Titre: Title:	Stabilisation de la génération de colonnes par chaînes d'inégalités duales : Application au problème d'horaires d'autobus électriques avec dépôts multiples
Auteur: Author:	Antoine Demers-Bergeron
Date:	2025
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Demers-Bergeron, A. (2025). Stabilisation de la génération de colonnes par chaînes d'inégalités duales : Application au problème d'horaires d'autobus électriques avec dépôts multiples [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/68824/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Stabilisation de la génération de colonnes par chaînes d'inégalités duales : Application au problème d'horaires d'autobus électriques avec dépôts multiples

ANTOINE DEMERS-BERGERON
Département de mathématiques et de génie industriel

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

Juin 2025

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Stabilisation de la génération de colonnes par chaînes d'inégalités duales : Application au problème d'horaires d'autobus électriques avec dépôts multiples

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

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

DÉDICACE

À Charles...

REMERCIEMENTS

Merci à Daphnée, ma copine, d'avoir été présente et de m'avoir soutenu tout au long de ce processus. Les deux dernières années ont été chargées et stressantes. Je suis reconnaissant de t'avoir eu à mes côtés à travers tout ça.

Merci à ma famille et mes amis. Même éparpillés aux quatre coins de la province, votre amour et votre soutien m'ont beaucoup aidé tout au long de mes études.

Je tiens à terminer avec un énorme merci à mon directeur de recherche Guy Desaulniers qui a été pour moi un guide et un mentor hors pair. Merci de m'avoir offert cette opportunité, à moi qui ne connaissais rien à l'optimisation ou à la recherche opérationnelle. J'ai beaucoup appris dans les deux dernières années et j'emporte dans ma nouvelle vie professionnelle tout le bagage que tu m'as transmis. Merci encore pour ton écoute et pour ton dévouement envers tes étudiants (même si je suis d'avis que tu devrais prendre plus de temps pour toi) et au plaisir de te recroiser dans les bureaux de GIRO lorsque tu viendras y faire un tour.

RÉSUMÉ

L'intégration rapide des autobus électriques dans les réseaux de transport en commun introduit de nouveaux défis, tels qu'une autonomie limitée et des temps de recharge prolongés. Ces contraintes compliquent le problème d'horaires d'autobus électriques avec dépôts multiples (MDEVSP), qui consiste à assigner un ensemble de trajets à des autobus provenant de différents dépôts tout en assurant une gestion adéquate du niveau de batterie. Le MDEVSP est souvent résolu à l'aide de la génération de colonnes, une méthode qui souffre toutefois de dégénérescence, i.e., que la valeur de la solution courante n'évolue pas nécessairement à chaque itération. Cette dégénérescence provient du fait qu'il y a plusieurs solutions duales associées à la même solution primale. Ainsi, une façon de limiter la dégénérescence consiste à réduire l'espace dual en ajoutant des inégalités duales au modèle.

Dans ce mémoire, nous poursuivons les travaux de Popovic (2023) qui a développé une approche basée sur l'apprentissage machine pour prédire des chaînes d'inégalités duales et les intégrer au problème maître de la génération de colonnes afin de stabiliser les variables duales, réduire la dégénérescence et accélérer la convergence. Nous poussons plus loin ces idées qui ont été testées sur la relaxation linéaire du problème et nous les appliquons sur la résolution complète en appliquant une stratégie de branchement heuristique adaptée. Bien que cette méthode montre un potentiel intéressant, nos résultats expérimentaux révèlent des améliorations limitées dans les cas testés. Par la suite, nous menons une analyse détaillée des conditions nécessaires à l'efficacité de ces inégalités et identifions les principales limites de l'approche par apprentissage machine. À partir de ces constats, nous terminons en proposant une méthode alternative consistant à construire une longue chaîne d'inégalités duales à partir de la solution duale de la première relaxation linéaire. Cette stratégie permet d'obtenir jusqu'à 30% d'accélération sur l'ensemble des instances testées, tout en offrant une solution plus fiable et plus facile à implémenter.

ABSTRACT

The rapid integration of electric buses into public transportation networks introduces new challenges, such as limited range and extended charging times. These constraints complicate the Multi-Depot Electric Vehicle Scheduling Problem (MDEVSP), which requires assigning trips to buses from various depots while managing battery levels. The MDEVSP is often solved using column generation, but this technique suffers from degeneracy, i.e. that the current solution does not necessarily evolve at every iteration. This degeneracy comes from the fact that there are many dual solutions associated with the same primal solution. Therefore, one way to limit degeneracy consists in reducing the dual space by adding dual inequalities to the model.

In this thesis, we build upon the work of Popovic (2023) who developed a machine-learning-based approach to predict dual inequalities and incorporate them into the master problem of the column generation to stabilize dual variables, reduce degeneracy, and accelerate the convergence of the algorithm. We take these ideas that were tested on the linear relaxation of the problem, and we apply them on the complete resolution by using an adapted heuristic branching strategy. Although the method shows theoretical potential, our results indicate limited performance improvements in the tested scenarios. We conduct a detailed analysis of the structural conditions required for these dual inequalities to be effective and identify key limitations of the machine-learning-based approach. From this insight, we propose an alternative method that constructs a long chain of dual inequalities from the dual solution of the first linear relaxation. This strategy achieves up to 30% acceleration across all tested instances and offers a more reliable and easier-to-implement solution.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

MDVSP	Problème d'horaire de véhicules avec dépôts multiples
MDEVSP	Problème d'horaire de véhicules électriques avec dépôts multiples
GNN	Réseau de neurones graphique
PMNE	Problème maître en nombres entiers
PM	Problème maître
PMR	Problème maître réduit
PM^{ID}	Problème maître avec inégalités duales
MIP	Modélisation en nombres entiers mixtes (*Mixed integer programming*)

CHAPITRE 1 INTRODUCTION

L'électrification des transports est un enjeu majeur dans la sphère politique actuelle. Dans son *Plan pour une économie verte 2030* publié en 2020, le Gouvernement du Québec mentionne que le secteur des transports est le "*principal secteur émetteur de gaz à effet de serre et un secteur fortement dépendant des énergies fossiles importées*" (Gouvernement du Québec, 2020). En effet, selon ce même rapport, en 2017, 43.3% des émissions de gaz à effet de serre étaient émises par le secteur des transports, ce qui en faisait le premier secteur émetteur. De plus, on estime que 79.6% de ces émissions sont attribuables au transport routier. C'est pourquoi, dans son plan, le Gouvernement du Québec prévoit investir massivement dans l'électrification des transports, notamment dans le secteur des transports collectifs. Les cibles sont que d'ici à 2030, "les autobus électriques représentent 55% du parc total d'autobus urbains" et "*65% de l'ensemble des autobus scolaires en circulation*"(Gouvernement du Québec, 2020). Pour atteindre ces objectifs, en 2023, un groupe de ministres annonçait d'ailleurs "*des investissements de plus de 780 millions de dollars du gouvernement du Canada et de plus de 1,1 milliard de dollars du gouvernement du Québec pour l'acquisition de 1229 autobus électriques et l'amélioration du réseau de transport collectif électrique au Québec*" (Cabinet de la vice-première ministre et ministre des Transports et de la Mobilité durable, 2023). Mentionnons également que si l'on additionne les montants prévus par les principales sociétés de transport collectif de la province, c'est près de 13,1 milliards de dollars qui seront investis au Québec dans l'électrification du réseau sur les dix prochaines années (Chouinard, 2024). Évidemment, des changements aussi importants et rapides apportent leur lot d'enjeux logistiques et c'est dans cette logistique entourant l'implantation d'une flotte d'autobus électriques que les contributions de ce mémoire se situent.

La planification des opérations d'une société de transport en commun comporte plusieurs étapes au cours desquelles des méthodes de recherche opérationnelle peuvent être appliquées. Par exemple, pour l'implantation d'une flotte d'autobus électriques, on peut séparer cette planification en cinq étapes (Desaulniers et Hickman (2007), Zhang *et al.* (2023)) :

1. Élaboration des lignes d'autobus
2. Déploiement des infrastructures de recharge
3. Conception de l'horaire des trajets sur chaque ligne
4. Construction des horaires d'autobus
5. Détermination des horaires des chauffeurs d'autobus

Dans ce mémoire, on s'intéressera à l'étape 4, la planification des horaires d'autobus électriques. Nous en parlerons plus exhaustivement un peu plus loin, mais le problème d'horaires d'autobus à essence avec dépôts multiples (MDVSP) est un problème bien documenté et étudié depuis longtemps. Étant donné un ensemble de trajets à couvrir et une flotte d'autobus par dépôt, il consiste à déterminer des horaires réalisables (suites de trajets débutant et finissant au même dépôt) de façon à ce que chaque trajet soit couvert par un véhicule tout en respectant la disponibilité des autobus par dépôt et en minimisant les coûts d'opérations. Par contre, l'arrivée des autobus électriques dans les réseaux de transports collectifs vient grandement complexifier ce problème, ce qui force l'industrie à revoir ses méthodes et à développer de nouveaux outils plus efficaces pour résoudre ces problèmes. En effet, lorsque l'on passe du MDVSP au problème d'horaires d'autobus électriques avec dépôts multiples (MDEVSP), on doit jongler avec une autonomie moins grande, des temps de recharge lents, un nombre de bornes de recharge limité et, dans certains cas, une capacité limitée du réseau électrique.

Pour résoudre le MDEVSP, la méthode la plus répandue est la génération de colonnes qui s'appuie sur un modèle impliquant des variables d'horaires d'autobus. À partir d'un modèle restreint à un petit sous-ensemble d'horaires réalisables, elle ajoute progressivement à ce modèle de nouvelles variables identifiées en résolvant un sous-problème jusqu'à atteindre l'optimalité de la relaxation linéaire. Une solution est ensuite obtenue en imbriquant cette méthode de génération de colonnes dans un algorithme de *branch-and-bound*. Lorsque les instances sont très grandes et contiennent un très grand nombre de trajets et d'autobus, il risque d'y avoir un grand nombre de variables prenant une valeur nulle dans la base, plusieurs solutions de base réalisables et, donc, plusieurs solutions duales associées, ce qui peut créer de la dégénérescence, i.e., que la valeur de la solution courante ne s'améliore pas nécessairement avec les itérations. Cette dégénérescence ralentit grandement la convergence de la génération de colonnes et allonge les temps de calcul.

1.1 Objectifs de recherche

Plusieurs techniques peuvent être employées pour combattre la dégénérescence. Dans ce mémoire, nous nous intéressons aux travaux de Popovic (2023) qui s'est attaqué au problème de la dégénérescence dans la génération de colonnes en introduisant des inégalités duales prédites par apprentissage automatique et a obtenu de bons résultats pour l'accélération de la résolution de la relaxation linéaire au noeud 0. Notre objectif est de pousser cette idée plus loin en appliquant ces inégalités duales pour la résolution complète du problème, en explorant davantage leur potentiel et en définissant plus clairement les conditions nécessaires

pour qu'elles aient un impact important sur l'accélération de la génération de colonnes dans le contexte du MDEVSP.

1.2 Plan du mémoire

La suite de ce mémoire est divisée en cinq chapitres. Nous commençons par une revue de littérature qui fait un survol des connaissances concernant le MDVSP, le MDEVSP, les méthodes de gestion de la dégénérescence et le potentiel des inégalités duales dans ce domaine. Nous survolons ensuite quelques types de stratégies de branchement et quelques heuristiques de recherche de solution entière. Nous terminons la revue de littérature en passant en revue les différentes contributions de Popovic (2023) pour poser les bases sur lesquelles nos travaux reposent.

Le chapitre suivant présente une définition détaillée du problème, le modèle mathématique utilisé et une méthode de résolution de base. Nous présentons le MDEVSP, les contraintes qui y sont associées, puis nous formulons le programme linéaire en nombres entiers qui sera utilisé pour la suite des travaux. Dans ce chapitre, nous décrivons également la génération de colonnes, ainsi que le sous-problème qui s'y rattache, puis nous présentons une heuristique de recherche de solution entière et différents types de branchement.

Le chapitre 4 porte sur la gestion de la dégénérescence. C'est dans ce chapitre que nous présentons en détail toutes les méthodes que nous utilisons pour combattre la dégénérescence. Nous présentons la perturbation, puis nous faisons un retour sur la stratégie de prédiction d'inégalités duales présentée par Popovic (2023) et terminons en présentant un processus de réparation servant à retrouver une solution réalisable dans le cas où de mauvaises inégalités y sont ajoutées.

Le chapitre 5 concerne les résultats expérimentaux obtenus et leur analyse. On y voit que la stratégie proposée initialement ne nous permet pas d'obtenir d'amélioration intéressante des temps de calcul pour le MDEVSP, mais nous poursuivons avec une analyse approfondie des chaînes d'inégalités duales. Nous expliquons les faiblesses de la prédiction d'inégalités duales par apprentissage machine dans notre contexte tout en démontrant qu'elle a toujours un potentiel important pour la stabilisation et l'accélération de la génération de colonnes. Nous terminons en présentant une méthode alternative n'utilisant pas d'apprentissage machine, mais exploitant plutôt la solution de la relaxation linéaire du problème maître en nombres entiers pour générer une chaîne d'inégalités duales.

Finalement, nous tirons nos conclusions au chapitre 6. Nous faisons une synthèse des travaux et un rappel des idées proposées et des résultats obtenus, et nous terminons en mentionnant

quelques pistes qui mériteraient d'être explorées pour rendre la prédiction d'inégalités duales viable et efficace.

CHAPITRE 2 REVUE DE LITTÉRATURE

Dans ce chapitre, nous ferons un résumé des connaissances sur les différents aspects importants de ce mémoire. Nous commencerons par décrire ce que l'on connaît du MDVSP et du MDEVSP, puis nous aborderons les différentes méthodes développées pour contrer la dégénérescence et nous terminerons en décrivant les progrès faits par Popovic (2023) ainsi que les contributions du présent mémoire.

2.1 Problème d'horaires d'autobus avec dépôts multiples

Cette section présente une revue des méthodes de résolution du MDVSP à travers les années. Nous commencerons par discuter du cas avec autobus à essence, puis nous présenterons la littérature concernant le cas avec autobus électriques.

2.1.1 Problème d'horaires d'autobus à essence avec dépôts multiples

Le MDVSP vise à assigner des véhicules à un ensemble de trajets donné où chaque trajet doit avoir lieu à une certaine heure tout en considérant certaines contraintes pratiques comme le nombre de dépôts, les types de véhicules disponibles et la capacité des dépôts (Kliewer *et al.*, 2006). Une solution optimale sera donc caractérisée par une quantité minimale de véhicules et un minimum de temps d'attente de façon à réduire les coûts d'opération. Ce problème, démontré comme NP-difficile par Bertossi *et al.* (1987), a été résolu de multiples façons au cours des 40 dernières années.

Par exemple, Carpaneto *et al.* (1989) introduisent un modèle de réseau dans lequel on a un nœud pour chaque dépôt, un nœud pour chaque trajet à couvrir et un arc pour chaque connexion réalisable. Le MDVSP devient donc un problème où l'on doit trouver un ensemble de circuits contenant un seul nœud *dépôt* tel que tous les nœuds *trajet* sont visités exactement une fois et que la somme des coûts des arcs est minimale. Ils introduisent ensuite un critère de dominance reposant sur l'idée que l'on peut élaguer un nœud α de l'arbre de branchement s'il existe un autre nœud β tel que $c^\beta < c^\alpha$ et $F^\beta \subseteq F^\alpha$ (où c^i est le coût de la solution au nœud i et F^i représente l'ensemble des arcs "interdits" au nœud i), puis proposent de résoudre le MDVSP à l'aide d'un algorithme de *branch and bound*.

Par la suite, Ribeiro et Soumis (1994) développent une nouvelle formulation en modélisant le MDVSP comme un problème de partitionnement d'ensemble et montrent que cette nouvelle formulation permet d'obtenir des bornes inférieures plus serrées et de résoudre des problèmes

4 à 5 fois plus grands qu'avec les méthodes utilisées jusque-là. Ils utilisent ensuite une approche de génération de colonnes insérée dans un algorithme de *branch and bound* pour résoudre le problème. Notons également que les logiciels de GENCOL et CPLEX utilisés dans cet article sont des versions antérieures de ce qui sera utilisé dans ce mémoire. Quelques années plus tard, Desaulniers *et al.* (1998) poussent ces idées un peu plus loin et considèrent le cas avec fenêtres de temps. Pour simplifier le problème, ils proposent d'approximer le temps d'attente entre deux tâches par le temps d'attente minimal, permettant de résoudre le problème avec une structure se rapprochant du cas sans fenêtres de temps.

De leur côté, Hadjar *et al.* (2006) reprennent plusieurs idées de ce dernier article, mais complexifient la stratégie de recherche de solutions entières en utilisant, en ordre de priorité, des plans coupants, une *colour partitioning method* et une méthode de fixation d'inter-tâches. La *colour partitioning method* sépare, pour chaque trajet, l'ensemble des dépôts en deux sous-ensembles de taille égale (si possible) et force le trajet à être couvert par des itinéraires débutant seulement à des dépôts venant de l'un des deux sous-ensembles. Ils utilisent ensuite une stratégie de type *best-then-depth* qui alterne entre des stratégies "meilleur d'abord" et "profondeur d'abord" tout en ajoutant des plans coupants avant chaque branchement, permettant d'atteindre rapidement une solution entière.

Enfin, Kliewer *et al.* (2006) proposent un réseau espace-temps pour modéliser le MDVSP, réduisant considérablement la taille du problème par rapport à d'autres modélisations comme les modèles de partitionnement d'ensemble et les modèles de réseau de connexions. On invite le lecteur à consulter Desaulniers et Hickman (2007) et Ibarra-Rojas *et al.* (2015) pour des revues plus complètes sur la planification d'horaires d'autobus.

2.1.2 Problème d'horaire d'autobus électriques avec dépôts multiples

Maintenant, tournons-nous vers la variante de ce problème qui nous intéresse davantage, le problème d'horaire d'autobus électriques avec dépôts multiples (MDEVSP). L'enjeu est le même, mais l'utilisation de véhicules électriques vient ajouter des contraintes importantes comme une autonomie moins grande, des temps de recharge plus longs et un nombre d'infrastructures de recharge limité qui doivent être considérées dans la planification des horaires (Zhang *et al.*, 2023, Sassi et Oulamara, 2017). Pour le MDEVSP, on a affaire à une littérature beaucoup plus récente qui se densifie d'année en année. Cette section présente un résumé des avancées dans le domaine et des différentes méthodes qui ont été proposées dans les 10 dernières années.

Li (2014) montre que le MDEVSP, même avec un seul dépôt, est également NP-difficile et propose une méthode de *branch-and-price* inspirée par Barnhart *et al.* (1998) pour le résoudre.

Une version tronquée de la génération de colonnes est ensuite introduite pour les instances à grande échelle, puis la solution générée par cette méthode est améliorée en utilisant une recherche locale basée sur une version personnalisée des méthodes proposées par Lin (1965), Potvin *et al.* (1996), Ibaraki *et al.* (2005) et Taillard *et al.* (1997).

Sassi et Oulamara (2017) se concentrent sur le cas avec un seul dépôt et proposent une méthode exacte à deux étapes où la première maximise la distance parcourue par les véhicules et la deuxième utilise ce résultat pour résoudre un problème mixte en nombres entiers (MIP). Ils présentent ensuite deux heuristiques. La première affecte un ensemble de tâches à chaque véhicule à l'aide du problème de clique de poids maximum et planifie la recharge en utilisant une formulation de flot à coût minimum. La deuxième génère une solution réalisable pour l'affectation des tâches aux véhicules et leur recharge, puis minimise les coûts de recharge à l'aide d'un algorithme d'horaire de recharge global.

Montoya *et al.* (2017) introduisent dans le problème le concept de recharge non linéaire. En effet, ils se basent entre autres sur les travaux de Pelletier *et al.* (2017) et Hoimoja *et al.* (2012) pour montrer que la fonction de recharge des batteries n'est pas linéaire et devrait plutôt être approximée par une fonction linéaire par morceaux. Ils montrent ensuite que de négliger la nature non linéaire de la fonction de recharge peut mener à des solutions non réalisables ou plus coûteuses.

Liu et Ceder (2020) proposent une formulation bi-objectif où le premier objectif minimise le nombre total de véhicules nécessaires et le deuxième minimise le nombre de chargeurs nécessaires. Il présente ensuite deux méthodes : une procédure en deux étapes basée sur la méthode lexicographique et utilisant la théorie de la fonction déficit (*DF theory*), ainsi qu'une méthode basée sur une version ajustée de la solution au problème de flot maximum développée par Ford et Fulkerson (1962). On réfère le lecteur à Liu et Ceder (2017) pour une définition complète de la théorie de la fonction déficit.

Pour résoudre le cas avec plusieurs dépôts, Wang *et al.* (2021) proposent une approche mêlant un algorithme génétique à la génération de colonnes. Une heuristique est utilisée pour générer un ensemble de colonnes (horaires) initiales, puis cet ensemble permet d'initialiser un algorithme de génération de colonnes pour générer un plus grand nombre d'horaires. Finalement, l'algorithme génétique est appliqué pour déterminer la solution finale comme un sous-ensemble des horaires générés.

Zhang *et al.* (2021) formulent un modèle de partitionnement d'ensemble dans lequel ils intègrent la fonction de recharge non linéaire ainsi que les coûts liés à la dégradation des batteries. Une méthode de *branch-and-price* couplée à une technique de stabilisation du dual est utilisée pour résoudre le problème et il est démontré que considérer l'usure de la batte-

rie dans le modèle peut diminuer les coûts et grandement augmenter la durée de vie de la batterie.

De leur côté, Zhou *et al.* (2022) proposent un modèle non linéaire et non convexe qu'ils décomposent en deux problèmes linéaires mixtes en nombres entiers. Ils introduisent ensuite trois familles d'inégalités valides qu'ils insèrent dans le primal pour accélérer la résolution.

Enfin, Gerbaux *et al.* (2025) utilisent une heuristique de génération de colonnes qui repose sur l'utilisation de réseaux de connexions réduits pour générer les horaires. Ils cherchent donc à choisir un sous-ensemble d'arcs et explorent plusieurs méthodes pour y arriver. Ils prouvent ensuite que la méthode la plus efficace semble être de combiner une heuristique gloutonne à l'utilisation d'un réseau de neurones graphique (GNN) qui estime la probabilité qu'un arc fasse partie d'une solution optimale. L'apprentissage machine offre de nouvelles possibilités dans la résolution du MDEVSP par son aptitude à prévoir ou à approximer des solutions. C'est dans cette lignée que les travaux de ce mémoire s'inscriront.

Ceci brosse donc un portrait global des avancées dans la planification d'horaires d'autobus électriques dans les dernières années. Cependant, notons tout de même que plusieurs problèmes connexes sont étudiés. Par exemple, les différentes stratégies et méthodes de recharge (Brinkel *et al.*, 2023, Li, 2014, Yang *et al.*, 2018) et la planification de la flotte et des infrastructures (Yildirim et Yildiz, 2021, An, 2020, Li *et al.*, 2018, Zhang *et al.*, 2022, Perumal *et al.*, 2022, Dirks *et al.*, 2022) mériteraient des revues aussi détaillées. On réfère le lecteur à Perumal *et al.* (2022), Zhang *et al.* (2023) et Gkiotsalitis *et al.* (2023) pour des revues plus complètes des travaux sur le MDEVSP et ses problèmes connexes.

2.2 Dégénérescence et inégalités duales

La génération de colonnes est une méthode qui a fait ses preuves pour la résolution de problèmes de grande taille. Elle donne souvent de meilleures bornes et permet de traiter certaines non-linéarités dans le sous-problème (Gschwind et Irnich, 2016). Cependant, lorsque la taille et la difficulté du problème augmentent, la dégénérescence du problème peut causer une convergence lente qu'on appelle le "*long-tail effect*" dans la littérature (Ben Amor *et al.*, 2006). Cette dégénérescence est fréquente dans les problèmes de planification d'horaire d'autobus, puisque l'on a souvent un grand nombre de véhicules identiques et un grand nombre de trajets, ce qui donne généralement lieu à un grand nombre de solutions ayant la même valeur. Plusieurs méthodes ont été développées pour contrer cette dégénérescence. Cette section en présente quelques-unes.

Ben Amor *et al.* (2006) introduisent le concept d'inégalités duales optimales (*dual-optimal inequalities*) qu'on ajoute au problème dual pour accélérer et stabiliser le processus de convergence. Ces inégalités sont satisfaites par au moins une solution duale de façon à ce que la valeur optimale de la fonction objectif reste la même. Ils séparent d'ailleurs ces inégalités en deux types : les inégalités duales optimales qui sont satisfaites par toutes les solutions duales optimales et les inégalités duales optimales profondes (*deep dual-optimal inequalities*) qui sont satisfaites par au moins une solution duale optimale. Les inégalités restreignent l'ensemble des valeurs duales possibles, ce qui réduit la taille et la fréquence des oscillations dans l'espace dual. Ils présentent deux types d'inégalités duales et proposent ensuite deux méthodes générales pour retrouver la réalisabilité du primal dans le cas où les inégalités ajoutées rendraient le problème irréalisable. Gschwind et Irnich (2016) poussent un peu plus loin l'utilisation des inégalités duales en introduisant de nouveaux types d'inégalités duales optimales profondes. Ils proposent également d'ajouter dynamiquement les inégalités duales et présentent plusieurs algorithmes exacts et heuristiques pour y arriver. Enfin, ils montrent qu'il peut être avantageux dans certains cas de sur-stabiliser la génération de colonnes à l'aide d'inégalités duales "fausses" (*non-DDOIs*) et d'utiliser un processus de réparation pour ensuite rendre le problème réalisable à nouveau. Plusieurs autres auteurs comme Nishi *et al.* (2011), Yarkony *et al.* (2021) et Mhamedi *et al.* (2022) utilisent les inégalités duales pour stabiliser la génération de colonnes. C'est d'ailleurs cette méthode qui nous intéressera dans ce mémoire.

Elhallaoui *et al.* (2005) présentent l'idée d'une agrégation dynamique des contraintes pour stabiliser la génération de colonnes. Cette méthode s'applique aux contraintes de partitionnement d'ensemble et repose sur le fait que "*pour un ensemble C de trajets donné, deux tâches w_1 et w_2 dans W sont dites équivalentes par rapport à C si tous les trajets dans C couvrent soit les deux tâches ou aucune des deux.*[1]" On utilise donc cette relation pour séparer les tâches en classes d'équivalence, puis on utilise certains critères d'équivalence qu'ils définissent pour agréger certaines contraintes ensemble et ainsi remplacer un groupe de contraintes par une seule contrainte agrégée. Ce concept est développé davantage dans Elhallaoui *et al.* (2008) et Elhallaoui *et al.* (2010). Bouarab *et al.* (2017) proposent également une version améliorée de l'agrégation dynamique de contraintes pour les problèmes de partitionnement d'ensemble qui résout à chaque itération un problème complémentaire. Ce problème complémentaire vise à obtenir une combinaison convexe de variables qui, une fois entrées dans la base, assure une décroissance de la valeur de la fonction objectif, évitant ainsi des pivots dégénérés.

1. Traduction libre.

Oukil *et al.* (2007) utilisent ce qu'ils appellent une *génération de colonnes stabilisée* en définissant un centre de stabilité et une fonction de pénalité (*penalty function*). Une méthode de réduction du réseau est proposée, permettant d'obtenir une borne inférieure sur la solution en nombres entiers et une solution duale réalisable. Cette solution duale est utilisée comme centre de stabilité initial. Par la suite, la fonction de pénalité est un terme que l'on ajoute à la fonction objectif du dual qui pénalise les mouvements loin du centre. Cette méthode vise à guider le progrès des variables duales pour que des colonnes de meilleure qualité soient générées. Notons que les travaux de Oukil et al. sont inspirés principalement des méthodes développées par du Merle *et al.* (1999) et Ben Amor et Desrosiers (2006).

Finalement, Benchimol *et al.* (2012) proposent une agrégation dynamique des contraintes stabilisée qui combine les concepts d'agrégation dynamique des contraintes d'Elhallaoui *et al.* (2005) et de stabilisation des variables duales d'Oukil *et al.* (2007). Cette méthode permet de combattre la dégénérescence simultanément dans le primal et dans le dual. Pour une revue des différentes méthodes de gestion de la dégénérescence, on réfère à Pessoa *et al.* (2018).

2.3 Stratégies de branchement heuristiques

Comme nous le mentionnons à la section 2.4, les travaux de Popovic (2023) se concentrent uniquement sur la résolution de la première relaxation linéaire. Une partie importante du présent mémoire sera donc de vérifier comment se comporte notre méthode avec prédiction d'inégalités duales dans le cas où l'on résout le problème à l'optimalité, ce qui implique de choisir une stratégie de recherche de solution entière. Comme on vise à développer une méthode de résolution applicable par notre partenaire industriel, nous nous concentrerons sur des heuristiques semblables à celles utilisées dans l'industrie. Cette section présente une brève revue de quelques stratégies de branchement heuristiques intéressantes permettant d'obtenir une solution entière réalisable plus rapidement que les stratégies de branchement classiques.

L'heuristique d'arrondissement (*rounding heuristic*) consiste à choisir itérativement une variable fractionnaire dans la solution de la relaxation linéaire du problème et à fixer sa borne inférieure à l'entier supérieur ou sa borne supérieure à l'entier inférieur (Sadykov *et al.*, 2019). Plusieurs règles différentes sont utilisées dans la littérature pour choisir quelles variables arrondir. La solution arrondie obtenue peut alors être non réalisable et une heuristique, e.g., de recherche locale, doit être appliquée pour trouver une solution réalisable. On invite le lecteur à consulter Berthold (2006) pour une revue de ces règles.

Le *diving* se rapproche de l'heuristique d'arrondissement, mais en diffère par le fait que l'on réoptimise la relaxation linéaire après avoir fixé une ou des variables (Sadykov *et al.*,

2019, Joncour *et al.*, 2010). Comme son nom l'indique, elle vise à plonger dans l'arbre de branchement de façon à trouver une bonne solution entière rapidement. À chaque nœud, on choisit heuristiquement une branche et on élimine toutes les autres. Comme exemple dans un contexte de génération de colonnes, on peut citer le *pure diving* présenté par Sadykov *et al.* (2019). Il s'agit d'un *diving* qui, itérativement, résout la relaxation linéaire, choisit un ensemble de colonnes prenant une valeur fractionnaire, arrondit leur valeur à l'entier supérieur ou inférieur, puis résout la nouvelle relaxation linéaire.

Une autre variante intéressante est le *Sub-MIPing* (Sadykov *et al.*, 2019). C'est une heuristique qui va résoudre exactement ou approximativement un MIP, mais qui, avant de le faire, va tenter de réduire la taille du problème. Pour y arriver, on peut par exemple fixer la valeur de quelques variables par une heuristique d'arrondissement ou un *diving*, puis après un certain nombre d'itérations, on résout le MIP restreint résultant. Cette heuristique cherche à trouver un compromis entre la vitesse d'obtention d'une solution et la qualité de la solution trouvée en n'explorant qu'une partie ciblée de l'arbre de recherche.

Dans ce mémoire, la méthode qui sera utilisée est celle du *diving* puisque c'est l'heuristique qui est favorisée par notre partenaire GIRO. Pour une présentation plus complète des heuristiques de recherche de solution entière, on réfère le lecteur à Sadykov *et al.* (2019), Berthold (2006) et Quesnel *et al.* (2017).

2.4 Contributions de Popovic (2023)

Ce mémoire poursuit les travaux débutés par Popovic (2023) dans son mémoire de maîtrise. Il est donc important de décrire ses progrès avant de présenter nos contributions puisque nous réutiliserons les outils qu'il a créés.

2.4.1 Théorie

Les ambitions des travaux de Popovic (2023) étaient de développer une méthode utilisant l'apprentissage machine pour prédire des inégalités duales et les insérer dans le problème dans le but de réduire la dégénérescence et d'accélérer la résolution par génération de colonnes. Il est supposé que les inégalités les plus faciles à prédire par apprentissage machine sont les inégalités de la forme

$$\pi_i \geq \pi_j \tag{2.1}$$

car elles contiennent uniquement deux variables duales. Il mentionne ensuite que "*la méthode*

la plus simple et la plus prometteuse semble être d'ordonner les variables duales. Pour ce faire, on génère des chaînes d'inégalités ayant la forme suivante :

$$\pi_{s_1} \geq \pi_{s_2} \geq \pi_{s_3} \geq ... \geq \pi_{s_q} \tag{2.2}$$

où q est la longueur de la chaîne d'inégalités." On utilise donc un GNN pour prédire la relation ($\leq$ ou $\geq$) pour chaque paire de variables duales, puis on crée un graphe $\mathcal{G}^k$ pour chaque dépôt k où chaque nœud représente une variable duale et on ajoute un arc entre deux nœuds s'il est prédit que $\pi_i \geq \pi_j$. On ajoute un nœud source et un nœud puits, on assigne un coût de -1 à chaque arc, puis on utilise l'algorithme de Bellman-Ford (Bellman, 1958) pour trouver des chemins de coût minimal. On obtient donc des chaînes d'inégalités de la forme (2.2). On peut ensuite insérer toutes ces inégalités dans le problème maître (PM).

Le présent mémoire se concentre davantage sur l'aspect mathématique de ce projet. Nous n'entrerons donc pas dans les détails concernant la section "apprentissage machine" de ces travaux. On réfère le lecteur à la section 4.3 de Popovic (2023) pour une description exhaustive du modèle d'apprentissage supervisé contenant un GNN et un perceptron multicouches. Ce qui est important de mentionner ici est que ce modèle prend en entrée un graphe représentant le réseau de connexions (détaillé à la section 3.2 du présent mémoire) avec des vecteurs de caractéristiques propres à chaque nœud et arc, et donne en sortie une estimation de la probabilité $\hat{y}_{ij}$ que $\pi_i \geq \pi_j$ pour chaque paire d'inégalités duales (π_i, π_j).

Comme on utilise un modèle d'apprentissage machine pour prédire le sens des inégalités, il est probable que des erreurs soient commises et que de fausses inégalités soient ajoutées au modèle. C'est pour réduire ces erreurs qu'un processus de réparation est introduit. Tout le processus de prédiction et de sélection des inégalités duales ainsi que le processus de réparation seront abordés en détail dans le chapitre 4 de mon mémoire. Nous présenterons également une version complète du problème maître avec inégalités duales.

2.4.2 Résultats

Comme on cherche à accélérer la résolution du MDEVSP, les deux résultats qui nous permettront d'évaluer l'efficacité de notre méthode seront le facteur d'accélération et le gap (écart entre la valeur optimale du PM et celle du problème maître avec inégalités duales (PM^{ID})). Popovic les calcule de la façon suivante :

$$Acc = \frac{temps}{temps^{ID}} \tag{2.3}$$

$$gap = \frac{(val - val^{ID})}{val} \tag{2.4}$$

où $temps$ et $temps^{ID}$ sont les temps de résolution du PM et du PM^{ID} alors que val et val^{ID} sont les valeurs optimales des deux problèmes maîtres. La valeur optimale peut différer, car en ajoutant des inégalités duales, on relaxe le problème, ce qui peut nous permettre d'obtenir une borne inférieure plus petite sur la valeur de la fonction objectif du problème. Il a commencé par tester sur 50 instances différentes (25 de taille moyenne (M) avec environ 850 trajets à couvrir et 25 de grande (G) taille avec environ 1050 trajets à couvrir) l'accélération théoriquement atteignable sur la relaxation linéaire du problème si l'on avait un modèle de prédiction avec une précision de 100% et a montré qu'on pouvait obtenir des facteurs d'accélération moyens de 5.33 pour les instances moyennes et de 3.34 pour les grandes instances. Notons qu'il s'agit ici de la moyenne géométrique.

Ensuite, une analyse intéressante qui a été faite par Popovic et que nous pousserons plus loin dans ce mémoire est l'analyse du pourcentage d'inégalités valides en fonction de la valeur du seuil d'acceptation β. Ce seuil est utilisé pour trier les inégalités que l'on considère lorsque l'on sélectionne lesquelles ajouter au problème. On considère uniquement l'inégalité $\pi_i \geq \pi_j$ si $\hat{y}_{ij} \geq 1 - \beta$ et l'inégalité inverse si $\hat{y}_{ij} < \beta$. Le tableau 2.1 montre les précisions et le nombre d'inégalités prédites pour différentes valeurs de β entre 0.15 et 0.45. La précision correspond au pourcentage d'inégalités prédites par le modèle d'apprentissage machine qui sont valides dans la solution du PM. Il juge, à la lumière de ces résultats (tableau 2.1), qu'un seuil d'acceptation de 0.15 donne une proportion d'inégalités valides suffisante.

Tableau 2.1 Analyse des performances en fonction du seuil d'acceptation β (Popovic, 2023)

	Précision		Nb. Inég.	
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

Finalement, après avoir testé plusieurs configurations, il montre que celle qui semble donner le meilleur compromis entre l'accélération, le pourcentage de trajets couverts et la détérioration de la borne inférieure est celle où le coefficient ϕ_{ij} (défini au chapitre 4 du présent mémoire)

est de 1 pour toutes les paires i, j et où on permet une seule itération du processus de recouvrement. Avec ces paramètres, il obtient un facteur d'accélération moyen de 1.28 sur les instances et un gap de 0.10%. Il mentionne également que les inégalités duales semblent surtout efficaces pour les instances très dégénérées et que, dans les cas moins dégénérés, la technique de perturbation semble suffisante. Nous invitons le lecteur à lire la section 5.3 de Popovic (2023) pour l'ensemble des résultats expérimentaux.

2.5 Contributions de ce mémoire

Ce mémoire apportera plusieurs nouvelles idées. Nous ferons une analyse approfondie des chaînes d'inégalités duales, puis nous appliquerons les idées développées par Popovic (2023) sur la résolution complète du problème et expliquerons leurs faiblesses dans ce contexte. Notons qu'un branchement heuristique standard a finalement été suffisant pour prendre en compte les inégalités duales. Nous montrerons que pour accélérer significativement la résolution du MDEVSP par génération de colonnes, les inégalités duales ajoutées au problème doivent être suffisamment serrées, sinon elles risquent de n'avoir aucun effet ou même de ralentir la convergence, ce qui n'a jamais été étudié auparavant dans la littérature. Notre analyse démontrera également que l'ajout d'un petit nombre d'inégalités fausses dans le problème peut grandement ralentir la convergence, ce qui rend moins viable la méthode de prédiction proposée par Popovic (2023).

Ensuite, nous introduirons une nouvelle stratégie mettant de côté l'apprentissage automatique. En effet, au chapitre 5, nous montrerons qu'une fois appliquées à la résolution complète, les chaînes d'inégalités prédites par apprentissage machine ne sont pas suffisantes pour stabiliser la génération de colonnes et accélérer la résolution. En nous basant sur l'analyse des chaînes d'inégalités duales que nous avons faite, nous avons adapté notre approche pour exploiter la solution connue du problème maître plutôt que de tenter de prédire des inégalités. Notre méthode utilise la solution de la première relaxation linéaire du problème maître pour créer des chaînes d'inégalités duales valides et accélérer la suite de la résolution, ce qui, à notre connaissance, n'a jamais été documenté dans la littérature. Nous montrerons que cette nouvelle approche, bien qu'elle ne soit appliquée qu'au-delà de la première relaxation linéaire, accélère la résolution du problème d'environ 30% en n'affectant que de façon négligeable la qualité de la solution.

CHAPITRE 3 DÉFINITION DU PROBLÈME, MODÉLISATION ET MÉTHODE DE RÉSOLUTION DE BASE

Dans ce chapitre, nous commençons par définir de façon détaillée le MDEVSP et présentons le réseau de connexions qui peut être utilisé pour représenter implicitement l'ensemble des horaires réalisables. Nous proposons ensuite une formulation mathématique pour le MDEVSP et décrivons une méthode de génération de colonnes de base (i.e., sans inégalités duales) pour le résoudre.

3.1 Problème d'horaires d'autobus électriques avec dépôts multiples

Le MDEVSP consiste à assigner une flotte de véhicules à un ensemble de trajets en s'assurant que chaque trajet soit couvert par un autobus et que le niveau de charge des véhicules soit toujours suffisant. On doit affecter un horaire réalisable à chaque autobus utilisé et cet horaire doit obligatoirement débuter et terminer au même dépôt. L'objectif ici, comme dans la majorité des problèmes de ce type, est de minimiser les coûts d'opération qui incluent un coût fixe pour chaque autobus utilisé, des coûts d'attente en dehors des dépôts et des coûts de déplacement.

On définit donc un ensemble de dépôts $\mathbf{K} = \{k_1, k_2, ..., k_m\}$, où le dépôt k a g^k autobus disponibles, un ensemble de trajets $\mathbf{T} = \{t_1, t_2, ..., t_n\}$ et un ensemble de stations de recharge $\mathbf{H} = \{h_1, h_2, ..., h_r\}$, la station h ayant un nombre limité de bornes b_h. Pour les autobus, on doit également définir leur autonomie maximale, la quantité d'énergie consommée par unité de temps de déplacement (e_d), la quantité d'énergie consommée par unité de temps d'attente (e_a) et la quantité d'énergie rechargée par unité de temps lorsqu'ils sont branchés. On associe un coût fixe à chaque véhicule utilisé (c_f), à chaque unité de temps en attente en dehors des dépôts (c_a) et à chaque unité de distance parcourue par les véhicules (c_d).

Nous avons abordé brièvement à la section 2.1.2 les travaux de Montoya *et al.* (2017) et Pelletier *et al.* (2017) et leur importance dans notre approche face à la planification de la recharge de véhicules. En effet, dans ce mémoire, nous utilisons une fonction de recharge linéaire par morceaux basée sur leurs travaux. Ils ont démontré qu'il est inapproprié d'approximer la recharge d'un véhicule de façon linéaire, car pour des raisons de préservation de la batterie, la borne de recharge diminue le courant lorsque la batterie est presque pleine. La figure 3.1 présente une courbe de recharge typique. La recharge se fait donc en deux phases. Dans la première, le courant est constant et la tension augmente. Le niveau de recharge augmente

de façon linéaire jusqu'à atteindre environ 80%. Ensuite, la tension reste constante, mais le courant diminue exponentiellement jusqu'à ce que le niveau de recharge maximal soit atteint.

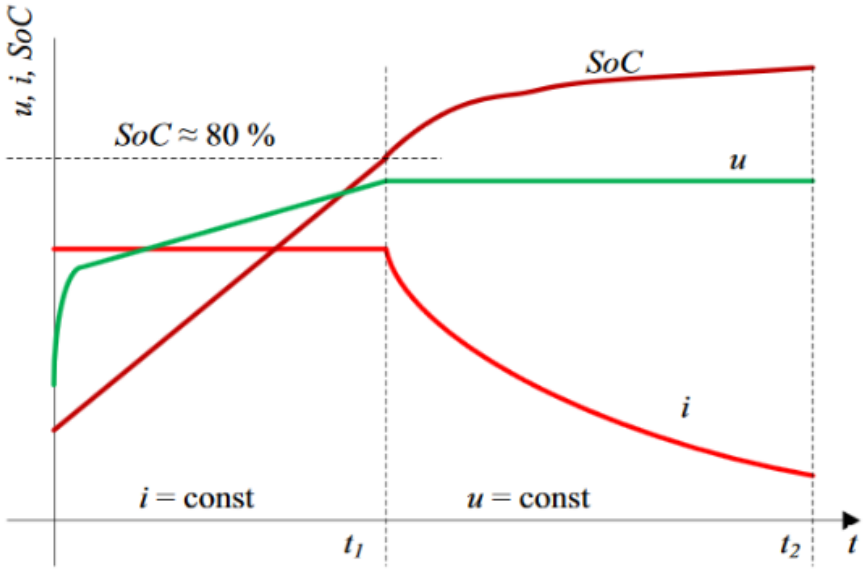

FIGURE 3.1 Courbe de recharge typique en fonction du temps t. SoC correspond au niveau de charge, i correspond au courant et u correspond à la tension. (Source : Hoimoja *et al.*, 2012)

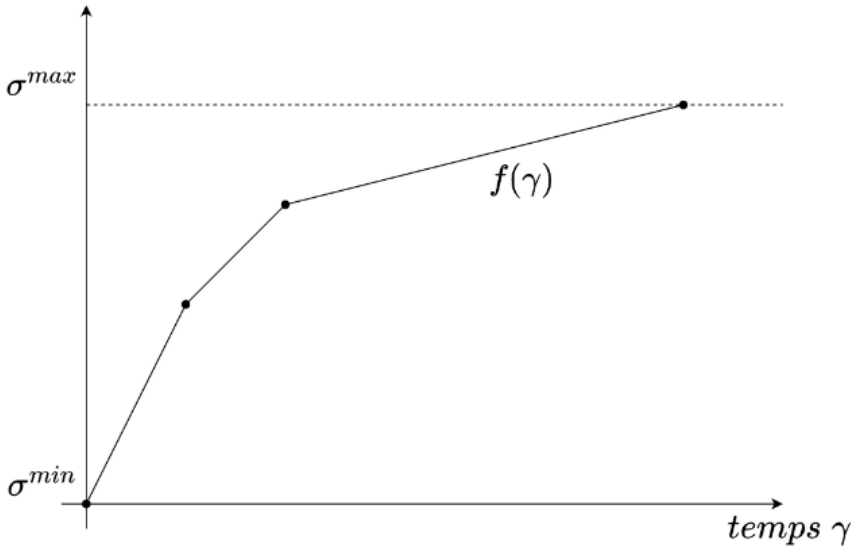

FIGURE 3.2 Fonction de recharge linéaire morceaux. (Source : Popovic, 2023)

La figure 3.2 présente la fonction $f(\gamma)$ de charge linéaire par morceaux utilisée pour la

suite des travaux. σ^{min} et σ^{max} représentent respectivement les niveaux de charge minimum et maximum imposés sur les véhicules. L'état de charge de la batterie doit donc toujours se situer entre ces deux valeurs inclusivement. L'état de recharge de la batterie après une activité de recharge de Δ unités de temps est donné par l'équation suivante :

$$q^{fin} = f(f^{-1}(q^{ini}) + \Delta) \tag{3.1}$$

où q^{fin} est le niveau de charge final et q^{ini} est le niveau de charge initial.

3.2 Modélisation

Dans cette section, nous commençons par présenter le réseau de connexions représentant l'ensemble des horaires réalisables Ω et la façon dont on le construit. Nous pourrons ensuite formuler le problème maître en nombres entiers et ses différents paramètres.

3.2.1 Construction du réseau de connexions

Comme l'ensemble des horaires réalisables Ω est très grand et difficile (voire impossible) à énumérer, il est représenté implicitement sous forme d'un réseau de connexions $\mathcal{G}^k$ pour chaque dépôt k. Les tableaux 3.1 et 3.2 résument les nœuds et arcs de ce réseau qui sont détaillés dans la suite de cette section. Le tout est illustré à la figure 3.3. Notons que dans les tableaux, d_{ij} représente la distance entre les nœuds i et j.

Tableau 3.1 Définition des noeuds du réseau de connexions

Noeuds	Définition
$\overline{o}^k$	Départ du dépôt k
$\underline{o}^k$	Arrivée au dépôt k
w_p^h, $p \in \mathbf{P}$, $h \in \mathbf{H}$	Attente à la station de recharge h pendant la période $p-1$
r_p^h, $p \in \mathbf{P}$, $h \in \mathbf{H}$	Recharge à la station de recharge h pendant la période $p-1$
$t_i \in \mathbf{T}$	Trajet i
$s_p^{k'}$, $p \in \mathbf{P}$	Retour et attente au dépôt k pendant la période $p-1$

Tableau 3.2 Définition et coût des arcs du réseau de connexions

Arcs	Définition	Coût
$(\overline{o}^k, t_i)$ $\forall t_i \in \mathbf{T}$	Arcs sortant du dépôt vers chaque trajet	$c_d d_{ki} + c_f$
$(t_i, \underline{o}^k)$ $\forall t_i \in \mathbf{T}$	Arcs entrant au dépôt à partir de chaque trajet	$c_d d_{ik}$
(t_i, t_j) $\forall t_i, t_j \in \mathbf{T}$ t.q. (t_i, t_j) satisfait (3.2) et (3.3)	Arc entre deux trajets pour chaque paire de trajets valide	$c_d d_{ij} + c_a W_{ij}$
(t_i, w_p^h) $\forall t_i \in \mathbf{T}, \forall h \in \mathbf{H}$ avec p défini par (3.4)	Arcs entre chaque trajet et le premier nœud d'attente atteignable pour chaque station de recharge	$c_d d_{ih}$
$(t_i, s_p^{k'})$ $\forall t_i \in \mathbf{T}$ avec p défini par (3.6)	Arcs entre chaque trajet et le premier nœud de retour et d'attente au dépôt	$c_d d_{ik}$
$(s_{p_i}^{k'}, t_j)$ $\forall t_j \in \mathbf{T}$ t.q. (3.7) est satisfait	Arcs entre chaque nœud de retour et d'attente au dépôt et chaque trajet atteignable en quittant le dépôt à la fin de la période	$c_d d_{ki}$
$(s_{p_i}^{k'}, s_{p_{i+1}}^{k'})$ $\forall p_i \in \mathbf{P} = \{p_1, ..., p_n\} \setminus \{p_n\}$	Arcs d'attente au dépôt	0
$(w_{p_i}^h, w_{p_{i+1}}^h)$ $\forall p_i \in \mathbf{P} = \{p_1, ..., p_n\} \setminus \{p_n\}$	Arcs d'attente à une station de recharge	$c_a \lambda$
$(w_{p_i}^h, t_j)$ $\forall t_j \in \mathbf{T}$ t.q. (3.5) est satisfait	Arcs entre chaque noeud d'attente à une station de recharge et chaque trajet atteignable en quittant la station à la fin de la période p_i	$c_d d_{hj}$

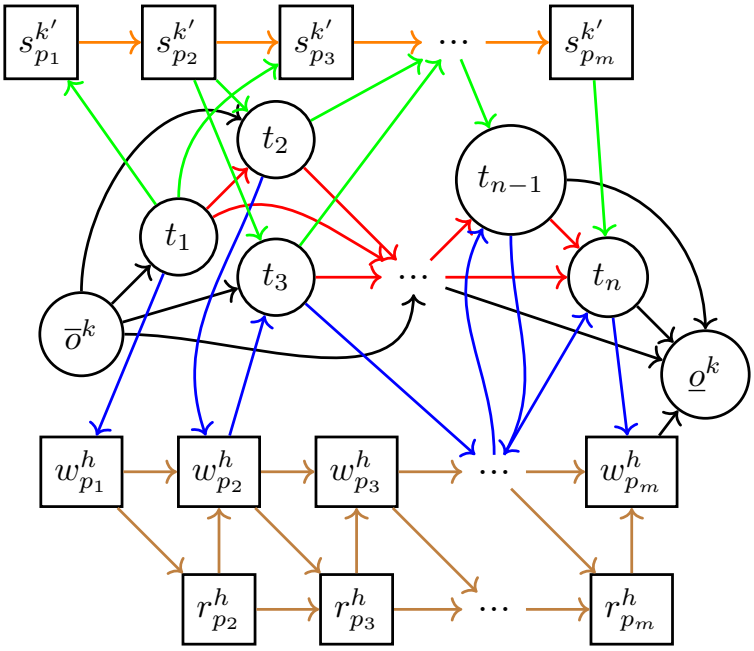

FIGURE 3.3 Réseau $\mathcal{G}^k$ pour le dépôt k (Source : Gerbaux *et al.*, 2025)

Pour commencer, définissons un horaire réalisable comme une suite de trajets et de périodes de recharge que peut réaliser un véhicule dans une journée en maintenant un niveau de recharge entre σ^{min} et σ^{max}. Notons également que les autobus doivent débuter et terminer leur journée au même dépôt.

Comme les stations de recharge possèdent un nombre limité de bornes, on doit pouvoir surveiller le nombre de véhicules à une station en tout temps. Pour faciliter cette tâche, on utilise une discrétisation du temps en courts intervalles de λ unités de temps (minutes). De cette façon, on obtient $\mathbf{P} = \{p_1, p_2, ..., p_z\}$, l'ensemble des z périodes de temps qui forment l'horizon d'optimisation. Chaque période $p \in \mathbf{P}$ débute au temps D_p et se termine au temps $F_p = D_p + \lambda$. On impose alors qu'une activité de recharge ne peut commencer qu'au début d'une période (D_{p_i}) et terminer à la fin d'une période (F_{p_j}, $j \geq i$). Si l'état de la batterie atteint σ^{max} avant la fin d'une période de recharge, on considère quand même que l'autobus occupe la borne jusqu'à la fin de la période. Pour chaque période $p \in \mathbf{P}$ et chaque station de recharge $h \in \mathbf{H}$, on définit un nœud d'attente w^h_p et un nœud de recharge r^h_p ainsi qu'un arc entre chaque paire ($w^h_{p_i}$, $w^h_{p_{i+1}}$), ($w^h_{p_i}$, $r^h_{p_{i+1}}$), ($r^h_{p_i}$, $w^h_{p_i}$) et ($r^h_{p_i}$, $r^h_{p_{i+1}}$).

Définissons ensuite un nœud "départ" ($\underline{o}^k$) et un nœud "arrivée" ($\overline{o}^k$) ainsi qu'un nœud pour chaque trajet (t_i). On ajoute des arcs ($\overline{o}^k$, t_i) et (t_i, $\underline{o}^k$) pour chaque trajet et un arc entre deux trajets consécutifs (t_i, t_j) pour chaque paire valide. Notons qu'une paire de trajets est considérée valide si un autobus peut effectuer le trajet t_j tout de suite après avoir complété le

trajet t_i et si le temps total entre les deux trajets n'excède pas δ unités de temps. On ajoute donc l'arc seulement si les deux conditions suivantes sont respectées :

$$\alpha_i^e + \theta_{ij} \leq \alpha_j^s \tag{3.2}$$

$$\alpha_j^s - \alpha_i^e \leq \delta \tag{3.3}$$

où α_i^s est le temps de début du trajet i, α_i^e est le temps de fin du trajet et θ_{ij} est le temps de déplacement entre la fin du trajet i et le début du trajet j.

Après chaque trajet, les autobus peuvent aller vers une station de recharge. On ajoute donc un arc $(t_i, w^h_{p_{i,h}})$ reliant chaque trajet au nœud d'attente le plus tôt qui peut être atteint à chaque station de recharge $h \in \mathbf{H}$. Un nœud d'attente peut être atteint si le temps de déplacement (θ_{ih}) entre la fin du trajet et la station de recharge h est plus petit que l'écart de temps entre la fin du trajet et le début de la période de temps p. On peut utiliser l'expression suivante pour déterminer quelle période d'attente peut être atteinte en premier :

$$p_{i,h} = min\{p \in \mathbf{P} | D_p \geq \alpha_i^e + \theta_{i,h}\} \tag{3.4}$$

On ajoute aussi les arcs $(w^h_{p_{h,j}}, t_j)$ entre les nœuds d'attente et les trajets pouvant être effectués en quittant la station de recharge h au temps F_p. Pour chaque trajet $t_j \in \mathbf{T}$, on a donc un seul arc $(w^h_{p_{h,j}}, t_j)$ par station de recharge $h \in \mathbf{H}$. Pour chaque trajet t_j et chaque station de recharge h, on détermine la période $p_{h,j}$ la plus tardive à laquelle l'autobus peut quitter la station pour effectuer le trajet :

$$p_{h,j} = \max\{p \in \mathbf{P} | F_p + \theta_{hj} \leq \alpha_j^s\} \tag{3.5}$$

Finalement, on ajoute, pour chaque période de temps p, un nœud $(s^{k'}_{p_{k',j}})$ d'attente au dépôt. Pour chacun de ces nœuds, on ajoute un arc $(t_i, s^{k'}_{p_{i,k}})$ si la période $p_{i,k'}$ est la première à laquelle un autobus pourrait revenir au dépôt k' après avoir effectué le trajet t_i. Pour déterminer la période $p_{i,k'}$, on remplace dans l'expression (3.4) $\theta_{i,h}$ par $\theta_{i,k'}$, la distance entre la fin de trajet t_i et le dépôt k' pour obtenir :

$$p_{i,k'} = \min\{p \in \mathbf{P} | D_p \geq \alpha_i^e + \theta_{i,k'}\} \tag{3.6}$$

On ajoute également un arc $(s^{k'}_{p_{k',j}}, t_j)$ pour chaque trajet t_j en reliant à chaque trajet, le

dernier nœud d'attente duquel un autobus peut partir pour l'effectuer. Similairement à (3.5), on détermine la période selon :

$$p_{k',j} = \max\{p \in \mathbf{P} | F_p + \theta_{k',j} \leq \alpha_j^s\} \tag{3.7}$$

Pour minimiser le nombre d'autobus utilisés, un coût fixe est ajouté à chaque arc sortant du nœud départ $\overline{o}^k$. Ensuite, les coûts sont calculés en fonction des temps de déplacement à vide et des temps d'attente hors dépôt. La formule générale pour calculer le coût sur un arc c_{ij} est :

$$c_{ij} = \begin{cases} c_d d_{ij} + c_a W_{ij} + c_f & \text{si } i = \overline{o}^k \\ c_d d_{ij} + c_a W_{ij} & \text{sinon} \end{cases} \tag{3.8}$$

où d_{ij} est le temps de déplacement à vide entre le nœud i et le nœud j et W_{ij} correspond au temps d'attente hors dépôt entre les deux nœuds.

L'ensemble des horaires possibles à partir du dépôt k (Ω_k) sera donc l'ensemble des chemins possibles entre $\overline{o}^k$ et $\underline{o}^k$ où le niveau de charge de la batterie peut demeurer entre σ^{min} et σ^{max}. Pour tenir compte du niveau de charge le long d'un horaire, on ajoute des contraintes de ressources qui sont traitées dans l'algorithme d'étiquetage qui sera détaillé à la section 3.3.2.

3.2.2 Problème maître en nombres entiers

Le MDEVSP peut se formuler comme un programme linéaire en nombres entiers faisant appel aux variables binaires x_s, $s \in \Omega^k$. Une variable binaire x_s prend la valeur 1 si l'on associe l'horaire s à un autobus et 0 sinon. On définit également c_s, le coût de l'horaire s, qui est la somme des coûts sur les arcs qui forment l'horaire dans le graphe $\mathcal{G}^k$. Définissons le problème maître en nombres entiers (PMNE) :

$$\min \quad \sum_{k \in \mathbf{K}} \sum_{s \in \Omega^k} c_s x_s \tag{3.9a}$$

$$\text{s.c.} \quad \sum_{k \in \mathbf{K}} \sum_{s \in \Omega^k} a_s^t x_s = 1 \qquad \forall t \in \mathbf{T} \tag{3.9b}$$

$$\sum_{s \in \Omega^k} x_s \leq g^k \qquad \forall k \in \mathbf{K} \tag{3.9c}$$

$$\sum_{k \in \mathbf{K}} \sum_{s \in \Omega^k} o_s^{p,h} x_s \leq b_h \qquad \forall h \in \mathbf{H}, p \in \mathbf{P} \tag{3.9d}$$

$$x_s \in \{0, 1\} \qquad \forall k \in \mathbf{K}, s \in \Omega^k \tag{3.9e}$$

Le tableau 3.3 présente et définit les différents paramètres et variables. L'objectif est de minimiser la somme des coûts des horaires choisis (3.9a) tout en s'assurant que chaque trajet est couvert exactement une fois (3.9b), qu'on n'utilise jamais plus d'autobus que le nombre disponible à chaque dépôt (3.9c) et qu'on ne dépasse jamais la capacité des stations de recharge (3.9d).

Tableau 3.3 Définition des paramètres et variables du PMNE

Paramètre	Définition
c_s, $\forall s \in \Omega^k$	Coût de l'horaire s
a_s^t, $\forall t \in \mathbf{T}$, $\forall s \in \Omega^k$	Égal à 1 si le trajet t est inclus dans l'horaire s, 0 sinon
g^k, $\forall k \in \mathbf{K}$	Nombre d'autobus disponibles au dépôt k
$o_s^{p,h}$, $\forall p \in \mathbf{P}$, $\forall h \in \mathbf{H}$, $\forall s \in \Omega^k$	Égal à 1 si l'horaire s inclut une recharge à la station h pendant la période p, 0 sinon
b_h, $\forall h \in \mathbf{H}$	Nombre de bornes de recharge disponibles à la station h
x_s, $\forall s \in \Omega^k$	Variable égale à 1 si un autobus est assigné à l'horaire s, 0 sinon

3.3 Méthode de résolution de base

Pour résoudre le MDEVSP, nous proposons d'utiliser une méthode de génération de colonnes imbriquée dans une méthode de branchement heuristique, soit une méthode de base très répandue pour résoudre des problèmes contenant un très grand nombre de variables. Nous présentons d'abord la génération de colonnes (section 3.2.1) qui permet de résoudre les relaxations linéaires et décrivons ensuite la stratégie de branchement (section 3.2.2) pour obtenir des solutions entières.

3.3.1 Génération de colonnes

La relaxation linéaire du modèle (3.9) est appelée le problème maître (PM). Le PM se résout par génération de colonnes (Desaulniers *et al.*, 2005). Un algorithme de génération de colonnes est itératif. À chaque itération, un PM restreint à un sous-ensemble Ω_r des variables x_s (dénoté PMR) est d'abord résolu pour donner une paire de solutions primale et duale optimales. On résout ensuite un sous-problème par dépôt qui a pour but de trouver s'il existe des variables x_s, $s \in \Omega^k$, qui ont un coût réduit négatif par rapport à la solution duale du PMR. Si de telles variables (colonnes) sont trouvées, elles sont ajoutées au PMR et une nouvelle itération débute. Sinon, la solution primale du PMR est aussi optimale pour le PM et l'algorithme s'arrête. Ω_0 correspond au sous-ensemble initial de variables et Ω^- correspond à l'ensemble des colonnes de coût réduit négatif identifiées grâce au sous-problème. L'algorithme 1 décrit le pseudo-code d'un tel algorithme.

Algorithme 1 Génération de colonnes

$\Omega_r \leftarrow \Omega_0 \subset \Omega$
Tant que vrai **faire**
  Résolution du PMR avec le sous-ensemble de variables $x_s, s \in \Omega_r$
  Résolution des sous-problèmes sur les réseaux $\mathcal{G}^k, k \in K$
  $\Omega^- \leftarrow$ Ensemble des colonnes de coût réduit négatif
  Si $\Omega^- = \emptyset$ **alors STOP, Solution optimale atteinte**
  Sinon
    $\Omega_r \leftarrow \Omega_r \cup \Omega^-$
  Fin Si
Fin Tant que

De cette façon, on améliore itérativement la solution du problème réduit jusqu'à atteindre la solution optimale. La complexité de cette méthode se trouve principalement dans l'élaboration du sous-problème. La section 3.3.2 décrit en détail l'ajustement que l'on fait sur les coûts des arcs dans les graphes $\mathcal{G}^k$ ainsi que l'algorithme utilisé pour trouver les colonnes ayant un coût négatif.

3.3.2 Sous-problèmes

Pour trouver les variables à coût réduit négatif, on peut résoudre un problème de plus court chemin sur chaque réseau $\mathcal{G}^k, k \in \mathbf{K}$ dans lequel on modifie les coûts des arcs de façon à ce que la somme des coûts des arcs le long d'un chemin soit égale au coût réduit de cet horaire. Définissons donc le coût réduit d'une variable x_s.

Soit $(\pi_t)_{t\in T}$, $(\lambda_k)_{k\in K}$ et $(\mu_{p,h})_{p\in P,h\in H}$ les vecteurs de variables duales associés aux contraintes (3.9b) à (3.9d), respectivement. Le coût réduit $\bar{c}_s$ de la variable x_s est :

$$\bar{c}_s = c_s - \sum_{t\in\mathbf{T}} \pi_t a_s^t - \lambda_k - \sum_{h\in\mathbf{H}} \sum_{p\in\mathbf{P}} \mu_{p,h} o_s^{p,h} \tag{3.10}$$

Dans le graphe, on associe aux arcs (i,j) les coûts ajustés $\bar{c}_{ij}$ suivants (où n est un nœud quelconque) :

- Pour chaque arc (t,n) sortant d'un nœud t : $\bar{c}_{t,n} = c_{t,n} - \pi_t$
- Pour chaque arc $(\overline{o}^k, n)$ sortant du dépôt k : $\bar{c}_{\overline{o}^k,n} = c_{\overline{o}^k,n} - \lambda_k$
- Pour chaque arc (w_{p-1}^h, r_p^h) entrant à un nœud de recharge r_p^h : $\bar{c}_{w_{p-1}^h, r_p^h} = c_{w_{p-1}^h, r_p^h} - \mu_{p-1,h}$
- Pour chaque arc (r_{p-1}^h, r_p^h) : $\bar{c}_{r_{p-1}^h, r_p^h} = c_{r_{p-1}^h, r_p^h} - \mu_{p-1,h}$
- Pour tous les autres arcs (i,j) : $\bar{c}_{i,j} = c_{i,j}$

Avec ces coûts ajustés, le coût de tout chemin entre $\overline{o}^k$ et $\underline{o}^k$ est égal au coût réduit de l'horaire qui correspond à ce chemin. Par contre, pour garantir que le chemin est réalisable par rapport à l'état de charge de l'autobus et que chaque recharge se fait de façon continue lors d'une visite à une station de recharge, on doit ajouter des contraintes de ressource. Entre autres, une ressource indique le niveau de charge à chaque nœud visité, niveau qui diminue lors d'un déplacement et augmente lors d'une recharge. Ce niveau doit demeurer entre σ_{min} et σ_{max}. Le sous-problème associé au dépôt k correspond alors à un problème de plus court chemin avec contraintes de ressources (Irnich et Desaulniers, 2005), qui peut se résoudre à l'aide d'un algorithme d'étiquetage. Cet algorithme associe une étiquette initiale au nœud $\overline{o}^k$, puis crée des chemins partiels en prolongeant cette étiquette et ses successeurs. Une règle de dominance qui élimine les étiquettes ne pouvant mener à un chemin optimal est appliquée pour éviter d'énumérer tous les chemins réalisables. Comme cette partie n'est pas centrale à nos travaux, nous référons le lecteur à Gerbaux *et al.* (2025) pour plus de détails.

3.3.3 Méthodes de branchement

L'algorithme de génération de colonnes permet de résoudre la relaxation linéaire du problème, mais ne renvoie pas nécessairement une solution entière. Pour obtenir une solution entière optimale, on pourrait appliquer une méthode de *branch-and-price* qui explore un arbre de branchement en imposant des décisions de branchement dichotomiques. Toutefois, étant donné la taille des instances du MDEVSP à résoudre par notre partenaire industriel

GIRO, nous proposons de faire une exploration heuristique de cet arbre de recherche, soit en utilisant un branchement heuristique appelé *diving*. C'est une stratégie de branchement en profondeur sans retour en arrière qui vise à créer une seule branche à chaque nœud, de façon à obtenir une solution entière rapidement. Cette stratégie n'assure pas de trouver une solution optimale, mais dans l'industrie, les enjeux de temps de calcul sont priorisés par rapport à la qualité de la solution. Comme l'objectif de ce mémoire est de développer une méthode de résolution applicable dans l'industrie, c'est cette stratégie de branchement que nous utilisons. Pour faire la descente dans l'arbre, trois types de décisions peuvent être imposés. On les décrit ici en ordre de priorité. Notons qu'en réalité, un autre type de décision est possible. On utilise des variables de perturbation comme méthode supplémentaire pour contrer la dégénérescence, et l'autre type de décision qui peut être pris lors du branchement est de simplement retirer ces variables. Nous expliquerons en détails la perturbation et son utilité à la section 4.1.

Le premier type que l'on priorise est la fixation de colonnes. Pour l'appliquer, on sélectionne les variables ayant les valeurs les plus proches de 1 et supérieures à un seuil minimum, puis on les fixe simplement à 1 dans le PM. Notons que dans notre cas, on ne fait que fixer les variables à l'entier supérieur. Cette façon de faire est plus efficace, car en fixant une variable x_s à 1, on fixe indirectement à 0 toutes les variables associées aux autres horaires couvrant un des trajets couverts par s.

Le deuxième type dans l'ordre de priorité est la fixation d'intertâches. Il est semblable à la fixation de colonnes, mais s'applique directement sur des séquences de deux trajets. On sélectionne les flots fractionnaires entre deux trajets effectués consécutivement dans un horaire et prenant des valeurs proches de 1 et on les fixe à l'entier supérieur. En pratique, cette méthode va imposer que le trajet j soit effectué après le trajet i dans les sous-problèmes en ajoutant des contraintes de ressources. Ici, on ne fixe pas les arcs directement, ce qui veut dire que l'autobus pourrait, par exemple, aller à la recharge entre les deux trajets. On impose seulement que ces deux trajets soient effectués un à la suite de l'autre.

Pour terminer, le dernier type de branchement que l'on utilise si les deux autres ne sont pas suffisants est la fixation du flot sur un arc. Ce type de branchement est rarement appliqué, car il n'est pas compatible avec le *diving* que l'on tente de faire. En effet, ce branchement vient agir directement sur les flots des arcs dans un graphe $\mathcal{G}^k$ et crée deux branches en sélectionnant un arc ayant un flot fractionnaire $\bar{y}$ et en le fixant dans une branche, à $y \geq \lceil \bar{y} \rceil$ et dans l'autre, à $y \leq \lfloor \bar{y} \rfloor$. Pour l'appliquer, on ajoute une contrainte dans le PM, à l'exception

du cas où on impose $y \leq 0$. Dans ce cas, on se contente simplement de retirer l'arc du graphe $\mathcal{G}^k$ correspondant. Les deux autres types de branchement sont suffisants pour assurer des flots entiers sur les arcs entre les trajets, mais peuvent laisser des flots fractionnaires sur les arcs de recharge, d'attente et sur ceux allant à un dépôt en milieu de journée. La fixation de flot sur un arc vient corriger cela et force ces arcs à avoir des flots entiers.

On peut donc résumer la méthode de résolution de base comme suit : on applique la génération de colonnes sur le PM jusqu'à obtenir une solution optimale du PM initial. S'il ne s'agit pas d'une solution entière, on évalue les différents types de branchement selon la solution obtenue et on applique celui qui obtient le meilleur score. On réapplique ensuite la génération de colonnes jusqu'à atteindre la solution optimale avec les nouvelles contraintes imposées ou jusqu'à ce que l'on atteigne la borne inférieure du problème. On répète l'opération à chaque nœud de branchement et on cesse l'exploration si la résolution d'une relaxation linéaire renvoie une solution entière ou si la borne inférieure fournie par une relaxation linéaire est plus grande ou égale au coût de la meilleure solution entière trouvée.

CHAPITRE 4 GESTION DE LA DÉGÉNÉRESCENCE

Le modèle (3.9) du MDEVSP comporte un très grand nombre de variables et des contraintes de partitionnement d'ensemble (3.9b) qui sont reconnues pour engendrer de la dégénérescence, particulièrement lorsque les variables comportent un assez grand nombre de coefficients égaux à 1 dans ces contraintes (par exemple, plus de 10 trajets par horaire d'autobus en moyenne). Cette dégénérescence ralentit la convergence de la génération de colonnes et aggrave ce qu'on appelle l'effet de queue (*long-tail effect*) où, lorsque l'on approche de la valeur optimale du problème maître, on fait de très nombreuses itérations de génération de colonnes sans arriver à faire diminuer rapidement la valeur de la fonction objectif. Cela se traduit généralement par de grandes oscillations de la valeur des variables duales et des temps de calcul très longs. Ce mémoire vise à développer une méthode pour combattre la dégénérescence et réduire son impact sur les temps de calcul. Nous avons mentionné plusieurs approches à la section 2.2 desquelles nous nous sommes inspirés pour développer une nouvelle stratégie prometteuse. Ce chapitre fera une description détaillée des méthodes utilisées pour combattre la dégénérescence, soit principalement celles introduites par Popovic (2023), auxquelles nous ajoutons quelques raffinements. Nous commencerons par décrire la méthode de perturbation, puis nous présenterons l'approche utilisant des inégalités duales en parlant de leur potentiel dans une première section, en décrivant comment on les génère dans une seconde section et, finalement, en présentant le problème maître avec inégalités duales (PM^{ID}) dans une troisième section. Nous terminerons ensuite ce chapitre en définissant un processus de réparation applicable si les inégalités duales générées nous empêchent d'obtenir une solution réalisable au problème maître original.

4.1 Perturbation

La première technique que l'on utilise pour combattre la dégénérescence est la perturbation (Charnes, 1952). Elle permet de couvrir les trajets légèrement plus ou légèrement moins qu'une fois en ajoutant des variables de perturbation sur certaines contraintes. Pour chaque trajet $t \in \mathbf{T}$, on considère les variables η_t^+ et η_t^- et on les borne par les intervalles $[0, \epsilon_t^+]$ et $[0, \epsilon_t^-]$ où ϵ_t^+ et ϵ_t^- sont choisies aléatoirement dans $[0, 10^{-4}]$. Avec ces nouvelles variables, les contraintes (3.9b) deviennent :

$$\sum_{k \in \mathbf{K}} \sum_{s \in \Omega^k} a_s^t x_s + \eta_t^+ - \eta_t^- = 1 \quad \forall t \in T \tag{4.1}$$

auxquelles s'ajoutent les bornes :

$$0 \leq \eta_t^+ \leq \epsilon_t^+ \quad \forall t \in \mathbf{T} \tag{4.2a}$$

$$0 \leq \eta_t^- \leq \epsilon_t^- \quad \forall t \in \mathbf{T} \tag{4.2b}$$

Cette formulation permet, comme on l'a mentionné, de violer très légèrement les contraintes et l'ajout de petites variations aléatoires réduit le nombre de variables nulles dans les solutions de base. Cela permet de réduire la dégénérescence et d'accélérer la convergence de l'algorithme de génération de colonnes.

Nous avons mentionné à la section 3.3.3 qu'un type de décision supplémentaire impliquant les variables de perturbation était possible lors du choix de type de branchement. En effet, les variables de perturbation sont insérées dans le PM avant de débuter la résolution et, par leur nature, permettent rarement d'obtenir une solution entière puisque les contraintes (4.1) autorisent la couverture partielle d'un trajet. Ces variables sont, donc, généralement retirées lors des derniers nœuds de branchement, lorsque la taille du problème maître est rendue suffisamment petite et que le risque de dégénérescence est presque nul.

4.2 Inégalités duales

La technique de perturbation que nous utilisons réduit la dégénérescence, mais dans certains cas, les temps de calcul restent très longs. Dans ce mémoire, nous cherchons surtout à développer une méthode additionnelle pour réduire davantage la dégénérescence et accélérer la résolution. Nous avons présenté à la section 2.2 le potentiel des inégalités duales pour stabiliser et accélérer la génération de colonnes. En effet, il a été prouvé que l'ajout d'inégalités duales dans la formulation de certains problèmes permet de réduire la taille et la fréquence des oscillations des variables dans l'espace dual, ce qui accélère la convergence.

Dans notre cas, on a trois types de variables duales sur lesquelles on peut appliquer des inégalités. Pour chaque contrainte (3.9b), (3.9c) et (3.9d), il existe, respectivement, une variable π_t, λ_k et $\mu_{p,h}$.

Concentrons-nous sur les variables π_t. La valeur d'une de ces variables duales représente le coût supplémentaire à payer dans la solution optimale pour couvrir le trajet t. C'est sur ces variables qu'on appliquera les inégalités duales. Soit une inégalité de la forme suivante :

$$\pi_i \geq \pi_j - \phi_{ij}, \quad \phi_{ij} \geq 0 \tag{4.3}$$

On borne respectivement les deux variables duales π_i et π_j inférieurement et supérieurement et on donne une petite marge de manœuvre en y ajoutant le coefficient ϕ_{ij}. S'ensuit que si on ajoute une autre égalité $\pi_j \geq \pi_k - \phi_{jk}$, on peut contraindre la valeur de π_j entre $\pi_k - \phi_{jk}$ et $\pi_i + \phi_{ij}$. En bornant ainsi les valeurs des variables duales, on vient, comme c'était notre intention, limiter l'amplitude des oscillations de ces variables dans l'espace dual. Pour utiliser cette idée à son plein potentiel en supposant que les coefficients ϕ sont tous nuls, on peut chercher à créer des chaînes d'inégalités de type

$$\pi_i \geq \pi_j \geq \pi_k \geq \cdots \geq \pi_n \tag{4.4}$$

et ainsi stabiliser un grand nombre de variables à la fois. Le problème qui survient est que, dans la plupart des cas mentionnés dans la littérature, les inégalités duales qui sont ajoutées sont des inégalités que l'on peut déterminer à partir de la structure même du problème. Pour un bon exemple, on invite le lecteur à consulter Mhamedi *et al.* (2022) qui applique ce genre d'idées sur le problème de tournées de véhicules à deux échelons. Dans le MDEVSP, aucune structure du genre n'est connue pour identifier des inégalités duales satisfaites par une solution duale optimale du PM. On doit donc trouver une méthode alternative pour générer les chaînes d'inégalités.

4.2.1 Prédiction et sélection d'inégalités duales

Dans cette section, nous présentons les méthodes utilisées pour générer les chaînes d'inégalités duales à ajouter au PM. L'objectif est d'utiliser un GNN pour faire des prédictions sur le sens des inégalités entre les différentes variables duales $\pi_t, t \in \mathbf{T}$ et d'ensuite utiliser ces prédictions pour construire un graphe qui servira à déduire des chaînes d'inégalités. On réfère le lecteur à Popovic (2023) pour la description complète du GNN et des méthodes d'apprentissage machine utilisées. Par contre, toutes les autres étapes du processus seront expliquées en détail dans cette section.

Le GNN développé par Popovic prend en entrée un graphe $\mathcal{Q}$ représentant le réseau de connexions avec des vecteurs de caractéristiques propres à chaque nœud et arête et renvoie en sortie le vecteur $\vec{\hat{y}}$ contenant pour chaque paire de variables duales (π_i, π_j), une estimation de la probabilité $\hat{y}_{ij}$ que π_i soit plus grand ou égal à π_j. On peut ensuite utiliser ces prédictions pour construire un graphe $\mathcal{GI}$ (figure 4.1) contenant un nœud pour chaque variable duale π_t, ainsi qu'un nœud source (S) et un nœud puits (P). Des arcs (en noir) relient le nœud S à chaque nœud π_t de même que chaque nœud π_t au nœud P. On ajoute ensuite les arcs suivants (en bleu) :

— Si $\hat{y}_{ij} > (1 - \beta)$, on ajoute l'arc (π_i, π_j)
— Si $\hat{y}_{ij} < \beta$, on ajoute l'arc (π_j, π_i)

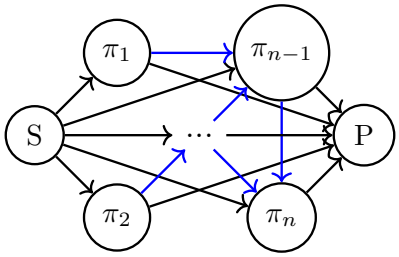

FIGURE 4.1 Graphe $\mathcal{GI}$

où β est un seuil d'acceptation. Notons que dans ses travaux, Popovic a choisi un seuil d'acceptation de 0.15, mais des tests supplémentaires sur des valeurs de β plus faibles seront présentés à la section 5.3.2. L'approche présentée ici se base sur le postulat que, si la probabilité que π_i soit plus grand que π_j est très basse, cela implique que la probabilité que π_j soit plus grande que π_i est grande. Puisque l'on travaille avec des probabilités et que le modèle de prédiction peut faire des erreurs, il est possible que cette stratégie crée des cycles dans le graphe. Deux stratégies sont utilisées pour réduire la probabilité d'erreurs dans le graphe.

La première repose sur l'idée que "*plus souvent une variable duale est prédite comme étant inférieure à une autre, plus sa valeur devrait être petite*" (Popovic, 2023). Soit le degré d'un nœud π_i :

$$\deg(\pi_i) = \text{in}(\pi_i) - \text{out}(\pi_i) \tag{4.5}$$

où $\text{in}(\pi_i)$ est le nombre d'arcs entrant dans le nœud et $\text{out}(\pi_i)$ est le nombre d'arcs sortant du nœud. Rappelons que le nombre d'arcs entrant dans un nœud de ce graphe correspond au nombre de fois que cette variable a été prédite comme inférieure à une autre et le nombre d'arcs sortants correspond au nombre de fois où elle a été prédite comme supérieure. On peut supposer que si on prédit $\pi_i \geq \pi_j$, mais que $\deg(\pi_i) > \deg(\pi_j)$, il est probable que cette inégalité soit une erreur, car la valeur de π_i devrait être plus petite que π_j. Pour réduire les probabilités d'erreur, on retire donc tous ces arcs et on garde uniquement les arcs (π_i,π_j) tels que $\deg(\pi_i) < \deg(\pi_j)$.

On veut trouver un chemin sans cycle du noeud S au noeud P pour obtenir une chaîne d'inégalités duales (2.2). Pour ce faire, Popovic (2023) propose de résoudre un problème de plus court chemin en mettant un coût de -1 sur chaque arc afin de trouver la chaîne la plus longue. Pour cette raison, on doit vérifier si le graphe contient des cycles. Comme on utilise

l'algorithme de Bellman-Ford pour trouver le chemin de coût minimum, le graphe ne doit contenir aucun cycle de coût négatif. Si des cycles sont détectés, on retire simplement tous les arcs qui les composent. Comme il est impossible de déterminer où le modèle de prédiction a fait une erreur, retirer tous les arcs du cycle nous permet d'être certains que nous avons retiré l'arc qui est problématique. De plus, si l'on diminue davantage la valeur du seuil β comme nous le ferons dans le prochain chapitre, on remarque que très peu de cycles sont créés. Le choix de retirer tous les arcs dans les cycles n'a donc que très peu d'influence sur la densité d'arcs dans le graphe.

Finalement, si le graphe ne contient aucun cycle de coût négatif, l'algorithme de Bellman-Ford trouve le plus court chemin ou le chemin de coût minimum entre le nœud source et le nœud puits. Puisque les coûts sur tous les arcs dans le graphe sont égaux et négatifs, appliquer Bellman-Ford sur celui-ci renverra le plus long chemin possible entre les deux nœuds. Une fois la chaîne d'inégalités identifiée, on retire tous les arcs correspondants du graphe $\mathcal{GI}$, puis on répète l'opération jusqu'à ce que l'on ait atteint le nombre maximal d'inégalités choisi ou jusqu'à ce que le graphe soit vide. L'algorithme 2 présente l'algorithme complet de tri des arcs dans le graphe $\mathcal{GI}$ et de sélection des chaînes d'inégalités sous forme de pseudocode.

Dans l'algorithme 2, les fonctions `nodes`(G) et `arcs`(G) renvoient respectivement la liste des nœuds et des arcs compris dans le graphe G. Ensuite, les fonctions `from`(arc) et `to`(arc) renvoient le nœud d'origine et de destination de l'arc donné en argument. La fonction `removeArc`(arc, G) retire simplement l'arc donné en argument du graphe G et, de la même façon, la fonction `removeCycle`(cycle, G) retire tous les arcs qui forment le cycle du graphe G si un cycle de coût négatif est trouvé par l'algorithme de Bellman-Ford. En effet, la fonction `BellmanFord`(S,P,$\mathcal{GI}$) renvoie, si elle en trouve un, la liste des nœuds formant un cycle de coût négatif. Si le graphe n'en contient pas, la fonction renvoie le chemin de coût minimum entre S et P. La fonction `len`(x) donne le nombre d'arcs dans la chaîne ou le cycle qui est donné en argument et, pour finir, les fonctions `extractIneqs`(chaîne) et `removeArcsIneqs`($\mathcal{GI}$, ineqsChaîne) retournent la liste des arcs compris dans la chaîne donnée en argument et retirent toutes ces inégalités du graphe $\mathcal{GI}$, respectivement.

4.2.2 Problème maître avec inégalités duales

Maintenant, voyons comment l'ajout de ces inégalités affecte la formulation du PM. En plus du coefficient ϕ_{ij}, chaque inégalité ajoutée dans le problème dual ajoutera une nouvelle variable w_{ij} au primal. Si la nouvelle variable w_{ij} prend une valeur ($w_{ij} > 0$), cela signifie que le trajet i sera couvert $1 + w_{ij}$ fois et que le trajet j sera couvert $1 - w_{ij}$ fois. Soulignons également que dans le primal, ϕ_{ij} devient un coefficient de la nouvelle variable w_{ij}. En termes

Algorithme 2 Sélection des inégalités duales

Entrée: $\mathcal{GI}$, nbMaxIneq
  Pour $\pi_i \in$ nodes($\mathcal{GI}$) **faire** ▷ Calcul des degrés
    deg(π_i) ← in(π_i) - out(π_i)
  Fin Pour
  Pour arc in arcs($\mathcal{GI} \backslash \{S, P\}$) **faire** ▷ Suppression d'arcs selon les degrés
    $\pi_{début} \leftarrow from(arc)$
    $\pi_{fin} \leftarrow to(arc)$
    Si deg($\pi_{début}$) > deg(π_{fin}) **alors**
      removeArc(arc, $\mathcal{GI}$)
    Fin Si
  Fin Pour

  Tant que vrai **faire** ▷ Suppression des cycles
    cycle_neg ← BellmanFord(S,P,$\mathcal{GI}$)
    Si len(cycle_neg) > 0 **alors**
      $\mathcal{GI}$ ←removeCycle(cycle_neg, $\mathcal{GI}$)
    Sinon
      Sortie
    Fin Si
  Fin Tant que

  ineqs ← ∅
  Tant que |ineqs| ≤ nbMaxIneq **et** arcs($\mathcal{GI}$) ≠ ∅ **faire** ▷ Sélection des chaînes
    chaîne ← BellmanFord(S,P,$\mathcal{GI}$)
    Si len(chaîne) > 3 **alors**
      ineqsChaîne ← extractIneqs(chaîne)
      ineqs ← ineqs ∪ ineqsChaîne
      $\mathcal{GI}$ ← removeArcsIneqs($\mathcal{GI}$, ineqsChaîne)
    Sinon
      Sortie
    Fin Si
  Fin Tant que

pratiques, ϕ_{ij} est le coût supplémentaire à payer pour une augmentation d'une unité de $w_{ij} > 0$. On obtient la forme suivante pour le PM^{ID} :

$$\min \quad \sum_{k\in\mathbf{K}} \sum_{s\in\Omega^k} c_s x_s + \sum_{i\in\mathbf{E}} \sum_{j\in\mathbf{E}_i^+} \phi_{ij} w_{ij} \tag{4.6a}$$

$$\text{s.c.} \quad \sum_{k\in\mathbf{K}} \sum_{s\in\Omega^k} a_s^i x_s = 1 \qquad \forall i \in \mathbf{T}\backslash\mathbf{E} \tag{4.6b}$$

$$\sum_{k\in\mathbf{K}} \sum_{s\in\Omega^k} a_s^i x_s + \sum_{j\in\mathbf{E}_i^+} w_{ji} - \sum_{j\in\mathbf{E}_i^-} w_{ij} = 1 \qquad \forall i \in \mathbf{E} \tag{4.6c}$$

$$\sum_{s\in\Omega^k} x_s \leq g^k \qquad \forall k \in \mathbf{K} \tag{4.6d}$$

$$\sum_{k\in\mathbf{K}} \sum_{s\in\Omega^k} o_s^{p,h} x_s \leq b_h \qquad \forall h \in \mathbf{H}, p \in \mathbf{P} \tag{4.6e}$$

$$x_s \geq 0 \qquad \forall k \in \mathbf{K}, s \in \Omega^k \tag{4.6f}$$

$$w_{ij} \geq 0 \qquad \forall i \in \mathbf{E}, j \in \mathbf{E}_i^+ \tag{4.6g}$$

où $\mathbf{E}$ est l'ensemble des trajets impliqués dans des inégalités duales, $\mathbf{E}_i^+$ est l'ensemble des trajets j tel qu'il existe une inégalité de la forme $\pi_j \geq \pi_i - \phi_{ji}$ et $\mathbf{E}_i^-$ est l'ensemble des trajets j tel qu'il existe une inégalité de la forme $\pi_i \geq \pi_j - \phi_{ij}$. On remarque que l'ajout des inégalités (et donc des variables w_{ij}) induit une relaxation du PM ce qui veut dire que la valeur optimale de PM^{ID} pourrait être une borne inférieure de moins bonne qualité que celle de PM lorsqu'au moins une variable w_{ij} prend une valeur positive. Ainsi, fixer ϕ_{ij} à une valeur positive favorise w_{ij} à prendre une petite valeur, préférablement 0. Toutefois, une trop grande valeur de ϕ_{ij} induit une inégalité duale $\pi_j \geq \pi_i - \phi_{ij}$ très peu contraignante, ce qui rendrait inutile l'ajout de cette inégalité. Pour nos tests, on a choisi une petite valeur pour ϕ_{ij}, soit $\phi_{ij} = 1$.

Comme le modèle de prédiction que l'on utilise peut faire des erreurs, il peut arriver que certaines inégalités que l'on ajoute soient invalides et qu'elles ne permettent pas d'atteindre une solution duale optimale du PM. Pour contourner ce problème, on peut faire appel à un processus de réparation.

4.2.3 Processus de réparation

Cette section présente le processus de réparation (nommé "processus de recouvrement" par Popovic (2023) ou encore "recovery process" par Gschwind et Irnich (2016)) qui sera utilisé pour retrouver autant que possible une solution réalisable du PM dans le cas où des inégalités

invalides seraient ajoutées au problème.

Pour commencer, on doit identifier pendant la résolution du problème, quelles inégalités semblent fausses. Popovic (2023) a indiqué que "*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*". C'est, en partie, ce qui sert de signal pour appliquer le processus de réparation. Si, pendant Δ itérations de génération de colonnes, la valeur de la fonction objectif ne diminue pas d'au moins g unités, et que des variables w_{ij} prennent des valeurs positives dans la solution du PMR, on arrête la résolution et on applique le processus de réparation. L'algorithme 3 décrit le processus qui est appliqué lorsque l'on arrête la résolution.

Algorithme 3 Processus de réparation

Entrée: PM^{ID}, w_{ij} $\forall(i,j)$ t.q. $w_{ij} > 0$, `recoveryCount`[i,j]
  Pour tout w_{ij} **faire**
    Si `recoveryCount`[i,j] $= 0$ **alors**
      Inversion de l'inégalité associée à la paire d'indices (i,j)
      `recoveryCount[i,j]++`
    Sinon Si `recoveryCount`[i,j] $= 1$
      Incrémentation de ϕ_{ij}
      `recoveryCount[i,j]++`
    Sinon Si `recoveryCount`[i,j] $= 2$
      Retrait de l'inégalité associée à la paire d'indices (i,j)
    Fin Si
  Fin Pour

La variable `recoveryCount`[i,j] tient le compte du nombre de fois que le processus de réparation a été appliqué sur une variable w_{ij} au cours de toutes les itérations de génération de colonnes. Cela nous permet de faire une opération différente sur l'inégalité chaque fois qu'elle est identifiée comme potentiellement fausse. En effet, la première fois qu'on applique le processus de réparation sur une inégalité associée à la paire d'indices (i,j), on inverse l'inégalité. Ce qui veut dire que $\pi_i \geq \pi_j - \phi_{ij}$ devient $\pi_j \geq \pi_i - \phi_{ij}$. Cette opération peut nous aider à retrouver une solution réalisable du PM, car si l'inégalité initiale était bel et bien fausse, son inverse devrait être vraie. Ensuite, la deuxième fois qu'une variable w_{ij} prend une valeur positive, on incrémente la valeur de ϕ_{ij} (de 1 unité dans nos tests). On pénalise donc davantage le fait que w_{ij} prenne une valeur positive en augmentant le coût associé, ce qui favorise le cas où les trajets sont couverts une et une seule fois. Finalement, si la variable w_{ij} prend une valeur positive une troisième fois, on retire simplement l'inégalité (et la variable du même coup) correspondante du PM.

Notons que dans ses travaux, Popovic (2023) utilisait seulement les deux dernières étapes du processus de réparation, c'est-à-dire qu'il ne faisait pas d'inversion des inégalités. Nous

testerons les deux variantes de la méthode au chapitre suivant. Soulignons également qu'on limite le nombre d'inégalités modifiées à chaque appel de la procédure de réparation et qu'on limite aussi le nombre d'appels à cette procédure. Ainsi, il est possible de terminer la résolution du PM^{ID} avec des variables w_{ij} prenant une valeur positive et, donc, une borne inférieure de moins bonne qualité que celle qui pourrait être obtenue en résolvant le PM directement. Dans ce cas, certains trajets peuvent être couverts plus d'une fois et d'autres moins d'une fois. Ces anomalies devront être corrigées en cours de branchement pour obtenir une solution entière réalisable au problème.

CHAPITRE 5 PRÉSENTATION ET ANALYSE DES RÉSULTATS

Présentons maintenant l'ensemble des résultats obtenus. Dans ce chapitre, nous commencerons, à la section 5.1, par introduire les instances utilisées et définir l'ensemble des paramètres qui restent fixes dans les différents tests que nous avons effectués. Ensuite, la section 5.2 présentera quelques modifications que nous avons faites par rapport aux travaux de Popovic (2023). La section 5.3 présentera une analyse que nous avons faite sur les chaînes d'inégalités violées et une série de tests que nous avons faits pour trouver une meilleure valeur de seuil d'acceptation. À la section 5.4, nous entrerons dans le vif du sujet. Nous y présenterons et analyserons l'ensemble des résultats obtenus et les différentes configurations de paramètres sur lesquelles nous avons fait des tests. Nous verrons dans cette section que la méthode proposée n'apporte pas les résultats attendus, ce qui nous mènera à la section 5.5, où nous ferons une étude plus approfondie des chaînes d'inégalités duales. Nous utiliserons ensuite ces nouveaux résultats pour proposer, dans une dernière section, une nouvelle approche mettant de côté l'apprentissage machine et qui performe mieux que ce qui a été proposé plus tôt au niveau de l'accélération de la résolution et de la qualité de la solution.

5.1 Instances étudiées et paramètres

Les instances étudiées dans ce mémoire nous ont été fournies par Popovic (2023) et sont les mêmes qui ont été étudiées dans son mémoire. Nous invitons donc le lecteur à en consulter la section 5.1 pour comprendre comment elles sont générées. Pour les besoins de nos travaux, mentionnons seulement que le réseau est composé de 4 lignes aller-retour (donc 8 lignes au total) et de deux dépôts. Pour avoir plusieurs instances, on modifie aléatoirement les heures de départ des trajets et on augmente la fréquence des lignes si l'on veut obtenir des instances de tailles différentes. Au total, on étudie un ensemble de 50 instances divisé en 25 instances de taille moyenne (M), contenant environ 850 trajets à couvrir, et 25 instances de grande taille (G) contenant environ 1050 trajets à couvrir.

Toutes nos instances sont résolues à l'aide de la version 4.5 de GENCOL (Desrosiers, 2010), un algorithme de génération de colonnes qui fait appel à CPLEX (version 22.1) pour résoudre le PMR. Les ordinateurs utilisés pour nos travaux possèdent des processeurs Intel(R) Core(TM) i7-12700 à 3.80 GHz, utilisent jusqu'à 64 Gb de RAM et exécutent la version 5.14 de Linux.

Le tableau 5.1 présente l'ensemble des paramètres qui ont été utilisés pour générer les instances et qui sont fixes sur l'ensemble des tests effectués dans ce mémoire. Rappelons à quoi

correspondent les différents paramètres. c_f est le coût fixe associé à l'utilisation d'un autobus, c_a est le coût par minute d'attente hors dépôt et c_d est le coût par minute de déplacement à vide. e_a et e_d sont respectivement la quantité d'énergie consommée par unité de temps d'attente et de déplacement. δ est le temps maximal entre deux trajets pour qu'il existe une connexion directe entre les deux et λ correspond à la durée de chaque période de temps $p \in \mathbf{P}$. Finalement, b_h est le nombre de bornes de recharge pour chaque station $h \in \mathbf{H}$ et σ_{min} et σ_{max} sont respectivement le niveau de batterie minimum et maximum qui doivent être respectés par les autobus en tout temps.

Tableau 5.1 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}	100 000 Wh

5.2 Différences avec les travaux de Popovic

Dans la section 5.4, nous présentons les résultats obtenus pour les différentes configurations que nous avons testées. Par contre, avant d'entrer dans la présentation et l'analyse de ces résultats, il est important d'aborder quelques modifications que nous avons faites par rapport aux travaux de Popovic (2023).

Premièrement, les ordinateurs que nous utilisons sont plus puissants, car ils utilisent de meilleurs processeurs et possèdent une plus grande quantité de mémoire vive. Les temps de calcul sont donc naturellement moins longs.

Ensuite, nous avons remarqué que l'algorithme de barrière utilisé pour résoudre le PMR était suivi d'un crossover à chaque itération de génération de colonnes. Cet algorithme de crossover permet de s'assurer que la solution obtenue est une solution de base (Vavasis et Ye, 2000), ce qui n'est pas utile sauf à la dernière itération. Comme cet algorithme consommait une très grande portion du temps de calcul, nous l'avons retiré, estimant que l'algorithme de barrière était suffisant, ce qui a grandement accéléré la résolution des instances. L'impact de

ces modifications est présenté à la section 5.4.

5.3 Pré-tests

Avant d'appliquer la prédiction d'inégalités duales sur la résolution complète du problème, certains tests ont été faits pour tenter de voir si d'autres conclusions pouvaient être tirées des résultats de Popovic (2023). Dans un premier temps, nous avons étudié les chaînes d'inégalités violées dans les solutions duales optimales obtenues par Popovic, à la recherche d'une certaine structure qui nous permettrait de mieux identifier les inégalités non valides parmi celles qui ont été ajoutées au problème. Les résultats de ces tests sont présentés à la sous-section 5.3.1. Ensuite, nous avons tenté de diminuer davantage le seuil d'acceptation (β) par rapport à la valeur de 0.15 choisie par Popovic pour étudier son impact sur le pourcentage d'inégalités valides introduites dans le PM et sur la densité d'arcs dans le graphe $\mathcal{GI}$. Les résultats obtenus sont présentés à la sous-section 5.3.2.

5.3.1 Tests sur les chaînes d'inégalités violées

Dans cette sous-section, nous présentons l'étude que nous avons effectuée sur les chaînes d'inégalités violées dans les solutions duales optimales obtenues par Popovic (2023). Comme on insère des chaînes d'inégalités dans le problème, une seule inégalité fausse peut avoir un impact sur la valeur de plusieurs variables dans une solution duale. On suppose donc que, si l'on détecte une chaîne d'inégalités violées dans une solution, il pourrait être possible, selon les valeurs prises par les variables w_{ij} associées à ces inégalités, d'identifier si une seule inégalité dans la chaîne est non valide. Définissons une chaîne d'inégalités violées comme une chaîne d'inégalités de longueur n

$$\pi_i \geq \pi_j \geq \cdots \geq \pi_n \tag{5.1}$$

telle que toutes les variables w_{ij} associées aux inégalités prennent une valeur positive. On peut comparer les valeurs des variables duales dans ces chaînes d'inégalités violées aux valeurs que ces mêmes variables prennent dans la solution duale optimale connue du PM. Des tests ont été faits sur plusieurs instances et, en mettant en parallèle les valeurs des variables w_{ij}, des variables duales dans le cas avec inégalités duales et des variables duales dans le cas sans inégalités duales, nous n'avons pas été en mesure d'exhiber une structure exploitable pour le traitement des chaînes d'inégalités duales violées. Notons tout de même qu'ici, nous ne connaissons qu'une seule solution duale optimale et nous comparons la solution obtenue à celle-ci. Il est donc possible que nous n'ayons simplement pas toutes les informations

nécessaires pour trouver la structure que nous cherchons dans les chaînes d'inégalités duales.

5.3.2 Tests sur le paramètre β

L'autre aspect qui a été testé sur le noeud 0 avant d'ajouter le branchement dans la résolution du problème concerne le choix du seuil d'acceptation β. En effet, dans les travaux de Popovic (2023), une valeur de $\beta = 0.15$ avait été choisie, mais aucun test n'était présenté avec des valeurs inférieures à 0.15. Nous avons donc voulu voir si, en diminuant la valeur de β, on pouvait obtenir un pourcentage d'inégalités valides plus grand tout en gardant un nombre suffisant d'inégalités prédites pour avoir de longues chaînes d'inégalités. Notons qu'une inégalité est considérée valide si elle est satisfaite par la solution duale optimale obtenue sans inégalités duales. Nous avons testé différentes valeurs de β entre 0.01 et 0.15 pour lesquelles nous avons calculé le pourcentage d'inégalités valides moyen parmi celles prédites (précision), le nombre d'inégalités moyen dans le graphe $\mathcal{GI}$ et la longueur moyenne des chaînes sélectionnées si on impose un maximum de 900 inégalités duales. Les résultats sont présentés dans le tableau 5.2.

Tableau 5.2 Pourcentage d'inégalités valides, nombre d'inégalités dans le graphe $\mathcal{GI}$ et longueur moyenne des chaînes en fonction du seuil d'acceptation $\beta \leq 0.15$

	Précision		Inégalités		Longueur chaînes	
β	M	G	M	G	M	G
0.01	99.7%	99.4%	136 840.4	219 590.2	7.9	8.7
0.03	99.2%	98.8%	175 843.2	281 379.8	11.6	12.8
0.05	98.6%	98.3%	199 059.6	316 198.9	13.8	15.7
0.1	97.5%	97.2%	236 881.9	372 634.4	20.9	24.4
0.15	96.3%	96.1%	264 054.4	413 346.0	28.1	32.9

Comme on peut s'y attendre, on voit que plus on diminue β, plus le nombre d'inégalités et la longueur moyenne des chaînes diminuent. Cependant, en diminuant β, on augmente considérablement la précision jusqu'à obtenir des pourcentages d'inégalités valides supérieurs à 99% lorsque le seuil d'acceptation est égal à 0.01. Évidemment, on cherche toujours à avoir la plus grande précision possible sur les prédictions, mais on doit également trouver le juste milieu entre la précision et le nombre d'inégalités prédites. Une précision plus grande réduit le nombre d'inégalités fausses insérées, mais des chaînes plus longues restreindront davantage l'espace dual et devraient avoir un effet plus grand sur l'accélération. On doit donc aussi considérer l'effet de la variation du seuil d'acceptation sur les temps de résolution et sur l'écart avec la solution sans inégalités duales.

Le tableau 5.3 présente les facteurs d'accélération moyens et les écarts moyens avec la solution sans inégalités duales pour les différentes valeurs de β. Nous y présentons également les résultats sur la résolution complète du problème, car ils sont importants dans le choix de la valeur de β. Rappelons que les deux métriques principales utilisées pour analyser la performance des différentes instances sont le facteur d'accélération (2.3) et le gap (2.4) présentés à la section 2.4.2. Un gap positif signifie qu'en ajoutant les inégalités duales on obtient une plus petite valeur de la fonction objectif et, à l'inverse, un gap négatif signifie qu'on obtient une plus grande valeur. Les bornes inférieures plus petites au nœud 0 sont expliquées par le fait qu'ajouter des inégalités duales relaxe le problème. En effet, l'ajout des variables w_{ij} permettant de couvrir les trajets légèrement plus ou moins qu'une fois nous permet d'obtenir de meilleures bornes inférieures. À l'inverse, les gaps négatifs obtenus sur la résolution complète du problème représentent une perte dans la qualité de la solution par rapport à la solution sans inégalités duales. Dans le tableau, M correspond aux instances de taille moyenne comprenant environ 850 trajets et G, aux instances de grande taille comprenant environ 1050 trajets.

Tableau 5.3 Facteurs d'accélération et gaps pour différentes valeurs de β

	Noeud 0				Résol. complète			
	Acc.		Gap (%)		Acc.		Gap (%)	
β	M	G	M	G	M	G	M	G
0.01	1.01	1.04	0.003	0.02	0.80	0.91	0.0005	0.003
0.03	0.84	0.82	0.05	0.15	0.73	0.62	0.002	-0.04
0.05	0.96	0.94	0.09	0.15	0.79	0.62	0.005	-0.04
0.10	1.41	1.22	0.16	0.14	0.52	0.45	0.06	-0.09
0.15	1.51	1.33	0.15	0.15	0.48	0.36	0.24	-0.12

On y constate que, comme dans les travaux de Popovic, les instances où $\beta = 0.15$ sont celles qui ont la meilleure accélération au nœud 0, mais ce sont également celles qui ont la pire accélération sur la résolution complète du problème et les plus grands écarts avec la solution sans inégalités duales. À l'inverse, les instances où $\beta = 0.01$ n'ont presque pas d'accélération au nœud 0, mais elles ont des gaps très faibles et présentent les facteurs d'accélération les plus prometteurs sur la résolution complète du problème. C'est donc cette valeur que nous utiliserons comme seuil d'acceptation pour la suite des travaux. Soulignons tout de même qu'aucun de ces groupes d'instances ne présente un facteur d'accélération plus grand que 1, ce qui signifie que dans cette forme, notre méthode ne semble pas appropriée pour accélérer la génération de colonnes. Dans les prochaines sections, nous faisons varier quelques paramètres de nos instances pour tenter de trouver une configuration où l'on obtient des accélérations significatives sur la résolution complète du problème.

5.4 Résultats avec branchement

Présentons maintenant les résultats complets obtenus en appliquant nos méthodes pour une résolution complète du problème. Ici, les deux groupes M et G seront subdivisés en deux sous-groupes selon le temps de résolution médian du cas sans inégalités duales. Les instances prenant plus de temps que le temps médian à être résolues seront catégorisées comme "lentes" (L) et celles prenant moins de temps seront catégorisées comme "rapides" (R). On présentera donc les facteurs d'accélération des instances (G L), (G R), (M L) et (M R) de façon indépendante. Cela nous permet notamment d'isoler l'impact de notre méthode sur les instances les plus dégénérées, i.e., les instances grandes et lentes, qui sont les instances que notre méthode vise à accélérer. Tout au long de cette section, nous présentons en parallèle les temps moyens de résolution du PM (sans inégalités duales) ainsi que les facteurs d'accélération et les gaps associés au PM^{ID} (avec inégalités duales).

Le tableau 5.4 montre les facteurs d'accélération pour le cas où on impose une limite de 900 inégalités duales à sélectionner avec $\beta = 0.01$, $\phi = 1$ et aucune itération du processus de réparation. Ici, on veut surtout montrer l'impact des modifications présentées à la section 5.2 sur les temps de calcul et les facteurs d'accélération. Ce sera en quelque sorte notre valeur de référence à laquelle nous comparerons les autres configurations dans les sous-sections suivantes.

Tableau 5.4 Accélérations et gaps par rapport à la solution obtenue sans inégalités duales, 900 inégalités, aucune itération du processus de réparation

	PM		PM^{ID}					
	Noeud 0	Résol. complète	Noeud 0			Résol. complète		
Taille	Temps (s)	Temps (s)	Acc. L	Acc. R	Gap (%)	Acc. L	Acc. R	Gap (%)
M	334	703	1.14	0.90	0.003	0.80	0.80	-0.0005
G	894	1636	1.17	0.91	0.02	1.14	0.70	-0.003

Un résultat important à discuter ici est le temps moyen de résolution du PM. Dans ses travaux, Popovic (2023) a présenté des temps moyens de résolution de la première relaxation linéaire pour le cas sans inégalités duales de 2273.65 secondes pour les instances M et de 7675.69 secondes pour les instances G. Ici, pour la même opération, nous obtenons des temps de calcul moyens respectifs de 334 et 894 secondes. Pour la résolution complète du problème, nos temps de calcul moyens sont de 703 secondes et 1636 secondes pour les instances M et G respectivement. Cette différence dans les temps de calcul montre bien l'impact du retrait du crossover et de l'utilisation d'ordinateurs plus performants.

Les résultats relatifs aux facteurs d'accélération et aux gaps sont des versions plus détaillées de ceux présentés au tableau 5.3. On peut y voir que le seul groupe pour lequel on obtient une accélération sur la résolution complète du problème est le groupe d'instances (G L), ce qui correspond aux instances visées par notre méthode. Cependant, une accélération de 14% en moyenne est peu significative. En effet, on considère qu'une accélération inférieure à 20% est négligeable, car elle peut tomber dans la marge d'erreur expérimentale et son bénéfice ne justifie souvent pas l'effort nécessaire à son implémentation. C'est pourquoi nous poursuivrons cette section en testant différentes configurations et nous verrons s'il est possible d'améliorer légèrement les résultats en modifiant certains paramètres.

5.4.1 Ajout du processus de réparation

Dans cette section, nous ajoutons le processus de réparation à l'algorithme de résolution du problème et nous étudions son impact sur les temps de calcul et sur le gap obtenu. Comme nous l'avons mentionné à la section 4.2.3, nous introduisons deux variantes du processus de réparation. La première, appelée "processus de réparation classique" est la stratégie développée par Popovic (2023) où l'on incrémente ϕ_{ij} la première fois que la variable correspondante w_{ij} prend une valeur positive, puis on retire l'inégalité la seconde fois. Dans la deuxième variante nommée "processus de réparation avec inversion", on commence par inverser l'inégalité la première fois que w_{ij} prend une valeur positive, puis on suit ensuite le processus développé par Popovic. Le tableau 5.5 présente les résultats pour les deux variantes et les compare avec les accélérations et les gaps obtenus sans réparation. Les résultats sans réparation correspondent à ceux présentés au tableau 5.4.

On remarque que, bien qu'ils présentent des gaps semblables, le processus de réparation classique permet d'obtenir des facteurs d'accélération moyens légèrement de meilleure qualité. Nous estimons que d'inverser naïvement les inégalités la première fois qu'elles sont détectées comme violées est peut-être une approche trop simpliste et ne nous permet pas de converger rapidement vers une solution duale du PM où toutes les inégalités duales sont satisfaites. Nous avons mentionné à la section 5.3.1 que lorsque l'on détecte une chaîne d'inégalités violées, il est possible qu'une seule d'entre elles soit en réalité fausse. La stratégie du processus de réparation avec inversion ne prend pas cela en compte et peut, à certains moments, inverser l'entièreté des inégalités d'une chaîne si toutes les variables w_{ij} associées à celles-ci prennent une valeur positive. Une telle opération ne nous permet pas de converger vers la solution duale optimale connue et peut créer plus de problèmes qu'elle n'en résout.

Tableau 5.5 Accélérations et gaps avec la solution obtenue sans inégalités duales, 900 inégalités, processus de réparation classique, réparation avec inversion et sans réparation

	Noeud 0			Résol. complète		
Taille	Acc. L	Acc. R	Gap (%)	Acc. L	Acc. R	Gap (%)
Réparation classique						
M	1.02	0.58	0.0007	0.79	0.55	-0.0001
G	0.92	0.81	0.01	0.70	0.56	-0.0002
Réparation avec inversion						
M	0.83	0.75	0.0007	0.67	0.71	-0.001
G	0.89	0.57	0.003	0.66	0.52	-0.002
Sans réparation						
M	1.14	0.90	0.003	0.80	0.80	-0.0005
G	1.17	0.91	0.02	1.14	0.70	-0.003

Les processus de réparation sont également généralement coûteux en temps de calcul, car une réoptimisation est nécessaire après chaque modification faite aux contraintes, ce qui explique que les facteurs d'accélération obtenus soient plus faibles. Notons tout de même que l'objectif premier du processus de réparation est d'améliorer la solution obtenue par rapport aux instances où on ne l'utilise pas et que cet objectif est accompli sur l'ensemble des instances par le processus de réparation classique. Par contre, nous jugeons que l'amélioration dans la qualité de la solution est trop minime pour justifier la perte au niveau des temps de calcul. Nous avons donc décidé de mettre de côté le processus de réparation pour la suite des travaux.

5.4.2 Variation du nombre d'inégalités

Dans cette section, nous étudions l'impact du nombre d'inégalités duales ajoutées au problème sur le facteur d'accélération et sur le gap. On résout le MDEVSP comme on l'a fait dans les sections précédentes, mais on modifie le nombre maximum d'inégalités que l'on ajoute au problème lors de la phase de sélection des inégalités. On impose dans un cas un maximum de 600 inégalités et dans l'autre un maximum de 1200 inégalités. Dans tous les cas, le maximum d'inégalités a été atteint. Le tableau 5.6 montre les résultats avec 600, 900 et 1200 inégalités. Encore une fois, les résultats avec 900 inégalités sont ceux présentés au tableau 5.4. L'objectif de ces tests était de voir si les mauvaises performances de notre méthode jusqu'ici pouvaient être causées par une sur-stabilisation du problème ou, à l'inverse, par un trop petit nombre d'inégalités ajoutées.

TABLEAU 5.6 Accélérations et gaps avec la solution obtenue sans inégalités duales, 600 à 1200 inégalités, aucune itération du processus de réparation

	Noeud 0			Résol. complète		
Taille	Acc. L	Acc. R	Gap (%)	Acc. L	Acc. R	Gap (%)
600 inégalités						
M	1.03	0.98	0.002	0.77	0.90	-0.0003
G	1.14	1.02	0.02	1.01	0.67	-0.002
900 inégalités						
M	1.14	0.90	0.003	0.80	0.80	-0.0005
G	1.17	0.91	0.02	1.14	0.70	-0.003
1200 inégalités						
M	0.97	1.03	0.002	0.77	0.84	-0.0007
G	1.20	0.94	0.03	0.84	0.67	-0.0006

En comparant ces résultats avec ceux obtenus avec 900 inégalités duales, on remarque que dans l'ensemble, ces deux configurations performent légèrement moins bien que le cas avec 900 inégalités duales, notamment au niveau de l'accélération des instances (G L) que nous avons déjà identifiée comme le critère le plus important. En moyenne, considérer 900 inégalités ou 600 donne une performance semblable. Lors de la sélection des inégalités, nous retirons les arcs qui forment les chaînes d'inégalités que nous ajoutons au problème, mais pas les nœuds, ce qui permet à une même variable d'être impliquée dans plusieurs inégalités duales. Il est donc possible que certaines inégalités redondantes soient ajoutées au PM, ce qui expliquerait que des différences de 300 et 600 inégalités n'aient que peu d'effet sur la qualité des solutions et sur l'accélération obtenue.

Pour conclure cette section, de toutes les configurations testées, l'algorithme sans processus de réparation et contenant un maximum de 900 inégalités duales est celui qui performe le mieux. Cependant, même les variantes qui performent le mieux ne présentent pas des facteurs d'accélération satisfaisants et causent même des temps de calcul plus longs sur la majorité des instances moins dégénérées. Nous tenions à présenter ces résultats parce qu'ils témoignent des nombreux tests que nous avons effectués et montrent tout de même que la prédiction d'inégalités duales a un certain potentiel. Cependant, les résultats obtenus sont loin d'être ceux que nous attendions, ce qui nous a poussés à remettre en question notre compréhension des inégalités duales et à revérifier certaines hypothèses qui ont été faites à différentes étapes du projet. La suite de ce chapitre sera une analyse des chaînes d'inégalités valides que nous

effectuons dans le but d'approfondir notre compréhension des conditions nécessaires à la stabilisation de la génération de colonnes à l'aide de chaînes d'inégalités duales.

5.5 Étude des inégalités valides

Les résultats présentés à la section précédente ont démontré que notre méthode de prédiction d'inégalités duales, bien qu'elle sélectionne des inégalités valides avec plus de 99% de précision, n'est pas efficace pour accélérer la génération de colonnes dans le contexte du MDEVSP. Du moins, elle n'est pas efficace dans sa forme actuelle, ce qui nous a confirmé que notre compréhension des chaînes d'inégalités duales et des conditions nécessaires pour qu'elles stabilisent la génération de colonnes n'était pas complète. Nous avons donc décidé de pousser plus loin les tests qui avaient été faits par Popovic (2023) sur des chaînes d'inégalités duales valides. Dans ses tests, Popovic créait un graphe $\mathcal{GI}$ à partir des valeurs des variables duales de la solution duale optimale connue d'une instance et sélectionnait ensuite les trois chaînes les plus longues qu'il pouvait trouver dans ce graphe. Il insérait ensuite ces chaînes d'inégalités dans le PM et résolvait la première relaxation linéaire du problème. À l'aide de cette méthode, il a montré des facteurs d'accélération moyens respectifs de 5.33 pour les instances de taille moyenne et de 3.34 pour les instances de grande taille. Dans cette section, nous aborderons quelques modifications que nous avons faites à cette approche et qui nous permettront de faire une analyse plus approfondie de l'impact de la forme des chaînes d'inégalités duales sur l'accélération de la génération de colonnes.

La première modification que nous avons faite est dans la façon de sélectionner les inégalités. Dans la solution duale optimale connue, nous avons simplement décidé de classer les valeurs des variables duales en ordre croissant, ce qui nous permet de créer une longue chaîne d'inégalités valides reliant toutes les variables duales. On stabilise ainsi toutes les variables duales à l'aide d'une seule chaîne, ce qui évite d'ajouter des inégalités redondantes qui augmenteraient inutilement la taille du problème.

L'autre modification que nous avons faite est l'ajout de variations sur cette longue chaîne de façon à étudier différentes propriétés. Introduisons les deux types de variations que l'on propose. Dans un premier temps, on découpe la chaîne d'inégalités en plus petits segments en retirant des inégalités. Par exemple, pour une chaîne de 300 inégalités, si l'on veut 100 chaînes au lieu d'une, on peut garder les trois premières et retirer la suivante, puis garder les trois inégalités suivantes, et ainsi de suite, de façon à avoir 100 chaînes indépendantes de longueur égale. Ce type de manipulations nous permet d'étudier l'impact de la longueur des chaînes d'inégalités sur l'accélération de la résolution du problème. Comme une partie importante de notre démarche jusqu'ici visait à favoriser la création de longues chaînes d'inégalités, ce test

nous permettra de confirmer s'il s'agissait de la bonne approche.

Dans un deuxième temps, nous modifions la longue chaîne d'inégalités en y intégrant des "sauts". Une fois les variables duales classées en ordre croissant dans la solution duale optimale connue, on peut, au lieu de sélectionner les variables dans l'ordre pour créer des inégalités, les sélectionner en faisant des sauts de longueur constante. La figure 5.1 montre le cas où l'on aurait un total de 9 variables duales et on ferait des sauts de 3. Dans cette figure, les variables duales π_1 à π_9 sont en ordre croissant et chaque couleur correspondrait à une chaîne d'inégalités différente.

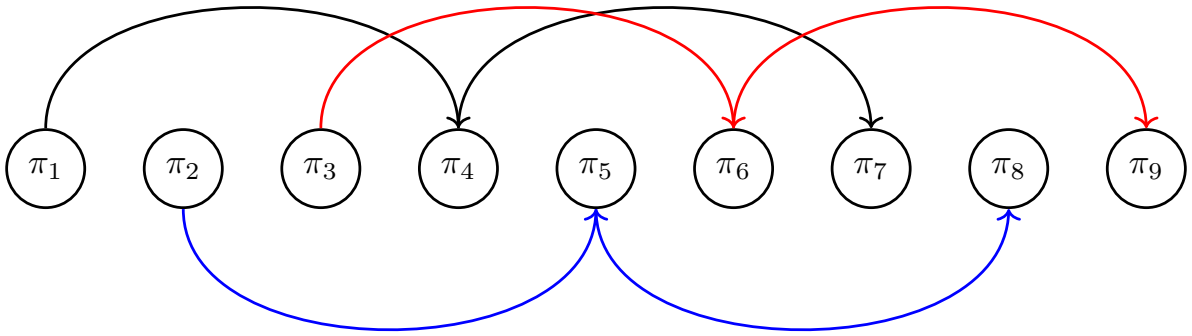

FIGURE 5.1 Sauts dans les chaînes d'inégalités valides

Cela veut dire que les valeurs réelles des deux variables impliquées dans une inégalité seront plus éloignées, ce qui nous permet de créer des inégalités moins serrées. Il avait été supposé par Popovic (2023) que les inégalités générées par sa méthode étaient "assez serrées", mais aucune réelle mesure de la proximité entre les variables impliquées n'avait été faite. Nous introduisons donc cette variation sur la longue chaîne d'inégalités pour mesurer l'impact de l'écart moyen entre les variables impliquées dans les inégalités sur l'accélération. L'objectif est également de trouver, s'il en existe une, la valeur d'écart à partir de laquelle les inégalités ne sont plus assez serrées pour stabiliser la génération de colonnes. Le tableau 5.7 montre le nombre d'inégalités moyen, la longueur moyenne des chaînes, l'écart moyen entre les variables impliquées dans les inégalités et le facteur d'accélération pour différentes variations de chaînes d'inégalités valides. Notons que pour cette analyse, nous avons considéré seulement les 12 instances les plus dégénérées, donc les instances (G L).

Tableau 5.7 Analyse des chaînes d'inégalités valides pour les instances grandes et lentes

Nb inég.	Long. moy. chaînes	Écart moy.	Acc. tot.
1 chaîne complète			
975.32	976.32	0.98	3.08
10 chaînes			
964.48	90.99	0.98	2.96
100 chaînes			
896.36	9.22	1.05	2.75
400 chaînes			
501.09	2.02	1.15	1.96
Sauts de 100			
874.38	9.67	67.05	1.58
Sauts de 500			
473.82	2.09	144.88	1.009

Plusieurs conclusions intéressantes peuvent être tirées de ce tableau. Si on compare les cas avec une chaîne complète, 10 chaînes et 100 chaînes, on voit que même si l'on diminue drastiquement la longueur moyenne des chaînes d'inégalités, et si le nombre d'inégalités et l'écart moyen ne sont pas grandement affectés, on maintient de très bons facteurs d'accélération. À l'inverse, si on compare les instances avec 100 chaînes et les instances avec des sauts de 100, on remarque que pour un nombre d'inégalités et des longueurs moyennes des chaînes semblables, mais un écart moyen plus grand, on perd une grande partie de l'accélération. La même remarque peut être faite si l'on compare le cas avec 400 chaînes et celui avec des sauts de 500. Dans ce dernier cas, on perd toute accélération. Ces résultats montrent donc que même si 100% des inégalités ajoutées au problème sont respectées par la solution duale optimale connue, cela ne nous assure pas d'accélérer la génération de colonnes. En effet, il est aussi nécessaire que ces inégalités soient suffisamment serrées pour contraindre les oscillations des inégalités duales et stabiliser le problème.

La figure 5.2 montre les valeurs des facteurs d'accélération en fonction de l'écart moyen pour les instances contenant des chaînes d'inégalités valides avec des sauts de 100, 200, 300, 400 et 500 variables. Nous avons fait une régression sur les quatre premiers points pour voir comment la relation entre le facteur d'accélération et l'écart moyen évolue pour des écarts moyens autour de 100. Cette figure peut nous aider à établir une estimation de l'écart moyen nécessaire pour que les chaînes d'inégalités stabilisent la génération de colonnes de façon significative. On rappelle qu'on considère que sur l'ensemble d'instances sur lesquelles nous

effectuons nos tests, une accélération inférieure à 20% est négligeable, car elle peut tomber dans la marge d'erreur expérimentale. La régression de la figure 5.2 croise l'axe Acc tot= 1.2 lorsque l'écart moyen est environ égal à 101, ce qui démontre que, même lorsque 100% des inégalités ajoutées au problème sont satisfaites par la solution duale optimale connue, on doit s'assurer que l'écart moyen entre la valeur des variables impliquées dans les inégalités est inférieur à 101 si l'on souhaite obtenir une accélération significative de la génération de colonnes. Il est donc raisonnable de penser que dans le cas où l'on prédit des inégalités et où certaines inégalités ne sont pas satisfaites par la solution duale optimale connue, un écart moyen semblable ou plus bas serait nécessaire pour obtenir une accélération significative.

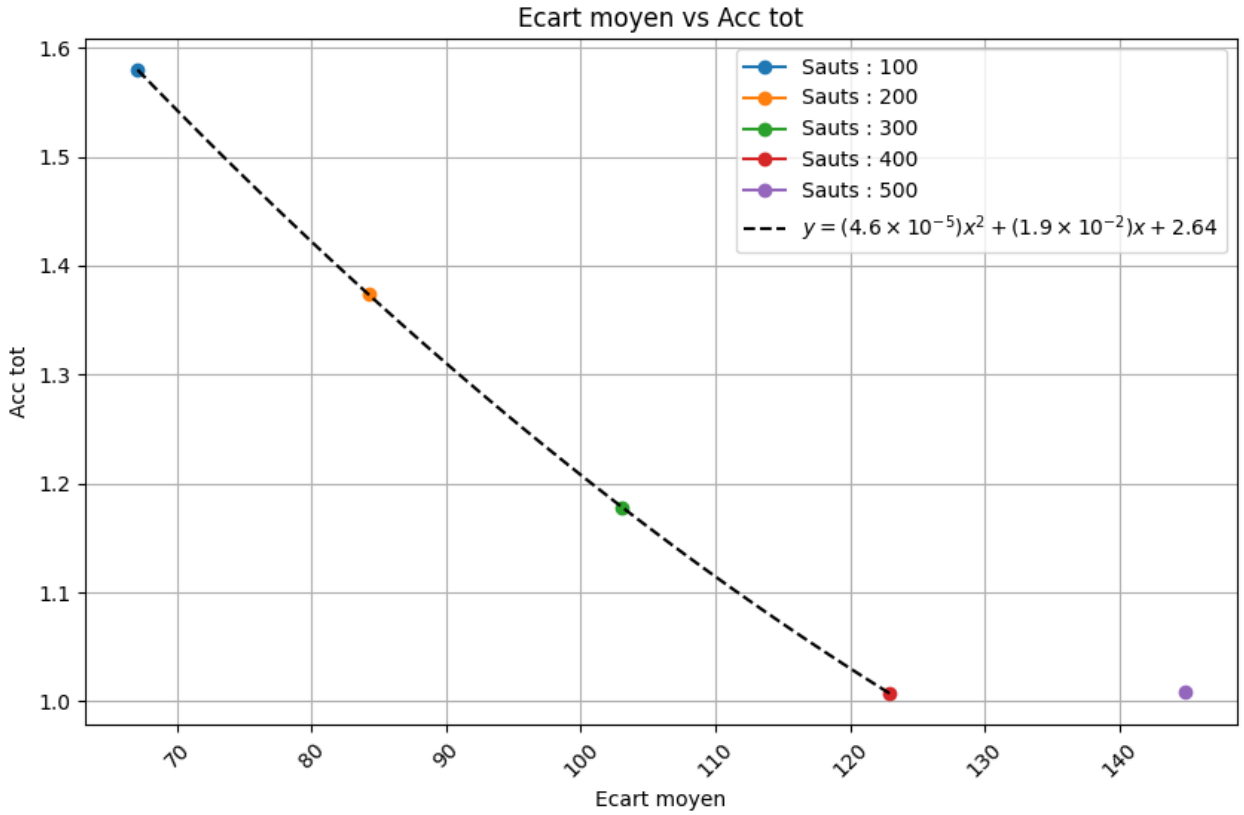

FIGURE 5.2 Facteur d'accélération en fonction de l'écart moyen pour les instances avec saut dans la chaîne entre 100 et 500 variables

Nous avons donc rassemblé toutes les instances avec prédiction d'inégalités que nous avons présentées plus tôt dans le tableau 5.8 et nous y présentons les écarts moyens et les facteurs d'accélération pour les instances (G L). Avec $\beta = 0.15$, on voit que l'écart moyen est beaucoup plus faible, mais que la trop grande proportion d'inégalités fausses cause d'autres problèmes. Nous parlerons davantage des problèmes causés par les inégalités fausses un peu plus loin. On peut également y voir que pour l'ensemble des instances étudiées avec $\beta = 0.01$, l'écart moyen

est supérieur à la valeur maximale de 101 que nous avons estimée comme nécessaire pour obtenir une accélération. Ce résultat nous laisse croire que les inégalités ajoutées aux problèmes par notre méthode de prédiction ne sont pas assez serrées pour stabiliser et accélérer la génération de colonnes. C'est probablement ce qui explique qu'aucune des configurations testées ne crée une accélération sur la résolution complète du problème, malgré le potentiel que cette méthode présentait.

Tableau 5.8 Facteur d'accélération et écart moyen pour les différentes instances étudiées

Instances	**Écart moy.**	**Acc. tot.**	**% inég. valides**
900 inég., $\beta = 0.15$	38.2	0.36	88.4
900 inég., $\beta = 0.01$	127.3	0.96	98.0
600 inég., $\beta = 0.01$	112.1	1.01	97.9
1200 inég., $\beta = 0.01$	128.9	0.84	98.0
900 inég., $\beta = 0.01$, répa. classique	121.85	0.70	98.0
900 inég., $\beta = 0.01$, répa. inversion	122.12	0.66	98.0

Pour vérifier cette conclusion, on peut prendre l'écart entre les variables dans les inégalités prédites et calculer quel pourcentage de celles-ci satisfait l'écart moyen maximum. C'est ce que nous présentons dans le tableau 5.9. On présente les pourcentages d'inégalités ayant un écart moyen inférieur à 101 à deux moments dans le processus. Premièrement, on vérifie cette valeur dans les prédictions en sortie du modèle d'apprentissage machine (prédictions). Pour calculer ce pourcentage, on sélectionne l'ensemble des inégalités où la probabilité $\hat{y}_{ij}$ prédite par le modèle d'apprentissage machine est plus petite que $1-\beta$ ou plus grande que β, comme on le fait lorsque l'on construit le graphe $\mathcal{GI}$. Notons qu'ici, on ne retire pas les cycles et on ne fait pas de tri selon les degrés des nœuds dans le graphe. Pour chaque inégalité, on calcule l'écart entre les deux variables en se fiant sur les valeurs qu'elles prennent dans la solution duale optimale connue. On peut ensuite simplement obtenir le pourcentage de ces inégalités qui ont un écart inférieur à 101. Ensuite, la colonne "Sélection" présente ces mêmes pourcentages, mais parmi les inégalités qui ont été insérées dans le problème. On calcule l'écart pour chacune des inégalités duales sélectionnées et on obtient le pourcentage de celles-ci qui sont inférieures à 101. Dans cette colonne, les pourcentages sont présentés pour les instances avec un maximum de 900 inégalités duales.

TABLEAU 5.9 Pourcentage d'inégalités sous le seuil maximum d'écart moyen parmi les inégalités prédites pour différentes valeurs de β

	Inégalités sous le seuil d'écart moyen (%)	
Seuil d'acceptation (β)	Prédictions	Sélection
0.01	78.55	86.74
0.03	80.60	90.12
0.05	81.60	91.76
0.1	83.06	94.40
0.15	83.90	95.85

Contrairement à ce que nous avions avancé, les résultats présentés au tableau 5.9 montrent que la grande majorité des inégalités duales ajoutées aux problèmes devraient être suffisamment serrées pour stabiliser la génération de colonnes. Le manque d'accélération n'est donc pas uniquement causé par le fait que les inégalités ne sont pas assez serrées. L'écart moyen pour ces instances reste élevé, mais cette valeur élevée est probablement causée par quelques très grandes valeurs qui font augmenter la moyenne.

Il est difficile d'expliquer avec certitude pourquoi la méthode proposée dans ce mémoire n'est pas efficace. Certes, nous avons montré que l'écart moyen entre les variables duales impliquées dans les inégalités n'est peut-être pas suffisamment petit pour générer une accélération, mais les derniers résultats montrent que la majorité des inégalités sont assez serrées. L'hypothèse que nous voulons avancer est qu'il est possible que le faible pourcentage d'inégalités fausses ajoutées au problème soit partiellement responsable de l'échec de cette méthode. À l'aide des chaînes d'inégalités valides, nous avons démontré que lorsque 100% des inégalités ajoutées au problème sont valides, l'écart moyen est déterminant dans leur capacité à accélérer la résolution. Par contre, même si nous avons montré que la majorité des inégalités prédites et sélectionnées devraient être assez serrées, nous n'obtenons pas les accélérations attendues. Un autre résultat qui pointe dans cette direction est que l'on obtient systématiquement de meilleures accélérations au noeud 0 que lors de la résolution complète du problème. Il serait raisonnable de penser que l'accélération serait semblable tout au long de la résolution, mais ce n'est pas le cas. Une chose qu'il est important de noter est que lorsque l'on fixe des colonnes dans les décisions de branchement, on retire les inégalités duales associées du PM^{ID}. En effet, lorsque l'on fixe une variable x_s, on assure que tous les trajets t tels que $a_s^t = 1$ inclus dans l'horaire s sont couverts. Les variables π_t associées aux contraintes de couverture (3.9b) sont alors fixées à zéro et on retire toutes les inégalités duales dans lesquelles ces variables sont impliquées. Le problème est que, si l'inégalité duale retirée est fausse, cela équivaut à

retirer l'inégalité comme on le fait dans le processus de réparation, à l'exception près que cette fois, l'une des deux variables est déjà fixée à zéro. Une réoptimisation est donc nécessaire, puisqu'on retire une inégalité qui nous empêchait de converger vers la solution duale optimale. Ce qui semble arriver, c'est donc que la solution vers laquelle l'algorithme tente de converger va constamment changer au fil de la résolution, forçant à chaque fois une réoptimisation, ce qui annule en quelque sorte l'accélération que l'on pourrait obtenir grâce à la stabilisation de la génération de colonnes.

Pour vérifier cette dernière hypothèse, nous avons appliqué notre processus de prédiction et de sélection des chaînes d'inégalités duales sur les instances (G L), puis nous avons retiré toutes les inégalités fausses en se basant sur la solution duale optimale connue. Les accélérations et les gaps obtenus de cette façon sont présentés au tableau 5.10.

Tableau 5.10 Accélération des instances (G L) pour le cas avec 900 inégalités duales et retrait des inégalités fausses

	Noeud 0		Résol. complète	
β	Acc.	Gap (%)	Acc.	Gap (%)
0.01	1.07	0.0	1.05	0.0
0.15	1.44	0.0	1.29	0.0

Ces résultats viennent confirmer notre hypothèse. Au tableau 5.8, nous avons montré qu'un β de 0.01 donnait un écart moyen de 127.3 alors qu'un β de 0.15 nous permet d'obtenir un écart moyen de 38.2. Si l'on retire toutes les inégalités fausses de ces deux groupes d'instances, on obtient une accélération négligeable pour $\beta = 0.01$, mais 30% d'accélération pour $\beta = 0.15$, ce qui confirme que le petit pourcentage d'inégalités fausses nous empêchait d'obtenir une accélération, mais également qu'une accélération ne peut être obtenue si l'écart moyen est trop grand.

5.6 Nouvelle approche : génération d'une longue chaîne d'inégalités au noeud 0

L'ensemble de ce chapitre jusqu'ici a servi à démontrer que la méthode de prédiction et de sélection d'inégalités duales par apprentissage machine que nous avons développée n'est pas efficace en raison des inégalités fausses ajoutées au problème et de l'écart moyen qui est trop grand entre les variables impliquées dans les inégalités duales. Cette section vient présenter une nouvelle approche développée à partir des nombreuses explorations que nous avons effectuées au niveau des chaînes d'inégalités duales et des conditions nécessaires à la

stabilisation de la génération de colonnes. Il s'agit d'une stratégie très simple à implémenter, mais qui ne semble pas avoir été explorée auparavant, du moins, pas à notre connaissance.

L'approche proposée met de côté le modèle d'apprentissage machine et utilise plutôt la structure de la solution duale de la première relaxation linéaire du problème pour sélectionner des inégalités duales. Comme nous l'avons montré à la section 5.5, l'utilisation de chaînes d'inégalités valides permet d'obtenir de très bons facteurs d'accélération. Nous avons, entre autres, obtenu une accélération moyenne de 308% sur les instances les plus dégénérées lorsque nous créons une longue chaîne d'inégalités valides reliant toutes les variables duales. C'est cette idée que l'on veut exploiter ici. Ce que l'on propose, c'est de résoudre d'abord la première relaxation linéaire du problème et d'en extraire la solution duale. Dans cette solution duale, on peut classer les valeurs de toutes les variables duales $\pi_t, t \in \mathbf{T}$, en ordre croissant, ce qui nous permet de créer une longue chaîne d'inégalités duales à l'image des tests effectués précédemment et de l'ajouter au PM après le nœud 0. On poursuit ensuite la résolution avec ces inégalités duales en surplus. Les facteurs d'accélération obtenus grâce à cette méthode de résolution sont présentés dans le tableau 5.11.

Tableau 5.11 Accélérations et gaps avec la solution obtenue sans inégalités duales, génération d'une longue chaîne d'inégalités après le noeud 0

	Noeud 0			Résol. complète		
Taille	Acc. L	Acc. R	Gap (%)	Acc. L	Acc. R	Gap (%)
M	1.00	1.00	0.0	1.32	1.34	0.0
G	1.00	1.00	0.0	1.27	1.24	0.0

En moyenne, cette méthode nous permet d'obtenir une accélération de 30% sur l'ensemble des instances étudiées en ne créant essentiellement aucune perte au niveau de la qualité de la solution. Par contre, cette méthode a aussi des faiblesses, puisqu'elle ne nous permet pas d'accélérer la résolution de la première relaxation linéaire du problème, qui correspond souvent à une grande portion des temps de calcul pour les instances dégénérées. C'est d'ailleurs pour cette raison que cette approche donne de meilleurs facteurs d'accélération sur les instances M. La première relaxation linéaire correspond simplement à une moins grande portion des temps de calcul sur ces instances.

Pour pallier cette faiblesse, nous avons tenté d'appliquer cette approche aux instances utilisant des inégalités duales prédites qui, elles, présentaient de petites accélérations au nœud 0. On prédit et on sélectionne les inégalités duales avant de débuter la résolution, puis on génère une longue chaîne après le nœud 0 en suivant la démarche décrite plus tôt. On peut

ensuite retirer toutes les inégalités duales qui étaient dans la formulation initiale du problème, puisqu'elles sont redondantes avec les nouvelles inégalités qui sont plus serrées. Les résultats obtenus sont présentés au tableau 5.12

Tableau 5.12 Accélérations et gaps avec la solution obtenue sans inégalités duales, 900 inégalités et génération d'une longue chaîne d'inégalités après le noeud 0

	Noeud 0			Résol. complète		
Taille	Acc. L	Acc. R	Gap (%)	Acc. L	Acc. R	Gap (%)
M	1.15	0.92	0.003	1.12	1.09	-0.0001
G	1.20	0.96	0.02	1.1	0.90	-0.003

Comme prévu, l'ajout d'inégalités duales prédites permet d'obtenir une faible accélération sur la première relaxation linéaire du problème, mais cette accélération ne se transpose pas sur la résolution complète du problème après avoir généré la longue chaîne d'inégalités. Ces résultats sont encore une fois en accord avec l'hypothèse que nous avons émise plus tôt. Comme le PM^{ID} initial contient un petit pourcentage d'inégalités fausses, la longue chaîne d'inégalités duales qui sera générée en contiendra aussi. Or, les inégalités fausses semblent forcer des réoptimisations lorsqu'elles sont retirées dans les décisions de branchement et ralentissent la résolution du problème. Selon nous, c'est ce qui explique que cette approche performe moins bien.

Nous avons déjà mentionné que dans l'industrie, les temps de résolution sont souvent priorisés par rapport à la qualité de la solution. Dans cette optique, une stratégie souvent utilisée est d'arrêter la résolution lorsque la valeur de la fonction objectif ne diminue plus assez rapidement entre les itérations de génération de colonnes. On dit alors que l'on "coupe les queues". Dans le tableau 5.13, on présente les facteurs d'accélération obtenus en appliquant cette stratégie en plus de générer une longue chaîne d'inégalités duales après le nœud 0. Ici, on compare les résultats obtenus à ceux obtenus en résolvant sans inégalités duales mais en coupant les queues.

TABLEAU 5.13 Accélérations et gaps avec la solution obtenue sans inégalités duales, génération d'une longue chaîne d'inégalités après le noeud 0 et queues coupées

	PM		PM^{ID}					
	Noeud 0	Résol. complète	Noeud 0			Résol. complète		
Taille	Temps (s)	Temps (s)	Acc. L	Acc. R	Gap (%)	Acc. L	Acc. R	Gap (%)
M	140	546	1.00	1.00	0.0	1.59	1.20	-0.02
G	294	1119	1.00	1.00	0.0	1.22	1.05	-0.21

On voit que, même dans le cas où on coupe les queues, notre approche permet d'obtenir des accélérations avoisinant 59% et 22% respectivement, pour les instances (M L) et (G L). Ce résultat prouve que la génération d'une chaîne d'inégalités duales au nœud 0 peut être applicable dans un contexte industriel où on coupe les queues pour accélérer la résolution. Évidemment, cela cause de légères pertes au niveau de la qualité de la solution, mais c'est un résultat qui est attendu dans ce contexte.

En résumé, nous avons montré que la prédiction d'inégalités duales est très difficile à appliquer dans le contexte du MDEVSP, car elle nécessite à la fois un modèle d'apprentissage machine qui prédit le sens des inégalités avec une précision presque parfaite et une méthode de sélection qui choisit les inégalités duales les plus serrées possibles. Un modèle d'apprentissage machine comme celui développé par Popovic (2023) fera toujours un certain pourcentage d'erreur face à des problèmes complexes et dégénérés comme le MDEVSP. De plus, comme nous n'avons aucune information sur la solution duale optimale lorsque l'on sélectionne les inégalités, il est impossible de savoir si celles choisies seront assez serrées. Cependant, la nouvelle approche proposée où l'on génère une chaîne d'inégalités duales à partir de la solution duale de la première relaxation linéaire montre des résultats encourageants. Elle montre que l'on peut utiliser la solution duale pour être certain de créer des inégalités suffisamment serrées et valides. Bref, bien que l'idée principale présentée dans ce mémoire ne soit pas aussi efficace que nous l'attendions, elle nous a permis d'introduire une stratégie plus simple, applicable dans l'industrie et accélérant d'environ 30% les instances dégénérées du MDEVSP.

CHAPITRE 6 CONCLUSION

Nous pouvons maintenant conclure ce mémoire en résumant l'ensemble des travaux, en discutant des limitations et faiblesses des approches proposées, puis en proposant quelques pistes d'amélioration à explorer pour la suite.

6.1 Synthèse des travaux et limitations

Dans ce mémoire, nous nous sommes intéressés au MDEVSP et à sa résolution par une heuristique de génération de colonnes. Nous avons montré que, dans d'autres contextes, l'utilisation d'inégalités duales pour stabiliser et accélérer la génération de colonnes a fait ses preuves et nous avons détaillé les contributions de Popovic (2023), montrant que la stratégie de prédiction de chaînes d'inégalités duales par apprentissage machine montrait un grand potentiel lorsque appliquée sur la première relaxation linéaire du problème.

Nous avons ensuite formulé le MDEVSP comme un programme linéaire en nombres entiers avec des contraintes de partitionnement d'ensemble. Nous avons présenté l'algorithme de génération de colonnes utilisé et formulé le sous-problème à l'aide d'un réseau de connexions où les arcs représentent les connexions valides et les nœuds représentent les différents trajets, les périodes d'attente et les périodes de recharge. Nous avons ensuite présenté la stratégie de branchement heuristique choisie, puis discuté de la dégénérescence et de son impact sur les temps de calcul.

Après avoir posé toutes ces notions importantes, nous sommes entrés davantage dans les contributions que nos travaux viennent apporter. Nous avons détaillé la stratégie développée par Popovic (2023) que nous utilisons pour prédire et sélectionner des chaînes d'inégalités duales. Pour résumer, nous utilisons un modèle d'apprentissage machine pour prédire la probabilité qu'une variable duale soit plus grande qu'une autre, pour chaque paire de variables duales. Nous utilisons ensuite ces probabilités pour générer des chaînes d'inégalités contenant le plus de variables duales possibles, ce qui nous permet de stabiliser la génération de colonnes en contraignant les oscillations dans la valeur des variables duales. Nous avons ensuite présenté deux variantes d'un processus de réparation, permettant d'améliorer la qualité de la solution dans le cas où le modèle d'apprentissage machine ferait des erreurs et introduirait de fausses inégalités.

Une fois tous les outils en mains, nous avons présenté les résultats obtenus. Nous avons montré que, grâce à des ordinateurs plus performants et une modification au niveau des

algorithmes de résolution utilisés, nous obtenons des temps de calcul beaucoup plus petits que ce qui était présenté par Popovic (2023) pour les mêmes instances. Nous avons également présenté quelques tests que nous avons effectués, montrant, entre autres, qu'en utilisant un seuil d'acceptation β plus faible que celui utilisé dans les travaux de Popovic, il est possible de réduire le nombre de fausses inégalités ajoutées au problème et d'atteindre un pourcentage d'inégalités valides dans les prédictions supérieur à 99%. Nous avons donc utilisé une valeur de $\beta = 0.01$ pour l'ensemble de nos tests. La suite a été une présentation des facteurs d'accélération et des gaps obtenus pour différentes configurations. Nous avons testé les deux processus de réparation et fait varier le nombre maximum d'inégalités duales entre 600 et 1200, ce qui nous a permis de montrer que la stratégie proposée n'était pas efficace.

Nous avons ensuite effectué une analyse des chaînes d'inégalités valides et des conditions nécessaires à la stabilisation de la génération de colonnes à l'aide de celles-ci. Cela nous a permis de mettre en lumière qu'il est important que les inégalités ajoutées au problème soient suffisamment serrées. Nous avons estimé que dans le cas où 100% des inégalités sont valides, l'écart moyen maximum entre les variables impliquées dans les inégalités duales pour obtenir une accélération significative se situe autour de 101. Des tests supplémentaires nous ont permis de montrer que l'écart moyen dans les instances avec prédiction d'inégalités duales était trop grand, mais que ce n'était pas la seule cause de l'échec de la méthode proposée. En effet, nous avons également prouvé que, même si un très petit pourcentage des inégalités ajoutées au problème sont fausses, cela ralentit grandement la résolution lorsque l'on utilise un branchement heuristique avec fixation de colonnes. Nous avons donc pu conclure que pour obtenir une accélération significative grâce à des chaînes d'inégalités duales appliquées au MDEVSP, les inégalités doivent être presque toutes satisfaites par la solution duale optimale du problème et elles doivent être suffisamment serrées. Considérant cela, il n'est pas surprenant que l'utilisation d'inégalités duales prédites par apprentissage machine ne donne pas de très bonnes performances. Un modèle comme celui-ci présentera toujours un risque d'erreurs et, comme nous n'avons pas d'informations sur la solution duale optimale lorsque l'on sélectionne les inégalités, il est impossible de favoriser les inégalités qui sont serrées. En somme, la nature même du problème rend difficile l'utilisation efficace de l'apprentissage machine pour générer des inégalités à la fois valides et serrées, ce qui limite fortement le potentiel de cette approche dans le contexte du MDEVSP.

Nous avons terminé en proposant une approche alternative, mettant de côté l'apprentissage machine. Cette nouvelle stratégie crée une longue chaîne d'inégalités duales à partir de la solution duale de la première relaxation linéaire du problème et l'utilise pour accélérer la suite de la résolution. On obtient ainsi des accélérations d'environ 30% sur l'ensemble des instances étudiées tout en n'ayant essentiellement aucune perte au niveau de la qualité de la

solution. Cette méthode se présente comme une alternative plus facile à implémenter que la prédiction d'inégalités duales par apprentissage machine et permet d'obtenir de meilleures accélérations.

Par contre, cette approche a aussi ses limitations. Comme les inégalités sont ajoutées après la résolution de la première relaxation linéaire et que cette étape correspond souvent à une fraction importante des temps de calcul, le potentiel d'accélération de cette méthode est relativement limité. Dans un cas où la résolution de la première relaxation linéaire serait beaucoup plus longue, l'accélération obtenue grâce à la génération d'une longue chaîne d'inégalités après le nœud 0 pourrait être négligeable.

6.2 Améliorations futures

Malgré les résultats décevants, nous pensons toujours que la prédiction d'inégalités duales par apprentissage machine présente un certain potentiel pour l'accélération de la génération de colonnes. Si l'on arrivait à développer un modèle capable d'estimer la valeur des variables duales dans la solution duale optimale avec une bonne précision, il serait possible de favoriser la sélection d'inégalités plus serrées et d'obtenir des écarts moyens plus petits.

Il serait également intéressant de tester une stratégie de branchement heuristique qui ne retire pas les inégalités duales lorsque des variables sont fixées. De cette façon, on éviterait les nombreuses réoptimisations qui ralentissent grandement la résolution. Un tel ajustement pourrait rendre utile la prédiction de chaînes d'inégalités duales et l'accélération observée au nœud 0 pourrait se poursuivre sur la suite de la résolution.

Enfin, il serait pertinent d'essayer d'appliquer l'approche alternative proposée sur d'autres types de problèmes. La génération d'une longue chaîne d'inégalités après le nœud 0 est une stratégie simple à implémenter et nous avons montré qu'elle présente des accélérations significatives lorsqu'elle est appliquée sur les variables duales associées à des contraintes de partitionnement d'ensemble. Il est donc raisonnable de penser qu'elle pourrait être appliquée à n'importe quel problème utilisant ce type de contraintes et présenter des accélérations similaires. Il pourrait même être intéressant d'apparier cette stratégie avec des idées similaires à celles développées par Mhamedi *et al.* (2022) et Gschwind et Irnich (2016). Cela nous permettrait d'utiliser la structure du problème pour générer quelques inégalités duales que nous savons valides, puis d'améliorer davantage la stabilisation en générant la longue chaîne d'inégalités après le nœud 0. Bref, même si cette stratégie présente des accélérations significatives lorsqu'elle est implémentée seule, nous pensons qu'elle aurait un potentiel encore plus grand si elle complétait une autre heuristique.

Pour terminer, nous estimons que nous avons su montrer le potentiel de nos approches en exposant leurs forces et en expliquant leurs faiblesses. Maintenant, nous pensons que tous les outils sont en place et que, à l'aide des ajustements proposés, la stratégie présentée dans ce mémoire pourrait permettre d'accélérer significativement la résolution du MDEVSP par génération de colonnes et ainsi ouvrir la porte à davantage d'applications de ce genre dans l'industrie.

RÉFÉRENCES

K. An : Battery electric bus infrastructure planning under demand uncertainty. *Transportation Research Part C : Emerging Technologies*, 111:572–587, 2020.

C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh et P.H. Vance : Branch-and-price : Column generation for solving huge integer programs. *Operations Research*, 46(3):316–29, 1998.

R. Bellman : On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

H. Ben Amor et J. Desrosiers : A proximal trust-region algorithm for column generation stabilization. *Computers & Operations Research*, 33(4):910–927, 2006.

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

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

T. Berthold : Primal heuristics for mixed integer programs. Mémoire de maîtrise, Technische Universität Berlin, 2006.

A. Bertossi, P Carraresi et G Gallo : On some matching problems arising in vehicle scheduling models. *Networks*, 17(3):271–281, 1987.

H. Bouarab, I. El Hallaoui, A. Metrane et F. Soumis : Dynamic constraint and variable aggregation in column generation. *European Journal of Operational Research*, 262 (3):835–50, 2017.

N. Brinkel, M. Zijlstra, R. Bezu, T. Twuijver et I. Lampropoulos : A comparative analysis of charging strategies for battery electric buses in wholesale electricity and ancillary services markets . *Transportation Research Part E Logistics and Transportation Review*, 172:103085, 2023.

Cabinet de la vice-première ministre et ministre des Transports et de la Mobilité durable : Le gouvernement du Canada et le gouvernement du Québec concrétisent le plus important projet d'acquisition d'autobus électriques en Amérique du Nord, 2023. URL `https://www.quebec.ca/nouvelles/actualites/details/le-gouvernement-du-canada-et-le-gouvernement-du-quebec-concretisent-le-plus-important-projet-dacquisition-dautobus-electriques-en-amerique-du-nord-47644`.

G. Carpaneto, M. Dell'amico, M. Fischetti et P. Toth : A branch and bound algorithm for the multiple depot vehicle scheduling problem. *Networks*, 19(5):531–548, 1989.

A. Charnes : Optimality and degeneracy in linear programming. *Econometrica*, 20(2):160–170, 1952.

É. Chouinard : 13 G pour électrifier le transport collectif : Québec fait fausse route, dit une experte. Radio-Canada, 2024. URL `https://ici.radio-canada.ca/nouvelle/2051399/autobus-electrique-investissement-transport-collectif`.

G. Desaulniers, J. Desrosiers et M. Solomon : *Column generation*, volume 5. Springer Science & Business Media, New York, 2005.

G. Desaulniers et M. Hickman : Public transit. *In* C. Barnhart et G. Laporte, éditeurs : *Transportation*, volume 14 de *Handbooks in Operations Research and Management Science*, pages 69–127. Elsevier, 2007.

G. Desaulniers, J. Lavigne et F. Soumis : Multi-depot vehicle scheduling problems with time windows and waiting costs. *European Journal of Operational Research*, 111(3):479–494, 1998.

J. Desrosiers : Gencol : Une équipe et un logiciel d'optimisation. Les Cahiers du GERAD G-2010-06, Groupe d'études et de recherche en analyse des décisions, Montréal, 2010. URL `https://www.gerad.ca/fr/papers/G-2010-06`.

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

O. du Merle, D. Villeneuve, J. Desrosiers et P. Hansen : Stabilized column generation. *Discrete Mathematics*, 194(1-3):229–237, 1999.

I. Elhallaoui, G. Desaulniers, A. Metrane et F. Soumis : Bi-dynamic constraint aggregation and subproblem reduction. *Computers and Operations Research*, 35(5):1713–24, 2008.

I. Elhallaoui, A. Metrane, F. Soumis et G. Desaulniers : Multi-phase dynamic constraint aggregation for set partitioning type problems. *Mathematical Programming*, 123(2):345–370, 2010.

I. Elhallaoui, D. Villeneuve, F. Soumis et G. Desaulniers : Dynamic aggregation of set-partitioning constraints in column generation. *Operations Research*, 53(4):632–45, 2005.

L. R. Ford et D. R. Fulkerson : *Flows in Networks*. Princeton University Press, 1962.

J. Gerbaux, G. Desaulniers et Q. Cappart : A machine-learning-based column generation heuristic for electric bus scheduling. *Computers & Operations Research*, 173:106848, 2025.

K. Gkiotsalitis, O. Cats, T. Liu et J. M. Bult : An exact optimization method for coordinating the arrival times of urban rail lines at a common corridor. *Transportation Research Part E : Logistics and Transportation Review*, 178:103265, 2023.

Gouvernement du Québec : Plan pour une économie verte 2030, 2020. URL `https://cdn-contenu.quebec.ca/cdn-contenu/adm/min/environnement/publications-adm/plan-economie-verte/plan-economie-verte-2030.pdf`.

T. Gschwind et S. Irnich : Dual Inequalities for Stabilized Column Generation Revisited. *INFORMS Journal on Computing*, 28(1):175–194, 2016.

A. Hadjar, O. Marcotte et F. Soumis : A branch-and-cut algorithm for the multiple depot vehicle scheduling problem. *Operations Research*, 54(1):130–149, 2006.

H. Hoimoja, A. Rufer, G. Dziechciaruk et A. Vezzini : An ultrafast EV charging station demonstrator. *In 2012 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM 2012), 20-22 June 2012*, 2012 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM 2012), pages 1390–5, Piscataway, NJ, USA, 2012. IEEE.

T. Ibaraki, S. Imahori, M. Kubo, T. Masuda, T. Uno et M. Yagiura : Effective local search algorithms for routing and scheduling problems with general time-window constraints. *Transportation Science*, 39(2):206–232, 2005.

O. J. Ibarra-Rojas, F. Delgado, R. Giesen et J. C. Muñoz : Planning, operation, and control of bus transport systems : A literature review. *Transportation Research Part B : Methodological*, 77:38–75, 2015.

S. Irnich et G. Desaulniers : Shortest path problems with resource constraints. *In* G. Desaulniers, J. Desrosiers et M. Solomon, éditeurs : *Column Generation*, pages 33–65. Springer US, Boston, MA, 2005.

C. Joncour, S. Michel, R. Sadykov, D. Sverdlov et F. Vanderbeck : Column Generation based Primal Heuristics. *Electronic Notes in Discrete Mathematics*, 36:695–702, 2010.

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

J. Li : Transit Bus Scheduling with Limited Energy. *Transportation Science*, 48(4):521–539, 2014.

L. Li, H. Lo, F. Xiao et X. Cen : Mixed bus fleet management strategy for minimizing overall and emissions external costs. *Transportation Research Part D : Transport and Environment*, 60:104–118, 2018.

S. Lin : Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269, 1965.

T. Liu et A. Ceder : Deficit function related to public transport : 50 year retrospective, new developments, and prospects. *Transportation Research Part B : Methodological*, 100:1–19, 2017.

T. Liu et A. Ceder : Battery-electric transit vehicle scheduling with optimal number of stationary chargers. *Transportation Research, Part C : Emerging Technologies*, 114:118–39, 2020.

T. Mhamedi, H. Andersson, M. Cherkesly et G. Desaulniers : A Branch-Price-and-Cut Algorithm for the Two-Echelon Vehicle Routing Problem with Time Windows. *Transportation Science*, 56(1):245–264, 2022.

A. Montoya, C. Guéret, J. E. Mendoza et J. G. Villegas : The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B : Methodological*, 103:87–110, 2017.

T. Nishi, Y. Muroi et M. Inuiguchi : Column generation with dual inequalities for railway crew scheduling problems. *Public Transport*, 3(1):25–42, 2011.

A. Oukil, H. Ben Amor, J. Desrosiers et H. El Gueddari : Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems. *Computers and Operations Research*, 34(3):817–834, 2007.

S. Pelletier, O. Jabali, G. Laporte et M. Veneroni : Battery degradation and behaviour for electric vehicles : Review and numerical analyses of several models. *Transportation Research Part B : Methodological*, 103:158–187, 2017.

S. Perumal, R. M. Lusby et J. Larsen : Electric bus planning & scheduling : A review of related problems and methodologies. *European Journal of Operational Research*, 301 (2):395–413, 2022.

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

L. Popovic : Apprentissage d'inégalités duales pour la génération de colonnes appliquée au problème d'horaires d'autobus électriques avec dépôts multiples. mémoire de maîtrise, Dép. de génie informatique et génie logiciel, École Polytechnique de Montréal, Montréal, QC, 2023. URL `https://publications.polymtl.ca/54127/`.

J.-Y. Potvin, T. Kervahut, B.-L. Garcia et J.-M. Rousseau : The vehicle routing problem with time windows. I. Tabu search. *INFORMS Journal of Computing*, 8(2):158–64, 1996.

F. Quesnel, G. Desaulniers et F. Soumis : A new heuristic branching scheme for the crew pairing problem with base constraints. *Computers and Operations Research*, 80:159–172, 2017.

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

R. Sadykov, F. Vanderbeck, A. Pessoa, I. Tahiri et E. Uchoa : Primal Heuristics for Branch and Price : The Assets of Diving Methods. *INFORMS Journal on Computing*, 31 (2):251–267, 2019.

O. Sassi et A. Oulamara : Electric vehicle scheduling and optimal charging problem : Complexity, exact and heuristic approaches. *International Journal of Production Research*, 55(2):519–35, 2017.

E. Taillard, P. Badeau, M. Gendreau, F. Guertin et J.-Y. Potvin : A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31(2):170–86, 1997.

S. Vavasis et Y. Ye : "a simplification to “a primal-dual interior point method whose running time depends only on the constraint matrix”". *In "High Performance Optimization"*, pages 233–243. Springer US, Boston, MA, 2000.

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

C. Yang, W. Lou, J. Yao et S. Xie : On Charging Scheduling Optimization for a Wirelessly Charged Electric Bus System. *IEEE Transactions on Intelligent Transportation Systems*, 19(6):1814–1826, 2018.

J. Yarkony, N. Haghani et A. Regan : Detour dual optimal inequalities for column generation with application to routing and location. *ArXiv*, 2021. URL `https://arxiv.org/abs/2108.09233`.

S. Yildirim et B. Yildiz : Electric bus fleet composition and scheduling. *Transportation Research Part C : Emerging Technologies*, 129:103197, 2021.

L. Zhang, Y. Han, J. Peng et Y. Wang : Vehicle and Charging Scheduling of Electric Bus Fleets : A Comprehensive Review. *Journal of Intelligent and Connected Vehicles*, 6 (3):116–124, 2023.

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

L. Zhang, Z. Zeng et K. Gao : A bi-level optimization framework for charging station design problem considering heterogeneous charging modes. *Journal of Intelligent and Connected Vehicles*, 5(1):8–16, 2022.

Y. Zhou, Q. Meng et G. Ong : Electric Bus Charging Scheduling for a Single Public Transport Route Considering Nonlinear Charging Profile and Battery Degradation Effect. *Transportation Research Part B : Methodological*, 159:49–75, 2022.