui rentrent dans ce nœud-là. De même, la fonction de perte n'est calculée que sur les arcs reliant deux trajets ou un trajet et la recharge ou l'attente au dépôt. Ce modèle donne clairement des résultats moins bons.

Il est également possible de modifier légèrement le graphe sur lequel l'apprentissage est fait pour garder un nombre limité d'arcs entrants et sortants de chaque nœud en gardant les arcs de meilleur coût. L'apprentissage est légèrement plus rapide, mais les solutions issues de ce modèle sont un peu moins bonnes.

Importance de l'optimalité des solutions sur lesquelles on apprend

Puisque l'on apprend sur des instances résolues de manière heuristique, on peut se demander si le fait que les solutions du jeu de données d'apprentissage ne soient pas optimales détériore la qualité de l'apprentissage. Une expérience a permis de voir que l'apprentissage n'était pas dégradé et était même plus favorable si, lorsque l'on teste, la résolution est aussi heuristique.

Itérations sur le choix d'arcs

Il est possible que les arcs choisis par les méthodes présentées précédemment aient laissé de côté des arcs qui apparaissent dans la solution optimale et qu'il serait bénéfique d'ajouter au sous-ensemble d'arcs. Pour identifier ces arcs-là, on utilise les valeurs duales de la relaxation linéaire calculées dans la GC pour calculer des coûts réduits approximatifs de tous les arcs de

manière similaire à [79]. La manière de calculer ces coûts réduits approximatifs est détaillée dans l'annexe C. On peut, à partir de ces coûts réduits, choisir d'ajouter un sous-ensemble d'arcs au sous-ensemble d'arcs en fixant des seuils sur la valeur des coûts réduits et éventuellement des scores du réseau de neurones. Cette méthode permet effectivement, à partir d'une première solution telle qu'obtenue par `rg2`, de décroître le coût de la solution et d'aboutir à une solution presque optimale. Cependant, le calcul des coûts réduits approximatifs est assez long et chaque itération avec un sous-ensemble d'arcs donnés est trop longue pour que l'on ait une convergence vers une solution de qualité acceptable en un temps qui est intéressant.

CHAPITRE 5 EXPÉRIENCES

Dans ce chapitre, les différentes expérimentations menées sont présentées et détaillées. Tout d'abord, les instances du MDEVSP utilisées sont présentées dans la section 5.1. Ensuite, les paramètres utilisés pour le BP et l'apprentissage supervisé sont donnés dans la section 5.2. Enfin, la section 5.3 présente les différentes expériences et leurs résultats.

5.1 Présentation des instances

Différentes instances ont été utilisées pour le développement des méthodes proposées. Seulement les instances utilisées pour présenter les résultats de la section 5.3 sont présentées ici.

Tout d'abord, les instances sont générées à partir de réels horaires de la STM (Société de transport de Montréal) comme dans [80]. L'ensemble des lignes et de leurs horaires sont récupérés. On choisit ensuite un sous-ensemble de lignes pour créer un type d'instances. Dans ce travail, deux types d'instances ont été utilisés. Le premier type d'instances utilise le réseau 1, comporte 4 lignes (205, 206, 208, 407), parcourues dans les deux sens, et est illustré dans la figure 5.1. Le réseau 2 comporte 10 lignes différentes (13, 30, 93, 95, 19, 52, 53, 15, 61, 715) et est illustré dans la figure 5.2. Pour chaque type d'instances, on associe 2 dépôts potentiels et 2 stations de recharge possibles. Pour les instances de type 1, des tests ont été réalisés avec un nombre réduit de dépôts et de stations de recharge.

Pour obtenir des instances variées d'une taille souhaitée, on récupère le nombre de trajets par heure sur chaque ligne. Pour créer une instance, on modifie aléatoirement entre -1 et 1 le nombre de trajets par heure sur chaque ligne. On crée ensuite les horaires de chaque ligne en utilisant un paramètre donnant la fréquence maximale de la ligne la plus fréquente à son heure de pointe. Les horaires des autres lignes pour toutes les heures sont déterminées proportionnellement au rapport de leur fréquence à la fréquence maximale. Les horaires pour chaque heure sont créées de manière régulière. On obtient alors pour chaque ligne les différentes heures de départ dans un sens de la ligne et dans l'autre. Cette méthode permet d'avoir beaucoup d'instances avec des horaires différents, une répartition dans le temps des trajets réaliste et un nombre de trajets contrôlable, ce qui est essentiel pour réaliser l'apprentissage automatique.

Pour chaque instance, à partir des horaires créés, on construit le graphe détaillé dans la section 3.3. Pour cela, on utilise les valeurs numériques données dans le tableau 5.2.

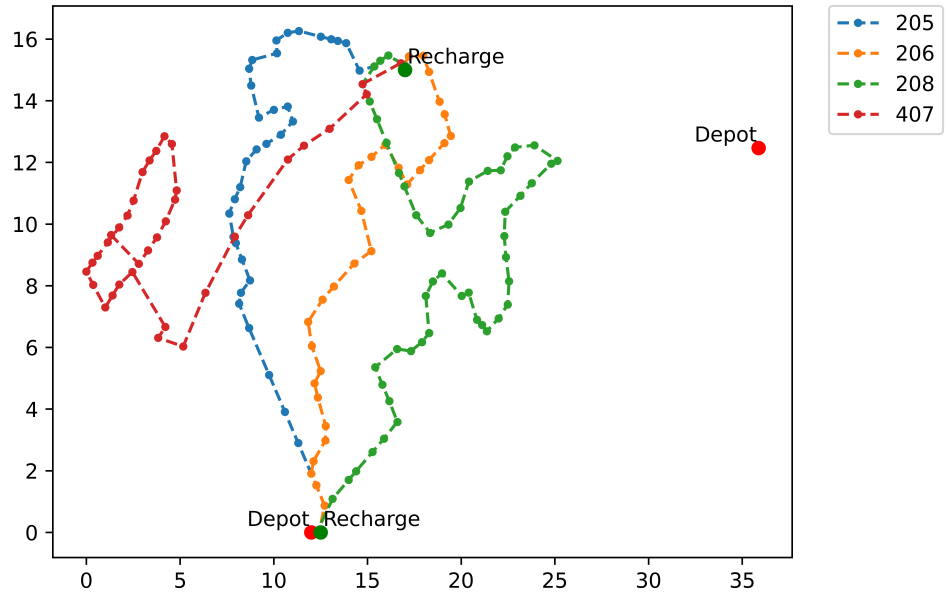

FIGURE 5.1 Réseau 1

On crée ainsi des instances sur 2 réseaux différents, de taille différente [1], avec 1 ou 2 dépôts et 1 ou 2 stations de recharge. Les différentes instances utilisées sont classées dans le tableau 5.1. La première colonne donne le nom des instances, la deuxième le réseau utilisé, la troisième le nombre de dépôts, la quatrième le nombre de stations de recharge, la cinquième la taille des instances avec le nombre de trajets moyen, la sixième la possibilité de retourner attendre au dépôt dans la journée, la septième le nombre de véhicules maximum par dépôt, la huitième le nombre de bornes à chaque station de recharge et la neuvième le nombre d'instances de chaque type généré.

TABLEAU 5.1 Instances utilisées

Nom	Réseau	m	r	n	Retour	v^d	u^h	Instances
A	1	1	1	686,4	Oui	60	4	500
B	1	1	2	686,4	Oui	60	2	500
C	1	2	2	686,4	Oui	30	2	500
D	1	1	1	686,4	Non	60	4	500
E	2	2	2	786,7	Oui	50	2	500
F	2	2	2	2600,4	Oui	200	5	50

De plus, la fonction de recharge f est décrite par un taux de recharge (en Wh par minute)

1. Le temps entre deux trajets pour la ligne à la fréquence maximale à l'heure de pointe est de 3 minutes pour les instances A, B, C et D, de 7 minutes pour les petites instances E et 2 minutes pour les grandes instances F

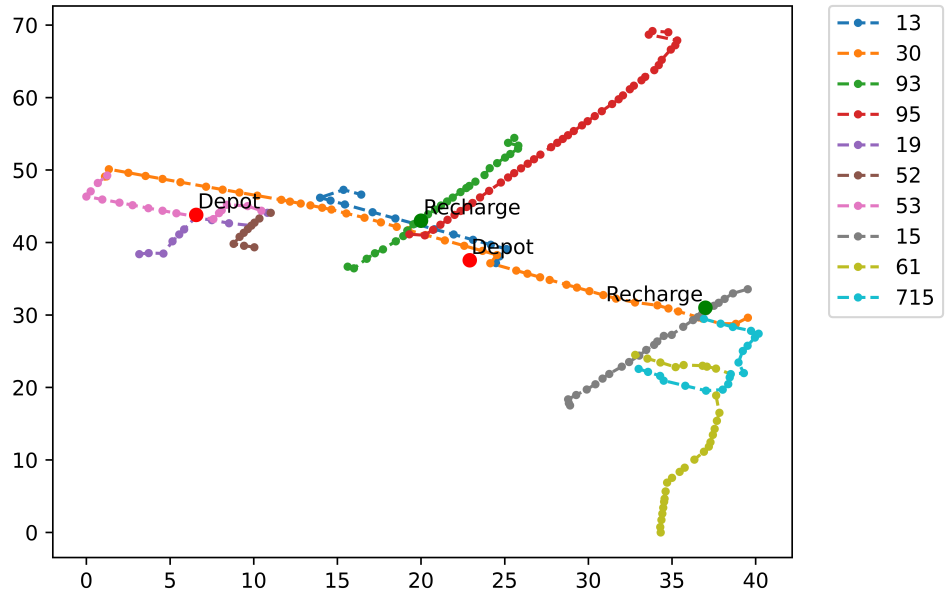

FIGURE 5.2 Réseau 2

de 1539,7 entre 0 et 55 minutes de recharge puis 716,0 entre 55 et 69 minutes de recharge et 240,5 entre 69 et 91 minutes de recharge.

5.2 Implémentation

5.2.1 Résolution par BP

Les instances du MDEVSP décrites précédemment sont résolues avec le logiciel de génération de colonnes Gencol [81]. À chaque itération de GC, on génère beaucoup de colonnes et on en ajoute un nombre conséquent au problème maitre restreint. Différentes techniques sont utilisées pour ajouter des colonnes diversifiées, contrôler le nombre de colonnes que l'on ajoute et nettoyer le problème maitre au besoin. Le problème maitre restreint est résolu par un algorithme barrière. En effet, des tests préliminaires ont montré que cet algorithme fonctionne mieux que l'algorithme du simplexe lorsque les instances sont grandes et que les problèmes sont très dégénérés. Lorsque l'on cherche à brancher à la fin de la résolution d'un noeud par génération de colonnes, on évalue 4 branchements possibles : fixer une colonne, fixer une intertâche, brancher sur le flot d'un arc et enlever les perturbations. Lorsque plusieurs branchements sont possibles, on privilégie la fixation de colonnes puis d'intertâches puis la levée des perturbations et enfin le branchement sur les arcs. Il est possible de fixer plusieurs colonnes en même temps et plusieurs intertâches en même temps.

Avec ces paramètres, le temps de calcul moyen et le coût moyen des solutions pour les différentes instances sont présentés dans le tableau 5.3. La première colonne donne le type d'instances, la deuxième le nombre moyen de trajets dans les instances, la troisième le nombre moyen de nœuds dans le graphe qui modélise le SP, la quatrième le nombre moyen d'arcs dans ce même graphe, la cinquième le temps moyen de résolution, la sixième le coût moyen des solutions trouvées et la septième le nombre moyen de véhicules utilisés. On observe qu'avec un nombre de trajets constant, si l'on augmente le nombre de dépôts et/ou de stations de recharge, les nombres d'arcs et de nœuds augmentent, le temps de résolution augmente et le coût de la solution diminue. Il est intéressant de voir que si l'on retire la possibilité d'aller attendre au dépôt en journée, le temps de résolution augmente énormément et le coût de la solution augmente aussi un peu. Le réseau choisi impacte aussi beaucoup la solution : en effet, les instances C et E ont le même nombre de dépôts et de stations de recharge, les instances ont un peu plus de trajets, pourtant, les instances E ont un nombre de véhicules plus petit. Enfin, lorsque le nombre de trajets augmente beaucoup (instances F), le temps de résolution, le coût de la solution et le nombre de véhicules augmentent aussi beaucoup.

Tableau 5.2 Paramètres utilisés

Paramètre	Valeur
Vitesse moyenne α	18 km/h
Capacité de la batterie σ_{max}	100 kWh [2]
État de charge minimal σ_{min}	0 kWh
Énergie consommée en mouvement e_{mouv}	1050 Wh/km
Énergie consommée à l'arrêt e_{arr}	11000 Wh/h
Période d'échantillonnage δ	15 minutes
Temps maximum entre la fin d'un trajet et le début d'un autre β	45 minutes
Coût fixe d'un véhicule c_f	1000
Coût d'attente par minute à un arrêt c_a	2
Coût de voyage à vide par minute c_v	4
Coût d'attente à la recharge par minute c_r^v	2
Coût fixe d'aller à la recharge c_r^f	30
Coût fixe de retourner attendre au dépôt c_d^f	30

5.2.2 Procédure d'apprentissage

Pour un jeu de données fixé avec un certain type d'instances, on résout toutes les instances par GC comme expliqué précédemment. On procède ensuite à l'apprentissage. Celui-ci est

2. Cette valeur est cohérente avec les valeurs données dans [19]. Cette valeur de batterie permet d'effectuer une journée complète avec 2 recharges de 15 minutes en moyenne.

Tableau 5.3 Résolution par BP des instances

Instance	Trajets	Nœuds	Arcs	Temps (en s)	Coût	Véhicules
A	686,4	953,1	20728,4	465,5	58970,4	41,9
B	686,4	1129,6	22446,0	474,7	58473,7	41,9
C	686,4	1219,8	25276,7	486,2	52870,0	41,7
D	686,4	864,9	19273,7	1262,3	63647,3	41,9
E	786,7	1339,4	23681,4	646,4	54681,7	29,0
F	2600,4	3156,7	17781,9	19300,3	141910,2	85,9

implémenté en Python en utilisant la bibliothèque DGL [82]. On divise ce jeu de données en 70% d'ensemble d'apprentissage, 15% d'ensemble de validation et 15% d'ensemble de test. Chaque ensemble est ensuite découpé en lots de 5 instances, chaque lot a alors plusieurs milliers d'arcs. Pour chaque instance, on extrait les caractéristiques des différents nœuds et arcs. En particulier, on étiquette chaque arc en fonction de s'il fait partie de la solution optimale trouvée ou non. On crée le réseau de neurones avec $L = 25$ couches de neurones de dimension $d_h = 128$.

On commence alors l'apprentissage. On effectue au maximum 200 itérations d'apprentissage (*epoch*). Lors d'une itération d'apprentissage, on passe au travers de tous les lots de l'ensemble d'apprentissage. Pour chaque lot, on calcule la sortie du réseau de neurones puis on évalue la fonction de perte sur cette sortie par rapport aux étiquettes d'apprentissage. La fonction de perte utilisée est la fonction entropie binaire croisée avec un poids de 0,1 pour la classe négative et 1 pour la classe positive. Les poids du réseau de neurones sont alors ajustés à l'aide de l'optimiseur Adam [83] de sorte à minimiser la fonction de perte. Le taux d'apprentissage est fixé à 0,0001. A la fin de chaque itération d'apprentissage, la fonction de perte est évaluée sur l'ensemble de validation. Si la valeur de la fonction de perte n'a pas diminué depuis plus de 5 itérations d'apprentissage, on stoppe l'apprentissage prématurément. L'ensemble des hyperparamètres du réseau de neurones et de l'apprentissage est récapitulé dans le tableau 5.4.

Un autre indicateur est calculé sur l'ensemble d'apprentissage et l'ensemble de validation pour suivre l'apprentissage : l'aire sous la courbe ROC (*Receiver Operating Characteristic*). Cette courbe est tracée pour évaluer la qualité d'un prédicteur. Supposons qu'on ait des étiquettes binaires $\{l_1, ..., l_K\}$ et des prédictions $\{s_1, ..., s_K\}$ de valeur entre 0 et 1. Pour chaque valeur $s \in \{0, s_1, ..., s_K, 1\}$, on place un point d'abscisse $(TP(s), (FP(s))$ dans un repère allant de $(0, 0)$ à $(1, 1)$. $TP(s)$ est le taux de vrais positifs si l'on utilise la valeur s comme seuil pour déterminer si les prédictions sont positives ou négatives : $TP(s) = \frac{Card(\{i|s_i \geq s, p_i=1\})}{Card(\{i|p_i=1\})}$. $FP(s)$ est le taux de faux positifs avec le seuil s : $FP(s) = \frac{Card(\{i|s_i \geq s, p_i=0\})}{Card(\{i|p_i=0\})}$. Plus l'aire sous la

courbe ROC est proche de 1, plus le modèle est bon et, plus il est proche de 0,5, plus le modèle se rapproche d'un prédicteur aléatoire. Une fois le modèle appris, on peut l'utiliser pour faire des prédictions comme décrit dans la section suivante.

Tableau 5.4 Récapitulatif des hyperparamètres du GNN

Paramètre	Valeur numérique
Nombre de couches L	25
Dimension des neurones associés à chaque nœud et chaque arc d_h	128
Dimension des caractéristiques d'entrée des nœuds d_n	Entre 5 et 56
Dimension des caractéristiques d'entrée des arcs d_e	Entre 3 et 8
Taille des lots	5
Itérations d'apprentissage	200
Taux d'apprentissage	0,0001
Nombre d'itérations sans amélioration au bout desquelles on arrête l'apprentissage	5

5.3 Résultats

Dans cette section, les résultats des différents apprentissages (section 5.3.1) et des méthodes envisagées pour résoudre le MDEVSP (section 5.3.2) sont présentés.

5.3.1 Apprentissage

Pour tous les types d'instance, on réalise l'apprentissage d'un réseau de neurones avec les caractéristiques décrites dans la section 4.2.2. On appelle ces modèles `A` à `F`. En plus, pour les instances de type F, on apprend un modèle `Transfert` sur ces instances en utilisant le modèle pré-entrainé sur les instances de type E, cette technique relève de l'apprentissage par transfert. On apprend aussi un réseau de neurones `ReduitA` sur les instances de type A en utilisant seulement les caractéristiques suivantes : `duree`, `nbDep10`, `nbDepId10`, `nbFin10`, `nbFinId10`, `rg`, `rgId`, `rgC`. Ces caractéristiques ne dépendent pas du réseau à partir duquel sont formées les instances, contrairement aux caractéristiques utilisées dans les modèles précédents. Les mêmes ensembles d'entrainement, de validation et de test sont utilisés pour un même type d'instances.

Les courbes de fonction de perte et d'aire sous la courbe ROC de l'ensemble de validation pour chaque modèle sont présentées dans les figures 5.3 et 5.4.

Tout d'abord, les courbes d'apprentissage des modèles `A`, `B`, `C` sont assez semblables. Le mo-

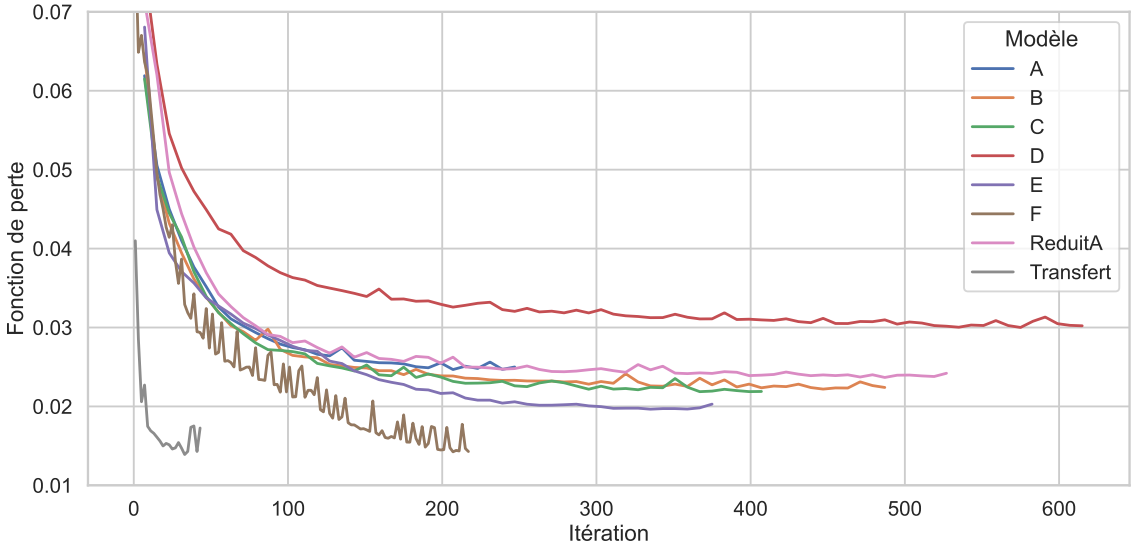

FIGURE 5.3 Fonction de perte sur l'ensemble de validation pour les différents modèles

dèle `D` a de moins bonnes performances que les modèles précédents. Cela s'explique par le fait que les instances D sont plus difficiles à résoudre sans le retour au dépôt [3]. Ensuite, la courbe du modèle `E` suit une trajectoire similaire, mais l'apprentissage est plus rapide et plus stable avec des meilleurs résultats. Cela est dû au nombre de véhicules plus petit dans les solutions optimales. L'apprentissage du modèle `F` est très chaotique à cause du faible nombre d'instances. On voit que le modèle `Transfert` permet un apprentissage beaucoup plus stable, plus court et avec des résultats similaires. Enfin, le modèle `ReduitA` est légèrement moins performant, mais l'apprentissage est plus stable et permet de faire plus d'itérations d'apprentissage. Pour un apprentissage stable avec de bons résultats, il faut donc privilégier des instances faciles à résoudre (on peut les identifier par le faible rapport entre le temps de résolution par BP et le nombre de trajets) et avoir beaucoup d'instances. Chacun de ces modèles est par la suite utilisé pour faire des prédictions sur les instances du type correspondant à l'ensemble d'entraînement comme décrit dans la section qui suit.

5.3.2 Optimisation en réduisant le nombre d'arcs

Dans cette section, on teste les différentes méthodes présentées dans la section 4.2 sur les instances introduites dans la section 5.1 en utilisant les modèles de la section 5.3.1. La

3. En effet, la possibilité de retourner au dépôt permet d'avoir des routes plus flexibles et la contrainte des β minutes entre 2 trajets est moins contraignante

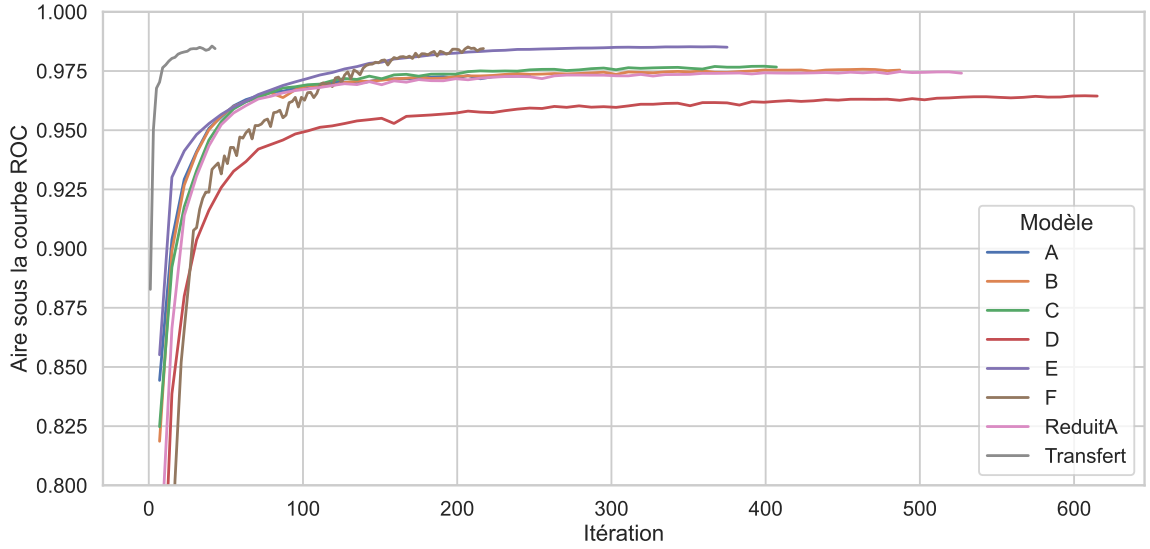

FIGURE 5.4 Aire sous la courbe ROC sur l'ensemble de validation pour les différents modèles

première série d'expériences évalue à la fois les différentes méthodes de sélection d'arcs à partir des prédictions, la sélection d'arcs à partir de l'algorithme glouton et de ses variantes et la combinaison des deux types de sélections. C'est la combinaison de la méthode `rg2` de sélection d'arcs à partir des prédictions et de l'algorithme glouton qui performe le mieux. Les trois autres séries d'expérience évaluent la robustesse de cette méthode : sa sensibilité au type d'instances, la généralisation du GNN utilisé à des instances plus grandes et la généralisation du GNN utilisé à d'autres types d'instances.

Comparaison des méthodes de sélection d'arcs à partir des prédictions et l'algorithme glouton

La première expérience est réalisée uniquement sur les instances A avec le modèle `A`. Elle consiste à tester les différentes manières de réduire le nombre d'arcs et voir quelle influence cela a sur le temps de calcul et la qualité de la solution. Pour cela, on calcule le ratio de temps et l'écart à la solution avec tous les arcs utilisés. Si l'on note t_0 et z_0 le temps de résolution et le coût de la solution, respectivement, d'une instance résolue par BP et avec tous les arcs dans le modèle, et si l'on note t_1 et z_1 le temps de résolution et le coût de la solution d'une instance résolue par une méthode 1 qui combine une réduction du nombre d'arcs dans le modèle, suivie d'une résolution par BP, le ratio t est $\frac{t_1}{t_0}$ et l'écart z est $\frac{z_1 - z_0}{z_0} - 1$.

Les résultats des différentes méthodes se basant sur les prédictions du modèle `A` sont rapportés

dans le tableau 5.5. Les résultats présentés sont moyennés sur l'ensemble de test. La méthode 0 correspond à la résolution en considérant tous les arcs. La colonne t donne le temps de calcul de l'algorithme BP, la colonne Ratio t donne le gain en temps de calcul, la colonne Écart z donne l'augmentation du coût de la solution, la colonne v donne le nombre de véhicules utilisés dans la solution et la colonne Arcs donne le nombre d'arcs dans le graphe sur lequel on résout le problème. Le temps de calcul ne prend pas en compte le temps de prédiction à partir des modèles, celui-ci est de l'ordre de 5 secondes et peut donc être négligé. Pour rappel, les descriptions des différentes méthodes sont rassemblées dans le tableau 4.1.

Tableau 5.5 Réduction du nombre d'arcs des instances A à partir des scores du modèle `A`

Méthode	t (s)	Ratio t (%)	Écart z (%)	v	Arcs
`0`	456,03	100,00	0,00	41,84	20718
`rg1`	9,97	2,27	161,81	124,89	2821
`rg2`	85,74	19,85	12,51	48,57	3623
`rg3`	188,10	42,85	1,62	42,53	4382
`s01`	233,97	52,73	0,46	42,03	5079
`s02`	199,70	45,24	1,76	42,77	4483
`f20`	117,57	26,79	26,99	48,80	7915
`rg2Rand`	48,07	11,39	111,55	75,05	3778
`rg2IT`	225,59	51,43	1,47	42,23	4886

Les différentes manières de choisir les arcs à partir des prédictions permettent différents compromis entre la réduction du temps de calcul et l'augmentation du coût de la solution. On cherche à choisir une seule méthode de sélection d'arcs à partir des prédictions pour les tests suivants. Au sein d'un même type de méthodes de sélection d'arcs (`rgX` ou `sX`), la valeur `X` permet de choisir un compromis entre gain de temps et perte de qualité de la solution. On cherche dans un premier temps à choisir un type de méthodes, puis on choisira la valeur `X` dans un deuxième temps. On observe que la méthode `rg3` domine la méthode `s02` et que la méthode `rg2` domine la méthode `f20`. On préfère donc, dans la suite, les méthodes de sélection d'arcs à partir du rang de type `rgX`. Ensuite, il nous faut choisir une valeur `X` pour le type de méthodes `rgX`. on choisit de se concentrer sur la méthode `rg2` qui permet d'avoir une réduction du temps de calcul important (20%) tout en ayant une solution pas trop mauvaise (écart de $12,5\%$). Par ailleurs, on cherche à évaluer la qualité du modèle de prédiction. On sélectionne alors des arcs par la méthode `rg2` avec des prédictions aléatoires, on note cette méthode `rg2Rand`. En comparant les méthodes `rg2Rand` et `rg2`, on constate que le modèle `A` permet bien un apprentissage. Une autre manière de choisir les arcs est de garder tous les arcs allant à la recharge ou à l'attente au dépôt et de réduire uniquement le nombre d'arcs entre deux trajets avec la méthode `rg2`. On note cette méthode `rg2IT`. On

parvient alors à résoudre les instances en 51% du temps de calcul et l'on a un écart moyen de 1,5%. Cette méthode offre un nouveau compromis entre temps de calcul et qualité de la solution. On ne l'utilise plus dans la suite, car le temps de calcul est assez élevé, on préfèrera la méthode `rg2` simple pour tous les choix d'arcs à partir de prédictions.

Il est aussi possible de résoudre à l'aide de l'algorithme glouton les instances A (voir section 4.2.1) et de combiner les solutions de l'algorithme glouton aux prédictions du modèle `A`. Les différentes méthodes et leurs résultats sont présentés dans le tableau 5.6. Cette fois-ci, le temps de calcul de l'algorithme glouton est pris en compte dans toutes les méthodes faisant appel à l'algorithme glouton. Ce temps-là est assez court, mais peut varier du simple au double suivant si l'on crée plusieurs solutions ou non. La méthode `initRandS3` est la méthode `initPredS3` avec des scores complétement aléatoires. En comparant les méthodes `initH` et `initSsM3`, on observe que les différentes versions de l'algorithme glouton apportent aussi différents compromis entre temps de résolution et qualité de la solution. Parmi toutes ces variantes, on préfère les méthodes `initH` et `initS1M3` qui sont les plus rapides. Entre ces deux méthodes, on préfère la méthode `initH` qui permet d'avoir le plus petit écart z. De plus, on observe que l'ajout des prédictions directement dans l'algorithme glouton permet d'améliorer ce dernier comme on peut le voir en comparant les méthodes `initPredS3` et `initS3M3`. Cela montre une fois de plus que le GNN est capable d'identifier des arcs intéressants. De plus, en comparant les méthodes `initRandS3` et `initPredS3`, on arrive à la même conclusion.

Tableau 5.6 Résolution à l'aide de l'algorithme glouton des instances A

Méthode	t (s)	Ratio t (%)	Écart z (%)	v	Arcs
`0`	456,03	100,00	0,00	41,84	20718
`initH`	11,67	2,78	27,57	45,09	2563
`initS1M3`	11,42	2,68	30,08	45,27	2556
`initS3M3`	84,10	19,12	8,28	42,85	3383
`initS5M3`	128,65	29,15	4,27	42,26	3864
`initPredS3`	60,58	13,50	5,20	42,38	3230
`initRandS3`	104,84	28,82	11,96	42,57	3369
`rg1_initS3M3`	113,06	25,84	2,29	42,15	3772
`rg2_initH`	126,79	29,24	1,37	42,12	3852

Enfin, l'union des arcs sélectionnés par des prédictions et des arcs sélectionnés par l'algorithme glouton dans les méthodes `rg1_initS3M3` et `rg2_initH` donne des résultats très intéressants avec un écart à la solution avec tous les arcs d'environ 2% et une réduction du temps de calcul de 70%. Il est très important de constater que la combinaison des prédictions et de l'algorithme glouton donne de meilleurs résultats que les méthodes séparées. On explique cela par le fait que les prédictions permettent d'identifier des arcs intéressants localement tandis

que l'algorithme glouton identifie des arcs intéressants pour la construction de routes entières. Ainsi, une partie des arcs identifiés est commune aux méthodes basées sur les prédictions et à l'algorithme glouton, mais c'est l'union des arcs identifiés dans seulement une des méthodes qui permet d'avoir un écart aussi faible. Par exemple, on voit que l'ajout des arcs de `initH` à ceux de `rg2` augmente seulement le nombre d'arcs moyen de 3623 à 3852 soit 200 arcs de plus, mais ces 200 arcs de plus permettent de faire passer l'écart de 12,5% à 1,4% alors que le ratio du temps de calcul n'augmente que de 20% à 29%. Dans la suite, on choisit donc de s'intéresser à la combinaison d'une méthode basée sur les prédictions et de l'algorithme glouton. On choisit pour cela de rapporter les résultats uniquement pour la méthode `rg2_initH` et les méthodes qui la composent, à savoir `rg2` et `initH`.

Résultats sur les autres types d'instances de la combinaison de `rg2` et de l'algorithme glouton

Les résultats obtenus de la deuxième expérience sont présentés dans le tableau 5.7. Elle vise à évaluer la sensibilité de la méthode `rg2_initH` et des méthodes qui la composent au type d'instances. Le type d'instances peut changer à cause du réseau utilisé, du nombre de dépôts, du nombre de stations de recharge ou de la possibilité de retourner attendre au dépôt. La première chose que l'on peut remarquer, c'est que plus le nombre d'arcs parmi lesquels il faut choisir augmente (lorsque le nombre de stations de recharge augmente par exemple), moins les résultats de la méthode `rg2` sont bons. En effet, entre les instances A et B, le ratio de temps de calcul pour la méthode `rg2` est presque le même, alors que l'écart z augmente de 4 points. De plus, pour les instances D où le retour au dépôt n'est pas autorisé, l'apprentissage avait de moins bons résultats et cela se reflète dans les résultats de la méthode `rg2`. Par ailleurs, les résultats peuvent varier énormément si l'on change le type de réseau. En effet, les instances E ont plus de trajets, mais exigent beaucoup moins de véhicules, et l'apprentissage et les méthodes qui en découlent sont plus performantes. Cela est aussi dû au fait que la méthode `rg2` sélectionne plus d'arcs et qui a pour conséquence d'augmenter le temps de calcul. À l'inverse, lorsque le nombre de bornes de recharge augmente, l'algorithme glouton performe mieux. Cela se voit en comparant la méthode `initH` sur les instances A et B (écart de 28% et 19% respectivement). Par contre, il fonctionne beaucoup moins bien sur les instances E qui ont des horaires plus denses (écart de 58%). Quant à la méthode `rg2_initH`, elle permet pour toutes les instances une nette amélioration du coût de la solution par rapport à l'utilisation séparée des méthodes qui la composent. La combinaison donne de bons résultats dans tous les cas, même si ceux-ci varient, avec un écart z de quelques pourcents et un ratio de temps de calcul d'au plus 35%. On voit que la combinaison des méthodes `rg2` et `initH` fonctionne moins bien quand l'une des deux méthodes donne des résultats légèrement moins bons ou un

long temps de calcul, comme c'est le cas pour les instances E.

Tableau 5.7 Sensibilité des méthodes au type d'instances

Instance	Méthode	t (s)	Ratio t (%)	Écart z (%)	v	Arcs
A	`0`	456,03	100,00	0,00	41,84	20718
A	`rg2`	85,74	19,85	12,51	48,57	3623
A	`initH`	11,67	2,78	27,57	45,09	2563
A	`rg2_initH`	126,79	29,24	1,37	42,12	3852
B	`0`	463,11	100,00	0,00	42,00	22224
B	`rg2`	76,23	17,51	16,24	50,05	3909
B	`initH`	12,42	2,89	18,60	43,53	2901
B	`rg2_initH`	122,60	27,96	2,06	42,44	4137
C	`0`	505,41	100,00	0,00	41,65	25609
C	`rg2`	75,55	15,21	15,04	48,76	5436
C	`initH`	11,96	2,50	18,71	42,69	4380
C	`rg2_initH`	126,49	25,56	2,13	42,11	5643
D	`0`	1479,73	100,00	0,00	41,85	19726
D	`rg2`	56,87	5,17	21,19	57,20	3601
D	`initH`	10,85	1,04	26,30	45,07	2479
D	`rg2_initH`	149,25	12,49	1,58	42,29	3866
E	`0`	693,08	100,00	0,00	29,05	23904
E	`rg2`	191,82	27,85	5,40	31,31	6108
E	`initH`	27,62	4,04	57,82	41,77	4957
E	`rg2_initH`	241,11	35,21	3,67	30,35	6366

Généralisation de GNN à des instances plus grandes

La troisième expérience, dont les résultats sont rapportés dans le tableau 5.8, vise à généraliser la méthode `rg2_initH` pour des instances de plus grande taille. Premièrement, pour les grandes instances F, l'algorithme `initH` basé uniquement sur la sélection d'arcs par l'algorithme glouton ne performe pas très bien (écart moyen de 66%). Ensuite, les méthodes `rg2` et `rg2_initH` utilisant le modèle `F` permettent de réduire le temps de calcul tout en contrôlant l'écart, mais leurs résultats sont moins bons que sur les instances plus petites (écarts moyens de 22% et 5%, respectivement). Cela est dû au fait que les instances sont difficiles à résoudre et que l'apprentissage sur un nombre réduit d'instances n'est pas suffisant, comme on pouvait le voir sur les courbes d'apprentissage très chaotiques. Une idée est alors d'utiliser le modèle `E` qui a été entrainé sur plus d'instances plus petites. On voit qu'utiliser directement le modèle `E` donne de très mauvais résultats : écart moyen de plus de 1000% pour `rg2` et de 41% pour `rg2_initH`. Une seconde idée est alors d'utiliser le modèle `Transfert` qui permet

d'utiliser le modèle `E` déjà entrainé et de le peaufiner pour les instances F. Ce modèle avait de bons résultats lors de l'apprentissage. Ce modèle a effectivement des résultats légèrement meilleurs au niveau de la qualité de la solution que le modèle `F` quand l'on regarde les méthodes `rg2_initH` (`F`) et `rg2_initH` (`Transfert`) : l'écart moyen passe de 5,0% à 4,2% avec un ratio du temps de calcul qui augmente de 49% à 51%. On pourrait penser que l'on arrive juste à un nouveau compromis entre temps et qualité de la solution en utilisant le modèle `Transfert` plutôt que le modèle `F`, mais d'autres tests ont montré que si l'on sélectionne plus d'arcs à partir du modèle `F`, on aboutit effectivement à une meilleure solution, mais on perd beaucoup en réduction du temps de calcul. En conclusion, la méthode `rg2_initH` fonctionne toujours sur les grandes instances mais le gain de temps de calcul est moins important et on peut obtenir de meilleurs résultats en utilisant l'apprentissage par transfert.

Tableau 5.8 Généralisation des modèles pour les grandes instances F

Méthode (modèle)	t (s)	Ratio t (%)	Écart z (%)	v	Arcs
`0`	15476,12	100,00	0,00	84,5	167520
`initH`	706,95	4,75	66,04	126,62	13775
`rg2` (`F`)	5111,94	34,23	21,56	114,50	17078
`rg2_initH` (`F`)	7303,28	48,55	5,07	88,25	17864
`rg2` (`E`)	26,82	0,18	1055,63	1441,88	21509
`rg2_initH` (`E`)	3990,74	26,66	41,14	108,38	23846
`rg2` (`Transfert`)	5755,67	37,93	17,99	110,00	17209
`rg2_initH` (`Transfert`)	7684,21	51,01	4,24	87,12	17948

Généralisation de GNN à d'autres types d'instances

La quatrième et dernière expérience a pour but d'évaluer la possibilité d'utiliser un modèle appris sur un type d'instances à un autre type d'instances et, par conséquent, de généraliser la méthode `rg2_initH` développée sur un type d'instances à un autre type d'instances. En plus de la taille des instances qui peut changer le type d'instances comme exploré dans l'expérience précédente, on peut aussi modifier le nombre de dépôts ou même le réseau par exemple. Pour évaluer la possibilité d'utiliser un GNN appris sur un type d'instances à un autre type d'instances, on utilise le modèle `ReduitA` entrainé sur les instances A et qui peut être appliqué à n'importe quel type d'instances (puisqu'il ne dépend pas du réseau). On compare alors pour chaque type d'instances le résultat de la méthode `rg2_initH` en utilisant le modèle associé à l'instance et la méthode `rg2_initH` avec le modèle `ReduitA`. Les résultats sont recensés dans le tableau 5.9. On remarque que le modèle `ReduitA` donne de bons résultats pour les instances A, B, C et D même si les résultats sont moins bons qu'en

utilisant les modèles `A`, `B`, `C` et `D`. Par contre, lorsque l'on change de réseau avec les instances E et F, les résultats sont beaucoup moins bons. Cela s'explique par le fait que les modèles GNN sont très dépendants du réseau routier sous-jacent même si les paramètres d'entrée ne dépendent pas du réseau et par le fait que les modèles GNN ont du mal à généraliser un apprentissage sur un graphe à un graphe très différent. On ne peut donc pas transposer directement un modèle appris sur un réseau à un autre réseau. On pourrait par contre faire de l'apprentissage par transfert comme fait pour les grandes instances et espérer obtenir de meilleurs résultats.

Tableau 5.9 Généralisation de `ReduitA` à d'autres types d'instances

Instance	Modèle	t (s)	Ratio t (%)	Écart z (%)	v	Arcs
A	`A`	126,79	29,24	1,37	42,12	3852
A	`ReduitA`	125,06	28,69	1,62	42,28	3838
B	`B`	122,60	27,96	2,06	42,44	4137
B	`ReduitA`	124,28	28,21	2,22	42,29	4230
C	`C`	126,49	25,56	2,13	42,11	5643
C	`ReduitA`	120,89	24,41	4,09	42,03	5859
D	`D`	149,25	12,49	1,58	42,29	3866
D	`ReduitA`	133,23	10,97	2,66	42,48	3745
E	`E`	241,11	35,21	3,67	30,35	6366
E	`ReduitA`	187,93	27,38	35,06	36,84	7627
F	`F`	7303,28	48,55	5,07	88,25	17864
F	`ReduitA`	3672,91	24,32	48,44	113,62	22477

CHAPITRE 6 CONCLUSION ET RECOMMANDATIONS

Le travail présenté ici propose une nouvelle méthode de résolution du EVSP. Tout d'abord, le EVSP a été formulé comme un PLNE de partitionnement d'ensemble où les variables sont associées à des routes dans un graphe. La modélisation prend en compte plusieurs dépôts, plusieurs stations de recharge, le nombre de chargeurs disponibles à une station de recharge, un seul type de véhicule électrique, un temps de recharge linéaire par morceaux et la recharge partielle.

Ensuite, une méthode de résolution basée sur un BP heuristique et une réduction de la taille du modèle ont été proposés. La réduction de la taille est faite par l'élimination d'arcs dans le graphe avant la résolution par BP. Deux méthodes sont combinées pour conserver les arcs : un algorithme glouton qui trouve rapidement une solution réalisable et un algorithme qui sélectionne un petit nombre d'arcs par nœud dans le graphe à partir d'une prédiction d'un GNN entrainé sur des instances résolues par GC. Certains arcs, comme les arcs entre les dépôts et les trajets, sont systématiquement conservés pour assurer la faisabilité du problème. Tous les autres arcs sont éliminés.

Les tests ont été réalisés sur des instances générées aléatoirement à partir de données réelles de la ville de Montréal. Les tests ont montré que les deux méthodes d'élimination d'arcs séparément ne performent pas très bien, mais donne des bons résultats quand on les combine. Pour des petites instances (700 trajets), cette méthode permet de réduire le temps de résolution d'au moins 65% en limitant l'écart à la valeur optimale à 3,7%. Pour des instances de 2500 trajets, cette méthode permet de réduire le temps de calcul de 49% tout en limitant l'écart à la valeur optimale à 4,2%. Des tests ont aussi été faits pour évaluer la possibilité d'utiliser un GNN entrainé sur un autre réseau routier ou sur des instances plus petites. Les résultats montrent que l'application directe de ce GNN ne fonctionne pas très bien, mais qu'il est possible de faire de l'apprentissage par transfert de ce GNN vers les instances qui nous intéressent, pour limiter le temps d'apprentissage et la taille des jeux de données d'entrainement.

La méthode proposée permet donc d'accélérer le temps de résolution par GC du problème du EVSP grâce à une heuristique gloutonne et à l'apprentissage supervisé d'un GNN tout en conservant une bonne qualité de la solution.

La plus grande limitation de la solution proposée est qu'il faut résoudre beaucoup d'instances pour entrainer le GNN avant de pouvoir appliquer la méthode. En plus du nombre d'instances à résoudre, il faut résoudre des instances assez proches du problème que l'on souhaite résoudre

avec au moins un nombre de trajets similaire et la même topographie du réseau des lignes de bus. Cette limitation est atténuée par la possibilité de faire de l'apprentissage par transfert : il est possible d'entrainer un GNN sur un type d'instances puis de finaliser l'apprentissage sur un nombre réduit d'instances qui nous intéressent.

Le travail présenté pourrait être amélioré en cherchant comment mieux généraliser les GNN entrainés sur un type d'instances à un autre type d'instances. Une manière de mieux généraliser pourrait être d'entrainer le modèle sur des instances aux tailles et aux types variés. On pourrait étudier l'influence de la taille des jeux de données d'entrainement sur la qualité de l'apprentissage pour essayer de réduire la taille. Une autre manière d'avoir des instances plus facilement pourrait être de les résoudre avec un algorithme plus rapide. On pourrait aussi chercher d'autres modèles d'apprentissage que le GNN proposé ou améliorer l'entrainement ou les paramètres d'entrée pour améliorer les performances. Par exemple, on pourrait explorer les modèles d'apprentissage par renforcement.

Il est aussi possible de chercher comment l'apprentissage automatique pourrait imiter l'heuristique pour se passer de l'heuristique. On pourrait explorer d'autres méthodes de réduction du nombre de variables à combiner avec celle présentée ici, par exemple construire des chaines de trajets à desservir les uns à la suite des autres. On pourrait également implémenter des algorithmes de recherche locale à partir d'une solution déduite des prédictions qui permettrait de remplacer la partie gloutonne de la méthode par un algorithme plus intelligent et donc plus efficace.

La méthode proposée pourrait aussi être validée sur des instances encore plus grandes avec plus de trajets ou plus de dépôts ou plus de stations de recharge. On pourrait aussi complexifier la modélisation du EVSP en ajoutant la possibilité de choisir où se situe les bornes de recharge ou en optimisant en même temps les horaires des chauffeurs de bus.

RÉFÉRENCES

[1] Green & healthy streets declaration. [En ligne]. Disponible : https://www.c40.org/declarations/green-healthy-streets-declaration/

[2] C. H. Häll *et al.*, "Adjustments of public transit operations planning process for the use of electric buses," *Journal of Intelligent Transportation Systems*, vol. 23, n°. 3, p. 216–230, 2019.

[3] G. Desaulniers et M. D. Hickman, "Public transit," dans *Handbooks in Operations Research and Management Science*, C. Barnhart et G. Laporte, édit. Elsevier, 2007, vol. 14, p. 69–127.

[4] A. A. Bertossi, P. Carraresi et G. Gallo, "On some matching problems arising in vehicle scheduling models," *Networks*, vol. 17, n°. 3, p. 271–281, 1987.

[5] O. Sassi et A. Oulamara, "Electric vehicle scheduling and optimal charging problem : complexity, exact and heuristic approaches," *International Journal of Production Research*, vol. 55, n°. 2, p. 519–535, 2017.

[6] A. Zhang *et al.*, "The effect of nonlinear charging function and line change constraints on electric bus scheduling," *Promet-Traffic&Transportation*, vol. 33, n°. 4, p. 527–538, 2021.

[7] A.-S. Pepin *et al.*, "A comparison of five heuristics for the multiple depot vehicle scheduling problem," *Journal of Scheduling*, vol. 12, p. 17–30, 2009.

[8] L. Zhang, S. Wang et X. Qu, "Optimal electric bus fleet scheduling considering battery degradation and non-linear charging profile," *Transportation Research Part E : Logistics and Transportation Review*, vol. 154, p. 102445, 2021.

[9] T. Liu et A. A. Ceder, "Battery-electric transit vehicle scheduling with optimal number of stationary chargers," *Transportation Research Part C : Emerging Technologies*, vol. 114, p. 118–139, 2020.

[10] S. S. Perumal *et al.*, "Solution approaches for integrated vehicle and crew scheduling with electric buses," *Computers & Operations Research*, vol. 132, p. 105268, 2021.

[11] S. Bunte et N. Kliewer, "An overview on vehicle scheduling models," *Public Transport*, vol. 1, n°. 4, p. 299–317, 2009.

[12] C. C. Ribeiro et F. Soumis, "A column generation approach to the multiple-depot vehicle scheduling problem," *Operations Research*, vol. 42, n°. 1, p. 41–52, 1994.

[13] G. Carpaneto *et al.*, "A branch and bound algorithm for the multiple depot vehicle scheduling problem," *Networks*, vol. 19, n°. 5, p. 531–548, 1989.

[14] A. Löbel, "Vehicle scheduling in public transit and lagrangean pricing," *Management Science*, vol. 44, n°. 12-part-1, p. 1637–1649, 1998.

[15] A. Hadjar, O. Marcotte et F. Soumis, "A branch-and-cut algorithm for the multiple depot vehicle scheduling problem," *Operations Research*, vol. 54, n°. 1, p. 130–149, 2006.

[16] N. Kliewer, T. Mellouli et L. Suhl, "A time–space network based exact optimization model for multi-depot bus scheduling," *European Journal of Operational Research*, vol. 175, n°. 3, p. 1616–1627, 2006.

[17] V. Gintner, N. Kliewer et L. Suhl, "Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice," *OR Spectrum*, vol. 27, p. 507–523, 2005.

[18] P. C. Guedes et D. Borenstein, "Column generation based heuristic framework for the multiple-depot vehicle type scheduling problem," *Computers & Industrial Engineering*, vol. 90, p. 361–370, 2015.

[19] N. Dirks *et al.*, "A concise guide on the integration of battery electric buses into urban bus networks," *arXiv preprint arXiv :2104.10752*, 2021.

[20] N. Dirks, M. Schiffer et G. Walther, "On the integration of battery electric buses into urban bus networks," *Transportation Research Part C : Emerging Technologies*, vol. 139, p. 103628, 2022.

[21] M. Rinaldi *et al.*, "Mixed-fleet single-terminal bus scheduling problem : Modelling, solution scheme and potential applications," *Omega*, vol. 96, p. 102070, 2020.

[22] A. Haghani et M. Banihashemi, "Heuristic approaches for solving large-scale bus transit vehicle scheduling problem with route time constraints," *Transportation Research Part A : Policy and Practice*, vol. 36, n°. 4, p. 309–333, 2002.

[23] Z. Chao et C. Xiaohong, "Optimizing battery electric bus transit vehicle scheduling with battery exchanging : Model and case study," *Procedia-Social and Behavioral Sciences*, vol. 96, p. 2725–2736, 2013.

[24] J.-Q. Li, "Transit bus scheduling with limited energy," *Transportation Science*, vol. 48, n°. 4, p. 521–539, 2014.

[25] J. D. Adler et P. B. Mirchandani, "The vehicle scheduling problem for fleets with alternative-fuel vehicles," *Transportation Science*, vol. 51, n°. 2, p. 441–456, 2017.

[26] J. Reuer, N. Kliewer et L. Wolbeck, "The electric vehicle scheduling problem : A study on time-space network based and heuristic solution," dans *Proceedings of the Conference on Advanced Systems in Public Transport (CASPT)*, 2015.

[27] T. Paul et H. Yamada, "Operation and charging scheduling of electric buses in a city bus route network," dans *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, p. 2780–2786.

[28] M. Wen *et al.*, "An adaptive large neighborhood search heuristic for the electric vehicle scheduling problem," *Computers & Operations Research*, vol. 76, p. 73–83, 2016.

[29] A. Montoya *et al.*, "The electric vehicle routing problem with nonlinear charging function," *Transportation Research Part B : Methodological*, vol. 103, p. 87–110, 2017.

[30] M. E. van Kooten Niekerk, J. Van den Akker et J. Hoogeveen, "Scheduling electric vehicles," *Public Transport*, vol. 9, p. 155–176, 2017.

[31] N. Olsen et N. Kliewer, "Scheduling electric buses in public transport : Modeling of the charging process and analysis of assumptions," *Logistics Research*, vol. 13, n°. 1, p. 4, 2020.

[32] M. Jiang, Y. Zhang et Y. Zhang, "Multi-depot electric bus scheduling considering operational constraint and partial charging : A case study in shenzhen, china," *Sustainability*, vol. 14, n°. 1, p. 255, 2021.

[33] A. Zhang *et al.*, "Mixed electric bus fleet scheduling problem with partial mixed-route and partial recharging," *International Journal of Sustainable Transportation*, vol. 16, n°. 1, p. 73–83, 2022.

[34] M. Janovec et M. Kohání, "Exact approach to the electric bus fleet scheduling," *Transportation Research Procedia*, vol. 40, p. 1380–1387, 2019.

[35] N. Olsen *et al.*, "A literature overview on scheduling electric vehicles in public transport and location planning of the charging infrastructure," *Free University Berlin, School of Business & Economics Discussion Papers*, n°. 2020/16, 2020.

[36] S. S. Perumal, R. M. Lusby et J. Larsen, "Electric bus planning & scheduling : A review of related problems and methodologies," *European Journal of Operational Research*, vol. 301, n°. 2, p. 395–413, 2022.

[37] Y. Zhou, Q. Meng et G. P. Ong, "Electric bus charging scheduling for a single public transport route considering nonlinear charging profile and battery degradation effect," *Transportation Research Part B : Methodological*, vol. 159, p. 49–75, 2022.

[38] N. Olsen, N. Kliewer et L. Wolbeck, "A study on flow decomposition methods for scheduling of electric buses in public transport based on aggregated time–space network models," *Central European Journal of Operations Research*, vol. 30, p. 883—919, 2022.

[39] M. Jiang et Y. Zhang, "A branch-and-price algorithm for large-scale multidepot electric bus scheduling," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[40] M. de Vos et R. N. van Lieshout, "Electric vehicle scheduling with capacitated charging stations and partial charging," *arXiv preprint arXiv :2207.13734*, 2022.

[41] Ş. Yıldırım et B. Yıldız, "Electric bus fleet composition and scheduling," *Transportation Research Part C : Emerging Technologies*, vol. 129, p. 103197, 2021.

[42] M. Alvo, G. Angulo et M. A. Klapp, "An exact solution approach for an electric bus dispatch problem," *Transportation Research Part E : Logistics and Transportation Review*, vol. 156, p. 102528, 2021.

[43] C. Wang, C. Guo et X. Zuo, "Solving multi-depot electric vehicle scheduling problem by column generation and genetic algorithm," *Applied Soft Computing*, vol. 112, p. 107774, 2021.

[44] R. Jovanovic *et al.*, "A grasp approach for solving large-scale electric bus scheduling problems," *Energies*, vol. 14, n°. 20, p. 6610, 2021.

[45] G.-J. Zhou *et al.*, "Collaborative optimization of vehicle and charging scheduling for a bus fleet mixed with electric and traditional buses," *IEEE Access*, vol. 8, p. 8056–8072, 2020.

[46] M. Rogge *et al.*, "Electric bus fleet size and mix problem with optimization of charging infrastructure," *Applied Energy*, vol. 211, p. 282–295, 2018.

[47] X. Li *et al.*, "Joint optimization of regular charging electric bus transit network schedule and stationary charger deployment considering partial charging policy and time-of-use electricity prices," *Journal of Advanced Transportation*, vol. 2020, p. 1–16, 2020.

[48] N. Olsen et N. Kliewer, "Location planning of charging stations for electric buses in public transport considering vehicle scheduling : A variable neighborhood search based approach," *Applied Sciences*, vol. 12, n°. 8, p. 3855, 2022.

[49] J. Xiong *et al.*, "Mixed optimization on vehicle scheduling and recharge scheduling of plug-in electric buses with consideration of partial recharge," *Journal of Transportation Engineering, Part A : Systems*, vol. 148, n°. 2, p. 04021111, 2022.

[50] J. Wang *et al.*, "Collaborative optimization of vehicle and crew scheduling for a mixed fleet with electric and conventional buses," *Sustainability*, vol. 14, n°. 6, p. 3627, 2022.

[51] L. Li, H. K. Lo et F. Xiao, "Mixed bus fleet scheduling under range and refueling constraints," *Transportation Research Part C : Emerging Technologies*, vol. 104, p. 443–462, 2019.

[52] J. Teng, T. Chen et W. D. Fan, "Integrated approach to vehicle scheduling and bus timetabling for an electric bus line," *Journal of Transportation Engineering, Part A : Systems*, vol. 146, n°. 2, p. 04019073, 2020.

[53] Y. Liu *et al.*, “A two-stage approach with a departure time based solution representation for electric bus vehicle scheduling,” *IEEE Access*, vol. 10, p. 112 799–112 811, 2022.

[54] M. Duan *et al.*, “Reforming mixed operation schedule for electric buses and traditional fuel buses by an optimal framework,” *IET Intelligent Transport Systems*, vol. 15, nº. 10, p. 1287–1303, 2021.

[55] W. Wu *et al.*, “The multi-depot electric vehicle scheduling problem with power grid characteristics,” *Transportation Research Part B : Methodological*, vol. 155, p. 322–347, 2022.

[56] J. Ji, Y. Bie et L. Wang, “Optimal electric bus fleet scheduling for a route with charging facility sharing,” *Transportation Research Part C : Emerging Technologies*, vol. 147, p. 104010, 2023.

[57] Y. Bengio, A. Lodi et A. Prouvost, “Machine learning for combinatorial optimization : a methodological tour d’horizon,” *European Journal of Operational Research*, vol. 290, nº. 2, p. 405–421, 2021.

[58] M. Lombardi et M. Milano, “Boosting combinatorial problem modeling with machine learning,” *arXiv preprint arXiv :1807.05517*, 2018.

[59] A. Parmentier, “Learning to approximate industrial problems by operations research classic problems,” *Operations Research*, vol. 70, nº. 1, p. 606–623, 2022.

[60] M. Morabit, G. Desaulniers et A. Lodi, “Machine-learning–based column selection for column generation,” *Transportation Science*, vol. 55, nº. 4, p. 815–831, 2021.

[61] ——, “Machine-learning–based arc selection for constrained shortest path problems in column generation,” *INFORMS Journal on Optimization*, 2022.

[62] A. Lodi et G. Zarpellon, “On learning and branching : a survey,” *Top*, vol. 25, p. 207–236, 2017.

[63] A. T. Dauer et B. d. A. Prata, “Variable fixing heuristics for solving multiple depot vehicle scheduling problem with heterogeneous fleet and time windows,” *Optimization Letters*, vol. 15, nº. 1, p. 153–170, 2021.

[64] N. Vesselinova *et al.*, “Learning combinatorial optimization on graphs : A survey with applications to networking,” *IEEE Access*, vol. 8, p. 120 388–120 416, 2020.

[65] Q. Wang et C. Tang, “Deep reinforcement learning for transportation network combinatorial optimization : A survey,” *Knowledge-Based Systems*, vol. 233, p. 107526, 2021.

[66] Q. Cappart *et al.*, “Combinatorial optimization and reasoning with graph neural networks,” *arXiv preprint arXiv :2102.09544*, 2021.

[67] E. Khalil *et al.*, "Learning combinatorial optimization algorithms over graphs," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

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

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

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

[71] 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, 2020.

[72] J. Zhou *et al.*, "Graph neural networks : A review of methods and applications," *AI open*, vol. 1, p. 57–81, 2020.

[73] J. Desrosiers *et al.*, "Time constrained routing and scheduling," dans *Handbooks in Operations Research and Management Science*, M. Ball *et al.*, édit. Elsevier, 1995, vol. 8, p. 35–139.

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

[75] G. B. Dantzig et P. Wolfe, "Decomposition principle for linear programs," *Operations Research*, vol. 8, n°. 1, p. 101–111, 1960.

[76] S. Irnich et G. Desaulniers, "Shortest path problems with resource contraints," dans *Column generation*, G. Desaulniers, J. Desrosiers et M. M. Solomon, édit. Springer Science & Business Media, 2006, vol. 5.

[77] L. Desfontaines et G. Desaulniers, "Multiple depot vehicle scheduling with controlled trip shifting," *Transportation Research Part B : Methodological*, vol. 113, p. 34–53, 2018.

[78] C. K. Joshi *et al.*, "Learning the travelling salesperson problem requires rethinking generalization," *Constraints*, vol. 27, n°. 1-2, p. 70–98, 2022.

[79] S. Irnich *et al.*, "Path-reduced costs for eliminating arcs in routing and scheduling," *INFORMS Journal on Computing*, vol. 22, n°. 2, p. 297–313, 2010.

[80] J. Brasseur, "Accélération d'une méthode d'agrégation dynamique de contraintes par apprentissage automatique pour le problème de construction d'horaires de conducteurs d'autobus," Mémoire de maîtrise, Ecole Polytechnique, Montréal (Canada), 2022.

[81] J. Desrosiers, "GENCOL : Une équipe et un logiciel d'optimisation," dans *Combinatorial Optimization in Practice*, A. Bui et I. Tseveendorj, édit., 2010, vol. 8, n°. 2, p. 61–96.

[82] M. Wang *et al.*, "Deep graph library : A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv :1909.01315*, 2019.

[83] D. P. Kingma et J. Ba, "Adam : A method for stochastic optimization," *arXiv preprint arXiv :1412.6980*, 2014.

ANNEXE A ALGORITHME D'ÉTIQUETAGE

Cette annexe vise à détailler le fonctionnement de l'algorithme d'étiquetage utilisé pour trouver les plus courts chemins avec contraintes dans le SP de la GC. Tout d'abord, pour la modélisation choisie du MDEVSP, on détaille les différentes ressources : leurs fenêtres de validité sur les nœuds et leur valeur sur les arcs dans les tableaux A.1 et A.2. σ correspond à la ressource déchargement de la batterie, r^1 à la première ressource pour les stations de recharge et r^2 à la deuxième ressource pour les stations de recharge. Les valeurs de consommation de la ressource déchargement de la batterie ont déjà été explicités pour les arcs dans la section 3.3.

Tableau A.1 Fenêtres de ressources sur les nœuds

Nœud	σ	$[r^{1,min}, r^{1,max}]$	$[r^{2,min}, r^{2,max}]$
o^d	[0,0]	[0,0]	[0,0]
k^d, s_p^d, w_p^h	[0,σ_{max}]	[0,1]	[0,1]
t_i	[0,σ_{max}]	[0,0]	[0,0]
r_p^h	[0,σ_{max}]	[1,1]	[0,1]

Tableau A.2 Consommation de ressources sur les arcs

Arc	r^1	r^2
(o^d, t_i), (t_i, k^d), (t_i, t_j), (t_i, s_p^d), (s_p^d, t_i), (s_p^d, s_{p+1}^d), (s_N^d, k^d), (w_p^h, w_{p+1}^h), (r_p^h, r_{p+1}^h), (r_p^h, w_p^h), (w_N^h, k^d)	0	0
(t_i, w_p^h)	0	1
(w_p^h, t_i)	-1	0
(w_p^h, r_{p+1}^h)	1	-1

Nous détaillons maintenant certaines parties de l'algorithme d'étiquetage. Il consiste à partir du noeud origine du graphe et à, itérativement, prolonger des chemins des derniers noeuds atteints vers les noeuds voisins aux derniers noeuds atteints. A chaque noeud, on garde en mémoire des étiquettes spécifiques aux chemins en construction. Le prolongement de ces étiquettes aux noeuds voisins est décrite dans ce qui suit. Considérons un arc $a = (i, j)$. Supposons que l'on soit en train de construire un chemin de l'origine du graphe et qui finit à i avec un coût réduit C_i et que l'on souhaite le prolonger à j en passant pas l'arc a. En notant $L_i = (C_i, \sigma_i, r_i^1, r_j^2)$ l'étiquette au noeud i, l'étiquette $L_j = (C_j, \sigma_j, r_j^1, r_j^2)$ au noeud j se

calcule par les formules (A.1) à (A.4). L'étiquette L_j est réalisable uniquement si $\sigma_j \leq \sigma_{max}$, $r_j^1 \leq r_j^{1,max}$ et $r_j^2 \leq r_j^{2,max}$.

$$C_j = C_i + \bar{c}_a \tag{A.1}$$

$$\sigma_j = \begin{cases} f(\sigma_i, \delta) & \text{si } j \text{ est de la forme } r_p^h \\ \sigma_i + \sigma_a & \text{sinon.} \end{cases} \tag{A.2}$$

$$r_j^1 = \max(r_j^{1,min}, r_i^1 + r_a^1) \tag{A.3}$$

$$r_j^2 = \max(r_j^{2,min}, r_i^2 + r_a^1) \tag{A.4}$$

L'étiquette au noeud origine du graphe est $(0, 0, 0, 0)$.

De plus, lorsqu'il y a plusieurs étiquettes à un même noeud, on utilise les règles de dominance qui suivent pour réduire le nombre d'étiquettes et garder uniquement les plus intéressantes. Supposons que l'on ait deux étiquettes L_{j1} et L_{j2} au noeud j. L'algorithme d'étiquetage élimine l'étiquette L_{j2} si celle-ci est dominée par L_{j1} soit si : $C_{j1} \leq C_{j2}$, $\sigma_{j1} \leq \sigma_{j2}$, $r_{j1}^1 \leq r_{j2}^1$ et $r_{j1}^2 \leq r_{j2}^2$.

ANNEXE B SOUS-ALGORITHMES DE L'ALGORITHME GLOUTON

Algorithme 3 : Construction de E_r et E_s

Données : $\mathcal{G}$, τ, $\tilde{A}$, $\tilde{t_i}$

début

$E_r = \left\{(t_j, h, l, m) | t_j \in \tau, h \in H, l < m \in \{1, ..., N\}, (t_j, w^h_{p_l}) \in \tilde{A}, (w^h_{p_m}, \tilde{t_i}) \in \tilde{A}\right\}$;

`/* On cherche les stations de recharge qui peuvent être atteintes après un trajet de` τ `et avant le trajet` $\tilde{t_i}$ `*/`

$E_s = \left\{(t_j, d, l, m) | t_j \in \tau, d \in D, l < m \in \{1, ..., N\}, (t_j, s^d_{p_l}) \in \tilde{A}, (s^d_{p_m}, \tilde{t_i}) \in \tilde{A}\right\}$;

`/* Idem pour l'attente aux dépôts */`

retourner E_r *et* E_s

Algorithme 4 : Arcs de moindre coût à partir de E_r et E_s

Données : $\mathcal{G}$, E_r, E_s, $\tilde{t_i}$

début

$\forall (t_j, h, l, m) \in E_r,\ \phi_i(t_j, h, l, m) = c_{(t_j, w^h_{p_l})} + c_{(w^h_{p_m}, \tilde{t_i})} + \sum_{k=l}^{m-1} c_{(w^h_{p_k}, w^h_{p_{k+1}})}$; `/* Coût de passer par une station de recharge` h `*/`

$\forall (t_j, s, l, m) \in E_s,\ \psi_i(t_j, d, l, m) = c_{(t_j, s^d_{p_l})} + c_{(s^d_{p_m}, \tilde{t_i})}$; `/* Coût de passer par une attente à un dépôt` s`, l'attente au dépôt est gratuite */`

$(t_{j_0}, x, l_0, m_0) = \arg\min(\left\{\phi_i(t_j, h, l, m) | (t_j, h, l, m) \in E_r\right\} \cup \left\{\psi_i(t_j, d, l, m) | (t_j, s, l, m) \in E_s\right\})$;

si $x \in H$ **alors**

retourner $(t_{j_0}, w^x_{p_{l_0}})$ *et* $(w^x_{p_{m_0}}, \tilde{t_i})$

sinon

retourner $(t_{j_0}, s^x_{p_{l_0}})$ *et* $(s^x_{p_{m_0}}, \tilde{t_i})$

Algorithme 5 : Terminer une route

Données : $\mathcal{G}$, $(L^d)_{d\in D}$, $(\Sigma^d)_{d\in D}$, c_{tot}, $\tilde{A}$, τ, j

début

Soient d_0 le dépôt de la route j et t_{i_0} le dernier trajet de cette route;

$E_k = \left\{a | a = (t_{i_0}, k^{d_0}) \in \tilde{A}\right\}$; `/* Par construction,` E_k `contient 0 ou 1 arc */`

si $E_k \neq \emptyset$ **alors**

Soit a le seul élément de E_k;

Mettre à jour $(L^d)_{d\in D}$, $(\Sigma^d)_{d\in D}$, c_{tot}, $\tilde{A}$, τ avec l'arc a en plus (alg. 6);

retourner $(L^d)_{d\in D}$, $(\Sigma^d)_{d\in D}$, c_{tot}, $\tilde{A}$, τ

Algorithme 6 : Mettre à jour $(L^d)_{d\in D}$, $(\Sigma^d)_{d\in D}$, c_{tot}, $\tilde{A}$, τ

Données : $\mathcal{G}$, $(L^d)_{d\in D}$, $(\Sigma^d)_{d\in D}$, c_{tot}, $\tilde{A}$, τ, $(y^d)_{(d\in D)}$, arcs A à rajouter

début

si *le premier arc à rajouter commence à un dépôt* d_0 **alors**

Ajouter une route à L^{d_0} et à Σ^{d_0};

$k_0 = y^{d_0}$;

sinon

Trouver le dépôt d_0 et le numéro k_0 de la route auquel rajouter les arcs A à partir du premier noeud des arcs A et de $(L^d)_{d\in D}$;

pour $a = (i, j) \in A$ **faire**

$\Sigma^d_{k_0} \leftarrow \Sigma^d_{k_0} + \sigma_a$; `/* Ajuster l'état de charge */`

$c_{tot} \leftarrow c_{tot} + c_a$; `/* Ajuster le coût de la solution */`

Ajouter l'arc a à $L^d_{k_0}$; `/* Mettre à jour les arcs dans la route */`

si $i \in T$ **alors**

Enlever de $\tilde{A}$ tous les arcs sortant de i; `/* Il n'est plus possible d'ajouter des arcs à la suite du trajet` i `*/`

Enlever i de τ;

si $j \in R$ **alors**

Retrouver à partir de A le temps Δ passé à la recharge;

$\Sigma^d_{k_0} \leftarrow \mathrm{f}(\Sigma^d_{k_0}, \max(\delta, \frac{\Delta}{3}))$; `/* Prendre en compte une recharge partielle */`

$c_{tot} \leftarrow \mathrm{c}_{tot} + \Delta c^v_r$; `/* Prendre en compte le coût d'aller à la recharge */`

si $j \in T$ **alors**

Ajouter j à τ;

Enlever de $\tilde{A}$ tous les arcs a' sortant de j pour lesquels $\Sigma^d_{k_0} + \sigma_{a'} \geq \sigma_{max}$; `/* Enlever les arcs irréalisables */`

retourner $(L^d)_{d\in D}$, $(\Sigma^d)_{d\in D}$, c_{tot}, $\tilde{A}$, τ

ANNEXE C COÛTS RÉDUITS APPROXIMATIFS

Soit $(\pi_t)_{t\in T}$ les variables duales associées aux contraintes $\sum_{d\in D}\sum_{s\in S^d} a_s^t \times x_s = 1,\ \forall t \in T$. On a donc une variable duale pour chaque noeud t_i du graphe. Pour chaque arc (i,j) reliant le trajet t_i au trajet t_j, on note c_{ij} son coût et σ_{ij} la consommation en énergie sur cet arc.

De plus, on utilise aussi les valeurs des plus courts chemins du noeud origine vers chaque noeud t_i. Il y en a plusieurs à cause de la ressource énergie. Pour limiter ce nombre, on regroupe les valeurs des plus courts chemins par paquets en fonction de l'état de charge. On choisit un nombre de paquets R et pour chaque paquet k, chaque dépôt d et chaque trajet t_i, s'il y a plusieurs plus courts chemins qui arrivent au noeud t_i avec un état de charge de la batterie compris entre $\frac{k-1}{R}\times\sigma_{max}$ et $\frac{k}{R}\times\sigma_{max}$, on ne garde seulement la valeur la plus faible et on la note $l_k^{t_i,d}$.

Le coût réduit approximatif α_{ij} de l'arc (i,j) est calculé par la formule :

$$\alpha_{ij} = c_{ij} - \pi_j + \min_{d\in D} \min_{\substack{k\in[\![1,R]\!],\ p\in[\![1,R]\!] \\ \frac{p}{R}\times\sigma_{max}+\sigma_{ij}\geq\frac{k}{R}\times\sigma_{max}}} \left(l_k^{t_i,d} - l_p^{t_j,d}\right) \tag{C.1}$$

Il peut arriver qu'aucune combinaison de plus court chemin ne soit compatible d'un point de vue de l'énergie. Dans ce cas là, le coût réduit est calculé par :

$$\alpha_{ij} = c_{ij} - \pi_j + \min_{d\in D}\left(l_{k^*}^{t_i,d} - l_{p^*}^{t_j,d}\right) \tag{C.2}$$

où k^*, p^* sont, respectivement, les plus petits indices k et p tels que $l_k^{t_i,d}$ et $l_p^{t_i,d}$ existent.

ANNEXE D REPRÉSENTATION VISUELLE D'UNE SOLUTION

Dans la figure D.1, est représenté une solution pour une instance de type A. Chaque ligne correspond à une route. Une route est composée de différents éléments : trajets, recharges, attentes à différents endroits et voyages à vide. Les différents éléments sont représentés par un rectangle de couleur dont la longueur est proportionnelle à la durée de l'évènement. On observe que très peu de voyages à vide sont effectués, à part au début et à la fin des routes. Il y a aussi relativement peu d'attentes aux arrêts. Les véhicules vont se recharger entre 1 et 3 fois dans la journée lorsque les routes sont longues. Seulement un petit nombre de véhicules retournent attendre au dépôt et lorsqu'ils s'y retournent, c'est pour une longue période.

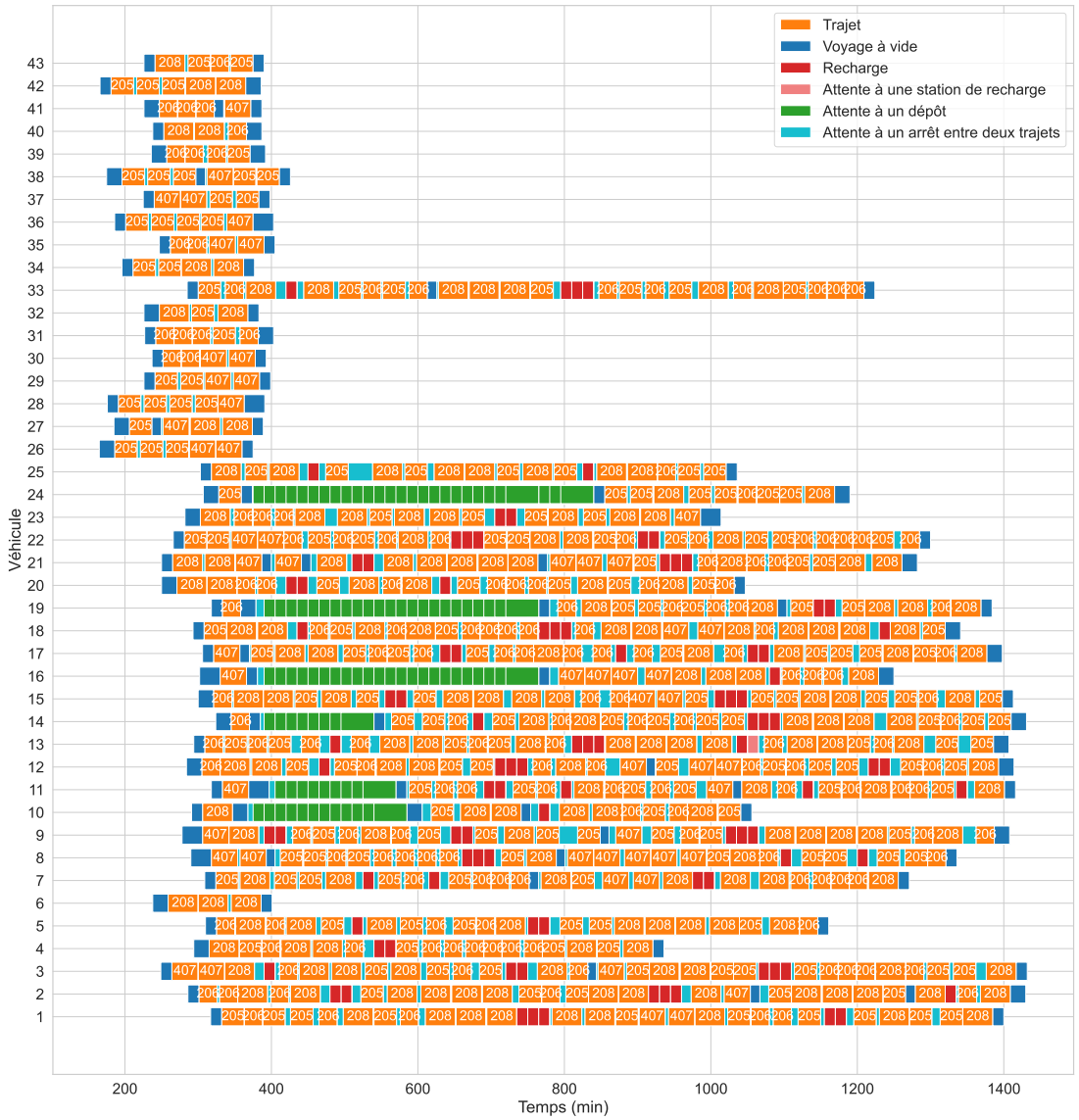

FIGURE D.1 Exemple de solution pour une instance de type A