Titre: Title:	Apprentissage d'inégalités duales pour la génération de colonnes appliquée au problème d'horaires d'autobus électriques avec dépôts multiples
Auteur: Author:	Louis Popovic
Date:	2023
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Popovic, L. (2023). Apprentissage d'inégalités duales pour la génération de colonnes appliquée au problème d'horaires d'autobus électriques avec dépôts multiples [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/54127/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/54127/
Directeurs de recherche: Advisors:	Quentin Cappart, & Guy Desaulniers
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Apprentissage d'inégalités duales pour la génération de colonnes appliquée au problème d'horaires d'autobus électriques avec dépôts multiples

LOUIS POPOVIC
Département de génie informatique et génie logiciel

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

Juin 2023

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Apprentissage d'inégalités duales pour la génération de colonnes appliquée au problème d'horaires d'autobus électriques avec dépôts multiples

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

Gilles PESANT, président
Quentin CAPPART, membre et directeur de recherche
Guy DESAULNIERS, membre et codirecteur de recherche
Issmaïl EL HALLAOUI, membre

DÉDICACE

À Georges

REMERCIEMENTS

Je tiens tout d'abord à remercier profondément mon directeur de recherche Quentin Cappart et mon codirecteur de recherche Guy Desaulniers pour leur soutien et leur engagement tout au long de ma maîtrise. Leurs précieux conseils, leur disponibilité et leurs explications claires ont grandement contribué à l'achèvement de mon mémoire. Leurs connaissances et leur expérience m'ont permis d'apprendre énormément sur le sujet de ma recherche et ont contribué à façonner mon approche méthodologique.

Je remercie également Gilles Pesant et Issmaïl El Hallaoui d'avoir accepté de faire partie du jury.

Merci aux membres du laboratoire de Quentin d'avoir illuminé mes journées passées au bureau. Merci particulièrement à Léo de m'avoir transmis une variété de connaissances, toutes plus intéressantes les unes que les autres.

Un dernier grand merci à mes parents, ma copine et mes amis de m'avoir supporté psychologiquement et financièrement tout au long de mes études. Sans eux, les dernières années auraient été beaucoup plus difficiles.

RÉSUMÉ

Le transport en commun joue un rôle primordial dans notre société. En effet, il offre une solution de mobilité pratique, durable et économique. De plus, les autobus électriques, de plus en plus présents, permettent de réduire davantage les émissions de gaz à effet de serre et contribuent ainsi à la préservation de notre environnement. L'un des aspects cruciaux de la mise en place d'un système de transport en commun efficace est la planification des horaires d'autobus électrique provenant de différents dépôts. Considérant un ensemble de trajets d'autobus à couvrir et une flotte d'autobus électriques provenant de différents dépôts, la compagnie d'autobus doit générer des horaires efficaces afin d'assurer la couverture de l'ensemble des trajets tout en minimisant le coût d'opération. Dans la littérature, c'est ce qu'on appelle le problème d'horaires d'autobus électriques avec dépôts multiples (MDEVSP). Résoudre ce problème s'avère très complexe puisqu'il est NP-difficile.

Une méthode de résolution fréquemment employée pour résoudre le MDEVSP est la génération de colonnes. Cet algorithme résout ce problème progressivement en insérant de nouveaux itinéraires à chaque itération. Cependant, certaines instances du MDEVSP souffrent de dégénérescence et la résolution prend énormément de temps.

Dans ce mémoire, on propose de contrer les effets indésirables de la dégénérescence et d'accélérer la résolution d'instances du MDEVSP à l'aide inégalités duales. Plus précisément, on commence par générer un ensemble d'instances réalistes du MDEVSP qu'on représente sous forme de réseaux de connexions, où chaque nœud représente un trajet à couvrir et où chaque arc représente une connexion valide. Une fois la relaxation linéaire du problème maître de ces instances résolues, on utilise les valeurs des variables duales associées aux trajets pour entrainer un modèle d'apprentissage automatique. Ce modèle, composé d'un réseau de neurones graphiques et d'un perceptron multicouche, prédit, pour chaque paire de variables duales d'une instance, le sens de l'inégalité ($\geq$ ou $\leq$). On utilise ensuite ces prédictions pour extraire un ensemble d'inégalités duales qu'on ajoute au primal de l'instance à l'aide de nouvelles variables. De plus, puisqu'on introduit potentiellement de fausses inégalités duales, on propose un processus de recouvrement permettant de détecter et corriger les inégalités duales potentiellement invalides.

On utilise des instances réalistes fournies par un partenaire industriel afin d'évaluer la méthode proposée. Les résultats démontrent que cette méthode permet d'accélérer la résolution de la relaxation linéaire du problème maître d'instances très dégénérées. En effet, ces inégalités duales permettent d'accélérer la résolution d'instances très dégénérées, mais elles sont

inutiles pour les instances moins dégénérées. Une des raisons est que le processus de recouvrement, qui permet de corriger certaines inégalités invalides, relance la résolution à plusieurs reprises et augmente considérablement le temps de résolution total. On estime cependant qu'avec un modèle plus précis et un processus de recouvrement plus efficace, la méthode proposée pourrait offrir de meilleures accélérations.

ABSTRACT

Public transit plays a vital role in our society. Indeed, it offers a practical, sustainable and economical mobility solution. In addition, electric buses, which are becoming more and more popular, further reduce greenhouse gas emissions and thus contribute to the preservation of our environment. One of the crucial aspects of building an efficient transit system is scheduling electric buses from multiple depots. Considering a set of bus routes to be covered and a fleet of electric buses from different depots, the bus company must generate efficient schedules in order to ensure coverage of all routes while minimizing the cost of operation. In the literature, this is known as the Multiple Depot Electric Vehicle Scheduling Problem (MDEVSP). Solving this problem turns out to be very complex since it is NP-hard.

Column generation is a method commonly used to solve the MDEVSP. This algorithm solves this problem gradually by inserting new routes at each iteration. However, some instances of the MDEVSP suffer from degeneracy and the resolution takes a very long time.

In this thesis, we propose to counter the undesirable effects of degeneracy and accelerate the resolution of instances of the MDEVSP using dual inequalities. More precisely, we start by generating a set of realistic instances of the MDEVSP which we represent using a connections network, where each node represents a path to be covered and where each arc represents a valid connection. Once the linear relaxation of the master problem of these instances has been solved, we use the values of the dual variables associated with the bus trips to train a machine learning model. This model, composed of a graph neural network and a multilayer perceptron, predicts, for each pair of dual variables of an instance, the direction of the inequality ($\geq$ or $\leq$). We then use these predictions to extract a set of dual inequalities that we add to the primal of the instance using new variables. Moreover, since we potentially introduce invalid dual inequalities, we propose a recovering process allowing to detect and correct potentially invalid dual inequalities.

Realistic instances provided by an industrial partner are used to evaluate the proposed method. The results show that the proposed method can speed up the resolution of the linear relaxation of the master problem of very degenerate instances. Indeed, the predicted dual inequalities make it possible to accelerate the resolution of very degenerate instances, but they are useless for less degenerate instances. One of the reasons is that the recovery process, which makes it possible to correct certain invalid inequalities, restarts the resolution several times and considerably increases the total resolution time. However, it is believed that with a more accurate model and a more efficient recovery process, the proposed method

could offer better speedups.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

MDEVSP	Problème d'horaires de véhicules électrique avec dépôts multiples
GNN	Réseau de neurones graphique
MDVSP	Problème d'horaires de véhicules avec dépôts multiples
SDEVSP	Problème d'horaires de véhicules électriques avec dépôt unique
SDVSP	Problème d'horaires de véhicules avec dépôt unique
PM^e	Problème maître en nombres entiers
PM	Relaxation linéaire du problème maître
PMR	Problème maître restreint
SP	Sous-problème
PM^{ID}	Problème maître avec inégalités duales

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

1.1 Contexte de la recherche

Le transport collectif est un service essentiel pour les villes puisqu'il apporte toute sorte de bienfaits économiques, sociaux et environnementaux. Un système de transport en commun efficace permet de réduire la pollution atmosphérique, la congestion et les frais de déplacement (LaPresse, 2021).

Pour ces raisons, l'optimisation des transports en commun est un enjeu primordial pour les villes. En effet, pour être utilisé fréquemment par les citoyens, le service offert doit être de qualité et ponctuel afin de permettre aux citoyens de se déplacer facilement. Les compagnies de transport en commun utilisent souvent des systèmes d'aide à la décision pour optimiser leur service. En ce qui concerne les réseaux d'autobus, il existe 4 étapes importantes d'optimisation :

1. **Planification du réseau**. La première étape consiste à déterminer les trajets des lignes d'autobus afin de desservir les zones à fortes demandes. Pour chaque ligne d'autobus, on détermine les lieux de départ et d'arrivée ainsi que les arrêts tout au long du parcours.
2. **Planification des horaires**. Ensuite, on détermine les fréquences et les horaires des départs de chaque ligne, en veillant à ce que les intervalles entre les départs soient assez réguliers et adaptés à la demande des utilisateurs. Les horaires doivent tenir compte de plusieurs facteurs tels que la zone desservie, le jour de la semaine et la période de la journée.
3. **Planification des itinéraires d'autobus**. Après avoir défini l'ensemble de trajets à couvrir, on planifie les itinéraires d'autobus. Considérant la flotte d'autobus disponible et l'ensemble des trajets à couvrir, on assigne aux autobus des suites de trajets à effectuer. Le but est de générer des itinéraires qui couvrent l'ensemble des trajets tout en minimisant les coûts d'opération.
4. **Planification des horaires des chauffeurs**. Finalement, la dernière étape consiste à affecter des itinéraires ou des segments d'itinéraires d'autobus aux chauffeurs. Cette planification doit prendre en considération les différentes contraintes de ressources telles que les pauses, le nombre d'heures de travail consécutif et les congés.

En pratique, il est très difficile d'optimiser conjointement ces 4 étapes puisque les compagnies sont confrontées à des systèmes composés de plusieurs dizaines de lignes et de centaines, voire

milliers, d'autobus et chauffeurs. Ces 4 étapes sont donc souvent résolues indépendamment, ce qui fait en sorte que l'on se retrouve avec un service qui n'est pas optimal dans l'ensemble, mais tout de même satisfaisant.

Récemment, les différentes étapes de planification ont vu leur complexité augmenter avec la croissance en popularité des autobus électriques. En effet, lorsqu'on utilise des autobus à essence, l'autonomie est très grande, les temps de ravitaillement sont rapides et les stations d'essence fréquentes. À l'inverse, lorsqu'on utilise des autobus électriques, l'autonomie est beaucoup plus restreinte, les temps de recharge sont très lents et les stations de recharge plus rares, ce qui complexifie énormément les étapes de planification.

Dans le cadre de ce mémoire, on se concentre sur la planification des itinéraires d'autobus électriques avec dépôts multiples. Plus précisément, ce problème consiste à assigner des séquences de trajets et de recharges à différents autobus électriques provenant de différents dépôts. Dans la littérature, ce problème se nomme le *problème d'horaires de véhicules électriques avec dépôts multiples* (MDEVSP). Aujourd'hui, une des méthodes de résolution fréquemment employée pour résoudre ce problème est la génération de colonnes. Sommairement, cet algorithme résout progressivement le problème et y ajoutant de nouvelles variables (des itinéraires dans le cas du MDEVSP) à chaque itération. À chaque étape de l'algorithme, on résout un sous-problème pour déterminer quelles nouvelles variables ajouter au modèle, jusqu'à ce que la solution optimale soit atteinte. Cet algorithme, qui est présenté en détail à la Section 3.5, permet d'obtenir une solution optimale. Cependant, lorsque le problème est complexe et les instances très grandes, cet algorithme devient couteux en termes de temps et converge difficilement. Cette dégénérescence apparaît lorsqu'il existe plusieurs solutions de base réalisables qui se trouvent au même point extrême du polyèdre représentant l'espace des solutions réalisables. Cela peut rendre plus difficile le choix des variables à ajouter au problème maître et causer des inefficacités dans l'algorithme du simplexe, utilisé pour le résoudre. En effet, durant plusieurs itérations, des variables peuvent être insérées dans la base sans jamais améliorer la valeur de la fonction objectif.

1.2 Objectifs de recherche

Une façon de combattre la dégénérescence est de réduire l'espace de recherche en ajoutant des inégalités duales valides. Chaque programme linéaire sous forme standard possède une représentation duale où les contraintes correspondent à des variables et les variables à des contraintes. Ajouter une inégalité au dual d'un programme se traduit par l'ajout d'une variable au primal. Cependant, obtenir des inégalités duales valides et pertinentes est une tâche difficile. En effet, ces inégalités sont souvent très spécifiques aux instances et nécessitent une

analyse exhaustive du problème. De plus, la seule façon d'obtenir les valeurs des variables duales (et ainsi générer des inégalités duales valides) est de résoudre à l'optimalité l'instance. On propose donc d'utiliser de l'apprentissage supervisé basé sur des résolutions d'instances du MDEVSP pour apprendre à prédire ces inégalités duales. Aussi, puisqu'on utilise un modèle de prédiction imparfait, on introduit certaines inégalités duales invalides. On propose donc un processus qui tente de détecter ces inégalités duales invalides et de les réparer, afin d'obtenir une meilleure solution. Finalement, on évalue cette approche à l'aide d'instances réalistes provenant d'un réseau d'autobus fourni par un partenaire industriel. Les résultats obtenus permettent de confirmer le potentiel de cette approche.

Plus précisément, les contributions de ce mémoire sont les suivantes :

1. On propose un modèle d'apprentissage supervisé qui utilise des réseaux de neurones graphiques pour prédire des inégalités duales d'instances du MDEVSP ;
2. On présente un algorithme qui sélectionne de façon efficace un sous-ensemble d'inégalités duales prédites pertinentes à ajouter aux instances afin d'accélérer la résolution de celles-ci ;
3. On introduit un processus de recouvrement permettant de détecter et réparer des inégalités duales potentiellement invalides qui ont été introduites dans les instances.

Ce travail a été fait en collaboration avec GIRO, une compagnie basée à Montréal œuvrant, entre autres, dans le développement d'algorithmes d'optimisation des transports en commun.

1.3 Plan du mémoire

On commence par présenter quelques travaux effectués concernant le MDEVSP et d'autres problèmes connexes, ainsi que sur la popularité de l'hybridation entre l'apprentissage machine et les problèmes d'optimisation combinatoires (chapitre 2). Ensuite, dans le chapitre 3, on définit formellement le problème du MDEVSP ainsi que l'algorithme de génération de colonnes utilisé pour le résoudre. On discute aussi des limitations de cet algorithme. Au chapitre 4, on présente la contribution principale du mémoire en définissant les inégalités duales ajoutées, en présentant le modèle d'apprentissage machine développé permettant de prédire et d'ajouter ces inégalités aux instances et en introduisant le processus de réparation. Au chapitre 5, on présente des résultats théoriques et expérimentaux afin de critiquer et analyser les performances du modèle sur des instances réalistes. Finalement, on conclut au chapitre 6 ce mémoire en revenant sur la méthode proposée et on suggère des améliorations et des idées de continuation.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre est divisé en quatre parties : on commence par présenter quelques travaux portant sur différents problèmes d'horaires d'autobus. Ensuite, on aborde différentes approches utilisées pour accélérer la convergence de la génération de colonnes. De plus, on présente des travaux combinant l'optimisation combinatoire et l'apprentissage machine. Finalement, on introduit les réseaux de neurones gaphiques (GNNs), une technologie utilisée dans ce travail de recherche. On conclut ce chapitre en présentant la contribution de ce mémoire.

2.1 Problèmes d'horaires d'autobus

On présente ici quelques travaux réalisés sur des problèmes d'horaires d'autobus. On commence par introduire le problème d'horaires d'autobus avec dépôts multiples (MDVSP) et la façon dont il est résolu à l'optimalité. Ensuite, on aborde des problèmes d'horaires d'autobus électriques. On sépare ces problèmes en deux catégories : les problèmes d'horaires d'autobus électriques avec dépôt unique (SDEVSP) et ceux avec dépôts multiples (MDEVSP). On discute des différentes méthodes de résolution et des différentes hypothèses posées dans les définitions des problèmes.

2.1.1 Problème d'horaires d'autobus avec dépôts multiples

Le MDVSP est un problème qui a été démontré par Bertossi *et al.* (1987) comme étant NP-difficile. Carpaneto *et al.* (1989) proposent de modéliser ce problème à l'aide d'un réseau de connexions, où chaque noeud représente un trajet à couvrir et chaque arc une connexion, et de le résoudre de manière optimale en utilisant une méthode de séparation et d'évaluation (*branch and bound*). Plus tard, Ribeiro et Soumis (1994) considèrent le MDVSP comme un problème de partitionnement d'ensemble et suggèrent de le résoudre en utilisant un algorithme de génération de colonnes. Ces travaux sont pertinents, car ils introduisent une modélisation et une méthode de résolution qu'on utilisera pour résoudre le MDEVSP. Löbel (1999) introduit une nouvelle méthode de résolution exacte, appelée *Lagrangean Pricing*. Cette technique se base sur la formulation multi-flot du MDVSP. Il utilise la relaxation Lagrangienne du MDVSP et la génération de colonnes pour résoudre à l'optimalité, ou presque, des instances du MDVSP. Kliewer *et al.* (2006) proposent de modéliser le MDEVSP à l'aide d'un réseau espace-temps. Contrairement au réseau de connexions, le réseau espace-temps représente chaque mouvement d'autobus possible par un arc dans le temps et l'espace. Cette

modélisation permet de réduire le nombre d'arcs nécessaire et permet de résoudre efficacement de grandes instances du MDVSP à l'aide d'un programme en nombres entiers. On réfère le lecteur à Desrosiers *et al.* (1995) et Desaulniers et Hickman (2007) pour des revues plus complètes sur le MDVSP.

2.1.2 Problème d'horaires d'autobus électriques

On se concentre maintenant sur différentes variantes du problème d'horaires de véhicules électriques. Contrairement aux problèmes d'horaires de véhicules à essence, les problèmes d'horaires de véhicules électriques doivent tenir compte de plusieurs contraintes supplémentaires, comme le soulignent Sassi et Oulamara (2014) :

— L'autonomie des véhicules électriques est limitée et ne peut pas être considérée comme suffisante pour la totalité de l'horizon d'optimisation.
— Les temps de chargement des véhicules électriques ne peuvent pas être négligés et doivent être considérés dans la planification d'horaires.
— En raison du nombre limité d'infrastructures de recharge, il est nécessaire de planifier des détours et des visites aux stations de recharge lors de la planification d'horaires.

Situation avec un seul dépôt

On présente maintenant quelques travaux se concentrant sur le SDEVSP. Contrairement au problème d'horaires de véhicules avec dépôt unique (SDVSP), qui peut être résolu en temps polynomial, le SDEVSP est NP-difficile.

Li (2014) propose un modèle avec échange de batterie ou recharge rapide. Dans ce modèle, les temps et les coûts d'échange et de recharge sont constants. Il propose aussi un modèle avec des contraintes de longueur maximale des itinéraires. Ces deux modélisations sont résolues à l'aide d'un algorithme de génération de colonnes. Sassi et Oulamara (2014) complexifient ce problème en introduisant une fonction de charge linéaire, où la quantité d'énergie rechargée dépend du temps. Ils proposent une formulation de programmation linéaire mixte en nombres entiers et résolvent à l'optimalité des petites et moyennes instances. Pour les instances de plus grande taille, ils proposent deux heuristiques : une heuristique séquentielle qui construit des itinéraires pour chaque autobus un à la suite de l'autre et une heuristique globale qui construit l'ensemble des itinéraires simultanément.

Chao et Xiaohong (2013) présentent un modèle hybride entre les deux approches précédentes. Dans leur définition du problème, ils permettent aux autobus de changer leur batterie en un temps rapide et constant ou de la recharger. Encore une fois, la quantité d'énergie rechargée

dépend du temps de recharge. Ils proposent une formulation multi-objectif. Le premier objectif vise à réduire le nombre d'autobus et de batteries nécessaires, tandis que le second tente de minimiser la demande de charge totale pour l'ensemble des bornes de recharge. Pour résoudre le problème, ils présentent une approche d'optimisation multi-objectif adaptée, qui reprend les principes de base de l'algorithme génétique de tri non dominé (NSGA-II) introduit par Deb *et al.* (2000).

Van Kooten Niekerk *et al.* (2017) se concentrent sur le problème en excluant la possibilité d'échanger les batteries et en ne permettant que des visites aux stations de recharge. Ils présentent un premier modèle qui suppose que les temps de charge sont linéaires, que le prix de l'électricité est constant et qui ignore la dégradation de la batterie. Le deuxième modèle présenté est plus complexe et plus représentatif de la réalité ; il permet de décrire des temps de recharge selon n'importe quelle fonction pouvant être discrétisée, de prendre en considération la variabilité du prix de l'électricité durant l'horizon d'optimisation et d'inclure les coûts liés à la dégradation de la batterie. De petites et moyennes instances des deux modèles peuvent être résolues à l'optimalité en utilisant la programmation linéaire en nombres entiers. Une méthode de résolution basée sur la génération de colonnes est présentée afin de résoudre les grandes instances du deuxième modèle sans toutefois garantir l'optimalité.

Ces travaux démontrent qu'il existe plusieurs façons de modéliser la recharge des batteries. En effet, certaines modélisations sont plus simples, mais plus faciles à résoudre, tandis que d'autres utilisent une modélisation plus réaliste et complexe, mais plus difficile à résoudre. On utilisera, dans la définition du problème, une approximation de la vraie fonction de charge par une fonction linéaire par morceaux qui a été démontrée par Montoya *et al.* (2017) comme excellente par rapport à la vraie fonction de charge.

Avec dépôts multiples

On se concentre maintenant sur le MDEVSP. Encore une fois, on présente différentes modélisations et méthodes de résolution.

Guo *et al.* (2019) présentent un modèle de base du MDEVSP avec un temps de recharge linéaire. L'auteur propose une méthode de résolution hybride combinant un algorithme génétique et un algorithme de génération de colonnes. Cette méthode s'avère plus rapide que le *branch & price* traditionnel grâce à la substitution du *branch & bound* par un algorithme génétique.

Wu *et al.* (2022) abordent le MDEVSP à double objectif qui vise à minimiser les coûts opérationnels et la charge de pointe liée aux activités de recharge. L'auteur introduit une

modélisation de réseaux temps-étendu dans laquelle le temps est discrétisé en intervalles. On s'inspire d'ailleurs de cette modélisation pour représenter les activités de recharges.

Li *et al.* (2019) présentent une variante du MDVSP avec plusieurs types de véhicules incluant des autobus électriques. Ils définissent des contraintes liées à l'autonomie limitée et à la recharge et présentent une nouvelle modélisation du problème avec un réseau espace-temps-énergie (ETE) pour les autobus et un réseau espace-temps (ET) pour les passagers. Dans le réseau ETE, chaque nœud représente un lieu spécifique à un moment spécifique et à un niveau d'énergie spécifique. Ceci permet de garder une trace de la consommation d'énergie de chaque type de véhicule différent.

Finalement, Gkiotsalitis *et al.* (2023) étendent le problème d'horaires d'autobus avec dépôts multiples et fenêtres de temps (MDVSP-TW) au cas de véhicules électriques. Leur modèle de programmation linéaire mixte en nombres entiers considère le coût opérationnel et les temps d'attente. De plus, ils prennent en considération les capacités des différentes stations de recharge. Finalement, les auteurs proposent des inégalités valides qui permettent de réduire l'espace de recherche et, en conséquence, accélérer la résolution des instances.

On remarque que la littérature concernant le MDEVSP est moins abondante et plus récente que celle concernant le MDVSP et le SDEVSP, probablement dû au fait que c'est un problème moins considéré en pratique pour l'instant. Par contre, on note qu'il existe, encore une fois, plusieurs modélisations et méthodes de résolution différentes ayant chacune des avantages et désavantages. On conclut aussi qu'au meilleur de nos connaissances, il n'existe aucune approche résolvant le MDEVSP à l'aide de réseaux de neurones graphiques.

2.1.3 Dégénérescence

Les instances du MDVSP et du MDEVSP sont souvent dégénérées et les temps de résolution peuvent être très longs. Lorsqu'on résout ces instances avec un algorithme de génération de colonnes, la convergence peut être très lente et plusieurs itérations de génération de colonnes peuvent être nécessaires avant d'améliorer la solution. Il existe plusieurs techniques qui peuvent être utilisées afin de contrer ces effets indésirables. On présente maintenant quelques-unes de ces méthodes.

Une première approche consiste à ajouter des inégalités duales pour réduire l'espace dual comme nous le ferons dans ce mémoire. Ben Amor *et al.* (2006) proposent deux types d'inégalités duales valides afin d'accélérer et stabiliser le processus de convergence pour des problèmes de découpe de rouleaux. L'ensemble ou un sous-ensemble des solutions duales optimales satisfait les nouvelles contraintes ajoutées. Puisque l'ajout d'inégalités duales se

traduit par l'ajout de colonnes dans le primal, ceci peut mener à des solutions primales invalides. Les auteurs proposent alors deux méthodes pour réparer ces solutions invalides. Carvalho (2005), Gschwind et Irnich (2016) et Yarkony *et al.* (2021) proposent, eux aussi, des inégalités duales permettant de stabiliser la résolution du primal pour des problèmes de découpe et d'empaquetage.

Elhallaoui *et al.* (2005) proposent une technique d'agrégation dynamique de contraintes qui s'applique aux problèmes ayant des contraintes de partitionnement d'ensemble. Ils notent que si pour l'ensemble des itinéraires (colonnes) générés, un sous-ensemble de trajets (tâches) est soit couvert ou non, alors on peut voir ces tâches comme une seule tâche. Ils proposent alors de remplacer les contraintes de partitionnement correspondantes par une seule contrainte de partitionnement équivalente. Ils expliquent aussi que puisqu'on ne connait pas au préalable quelles tâches peuvent être regroupées, l'ensemble des contraintes du problème maître restreint doit être modifié dynamiquement lors du processus de résolution. Des versions améliorées ont été proposées par Elhallaoui *et al.* (2008) et Elhallaoui *et al.* (2010).

Des techniques de stabilisation des variables duales sont aussi employées pour contrer les effets de la dégénérescence. Ben Amor *et al.* (2004) proposent d'ajouter un terme de stabilisation à la formulation du dual afin de stabiliser l'algorithme de génération de colonnes et accélérer la résolution. Plus spécifiquement, ils démontrent l'efficacité de l'approche en l'appliquant à des instances de grande taille du MDVSP. Oukil *et al.* (2007) proposent aussi une méthode permettant de réduire la dégénérescence dans des instances du MDVSP. Leur approche permet d'insérer de nouvelles variables de perturbation dans le problème maître afin de stabiliser la résolution. Benchimol *et al.* (2012) proposent une technique combinant l'agrégation dynamique de contraintes et la stabilisation de variables duales pour résoudre des problèmes de partitionnement d'ensemble. Finalement, on réfère le lecteur à Pessoa *et al.* (2018) pour une revue plus complète sur les techniques de stabilisation de variables duales.

2.2 L'apprentissage machine pour résoudre des problèmes d'optimisation combinatoire

Récemment, plusieurs travaux se sont intéressés à l'utilisation de l'apprentissage machine pour résoudre des problèmes d'optimisation combinatoire. L'objectif est d'obtenir des résultats de qualité similaire, plus rapidement. Bengio *et al.* (2021) présentent trois familles de méthodes combinant l'apprentissage machine et l'optimisation combinatoire ; utiliser un modèle d'apprentissage machine pour résoudre directement les problèmes d'optimisation combinatoire, utiliser un modèle d'apprentissage machine pour configurer l'algorithme qui résout les problèmes d'optimisation combinatoire et utiliser un modèle d'apprentissage machine à

l'intérieur d'un solveur de problèmes d'optimisation combinatoire. L'approche proposée dans ce mémoire se retrouve dans la troisième famille. On réfère le lecteur à Bengio *et al.* (2021) pour une revue plus complète des récents travaux combinant l'apprentissage machine et l'optimisation combinatoire.

On retrouve plusieurs travaux qui utilisent de l'apprentissage par renforcement ou par imitation afin de reproduire des décisions efficaces, mais très coûteuses en termes de calculs, prises dans des algorithmes de résolution de problèmes combinatoires. L'idée générale est de définir une politique de décision efficace, mais coûteuse (exhaustive ou gloutonne par exemple) et de générer un ensemble d'entraînement entrée-sortie en utilisant cette politique. Les données d'entrée représentent l'état du problème (à l'aide d'un vecteur ou d'un graphe) au moment de prendre une décision et les données de sortie représentent la décision prise par la politique efficace, mais coûteuse. Ensuite, on utilise cet ensemble pour entraîner un modèle à imiter ou apprendre ces décisions. Dans cet ordre d'idée, Alvarez *et al.* (2017) présentent une méthode permettant de reproduire un branchement fort qui permet d'accélérer l'algorithme de résolution du *branch & bound* pour résoudre des problèmes mixtes en nombres entiers. Similairement, Paulus *et al.* (2022) présentent une méthode qui imite un algorithme d'anticipation glouton pour déterminer quels plans coupants ajouter afin d'obtenir les meilleures bornes inférieures possibles.

On peut aussi utiliser l'apprentissage machine pour trouver directement des solutions à différents problèmes d'optimisation combinatoire. Au lieu d'imiter une décision prise dans un processus de résolution comme présenté précédemment, l'objectif est de remplacer intégralement ce processus de résolution par un modèle d'apprentissage automatique. Encore une fois, le but est d'obtenir une solution, de qualité similaire, le plus rapidement possible. Contrairement aux méthodes présentées plus tôt, les modèles sont souvent entraînés sur des problèmes spécifiques.

Par exemple, Hottung *et al.* (2020) proposent un modèle d'apprentissage profond qui résout des instances du problème de pré-rassemblement de conteneurs (CPMP) rapidement. Le CPMP consiste à déterminer une séquence d'opérations minimale permettant de réduire le nombre de conteneurs bloquants dans une cour d'opérations. La méthode se base sur une recherche d'arborescence, où à chaque fois qu'un nœud est atteint dans l'arbre de l'espace de recherche, un modèle d'apprentissage profond prédit quel prochain nœud devrait être exploré afin de maximiser les chances d'obtenir une bonne solution rapidement. De plus, un second modèle est utilisé pour prédire la borne inférieure pouvant être atteinte si ce nœud est exploré. Encore une fois, l'entraînement est fait sur un ensemble d'instances résolues à l'aide d'un algorithme exact où les solutions sont représentées par des séquences d'opérations

et les états de la cour par des vecteurs.

Ces travaux mettent en avant les avantages de l'utilisation de l'apprentissage machine pour résoudre des problèmes d'optimisation combinatoire. Les méthodes présentées tirent parti du fait qu'elles apprennent à résoudre ces problèmes en utilisant des solutions d'instances précédemment résolues pour résoudre de nouvelles instances. Cela représente un avantage significatif par rapport aux méthodes de résolution exacte, qui doivent recommencer la résolution à partir de zéro à chaque fois qu'une nouvelle instance est rencontrée. En réutilisant les solutions précédemment trouvées, les méthodes basées sur l'apprentissage machine permettent de réduire considérablement le temps de calcul nécessaire pour obtenir de bonnes solutions à de nouveaux problèmes d'optimisation combinatoire.

2.3 Réseaux de neurones graphiques

Puisque les réseaux de neurones graphiques (GNNs pour *graph neural networks* en anglais) sont une partie intégrale de la méthode présentée dans ce mémoire, on présente maintenant quelques références et travaux importants.

Les GNNs (Scarselli *et al.* (2009), Gilmer *et al.* (2017)) sont des structures d'apprentissage automatique qui sont en mesure d'opérer sur des données structurées sous forme de graphes et de distinguer différents schémas structurels dans des graphes (Cappart *et al.* (2021)). Contrairement aux réseaux de neurones classiques qui traitent les données en entrée sous forme de vecteurs ou de matrices, les GNNs prennent en compte les relations entre les différents nœuds et arêtes d'un graphe. Ils utilisent les relations entre les nœuds voisins et les arêtes connexes pour générer de nouvelles représentations vectorielles (*embeddings*) propre à chaque nœud, arêtes et/ou graphe. Ils utilisent des opérations de propagation des informations le long des arêtes afin de générer de nouvelles représentations vectorielles des nœuds et arêtes. En d'autres termes, les opérations de propagation permettent aux nœuds de propager de l'information sur leur contexte local aux nœuds voisins. Les nouvelles représentations apprises par le modèle sont utilisées pour des tâches de classification de nœuds, de graphes ou de prédiction d'arêtes. On réfère le lecteur à Wu *et al.* (2020) pour une revue plus complète sur les GNNs et leurs applications. De plus, on présente à la Section 4.2.1, le fonctionnement en détail d'un type de GNN utilisé dans cette recherche.

2.4 Contribution de ce mémoire

À la Section 2.1.2, on a examiné différentes façons de modéliser les temps de recharge des véhicules électriques : linéaires, constants ou par morceau. On a aussi mis en évidence que

la résolution de grandes instances du MDEVSP est difficile et que ces instances peuvent être très dégénérées. On a ensuite présenté quelques techniques qui permettent de combattre les effets indésirables de la dégénérescence telles que les inégalités duales, la stabilisation des variables duales et l'agrégation dynamique de contraintes. De plus, on a aussi constaté que l'apprentissage machine est de plus en plus utilisé pour obtenir de bonnes solutions rapidement à des problèmes d'optimisation combinatoire. Enfin, on a présenté les GNNs, des structures d'apprentissage machine utilisées pour exploiter des données structurées sous forme de graphe. Suite à cette revue de littérature, on se pose la question suivante : est-il possible d'utiliser des GNNs pour prédire des inégalités duales dans des instances du MDEVSP, modélisés sous forme de graphe et avec des temps de recharge linéaires par morceau, et ainsi combattre les effets indésirables de la dégénérescence et obtenir rapidement de bonnes solutions ?

On tente de répondre à cette question dans ce mémoire. Plus précisément, on propose :

— Un modèle d'apprentissage machine permettant de prédire des inégalités duales pour des instances du MDEVSP ;
— Un algorithme permettant de générer des séries d'inégalités duales efficaces à ajouter aux instances afin d'accélérer la résolution de celles-ci.

L'approche proposée semble être efficace pour certaines instances très dégénérées. Cependant, pour des instances moins dégénérées dont la résolution est déjà rapide, on conclut que l'approche n'est pas suffisante pour réduire les temps d'exécution. Finalement, on présente au chapitre 6 des suggestions d'améliorations pouvant potentiellement améliorer l'efficacité de cette approche.

CHAPITRE 3 DÉFINITION DU PROBLÈME ET MÉTHODE DE RÉSOLUTION

On présente à la Section 3.1 une définition générale du MDEVSP. Ensuite, à la Section 3.2, on donne une définition formelle du MDEVSP avec des notations qui seront utilisées tout au long du mémoire. À la section suivante, on introduit la formulation mathématique du MDEVSP. À la section 3.4 on définit le réseau de connexions utilisé. La section suivante présente l'algorithme de génération de colonne utilisée pour résoudre à l'optimalité les instances du MDEVSP. Finalement, on termine ce chapitre en présentant une technique permettant de réduire la dégénérescence.

3.1 Problème d'horaires de véhicules électriques

Le MDVSP est un problème d'optimisation combinatoire qui consiste à assigner des suites (itinéraires) de tâches (trajets) à des autobus provenant de différents dépôts. L'objectif est de couvrir toutes les tâches en minimisant le coût total d'opération. Plus précisément, le MDVSP implique un ensemble de dépôts avec des capacités définies en termes d'autobus ainsi qu'un ensemble de trajets à effectuer avec des heures et emplacements de départs et d'arrivées. L'objectif est d'assigner des itinéraires de trajets à des autobus qui commencent et finissent leur journée au même dépôt, tout en minimisant le coût total et en garantissant que chaque trajet soit couvert exactement une fois. Le MDVSP ne prend pas en considération les contraintes d'autonomie des autobus, car on suppose que les autobus à essence ont une autonomie suffisante, que les temps de remplissage sont négligeables et que les stations d'essence sont nombreuses. Le coût total est déterminé en fonction du nombre d'autobus utilisé, des temps d'attente et de la distance parcourue par l'ensemble des autobus.

Le MDEVSP est une variante du MDVSP impliquant l'utilisation d'autobus électriques. Contrairement aux véhicules à essence, les autobus électriques ont une autonomie limitée, les temps de recharge sont plus longs et les stations de recharge plus rares. Ceci doit être pris en considération lors de la planification des itinéraires. Étant donné un ensemble de dépôts et un ensemble de trajets à effectuer, un ensemble de stations de recharge avec des capacités définies en termes de bornes disponibles, l'autonomie maximale des autobus, leur consommation d'énergie par unité de distance parcourue ainsi que la quantité d'énergie rechargée par unité de temps, l'objectif du MDEVSP est le même que celui du MDVSP ; assigner des suites de trajets à des autobus tout en minimisant le coût total d'opération. Le coût total est calculé de la même façon que pour le MDVSP.

Finalement, le SDVSP et le SDEVSP sont des cas particuliers du MDVSP et du MDEVSP, respectivement. En effet, la seule différence est le fait qu'il y ait un seul dépôt. On note aussi que le SDVSP est un problème qui se résout en temps polynomial, mais que le SDEVSP ne l'est pas, à cause des contraintes d'autonomie.

3.2 Définition formelle du MDEVSP

On présente ici une définition plus formelle et plus complète du MDVSP et du MDEVSP. Les notations présentées ici seront utilisées tout au long du mémoire.

Le MDVSP peut être défini comme suit : On considère un ensemble de n trajets à effectuer $\mathcal{T} = \{t_1, t_2, t_3, ..., t_n\}$. Le trajet t commence à la station S_t au temps h_t^s et termine à la station E_t au temps h_t^e. Les autobus partent et reviennent au même dépôt. On dispose de m dépôts $\mathcal{K} = \{k_1, k_2, k_3, ..., k_m\}$. Le dépôt k a une capacité de g^k autobus. Soit θ_{ij} le temps nécessaire pour se rendre de la station où le trajet t_i termine (E_i) à la station où le trajet t_j débute (S_j). On définit la paire de trajets consécutifs (t_i, t_j) comme étant valide si un autobus peut compléter le trajet t_j immédiatement après avoir complété le trajet t_i et si le temps total entre ces deux trajets n'excède pas δ unités de temps. Ces deux conditions s'expriment mathématiquement comme suit :

$$h_i^e + \theta_{ij} \leq h_j^s \tag{3.1}$$

$$h_j^s - h_i^e \leq \delta \tag{3.2}$$

La Condition (3.2) provient d'un souci de productivité et empêche d'affecter un autobus et son chauffeur à un temps non productif (déplacement à vide, attente ou les deux) trop long. Il y a un temps d'attente si le véhicule arrive à S_j avant h_j^s. Ce temps d'attente s'exprime comme suit :

$$w(t_i, t_j) = h_j^s - h_i^e - \theta_{ij} \tag{3.3}$$

L'itinéraire d'un autobus est défini comme une suite de trajets telle que chaque paire de trajets consécutifs est considérée valide. Un itinéraire doit commencer et terminer au même dépôt. On dit qu'un autobus effectue une connexion lorsqu'il effectue deux trajets l'un à la suite de l'autre. Une connexion est une attente en station et/ou un trajet à vide entre deux stations (trajet haut-le-pied). L'objectif est d'assigner un ensemble d'itinéraires au moindre

coût aux autobus de sorte que chaque trajet dans $\mathcal{T}$ soit couvert exactement une fois et que les capacités des dépôts $\mathcal{K}$ soient respectés. Le coût d'un itinéraire (c_s) est calculé de la façon suivante :

$$c_s = c_f + c_d p_d + c_a p_a \tag{3.4}$$

Où c_f est un coût fixe lié à l'utilisation d'un autobus, c_d est le coût par unité de temps de déplacement à vide, p_d est le temps total de déplacement à vide de l'itinéraire, c_a est le coût par unité de temps d'attente et p_a est le temps total d'attente de l'itinéraire. On néglige les coûts associés au temps de déplacement des trajets, car ceux-ci doivent être effectués pour que la solution soit valide.

Pour définir le MDEVSP, on se base sur cette définition du MDVSP à laquelle on ajoute les spécifications suivantes. On suppose que les autobus débutent la journée avec une autonomie complète (σ_{max}) et que la consommation d'énergie est proportionnellement linéaire à la distance parcourue. L'état de charge de la batterie des autobus doit tout le temps se situer entre σ_{min} et σ_{max} inclusivement. Pour recharger leurs batteries, les autobus peuvent visiter n'importe quelle station de recharge et utiliser une borne. On dispose d'un ensemble de r stations de recharge $\mathcal{H} = \{h_1, h_2, ..., h_r\}$. Il y a b_h bornes disponibles à la station de recharge h. Plusieurs autobus peuvent se rendre simultanément à la même station de recharge, mais un maximum de b_h autobus peuvent recharger simultanément. La vitesse de recharge est la même pour toutes les bornes et stations. Elle est dérivée d'une fonction f linéaire par morceaux de l'état de charge initial et du temps de recharge. L'équation suivante donne l'état final (q^{fin}) d'une batterie qui a comme état initial q^{ini} après une recharge de Δ unités de temps :

$$q^{fin} = f\left(f^{-1}(q^{ini}) + \Delta\right) \tag{3.5}$$

La Figure 3.1 représente la fonction linéaire par morceaux f utilisée pour calculer la quantité d'énergie rechargée. En pratique, la vitesse à laquelle la batterie se recharge tend à diminuer, plus la batterie est pleine. En effet, les fonctions de recharge ne sont pas linéaires, car la tension et le courant varient en fonction du temps afin d'éviter d'endommager la batterie. Souvent, le courant de charge est maintenu constant durant la recharge des premiers 80% et ensuite, le courant diminue de façon exponentielle (Pelletier *et al.*, 2017). L'objectif et la façon de calculer le coût d'un itinéraire sont les mêmes que pour le MDVSP. Le Tableau 3.1 présente les différents types de déplacement qui sont considérés comme des déplacements haut-le-pied et donc en conséquence coûteux.

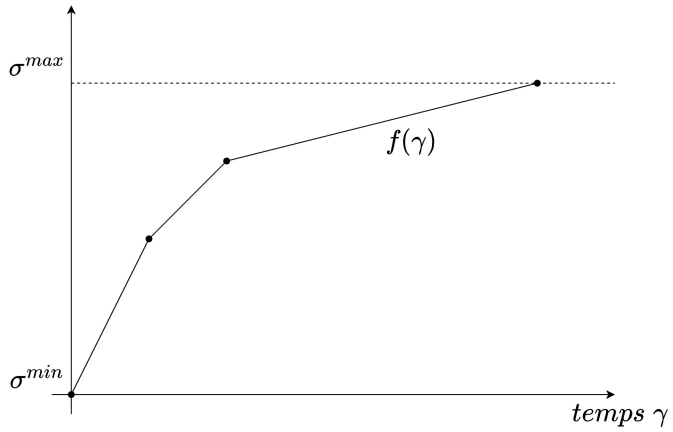


FIGURE 3.1 Fonction par morceaux de l'état de charge

TABLEAU 3.1 Types de déplacement haut-le-pied

Source	Destination(s)
Dépôt	Station de départ d'un trajet
Station d'arrivée d'un trajet	Station de départ d'un autre trajet, Dépôt, Station de recharge
Station de recharge	Station de départ d'un trajet Dépôt

3.3 Formulation mathématique du MDEVSP

Résoudre le MDEVSP consiste à définir et attribuer un itinéraire à chaque autobus, partant et revenant au même dépôt, de manière à couvrir chaque trajet exactement une fois. Les itinéraires doivent satisfaire les contraintes d'autonomie des véhicules et de connexions décrites précédemment.

On suppose qu'on connaît l'ensemble complet d'itinéraires valides et on définit Ω^k comme l'ensemble d'itinéraires valides partant de et revenant au dépôt k et $\Omega = \bigcup_k \Omega^k$ comme l'union de tous les itinéraires. Le MDEVSP peut alors s'écrire comme le programme linéaire en nombres entiers suivant. Les paramètres et variables sont décrits en détail dans le Tableau 3.2.

$$\min \quad \sum_{k\in\mathcal{K}} \sum_{s\in\Omega^k} c_s x_s \tag{3.6}$$

$$\text{s.t.} \quad \sum_{k\in K} \sum_{s\in\Omega^k} a_s^t x_s = 1 \qquad \forall t \in \mathcal{T} \tag{3.7}$$

$$\sum_{s\in\Omega^k} x_s \leq g^k \qquad \forall k \in \mathcal{K} \tag{3.8}$$

$$\sum_{k\in\mathcal{K}} \sum_{s\in\Omega^k} o_s^{p,h} x_s \leq b^h \quad \forall h \in \mathcal{H}, p \in \mathcal{P} \tag{3.9}$$

$$x_s \in \{0,1\} \qquad \forall k \in \mathcal{K}, s \in \Omega^k \tag{3.10}$$

Tableau 3.2 Description des paramètres et variables

Paramètre ou variable	Description	Domaine
$\mathcal{T}$	Ensemble des n trajets à couvrir	$\{t_1, t_2, ..., t_n\}$
c_s	Coût de l'itinéraire s	$\mathbb{R}^+$
$\mathcal{K}$	Ensemble des m dépôts	$\{k_1, k_2, ..., k_m\}$
g^k	Capacité du dépôt k	$\mathbb{N}$
Ω^k	Ensemble d'itinéraires valides partant et revenant au dépôt k	$\{s_1, s_2, ..., s_s\}$
a_s^t	Égal à 1 si l'itinéraire s couvre le trajet t, 0 sinon	{0,1}
$\mathcal{H}$	Ensemble des r stations de recharge	$\{h_1, h_2, ..., h_r\}$
b_h	Nombre de bornes de recharge disponibles à la station h	$\mathbb{N}$
$\mathcal{P}$	Ensemble des intervalles de temps utilisés pour discrétiser le temps	$\{p_1, p_2, ..., p_{\|P\|}\}$
$o_s^{p,h}$	Égal à 1 si l'itinéraire s inclut une opération de recharge à la station h dans l'intervalle de temps p, 0 sinon	{0,1}
x_s	Variable égale à 1 si l'itinéraire s est sélectionné, 0 sinon	{0,1}

L'objectif est de minimiser le coût total des itinéraires choisis (3.6), en garantissant que

chaque trajet soit couvert exactement une fois (3.7), que les capacités des dépôts (3.8) et des stations de recharge (3.9) soient respectées. Les variables de décision prennent la valeur 0 ou 1 (3.10).

3.4 Réseau de connexions

En pratique, l'ensemble des itinéraires réalisables Ω est très grand et il est souvent impossible à énumérer a priori. On représente plutôt cet ensemble d'itinéraires sous formes de réseaux de connexions (un réseau par dépôt).

Dans un réseau de connexions pour le MDVSP, chaque trajet est représenté par un nœud. Un arc existe entre le nœud t_i et le nœud t_j si et seulement si la paire de trajet (t_i, t_j) est considérée valide. On ajoute un nœud source et un nœud puits représentant le départ et l'arrivée au dépôt. Finalement, il y a un arc entre le nœud source et chaque trajet ainsi qu'un arc entre chaque trajet et le nœud puits.

Pour le MDEVSP, on utilise le même réseau que le MDVSP auquel on ajoute des nœuds et des arcs afin de représenter les activités de recharge aux différentes stations. Afin de monitorer les capacités en termes de bornes des différentes stations de recharge, on discrétise le temps. Cette discrétisation consiste à subdiviser l'horizon d'optimisation (24h par exemple) en plusieurs courts intervalles de durées égales (15 min. par exemple). On définit $P = \{p_1, p_2, ..., p_{|P|}\}$ comme l'ensemble des intervalles de durée λ considérés pour l'horizon d'optimisation. L'intervalle de temps p commence au temps D_p et termine au temps $F_p = D_p + \lambda$. Ainsi, une activité de recharge ne peut que commencer au début d'un intervalle de temps et finir à la fin d'un intervalle de temps, à moins que l'état de la batterie n'excède σ^{max}. Dans ce cas, on suppose que le véhicule occupe la borne jusqu'à la fin de l'intervalle en question. Ainsi, pour chaque station de recharge et chaque intervalle de temps, il y a un nœud d'attente et un nœud de recharge. Il y a aussi un arc entre chaque nœud *trajet* (t_i) et le premier nœud d'attente de chaque station de recharge h pouvant être atteint après avoir complété ce trajet. L'inégalité suivante est utilisée pour déterminer quel intervalle de temps peut être atteint en premier après avoir complété le trajet t_i :

$$\min_{p \in \mathcal{P}} \; D_p \geq h_i^e + \theta_{ih} \tag{3.11}$$

où θ_{ih} est le temps de déplacement entre E_i et la station de recharge h. On ajoute aussi, pour chaque nœud *trajet* t_i, un arc entre le nœud de la période d'attente la plus tardive à laquelle on peut partir pour arriver avant l'heure de départ du trajet. L'inégalité suivante est utilisée pour déterminer cette période :

$$\max_{p \in \mathcal{P}} F_p \leq h_i^s - \theta_{hi} \tag{3.12}$$

où θ_{hi} est le temps de déplacement entre la station de recharge h et S_i. Le temps d'attente à la station S_i lorsqu'un autobus quitte la station de recharge h à la fin de l'intervalle p est défini selon l'équation suivante :

$$w(p_h, t_i) = h_i^s - \theta_{hi} - F_p \tag{3.13}$$

où p_h est l'intervalle de recharge p à la station de recharge h et θ_{hi} est le temps de déplacement entre la station de recharge h et la station de départ du trajet i. Finalement, on ajoute des arcs entre les nœuds d'attente et de recharge d'intervalles successifs, entre les nœuds d'attente d'intervalles successifs et entre les nœuds de recharge d'intervalles successifs.

On présente ci-dessous une définition plus formelle du réseau de connexions pour le MDEVSP. Pour chaque dépôt $k \in K$, on définit un réseau $\mathcal{G}^k = (V^k, A^k)$ où V^k est l'ensemble des nœuds et A^k l'ensemble des arcs (voir Figure 3.2) L'ensemble des nœuds $V^k = \{o^k, d^k\} \cup \mathcal{T} \cup \mathcal{W} \cup \mathcal{C}$ contient une paire de nœuds o^k, d^k représentant le départ et l'arrivée au dépôt, un nœud par trajet, un nœud d'attente et un nœud de recharge pour chaque intervalle de temps et chaque station. Les nœuds sont décrits en détail dans le Tableau 3.3. L'ensemble $A^k = OUT \cup INT \cup INS \cup CX \cup TS \cup WW \cup WC \cup CC \cup CW \cup FS$ contient un arc par déplacement à vide possible (trajet haut-le-pied) et par action de recharge ou d'attente à une station. Les arcs sont décrits en détail dans le Tableau 3.4.

Tableau 3.3 Définitions des nœuds

Nœuds	Définition
o^k	Nœud représentant le départ du dépôt
d^k	Nœud représentant l'arrivée au dépôt
$\mathcal{T}$	Ensemble des nœuds de trajet
$\mathcal{W} = \left\{ (w_p^h)_{\substack{p \in \mathcal{P} \\ h \in \mathcal{H}}} \right\}$	Ensemble des noeuds d'attente
$\mathcal{C} = \left\{ (c_p^h)_{\substack{p \in \mathcal{P} \\ h \in \mathcal{H}}} \right\}$	Ensemble des noeuds de recharge

Les coûts des arcs sont calculés en fonction du temps de déplacement à vide et du temps

Tableau 3.4 Définitions des arcs

Ensemble	Arcs	Définition
OUT	$(o^k, t_i)\ \forall\, t_i \in \mathcal{T}$	Un arc à partir du dépôt vers chaque trajet
INT	$(t_i, d^k)\ \forall\, t_i \in \mathcal{T}$	Un arc vers le dépôt à partir de chaque trajet
INS	$(w^h_{\lvert P\rvert}, d^k)\ \forall\, h \in \mathcal{H}$	Un arc vers le dépôt à partir du dernier nœud d'attente de chaque station de recharge
CX	$(t_i, t_j)\ \forall\, t_i, t_j \in \mathcal{T}$ t.q. (t_i, t_j) satisfait (3.1) - (3.2)	Un arc de connexion entre chaque paire de trajets valide
TS	(t_i, w^h_p) $\forall\, t_i \in \mathcal{T},\ \forall\, h \in \mathcal{H}$ où p est défini par (3.11)	Un arc entre chaque trajet et le nœud d'attente le plus tôt pouvant être atteint de chaque station
WW	(w^h_p, w^h_{p+1}) $\forall\, p \in \mathcal{P},\ \forall\, h \in \mathcal{H}$	Un arc entre chaque paire de nœuds d'attente consécutifs à chaque station
WC	(w^h_p, c^h_{p+1}) $\forall\, p \in \mathcal{P},\ \forall\, h \in \mathcal{H}$	Un arc entre chaque nœud d'attente et le nœud de recharge de la période suivante à chaque station
CW	(c^h_p, w^h_{p+1}) $\forall\, p \in \mathcal{P},\ \forall\, h \in \mathcal{H}$	Un arc entre chaque nœud de recharge et le nœud d'attente de la période suivante à chaque station
CC	(c^h_p, c^h_{p+1}) $\forall\, p \in \mathcal{P},\ \forall\, h \in \mathcal{H}$	Un arc entre chaque paire de nœuds de recharge consécutifs à chaque station.
FS	(w^h_p, t_i) $\forall\, t_i \in \mathcal{T},\ \forall\, h \in \mathcal{H}$ où $p =$ est défini par (3.12)	Un arc vers chaque trajet à partir du nœud d'attente le plus tard pouvant être quitté de chaque station

d'attente. De plus, on ajoute un coût fixe d'utilisation d'autobus à chaque arc sortant du nœud o^k. Le coût d'un arc entre les nœuds i et j (c_{ij}) est calculé selon la formule suivante :

$$c_{ij} = \begin{cases} c_d d_{(ij)} + c_a w_{(ij)} + c_f & \text{si } a = o^k \\ c_d d_{(ij)} + c_a w_{(ij)} & \text{sinon} \end{cases} \tag{3.14}$$

où $d_{(ij)}$ est le temps de déplacement requis pour passer du nœud i au nœud j et $w_{(ij)}$ est le temps d'attente lorsque le nœud j est visité après le nœud i. Les coûts des arcs sont présentés en détail dans le Tableau 3.5.

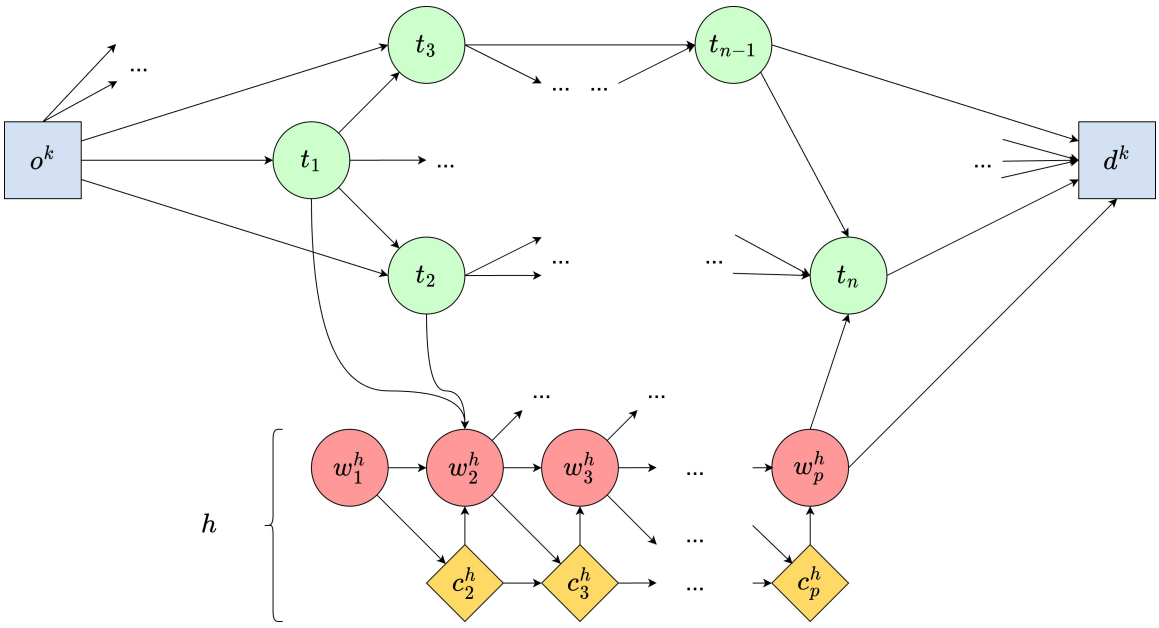


FIGURE 3.2 Réseau de connexions du sous-problème associé au dépôt k

TABLEAU 3.5 Coûts et consommations d'énergie des différents types d'arcs

Arcs	Énergie	Coût
OUT	$e_d d(o^k, S_i)$	$c_d d(o^k, S_i) + c_f$
IN	$e_d \left[d(S_i, E_i) + d(E_i, d^k) \right]$	$c_d d(E_i, d^k)$
CX	$e_d \left[d(S_i, E_i) + d(E_i, S_j) \right] + e_a w(t_i, t_j)$	$c_d d(E_i, S_j) + c_a w(t_i, t_j)$
TS	$e_d \left[d(S_i, E_i) + d(E_i, h) \right]$	$c_d d(E_i, h)$
WW	0	0
WC	$-f(\sigma, \gamma)$	0
CW	0	0
CC	$-f(\sigma, \gamma)$	0
FS	$e_t d(h, S_i) + e_a w(w_p^h, t_i)$	$c_d d(h, S_i) + c_a w(w_p^h, t_i)$

3.5 Algorithme de génération de colonnes

On présente maintenant l'algorithme de génération de colonnes utilisé afin de résoudre le MDEVSP. Cet algorithme permet de résoudre itérativement ce problème en générant des itinéraires valides à partir du réseau de connexions présenté précédemment.

Le problème maître en nombres entiers (PM^e) est défini selon (3.6)-(3.10). On obtient sa relaxation linéaire (PM) en relaxant les contraintes d'intégralité (3.10). Pour résoudre le PM, on commence par résoudre un problème maître restreint (PMR) qui contient un sous-ensemble d'itinéraires valides beaucoup plus petit $\Omega_r \subset \Omega$. Ensuite, lorsque ce PMR est résolu à l'optimalité, on obtient une variable duale π_t pour chaque contrainte (3.7), une variable duale λ_k pour chaque contrainte (3.8) et une variable duale μ_{hp} pour chaque contrainte (3.9). On peut ensuite calculer les coûts réduits $\bar{c}_s$ des itinéraires valides $s \in \Omega^k$ relativement au PMR selon l'expression suivante :

$$\bar{c}_s = c_s - \sum_{t \in T} \pi_t a_s^t - \sum_{h \in H} \sum_{p \in P} \mu_{hp} b_s^{ph} - \lambda_k \tag{3.15}$$

Pour espérer réduire la valeur de la Fonction Objectif (3.6), il faut trouver un itinéraire (une colonne) avec un coût réduit inférieur à 0 ($s \in \Omega$ t.q. $\bar{c}_s < 0$). S'il existe un ou plusieurs itinéraires avec un coût réduit négatif, on peut les ajouter à Ω_r, résoudre le nouveau PMR et potentiellement diminuer la valeur de l'Objectif (3.6). En revanche, si tous les itinéraires présentent des coûts réduits positifs ou nuls, cela implique que la solution du PMR obtenue est la solution optimale au PM.

Afin de déterminer s'il existe des itinéraires $s \in \Omega$ ayant un coût réduit négatif, il faut résoudre un problème de plus court chemin pour chaque réseau de connexions $\mathcal{G}^k$. Pour cela, il convient de modifier les coûts des arcs de sorte que la somme des coûts des arcs modifiés d'un itinéraire soit égale au coût réduit de cet itinéraire. Ensuite, en résolvant un problème de plus court chemin entre o^k et d^k pour chaque $k \in \mathcal{K}$, on est en mesure de trouver l'itinéraire (la colonne) à coût réduit minimal. On procède de la façon suivante pour modifier les coûts des arcs :

- On soustrait λ_k au coût de chaque arc sortant du nœud o^k
- On soustrait π_t au coût de chaque arc sortant du nœud t, $\forall\, t \in \mathcal{T}$
- On soustrait μ_{hp} au coût de chaque arc sortant d'un nœud de recharge w_p^h, $\forall\, p \in \mathcal{P}, h \in \mathcal{H}$

3.5.1 Définition du sous-problème

Trouver un chemin de coût réduit négatif implique la résolution d'un problème de plus court chemin avec contraintes de ressources, par dépôt. Ce problème, appelé sous-problème (SP), est utilisé pour générer de nouveaux itinéraires valides.

Le SP consiste à trouver le plus court chemin entre les nœuds o^k et d^k, en tenant compte de contraintes de ressources. Plus précisément, chaque arc des graphes $\mathcal{G}^k$ consomme ou produit une certaine quantité de ressources, et la somme des ressources utilisées est contrainte par une limite inférieure et supérieure à chaque nœud du graphe. Dans le contexte du MDEVSP, chaque arc consomme ou produit une certaine quantité d'énergie, et la somme totale de l'énergie consommée doit se situer entre σ^{min} et σ^{max} en tout temps. La consommation d'énergie de chaque arc est définie dans le Tableau 3.5 où e_d et e_a sont les taux de consommation d'énergie par unité de temps de déplacement et d'attente, respectivement et f est une fonction de charge linéaire par morceau dépendante de l'état de charge actuel (σ) et du temps de recharge (γ). On utilise aussi deux autres ressources (r_{rch} et r_{not_rch}) pour forcer une et une seule recharge consécutive lors d'une visite à une station de recharge.

Plus précisément, la ressource r_{rch} augmente de 1 uniquement lorsqu'un véhicule passe d'un nœud d'attente à un nœud de recharge (arc WC) et diminue de 1 uniquement lorsque le véhicule quitte la station de recharge (arc FS). Aux nœuds $\mathcal{W}$, la ressource r_{rch} est contrainte entre 0 et 1 ; aux nœuds $\mathcal{C}$, elle doit être égale à 1 ; et aux nœuds $\mathcal{T}$, elle doit être égale à 0. Ainsi, un véhicule ne peut pas effectuer deux recharges lors d'une même visite à une station de recharge. La ressource r_{not_rch} augmente de 1 uniquement lorsqu'un véhicule arrive à une station de recharge (arc TS) et diminue de 1 uniquement lorsqu'il passe à un nœud de recharge (arc WC). Aux nœuds $\mathcal{W}$ et $\mathcal{C}$, la ressource est contrainte entre 0 et 1 tandis qu'aux nœuds $\mathcal{T}$, elle doit être égale à 0. Ainsi, un véhicule est forcé à visiter au moins un nœud de recharge lorsqu'il visite une station.

On utilise un algorithme d'étiquetage dynamique pour résoudre le SP. Un tel algorithme représente les chemins partiels débutant au noeud o^k à l'aide d'étiquettes. À partir d'une étiquette initiale associée au noeud o^k, des chemins partiels de plus en plus longs sont construits en prolongeant cette étiquette et ces successeurs. Une règle de dominance qui élimine des étiquettes ne pouvant mener à un chemin optimal au noeud d^k est appliquée pour éviter d'énumérer tous les chemins réalisables. Pour plus de détails, on réfère le lecteur à Irnich et Desaulniers (2005).

3.5.2 Décisions de branchement

La solution optimale du PM peut ne pas être entière puisque les contraintes d'intégrité sont relaxées. Celle-ci peut permettre l'utilisation d'un nombre fractionnaire d'autobus et/ou d'itinéraires. Pour obtenir une solution entière optimale, la méthode du *branch and price* est souvent utilisée. Cette méthode crée des nœuds-fils au besoin en imposant des décisions de branchement sur les variables fractionnaires, puis résout la relaxation linéaire associée à ces nœuds-fils en utilisant à nouveau la génération de colonnes. Ce processus se répète jusqu'à ce qu'une solution entière soit trouvée et prouvée optimale. On peut décider, par exemple, de brancher sur le nombre de véhicules utilisés par dépôt, le flot d'un arc ou le dépôt couvrant ou pas un trajet.

Pour des instances de grande taille, on peut aussi utiliser un branchement heuristique, tel que le branchement en profondeur sans retour en arrière. Dans ce cas, à chaque nœud de l'arbre de branchement, une ou plusieurs variables sont fixées à une valeur entière. Par exemple, si $x_1 = 0.9$ dans la solution du PM, on force $x_1 = 1$, on génère un nouveau PM et on résout ce nouveau PM. Il est important de noter que ces techniques heuristiques ne garantissent pas de trouver la solution optimale, mais elles peuvent souvent fournir des solutions de bonne qualité en un temps raisonnable.

3.6 Réduction de la dégénérescence par pertubation

Les instances du MDEVSP sont souvent très dégénérées, en particulier lorsqu'il y a beaucoup de trajets à couvrir. La dégénérescence se produit lorsqu'il existe plusieurs solutions de base réalisables qui correspondent à la même valeur de la fonction objectif. Cela peut rendre plus difficile le choix de la colonne à ajouter au PMR et causer des inefficacités dans l'algorithme du simplexe, utilisé pour résoudre le PM. En effet, durant plusieurs itérations, des colonnes peuvent être insérées dans la base sans jamais améliorer la fonction objectif. Il est même possible de tomber dans un cycle et de ne jamais être en mesure de sortir d'un minimum local.

Il existe plusieurs techniques pour contrer ces effets indésirables de la dégénérescence (voir Section 2.1.3). Dans cette recherche, on utilise une technique de perturbation qui permet de couvrir les trajets légèrement plus ou légèrement moins qu'une fois. Cette approche consiste à modifier le PM^e (3.6)-(3.10) en ajoutant des variables de perturbation η_t^+ et η_t^- pour chaque trajet à couvrir. Ces nouvelles variables sont bornées supérieurement par ϵ_t^+ et ϵ_t^- respectivement qui sont choisies aléatoirement comme variables uniformes dans l'intervalle $[0, 10^{-4}]$. Le PM^e devient donc :

$$\min \quad \sum_{k\in\mathcal{K}} \sum_{s\in\Omega^k} c_s x_s \tag{3.16}$$

$$\text{s.t.} \quad \sum_{k\in K} \sum_{s\in\Omega^k} a_s^t x_s + \eta_t^+ - \eta_t^- = 1 \quad \forall t \in \mathcal{T} \tag{3.17}$$

$$(3.8) - (3.10)$$

$$0 \leq \eta_t^+ \leq \epsilon_t^+ \qquad \forall t \in \mathcal{T} \tag{3.18}$$

$$0 \leq \eta_t^- \leq \epsilon_t^- \qquad \forall t \in \mathcal{T} \tag{3.19}$$

Cette version du PM^e subit moins de dégénérescence car les solutions de base comprennent de nombreuses variables η_t^+ et η_t^- prenant de petites valeurs positives au lieu de variables $x_s = 0$. Cependant, même lorsque cette technique de perturbation est utilisée, certaines instances restent sujettes à la dégénérescence et la résolution du PM peut prendre beaucoup de temps. Dans ce mémoire, on propose une approche qui tente d'accélérer la résolution du PM d'instances présentant de la dégénérescence même lorsque cette technique de perturbation est utilisée. Comme mentionné dans le chapitre 2, il existe plusieurs techniques pour contrer ces effets indésirables. Dans le cadre de cette recherche, on propose une technique basée sur les inégalités duales.

CHAPITRE 4 PRÉDICTION D'INÉGALITÉS DUALES

Ce chapitre est divisé en 5 sections. On commence par définir les inégalités duales et expliquer comment les introduire dans le primal. Ensuite, on présente un modèle d'apprentissage supervisé qui prédit ces inégalités duales et les représente sous forme de graphe. Par la suite, on propose une méthode permettant de réduire les erreurs potentielles de prédiction. Ensuite, on présente une méthode pour extraire un sous-ensemble d'inégalités prédites à ajouter au primal. Enfin, on introduit un processus de réparation des solutions qui peut être utilisé lorsque des inégalités duales invalides sont introduites.

4.1 Inégalités duales

Une façon de réduire l'espace de recherche d'un programme linéaire consiste à ajouter des inégalités valides dans son dual (inégalités duales). Pour obtenir le dual d'un programme linéaire en forme standard, on transforme les contraintes du primal en variables de décision et les variables du primal en contraintes. À partir de ce programme dual, on peut déduire de nouvelles inégalités duales. Ces inégalités permettent de réduire l'espace dual tout en préservant au moins une solution duale optimale. La réduction de l'espace dual a pour effet d'accélérer la recherche d'une solution duale optimale, et d'améliorer la convergence de la génération de colonnes dans notre cas.

Comme mentionné précédemment, chaque contrainte du PM a une variable duale correspondante : une variable duale π_t pour chaque contrainte (3.7), une variable duale λ_k pour chaque contrainte (3.8) et une variable duale μ_{hp} pour chaque contrainte (3.9). Lorsqu'on obtient une solution optimale au PM, on obtient également les valeurs de ces variables duales. On utilise $\hat{\pi}_t$, $\hat{\lambda}_k$ et $\hat{\mu}_h p$ pour faire référence aux valeurs des variables duales dans la solution optimale du PM. Intuitivement, la variable duale π_t représente l'augmentation de la valeur de la fonction objectif qui résulte de l'augmentation d'une unité de la partie de droite de la contrainte de couverture associée au trajet t. En d'autres termes, la valeur de la variable duale représente le coût supplémentaire à payer dans la solution optimale pour couvrir le trajet t. Ainsi, si l'on connaît les valeurs des variables duales π_t pour chaque contrainte de couverture, on peut ajouter des inégalités duales valides qui permettent de réduire davantage l'espace de recherche du dual.

Supposons que la valeur de la variable duale π_i ($\hat{\pi}_i$) dans la solution est supérieure ou égale à la valeur de la variable duale π_j ($\hat{\pi}_j$) dans cette même solution, alors l'inégalité duale

$\pi_i \geq \pi_j - \phi_{ij}$ où $\phi_{ij} \geq 0$ est toujours valide et peut être introduite dans le primal. Cette inégalité duale est représentée dans le primal par une nouvelle variable w_{ij}. On définit le nouveau PM avec inégalités duales (PM^{ID}) de la façon suivante :

$$\min \quad \sum_{k \in \mathcal{K}} \sum_{s \in \Omega^k} c_s x_s + \phi_{ij} w_{ij} \tag{4.1}$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{s \in \Omega^k} a_s^t x_s = 1 \qquad \forall t \in \mathcal{T} \setminus \{i, j\} \tag{4.2}$$

$$\sum_{k \in K} \sum_{s \in \Omega^k} a_s^i x_s - w_{ij} = 1 \tag{4.3}$$

$$\sum_{k \in K} \sum_{s \in \Omega^k} a_s^j x_s + w_{ij} = 1 \tag{4.4}$$

$$(3.8) - (3.9)$$

où ϕ_{ij} devient un coefficient de coût associé à la nouvelle variable w_{ij}. On remarque que l'ajout d'une nouvelle variable au primal a pour effet de modifier les contraintes de couverture des trajets impliqués dans l'inégalité duale (contrainte de couverture des trajets i et j). De plus, si le coefficient ϕ_{ij} introduit dans la fonction objectif prend une valeur supérieure à zéro, w_{ij} prendra une valeur nulle dans la solution du PM^{ID}. En effet, on a deux cas :

- Si $\pi_i > \pi_j$ (couvrir le trajet i coûte plus cher que couvrir le trajet j), alors $w_{ij} = 0$ dans la solution. En effet, si w_{ij} prend une valeur positive dans la solution, le trajet i doit être couvert $1 + w_{ij}$ fois et le trajet j, $1 - w_{ij}$ fois, ce qui a comme conséquence d'augmenter la valeur de la fonction objectif puisque $\pi_i > \pi_j$.
- Si $\pi_i = \pi_j$ (couvrir le trajet i coûte aussi cher que couvrir le trajet j) et $\phi_{ij} > 0$, alors couvrir chaque trajet une fois ($w_{ij} = 0$) est préférable à couvrir un trajet $1 + w_{ij}$ fois et l'autre $1 - w_{ij}$ fois ($w_{ij} > 0$). En effet, si w_{ij} prend une valeur positive dans la solution, la fonction objectif augmente de ϕ_{ij}.

De plus, puisqu'on prédit des inégalités duales du type $\pi_i \geq \pi_j$ et que celles-ci pourraient être erronées, i.e., invalides, l'ajout d'un coefficient $\phi_{ij} > 0$ permet de combler certaines imprécisions.

4.2 Présentation du modèle d'apprentissage supervisé

Dans le cadre de ce mémoire, on s'intéresse uniquement aux inégalités duales impliquant deux variables duales ($\pi_i \geq \pi_j$). En effet, ce genre d'inégalité est plus facile à prédire à l'aide de

l'apprentissage machine que des inégalités impliquant 3 variables duales ou plus. On présente dans cette section un modèle d'apprentissage automatique qui tente d'apprendre à prédire ces inégalités. Ce modèle d'apprentissage utilise une modélisation du problème similaire à celle utilisée dans le SP afin d'apprendre à prédire des inégalités duales.

La Figure 4.1 présente les deux blocs qui composent ce modèle d'apprentissage. Le premier bloc consiste en un réseau de neurones à convolution qui prend en entrée un graphe $\mathcal{Q}$ représentant le réseau de connexions avec des vecteurs de caractéristiques propres à chaque nœud et arête, et génère un nouveau graphe $\mathcal{Q}'$ qui a la même structure que $\mathcal{Q}$, mais dont les nœuds et les arêtes ont de nouveaux vecteurs de caractéristiques appris. Le deuxième bloc du modèle d'apprentissage prend en entrée le nouveau graphe $\mathcal{Q}'$ et prédit le sens de l'inégalité pour chaque paire de variables duales $(\pi_i, \pi_j)\ \forall\ \pi_i, \pi_j \in \Pi$. On explique en détails ces deux parties du modèle dans les prochaines sections.

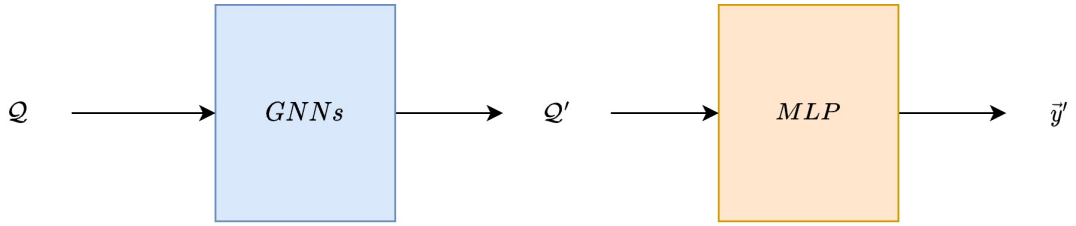


FIGURE 4.1 Vue d'ensemble du modèle d'apprentissage automatique

4.2.1 Bloc 1 : GNN

Les GNNs sont conçus pour traiter des données structurées sous forme de graphe et extraire des informations à partir d'un graphe initial en utilisant des opérations de propagation de message afin de diffuser des informations entre des nœuds voisins. Plus précisément, le bloc GNNs utilise une variante du graphe de connexions utilisé pour résoudre le SP présenté à la Section 3.4 et tire avantage des connexions entre les nœuds pour générer de nouvelles caractéristiques propres aux nœuds.

Le bloc GNNs est composé de plusieurs couches d'attention EGATConv (Kaminski *et al.*, 2021). EGATConv est un type de couche d'attention utilisé par les GNNs, inspirée des couches d'attention GATConv (Veličković *et al.*, 2017), qui propagent de l'information entre nœuds voisins, mais aussi avec les arêtes connectées aux nœuds. Chaque couche prend en entrée un graphe avec des caractéristiques propres à chaque nœud et à chaque arc et génère un nouveau graphe ayant la même structure, mais ayant des caractéristiques différentes pour chaque nœud et arc. Le graphe généré à la sortie d'une couche d'EGATConv est utilisé comme

graphe d'entrée dans la couche suivante. Les caractéristiques des nœuds du graphe généré par la dernière couche sont utilisées dans le deuxième bloc du modèle afin de prédire les sens des inégalités duales. Chaque nœud (resp. arc) du graphe possède un vecteur de caractéristiques initial $\vec{h}$ (resp. $\vec{e}$). Le Tableau 4.1 (resp. 4.2) décrit les caractéristiques du vecteur $\vec{h}$ (resp. $\vec{e}$). Ces caractéristiques sont normalisées afin d'augmenter la capacité d'apprentissage. On présente maintenant comment une couche d'EGATConv propage les informations des nœuds et des arcs et comment elle apprend de nouvelles caractéristiques.

Tableau 4.1 Définitions des caractéristiques des nœuds

Nom	Définition
$\{o, k, n, w, c\}$	Encodage *one-hot* du type de nœud. o : nœud source, k : nœud puits, n : nœud de trajet, w : nœud d'attente, c : nœud de recharge.
$start\_time$	Heure de départ du trajet en minutes si $n = 1$, 0 sinon.
end_time	Heure d'arrivée du trajet en minutes si $n = 1$, 0 sinon.
$trip\_time$	Durée du trajet en minutes $(h_t^e - h_t^s)$ si $n = 1$, 0 sinon.
$nDep10_t$	Nombre de trajets partant de S_t dans les 10 minutes suivant h_t^s si $n = 1$, 0 sinon.
$nbDep10id_t$	Nombre de trajets partant de S_t dans les 10 minutes suivant h_t^s et se rendant à E_t si $n = 1$, 0 sinon.
$nFin10_t$	Nombre de trajets arrivant à E_t dans les 10 minutes précédant h_t^e si $n = 1$, 0 sinon.
$nbFin10id_t$	Nombre de trajets arrivant à E_t dans les 10 minutes précédant h_t^e et qui partent de S_t si $n = 1$, 0 sinon.
$s\{*\}$	Encodage *one-hot* pour la station de départ du trajet. Une colonne $s\{X\}$ est créée pour chaque S_t différent. $s\{X\} = 1$ si le trajet t a comme origine la station X, 0 sinon.
$e\{*\}$	Encodage *one-hot* pour la station d'arrivée du trajet. Une colonne $e\{X\}$ est créée pour chaque E_t différent. $e\{X\} = 1$ si le trajet t a comme destination la station X, 0 sinon.

Tableau 4.2 Description des caractéristiques des arcs

Nom	Définition
cost	Coût de l'arc défini en (3.14)
energy	Énergie consommée ou rechargée sur l'arc
travel_time	Temps de déplacement
waiting_time	Temps d'attente
rg	Rang relatif de l'arc parmi les autres arcs partant du même nœud, en se basant sur l'heure de départ du trajet (ou de l'intervalle de recharge) à la destination de l'arc
rg_id	Rang relatif de l'arc parmi les autres arcs partant du même nœud et se rendant au même nœud, en se basant sur l'heure de départ du trajet (ou de l'intervalle de recharge) à la destination de l'arc

On définit $\boldsymbol{h}^l = \left\{\vec{h}_1^l, \vec{h}_2^l, ..., \vec{h}_N^l\right\}, \vec{h}_i^l \in \mathbb{R}^{F^l}$, comme l'ensemble des vecteurs de nœuds en entrée de la couche l et $\boldsymbol{e}^l = \left\{\vec{e}_{11}^l, \vec{e}_{12}^l, ..., \vec{e}_{NN}^l\right\}, \vec{e}_{ij}^l \in \mathbb{R}^{E^l}$, comme l'ensemble des vecteurs d'arcs en entrée de la couche l où N est le nombre de nœuds, F^l le nombre de caractéristiques par nœud en entrée de la couche l et E^l le nombre de caractéristiques par arc en entrée de la couche l. On note que lorsque $l = 1$, les vecteurs représentent les caractéristiques initiales.

La couche l génère un nouvel ensemble de nœuds et d'arcs ayant une cardinalité potentiellement différente ($F^{(l+1)}$ et $E^{(l+1)}$ respectivement), $\boldsymbol{h}^{(l+1)} = \left\{\vec{h}_1^{(l+1)}, \vec{h}_2^{(l+1)}, ..., \vec{h}_N^{(l+1)}, \right\}, \vec{h}_i^{(l+1)} \in \mathbb{R}^{F^{(l+1)}}$ et $\boldsymbol{e}^{(l+1)} = \left\{\vec{e}_1^{(l+1)}, \vec{e}_2^{(l+1)}, ..., \vec{e}_M^{(l+1)}, \right\}, \vec{e}_i^{(l+1)} \in \mathbb{R}^{E^{(l+1)}}$, où $\vec{h}_i^{(l+1)}$ et $\vec{e}_{ij}^{(l+1)}$ sont calculés selon les équations suivantes :

$$\vec{h}_i^{(l+1)} = \sum_{j \in \mathcal{N}_i} \alpha_{ij} \boldsymbol{W} \vec{h}_j^l + \vec{b} \tag{4.5}$$

$$\alpha_{ij} = \text{softmax}_j(u_{ij}) = \frac{\exp(u_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(u_{ik})} \tag{4.6}$$

$$u_{ij} = \vec{f}^{\intercal} \vec{e}_{ij}^{(l+1)} \tag{4.7}$$

$$\vec{e}_{ij}^{(l+1)} = \text{LeakyRELU}\left(\boldsymbol{A}\left[\vec{h}_i^l \| \vec{e}_{ij}^l \| \vec{h}_j^{\,l}\right]\right) \tag{4.8}$$

Dans ces expressions, on a :

— une matrice de poids $\boldsymbol{W} \in \mathbb{R}^{F^{(l+1)} \times F^l}$;
— un vecteur de biais $\vec{b} \in \mathbb{R}^{F^{(l+1)}}$;
— un coefficient d'attention u_{ij} et ce coefficient normalisé α_{ij} ;
— un vecteur de poids $\vec{f} \in \mathbb{R}^{E^{(l+1)}}$;
— une matrice de poids $\boldsymbol{A} \in \mathbb{R}^{(E^{(l+1)}) \times (2F^l + E^l)}$;
— une fonction d'activation LeakyReLU $\Big(f(x) = \texttt{max}(0.1x, x)\Big)$.

La fonction d'activation LeakyRELU (Maas *et al.*, 2013) est fondée sur la fonction ReLU, mais elle introduit une pente non nulle pour les valeurs négatives, afin d'éviter les problèmes de saturation. Le coefficient d'attention u_{ij} représente l'importance des caractéristiques du nœud j et de l'arc e_{ij} pour le nœud i. Ces coefficients d'attention normalisés sont ensuite utilisés pour calculer une combinaison linéaire des entités correspondantes ($\vec{h}_j^l \; \forall \; j \in \mathcal{N}_i$).

Pour stabiliser le processus d'apprentissage, on utilise un mécanisme d'attention multi-têtes. Plus précisément, K mécanismes d'attention indépendants et initialisés aléatoirement exécutent la transformation (4.5) et les caractéristiques résultantes sont concaténées. On obtient alors :

$$\vec{h}_i^{(l+1)} = \Big\Vert_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \boldsymbol{W}^k \vec{h}_j^l + \vec{b} \tag{4.9}$$

$$\vec{e}_{ij}^{(l+1)} = \Big\Vert_{k=1}^{K} \text{LeakyRELU}\left(\boldsymbol{A}^k\left[\vec{h}_i^l \| \vec{e}_{ij}^l \| \vec{h}_j^{\,l}\right]\right) \tag{4.10}$$

où $\Big\Vert$ est l'opérateur de concaténation, α_{ij}^k est le coefficient d'attention normalisé calculé par le mécanisme d'attention k et $\boldsymbol{W}^k$ et $\boldsymbol{A}^k$ sont les matrices de poids du mécanisme d'attention k. Il est important de noter que les vecteurs des nœuds et des arcs en sortie auront $KF^{(l+1)}$ et $KE^{(l+1)}$ caractéristiques, respectivement. À la sortie de chaque couche, en plus de concaténer les vecteurs générés par chaque mécanisme d'attention, on concatène les vecteurs de caractéristiques initiaux. On a alors :

$$\vec{h}_i^{(l+1)} = \left[\vec{h}_i^{(l+1)} \| \vec{h}_i^0\right] \tag{4.11}$$

$$\vec{e}_{ij}^{(l+1)} = \left[\vec{e}_{ij}^{(l+1)} \| \vec{e}_{ij}^{0}\right] \tag{4.12}$$

Finalement, à la sortie de la dernière couche d'EGATConv, on utilise la moyenne des K mécanismes d'attention pour générer les vecteurs de caractéristiques finaux $\boldsymbol{h}^f$ et $\boldsymbol{e}^f$. Ce sont ces vecteurs qui sont utilisés comme entrée pour la deuxième partie du modèle d'apprentissage. La Figure 4.2 représente le premier bloc du modèle tel que décrit en (4.5)-(4.12).

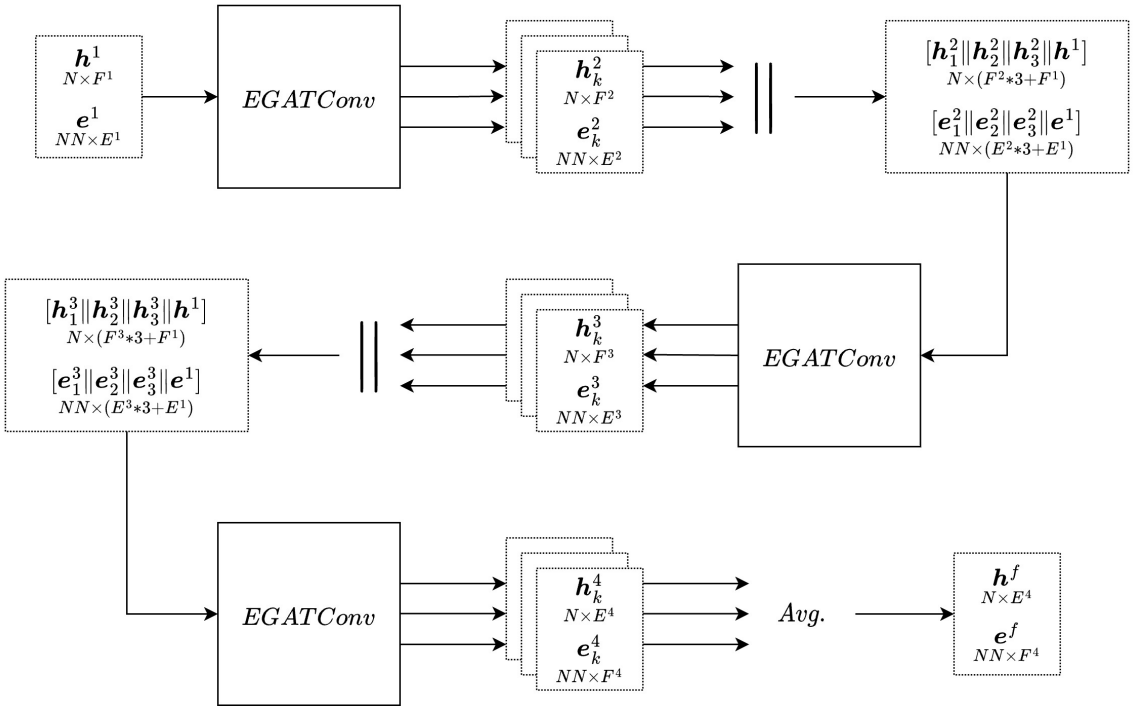


FIGURE 4.2 Structure du bloc GNN

4.2.2 Bloc 2 : Perceptron multicouches

Dans le deuxième bloc du modèle d'apprentissage, on utilise les nouveaux vecteurs de caractéristiques des nœuds générés par le bloc GNN. On utilise un perceptron multicouches qui prédit pour chaque paire de trajets $(t_i, t_j)\ \forall\ t_i, t_j \in \mathcal{T}$ le sens de l'inégalité entre la paire de variables duales (π_i, π_j) associées aux trajets correspondants. Comme on cherche à prédire uniquement des inégalités sur les variables duales associées aux trajets, on ne conserve que les vecteurs de caractéristiques de ces nœuds. Ensuite, pour chaque paire de trajets $(t_i, t_j)\ \forall\ t_i, t_j \in \mathcal{T}$, on génère un nouveau vecteur $\vec{d}_{ij}$ en concaténant les vecteurs $\vec{h}_i^f$ et $\vec{h}_j^f$. On définit cet ensemble de nouveaux vecteurs de la façon suivante :

$$\begin{aligned}\boldsymbol{d} &= \left\{ \left[\vec{h}_1^f \| \vec{h}_1^f\right], \left[\vec{h}_1^f \| \vec{h}_2^f\right], ..., \left[\vec{h}_1^f \| \vec{h}_N^3\right], ..., \left[\vec{h}_N^3 \| \vec{h}_1^3\right], ..., \left[\vec{h}_N^f \| \vec{h}_N^f\right] \right\} \\ &= \left\{ \left[\vec{h}_i^f \| \vec{h}_j^f\right] \,\middle|\, i, j \in N \right\}\end{aligned} \tag{4.13}$$

où $\vec{h}_i^f$ est le vecteur de caractéristiques du nœud i à la sortie de la dernière couche d'EGAT-Conv. Pour simplifier la notation, on définit $\vec{d}_{ij}$ comme étant la concaténation de $\vec{h}_i^f$ et $\vec{h}_j^f$ $\left(\left[\vec{h}_i^f \| \vec{h}_j^f\right]\right)$. On décide de conserver les vecteurs $\vec{d}_{ij}$ où $i = j$ pour la phase d'entraînement.

On utilise ensuite un perceptron multicouches avec L couches afin de prédire la relation entre les variables duales de chaque paire de trajets. Chaque vecteur de caractéristiques $\vec{d}_{ij}$ est passé à travers chaque couche. La première couche du perceptron $l = 1$ correspond aux caractéristiques initiales du vecteur. Les autres couches intermédiaires $l \in [2, L]$ du perceptron transforment le vecteur reçu en entrée en une représentation intermédiaire et applique une fonction d'activation non-linéaire. Le vecteur généré par la couche intermédiaire l est ensuite utilisée en entrée par la couche intermédiaire $l + 1$. La dernière couche $l = L$ représente la prédiction du modèle. On définit $\vec{s}_{ij}^{\,l} \in \mathbb{R}^{H^l}$ comme le vecteur d'entrée de la couche l où H^l est le nombre de caractéristiques du vecteur d'entrée de la couche l. Le vecteur intermédiaire généré par la couche $l \in [2, L]$ est défini ainsi :

$$\vec{s}_{ij}^{\,(l+1)} = \phi^l \left(\vec{s}_{ij}^{\,l} \boldsymbol{W}^{l\intercal} + \vec{p}^{\,l}\right) \tag{4.14}$$

où $\boldsymbol{W} \in \mathbb{R}^{H^{(l+1)} \times H^l}$ est la matrice de poids de la couche l, $\vec{p} \in \mathbb{R}^{(l+1)}$ est le vecteur de biais de la couche l et ϕ^l est la fonction d'activation non-linéaire utilisée à la couche l. On note que $\vec{s}_{ij}^{\,1} = \vec{d}_{ij}$, qu'on utilise une fonction d'activation ReLU lorsque $1 < l < L$ et sigmoïde lorsque $l = L$. Finalement, $s_{ij}^L = \hat{y}_{ij} \in [0, 1]$ représente la prédiction sur le sens de l'inégalité duale entre les variables duales associées aux trajets i et j. En effet, comme on utilise une fonction d'activation sigmoïde pour la dernière couche, $\hat{y}_{ij}$ est une valeur entre 0 et 1 inclusivement. En d'autres termes, $\hat{y}_{ij}$ est une estimation de la probabilité que l'inégalité $\pi_i \geq \pi_j$ soit valide. On utilise une fonction de perte d'entropie croisée binaire définie de la façon suivante :

$$l(\vec{\hat{y}}, \vec{y}) = LO = \texttt{mean}\left(\{l_{11}, l_{12}, l_{13}, ..., l_{NN}\}\right) \tag{4.15}$$

où NN est le nombre de paires de trajets différents. On définit l_{ij} de la façon suivante :

$$l_{ij} = y_{ij} \cdot \log \sigma(\hat{y}_{ij}) + (1 - y_{ij}) \cdot \log\left(1 - \sigma(\hat{y}_{ij})\right) \tag{4.16}$$

où $\sigma()$ est une fonction sigmoïde, $\hat{y}_{ij}$ est la prédiction du modèle et y_{ij} est égale à 1 si $\hat{\pi}_i \geq \hat{\pi}_j$, 0 sinon.

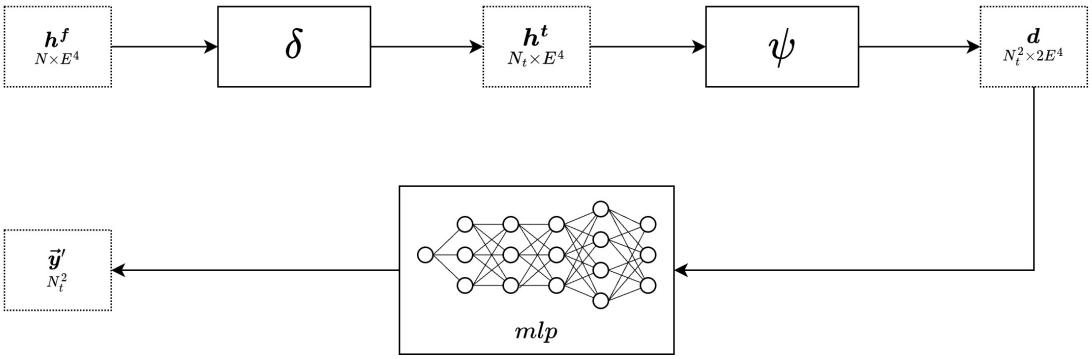


FIGURE 4.3 Structure du bloc MLP

La Figure 4.3 présente la structure du deuxième bloc du modèle. $\boldsymbol{h}^f$ représente l'ensemble des vecteurs de caractéristiques des nœuds générés par le bloc GNN. δ est une fonction qui filtre les nœuds et qui ne conserve que les nœuds représentant des trajets. ψ est une fonction qui génère toutes les paires de trajets possibles et qui concatène les vecteurs correspondants. mlp est un perceptron multicouches qui génère le vecteur $\vec{\hat{y}}$ contenant les prédictions $\hat{y}_{ij}$ pour chaque paire de variables duales (π_i, π_j) $\forall \pi_i, \pi_j \in \mathcal{T}$. Pour chaque paire de variables duales (π_i, π_j), on effectue les vérifications suivantes :

— Si $\hat{y}_{ij} > (1 - \beta)$, on accepte l'inégalité duale $\pi_i \geq \pi_j$
— Si $\hat{y}_{ij} < \beta$, on accepte l'inégalité duale $\pi_j \geq \pi_i$

où β est un seuil d'acceptation.

4.2.3 Représentation des inégalités duales

On représente l'ensemble des inégalités duales prédites et acceptées à l'aide d'un graphe que l'on dénote par $\mathcal{GI}^p$. $\mathcal{GI}^p = (V^p, A^p)$ où V^p est l'ensemble des nœuds et A^p l'ensemble des arcs. L'ensemble $V^p = \{o, s\} \cup \Pi$ contient un nœud *source*, un nœud *puits* et un nœud par variable duale associée aux contraintes (3.7) de couverture de trajet $\Pi = \{(\pi_i)_{t_i \in \mathcal{T}}\}$. L'ensemble $A^p = \{o \times \Pi\} \cup \{\Pi \times s\} \cup \Gamma$ contient des arcs entre le nœud *source* et les nœuds Π, des arcs entre les nœuds Π et le nœud *puits*, ainsi qu'un arc par inégalité duale prédite par le modèle, i.e., $\Gamma = \{(\pi_i, \pi_j) | (i \neq j) \wedge ((\hat{y}_{ij} > 1 - \beta) \vee (\hat{y}_{ij} < \beta))\}$. On assigne un coût de -1 à l'ensemble des arcs A^p pour être en mesure d'utiliser l'algorithme Bellman-Ford (Bellman, 1958) afin de retirer des cycles et extraire des inégalités duales. En effet, étant donné qu'un

modèle de prédiction imparfait est utilisé pour prédire les inégalités duales, des cycles et des arcs représentant des inégalités duales invalides peuvent être insérés dans le graphe. On présente à la section suivante une méthode permettant de détecter des erreurs potentielles de prédiction et des cycles.

4.3 Détection de cycles et réduction d'erreurs potentielles

Afin de détecter les erreurs potentielles, l'ensemble des prédictions est analysé pour déterminer si des arcs contredisent l'ensemble des prédictions. Plus précisément, on suppose que, plus souvent une variable duale est prédite comme étant inférieure à une autre, plus sa valeur devrait être petite, tandis que, plus souvent une variable duale est prédite comme étant supérieure à une autre, plus sa valeur devrait être grande. Par conséquent, si une variable duale qui a souvent été prédite comme étant inférieure à une autre est prédite comme étant supérieure à une variable duale qui a souvent été prédite comme étant supérieure à une autre, on en déduit que cette prédiction est fort probablement fausse. On explique maintenant, de façon formelle, comment détecter ces erreurs potentielles. Tout d'abord, pour chaque nœud π_i dans $\mathcal{GI}^p$, on définit $deg(\pi_i)$ de la façon suivante :

$$deg(\pi_i) = in(\pi_i) - out(\pi_i) \tag{4.17}$$

où $in(\pi_i)$ représente le nombre d'arcs entrants au nœud π_i (le nombre de fois que π_i a été prédite comme étant inférieure à une autre variable duale) et $out(\pi_i)$ représente le nombre d'arcs sortants du nœud π_i (le nombre de fois que π_i a été prédite comme étant supérieure à une autre variable duale). Ensuite, on retire de $\mathcal{GI}^p$ tous les arcs (π_i, π_j) tels que $deg(\pi_i) < deg(\pi_j)$.

Finalement, pour détecter les cycles dans $\mathcal{GI}^p$, on utilise une variante de l'algorithme de Bellman-Ford qui permet de détecter la présence de cycles dans un graphe. Pour chaque cycle détecté, on retire l'ensemble des arcs du cycle de $\mathcal{GI}^p$.

L'Algorithme 1 permet de réduire les erreurs potentielles ainsi que détecter et supprimer les cycles. Les fonctions `nodes`($\mathcal{GI}^p$) et `arcs`($\mathcal{GI}^p$) (lignes 1 et 4) retournent les nœuds et les arcs du graphe $\mathcal{GI}^p$, respectivement. Les fonctions `from`(e) et `to`(e) appelées aux lignes 5 et 6 sont des fonctions qui retournent le nœud d'origine et de destination de l'arc e, respectivement. `removeArc`($e, \mathcal{GI}^p$) (ligne 8) est une fonction qui retire l'arc e du graphe $\mathcal{GI}^p$. La fonction `BellmanFordNeg`($o, s, \mathcal{GI}^p$) appelée à la ligne 12 est une variante de l'algorithme Bellman-Ford qui permet de détecter la présence d'un cycle de coût négatif dans le graphe $\mathcal{GI}^p$ et qui retourne celui-ci s'il en existe un. Finalement, `removeCycle`(neg_cycle, $\mathcal{GI}^p$) (ligne 14)

est une fonction qui retire du graphe $\mathcal{GI}^p$ les arcs du cycle *neg_cycle*. Cet algorithme a une complexité temporelle dans le pire des cas de $\mathcal{O}(|V^p||A^p||A^p|)$. Les deux premières boucles `for` ont une complexité de $\mathcal{O}(|V^p|)$ et $\mathcal{O}(|A^p|)$, respectivement. Puisque l'on retire au moins 2 arcs à chaque fois qu'un cycle est détecté, il y a $|\frac{A^p}{2}|$ cycles dans le pire cas. La fonction `BellmanFord()` a une complexité de $\mathcal{O}(|V^p||A^p|)$ dans le pire des cas et peut être appelée jusqu'à $|\frac{A^p}{2}|$ fois. Cette complexité est très mauvaise, mais il est important de noter qu'il est pratiquement impossible de tomber dans ce pire cas. En effet, plusieurs démarches sont prises en considération lorsqu'on crée le graphe $\mathcal{GI}^p$ pour éviter d'insérer des arcs inutiles. Par exemple, avant d'insérer l'arc (π_i, π_j) dans le graphe, on vérifie si l'arc (π_j, π_i) n'est pas déjà inséré.

Algorithme 1 Réduction des erreurs potentielles et suppression des cycles

Entrée: $\mathcal{GI}^p$

pour tout π in `nodes`($\mathcal{GI}^p$) **faire**
  `deg`[π] $\leftarrow$ `in`(π) $-$ `out`(π)
fin pour
pour tout e in `arcs`($\mathcal{GI}^p$) **faire**
  $u \leftarrow$ `from`(e)
  $v \leftarrow$ `to`(e)
  si `deg`[v] $<$ `deg`[u] **alors**
    `removeArc`(e,$\mathcal{GI}^p$)
  fin si
fin pour
tant que `true` **faire**
  neg_cycle = `BellmanFord`($o, s, \mathcal{GI}^p$)
  si `len`(*neg_cycle*) > 0 **alors**
    $\mathcal{GI}^p \leftarrow$ `removeCycle`(*neg_cycle*, $\mathcal{GI}^p$)
  sinon
    sortie
  fin si
fin tant que
retourne $\mathcal{GI}^p$

En conséquence, le graphe $\mathcal{GI}^p$ retourné par cette fonction ne contient plus de cycles et les erreurs de prédiction sont réduites. On peut maintenant extraire des inégalités duales à partir de ce graphe.

4.4 Extraction d'inégalités duales

Il existe énormément d'inégalités duales prédites dans le graphe $\mathcal{GI}^p$. En effet, il existe au maximum une inégalité duale par paire de trajet. L'objectif est donc de sélectionner ju-

dicieusement un sous-ensemble d'inégalités duales pertinentes. Plusieurs sélections de sous-ensembles d'inégalités duales ont été considérées dans le but de déterminer lesquels étaient les plus efficaces pour accélérer la résolution du PM. Parmi les sélections étudiées, une consistait à regrouper les variables duales en fonction de leur valeur et d'ajouter une inégalité duale pour chaque paire de variables duales appartenant à deux groupes différents. Une autre méthode plus simple consistait à sélectionner aléatoirement une paire de variables duales et à ajouter l'inégalité entre ces variables duales. Finalement, après avoir effectué quelques tests sur des instances de petite taille, la méthode la plus simple et la plus prometteuse semble être d'ordonner les variables duales. Pour ce faire, on génère des séries d'inégalités ayant la forme suivante :

$$\pi_1 \geq \pi_2 \geq \pi_3 \geq ... \geq \pi_{s_i} \tag{4.18}$$

où s_i est la longueur de la série d'inégalités i. En effet, ces séries d'inégalités sont assez serrées et simples à extraire de $\mathcal{GI}^p$. On présente une méthode permettant de générer ces séries d'inégalités.

Pour trouver la plus longue série d'inégalités ayant la forme (4.18), on utilise encore une fois l'algorithme Bellman-Ford. Cet algorithme permet de trouver le chemin de coût minimal entre deux nœuds dans un graphe orienté et pondéré. Cet algorithme fonctionne également avec des graphes ayant des arêtes de coût négatif lorsque aucun cycle de coût négatif n'est présent, ce qui est le cas pour le graphe $\mathcal{GI}^p$. Comme tous les arcs ont un coût de -1, trouver le chemin le moins cher entre le nœud source (o) et le nœud puits (s) revient à trouver la série d'inégalités la plus longue.

On définit (o, I, s) comme le chemin de coût minimal entre o et s passant par l'ensemble des nœuds $I \subset \Pi$. La séquence de nœuds visités dans I est une série d'inégalités duales prédites de la forme (4.18). Pour générer une deuxième longue série d'inégalités, on retire l'arc (i, j) de $\mathcal{GI}^p$ pour chaque inégalité $\pi_i \geq \pi_j$ dans I et on relance l'algorithme Bellman-Ford pour obtenir un nouveau chemin de coût minimal. On répète ce processus jusqu'à ce qu'un nombre suffisant de séries d'inégalités ait été généré ou jusqu'à ce qu'il ne reste plus de chemins entre o et s.

L'algorithme 2 explique comment obtenir N longues séries d'inégalités à partir d'un graphe $\mathcal{GI}^p$ tel que décrit précédemment. La fonction `BellmanFord`$(o, s, \mathcal{GI}^p)$ appelée à la ligne 4 trouve le chemin de coût minimal entre le nœud o et le nœud s du graphe $\mathcal{GI}^p$. La fonction `len`$(path)$ (ligne 5) retourne le nombre de nœuds présents dans $path$. On vérifie qu'au moins 4 nœuds sont présents dans $path$ car sinon aucune inégalité ne peut être générée.

En effet, si *path* contient 3 nœuds, on a un chemin de la forme suivante : (o, π_j, s). La fonction `extractIneq`($path$), appelée à la ligne 6, retourne la séquence des nœuds présents dans *path* sans les nœuds o et s. Finalement, `removeIneq`($\mathcal{GI}^p, ineq$) (ligne 8) retire les arcs correspondant aux inégalités dans *ineq* du graphe $\mathcal{GI}^p$. L'algorithme a une complexité temporelle dans le pire des cas de $\mathcal{O}(N|V^p||A^p|)$. En effet, la fonction `BellmanFord()` a une complexité de $\mathcal{O}(|V^p||A^p|)$ et dans le pire des cas, cette fonction est appelée N fois. Les fonctions `extractIneq()` et `removeIneq()` ont quant à elles une complexité de $\mathcal{O}(|V^p|)$.

Algorithme 2 Extraction de N séries d'inégalités

Entrée: $\mathcal{GI}^p$

```
1:  series ← ∅
2:  n ← 1
3:  tant que n ≤ N faire
4:     path ← BellmanFord(o, s, GI^p)
5:     si len(path) > 3 alors
6:        ineq ← extractIneq(path)
7:        series ← series ∪ ineq
8:        GI^p ← removeIneq(GI^p, ineq)
9:        n ← n + 1
10:    sinon
11:       sortie
12:    fin si
13: fin tant que
14: retourne series
```

4.5 Processus de recouvrement

Comme on utilise un modèle de prédiction pour déterminer les inégalités duales à ajouter, il est fort probable que certaines inégalités non valides soient introduites. En conséquence, la solution du PM^{ID} pourrait ne pas être valide pour le PM puisque chaque trajet pourrait ne pas être couvert exactement une fois. En effet, à cause des nouvelles variables intégrées au primal, certaines des contraintes de couverture pourraient être satisfaites si une ou des nouvelles variables intégrées sont positives. On donne un exemple afin de démontrer ce qui arrive si une inégalité duale invalide est ajoutée au dual.

Supposons que l'on ajoute l'inégalité duale $\pi_i \geq \pi_j - \phi_{ij}$, mais qu'en réalité, dans une solution optimale du PM, $\pi_i < \pi_j - \phi_{ij}$. Dans ce cas, w_{ij} prendrait une valeur positive dans la solution du PM^{ID}. En effet, si w_{ij} prend une valeur positive dans la solution, ceci équivaut à couvrir le trajet i exactement $1 + w_{ij}$ fois et couvrir le trajet j exactement $1 - w_{ij}$, ce qui revient à un coût approximatif de $\pi_i(1 + w_{ij}) + \pi_j(1 - w_{ij}) + \phi_{ij}w_{ij}$. Au contraire, si w_{ij} prend une valeur

nulle dans la solution, ceci équivaut à couvrir le trajet i et j une fois chacun, ce qui revient à un coût de $\pi_i + \pi_j$. On montre maintenant que le coût associé à $w_{ij} > 0$ est inférieur à celui associé à $w_{ij} = 0$ sachant que $\pi_i < \pi_j - \phi_{ij}$:

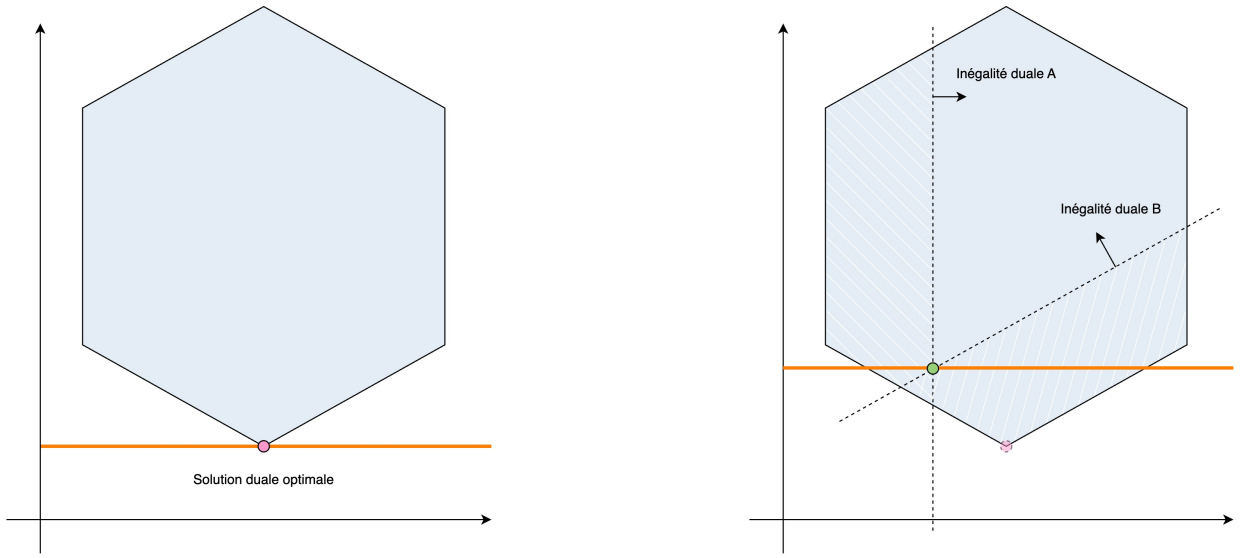
$$\begin{aligned}
& \pi_i(1 + w_{ij}) + \pi_j(1 - w_{ij}) + \phi_{ij}w_{ij} < \pi_i + \pi_j \\
\leftrightarrow \quad & \pi_i + \pi_i w_{ij} + \pi_j - \pi_j w_{ij} + \phi_{ij}w_{ij} < \pi_i + \pi_j \\
\leftrightarrow \quad & \pi_i w_{ij} - \pi_j w_{ij} + \phi_{ij}w_{ij} < 0 \\
\leftrightarrow \quad & w_{ij}\left(\pi_i - \pi_j + \phi_{ij}\right) < 0 \\
\leftrightarrow \quad & \pi_i - \pi_j + \phi_{ij} < 0 \\
\leftrightarrow \quad & \pi_i < \pi_j - \phi_{ij}
\end{aligned}$$

Grâce à cette observation, une fois le PM^{ID} résolu, on peut déterminer si des inégalités duales prédites et insérées au PM sont potentiellement invalides. En effet, si des variables w_{ij} prennent une valeur positive dans la solution, l'inégalité duale associée ($\pi_i \geq \pi_j - \phi_{ij}$) est potentiellement invalide. Il est important de noter que des variables w_{ij} prenant des valeurs positives dans la solution peuvent être associées à des inégalités valides. En effet, si une inégalité invalide est insérée dans le PM^{ID}, celle-ci peut rendre une inégalité, initialement valide, invalide selon la nouvelle solution duale optimale. On explique graphiquement cette situation à la Figure 4.4 où la ligne orange représente la fonction objectif duale à minimiser. La Figure 4.4a représente l'espace de solutions duales initial où la solution duale optimale est représentée par un point rose. Ensuite, à la Figure 4.4b, on réduit l'espace de solutions duales en insérant deux inégalités duales (représentées par des lignes pointillées) dont une invalide (inégalité B). Comme on peut le voir, l'inégalité duale A est inactive (valide) selon la solution duale optimale initiale (point rose) mais est active (invalide) selon la nouvelle solution duale optimale (point vert), lorsqu'on ajoute l'inégalité duale B. Ainsi, si cette inégalité n'était pas ajoutée, on obtiendrait une autre solution duale optimale.

Afin de corriger ces erreurs potentielles, un processus de recouvrement a été développé. Lorsqu'une solution au PM^{ID} contient une ou plusieurs variables w_{ij} ayant une valeur supérieure à 0, deux modifications peuvent être apportées au PM^{ID} :

1. Augmenter la valeur de ϕ_{ij} pour tenter de rendre l'inégalité $\pi \geq \pi_j - \phi_{ij}$ valide ;
2. Retirer l'inégalité duale $\pi_i \geq \pi_j - \phi_{ij}$ et en conséquence, retirer la variable w_{ij} du primal.

Ensuite, on relance la résolution de ce nouveau PM^{ID} modifié. On peut itérer ainsi jusqu'à ce



(a) Espace de solutions duales sans inégalités duales

(b) Espace de solutions duales avec inégalités duales

FIGURE 4.4 Ensembles de solutions duales avec et sans inégalités duales

qu'on obtienne une solution au PM^{ID} qui ne contient aucune variable w_{ij} supérieure à 0, ou lorsqu'un certain nombre d'itérations est atteint. Cependant, dans le deuxième cas, la solution obtenue n'est pas garantie d'être équivalente à la solution du PM et de couvrir l'ensemble des trajets exactement une fois. Pour accélérer ce processus de recouvrement, on définit aussi un critère d'arrêt prématuré. Si durant g itérations consécutives, la valeur de la fonction objectif ne diminue pas d'au moins Δ et qu'au moins une variable w_{ij} a une valeur supérieure à 0, alors la résolution du PM^{ID} est arrêtée prématurément. Ensuite, on modifie les coefficients ϕ_{ij} associés aux variables w_{ij} prenant des valeurs positives ou on retire l'inégalité associée à la variable si c'est la deuxième fois qu'elle prend une valeur supérieure à 0 dans une solution du PM^{ID}. Finalement, on relance la résolution du PM^{ID}. Cependant, afin de s'assurer d'avoir une solution optimale au PM^{ID}, le critère d'arrêt prématuré est désactivé après un certain nombre d'itérations. L'Algorithme 3 résume ce processus de recouvrement.

Algorithme 3 Processus de recouvrement

Entrée: PM^{ID}, `maxIter`

`stopSolvingPMID` $\leftarrow$ `false`
`recoveryCount` $\leftarrow 0$
tant que `!stopSolvingPMID` **faire**
  Initialiser le PMR à partir du PM^{ID}
  `stopSolvingPMR` $\leftarrow$ `false`
  tant que `!stopSolvingPMR` **faire**
    Résoudre le PMR
    si Val. obj. n'a pas diminué d'au moins Δ depuis g itérations **&** Au moins un $w_{ij} > 0$ **&** `recoveryCount` $<$ `maxIter` **alors**
      /* Arrêt précoce */
      `stopSolvingPMR` $\leftarrow$ `true`
    sinon
      Résoudre le SP
      si Existe $\bar{c}_s < 0$ **alors**
        Modifier le PMR
      sinon
        `stopSolvingPMR` $\leftarrow$ `true`
      fin si
    fin si
  fin tant que
  si Au moins un $w_{ij} > 0$ **alors**
    si `recoveryCount` $<$ `maxIter` **alors**
      Modifier le PM^{ID} en modifiant les inégalités duales dont $w_{ij} > 0$
      `recoveryCount++`
    sinon
      `stopSolvingPMID` $\leftarrow$ `true`
    fin si
  sinon
    `stopSolvingPMID` $\leftarrow$ `true`
  fin si
fin tant que

CHAPITRE 5 RÉSULTATS THÉORIQUES ET EXPÉRIMENTAUX

On présente maintenant les résultats de différentes expériences qui ont été réalisées dans le but d'évaluer l'impact des inégalités duales sur la résolution du MDEVSP. On introduit à la Section 5.1 les instances utilisées pour entraîner et tester le modèle de prédiction. Ensuite, à la Section 5.2, on présente des résultats obtenus à l'aide d'un modèle de prédiction parfait, dans le but de démontrer le potentiel théorique de l'approche. On poursuit à la Section 5.3 avec la présentation de résultats expérimentaux obtenus sur des instances réelles du MDEVSP à l'aide d'inégalités duales prédites par le modèle. Finalement, à la Section 5.4, on tente de démontrer que l'approche présentée peut être généralisée sur d'autres problèmes d'horaires d'autobus électriques en présentant des résultats d'expériences effectuées sur des instances du SDEVSP.

5.1 Génération d'instances

On utilise un générateur d'instances, introduit par Brasseur (2022), pour créer une banque d'instances diversifiées et réalistes du MDEVSP. Ce générateur est basé sur un sous-réseau, fourni par GIRO, qui comprend plusieurs lignes de bus ainsi que l'emplacement des stations de départ et d'arrivée, des dépôts et des stations de recharge. Le réseau est présenté à la Figure 5.1 et est composé de 4 lignes de bus aller-retour (pour un total de 8 lignes) et de deux dépôts. Les données initiales fournies par GIRO incluent un horaire de trajets pour chaque ligne. Pour avoir des instances variables, on modifie de façon aléatoire les heures de départs des trajets. Une journée est divisée en 5 intervalles de temps durant lesquels la fréquence sur chaque ligne demeure constante. Par la suite, pour chaque ligne et chaque intervalle, on tire un nombre aléatoire pour déterminer la variation de fréquence sur cet intervalle et cette ligne. On tire ensuite aléatoirement une heure de départ pour le premier trajet de cet intervalle. Les variations de la fréquence sur l'ensemble des intervalles et lignes sont corrélées : si on veut générer une instance qui contient plus de trajets que l'horaire initial, les variations seront à la hausse en moyenne et vice-versa. Les valeurs des différents paramètres des instances sont définies dans le Tableau 5.1. On définit deux types d'instances : les instances de moyenne taille (M) et de grande taille (G) ayant environ 850 et 1050 trajets à couvrir, respectivement. Les instances sont résolues à l'aide de GENCOL (Desrosiers, 2010), un algorithme de génération de colonnes faisant appel à CPLEX (version 20.1). On utilise un algorithme de barrière pour résoudre PMR. Les résolutions sont faites sur des ordinateurs ayant des processeurs Intel(R) Core(TM) i7-8700 fonctionnant à 3.20 GHz, utilisant un maximum de 8 Gb de RAM et

exécutant la version 3.10 de Linux. On impose une limite de 6 heures par résolution, après laquelle la résolution de la relaxation linéaire en cours se termine afin d'obtenir une borne inférieure valide. En conséquence, le temps total de la résolution peut excéder 6 heures. On présente à la Section 5.3.2 comment on utilise ces instances pour entraîner le modèle de prédiction.

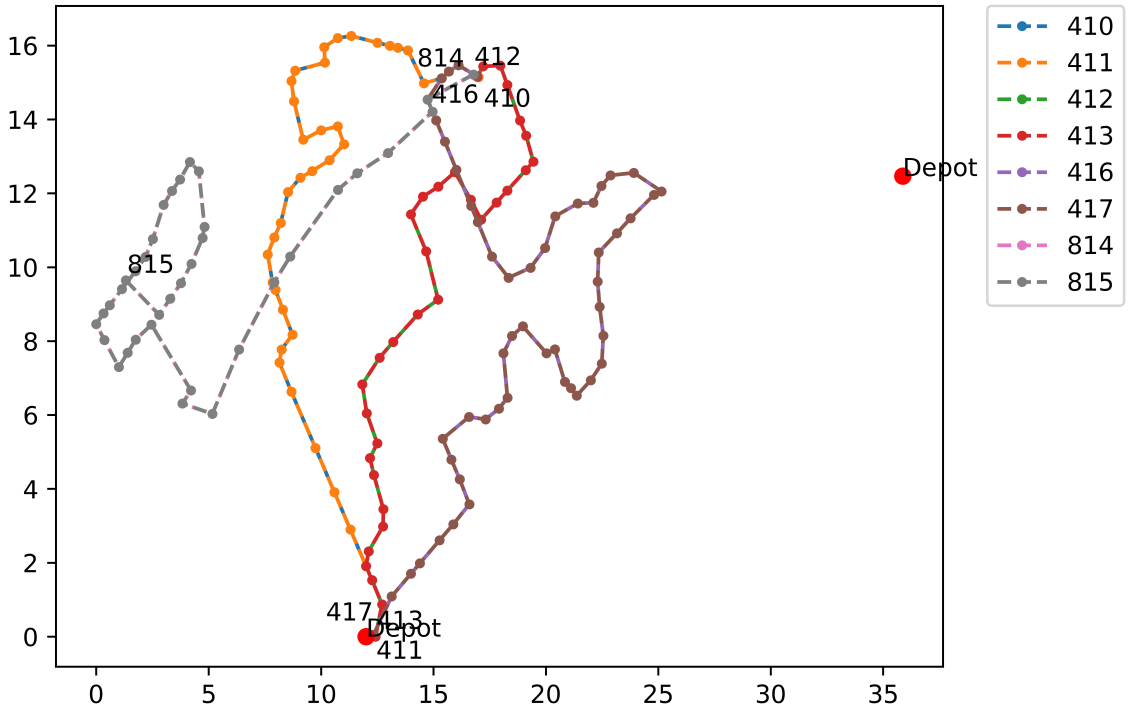


FIGURE 5.1 Réseau utilisé pour le MDEVSP

5.2 Résultats théoriques

Pour évaluer le potentiel théorique de l'approche présentée, on génère 25 instances M et 25 instances G avec une capacité maximale de 55 autobus par dépôt. Ensuite, on résout le PM de chaque instance et on récupère les valeurs des variables duales $\hat{\pi}_t$. Puisqu'on connait les valeurs de l'ensemble des variables duales $\hat{\pi}_t$, on peut construire directement le graphe $\mathcal{GI}$ correspondant où chaque arc représente une inégalité duale valide. À l'aide du graphe $\mathcal{GI}$, on choisit arbitrairement de générer $N = 3$ séries d'inégalités duales valides et on définit le PM^{ID} correspondant. Finalement, on résout le PM^{ID} de chaque instance et on compare les temps de résolution des PM^{ID} avec ceux des PM. Pour chaque instance, on calcule le facteur d'accélération (*Acc.*) de la façon suivante :

Tableau 5.1 Valeurs des paramètres des instances

Paramètre	Valeur
c_f	1000
c_a	2
c_d	4
e_a	183.33 Wh
e_d	315 Wh
δ	45 mins.
λ	15 mins.
b_h	3
σ_{min}	0 Wh
σ_{max}	100000 Wh

$$Acc = \frac{temps}{temps^{ID}} \tag{5.1}$$

où $temps$ et $temps^{ID}$ sont les temps de résolution du PM et du PM^{ID}, respectivement. On calcule également l'écart (gap) entre la valeur optimale du PM et celle du PM^{ID} selon l'équation suivante :

$$gap = \frac{(val - val^{ID})}{val} \tag{5.2}$$

où val est la valeur optimale du PM et val^{ID} celle du PM^{ID}. Finalement, on calcule aussi la proportion de trajets qui ne sont pas couverts exactement une fois dans la solution du PM^{ID} (*Non Cov.*). Comme expliqué précédemment, certaines inégalités duales introduites sont potentiellement invalides, ce qui peut entraîner une couverture de certains trajets plus ou moins qu'une fois dans la solution. Les résultats de chaque instance sont présentés en détail à l'Annexe A. Le Tableau 5.2 présente les moyennes des différentes métriques pour chaque taille d'instance où $|\mathcal{T}|$ est le nombre moyen de trajets à couvrir, *Nb. bus* est le nombre moyen d'autobus utilisés dans la solution, *Acc* est la moyenne géométrique des facteurs d'accélération et *Inég.* est le nombre moyen d'inégalités duales ajoutées au PM pour obtenir le PM^{ID} correspondant.

Tout d'abord, il est important de souligner que les temps de résolution des PM sont déjà assez rapides. En effet, l'utilisation de la technique de perturbation lors de la résolution du PM semble suffisante pour lutter contre la dégénérescence pour une majorité de ces instances

Tableau 5.2 Résultats théoriques moyens

		PM		PM^{ID}					
Taille	$\|\mathcal{T}\|$	Nb. bus	Temps CPU (s)	Nb. bus	Temps CPU (s)	Acc.	Gap	Non Cov.	Inég.
M	837.4	52.077	594.00	52.077	103.368	5.33	0.00%	0%	1671.68
G	1030.48	62.557	1260.42	62.557	175.32	3.34	0.00%	0%	2057.88

du MDEVSP. Cependant, étant donné que l'ensemble des inégalités duales introduites sont valides, on obtient de bonnes accélérations. En effet, la résolution du PM^{ID} est tout le temps (sauf une instance M) plus rapide que celle du PM et fournit exactement la même solution. Bien entendu, en pratique, il est impossible d'avoir un modèle de prédiction parfait ayant une précision de 100%. On est conscient que cette accélération ne peut pas être obtenue en pratique. Toutefois, ces résultats montrent le potentiel des inégalités duales à accélérer la résolution du MDEVSP, même lorsque qu'une technique de perturbation est utilisée.

5.3 Résultats expérimentaux

On évalue maintenant la capacité du modèle présenté à la Section 4.2 à prédire des inégalités duales, ainsi que l'impact de celles-ci sur l'accélération de la résolution du PM. Cette fois-ci, comme on cherche à entraîner le modèle, on génère 250 instances M et 250 instances G, en imposant une capacité maximale de 100 autobus par dépôt. Étonnamment, on constate que c'est en imposant une capacité par dépôt largement supérieure au nombre d'autobus nécessaire pour couvrir l'ensemble des trajets que ces instances subissent de la dégénérescence, même lorsque la technique de perturbation est utilisée. En effet, cette technique semble être suffisante pour contrer la dégénérescence lorsque les capacités des dépôts sont plus restrictives. Même si ces instances ne sont pas nécessairement réalistes, elles permettent de déterminer si des inégalités duales prédites par un modèle d'apprentissage automatique peuvent aider à accélérer la résolution d'instances souffrant de dégénérescence.

5.3.1 Implémentation du modèle de prédiction

Tel que présenté à la Figure 4.2, on utilise 3 couches d'EGATConv (DGL, 2023). Il y a 21 caractéristiques initiales par nœud et 7 par arêtes ($F^1 = 21$, $E^1 = 7$) et 32 caractéristiques par nœud et par arêtes pour les couches suivantes ($E^2 = F^2 = E^3 = F^3 = 32$). On utilise $K = 3$ mécanismes d'attention, ce qui donne un total de 94464 paramètres pour le bloc GNN. Pour le perceptron multicouches, on utilise 8 couches ayant respectivement 32, 64, 128, 128, 64, 32, 16 et 1 neurones. Les vecteurs d'entrée comprennent 64 caractéristiques.

Avec les vecteurs de biais, on a un total de 39905 paramètres à apprendre pour le perceptron multicouches. Finalement, on utilise l'algorithme d'optimisation Adam (Kingma et Ba, 2014) pour entraîner le modèle.

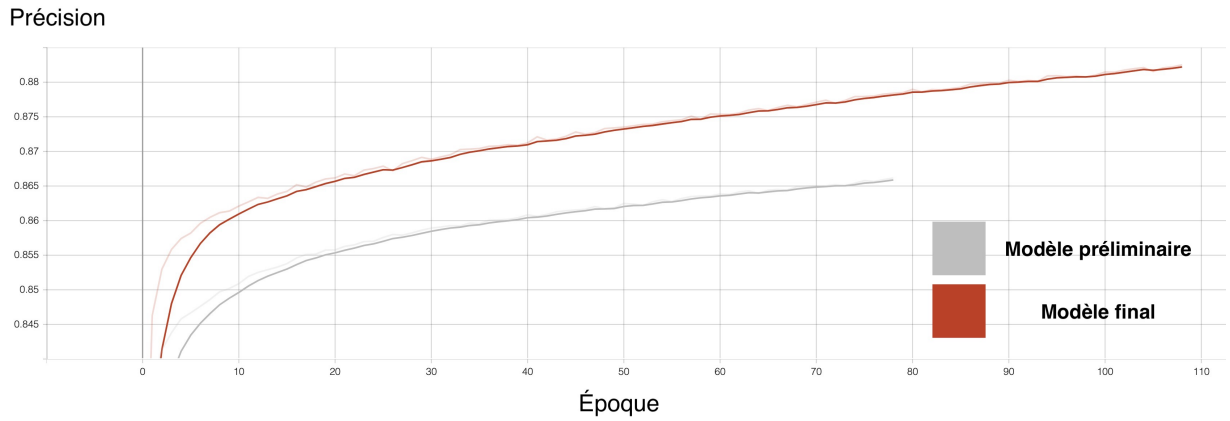
5.3.2 Entraînement du modèle

Encore une fois, on commence par résoudre le PM des 500 instances et récupérer les valeurs des variables duales des solutions de chaque instance. On divise les instances en ensemble d'entraînement, de validation et de test en utilisant 80%, 10% et 10% des instances de chaque taille, respectivement. On utilisera l'ensemble de test pour comparer les temps de résolution et la qualité des solutions du PM et du PM^{ID}. On entraîne le modèle sur une carte graphique NVIDIA Tesla P100 PCIe avec 16 GB de RAM dans un ordinateur Intel(R) Xeon(R) CPU E5-2650 v4 fonctionnant à 2.20GHz et exécutant la version 4.18 de Linux.

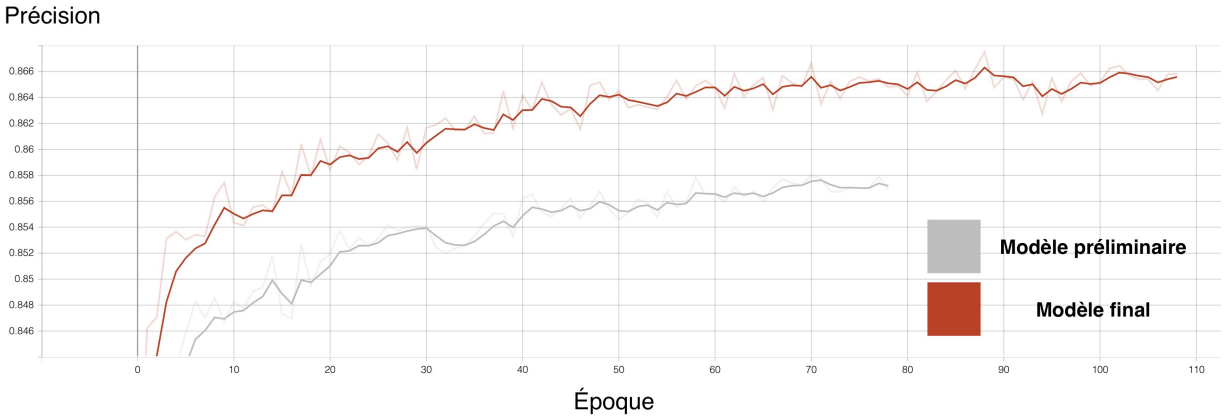
Lors de la phase d'entraînement, on utilise un seuil d'acceptation $\beta = 0.5$. Cela signifie que si $\hat{y}_{ij} > 0.5$, l'inégalité $\pi_i \geq \pi_j$ est prédite par le modèle comme étant valide. La fonction de perte utilisée est l'entropie croisée binaire définie selon l'Équation (4.15). Un critère d'arrêt précoce est mis en place durant l'entraînement. En effet, si la fonction de perte pour l'ensemble de validation n'a pas diminué durant p itérations consécutives, l'entraînement est arrêté. On présente les courbes de précision et de fonction de perte du modèle après chaque époque sur les ensemble d'entraînement et de validation aux Figures 5.2 et 5.3. La ligne grise représente une première version du modèle entraîné sur 800 instances de tailles variables allant de 650 à 1050 trajets, tandis que la ligne brune représente la version finale du modèle entraîné sur 200 instances de type M et 200 instances de type G. Les lignes pâles sont les valeurs à chaque époque, tandis que les lignes foncées sont ces données lissées. On remarque tout d'abord que le fait d'avoir moins d'instances, mais de tailles plus similaires améliore la performance du modèle. De plus, ces courbes suggèrent que le modèle apprend bien et que l'entraînement est arrêté prématurément avant de devenir sur-ajusté.

5.3.3 Précision

Une fois l'entraînement terminé, on évalue différents seuils d'acceptation β pour déterminer lequel permet à l'algorithme de générer des prédictions de qualité tout en ayant suffisamment d'inégalités prédites. En effet, si le seuil est trop strict, peu d'inégalités $\pi_i \geq \pi_j$ seront acceptées et le graphe $\mathcal{GI}^p$ ne sera peut-être pas assez dense. Cela risque de réduire le nombre d'inégalités pouvant être générées. D'un autre côté, si le seuil n'est pas assez strict, trop d'inégalités prédites pourraient être acceptées, augmentant ainsi le nombre d'inégalités invalides introduites. On cherche donc un seuil qui permet d'obtenir un équilibre entre la qualité et la

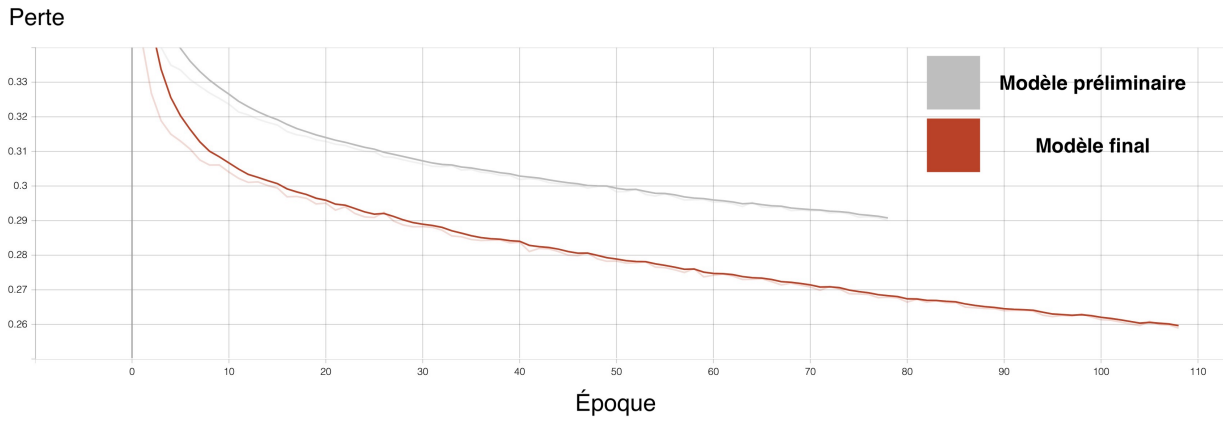


(a) Ensemble d'entraînement

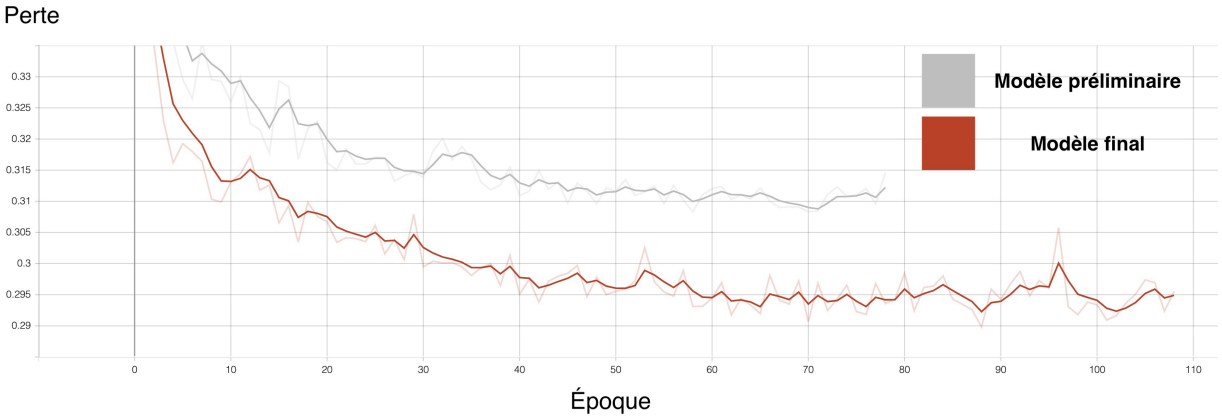


(b) Ensemble de validation

FIGURE 5.2 Précision par époque



(a) Ensemble d'entraînement



(b) Ensemble de validation

FIGURE 5.3 Perte par époque

quantité des prédictions. Pour déterminer ce seuil, on évalue différents seuils sur l'ensemble de validation en mesurant le pourcentage d'inégalités valides acceptées ainsi que le nombre total d'inégalités acceptées. On présente dans le Tableau 5.3 la précision et le nombre d'inégalités acceptées en fonction du seuil β.

TABLEAU 5.3 Analyse des performances en fonction du seuil d'acceptation β

	Précision		Nb. Ineq.	
β	M	G	M	G
0.15	94.7%	95.0%	275264	427927
0.2	93.4%	93.7%	296275	459932
0.25	92.0%	92.4%	314679	488256
0.3	90.6%	91.0%	331698	514665
0.35	89.2%	89.6%	348041	539769
0.4	87.7%	88.1%	364392	564235
0.45	86.2%	86.7%	379695	586923

Après avoir analysé ces résultats et testé rapidement les inégalités générées par les différents seuils sur quelques instances, on décide d'utiliser un seuil $\beta = 0.15$. En effet, on considère qu'une précision d'environ 95% sur l'ensemble des inégalités duales prédites est acceptable et que le nombre total d'inégalités acceptées est satisfaisant.

On évalue aussi le pourcentage d'inégalités valides introduites lorsque le modèle est utilisé avec un seuil $\beta = 0.15$. En effet, même si 95% des inégalités sont prédites correctement, le pourcentage d'inégalités valides introduites dans le PM, à l'aide de la méthode présentée à la Section 4.4 peut être différent. À l'aide du modèle et d'un seuil $\beta = 0.15$, on prédit la probabilité que l'inégalité $\pi_i \geq \pi_j$ soit valide pour chaque paire de variables duales et pour chaque instance de l'ensemble de test. Ensuite, on génère les graphes $\mathcal{GI}^p$ basés sur ces prédictions et on extrait $N = 25$ séries d'inégalités par instance pour générer les PM^{ID}. À l'aide des valeurs des variables duales $\hat{\pi}_t$, on vérifie chaque inégalité duale générée. En moyenne, on génère 638.92 inégalités duales dont 88.56% valides pour les instances M et 720.68 inégalités duales dont 86.78% valides pour les instances G. On remarque une grande différence entre la précision générale du modèle et la proportion d'inégalités valides introduites. Cette différence est probablement due au fait qu'on cherche à introduire des inégalités duales serrées et qu'il est plus difficile pour le modèle de prédire correctement ces inégalités. Cependant, on estime tout de même que la proportion d'inégalités valides introduites est suffisante.

5.3.4 Analyse du potentiel d'accélération

Ensuite, on résout les PM^{ID} générés et on compare les solutions et les temps de résolution avec les PM correspondants. Pour le processus de recouvrement, on définit $g = 15$ et $\Delta = 1$. Tel que présenté à la Section 5.2, on calcule l'accélération, le *gap* et la proportion de trajets qui ne sont pas couverts exactement une fois pour chaque instance. On calcule aussi la moyenne géométrique des facteurs d'accélération des instances dont le PM se résout rapidement (*Acc. R.*) et ceux dont le PM se résout lentement (*Acc. L.*). Le seuil pour distinguer les instances dont le PM se résout *lentement* et *rapidement* est déterminé en utilisant le temps médian de résolution du PM sur l'ensemble de test. Pour les instances M, le seuil est de 409 secondes tandis que pour les instances G, le seuil est de 2423 secondes. Ceci veut dire qu'une instance G dont le temps de résolution du PM est de 400 secondes est catégorisée comme *rapide*. En effet, il est important de distinguer les instances *rapides* des instances *lentes* puisque les instances dont le PM se résout déjà rapidement ont moins besoin d'être accélérées.

Configuration 1

On commence par résoudre le PM^{ID} en utilisant un coefficient $\phi_{ij} = 0$ pour chaque inégalité ajoutée. De plus, on ne permet aucune itération du processus de recouvrement. Ceci signifie que même si certaines variables y_{ij} ont une valeur supérieure à 0 dans la solution et qu'une inégalité potentiellement invalide a été introduite dans le PM^{ID}, on conserve cette solution. Les résultats détaillés de chaque instance sont présentés dans l'Annexe B et un résumé de ces résultats est présenté dans le Tableau 5.4.

Tableau 5.4 Moyennes des métriques des résultats obtenus à l'aide de la configuration 1

		PM		PM^{ID}							
Taille	$\lvert\mathcal{T}\rvert$	Nb. bus	Temps CPU (s)	Nb. bus	Temps CPU (s)	Acc.	Acc. L.	Acc. R.	Gap	Non Cov.	Inég.
M	871.12	53.797	2273.65	53.757	498.96	1.71	2.67	1.06	0.19%	6.01%	638.8
G	1081.64	64.117	7675.99	64.077	3371.56	1.99	3.93	0.95	0.17%	5.73%	720.68

La première chose que l'on remarque est que les facteurs d'accélération sont encourageants. En effet, la moyenne géométrique des facteurs d'accélération est de 1.71 et 1.99 pour les instances M et G, respectivement. Cependant, en examinant les résultats détaillés présentés dans l'Annexe B, on constate que les facteurs d'accélération varient considérablement, allant de 0.46 à 11.53 pour les instances M et de 0.19 à 61.01 pour les instances G. On note également que les instances *lentes* ont un facteur d'accélération considérablement plus élevé que les instances *rapides*. En effet, on remarque dans l'Annexe B, que les instances dont

le facteur d'accélération est inférieur à 1 (celles dont la résolution du PM^{ID} a pris plus de temps que celle du PM) sont souvent des instances où le PM est résolu rapidement et pour lesquelles la perturbation semble suffisante pour contrer les effets de la dégénérescence. Cela suggère que les inégalités duales prédites semble accélérer, d'une certaine façon, la résolution d'instances qui souffrent de dégénérescence, même lorsqu'on utilise une technique de perturbation. En contrepartie, on observe une dégradation de la qualité des solutions. En effet, même si le *gap* est relativement faible, la proportion de trajets qui ne sont pas couverts est non négligeable (6.01% et 5.73% pour les instances M et G, respectivement). Ceci est légèrement problématique, car pour utiliser ces inégalités duales en pratique, il faudrait définir un nouveau type de branchement adapté à cette situation.

Configuration 2

Pour la deuxième résolution, on utilise toujours un coefficient $\phi_{ij} = 0$, mais on autorise maintenant jusqu'à deux itérations du processus de recouvrement. On utilise alors le critère d'arrêt prématuré présenté à la Section 4.5 qui permet d'arrêter prématurément la résolution du PM^{ID} pour tenter de corriger les inégalités duales invalides introduites. Les résultats détaillés de chaque instance sont présentés dans l'Annexe C, et un résumé des résultats est présenté dans le Tableau 5.5.

Tableau 5.5 Moyennes des métriques des résultats obtenus à l'aide de la configuration 2

		PM		PM^{ID}							
Taille	$\|\mathcal{T}\|$	Nb. bus	Temps CPU (s)	Nb. bus	Temps CPU (s)	Acc.	Acc. L.	Acc. R.	Gap	Non Cov.	Inég.
M	871.12	53.797	2273.65	53.757	1558.10	0.72	1.12	0.45	0.11%	3.54%	638.92
G	1081.64	64.117	7675.99	64.077	7323.38	0.80	1.19	0.53	0.09%	3.60%	720.72

Les résultats obtenus avec la deuxième configuration sont plus mitigés que ceux obtenus à l'aide de la première configuration (Tableau 5.4). En effet, les facteurs d'accélération sont beaucoup plus faibles, allant de 0.11 à 3.31 pour les instances M et de 0.14 à 13.93 pour les instances G. De plus, la résolution du PM^{ID} a pris plus de temps que celle du PM pour presque la moitié des instances. Ceci est dû au fait que deux itérations du processus de recouvrement sont autorisées. À chaque itération du processus de recouvrement, on modifie le PM^{ID} et on relance la résolution, ce qui ralentit considérablement le temps total de résolution. On note que même si on utilise un critère d'arrêt précoce, la dernière résolution du PM^{ID} doit se terminer lorsque aucune colonne n'est générée afin de garantir l'optimalité. Par contre, encore une fois, on remarque que le facteur d'accélération des instances *lentes* est supérieur à celui des instances *rapides*. Ceci concorde avec les observations effectuées plus

haut stipulant que les inégalités duales sont beaucoup plus efficaces pour les instances dont le PM est très dégénéré.

Cependant, ces résultats permettent de démontrer la capacité du processus de recouvrement à corriger certaines inégalités duales invalides, ce qui se traduit par une solution de meilleure qualité. En effet, même si le nombre moyen de bus utilisés et le *gap* moyen sont similaires à ceux présentés dans le Tableau 5.4, la proportion de trajets qui ne sont pas couverts correctement est réduite d'environ 50%. Plus précisément, on passe d'environ 5.87% de trajets non couverts correctement à environ 3.57%.

En somme, ces résultats suggèrent que le processus de recouvrement permet de détecter et de réparer des inégalités potentiellement invalides, améliorant ainsi la qualité de la solution. Toutefois, ce processus semble ralentir de façon significative la résolution et peut rendre les temps de résolution du PM^{ID} similaires ou même pires que ceux du PM.

Configuration 3

Pour la troisième configuration, on utilise un coefficient $\phi_{ij} = 1$. De plus, on ne permet qu'une seule itération du processus de recouvrement. Les résultats détaillés de chaque instance sont présentés dans l'Annexe D, et un résumé des résultats est présenté dans le Tableau 5.6.

Tableau 5.6 Moyennes des métriques des résultats obtenus à l'aide de la configuration 3

		PM		PM^{ID}							
Taille	$\lvert\mathcal{T}\rvert$	Nb. bus	Temps CPU (s)	Nb. bus	Temps CPU (s)	Acc.	Acc. L.	Acc. R.	Gap	Non Cov.	Inég.
M	871.12	53.797	2273.65	53.757	668.86	1.27	1.83	0.86	0.12%	1.81%	638.6
G	1081.64	64.117	7675.99	64.077	6877.96	1.28	2.04	0.83	0.10%	1.11%	720.56

Les résultats obtenus sont prometteurs, puisqu'ils démontrent un bon compromis entre l'accélération et la qualité des solutions. Les facteurs d'accélération sont encourageants mais varient beaucoup, allant de 0.36 à 39 pour les instances M et de 0.01 à 38.48 pour les instances G. De plus, les proportions moyennes de trajets qui ne sont pas couverts exactement une fois sont nettement inférieures à celles obtenues avec les configurations précédentes, passant de 3.54% à 1.81% pour les instances M et de 3.60 à 1.11% pour les instances G. Ces améliorations sont probablement attribuables aux coefficients ϕ_{ij} non nuls, qui font en sorte que plus d'inégalités duales sont valides et plus de variables y_{ij} prennent la valeur 0, réduisant ainsi le nombre d'inégalités à réparer lors du processus de recouvrement.

Encore une fois, on observe une différence majeure entre les facteurs d'accélération des instances *lentes* et *rapides*, ce qui confirme une fois de plus notre hypothèse stipulant que les

inégalités duales sont plus efficaces pour les instances très dégénérées. On suppose donc que la technique de perturbation est suffisante pour contrer la dégénérescence dans ces cas-là et que les inégalités duales prédites accélèrent la résolution lorsque cette perturbation n'est pas suffisante. Ces résultats mettent également en évidence l'efficacité des coefficients ϕ_{ij} supérieurs à 0. Enfin, ils soulignent aussi le potentiel pratique de l'approche proposée, montrant qu'il est possible de trouver un compromis entre la rapidité de la résolution et la qualité des solutions.

Discussion des résultats

Les résultats obtenus avec les différentes configurations démontrent le potentiel et la pertinence de l'approche proposée dans ce mémoire. En effet, les résultats indiquent qu'un modèle d'apprentissage automatique, relativement simple, est en mesure de prédire le sens des inégalités duales pour la majorité des paires de variables duales, ce qui permet parfois d'accélérer la résolution du PM. Bien que l'on note une légère dégradation de la qualité des solutions lorsque ces inégalités duales prédites sont utilisées, le processus de recouvrement présenté à la Section 4.5 permet de réparer ces solutions et d'améliorer leur qualité. Cependant, il est important de souligner que cette procédure est coûteuse en termes de temps. Dans certains cas, cela peut entraîner des temps de résolution du PM^{ID} plus longs que ceux pour résoudre le PM. Toutefois, il convient de noter que la plupart des instances ayant des facteurs d'accélération inférieurs à 1 sont des instances dont le PM se résout déjà rapidement. Il est aussi intéressant de noter que les instances pour lesquelles le temps de résolution du PM est très élevé présentent généralement une accélération très importante. On note cependant quelques instances où le temps de résolution du PM est très élevé et l'accélération très faible. En conclusion, l'approche proposée semble avoir plus de potentiel pour les instances où la perturbation n'est pas suffisante pour contrer les effets de la dégénérescence, et où les temps de résolution du PM sont particulièrement longs. Cependant, comme on ne peut pas déterminer à l'avance quelles instances sont très dégénérées, la méthode proposée mérite d'être améliorée afin d'offrir, idéalement, des facteurs d'accélération supérieurs ou égaux à 1.

5.4 SDEVSP

On présente maintenant les résultats d'expériences effectuées sur des instances du SDEVSP. Tel que mentionné à la Section 5.3.4, les instances du MDEVSP utilisées subissent beaucoup de dégénérescence lorsqu'on impose une capacité par dépôt largement supérieure au nombre d'autobus nécessaire pour couvrir l'ensemble des trajets. De plus, lorsqu'on analyse les solutions de ces instances, on remarque que l'ensemble des autobus proviennent du même dépôt.

On se questionne donc à savoir comment se comporte l'approche proposée lorsqu'on modifie ces instances du MDEVSP pour obtenir des instances du SDEVSP. On utilise le même réseau que celui présenté à la Figure 5.1 auquel on retire un dépôt. On génère des instances du SDEVSP, on entraîne le modèle et on produit les PM^{ID} correspondants de la même façon que pour les instances du MDEVSP. Lors de la résolution des PM^{ID}, on utilise un coefficient $\phi_{ij} = 0$ et on n'autorise aucune itération du processus de recouvrement. Les résultats détaillés de chaque instance sont présentés dans l'Annexe E, et un résumé des résultats est présenté dans le Tableau 5.7.

Tableau 5.7 Moyennes des métriques des résultats obtenus sur les instances du SDEVSP

		PM		PM^{ID}							
Taille	$\lvert\mathcal{T}\rvert$	Nb. bus	Temps CPU (s)	Nb. bus	Temps CPU (s)	Acc.	Acc. L.	Acc. R.	Gap	Non Cov.	Inég.
M	871.12	52.797	5832.73	52.677	1586.29	2.48	6.64	0.85	0.30%	3.6%	351.52
G	1081.64	63.237	9324.27	63.153	8159.03	1.02	1.96	0.56	0.16%	2.84%	376.48

À première vue, les facteurs d'accélération semblent excellents. Cependant, lorsqu'on regarde les résultats détaillés, on réalise qu'encore une fois, ces facteurs varient énormément. Ils varient de 0.14 à 128.83 pour les instances M et de 0.04 à 42.93 pour les instances G. Cependant, encore une fois, on remarque que les inégalités duales semblent accélérer la résolution d'instances qui subissent beaucoup de dégénérescence même lorsque la perturbation est utilisée. En effet, la moyenne géométrique des facteurs d'accélération des instances *lentes* est de 6.64 pour les instances M et 1.96 pour les instances G. Il est aussi intéressant de noter que même avec un coefficient ψ_{ij} nul et aucune itération du processus de recouvrement, l'écart moyen et la proportion de trajets qui ne sont pas couverts correctement sont assez faibles. En effet, si on compare ces proportions moyennes (3.6% pour les instances M et 2.84% pour les instances L) à celles obtenues lors de la résolution des PM^{ID} du MDEVSP avec la même configuration (6.01% pour les instances M et 5.73% pour les instances G), on remarque que celles du SDEVSP sont beaucoup plus faibles.

Ces résultats démontrent le potentiel de généralisation de l'approche proposée. En effet, il semblerait que le modèle pourrait être utilisé pour plusieurs types de problèmes d'horaires d'autobus tant et aussi longtemps qu'ils puissent être représentés sous forme de graphe. On discute davantage des pistes de recherche future au prochain chapitre.

CHAPITRE 6 CONCLUSION

On conclut maintenant ce mémoire en résumant les travaux présentés et en discutant des limitations de l'approche proposée et des améliorations potentielles.

6.1 Synthèse des travaux

Dans ce mémoire, on a défini formellement les différentes contraintes reliées à l'autonomie limitée des véhicules électriques et on a formulé le MDEVSP sous la forme d'un programme linéaire en nombres entiers. Comme l'ensemble d'itinéraires valides est impossible à énumérer en pratique, on a modélisé le sous-problème à l'aide d'un réseau de connexions où les nœuds représentent les trajets à couvrir et des visites aux stations de recharge et où les arêtes représentent des connexions valides.

On a ensuite décrit l'algorithme de génération de colonnes utilisé pour résoudre le MDEVSP. On a démontré que cet algorithme permettait de résoudre à l'optimalité le problème maître, mais qu'il pouvait être très coûteux en termes de temps de calcul. En conséquence, on a discuté des limitations de cet algorithme et de la dégénérescence. On a aussi présenté une technique de perturbation utilisée pour réduire les effets indésirables de la dégénérescence.

Une fois le MDEVSP et la méthode de résolution employée bien définis, on est entré dans le cœur de la contribution de ce mémoire. On a commencé par introduire les inégalités duales en précisant que celles-ci, si valides, permettaient de réduire l'espace de solutions duales et ainsi accélérer la résolution du problème. On a ensuite proposé une approche permettant de prédire des inégalités duales. Cette approche est basée sur un GNN et un MLP et utilise un réseau de connexions pour représenter le problème. Puisqu'on utilise de l'apprentissage machine pour prédire les inégalités duales, les chances d'introduire de fausses inégalités sont élevées. Pour cette raison, on a proposé une méthode permettant de réduire les erreurs potentielles de prédiction ainsi qu'un processus de recouvrement permettant de réparer les solutions contenant des inégalités duales invalides.

Afin d'évaluer l'approche proposée, on a présenté des résultats de différentes expériences effectuées sur des instances réalistes. Les résultats ont permis de conclure que les inégalités duales prédites semblent accélérer la résolution d'instances très dégénérées. En effet, on a remarqué que les instances qui subissent beaucoup de dégénérescence semblent être souvent accélérées grâce aux inégalités duales. De plus, on a démontré que le processus de recouvrement semblait effectivement réparer les solutions et améliorer la qualité de celles-ci lorsqu'il

était utilisé. Cependant, ce processus ralentit beaucoup la résolution, menant parfois à des temps de résolution beaucoup plus lents. Finalement, on a essayé de démontrer la généralisation de l'approche en l'appliquant sur des instances du SDEVSP. Les résultats ont permis de conclure que l'approche proposée semble aussi être efficace pour des instances très dégénérées du SDEVSP.

6.2 Limitations et améliorations potentielles

On présente maintenant quelques limitations de l'approche présentée dans ce mémoire ainsi que des pistes d'améliorations.

Tout d'abord, on a remarqué que les facteurs d'accélération variaient énormément. En effet, pour chaque expérience effectuée, la différence entre le meilleur et le pire facteur d'accélération était conséquent. Ceci n'est pas désirable, car pour certaines instances, ajouter des inégalités duales se traduit par une résolution du PM^{ID} plus lente que pour le PM. Idéalement, on aurait des facteurs d'accélération constants, quitte à ce qu'ils soient plus faibles, afin d'être convaincu qu'à chaque fois qu'on ajoute des inégalités duales, la résolution puisse être accélérée. Il serait intéressant d'analyser les inégalités duales ajoutées et d'autres types d'inégalités duales afin de déterminer lesquels aident à accélérer la résolution et lesquels nuisent. Une autre technique intéressante, utilisée par Gschwind et Irnich (2016), serait d'ajouter des inégalités duales dynamiquement.

Deuxièmement, on a aussi remarqué que le processus de recouvrement est très coûteux en termes de temps de calcul. En effet, on a réalisé que plus on effectue d'itérations du processus de recouvrement, meilleure la qualité de la solution est, mais plus la résolution est lente. On estime qu'il est acceptable que la résolution prenne plus de temps lorsqu'on utilise ce processus, mais celle-ci ne devrait pas être plus lente que la résolution du PM sans inégalités. En effet, lorsque c'est le cas, les inégalités duales ajoutées sont inutiles. On conclut donc que le processus de recouvrement pourrait être amélioré. Une possibilité serait d'essayer d'identifier quelles inégalités duales sont invalides selon la solution duale optimale originale. En effet, tel qu'expliqué à l'aide de la Figure 4.4, certaines inégalités duales sont invalides (actives) puisqu'une seconde inégalité duale est invalide. Idéalement, on serait en mesure de détecter uniquement les inégalités duales invalides qui retirent la solution duale optimale de l'espace de solution initial.

De plus, il serait intéressant d'essayer de prédire si une instance subira de la dégénérescence ou non lors de la résolution. On a réalisé que les instances qui ne subissent pas beaucoup de dégénérescence n'ont pas nécessairement besoin d'inégalités duales. En effet, lorsque le PM

d'une instance est résolu rapidement, l'ajout d'inégalités duales ralentit souvent la résolution. En pratique, il est impossible de déterminer à l'avance si une instance subira de la dégénérescence durant la résolution et donc, on ne peut pas décider à l'avance si on ajoute des inégalités duales ou pas. On pourrait essayer d'utiliser des $GNNs$ afin de prédire si une instance, représentée par un réseau de connexions, est sujet à subir de la dégénérescence. Ainsi, on pourrait par exemple, décider avant de lancer la résolution, si on ajoute des inégalités duales ou non.

Finalement, une suite potentielle à cette recherche serait de développer une stratégie de branchement à utiliser lorsqu'il reste des inégalités duales invalides qui induisent une solution primale non réalisable. En effet, tel qu'expliqué à la Section 4.5, lorsqu'on impose une limite d'itérations du processus de recouvrement, la solution primale finale peut ne pas couvrir exactement une fois chaque trajet. Afin d'être en mesure d'utiliser cette approche en pratique, il serait important de déterminer une règle de branchement permettant éventuellement d'obtenir une solution réalisable où chaque trajet est couvert exactement une fois.

il faudrait mentionner qu'une suite à ce projet pourrait être de développer la stratégie de branchement lorsqu'il reste des inégalités invalides qui induisent une solution primale non réalisable.

Bref, on estime que l'approche proposée dans ce mémoire a permis de confirmer que les inégalités duales ont le potentiel d'accélérer la résolution d'instances du MDEVSP qui subissent beaucoup de dégénérescence. Cependant, cette approche mérite d'être améliorée et étudiée davantage afin de pouvoir être utilisée en pratique. C'est pour cette raison qu'on a proposé quelques pistes d'améliorations potentielles.

RÉFÉRENCES

Alejandro Alvarez, Quentin Louveaux et Louis Wehenkel : A machine learning-based approximation of strong branching. *INFORMS Journal on Computing*, 29:185–195, 01 2017.

Richard Bellman : On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

Hatem Ben Amor, Jacques Desrosiers et José Carvalho : Dual-optimal inequalities for stabilized column generation. *Operations Research*, 54:454–463, 06 2006.

Hatem Ben Amor, Jacques Desrosiers et Antonio Frangioni : Stabilization in column generation. *Les Cahiers du GERAD ISSN*, 711:2440, 2004.

Pascal Benchimol, Guy Desaulniers et Jacques Desrosiers : Stabilized dynamic constraint aggregation for solving set partitioning problems. *European Journal of Operational Research*, 223(2):360–371, 2012.

Yoshua Bengio, Andrea Lodi et Antoine Prouvost : Machine learning for combinatorial optimization : a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

Alan A. Bertossi, Paolo Carraresi et Giorgio Gallo : On some matching problems arising in vehicle scheduling models. *Networks*, 17:271–281, 1987.

Jonathan 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 D.E.A., Polytechnique Montréal, août 2022.

Quentin Cappart, Didier Chételat, Elias Khalil, Andrea Lodi, Christopher Morris et Petar Veličković : Combinatorial optimization and reasoning with graph neural networks. *arXiv preprint arXiv :2102.09544*, 2021.

G. Carpaneto, Mauro Dell'Amico, Matteo Fischetti et Paolo Toth : A branch and bound algorithm for the multiple depot vehicle scheduling problem. *Networks*, 19:531 – 548, 08 1989.

José Carvalho : Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, 17:175–182, 05 2005.

Zhu Chao et Chen Xiaohong : Optimizing battery electric bus transit vehicle scheduling with battery exchanging : Model and case study. *Procedia - Social and Behavioral Sciences*, 96:2725–2736, 2013.

Kalyanmoy Deb, Samir Agrawal, Amrit Pratap et Tanaka Meyarivan : A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization : Nsga-ii. *In Parallel Problem Solving from Nature PPSN VI : 6th International Conference Paris*, pages 849–858. Springer, 2000.

Guy Desaulniers et Mark D. Hickman : Chapter 2 public transit. *In* Cynthia Barnhart et Gilbert Laporte, éditeurs : *Transportation*, volume 14 de *Handbooks in Operations Research and Management Science*, pages 69–127. Elsevier, 2007.

Jacques Desrosiers : Gencol : une équipe et un logiciel d'optimisation. *Stud. Inform. Univ.*, 8:61–96, 2010.

Jacques Desrosiers, Yvan Dumas, Marius M. Solomon et François Soumis : Chapter 2 time constrained routing and scheduling. *In Network Routing*, volume 8 de *Handbooks in Operations Research and Management Science*, pages 35–139. Elsevier, 1995.

DGL : Egatconv, 2023. URL `https://docs.dgl.ai/en/latest/generated/dgl.nn.pytorch.conv.EGATConv.html`.

Issmail Elhallaoui, Guy Desaulniers, Abdelmoutalib Metrane et François Soumis : Bi-dynamic constraint aggregation and subproblem reduction. *Computers & Operations Research*, 35(5):1713–1724, 2008.

Issmail Elhallaoui, Abdelmoutalib Metrane, François Soumis et Guy Desaulniers : Multi-phase dynamic constraint aggregation for set partitioning type problems. *Mathematical Programming*, 123:345–370, 2010.

Issmail Elhallaoui, Daniel Villeneuve, François Soumis et Guy Desaulniers : Dynamic aggregation of set-partitioning constraints in column generation. *Operations Research*, 53(4):632–645, 2005.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals et George E Dahl : Neural message passing for quantum chemistry. *In International conference on machine learning*, pages 1263–1272, 2017.

K. Gkiotsalitis, C. Iliopoulou et K. Kepaptsoglou : An exact approach for the multi-depot electric bus scheduling problem with time windows. *European Journal of Operational Research*, 306(1):189–206, 2023.

Timo Gschwind et Stefan Irnich : Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, 28(1):175–194, 2016.

Congcong Guo, Chunlu Wang et Xingquan Zuo : A genetic algorithm based column generation method for multi-depot electric bus vehicle scheduling. *In Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '19, page 367–368, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367486.

André Hottung, Shunji Tanaka et Kevin Tierney : Deep learning assisted heuristic tree search for the container pre-marshalling problem. *Computers & Operations Research*, 113:104781, 2020.

Stefan Irnich et Guy Desaulniers : Shortest Path Problems with Resource Constraints. *In* Guy Desaulniers, Jacques Desrosiers et Marius M. Solomon, éditeurs : *Column Generation*, Springer Books, pages 33–65. Springer, December 2005.

Kamil Kaminski, Jan Ludwiczak, Maciej Jasinski, Adriana Bukala, Rafal Madaj, Krzysztof Szczepaniak et Stanislaw Dunin-Horkawicz : Rossmann-toolbox : a deep learning-based protocol for the prediction and design of cofactor specificity in rossmann-fold proteins. *bioRxiv*, 2021.

Diederik P Kingma et Jimmy Ba : Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*, 2014.

Natalia Kliewer, Taïeb Mellouli et Leena Suhl : A time–space network based exact optimization model for multi-depot bus scheduling. *European Journal of Operational Research*, 175(3):1616–1627, 2006.

LaPresse : Mobilité durable : mieux se déplacer pour moins polluer. *La Presse*, Avril 2021.

Jing-Quan Li : Transit bus scheduling with limited energy. *Transportation Science*, 48:521–539, 11 2014.

Lu Li, Hong Lo et Feng Xiao : Mixed bus fleet scheduling under range and refueling constraints. *Transportation Research Part C Emerging Technologies*, 104:443–462, 07 2019.

Andreas Löbel : Solving large-scale multiple-depot vehicle scheduling problems. *Computer-aided transit scheduling*, 471:193–220, 1999.

Andrew L Maas, Awni Y Hannun, Andrew Y Ng *et al.* : Rectifier nonlinearities improve neural network acoustic models. *In Proceeding International Conference on Machine Learning*, volume 30, page 3. Atlanta, Georgia, USA, 2013.

Alejandro Montoya, Christelle Guéret, Jorge E. Mendoza et Juan G. Villegas : The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B : Methodological*, 103(C):87–110, 2017.

Amar Oukil, Hatem Ben Amor, Jacques Desrosiers et Hicham Gueddari : Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems. *Computers & Operations Research*, 34, 03 2007.

Max B Paulus, Giulia Zarpellon, Andreas Krause, Laurent Charlin et Chris Maddison : Learning to cut by looking ahead : Cutting plane selection via imitation learning. *In International conference on machine learning*, pages 17584–17600, 2022.

Samuel Pelletier, Ola Jabali, Gilbert Laporte et Marco Veneroni : Battery degradation and behaviour for electric vehicles : Review and numerical analyses of several models. *Transportation Research Part B : Methodological*, 103(C):158–187, 2017.

Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa et François Vanderbeck : Automation and combination of linear-programming based stabilization techniques in column generation. *INFORMS Journal on Computing*, 30(2):339–360, 2018.

Celso C. Ribeiro et François Soumis : A Column Generation Approach to the Multiple-Depot Vehicle Scheduling Problem. *Operations Research*, 42(1):41–52, February 1994.

Ons Sassi et Ammar Oulamara : Electric vehicle scheduling and optimal charging problem : Complexity, exact and heuristic approaches. *International Journal of Production Research*, 55:519–535, 03 2014.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner et Gabriele Monfardini : The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Marcel van Kooten Niekerk, J.M. Akker et J.A. Hoogeveen : Scheduling electric vehicles. *Public Transport*, 9, 07 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio et Yoshua Bengio : Graph attention networks. *arXiv preprint arXiv :1710.10903*, 2017.

Weitiao Wu, Yue Lin, Ronghui Liu et Wenzhou Jin : The multi-depot electric vehicle scheduling problem with power grid characteristics. *Transportation Research Part B : Methodological*, 155:322–347, 2022.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang et S Yu Philip : A comprehensive survey on graph neural networks. *IEEE Transactions On Neural Networks And Learning Systems*, 32(1):4–24, 2020.

Julian Yarkony, Naveed Haghani et Amelia Regan : Detour dual optimal inequalities for column generation with application to routing and location. *arXiv preprint arXiv :2108.09233*, 2021.

ANNEXE A RÉSULTATS DÉTAILLÉS - THÉORIQUES

			PM		PM^{ID}					
Taille	ID	$\|\mathcal{T}\|$	Nb. bus	Temps CPU (s)	Nb. bus	Temps CPU (s)	Acc.	Gap	Couverts	Inég.
M	0	879	53.997	379.0	53.997	152.8	2.48	0.00%	100.00%	1755
	1	768	53.997	414.0	53.997	35.9	11.53	0.00%	100.00%	1532
	2	882	43.998	498.1	43.998	106.5	4.68	0.00%	100.00%	1760
	3	833	55.997	343.2	55.997	49.0	7.00	0.00%	100.00%	1663
	4	857	59.997	107.7	59.997	42.3	2.55	0.00%	100.00%	1711
	5	837	50.998	804.0	50.998	96.3	8.35	0.00%	100.00%	1671
	6	859	47.998	1429.5	47.998	70.2	20.36	0.00%	100.00%	1715
	7	857	53.997	140.0	53.997	46.2	3.03	0.00%	100.00%	1711
	8	829	55.997	616.8	55.997	59.5	10.37	0.00%	100.00%	1655
	9	831	53.997	291.8	53.997	123.8	2.36	0.00%	100.00%	1659
	10	868	47.998	252.1	47.998	70.7	3.57	0.00%	100.00%	1733
	11	827	53.997	414.6	53.997	768.5	0.54	0.00%	100.00%	1651
	12	849	50.997	3583.7	50.997	63.9	56.08	0.00%	100.00%	1695
	13	803	50.997	392.3	50.997	48.9	8.02	0.00%	100.00%	1603
	14	870	56.997	176.1	56.997	56.4	3.12	0.00%	100.00%	1737
	15	741	47.998	177.0	47.998	30.2	5.86	0.00%	100.00%	1479
	16	889	57.997	242.1	57.997	66.3	3.65	0.00%	100.00%	1775
	17	925	56.997	189.1	56.997	77.4	2.44	0.00%	100.00%	1847
	18	764	47.998	440.5	47.998	31.5	13.98	0.00%	100.00%	1525
	19	825	45.997	1791.2	45.997	67.2	26.65	0.00%	100.00%	1647
	20	805	53.997	455.8	53.997	59.5	7.66	0.00%	100.00%	1607
	21	808	50.997	123.9	50.997	43.9	2.82	0.00%	100.00%	1613
	22	866	49.997	555.7	49.997	79.8	6.96	0.00%	100.00%	1728
	23	875	48.998	844.8	48.998	221.9	3.81	0.00%	100.00%	1747
	24	788	48.998	186.9	48.998	115.6	1.62	0.00%	100.00%	1573

			PM		PM^{ID}					
Taille	ID	$\|\mathcal{T}\|$	Nb. bus	Temps CPU (s)	Nb. bus	Temps CPU (s)	Acc.	Gap	Couverts	Inég.
G	0	1083	68.997	750.8	68.997	140.6	5.34	0.00%	100.00%	2163
	1	966	59.997	388.1	59.997	94.1	4.12	0.00%	100.00%	1929
	2	1088	63.996	426.2	63.996	160.1	2.66	0.00%	100.00%	2173
	3	1037	63.997	235.0	63.997	129.4	1.82	0.00%	100.00%	2071
	4	1026	68.997	402.5	68.997	75.8	5.31	0.00%	100.00%	2048
	5	1043	57.997	241.9	57.997	98.6	2.45	0.00%	100.00%	2083
	6	1063	61.997	367.4	61.997	125.8	2.92	0.00%	100.00%	2123
	7	1017	63.996	280.2	63.996	118.5	2.36	0.00%	100.00%	2031
	8	1004	63.997	254.2	63.997	134.1	1.90	0.00%	100.00%	2005
	9	1049	63.997	227.3	63.997	114.6	1.98	0.00%	100.00%	2095
	10	1067	61.997	22241.7	61.997	1872.2	11.88	0.00%	100.00%	2131
	11	1021	63.997	189.6	63.997	60.2	3.15	0.00%	100.00%	2039
	12	1050	63.997	252.6	63.997	94.4	2.68	0.00%	100.00%	2097
	13	1024	63.997	214.3	63.997	73.6	2.91	0.00%	100.00%	2045
	14	1075	69.996	219.8	69.996	100.9	2.18	0.00%	100.00%	2147
	15	952	56.997	316.5	56.997	86.9	3.64	0.00%	100.00%	1901
	16	1056	68.996	251.9	68.996	115.0	2.19	0.00%	100.00%	2109
	17	1123	68.997	292.0	68.997	125.7	2.32	0.00%	100.00%	2243
	18	955	50.997	646.0	50.997	95.5	6.76	0.00%	100.00%	1907
	19	1004	51.998	1957.3	51.998	97.7	20.03	0.00%	100.00%	2005
	20	985	59.997	278.5	59.997	64.1	4.34	0.00%	100.00%	1967
	21	1010	57.997	300.9	57.997	123.0	2.45	0.00%	100.00%	2016
	22	1026	63.997	271.1	63.997	89.8	3.02	0.00%	100.00%	2049
	23	1035	63.997	247.0	63.997	107.2	2.30	0.00%	100.00%	2067
	24	1003	57.997	257.8	57.997	85.1	3.03	0.00%	100.00%	2003

ANNEXE B RÉSULTATS DÉTAILLÉS - CONFIGURATION 1

			PM		PM^{ID}					
Taille	ID	$\|\mathcal{T}\|$	Nb. bus	Temps CPU (s)	Nb. bus	Temps CPU (s)	Acc.	Gap	Couverts	Inég.
	225	913	47.998	32106.0	47.998	3755.5	8.55	0.23%	90.42%	616
	226	876	47.997	359.1	47.997	286.8	1.25	0.14%	95.07%	633
	227	757	51.998	116.6	51.998	252.7	0.46	0.33%	93.32%	617
	228	830	53.997	156.8	53.997	110.0	1.43	0.05%	93.93%	649
	229	884	47.998	737.6	47.998	1010.2	0.73	0.05%	95.80%	589
	230	940	53.997	294.4	53.997	372.9	0.79	0.04%	95.63%	676
	231	904	47.998	936.5	47.998	498.6	1.88	0.21%	93.23%	617
	232	823	41.998	256.9	41.998	283.8	0.91	0.17%	92.40%	586
	233	797	53.997	492.3	53.997	454.9	1.08	0.10%	94.96%	650
	234	925	58.997	231.0	58.997	272.4	0.85	0.05%	95.23%	641
	235	715	55.997	99.2	55.997	89.6	1.11	0.09%	93.81%	635
	236	954	63.997	2234.5	63.997	407.1	5.49	0.01%	93.91%	709
M	237	902	51.997	410.6	51.997	270.4	1.52	0.05%	94.32%	637
	238	854	53.997	1120.8	53.997	280.5	4.00	0.08%	95.06%	600
	239	752	45.998	1555.8	44.998	887.9	1.75	1.82%	95.17%	547
	240	1051	61.997	11483.7	61.997	995.8	11.53	0.11%	91.61%	693
	241	878	53.997	1380.0	53.997	211.2	6.53	0.08%	93.84%	664
	242	909	53.997	396.1	53.997	167.6	2.36	0.20%	95.91%	620
	243	887	55.997	499.1	55.997	136.6	3.65	0.20%	92.65%	713
	244	882	63.997	170.3	63.997	136.5	1.25	0.20%	93.74%	643
	245	901	55.997	147.5	55.997	122.3	1.21	0.08%	94.33%	620
	246	836	53.997	526.6	53.997	320.5	1.64	0.12%	92.92%	660
	247	865	53.997	277.2	53.997	459.6	0.60	0.13%	93.85%	616
	248	848	53.997	572.8	53.997	527.4	1.09	0.26%	95.61%	652
	249	895	57.997	279.9	57.997	163.1	1.72	0.06%	92.95%	687

			PM		PM^{ID}					
Taille	ID	$\|\mathcal{T}\|$	Nb. bus	Temps CPU (s)	Nb. bus	Temps CPU (s)	Acc.	Gap	Couverts	Inég.
	225	1129	61.997	23174.6	61.997	23635.4	0.98	0.25%	93.61%	711
	226	1066	56.997	525.7	56.997	299.3	1.76	0.16%	95.75%	644
	227	966	63.997	440.6	63.997	161.2	2.73	0.04%	93.89%	748
	228	1052	64.997	959.0	64.997	760.2	1.26	0.03%	94.75%	766
	229	1105	61.997	21703.5	61.997	21753.0	1.00	0.10%	95.74%	666
	230	1194	61.997	21803.4	61.997	1114.6	19.56	0.09%	94.80%	707
	231	1139	63.997	24190.0	63.997	721.2	33.54	0.06%	94.20%	731
	232	975	57.997	625.0	57.997	1257.5	0.50	0.16%	94.43%	688
	233	1008	64.997	5049.6	64.997	895.2	5.64	0.08%	93.81%	760
	234	1107	68.996	2445.8	68.996	12681.6	0.19	0.18%	94.73%	767
	235	906	63.997	211.3	63.997	307.7	0.69	0.11%	94.56%	713
	236	1171	68.996	3524.7	68.996	2197.0	1.60	0.04%	93.83%	740
G	237	1151	63.997	21846.7	63.997	2349.3	9.30	0.10%	95.04%	707
	238	1075	65.997	23197.7	64.997	2554.7	9.08	1.50%	93.46%	738
	239	959	50.997	503.3	50.997	741.6	0.68	0.20%	94.23%	598
	240	1395	67.997	7727.8	67.997	4956.2	1.56	0.32%	94.82%	750
	241	1064	59.997	2424.1	59.997	1222.6	1.98	0.05%	93.88%	707
	242	1116	63.996	22737.6	63.996	372.7	61.01	0.13%	94.70%	727
	243	1072	63.997	1776.7	63.997	1449.7	1.23	0.11%	93.36%	747
	244	1063	68.997	4746.9	68.997	1540.0	3.08	0.07%	92.05%	709
	245	1090	63.997	547.1	63.996	1265.8	0.43	0.06%	94.30%	719
	246	1050	69.997	361.8	69.997	219.8	1.65	0.07%	96.09%	722
	247	1039	68.997	817.6	68.997	1143.5	0.71	0.07%	94.02%	800
	248	1055	68.997	256.7	68.996	383.2	0.67	0.09%	91.72%	712
	249	1094	63.997	302.5	63.997	305.9	0.99	0.08%	95.05%	740

ANNEXE C RÉSULTATS DÉTAILLÉS - CONFIGURATION 2

			PM		PM^{ID}					
Taille	ID	$\|\mathcal{T}\|$	Nb. bus	Temps CPU (s)	Nb. bus	Temps CPU (s)	Acc.	Gap	Couverts	Inég.
	225	913	47.998	32106	47.998	13994.2	2.29	0.05%	97.37%	617
	226	876	47.997	359.1	47.997	489.3	0.73	0.03%	95.55%	635
	227	757	51.998	116.6	51.998	726.8	0.16	0.06%	95.75%	619
	228	830	53.997	156.8	53.997	376.5	0.42	0.01%	96.85%	649
	229	884	47.998	737.6	47.998	1792.8	0.41	0.01%	96.83%	591
	230	940	53.997	294.4	53.997	597.8	0.49	0.02%	98.40%	675
	231	904	47.998	936.5	47.998	1072.5	0.87	0.06%	96.67%	618
	232	823	41.998	256.9	41.998	303.8	0.85	0.05%	96.60%	586
	233	797	53.997	492.3	53.997	445	1.11	0.04%	95.98%	649
	234	925	58.997	231	58.997	1901.8	0.12	0.01%	98.70%	642
	235	715	55.997	99.2	55.997	99.8	0.99	0.04%	96.22%	631
	236	954	63.997	2234.5	63.997	1514.5	1.48	0.00%	96.12%	710
M	237	902	51.997	410.6	51.997	712	0.58	0.02%	96.01%	638
	238	854	53.997	1120.8	53.997	1030.7	1.09	0.02%	96.96%	600
	239	752	45.998	1555.8	44.998	578.2	2.69	1.67%	98.54%	547
	240	1051	61.997	11483.7	61.997	3471	3.31	0.06%	96.19%	693
	241	878	53.997	1380	53.997	1389.6	0.99	0.04%	96.58%	664
	242	909	53.997	396.1	53.997	583.5	0.68	0.05%	95.93%	619
	243	887	55.997	499.1	55.997	201.1	2.48	0.15%	96.50%	713
	244	882	63.997	170.3	63.997	681.3	0.25	0.12%	95.80%	644
	245	901	55.997	147.5	55.997	221.5	0.67	0.03%	97.56%	621
	246	836	53.997	526.6	53.997	254.8	2.07	0.02%	92.81%	659
	247	865	53.997	277.2	53.997	675.3	0.41	0.01%	96.53%	615
	248	848	53.997	572.8	53.997	5399.2	0.11	0.16%	95.50%	651
	249	895	57.997	279.9	57.997	439.4	0.64	0.01%	95.53%	687

			PM		PM^{ID}					
Taille	ID	$\|\mathcal{T}\|$	Nb. bus	Temps CPU (s)	Nb. bus	Temps CPU (s)	Acc.	Gap	Couverts	Inég.
	225	1129	61.997	23174.6	61.997	28497.6	0.81	0.08%	96.28%	711
	226	1066	56.997	525.7	56.997	1344.3	0.39	0.05%	97.47%	644
	227	966	63.997	440.6	63.997	443.2	0.99	0.01%	97.00%	748
	228	1052	64.997	959	64.997	415.7	2.31	0.01%	96.48%	766
	229	1105	61.997	21703.5	61.997	22656.7	0.96	0.03%	96.47%	666
	230	1194	61.997	21803.4	61.997	11883.8	1.83	0.03%	97.57%	707
	231	1139	63.997	24190	63.997	1736.5	13.93	0.01%	97.72%	730
	232	975	57.997	625	57.997	1601.5	0.39	0.01%	96.41%	688
	233	1008	64.997	5049.6	64.997	2401	2.1	0.01%	95.44%	760
	234	1107	68.996	2445.8	68.996	6568.2	0.37	0.09%	96.56%	767
	235	906	63.997	211.3	63.997	367.1	0.58	0.08%	96.12%	712
	236	1171	68.996	3524.7	68.996	5662	0.62	0.01%	95.90%	738
G	237	1151	63.997	21846.7	63.997	21682.2	1.01	0.04%	96.09%	706
	238	1075	65.997	23197.7	64.997	22496.1	1.03	1.37%	92.83%	737
	239	959	50.997	503.3	50.997	1476.5	0.34	0.05%	96.35%	597
	240	1395	67.997	7727.8	67.997	5505	1.4	0.18%	95.19%	749
	241	1064	59.997	2424.1	59.997	3298.9	0.73	0.01%	95.11%	708
	242	1116	63.996	22737.6	63.996	22758.5	1	0.04%	97.13%	727
	243	1072	63.997	1776.7	63.997	13132.8	0.14	0.03%	95.05%	748
	244	1063	68.997	4746.9	68.997	4811.6	0.99	0.01%	96.99%	709
	245	1090	63.997	547.1	63.996	1407.5	0.39	0.01%	97.52%	720
	246	1050	69.997	361.8	69.997	359.2	1.01	0.02%	97.71%	723
	247	1039	68.997	817.6	68.997	1229.3	0.67	0.02%	97.31%	800
	248	1055	68.997	256.7	68.996	481.2	0.53	0.04%	95.83%	714
	249	1094	63.997	302.5	63.997	868	0.35	0.03%	97.44%	743

ANNEXE D RÉSULTATS DÉTAILLÉS - CONFIGURATION 3

			PM		PM^{ID}					
Taille	ID	$\|\mathcal{T}\|$	Nb. bus	Temps CPU (s)	Nb. bus	Temps CPU (s)	Acc.	Gap	Couverts	Inég.
	225	913	47.998	32106	47.998	823.3	39	0.14%	98.57%	615
	226	876	47.997	359.1	47.997	772.4	0.46	0.03%	98.63%	633
	227	757	51.998	116.6	51.998	144.1	0.81	0.15%	96.16%	618
	228	830	53.997	156.8	53.997	179.5	0.87	0.00%	98.43%	648
	229	884	47.998	737.6	47.998	924.3	0.8	0.00%	99.55%	592
	230	940	53.997	294.4	53.997	258.8	1.14	0.02%	99.57%	676
	231	904	47.998	936.5	47.998	1000.9	0.94	0.07%	98.01%	617
	232	823	41.998	256.9	41.998	203.4	1.26	0.04%	98.05%	585
	233	797	53.997	492.3	53.997	189.6	2.6	0.03%	99.50%	650
	234	925	58.997	231	58.997	634.9	0.36	0.00%	99.57%	641
	235	715	55.997	99.2	55.997	88.2	1.12	0.03%	98.04%	632
	236	954	63.997	2234.5	63.997	697.6	3.2	0.00%	100.00%	710
M	237	902	51.997	410.6	51.997	903.6	0.45	0.02%	98.00%	637
	238	854	53.997	1120.8	53.997	1562.7	0.72	0.04%	99.06%	599
	239	752	45.998	1555.8	44.998	312.8	4.97	1.69%	98.53%	548
	240	1051	61.997	11483.7	61.997	1723.8	6.66	0.05%	98.86%	693
	241	878	53.997	1380	53.997	3553.1	0.39	0.01%	99.77%	664
	242	909	53.997	396.1	53.997	233.6	1.7	0.10%	98.68%	620
	243	887	55.997	499.1	55.997	225.4	2.21	0.14%	93.91%	713
	244	882	63.997	170.3	63.997	176.5	0.96	0.15%	97.85%	644
	245	901	55.997	147.5	55.997	178	0.83	0.03%	99.22%	618
	246	836	53.997	526.6	53.997	952.9	0.55	0.03%	93.78%	659
	247	865	53.997	277.2	53.997	288.5	0.96	0.02%	99.08%	616
	248	848	53.997	572.8	53.997	283.3	2.02	0.16%	95.39%	650
	249	895	57.997	279.9	57.997	410.2	0.68	0.01%	98.55%	687

			PM		PM^{ID}					
Taille	ID	$\|\mathcal{T}\|$	Nb. bus	Temps CPU (s)	Nb. bus	Temps CPU (s)	Acc.	Gap	Couverts	Inég.
	225	1129	61.997	23174.6	61.997	21994.3	1.05	0.15%	98.58%	711
	226	1066	56.997	525.7	56.997	850.9	0.62	0.06%	99.44%	643
	227	966	63.997	440.6	63.997	293.7	1.5	0.01%	99.59%	749
	228	1052	64.997	959	64.997	1206.8	0.79	0.00%	99.71%	766
	229	1105	61.997	21703.5	61.997	21605.2	1	0.05%	99.28%	664
	230	1194	61.997	21803.4	61.997	10582.1	2.06	0.05%	99.83%	706
	231	1139	63.997	24190	63.997	1015.3	23.83	0.01%	98.77%	730
	232	975	57.997	625	57.997	54370.1	0.01	0.09%	98.97%	689
	233	1008	64.997	5049.6	64.997	1417.1	3.56	0.01%	99.60%	760
	234	1107	68.996	2445.8	68.996	8097.7	0.3	0.10%	98.19%	765
	235	906	63.997	211.3	63.997	309.7	0.68	0.07%	96.68%	712
	236	1171	68.996	3524.7	68.996	8168.3	0.43	0.00%	99.57%	739
G	237	1151	63.997	21846.7	63.997	1523.3	14.34	0.06%	99.04%	707
	238	1075	65.997	23197.7	64.997	22081.7	1.05	1.42%	97.86%	737
	239	959	50.997	503.3	50.997	908.7	0.55	0.09%	98.43%	598
	240	1395	67.997	7727.8	67.997	11958.6	0.65	0.19%	98.35%	749
	241	1064	59.997	2424.1	59.997	1052.1	2.3	0.01%	98.87%	707
	242	1116	63.996	22737.6	63.996	590.9	38.48	0.06%	99.37%	726
	243	1072	63.997	1776.7	63.997	959.7	1.85	0.01%	97.85%	748
	244	1063	68.997	4746.9	68.997	757.7	6.26	0.01%	98.87%	711
	245	1090	63.997	547.1	63.996	422.6	1.29	0.01%	99.72%	720
	246	1050	69.997	361.8	69.997	507.1	0.71	0.01%	98.48%	723
	247	1039	68.997	817.6	68.997	426.8	1.92	0.02%	99.42%	800
	248	1055	68.997	256.7	68.996	337.8	0.76	0.06%	98.86%	714
	249	1094	63.997	302.5	63.997	510.8	0.59	0.03%	98.99%	740

ANNEXE E RÉSULTATS DÉTAILLÉS - SDEVSP

			PM		PM^{ID}					
Taille	ID	$\|\mathcal{T}\|$	Nb. bus	Temps CPU (s)	Nb. bus	Temps CPU (s)	Acc.	Gap	Couverts	Inég.
	225	913	46.998	22570.4	46.998	21603.7	1.04	0.16%	95.57%	378
	226	876	46.997	377.0	46.997	509.3	0.74	0.01%	99.32%	330
	227	757	50.998	20251.8	50.998	157.2	128.83	0.13%	95.88%	342
	228	830	52.997	371.1	52.997	155.4	2.39	0.06%	95.27%	353
	229	884	46.998	21761.7	46.998	696.4	31.25	0.09%	96.94%	358
	230	940	52.997	303.7	52.997	289.0	1.05	0.06%	96.17%	343
	231	904	46.998	1860.3	46.998	351.8	5.29	0.07%	96.89%	361
	232	823	40.998	184.8	40.998	1276.7	0.14	0.15%	96.45%	335
	233	797	52.997	1807.5	52.997	269.5	6.71	0.03%	96.35%	352
	234	925	57.997	21709.1	57.997	2898.6	7.49	0.04%	96.09%	355
	235	715	54.997	645.1	53.997	186.1	3.47	1.68%	98.17%	337
	236	954	62.997	465.0	61.997	1166.0	0.40	1.47%	96.10%	356
M	237	902	50.997	767.8	50.997	5436.7	0.14	0.12%	97.10%	358
	238	854	52.997	21602.1	52.997	533.4	40.50	0.08%	96.36%	368
	239	752	44.998	3193.2	43.998	224.7	14.21	2.13%	96.38%	293
	240	1051	60.997	607.9	60.997	792.5	0.77	0.13%	97.14%	354
	241	878	52.997	417.0	52.997	211.5	1.97	0.07%	96.91%	347
	242	909	52.997	514.7	52.997	387.1	1.33	0.16%	95.03%	390
	243	887	54.997	229.8	54.997	355.2	0.65	0.18%	95.69%	341
	244	882	62.997	1002.6	62.997	236.1	4.25	0.05%	96.24%	343
	245	901	54.997	207.8	54.997	209.8	0.99	0.23%	93.71%	368
	246	836	52.997	342.5	52.997	199.9	1.71	0.13%	95.44%	361
	247	865	52.997	197.0	52.997	622.9	0.32	0.03%	97.44%	340
	248	848	52.997	23838.4	52.997	497.2	47.95	0.09%	96.80%	370
	249	895	56.997	590.0	56.997	390.5	1.51	0.05%	96.52%	355

			PM		PM^{ID}					
Taille	ID	$\|\mathcal{T}\|$	Nb. bus	Temps CPU (s)	Nb. bus	Temps CPU (s)	Acc.	Gap	Couverts	Inég.
	225	1129	60.997	35244.5	60.997	2106.2	16.73	0.05%	98.23%	381
	226	1066	55.997	2095.4	55.997	22489.6	0.09	0.09%	97.92%	367
	227	966	62.997	1704.7	62.997	1075.7	1.58	0.06%	96.66%	367
	228	1052	63.997	3026.9	63.997	555.7	5.45	0.08%	98.19%	358
	229	1105	60.997	21616.5	60.997	26854.5	0.80	0.08%	96.27%	380
	230	1194	60.997	21603.6	60.997	21602.6	1.00	0.04%	98.32%	380
	231	1139	62.997	867.4	62.997	21613.2	0.04	0.14%	96.90%	401
	232	975	56.997	784.4	56.997	604.2	1.30	0.07%	97.13%	353
	233	1008	63.997	1872.9	63.997	927.5	2.02	0.07%	98.01%	359
	234	1107	67.996	616.8	67.996	4982.8	0.12	0.02%	97.64%	384
	235	906	62.997	919.6	62.997	242.0	3.80	0.06%	97.23%	359
	236	1171	67.996	1249.8	67.996	9304.1	0.13	0.06%	96.83%	421
G	237	1151	62.997	30317.0	62.997	21777.6	1.39	0.05%	97.13%	350
	238	1075	64.997	21929.5	63.997	24707.6	0.89	1.42%	96.65%	392
	239	959	49.997	540.7	49.997	552.7	0.98	0.09%	97.59%	365
	240	1395	69.995	21604.5	68.901	21602.6	1.00	1.03%	96.33%	365
	241	1064	58.997	615.0	58.997	13114.7	0.05	0.12%	97.27%	384
	242	1116	62.997	651.0	62.997	963.6	0.68	0.14%	95.05%	405
	243	1072	62.997	3666.2	62.997	949.0	3.86	0.08%	97.20%	349
	244	1063	67.997	53572.7	67.997	1247.8	42.93	0.05%	97.74%	368
	245	1090	62.997	706.0	62.996	790.6	0.89	0.07%	96.21%	398
	246	1050	68.997	934.4	68.997	425.2	2.20	0.04%	96.36%	387
	247	1039	67.997	5214.5	67.997	4274.8	1.22	0.09%	97.48%	373
	248	1055	67.997	1288.2	67.997	720.5	1.79	0.03%	97.34%	383
	249	1094	62.997	464.5	62.997	491.0	0.95	0.10%	97.33%	383