Titre: Title:	Problème d'horaire d'autobus avec dépôts multiples et modification contrôlée des heures de début des voyages
Auteur: Author:	Lucie Desfontaines
Date:	2017
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Desfontaines, L. (2017). Problème d'horaire d'autobus avec dépôts multiples et modification contrôlée des heures de début des voyages [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/2622/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/2622/
Directeurs de recherche: Advisors:	Guy Desaulniers
Programme: Program:	Maîtrise recherche en mathématiques appliquées

UNIVERSITÉ DE MONTRÉAL

PROBLÈME D'HORAIRE D'AUTOBUS AVEC DÉPÔTS MULTIPLES ET MODIFICATION CONTRÔLÉE DES HEURES DE DÉBUT DES VOYAGES

LUCIE DESFONTAINES
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
JUILLET 2017

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

PROBLÈME D'HORAIRE D'AUTOBUS AVEC DÉPÔTS MULTIPLES ET MODIFICATION CONTRÔLÉE DES HEURES DE DÉBUT DES VOYAGES

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

M. SOUMIS François, Ph. D., président
M. DESAULNIERS Guy, Ph. D., membre et directeur de recherche
M. FLEURENT Charles, Ph. D., membre

REMERCIEMENTS

Je tiens à remercier, en premier lieu, mon directeur de recherche Guy Desaulniers d'avoir accepté d'encadrer ma maîtrise à Polytechnique. Sa grande disponibilité et ses conseils avisés ont été une aide déterminante pour mener ce projet à bien. Ce fut un réel plaisir de travailler avec lui. Je remercie également l'équipe de GIRO d'avoir permis ce partenariat et pour les discussions fructueuses que nous avons pu avoir.

Je voudrais de plus remercier François Soumis et Charles Fleurent d'avoir accepté de faire partie du jury de ma maîtrise.

Un grand merci également au personnel du GERAD pour leur support et leur sympathie. Je voudrais souligner l'aide que m'a apportée François Lessard sur la programmation de certains algorithmes et les explications de Benoît Rochefort grâce auxquelles j'ai pu travailler plus efficacement. Ces deux années au GERAD et à Polytechnique auraient été bien tristes sans tous les étudiants que j'ai pu rencontrer. Merci pour tous les bons moments partagés !

Il me tient à coeur, enfin, d'exprimer ma reconnaissance envers tous les professeurs de mathématiques que j'ai eu la chance d'avoir au cours de ma scolarité et qui m'ont donné le goût de continuer dans cette voie.

RÉSUMÉ

Dans le domaine des transports en commun, et en particulier des réseaux d'autobus, organiser efficacement et à moindres coûts le réseau est crucial. A partir d'une grille horaire donnant les heures de départ des trajets de chaque ligne, la compagnie d'autobus doit organiser sa flotte de véhicules et son personnel afin d'assurer le service de transport. Dans ce problème de planification opérationnelle, nous nous intéressons à la gestion des véhicules. Il s'agit de construire le parcours de chacun des véhicules afin de couvrir tous les trajets désirés, et cela au coût le plus faible possible. Dès lors que la compagnie entrepose ses véhicules dans des dépôts différents, ce problème est très difficile à résoudre (classe des problèmes NP-difficiles). On l'appelle problème d'horaire d'autobus avec dépôts multiples ou MDVSP pour l'anglais Multiple Depot Vehicle Scheduling Problem.

Dans ce mémoire, on ajoute au MDVSP la possibilité de modifier les heures de départ initialement prévues. Cela laisse espérer une réduction du nombre d'autobus nécessaires et des trajets à vide moins coûteux pour la compagnie d'autobus.

Pour modéliser ce problème, nous introduisons un graphe hybride entre les deux réseaux classiques de modélisation du MDVSP, à savoir le réseau de connexions et le réseau espace-temps. Le premier modélise explicitement par un arc chacune des connexions possibles entre deux trajets, le second représente chaque mouvement envisageable d'un autobus par un arc de déplacement dans le temps et éventuellement l'espace. Le réseau hybride introduit permet de conserver l'adaptabilité du premier type de réseau et le faible nombre d'arcs du second. En terme de temps de calcul, ce nouveau réseau donne de très bonnes performances pour résoudre le MDVSP classique et le MDVSP avec modification des horaires de départ.

Pour résoudre les modèles ainsi construits, on décompose le modèle en deux entités, l'une construit des itinéraires d'autobus admissibles, l'autre choisit une combinaison de ces itinéraires de coût minimal. On peut ainsi résoudre la relaxation linéaire de notre problème par génération de colonnes et donc le problème en nombres entiers par un algorithme de *Branch and Price*. On décrit et implémente dans ce travail plusieurs méthodes pour accélérer cet algorithme, ralenti entre autres par la dégénérescence des problèmes. Les résultats obtenus montrent qu'on peut ainsi trouver des solutions quasi-optimales en des temps de calcul fortement réduits.

Sur des exemples de 300 ou 500 voyages où l'on permet de modifier les horaires de plus ou moins 2 minutes, on voit qu'il est possible de réduire le nombre de véhicules nécessaires de 1 à 3 unités. On étudie également une instance réelle de plus de 1000 voyages que l'on résout en un peu plus d'une heure. Les réductions de coût pour la compagnie d'autobus peuvent donc être importantes.

Cependant, modifier les heures de départ ne doit pas détériorer la qualité des horaires pour les utilisateurs du réseau. On voudrait, en particulier, garder des espacements égaux entre les trajets consécutifs d'une même ligne. Un équilibre doit donc être trouvé entre réduction des coûts pour la compagnie et qualité des horaires pour les passagers. Pour cela, on propose d'adopter une approche en deux phases : la première résout le MDVSP avec modification d'horaires, la seconde optimise les horaires de départ compatibles avec les itinéraires trouvés dans la première phase. Pour coordonner ces deux étapes, on tire profit du réseau hybride introduit précédemment. Les résultats montrent qu'on peut ainsi trouver de bons compromis entre le coût des itinéraires d'autobus et la qualité des horaires pour les passagers.

ABSTRACT

In the field of public transit, and especially of bus systems, it is crucial to organize the network efficiently and at a low cost. Given a timetable for each line, a bus company should determine the schedule of all its vehicles and drivers to provide the required service. In this operational planning problem, we will focus on vehicle scheduling. The aim is to build vehicle itineraries in order to cover all desired trips, and this at the lowest possible cost. If the company stores its buses in several depots, this problem is hard to solve (NP-hard problem). It is called Multiple Depot Vehicle Scheduling Problem, abbreviated by MDVSP hereafter.

In this work, we add to the classical MDVSP the possibility of modifying the initial timetable, in other words, we allow trip shifting. This way, one can expect to reduce the required number of vehicles and to limit the cost of deadhead trips, i.e. trips without any passengers.

To model this problem, we introduce a hybrid graph mixing the two classical graphs used to model MDVSP, namely the connection network and the time-space network. The first one represents explicitly every possible connection between two compatible trips, the second one models every possible bus movement through an arc representing a move in time and possibly in space. Our hybrid network keeps the modeling flexibility of the first type of network and the small number of arcs of the second one. In terms of computation time, this new network performs very well when solving both the classical MDVSP and the MDVSP with trip shifting.

In order to solve this problem, we split the model into two entities, one builds feasible bus itineraries, the other chooses a combination of these itineraries at the lowest cost. The linear relaxation of our problem could then be solved using a column generation method and the integer problem thus using a Branch and Price algorithm. In this work, we describe and implement several ways to speed up this algorithm, which is otherwise slowed down by degeneracy (among other things). For our test instances, we manage to find solutions very close to optimality with a substantial reduction of computation time.

On instances with 300 or 500 trips, where schedule shifts of up to two minutes were allowed, the bus company can save up to 3 vehicles. We studied also a real-life problem involving more than 1000 trips that we solved in a bit more than one hour. Costs reduction can therefore be very significant.

However, shifting trips should not be detrimental to the timetable quality for passengers. In particular, we wish to keep constant headways, i.e. the time lapses between two consecutive departures on the same line. Hence, a tradeoff should be found between cost reduction for the bus company and timetable quality for passengers. To this end, we propose to adopt a two-step algorithm: the first step solves the MDVSP with trip shifting, the second one optimizes the departure schedules while taking into consideration the itineraries found in the first step. To coordinate these two phases, we take advantage of the previously introduced hybrid network. Computational results with this algorithm show that one can quickly find interesting tradeoffs between itineraries costs and timetable quality.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

MDVSP	Multiple Depot Vehicle Scheduling Problem
SDVSP	Single Depot Vehicle Scheduling Problem
hlp	haut-le-pied
PLNE	Problème Linéaire en Nombres Entiers

CHAPITRE 1 INTRODUCTION

1.1 Contexte : optimisation des transports publics

Garantir la fiabilité et l'efficacité des réseaux de transport public est un enjeu majeur pour les villes et régions du monde entier. Les transports en commun se doivent, en effet, de chercher à satisfaire au mieux les besoins des usagers, tout en garantissant la maîtrise des coûts liés au fonctionnement de réseaux qui s'agrandissent au fil des années. La recherche opérationnelle apporte des solutions d'aide à la décision pour conseiller et automatiser la planification de ces réseaux de transport. On s'intéresse, dans ce travail, plus particulièrement au cas des réseaux d'autobus. Les étapes de décision auxquelles doit répondre une compagnie d'autobus peuvent être résumées de la manière suivante :

1. La construction des lignes du réseau. La première étape consiste à dessiner les lignes du réseau d'autobus, c'est-à-dire établir le trajet de chacune d'entre elles et déterminer la position des arrêts. C'est une étape stratégique dans l'optimisation d'un réseau de transport.
2. La construction des horaires de départ. Cette étape consiste à décider le nombre de départs sur chaque ligne et l'heure de chacun de ces départs. L'horaire construit doit satisfaire au mieux les besoins (estimés) des utilisateurs du réseau d'autobus. Généralement, on souhaite que les voyages de chaque ligne partent à une fréquence régulière et qu'à certains points dans le réseau, les passagers puissent effectuer facilement une correspondance entre plusieurs autobus.
3. La construction des itinéraires de véhicule. Il s'agit ici pour la compagnie d'autobus de déterminer l'itinéraire de chacun de ses véhicules, afin que tous les trajets déterminés à l'étape précédente soient couverts. Dans la plupart des situations, les véhicules sont entreposés dans différents dépôts. On parle donc de problème d'horaire d'autobus avec plusieurs dépôts, que l'on abrégera souvent dans la suite en MDVSP, pour l'anglais Multiple Depot Vehicle Scheduling Problem.
4. La construction des horaires journaliers pour les chauffeurs d'autobus. Il faut en effet attribuer un chauffeur à chaque autobus en circulation, tout en respectant les contraintes sur la durée du travail (pause, nombre d'heures travaillées par jour,...)
5. La construction des horaires hebdomadaires et mensuels des chauffeurs. Cette étape doit répondre aux objectifs d'équité des emplois du temps entre les chauffeurs et de prise en compte des repos/vacances.

De nombreux algorithmes exacts et heuristiques ont été proposés pour résoudre ces différentes étapes d'optimisation. On réfère à Desaulniers et Hickman (2007), Guihaire et Hao (2008) (sur les deux premières étapes) et Ibarra-Rojas et al. (2015) pour des revues détaillées des méthodes existantes.

Les problèmes réels rencontrés par les compagnies d'autobus sont en général de très grande taille, avec plusieurs milliers de trajets d'autobus à planifier chaque jour. Ainsi, la résolution de chacune des étapes décrites précédemment conduit à un problème difficile. Généralement, les décisions de planification des réseaux sont donc réalisées en plusieurs problèmes successifs. Cela entraîne une perte d'optimalité par rapport au problème global, mais insoluble, où l'on déciderait à la fois des lignes, des horaires, du trajet des véhicules et de l'emploi du temps des chauffeurs.

Plusieurs articles ont récemment proposé des méthodes pour intégrer la résolution de deux (voire trois) étapes successives. Nous nous intéressons ici à l'agrégation partielle des étapes de détermination des horaires et de construction des itinéraires de véhicule. Ce travail a été réalisé en partenariat avec l'entreprise GIRO, qui développe des logiciels d'aide à la décision pour les compagnies de transport.

1.2 Objectifs de ce projet

On se place dans le cas d'un réseau d'autobus existant pour lequel les lignes et les horaires de départ ont déjà été déterminés. L'étape classique de construction des itinéraires d'autobus (MDVSP) prend ces horaires comme données d'entrée et cherche les itinéraires avec les coûts les plus faibles pour couvrir tous les voyages requis. Dans ce projet, on permet de remettre partiellement en question les décisions sur les horaires de départ. On souhaite donc répondre à la question suivante : est-il possible de diminuer, et si oui de combien, le coût des itinéraires d'autobus si on permet de modifier légèrement les horaires de départ initialement prévus ? En terme d'algorithme, le défi consiste donc à résoudre efficacement le MDVSP en ajoutant de la flexibilité sur les choix des horaires de départ.

On souhaite donc modifier légèrement la grille initiale des horaires, et non reconstruire une nouvelle grille horaire. C'est pourquoi, on parlera dans la suite de "décalage de voyage". Idéalement, on voudrait décaler le moins de trajets possible pour ne pas perturber inutilement la

grille horaire. De plus, la grille horaire initiale possède en général certaines caractéristiques que l'on souhaite conserver, dans la mesure du possible. La qualité des horaires réside, entre autres, dans des espacements égaux entre les voyages d'une même ligne. En outre, les horaires de départ sont habituellement choisis pour faciliter les correspondances pour les passagers entre différents autobus ou entre un autobus et un autre moyen de transport.

Nous voulons donc développer une méthode qui permette de résoudre le MDVSP avec décalage possible des horaires qui assure que la qualité de la grille horaire reste bonne. C'est pourquoi, on parle de "modification contrôlée" des horaires de départ.

1.3 Plan du mémoire

On détaille, dans un premier temps, les travaux existants effectués sur le MDVSP classique et le MDVSP avec décalage d'horaires (chapitre 2). Notre algorithme de résolution est ensuite décrit aux chapitres 3 et 4. Dans le premier, on détaille et compare différents réseaux pour modéliser le problème de construction d'itinéraires de véhicules, dans les cas avec et sans décalage d'horaires. Puis, au chapitre 4, nous utilisons ces réseaux comme base d'un algorithme de génération de colonnes pour résoudre notre problème. On propose une version exacte et une version heuristique de l'algorithme. Enfin, au chapitre 5, nous nous intéressons au contrôle de la qualité des horaires, et proposons un algorithme en deux phases pour résoudre le MDVSP avec décalage contrôlé des horaires. Une conclusion du mémoire et quelques idées de prolongement sont enfin données au chapitre 6.

CHAPITRE 2 REVUE DE LITTÉRATURE

Cette section est structurée en trois parties : on détaille d'abord les méthodes exactes et heuristiques employées pour résoudre le problème d'itinéraires de véhicule avec dépôts multiples (en anglais Multiple Depot Vehicle Scheduling Problem (MDVSP)). Celles-ci vont nous servir de base pour résoudre le MDVSP avec décalage d'horaires, dont on explore les algorithmes, avec un ou plusieurs dépôts, dans la seconde partie. Enfin on évoquera certains problèmes proches du MDVSP avec décalage d'horaires.

2.1 Problème d'itinéraires de véhicules avec dépôts multiples

Nous détaillons tout d'abord une partie des travaux réalisés précédemment sur le MDVSP. Pour une revue exhaustive des problèmes d'itinéraires de véhicules avec un ou plusieurs dépôts, on réfère à Bunte et Kliewer (2009).

2.1.1 Approches exactes

Plusieurs approches exactes, qui reposent toutes sur la programmation linéaire en nombres entiers, existent. Elles diffèrent par le choix du réseau ou de l'algorithme de résolution.

Carpaneto et al. (1989) proposent une modélisation du MDVSP par un réseau de *connexions*, résolue grâce une méthode de séparation et évaluation (en anglais *Branch & Bound*). Ribeiro et Soumis (1994) conservent cette même modélisation mais la reformulent comme un problème de partitionnement d'ensemble, qui peut se résoudre à l'aide d'un algorithme de génération de colonnes. Ils montrent que cette technique, que nous utiliserons aussi dans la suite, permet d'augmenter la borne inférieure de la relaxation linéaire, et donc d'accélérer la résolution.

Par la suite, Hadjar et al. (2006) et Groiez et al. (2013) introduisent plusieurs techniques d'accélération du précédent algorithme, notamment en éliminant certains arcs par analyse des coûts réduits et en ajoutant des coupes d'optimalité dans l'arbre de branchement.

Les problèmes de MDVSP sont souvent très *dégénérés*, i.e., plusieurs solutions de même coût existent, et cela peut grandement ralentir la résolution. Oukil et al. (2007) appliquent donc une méthode de stabilisation au MDVSP pour réduire la dégénérescence. Cette stabilisation repose sur le contrôle des coûts duaux proposé par du Merle et al. (1999).

Une autre méthode est étudiée par Löbel (1998) : en s'appuyant sur une formulation multi-flot, il met en place un algorithme de génération de colonnes pour résoudre la relaxation linéaire. Cette fois-ci, les colonnes générées correspondent aux valeurs du flot sur les arcs de connexions. Pour générer de bonnes variables, l'auteur s'appuie sur deux relaxations lagrangiennes du MDVSP et l'évaluation standard du coût réduit des arcs. Dans de nombreux cas, il obtient des solutions entières, et dans les autres, une procédure d'arrondi permet d'atteindre une solution entière optimale - ou proche de l'optimalité.

Kliewer et al. (2006a) résolvent le MDVSP, non pas à l'aide d'un réseau de connexions, mais d'un réseau *espace-temps*. Cette modélisation permet de réduire le nombre d'arcs du graphe et se résout efficacement à l'aide d'un solveur en nombres entiers.

2.1.2 Approches heuristiques

Pour répondre au besoin des compagnies d'autobus d'obtenir rapidement des solutions pour des problèmes de très grande taille, de nombreuses heuristiques ont aussi été développées.

Pepin et al. (2009) présentent et comparent cinq heuristiques qui couvrent bien les méthodes de résolution du MDVSP existantes. Les trois premières utilisent les méthodes de programmation linéaire en nombres entiers, les deux autres sont des métaheuristiques de recherche locale.

- La première heuristique consiste à résoudre directement le MDVSP modélisé avec réseau espace-temps à l'aide d'un solveur linéaire en nombres entiers. En revanche, on n'explore pas la totalité de l'arbre de branchement et on s'arrête dès la première solution entière trouvée. Gintner et al. (2005) utilisent également une formulation similaire mais réduisent au préalable la taille du réseau en identifiant les arcs les plus prometteurs à partir des solutions de plusieurs problèmes d'horaires de véhicules avec un seul dépôt (SDVSP pour Single Depot Vehicle Scheduling Problem en anglais).
- La seconde est une heuristique basée sur une relaxation lagrangienne des contraintes de conservation du flot.
- Une méthode de *génération de colonne tronquée* est également étudiée. Elle consiste à arrêter prématurément la résolution d'une relaxation si la valeur de la solution stagne pendant plusieurs itérations de génération de colonnes. On évite ainsi de consacrer un temps important pour atteindre l'optimalité à chaque nœud, alors que cela n'est pas nécessaire pour obtenir de très bonnes solutions. Cette idée est aussi à la base des

heuristiques développées par Guedes et Borenstein (2015) et Guedes et al. (2016). Dans ce dernier, les auteurs réduisent préalablement le nombre d'arcs par une heuristique de choix des arcs qui semblent les plus pertinents.

— La quatrième méthode considérée est une métaheuristique de *recherche à voisinage large*. A chaque itération, on détruit une partie de la solution courante, en l'occurrence un certain nombre d'itinéraires de véhicule. On reconstruit ensuite une solution en optimisant la partie manquante par génération de colonnes.

— Enfin, Pepin et al. (2009) implémentent un algorithme de *recherche taboue*. Le principe est ici de chercher la meilleure solution parmi les voisins de la solution courante et d'itérer. Un voisin est obtenu en échangeant deux voyages entre deux itinéraires de véhicule. Lors de ces échanges, il est permis de transiter par des solutions dans lesquelles l'heure de certains départs est retardée. Ces solutions non-réalisables sont pénalisées mais permettent d'explorer plus rapidement l'espace des solutions.

Les tests réalisés par Pepin et al. (2009) montrent qu'au prix d'un temps de calcul légèrement plus long, c'est l'algorithme de génération de colonnes tronquée qui donnent les meilleures solutions. Si le temps est plus limité, alors la recherche à voisinage large fournit de très bons résultats.

Laurent et Hao (2009) proposent une autre métaheuristique de recherche locale et la comparent aux solutions de Pepin et al. (2009). A chaque itération, on peut effectuer de petits mouvements (échange de deux trajets entre deux véhicules) ou un *mouvement de bloc* dans lequel on déplace plusieurs voyages consécutifs d'un véhicule vers un autre et, si nécessaire, les voyages non compatibles sont redistribués aux autres véhicules. L'introduction de ce nouveau voisinage conduit à un algorithme très performant, meilleur que la recherche à voisinage large de Pepin et al. (2009). La génération de colonnes tronquée donne toutefois de meilleurs résultats.

2.2 Problème d'itinéraires de véhicules avec décalage d'horaires

Depuis une dizaine d'années, plusieurs approches ont été proposées pour résoudre le problème d'itinéraires de véhicules en intégrant la possibilité de modifier les horaires de départ initiaux. Ainsi les objectifs deviennent dans la plupart des cas doubles : diminuer/ne pas augmenter les coûts et préserver/améliorer la qualité des horaires. Les méthodes de résolution proposées prennent donc en compte un ou plusieurs des objectifs suivants :

— Minimiser le nombre de véhicules

— Diminuer le coût des trajets à vide et des attentes en station
— Conserver des espacements identiques entre les voyages consécutifs d'une même ligne
— Maximiser le nombre de correspondances efficaces pour les passagers
— Minimiser le nombre d'horaires de départ modifiés afin de limiter l'impact sur les utilisateurs.

L'ajout de flexibilité dans le MDVSP et la poursuite de plusieurs objectifs simultanément rendent le problème plus complexe. La littérature propose essentiellement des approches heuristiques pour résoudre ces problèmes. Avant de s'intéresser aux problèmes avec plusieurs dépôts, on détaille les approches pour des problèmes avec dépôt unique.

2.2.1 Approches pour le cas avec un seul dépôt

Contrairement au MDVSP qui est un problème NP-difficile (voir Bertossi et al. (1987)), le SDVSP est un problème qui peut se résoudre en temps polynomial, notamment par des algorithmes de flot.

Guihaire et Hao (2010) et Ibarra-Rojas et al. (2014) s'attaquent au problème de résolution simultanée des horaires de départ et des itinéraires de véhicules. Le premier propose une métaheuristique de recherche locale minimisant une somme pondérée des quatre premiers objectifs cités plus haut. Le second cherche les optimums de Pareto entre le nombre de véhicules et le nombre de transferts possibles en résolvant deux programmes linéaires en nombres entiers (PLNE) successifs.

Schmid et Ehmke (2015) proposent une méthodologie en deux phases pour résoudre un Single Depot Vehicle Scheduling Problem (SDVSP) avec fenêtres de temps larges pour des problèmes d'autobus ruraux. Dans un premier temps, ils minimisent, par un algorithme de recherche à voisinage large, le coût d'opération des véhicules puis trouvent l'horaire optimal en résolvant un PLNE pour avoir de bons espacements entre les voyages.

2.2.2 Approches pour le cas avec plusieurs dépôts

Méthodes heuristiques L'objectif de Petersen et al. (2012) est de résoudre un MDVSP tout en modifiant légèrement les horaires pour améliorer la qualité et le nombre de correspondances. Les auteurs utilisent une recherche à voisinage large, avec une méthode heuristique de reconstruction de la solution. Une métaheuristique de *recuit-simulé* est utilisée par Van den

Heuvel et al. (2008) pour réduire le coût des itinéraires d'autobus, avec plusieurs choix de type de véhicules par trajet.

Liu et Shen (2007) ont proposé une modélisation bi-niveau : le niveau supérieur est un MDVSP résolu par une recherche taboue, le niveau inférieur optimise les horaires pour minimiser le temps de transit des passagers. Les auteurs testent ce modèle sur un petit exemple. Fleurent et Lessard (2009) utilisent également une métaheuristique de recherche locale avec différents voisinages pour résoudre simultanément la planification des horaires et la construction des itinéraires de véhicules. Le problème est formulé à l'aide d'une fonction bi-objectif, l'utilisateur peut donc faire varier les poids accordés à la qualité de l'horaire et au coût de l'affectation des véhicules. Enfin une méthode heuristique pour répartir la charge de chaque véhicule et contrôler les espacements entre les voyage est étudiée par Ceder (2011).

Méthode exacte et heuristique Kliewer et al. (2006b) proposent des approches exactes et heuristiques pour résoudre le MDVSP avec décalage d'horaires. Les auteurs utilisent un réseau espace-temps dans lequel ils représentent les voyages décalés par des arcs *copie* du voyage initial. Seules les copies qui permettent une nouvelle connexion sont conservées. Le modèle multi-flot est ensuite résolu par un solveur en nombres entiers. Pour accélérer l'algorithme, une méthode de sélection heuristique des voyages "décalables" est proposée. En pénalisant les arcs copies, les auteurs conservent le plus possible la grille horaire initiale.

MDVSP avec fenêtre de temps Un problème similaire au MDVSP avec décalage d'horaires est le MDVSP avec fenêtres de temps (MDVSPTW). On considère alors que les heures de départ ne sont plus limitées à un nombre fini de possibilités mais peuvent être choisies dans un intervalle de temps continu. Ce problème a notamment été traité par Desaulniers et al. (1998) et Hadjar et Soumis (2009) qui le résolvent par génération de colonnes tronquée. Le premier prend en compte le coût exact des attentes entre deux voyages. Le second propose un moyen de réduire la largeur des fenêtres de temps et donc d'accélérer la résolution.

2.3 Problèmes connexes ou englobant le problème d'itinéraires de véhicules avec décalage d'horaires

Quelques auteurs se sont également intéressés à des problèmes qui intègrent le MDVSP avec décalage d'horaires à une autre étape du problème d'optimisation des transports publics décrit dans l'introduction.

2.3.1 Construction simultanée des horaires des véhicules et des chauffeurs avec décalage des heures de départ

Deux articles s'intéressent à l'intégration du MDVSP avec décalage d'horaires à la construction des horaires journaliers des chauffeurs d'autobus. Kéri et Haase (2008) proposent un modèle théorique pour résoudre simultanément ces trois étapes. Kliewer et al. (2012) étendent Kliewer et al. (2006b) pour intégrer la construction des horaires de chauffeurs. L'algorithme utilise la génération de colonnes et la relaxation lagrangienne des contraintes de couverture et de lien entre les trajets de véhicule et les horaires des chauffeurs. Une version exacte et une autre heuristique - par élimination des arcs les moins pertinents - sont proposées.

2.3.2 Construction des lignes et horaires d'autobus avec intégration des horaires de véhicules

De la même façon, certains travaux très récents ont cherché à intégrer la confection des lignes ou l'affectation des passagers au problème des itinéraires d'autobus. En particulier, Laporte et al. (2017) propose une méthode de confection simultanée des horaires de départ, de l'affectation des passagers et des trajets de véhicule. Cependant, les itinéraires d'autobus sont très contraints et donc plus simples à résoudre car seuls les allers-retours sur une même ligne sont permis. Enfin Schöbel (2017) effectue un travail théorique de construction et de classification des métaheuristiques envisageables pour résoudre simultanément la confection des lignes, celle des horaires et celle des itinéraires des autobus.

2.3.3 Problèmes similaires

Le MDVSP, utile pour les autobus, présente des similarités avec le problème d'horaires d'avions, à la différence notable que les trajets sans passagers ne sont pas autorisés pour les avions. En particulier, Bélanger et al. (2006) s'intéressent au cas où les horaires des vols peuvent être choisis dans une fenêtre de temps. L'horaire des vols doit être périodique de jour en jour et on pénalise les espacements trop courts entre les vols d'un même couple Origine-Destination. Le profit de la compagnie aérienne varie de plus avec les heures de départ. Les auteurs résolvent ce problème par génération de colonnes avec des coûts spécifiques sur les nœuds du réseau de sous-problème (et non seulement sur les arcs comme traditionnellement).

2.4 Contribution de ce mémoire

Nous avons vu en 2.2.2 plusieurs algorithmes de résolution du MDVSP avec décalage d'horaires ou avec fenêtres de temps. A l'instar de Kliewer et al. (2006a) et Kliewer et al. (2012), notre travail propose à la fois une méthode exacte et une méthode heuristique pour résoudre le MDVSP avec décalage de voyages. Cette double approche permet d'attester la pertinence des résultats obtenus heuristiquement. Cependant, contrairement à ces deux articles, nous utilisons un réseau de connexions (défini dans le chapitre suivant), et non un réseau espace-temps, afin de pouvoir adapter facilement le modèle à certaines contraintes opérationnelles des compagnies d'autobus. Ce choix conduit cependant à un ralentissement notable de la vitesse de résolution. Nous proposons donc un nouveau réseau hybride (voir chapitre 3) pour lequel l'ajout du décalage d'horaires a seulement une influence limitée sur la taille du réseau.

Enfin, l'approche en deux phases que nous proposons pour contrôler la qualité des horaires (voir chapitre 5) peut prendre en compte toutes les contraintes sur le contrôle des horaires sans interférer sur la complexité du MDVSP.

CHAPITRE 3 RÉSEAUX DE CONSTRUCTION DES ITINÉRAIRES D'AUTOBUS

Dans ce chapitre, nous allons présenter différents réseaux qui modélisent le problème de construction des itinéraires d'autobus. Ces réseaux seront ensuite intégrés à l'algorithme de génération de colonnes qui sera décrit au chapitre 4. On définit tout d'abord le MDVSP (section 3.1). Dans les deux sections suivantes, on compare plusieurs réseaux de modélisation du MDVSP puis du MDVSP avec décalage d'horaires.

3.1 Définition du problème et notations

On considère n voyages à effectuer $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$. Un voyage v part au temps t_v^s de la station S_v et arrive au temps t_v^e à sa station finale E_v. On dispose d'un ensemble de m dépôts $\mathcal{D} = \{d_1, d_2, \ldots, d_m\}$ de capacité respective $r_1, r_2, \ldots, r_m$.

On appellera *itinéraire* un enchaînement de voyages pouvant être effectué par un véhicule partant et revenant du même dépôt. On dira qu'un véhicule effectue une *connexion* lorsqu'il enchaîne deux voyages. Une connexion peut représenter une attente en station et/ou un trajet à vide entre deux stations - par la suite appelé trajet *haut-le-pied*. Il est également possible de faire une pause au dépôt initial.

On adopte la structure de coût suivante : il y a un coût fixe c_f pour l'utilisation journalière d'un véhicule, un coût c_a par minute d'attente en station, un coût c_v par minute de trajet à vide (que ce soit un trajet haut-le-pied ou entre le dépôt et une station). L'attente au dépôt n'est pas payée.

L'objectif du MDVSP est de trouver un ensemble d'itinéraires de véhicule qui couvre exactement une fois chaque voyage de $\mathcal{V}$ et qui respecte la capacité de chaque dépôt. Pour résoudre ce problème, nous représentons les itinéraires de véhicule par des chemins dans un réseau. On verra en détail au chapitre 4 la formulation mathématique et l'algorithme (une méthode de *Branch and Price*) que nous allons utiliser pour résoudre ce problème. Les réseaux permettront de créer et d'évaluer le coût des itinéraires admissibles par recherche de plus courts chemins dans un graphe.

On présente et compare aux sections 3.2 et 3.3 différents modèles de réseaux pour la résolution du MDVSP classique puis du MDVSP avec décalage d'horaires.

3.2 Réseaux sans décalage d'horaires

On présente dans cette section trois types de réseaux pour le MDVSP classique, i.e. sans décalage d'horaires. On s'intéresse d'abord au réseau de *connexions* dans lequel un arc représente une connexion possible entre deux voyages. Puis on verra une modélisation plus compacte en terme de nombre d'arcs, à savoir le réseau *espace-temps*. Enfin on introduira une formulation hybride entre ces deux modèles.

3.2.1 Le réseau de connexions

Le réseau le plus classique pour modéliser un MDVSP est un graphe dans lequel chaque nœud représente un voyage à couvrir et chaque arc représente une connexion admissible entre deux voyages. Plus précisément, pour chaque dépôt $d \in \mathcal{D}$, on crée un graphe $\mathcal{G}^d$ avec un nœud source et un nœud puits représentant le dépôt. Pour tout voyage de $\mathcal{V}$, on ajoute un *nœud voyage* qui est relié à la source et au puits. On crée un arc connexion entre les nœuds v_i et v_j si et seulement si on peut enchainer le voyage v_j après le voyage v_i, i.e., si le temps de trajet entre les stations E_i et S_j est inférieur à $t^s_{v_j} - t^e_{v_i}$. Le coût d'un arc connexion est le minimum entre :

- Le coût du voyage haut-le-pied (si la station d'arrivée de v_i est différente de la station de départ de v_j), plus le coût d'attente en station. On peut limiter le temps de connexion permis entre deux voyages utiles.
- Le coût associé à faire une pause au dépôt entre v_i et v_j, c'est-à-dire le coût d'un voyage à vide de E_i au dépôt d plus le coût d'un trajet de d à S_j. L'attente au dépôt est gratuite.

Les arcs provenant de la source ont tous un coût égal au coût fixe c_f d'utilisation d'un véhicule plus le coût du trajet du dépôt au départ du voyage. Le coût d'un arc retour au dépôt est celui du trajet de l'arrivée du voyage au dépôt.

La figure 3.1 donne un exemple de graphe de connexions avec quatre voyages pour un dépôt d donné. Les nœuds S et P correspondent respectivement à la source et au puits et représentent le dépôt d.

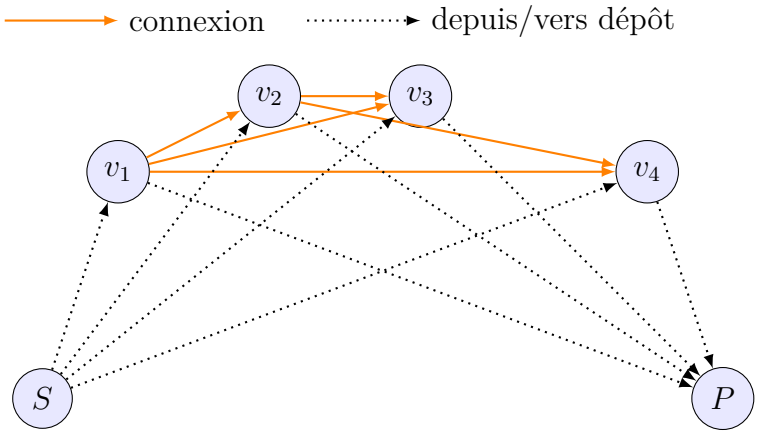

Figure 3.1 Réseau de connexions

3.2.2 Le réseau espace-temps

Pour résoudre le MDVSP, Kliewer et al. (2006a) ont proposé d'utiliser un réseau *espace-temps*, type de réseau introduit pour les problèmes d'horaires par Desrosiers et al. (1995). Celui-ci ne considère pas directement les connexions mais représente les déplacements dans l'espace et le temps. Ici un nœud représente un moment t donné à un endroit L donné et sera noté ici (t, L). Un arc représente donc un mouvement dans le temps et éventuellement l'espace et sera noté $(t, L) \mapsto (t', L')$. L'illustration 3.2 donne un aperçu de ce réseau pour 4 voyages qui circulent entre 3 stations différentes. On définit précisément les éléments du réseau espace-temps ci-dessous.

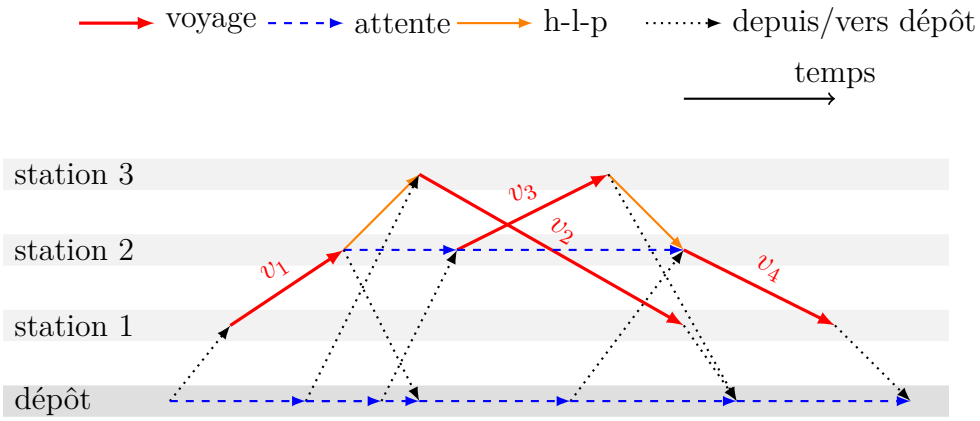

Figure 3.2 Réseau espace-temps

Soit $d \in \mathcal{D}$ un dépôt donné, le réseau $\mathcal{G}^d$ ainsi :

— A chaque voyage v, on associe un nœud départ (t_v^s, S_v) et un nœud arrivée (t_v^e, E_v) qu'on relie pour former un *arc voyage*.
— A chaque voyage v, on associe un *arc sortie du dépôt* et un *arc retour au dépôt* et les nœuds au dépôt correspondant $(t_v^s - t_{D,S_v}, D)$ et $(t_v^e + t_{E_v,D}, D)$.
On obtient les arcs : $(s_v - t_{D,S_v}, D) \mapsto (t_v^s, S_v)$ et $(t_v^e, E_v) \mapsto (t_v^e + t_{E_v,D}, D)$.
— Entre les nœuds associés au même lieu (station ou dépôt), on crée des *arcs attente*. Si pour un lieu L donné, on a les nœuds $(t_1, L), (t_2, L), \ldots, (t_k, L)$ avec $t_1 < t_2 < \ldots < t_k$, alors on crée les arcs $(t_1, L) \mapsto (t_2, L)$, $(t_2, L) \mapsto (t_3, L), \ldots$
— On ajoute des *arcs haut-le-pied* entre les nœuds arrivée et les nœuds départ de stations différentes quand la connexion est réalisable.

Le tableau 3.1 présente le coût d'un arc $(t, L) \to (t', L')$ en fonction de son type.

Tableau 3.1 Coût d'un arc $(t, L) \to (t', L')$ dans le réseau espace-temps

Type d'arc	**Coût**
voyage	0
sortie/retour dépôt	$c_v t_{L,L'}$
attente	$c_a(t' - t)$
haut-le-pied	$c_v t_{L,L'} + c_a(t' - t - t_{L,L'})$

Réduction du nombre d'arcs haut-le-pied

Comme proposé dans Kliewer et al. (2006a), la structure du réseau espace-temps permet de réduire considérablement le nombre d'arcs par rapport au réseau de connexions. Les arcs représentant des trajets haut-le-pied peuvent en effet être agrégés. Par exemple, considérons les trajets haut-le-pied (h-l-p) de la station 1 vers la station 2. On peut consécutivement faire les deux réductions suivantes (voir figure 3.6) :

1. si plusieurs arcs h-l-p partent du même nœud (représentant l'arrivée d'un voyage) vers la station 2, on garde seulement l'arc qui arrive le plus tôt à la station 2. En effet, les autres arcs peuvent être remplacés par cet arc h-l-p et l'attente à la station 2 (voir figure 3.4).
2. si plusieurs arcs partant de la station 1 arrivent au même nœud de la station 2, on ne garde que le dernier arc, c'est-à-dire celui partant le plus tard de la station 1 (voir figure 3.5).

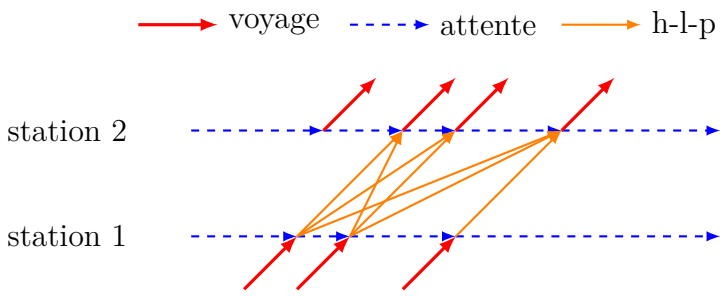

Figure 3.3 Réseau initial

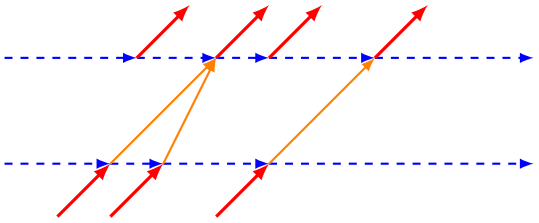

Figure 3.4 Étape 1

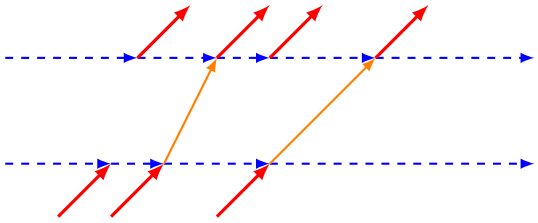

Figure 3.5 Étape 2

Figure 3.6 Réduction du nombre d'arcs haut-le-pied dans un réseau espace-temps

3.2.3 Discussion sur le choix du réseau de connexions ou espace-temps

La structure du réseau espace-temps permet de réduire significativement le nombre d'arcs représentant les trajets haut-le-pied possibles, et les attentes possibles. En terme de temps de résolution, on verra que ce réseau est effectivement beaucoup plus efficace. Cependant, le réseau de connexions permet de modéliser facilement un grand nombre de contraintes qui peuvent apparaître dans les problèmes réels. Prendre en compte ces contraintes avec le réseau espace-temps est en général possible mais au prix d'ajout de ressources ou de contraintes, ce qui ralentit l'algorithme.

Par exemple, les contraintes listées ci-dessous se modélisent par une simple modification de coût sur les arcs ou le retrait d'un arc dans le réseau de connexions :

- interdire l'enchaînement de deux voyages spécifiques
- pénaliser les changements de lignes
- pénaliser/interdire les connexions trop longues
- pénaliser les connexions trop courtes

Le réseau espace-temps ne permet pas de prendre en compte facilement ces contraintes pour

deux raisons : d'une part, il est bien adapté seulement pour les structures de coût linéaire en fonction du temps. D'autre part, comme les attentes en station sont agrégées, une partie de l'information "se perd" lorsqu'on construit un chemin.

Dans l'optique de proposer une modélisation qui soit facilement adaptable aux besoins des compagnies d'autobus, nous avons choisi d'utiliser un réseau type "connexion". On étudie dans la suite des moyens de réduire la taille du réseau de connexions présenté à la section 3.2.1.

3.2.4 Le réseau de connexions mixte

On présente ici une adaptation du graphe de connexions, initialement proposée par Desrosiers et al. (1995), dans laquelle on réduit le nombre d'arcs en utilisant le principe de construction du réseau espace-temps. En effet, comme expliqué dans la section 3.2.1, certains arcs de connexion représentent une pause au dépôt. Ces arcs sont potentiellement très nombreux car, à partir d'un certain temps d'attente, il devient avantageux de repasser par le dépôt. Dans le réseau de connexions *mixte*, on supprime ces arcs de pause par le dépôt et on les remplace par la combinaison suivante : un arc *retour au dépôt*, des arcs *attente au dépôt* et un arc *sortie du dépôt*. On peut ainsi espérer réduire de façon importante la taille du réseau.

La figure 3.7 montre comment est transformé le réseau de connexions classique de la figure 3.1. On évite ainsi les arcs $v_1 \rightarrow v_4$ et $v_2 \rightarrow v_4$ qui représentaient des passages au dépôt.

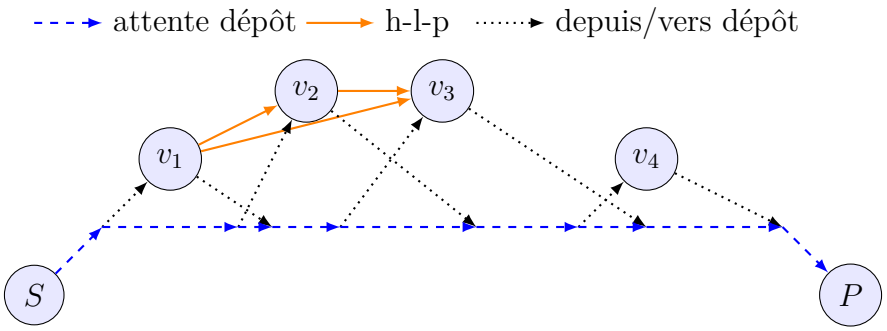

Figure 3.7 Réseau mixte de connexions

Agrégation des nœuds au dépôt

Pour le réseau mixte (valable aussi pour le réseau espace-temps), on peut de plus réduire le nombre de nœuds et d'arcs attente au dépôt en utilisant une technique d'agrégation classique,

illustrée à la figure 3.8.

Pour cela, on classe par ordre de temps croissant les événements départ du/retour au dépôt, puis :

— On forme les groupes de départs consécutifs et de retours consécutifs

— On regroupe chaque groupe de retours avec le groupe de départ qui le suit. Ainsi on s'assure qu'on n'aura jamais un chemin dans le graphe qui prendrait un départ avant l'heure de retour au dépôt prévue.

Réseau initial

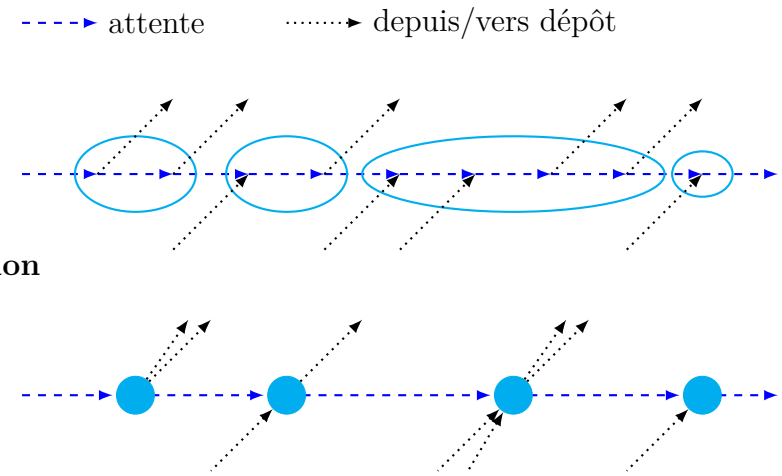

Figure 3.8 Agrégation des arcs attente au dépôt.

Remarque 1. *Cette modélisation peut être inadéquate ou inexacte si l'on veut contrôler précisément le temps passé au dépôt, par exemple pour des autobus électriques qui nécessitent un temps de charge.*

Conclusion sur les réseaux sans décalage d'horaires

Au chapitre 4, on étudiera numériquement l'influence du choix de réseau pour résoudre un MDVSP. On comparera donc :

— le réseau de connexions (appelé **cnx** dans les tableaux de résultat)

— le réseau espace-temps (appelé **ret**)

— le réseau de connexions mixte (appelé **mix**)

— le réseau de connexions mixte avec agrégation des nœuds au dépôt (appelé **mix***).

Même si le réseau espace-temps est plus compact que les réseaux de connexions et mixte, on ne souhaite pas l'utiliser afin d'avoir un modèle plus adaptable aux contraintes opérationnelles des compagnies d'autobus. Le défi de la section suivante sera donc de trouver une façon d'ajouter le décalage d'horaires à un réseau de connexions mixte, en gardant une taille de réseau raisonnable.

3.3 Réseaux avec décalage d'horaires

Les trois types de graphe pour le MDVSP détaillés précédemment vont nous servir de base pour le MDVSP avec décalage d'horaires. On adopte une approche discrète, i.e., pour chaque voyage v, on considère un ensemble fini d'heures de départ possibles. A chacune de ces heures, on associe une copie u du voyage v et on note M_v l'ensemble des copies représentant le voyage v. Dans les exemples, on considérera que pour tout $v \in \mathcal{V}$, $M_v = \{a, b, c\}$ est un ensemble de 3 copies avec : b la copie initiale, a la copie avancée de 2 min et c la copie retardée de 2 min.

L'objectif du MDVSP avec décalage d'horaires est de minimiser le coût des itinéraires de véhicule pour qu'exactement une copie de chaque voyage soit couverte, sous contrainte de capacité des dépôts.

On va donc construire des réseaux qui prennent en compte le fait qu'un voyage est maintenant associé à un ensemble de copies. Tout d'abord, on étudie une adaptation naïve des réseaux précédents par multiplication du nombre de voyages. Ces premiers modèles naïfs mèneront ensuite à un réseau plus compact utilisant des sous-réseaux de choix de copie.

3.3.1 Multiplication du nombre de voyages

Pour introduire le décalage d'horaires, on peut simplement considérer toutes les copies de voyage comme des voyages. Ainsi on remplace $\mathcal{V}$ par $\mathcal{M} = \bigcup_{v \in \mathcal{V}} M_v$, et la taille du graphe devient similaire à celle d'un problème avec $\sum_{v \in \mathcal{V}} |M_v|$ voyages. Cela augmente considérablement la taille du réseau car le nombre de connexions possibles d'un réseau à n voyages est grossièrement équivalent à n^2.

Réseau de connexions et réseau mixte Dans un réseau type connexions (réseau de connexions ou réseau de connexions mixte), on ajoute un nœud voyage par copie et autant d'arcs de connexions les reliant que nécessaires. La figure 3.9 donne la représentation de v_1 si trois copies sont possibles. On a ainsi 3 nœuds et les arcs connexion possibles pour chacun

d'entre eux. Si les copies sont proches dans le temps alors les connexions possibles sont très similaires d'une copie à l'autre. Dans cet exemple, on note toutefois deux différences : c permet une nouvelle connexion entrante tandis qu'une connexion sortante possible pour a et b ne l'est plus pour c.

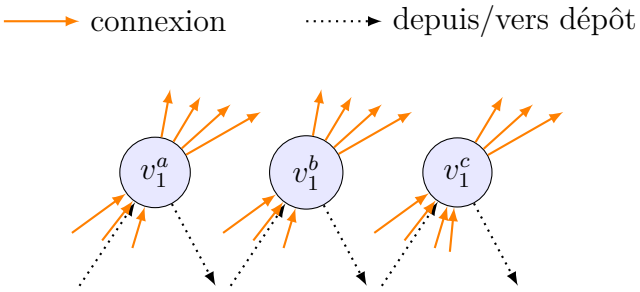

Figure 3.9 Représentation du voyage v_1 avec $M_v = \{a, b, c\}$ pour un réseau de connexions ou mixte

Réseau espace-temps Lorsque le décalage d'horaires est permis, il y a autant d'*arcs voyage* que de copies et le réseau espace-temps est construit comme précédemment. De la même façon qu'à la figure 3.9, on donne la représentation du voyage v_1 pour le réseau espace-temps à la figure 3.10. Cette fois-ci le nombre d'arcs haut-le-pied "n'explose" pas avec le nombre de copies car les arcs haut-le-pied redondants sont éliminés grâce à la réduction décrite à la figure 3.6.

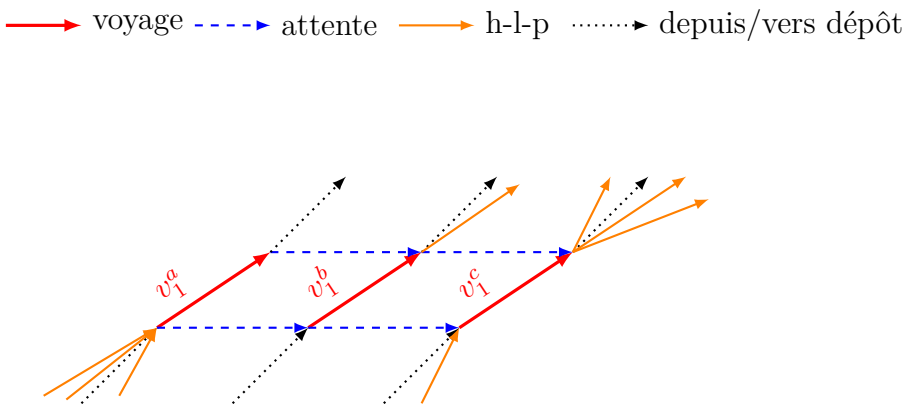

Figure 3.10 Représentation du voyage v_1 avec $M_v = \{a, b, c\}$ pour un réseau espace-temps

Conclusion Le modèle de la figure 3.9 ne met pas à profit la structure particulière des modèles avec décalage puisqu'il se contente de multiplier le nombre de voyages par le nombre

de copies. En effet, on peut s'attendre à ce que les copies d'un même voyage aient des arcs connexion fortement similaires entre elles, si les espacements entre les copies sont courts. De plus, on n'exploite pas le fait que seul un des nœuds peut être visité. Au contraire, la réduction d'arcs haut-le-pied possible avec le réseau espace-temps limite fortement le nombre d'arcs à introduire pour prendre en compte le décalage d'horaires. Notre objectif étant de trouver une modélisation type connexion et non espace-temps, nous proposons ci-dessous de remplacer les nœuds voyage par un *sous-réseau* espace-temps local.

3.3.2 Sous-réseau de choix de copie

Au vu de la conclusion précédente, un modèle intéressant serait de remplacer, à l'intérieur du réseau de connexions, chaque triplet de nœuds copies par un sous-réseau espace-temps comme celui de la figure 3.10 [1]. Ainsi on éviterait de multiplier inutilement les arcs de connexion. Cependant, le réseau de la figure 3.10 a le défaut d'introduire de la symétrie - plusieurs chemins équivalents sont possibles - ce qui ralentit l'algorithme général. On propose donc de "compacter" ce réseau ce qui donne le sous-graphe de la figure 3.11.

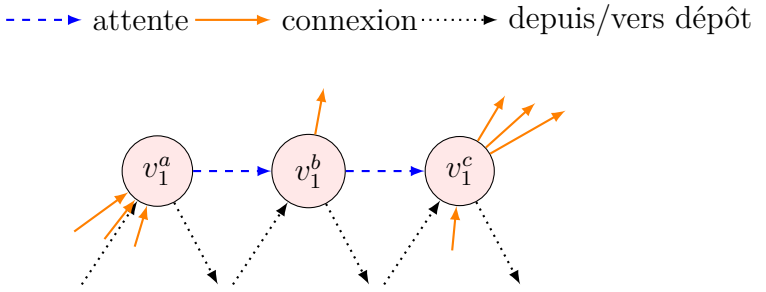

Figure 3.11 Sous-réseau de choix pour le voyage v_1 dans un réseau de connexions.

Dans l'exemple des figures 3.9 et 3.11, on passe de 21 à 8 arcs de connexion grâce à l'introduction de ce petit réseau de choix de copies. Dans la figure 3.11, chaque arc de connexion apparaît une seule fois, mais nous verrons plus bas que, dans certains cas, une même connexion peut être représentée par plusieurs arcs.

Non-fixation des copies Un chemin qui traverse le sous-réseau entre par une copie et sort par celle-ci ou par une copie ultérieure. Si l'on entre dans le sous-réseau par a et qu'on en sort par un arc connexion de b, alors les copies a et b sont admissibles. Ainsi, contrairement aux modèles des figures 3.9 et 3.10, choisir un chemin n'impose pas toujours la copie.

1. Dans lequel les arcs de connexions seraient traités comme les arcs haut-le -pied.

Nous verrons que cette caractéristique sera très importante pour le contrôle de la qualité des horaires (voir chapitre 5).

On peut de plus remarquer que le sous-réseau de la figure 3.11 met directement en évidence les connexions qui profitent du décalage d'horaires et leurs conséquences sur le choix de copie possible.

On explique dans le paragraphe suivant précisément comment sont créés les arcs de connexion entre les sous-réseaux et quelles copies sont reliées entre elles.

Construction des arcs de connexion entre les sous-réseaux Soit deux voyages v et v', on veut construire le ou les éventuels arcs de connexion reliant les copies de v à celles de v'. On suppose qu'on a p copies pour chacun de ces voyages et que l'écart Δ_t entre ces copies est constant. On note de plus t_1 l'heure de départ de la première copie de v. La dernière copie part donc à $t_p = t_1 + (p-1)\Delta_t$. On note de même $t_{1'}, \ldots, t_{p'}$ les heures de départ des copies de v'.

On note T le temps minimal nécessaire entre les départs des voyages v et v' (la durée du voyage v plus le temps pour aller de l'arrivée de v au départ de v'). On se place nécessairement dans l'un des cas suivants (voir figure 3.12) :

- **si** $T \leq t_{1'} - t_p$ alors la connexion est possible quelles que soient les copies v et v' choisies. On crée donc un unique arc de connexion entre v^p et v'^1
- **sinon si** $T \leq t_{1'} - t_p + \Delta_t = t'_1 - t_{p-1} = t_{2'} - t_p$ alors on relie d'une part v^{p-1} à v'^1 et d'autre part v^p à v'^2. On s'assure ainsi de n'empêcher aucune connexion réalisable.
- **sinon si** $T \leq t_{1'} - t_p + 2\Delta_t = t_{1'} - t_{p-2} = t_{2'} - t_{p-1} = t_{3'} - t_p$ alors on doit ajouter les 3 arcs suivants : $v^{p-2} \to v'^1$, $v^{p-1} \to v'^2$ et $v^p \to v'^3$.

- etc

- **sinon si** $T \leq t_{1'} - t_p + 2(p-1)\Delta_t = t_{p'} - t_1$ alors on crée l'arc $v^1 \to v'^p$ qui est le seul choix permettant d'enchaîner v et v'.
- **si** $T > t_{p'} - t_1$ alors la connexion n'est pas possible.

Ainsi, si le temps nécessaire pour connecter deux voyages est substantiellement plus court que la différence entre le temps de départ de ces voyages, on peut représenter la connexion

par un seul arc. Si le temps de connexion est proche du temps entre les deux voyages, alors il est nécessaire d'introduire plusieurs arcs "parallèles". Ces cas sont moins fréquents mais ils introduisent de la symétrie dans le réseau.

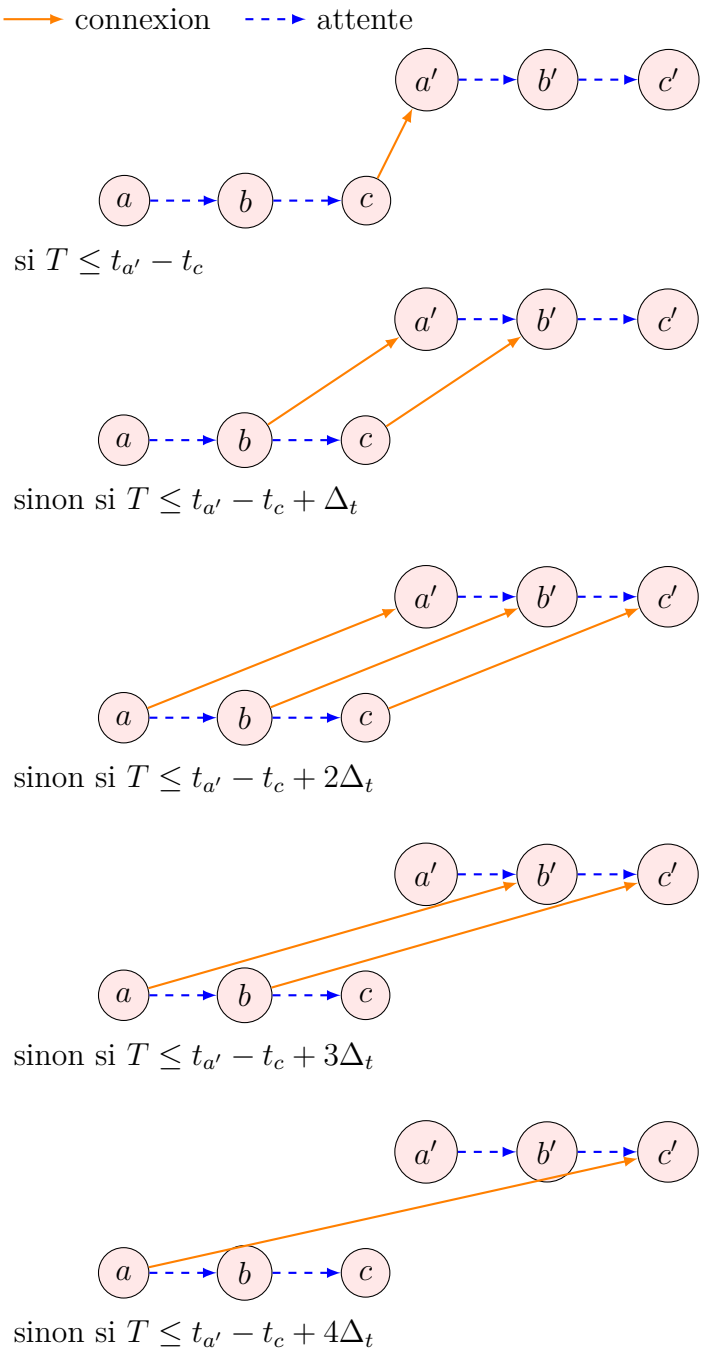

Figure 3.12 Représentation des 5 cas possibles de connexion lorsque v et v' ont chacun 3 copies $\{a, b, c\} = \{1, 2, 3\}$.

3.3.3 Conclusion sur les réseaux avec décalage d'horaires

Finalement, pour résoudre le MDVSP avec décalage d'horaires, nous pouvons utiliser un réseau de connexions mixte (avec agrégation des nœuds du dépôt) dans lequel les nœuds voyages sont remplacés par des sous-réseaux de choix de copie de la figure 3.11. Ce réseau est représenté schématiquement à la figure 3.13.

Nous avons ainsi vu qu'en combinant les idées du réseau espace-temps avec le modèle de connexions, on parvient à limiter la complexité du réseau, tout en gardant les avantages du modèle de connexions.

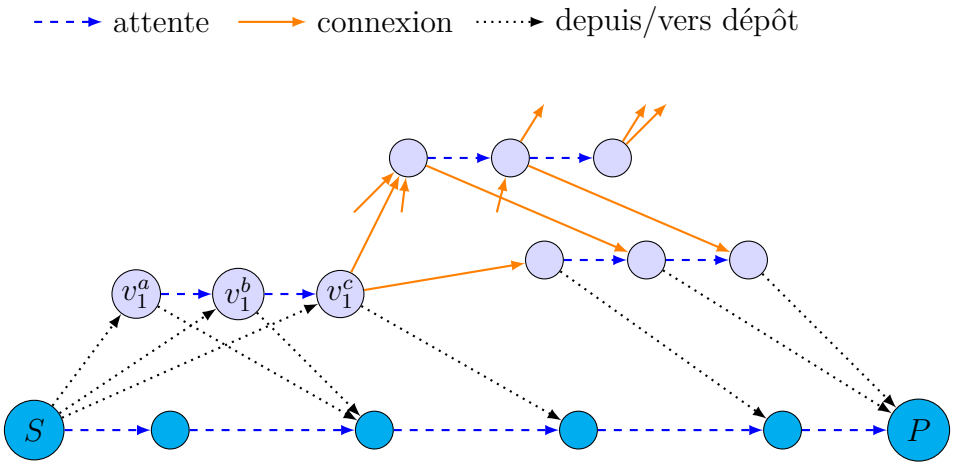

Figure 3.13 Schéma récapitulatif d'un réseau de connexions mixte avec agrégation des nœuds au dépôt, où chaque voyage est représenté par un sous-réseau avec 3 copies.

Dans le chapitre suivant (section 4.2.3), on comparera le nombre d'arcs et le temps de calcul nécessaire pour les différents réseaux évoqués avec trois copies, à savoir :

- le réseau de connexions avec multiplication des nœuds-voyages (noté **cnx_c3** dans les tableaux de résultat)
- le réseau mixte avec multiplication des nœuds-voyages (noté **mix*_c3**)
- le réseau espace-temps avec multiplication des arcs-voyages (noté **ret*_c3**)
- le réseau mixte avec les sous-réseaux de choix de copie de la figure 3.11 (noté **src*_c3**) ou les sous-réseaux de la figure 3.10 (noté **src2*_c3**) .

CHAPITRE 4 MÉTHODES DE RÉSOLUTION EXACTES ET HEURISTIQUES PAR GÉNÉRATION DE COLONNES

Les différents réseaux présentés au chapitre précédent sont le support des algorithmes de résolution que nous allons maintenant étudier. Ceux-ci sont tous fondés sur la technique de génération de colonnes, algorithme que l'on présente à la section 4.1. On analyse ensuite les temps de calcul pour les différents réseaux présentés au chapitre 3. La suite de ce chapitre porte sur des techniques d'accélération de l'algorithme : on verra d'abord la méthode (exacte) d'élimination d'arcs par analyse des coûts réduits puis une méthode heuristique. L'efficacité de ces techniques sera attestée numériquement dans la dernière section.

4.1 Résolution du MDVSP par génération de colonnes

Ribeiro et Soumis (1994) ont les premiers proposé d'utiliser un algorithme de génération de colonnes pour résoudre le MDVSP et cette approche a été conservée par la suite par de nombreux auteurs (voir la section 2.1).

4.1.1 Formulation par partitionnement en circuits

On présente d'abord la formulation du MDVSP classique, puis son adaptation lorsque les voyages peuvent être décalés.

Résoudre le MDVSP revient à chercher un ensemble d'itinéraires couvrant tous les voyages à coût minimum. Autrement dit, on partitionne l'ensemble $\mathcal{V}$ en itinéraires de véhicules, qui peuvent provenir de différents dépôts. Soit $\mathcal{G}^d$ l'un des graphes détaillés au chapitre précédent, associé à un dépôt $d \in \mathcal{D}$. Quel que soit le type de réseau choisi, on remarque qu'un chemin admissible de la source au puits dans $\mathcal{G}^d$ représente un itinéraire de véhicule partant et revenant au dépôt d. Notons alors Ω^d l'ensemble de ces chemins du réseau $\mathcal{G}^d$. Pour tout $p \in \Omega^d$, on calcule son coût c_p comme la somme des coûts des arcs de ce chemin. Pour tout $v \in \mathcal{V}$, on introduit le paramètre a_{vp} égal à 1 si l'itinéraire p contient le voyage v et à 0 sinon.

Le MDVSP peut donc s'écrire comme le problème linéaire en nombres entiers suivant, avec y_p la variable binaire égale à 1 si l'itinéraire p est choisi, à 0 sinon :

$$\min_{y} \sum_{d\in\mathcal{D}} \sum_{p\in\Omega^d} c_p y_p \tag{4.1}$$

$$s.c. \quad \sum_{d\in\mathcal{D}} \sum_{p\in\Omega^d} a_{vp} y_p = 1 \qquad \forall v \in \mathcal{V} \tag{4.2}$$

$$\sum_{p\in\Omega^d} y_p \leq r_d \qquad \forall d \in \mathcal{D} \tag{4.3}$$

$$y_p \in \{0,1\} \qquad \forall p \in \Omega^d,\ \forall d \in \mathcal{D} \tag{4.4}$$

On minimise le coût des itinéraires choisis (4.1), sous la contrainte de prise en compte exactement une fois de chaque voyage (4.2) et de respect de la capacité de chaque dépôt (4.3).

Le MDVSP avec décalage d'horaires peut s'écrire avec la même formulation en définissant cette fois-ci :

— l'ensemble Ω^d comme l'ensemble des chemins de la source au puits d'un réseau avec décalage d'horaires (réseaux décrits à la section 3.3)

— le paramètre a_{vp} est maintenant : $a_{vp} = 1$ si p couvre une copie de v et à 0 sinon.

4.1.2 Algorithme de génération de colonnes

Excepté les cas où le nombre de voyages n est très petit, il est déraisonnable de générer explicitement les ensembles Ω^d et encore plus d'introduire autant de variables dans un programme linéaire en nombres entiers. La génération de colonnes permet de résoudre ce problème en générant au fur et à mesure de bons chemins admissibles. Le principe général consiste à résoudre le problème avec seulement un petit sous-ensemble de Ω^d et, à partir de la solution trouvée, à générer quelques nouveaux itinéraires pertinents puis à les ajouter au sous-ensemble courant.

Formellement, on appelle (4.1)-(4.4) le *problème maître* (PM) et, en relaxant la contrainte d'intégrité (4.4), on définit sa relaxation linéaire (PM^{rl}). Pour résoudre (PM^{rl}), on commence par résoudre le problème maître *restreint* (PMR^{rl}) dans lequel $\Omega = \bigcup_d \Omega^d$ est remplacé par un de ses sous-ensembles $\Omega_R \subset \Omega$.

Supposons avoir résolu (PMR^{rl}) pour un certain ensemble Ω_R, on dispose alors des variables duales $(\lambda_v)_{v\in\mathcal{V}}$ et $(\mu_d)_{d\in\mathcal{D}}$ associées aux contraintes (4.2) et (4.3) respectivement. Soit $p \in \Omega^d$

un itinéraire admissible, son coût réduit relativement à (PMR^{rl}) est :

$$\bar{c}_p = c_p - \sum_{v \in \mathcal{V}} \lambda_v a_{vp} - \mu_d. \tag{4.5}$$

Deux cas se présentent alors :

1. S'il existe un ou plusieurs chemins $p \in \Omega$ tels que $\bar{c}_p < 0$ alors, en ajoutant ces itinéraires - ou colonnes - à Ω_R et en résolvant (PMR^{rl}) de nouveau, on peut diminuer la valeur de l'objectif (4.1).
2. Si, au contraire, il n'existe pas de tel chemin, on est assuré d'avoir atteint l'optimalité de (PM^{rl}) puisque toutes les variables ont un coût réduit positif ou nul.

Pour savoir si on se place dans le premier ou le second cas et générer les chemins p dans le premier cas, il suffit de résoudre un problème de plus court chemin dans chaque réseau $\mathcal{G}^d$. En effet, si on modifie le coût des arcs de $\mathcal{G}^d$ de façon adéquate [1], la somme des arcs d'un chemin de s à p donne le coût réduit (4.5) de l'itinéraire correspondant. Ce problème de génération des nouveaux itinéraires est appelé *sous-problème*.

Comme il y a un nombre fini de variables dans Ω, on atteint en un nombre fini d'itérations l'étape 2 et on résout ainsi (PM^{rl}). Si la solution optimale de (PM^{rl}) obtenue n'est pas entière, on peut prendre une décision de branchement (voir les méthodes décrites ci-dessous) et créer plusieurs nœuds fils que l'on résout eux aussi par génération de colonnes. Ainsi, on résout (PM) par un algorithme de séparation et évaluation (*Branch & Bound*) dans lequel chaque nœud (relaxation linéaire) est résolu par génération de colonnes, autrement dit on utilise un algortihme de *Branch & Price*.

On résume maintenant les étapes de l'algorithme de génération de colonnes.

Algorithme de résolution de (PM^{rl})

1. Choisir un ensemble d'itinéraires initial $\Omega_R \subset \Omega$ tel qu'une solution réalisable de (PM^{rl}) existe, ou ajouter des variables artificielles de coût important aux contraintes (4.2).
2. Résoudre (PMR^{rl}) par programmation linéaire (algorithme du simplexe primal ou dual, ou méthode de points intérieurs)

1. Cela dépend du réseau, par exemple pour un réseau de connexions, on soustrait au coût de tous les arcs sortants d'un noeud voyage v, la valeur de la variable duale λ_v.

3. Récupérer les valeurs des variables duales $(\lambda_v)_{v\in\mathcal{V}}$ et $(\mu_d)_{d\in\mathcal{D}}$ associées aux contraintes de (PMR^{rl})
4. Mettre à jour le coût des arcs des graphes $(\mathcal{G}^d)_{d\in\mathcal{D}}$ pour prendre en compte ces variables duales
5. Chercher le plus court chemin dans chacun des graphes $\mathcal{G}^d$ et calculer le coût minimal $\bar{c}_{min}$ de ces chemins.
   (a) si $\bar{c}_{min} \geq 0$, arrêter, on a atteint la solution optimale
   (b) sinon, ajouter un ou plusieurs chemins de coût réduit négatif à Ω_R et revenir à l'étape 2.

Méthodes de branchement exact

Supposons avoir résolu à l'un des nœuds de l'arbre de branchement, une relaxation linéaire (PM^{rl}) dont la valeur optimale est strictement inférieure à la meilleur solution entière connue jusque-là. On ne peut donc pas éliminer ce nœud. Si de plus la solution optimale de ce nœud ne vérifie pas les contraintes d'intégrité (4.4), plusieurs décisions de branchement peuvent être utilisées afin d'obtenir une solution entière, au terme de l'algorithme de séparation et évaluation. On donne ici les trois méthodes que nous avons utilisées dans l'algorithme exact de génération de colonnes.

1. Brancher sur des variables globales : si, par exemple, le nombre de véhicules partant d'un dépôt $\sum_{p\in\Omega_d} y_p$ n'est pas entier, on peut brancher sur la valeur de cette variable. Cette décision est imposée dans le problème maître.
2. Brancher sur la valeur des inter-tâches : s'il y a dans la solution relaxée un itinéraire qui comprend l'enchaînement (ou *inter-tâche*) des voyages v_1 - v_2 et un autre itinéraire avec l'enchaînement v_1 - v_3, on peut brancher sur ces inter-tâches. On crée alors un nœud fils dans lequel on impose, par exemple, que le voyage v_1 soit immédiatement suivi par v_2 et un autre dans lequel v_1 ne peut pas être suivi par v_2. Ces décisions de branchement sont gérées dans le sous-problème : lors de la construction des chemins dans le graphe, on retient la dernière tâche accomplie. On sait ainsi par quelles tâches on peut ou doit poursuivre le chemin. Les règles de dominance sur les étiquettes prennent en compte ces décisions. Pour les explications précises sur la gestion de ce type de décision, on réfère à Irnich et Desaulniers (2005).
3. Brancher sur la valeur du flot sur les arcs : dans une solution entière la valeur du flot passant sur chaque arc est nécessairement entière. Si ce n'est pas le cas, considérons un arc sur lequel le flot total est fractionnaire et strictement compris, par exemple,

entre 0 et 1. On crée alors deux nœuds fils : dans le premier, un flot nul est imposé sur cet arc, dans le second, un flot supérieur ou égal à 1. Pour imposer un flot nul sur un arc, il suffit de supprimer cet arc du graphe (et toutes les colonnes déjà générées qui contenaient cet arc). Pour imposer un flot égal à 1 sur un arc (i, j), on peut dans certains cas (graphe de connexion) supprimer dans le graphe du sous-problème tous les autres arcs partant du noeud i. Dans d'autres cas (notamment les réseaux espace-temps), cette décision doit être intégrée au problème maître.

S'il est possible brancher selon la première méthode, c'est toujours celle-ci qui sera choisie. Sinon, on utilise l'une des deux autres méthodes, avec une légère préférence pour la méthode numéro 2 (branchement sur les inter-tâches).

4.2 Comparaison numérique des différents réseaux avec l'algorithme exact

En utilisant l'algorithme de génération de colonnes, nous pouvons maintenant comparer l'efficacité des réseaux de génération des itinéraires présentés au chapitre 3. Après avoir décrit les instances utilisées, on étudiera à la section 4.2.2 les performances des réseaux pour le MDVSP classique et à la section 4.2.3 celles des réseaux avec décalage d'horaires.

4.2.1 Description des instances

On utilise des instances de $n = 200, 300$ voyages et un nombre de dépôt fixé $m = 2$. Ces problèmes sont générés à partir d'une instance réelle composée de 1047 voyages. On sélectionne aléatoirement [2] n voyages dans cette liste pour créer une instance de n voyages. On impose de plus un temps maximal sur la longueur d'une connexion lorsqu'on ne rentre pas au dépôt. Cette limite est fixée à 45 min. Le coût d'utilisation quotidien c_f d'un véhicule est fixé à 1000, le coût d'une minute d'attente est $c_a = 0.2$ et celui d'une minute de trajet à vide est $c_v = 0.4$.

On utilise ici la génération de colonnes exacte avec les trois méthodes de branchement décrites précédemment. On s'appuie sur le programme GENCOL (version 4.5) qui fait lui même appel à CPLEX 12.4.

2. On parcours la liste des 1047 voyages ordonnés par ordre croissant d'heure de départ et on accepte ou non avec une certaine probabilité fixe chaque voyage. Il est donc attendu que le nombre de voyages couverts par itinéraire augmente lorsqu'on considère des instances avec plus de voyages.

4.2.2 Réseaux sans décalage d'horaires

On compare cinq réseaux différents pour modéliser le MDVSP :

- **cnx** : le réseau de connexions classique, tel que représenté à la figure 3.1.
- **ret** : le **r**éseau **e**space-**t**emps, tel que représenté à la figure 3.2
- **ret*** : même réseau que **ret** mais on a agrégé les nœuds d'attente au dépôt comme à la figure 3.8 et supprimé les arcs haut-le-pied qui sont plus coûteux qu'un retour au dépôt.
- **mix** : le réseau de connexions mixte, tel que représenté à la figure 3.7.
- **mix*** : même réseau que **mix**, mais les nœuds d'attente au dépôt sont aggrégés comme à la figure 3.8.

Le tableau 4.1 donne les résultats obtenus en moyenne sur 10 instances de 200 et 300 voyages avec l'algorithme exact. Les colonnes des tableaux donnent dans l'ordre : le type de graphe utilisé, la valeur de la meilleure solution entière trouvée (**Entière**), la valeur de la relaxation linéaire (PM^{rl}) (**Relax**) et le nombre de véhicules nécessaires (**bus**). On détaille ensuite les performances en temps de calcul - les temps CPU sont toujours donnés en secondes : le temps de résolution total (**Total**), le temps pour résoudre le premier nœud de l'arbre, i.e. le problème (PM^{rl}) (**Nd0**) et la part du temps CPU total consacrée à résoudre le sous-problème (**S-P**). On donne aussi le nombre de nœuds dans l'arbre de branchement (**Nb. Nds**), le nombre total d'arcs dans les réseaux (**arcs**) et le nombre d'instances résolues à l'optimalité dans le temps imparti (**R**).

Pour tous les tests réalisés ici, on impose une limite de 3 heures de temps de calcul par instance. Dans les cas où l'instance n'est pas résolue, on considère que son temps de calcul est 3h=10800s. Lorsque le nombre d'instances résolues par type de réseau diffère, il faut donc être prudent quant à l'interprétation du temps de calcul.

Remarquons tout d'abord que le nombre d'arcs par réseau correspond bien à la réflexion menée au chapitre 3. Pour les instances de 300 voyages, le réseau espace-temps **ret*** est 19 fois plus petit que le réseau de connexions **cnx**. Le réseau mixte **mix*** est lui 16 fois plus petit. Les temps de calcul - et en particulier le temps consacré au sous-problème - reflètent les différences de taille des réseaux. Les réseaux **ret** et **mix** ont des performances relativement similaires. Les réseaux **mix** et **mix*** semblent donc bien répondre à l'objectif de trouver un réseau efficace comme **ret** mais qui conserve l'adaptabilité du réseau de connexions **cnx**. On remarque de plus que, lorsqu'on utilise le réseau **mix***, le temps de résolution de la première

relaxation linéaire est le plus court ; il en va de même pour le temps consacré à résoudre le sous-problème.

Remarque 2. *Il faut noter ici que les réseaux* ***ret*** *et* ***ret**** *ne modélisent pas exactement le même problème que les 3 autres types de réseaux. En effet, on a interdit dans les réseaux de connexions (mixte) que les connexions hors du dépôt de plus de 45 minutes, en enlevant les arcs correspondants. Si on voulait modéliser la même contrainte dans le réseau espace temps, il faudrait simplement ajouter une ressource de décompte de la durée entre deux trajets. Il n'est pas évident de savoir si cet ajout va ralentir la résolution du problème ou au contraire l'accélérer car l'espace des solutions est plus restreint. On observe donc dans plusieurs cas que la solution de* ***ret*** *est quelques unités plus faible que celle des autres réseaux.*

Remarque 3. *Dans les instances que nous avons créées et utilisées ici, le nombre de véhicules nécessaires est faible par rapport au nombre de voyages. On a ainsi environ 8.5 voyages par véhicules pour les instances de 200 voyages et 12 pour les instances de 300. Cela augmente la dégénérescence des problèmes, et donc leur difficulté de résolution. En comparaison, dans les instances décrites par Carpaneto et al. (1989) et utilisées à de nombreuses reprises dans la littérature, le nombre de voyages par véhicule se situe autour de 5-6.*

Tableau 4.1 Comparaison de 5 réseaux de modélisation du MDVSP. Résultats en moyenne sur 10 instances de 200 et 300 voyages avec un algorithme exact.

	Coût de la solution			**Temps CPU (s)**					
type	**Entière**	**Relax.**	**bus.**	**Total**	**Nd0**	**S-P**	**Nb. Nds**	**arcs**	**R**
200 voyages									
cnx	24279,5	24279,5	23,5	135,6	2,2	65,7	146,6	21186,3	10
mix	24279,5	24279,5	23,5	60,7	1,5	14,6	84,3	2703,4	10
mix*	24279,5	24279,5	23,5	56,0	1,3	8,6	82,6	2050,7	10
ret	24276,4	24276,4	23,5	72,7	1,7	25,8	77,9	2394,7	10
ret*	24276,4	24276,4	23,5	71,3	1,5	16,3	95,8	1718,9	10
300 voyages									
cnx	25712,9	26055,4	22,0	5412,9	22,2	3426,6	332,9	50168,3	9
mix	26055,4	26055,4	24,9	2889,5	9,0	667,3	369,4	4102,1	10
mix*	26055,4	26055,4	24,9	2900,4	8,2	396,6	404,3	3125,3	10
ret	26050,1	26050,1	24,9	2598,9	10,1	804,7	328,2	3609,3	10
ret*	26050,1	26050,1	24,9	2473,1	9,8	539,8	340,7	2597,8	10

4.2.3 Réseaux avec décalage d'horaires

On ajoute maintenant la possibilité de modifier les horaires de départ initiaux. Lorsqu'on autorise le décalage de voyages, on considère que tous les voyages de l'instance peuvent être avancés de 2 minutes, garder le même horaire ou être retardés de 2 minutes. On a donc 3 copies pour chaque voyage, avec des décalages de -2, 0 et 2 minutes.

On compare ici cinq réseaux pour modéliser le MDVSP avec décalage d'horaires :

- **cnx_c3** : le réseau de connexions classique **cnx** dans lequel on a multiplié les nœuds voyages par 3 comme à la figure 3.9.
- **ret*_c3** : le réseau espace-temps **ret*** avec 3 arcs voyages pour chaque voyage, tel que représenté à la figure 3.10
- **mix*_c3** : le réseau mixte **mix*** dans lequel on a multiplié les nœuds voyages par 3 comme à la figure 3.9.
- **src*_c3** : le réseau **mix*** avec **s**ous-**r**éseaux de **c**hoix de copie comme à la figure 3.13.
- **src2*_c3** : réseau **mix*** avec sous-réseaux de choix de copie non contractés, c'est-à-dire que 3.13 est remplacé par 3.10. Ce réseau est a priori moins efficace que **src*_c3** car il ajoute de la symétrie.

La tableau 4.2 donne, de la même manière que le tableau 4.1, les résultats pour les différents graphes avec décalage d'horaires. Remarquons tout d'abord les gains réalisés par rapport au modèle sans décalage : on passe de 23.5 à 22.1 véhicules en moyenne pour les problèmes à 200 voyages et de 24.9 à 23.4 en moyenne pour les problèmes de 300 voyages. On note cependant une forte augmentation du temps de calcul par rapport au tableau 4.1, en particulier pour les problèmes de 200 voyages.

La taille du réseau et les temps de calcul explosent pour le réseau de connexions avec multiplication naïve des nœuds voyages (**cnx_c3**), il est donc inenvisageable d'utiliser ce réseau. Les réseaux avec sous-réseaux de choix de copie **src*_c3** et **src2*_c3** semblent relativement efficaces et même meilleurs que **tsn*_c3**.

Cependant, la résolution par l'algorithme exact de génération de colonnes est relativement lente. De plus, certaines instances ne sont pas résolues, alors que des problèmes de 200-300 sont relativement petits par rapport aux instances réelles (jusqu'à plusieurs milliers de voyages). Au vu de ces résultats, on a donc cherché des stratégies d'accélération de la génération de colonnes. Deux se sont révélées particulièrement efficaces et sont présentées

Tableau 4.2 Comparaison de 5 réseaux de modélisation du MDVSP avec décalage d'horaires. Résultats en moyenne sur 10 instances de 200 et 300 voyages avec un algorithme exact.

	Coût de la solution			Temps CPU (s)			Nb.		
type	**Entière**	**Relax.**	**bus.**	**Total**	**Nd0**	**S-P**	**Nds**	**arcs**	**R**
200 voyages									
cnx_c3	22478,0	22478,0	21,8	2900,5	33,0	2700,5	360,9	185827,1	8
src_c3	22818,3	22806,6	22,1	1135,7	3,0	465,3	1207,7	4536,1	9
src2_c3	22818,3	22806,6	22,1	1182,2	3,5	564,6	836,0	5586,1	9
mix*_c3	22478,0	22478,0	21,8	2285,2	2,5	1173,2	2939,1	12634,1	8
tsn*_c3	22477,0	22477,0	21,8	2196,7	2,7	1553,0	1100,8	4931,5	8
300 voyages									
cnx_c3	309266,1	24394,2	9,6	10800,0	246,6	10371,2	66,5	444219,7	0
src*_c3	24394,2	24394,2	23,4	3884,4	12,3	1062,8	393,2	6519,7	9
src2*_c3	24394,2	24394,2	23,4	3814,0	13,3	1411,0	360,0	8019,7	9
mix*_c3	24394,2	24394,2	23,4	4093,5	14,5	1606,1	320,2	19390,7	9
tsn*_c3	24392,6	24392,5	23,4	4914,0	14,3	2338,8	368,4	7425,3	9

dans les sections 4.3 et 4.4.

4.3 Élimination d'arcs par analyse des coûts réduits

On propose ici une méthode d'accélération de la génération de colonnes qui sera utile dans la version exacte de notre algorithme. Cette méthode consiste à supprimer certains arcs dès lors qu'on peut s'assurer qu'ils ne peuvent pas faire partie d'une solution optimale. On présente à la section 4.3.1 le principe général d'élimination de variables par analyse des coûts réduits, puis deux méthodes d'évaluation des coûts réduits associés aux arcs (voir section 4.3.2). Les tests numériques seront présentés à la section 4.5 et montrent qu'éliminer ainsi des arcs peut réduire le temps de résolution de manière très significative.

4.3.1 Principe d'élimination de variables par analyse des coûts réduits

Cas général

Pour résoudre un problème en nombres entiers, on peut accélérer la résolution exacte de celui-ci, en utilisant la technique suivante de fixation de variables par analyse des coûts réduits. On considère (P) le problème suivant :

$$\min \ c^T x \tag{4.6}$$

$$s.c. \ Ax = b \tag{4.7}$$

$$x \in \mathcal{X} \tag{4.8}$$

On suppose que $\mathcal{X}$ contraint au moins une partie des variables à prendre des valeurs entières. A un certain moment de la résolution, on suppose disposer d'une solution entière x_E et d'une solution duale admissible y de la relaxation linéaire de (P). Alors le gap d'intégrité est borné supérieurement par $c^\top x_E - y^\top b$.

Proposition 1. *Si, après avoir résolu la relaxation linéaire de (P), une variable entière x_i a un coût réduit $\bar{c}_i$ strictement supérieur à $c^\top x_E - y^\top b$, alors, dans toute solution optimale de (P), cette variable est égale à sa borne inférieure (0 pour une variable binaire).*

Cas avec décomposition de Dantzig-Wolfe

On voudrait utiliser cette proposition pour réduire la complexité de notre problème (PM), en particulier en éliminant certains arcs, ce qui revient à fixer le flot sur ceux-ci à 0. Pour cela, il est nécessaire de formuler le problème selon la formulation multi-flot (P) donnée ci-après. Dans celle-ci, on cherche directement à construire des itinéraires valides en optimisant le flot sur chaque arc.

Par souci de simplicité, on présente la formulation multi-flot dans le cas d'un graphe de connexions (voir section 3.2.1) sans décalage d'horaires. Pour les autres réseaux décrits au chapitre 3, les modèles sont très similaires. Pour tout dépôt $d \in \mathcal{D}$, on construit un graphe $\mathcal{G}^d = (\mathcal{N}_d, \mathcal{A}_d)$ dans lequel $\mathcal{N}_d$ comporte un nœud source s, un nœud puits p et un nœud pour chaque voyage v. Les arcs de $\mathcal{A}_d$ représentent des connexions possibles entre deux voyages et les trajets au dépôt. La variable x^d_{ij} indique si un autobus affecté au dépôt d emprunte l'arc (i, j).

On définit le problème (P) ainsi :

$$\min_{x} \sum_{d\in\mathcal{D}} \sum_{(i,j)\in\mathcal{A}_d} c_{ij}^d x_{ij}^d \tag{4.9}$$

$$s.c. \qquad \sum_{d\in\mathcal{D}} \sum_{(i,j)\in\mathcal{A}_d} x_{ij}^d = 1 \qquad \forall i \in \mathcal{N}_d\backslash\{s,p\} \tag{4.10}$$

$$\sum_{(l,i)\in\mathcal{A}_d} x_{li}^d - \sum_{(i,j)\in\mathcal{A}_d} x_{ij}^d = 0 \qquad \forall\ i \in \mathcal{N}_d\backslash\{s,p\}, \forall d \in \mathcal{D} \tag{4.11}$$

$$\sum_{(s,i)\in\mathcal{A}_d} x_{si}^d \leq r_d \qquad \forall d \in \mathcal{D} \tag{4.12}$$

$$x_{ij}^d \in \mathbb{N} \qquad \forall (i,j) \in \mathcal{A}_d, \forall d \in \mathcal{D} \tag{4.13}$$

On minimise le coût des arcs empruntés (4.9), sous contrainte de couvrir exactement une fois chaque voyage (4.10), de respecter la conservation du flot dans chaque graphe (4.11) et la capacité des dépôts (4.11). On peut remarquer que (PM) est le résultat de la décomposition de Dantzig-Wolfe appliquée à (P) qui place les contraintes (4.11) dans le sous-problème.

On note $(\lambda_v)_{v\in\mathcal{V}}$ et $(\mu_d)_{d\in\mathcal{D}}$ les variables duales associées aux contraintes (4.10) et (4.12) respectivement. Enfin $(\pi_i^d)_{i\in\mathcal{N}_d, d\in\mathcal{D}}$ désignera les variables duales associées aux contraintes (4.11). Le problème dual (D) de la relaxation linéaire de (P) s'écrit :

$$\max_{\lambda,\pi,\mu} \sum_i \lambda_i + \sum_d r_d \mu_d \tag{4.14}$$

$$sc. \qquad \lambda_i - \pi_i^d + \pi_j^d \leq c_{ij}^d \qquad \forall (i,j) \in \mathcal{A}_d, i,j \notin \{s,p\}, \forall d \in \mathcal{D} \tag{4.15}$$

$$\lambda_i - \pi_i^d \leq c_{ip}^d \qquad \forall (i,p) \in \mathcal{A}_d, i \neq s, \forall d \in \mathcal{D} \tag{4.16}$$

$$\mu_d + \pi_i^d \leq c_{si}^d \qquad \forall (s,i) \in \mathcal{A}_d, i \neq p, \forall d \in \mathcal{D} \tag{4.17}$$

$$\mu_d \leq 0 \qquad \forall d \in \mathcal{D} \tag{4.18}$$

Dans le cas du problème (P), on peut réécrire la proposition 1. On note x_E une solution entière du problème (P) et y une solution admissible du problème dual (D). Les coûts réduits associés à y sont $\bar{c} = c - A^T y$ où A est la matrice des contraintes de (P).

Proposition 2. *S'il existe un triplet (i,j,d) tel que $\bar{c}_{ij}^d > c^T x_E - y^T b$ alors on peut supprimer la variable x_{ij}^d du problème (P) et donc l'arc (i,j) du graphe $\mathcal{G}^d$.*

On remarque que la résolution de la relaxation linéaire de (PM) (4.1)-(4.4) ne permet pas de connaître directement la valeur des variables duales π_i^d (conservation du flot dans le graphe) et donc d'appliquer directement la proposition 2. Dans la section suivante, on explique deux

méthodes pour calculer les coûts réduits des arcs du graphe sans résoudre le problème (P).

4.3.2 Méthodes mono- et bi-directionnelles d'évaluation des coûts réduits des arcs

On présente maintenant deux méthodes de calcul des coûts réduits des arcs (formellement des variables x_{ij}^d). La première méthode donne un moyen de calculer une solution duale et d'en déduire les coûts réduits. La seconde, que nous utiliserons lors des tests numériques, propose une approche plus générale et plus précise mais également plus coûteuse en temps de calcul.

Calcul d'une solution duale admissible : méthode monodirectionnelle

On suppose avoir résolu (PM^{rl}) à l'optimalité et connaître les valeurs duales λ_i^* et μ_d^* finales associées aux contraintes (4.2) et (4.3). Alors, d'après Hadjar et al. (2006), on peut obtenir la solution duale de (P) selon l'énoncé de la proposition 3.

Proposition 3. *Soit* $\Pi = ((\lambda_i), (\mu_d), (\pi_i^d))$ *le vecteur défini par :*

- $\lambda_i = \lambda_i^*, \ \forall i \in \mathcal{V}$
- $\mu_d = \mu_d^*, \ \forall d \in \mathcal{D}$
- π_i^d *est la longueur du plus court chemin de* s *au nœud* i *dans le graphe* $\mathcal{G}^d$ *(avec les couts réduits de la dernière itération).*

Alors $\Pi = ((\lambda_i), (\mu_d), (\pi_i^d))$ *est une solution duale optimale du problème* (P).

Une preuve de cette proposition est donnée dans Hadjar et al. (2006). Celle-ci démontre que Π satisfait les contraintes de (D).

A partir de la solution duale trouvée Π, on peut évaluer le coût réduit des arcs associé à cette solution duale, soit $d \in \mathcal{D}, i, j \in \mathcal{N}_d \backslash \{s, p\}$

$$\bar{c}_{ij}^d = c_{ij}^d - \lambda_i + \pi_i^d - \pi_j^d \qquad \text{si } (i,j) \in \mathcal{A}_d \tag{4.19}$$

$$\bar{c}_{sj}^d = c_{sj}^d - \mu_d - \pi_j^d \tag{4.20}$$

$$\bar{c}_{ip}^d = c_{ip}^d - \lambda_i + \pi_i^d \tag{4.21}$$

On note dans la suite $\tilde{c}_{ij}^d$ le coût actualisé d'un arc du graphe (ou, de manière équivalente, de combien cet arc contribue au coût réduit d'un itinéraire dans la formulation (PM)). Dans le

cas d'un graphe de connexion, on a :

$$\tilde{c}^d_{ij} = c^d_{ij} - \lambda_i \qquad \text{pour } (i,j) \in \mathcal{A}_d,\ i \neq s \tag{4.22}$$

$$\tilde{c}^d_{sj} = c^d_{sj} - \mu_d \qquad \text{pour } (s,j) \in \mathcal{A}_d \tag{4.23}$$

On pose de plus $\pi^d_s = \pi^d_p = 0$, alors on peut remplacer les équations (4.19) à (4.21) par la formulation générale suivante :

$$\forall d \in \mathcal{D}, (i,j) \in \mathcal{A}_d, \qquad \bar{c}^d_{ij} = \tilde{c}^d_{ij} + \pi^d_i - \pi^d_j \tag{4.24}$$

Analyse globale des coûts réduits : méthode bidirectionnelle

Comme proposé par Irnich et al. (2010), on peut éliminer un plus grand nombre d'arcs en utilisant une méthode d'évaluation bidirectionnelle des coûts réduits. En effet, la méthode précédente est fondée sur la recherche d'une solution duale optimale mais on peut se demander si, en prenant une autre solution duale, on pourrait éliminer d'autres arcs. Irnich et al. (2010) prouvent que le coût réduit maximal $\bar{C}^d_{ij}$ donné ci-dessous est toujours atteint par au moins une solution duale. On peut donc l'utiliser pour éliminer les arcs à partir de la proposition 2.

$$\bar{C}^d_{ij} = \max_{\Pi \in D^*} \bar{c}^d_{ij}(\lambda, \mu, \pi) = \tilde{c}^d_{ij} + l^{d\rightarrow}_i + l^{d\leftarrow}_j - l^{d\rightarrow}_p \tag{4.25}$$

avec :

- $l^{d\rightarrow}_i$ la valeur du plus court chemin de s à i dans le graphe $\mathcal{G}^d$ avec les coûts des arcs actualisés.
- $l^{d\leftarrow}_i$ la valeur du plus court chemin de i à p dans le même graphe.

Pour évaluer $\bar{C}^d_{ij}$, il faut donc résoudre deux problèmes de plus court chemin : le premier depuis la source vers tous les nœuds, le second depuis le puits vers tous les nœuds dans le graphe inversé (on change la direction de tous les arcs). Cette méthode bidirectionnelle est donc plus coûteuse en temps de calcul que la première mais elle assure la suppression de tous les arcs qui peuvent être éliminés par analyse des coûts réduits pour un gap donné. Les tests réalisés par Groiez et al. (2013) sur des instances du MDVSP montrent que le temps de calcul est significativement réduit par cette méthode.

4.3.3 Algorithme général d'élimination d'arcs

On donne ici les étapes de l'algorithme d'élimination d'arcs après la résolution de (PM^{rl}). On peut réaliser l'élimination d'arcs après chaque nœud de l'arbre de branchement ou seulement à certains nœuds.

Méthode :

1. Récupérer les variables duales $(\lambda^*,\ \mu^*)$ associées aux contraintes (4.2) et (4.3)
2. Calculer les coûts réduits associés aux arcs par l'une des deux méthodes présentées
3. Chercher, si nécessaire, une solution entière x_E de coût $z_E = c^T x_E$ et déterminer le gap d'intégrité courant : $z_E - c^T x_{LP}$ avec x_{LP} la solution optimale de la relaxation linéaire.
4. Éliminer les arcs qui ont un coût réduit dépassant strictement le gap
5. Éliminer les itinéraires (ou colonnes) de Ω_R déjà générés contenant ces arcs.

Plutôt que de chercher heuristiquement une solution entière x_E, on peut aussi supposer la valeur optimale du problème à partir de la valeur de la relaxation linéaire $c^\top x_{LP}$. Par exemple, on peut prendre $z_E = c^\top x_{LP} + \delta$ avec $\delta \geq 0$ supérieur au gap d'intégrité supposé. On se place ensuite dans l'un des cas suivants :

— Si l'algorithme ne trouve pas de solution, alors cette hypothèse était fausse et on doit recommencer en augmentant δ.

— Si on trouve une valeur optimale $z_E^* \leq z_E$ alors l'hypothèse était valide et on a atteint une solution optimale.

— Si on trouve une solution optimale telle que $z_E^* > z_E$ alors l'hypothèse était fausse, la solution optimale se trouve dans $]z_E, z_E^*]$ et il faut recommencer en augmentant δ (ou utiliser la valeur z_E^* trouvée).

4.4 Méthode de résolution heuristique par génération de colonnes

La méthode d'élimination d'arcs précédente permet d'accélérer l'algorithme sans perte d'optimalité. On verra à la section 4.5 que celle-ci diminue le temps de calcul de manière importante. Pour accélérer davantage l'algorithme et en faire une version heuristique, on propose ici de combiner deux stratégies, également basées sur la génération de colonnes. La première consiste à ne créer qu'un seul nœud fils dans l'arbre de branchement, ce qui réduit largement le nombre de relaxations linéaires à résoudre. La seconde diminue la dégénérescence du problème par perturbation des contraintes de partitionnement.

4.4.1 Branchement par fixation de colonnes

Au lieu d'utiliser les décisions de branchement exactes évoquées à la section 4.1.2, on propose ici d'utiliser une méthode de branchement heuristique. Celle-ci consiste à fixer définitivement un itinéraire à chaque nœud de l'arbre. Pour ce faire, après avoir résolu une relaxation linéaire dont la solution est fractionnaire, on évalue les variables y_p, c'est-à-dire le poids de l'itinéraire p dans la solution finale. On choisit l'itinéraire p pour lequel $y_p \in]0,1[$ est maximal et on fixe définitivement sa valeur à 1. À chaque nœud, on crée ainsi un seul fils. Cette technique ne garantit plus d'atteindre l'optimalité car le nœud frère $y_p = 0$ n'est jamais évalué (et ce serait algorithmiquement difficile de maintenir des décisions de la sorte dans l'arbre).

Cette technique de branchement par fixation des colonnes évite l'exploration d'un arbre de branchement large et limite ainsi le nombre de relaxations linéaires à résoudre. Dans le pire des cas ici, on résout autant de relaxations linéaires que de véhicules nécessaires, soit quelques dizaines de nœuds. En ce qui concerne la qualité des solutions, on verra que, dans de très nombreux cas, on parvient quand même à la valeur optimale ou on s'en éloigne très peu.

Il faut noter, cependant, que cette technique de fixation de colonnes peut conduire à un problème infaisable. En effet, il se peut qu'à un certain niveau de l'arbre de branchement, on ait fixé un ensemble de colonnes tel que les contraintes du problème maître ne puissent plus être satisfaites. Dans le cas d'un MDVSP, il se pourrait qu'on ne puisse plus satisfaire la contrainte de capacité du dépôt (4.3) après avoir fixé un certain nombre d'itinéraires. Cependant, dans les tests réalisés, cela ne s'est jamais produit.

4.4.2 Réduction de la dégénérescence par perturbation des contraintes de partitionnement

Les problèmes de MDVSP sont souvent très *dégénérés*, c'est-à-dire qu'il existe de nombreuses bases différentes qui conduisent à la même solution. Cela ralentit chaque résolution du (PMR^{rl}) et l'ajout de nouvelles colonnes (itinéraires) peut ne pas contribuer à améliorer la solution.

Oukil et al. (2007), qui résolvent le MDVSP eux aussi avec un algorithme de génération de colonnes, montrent l'accroissement du temps de calcul avec la *densité* des itinéraires, c'est-à-dire le nombre moyen de voyages par véhicule. En effet, plus le nombre de véhicules nécessaires est petit, plus le nombre de variables y_p non nulles est petit. À nombre de tâches

- donc de contraintes - égal, le problème est alors plus dégénéré. Oukil et al. (2007) utilisent une méthode de génération de colonnes *stabilisée*, i.e., ils contrôlent itérativement les valeurs des variables duales. Une étude qui combine cette méthode à l'agrégation dynamique de contraintes, un autre moyen de réduire la dégénérescence, a été réalisée par Benchimol et al. (2012) pour les problèmes de partitionnement d'ensembles, et donc en particulier le MDVSP.

Nous souhaitons que l'heuristique avec branchement par fixation de colonne, puisse résoudre rapidement des problèmes avec un grand nombre de voyages. Dans ces instances, la dégénérescence devient particulièrement problématique et on propose d'aborder ce problème par une technique de perturbation fixe des contraintes, plus simple que la génération de colonnes stabilisée.

Pour diminuer la dégénérescence des instances, on perturbe légèrement les contraintes de partitionnement des voyages (4.2). On autorise ainsi à sur-/sous-couvrir un peu certaines tâches. Pour tout voyage $v \in \mathcal{V}$, on introduit des variables d'écart z_v^+, z_v^-. Ces variables appartiennent à $[0, \epsilon_v^+]$ et $[0, \epsilon_v^-]$ où $\epsilon_v^+, \epsilon_v^-$ sont aléatoirement choisis dans un petit intervalle. On pénalise dans l'objectif les variables z_v^+, z_v^- par δ_v^+ et δ_v^-.

Le problème maître perturbé (PM_{pert}) s'écrit donc :

$$\min_{y,z} \sum_{d\in\mathcal{D}} \sum_{p\in\Omega^d} c_p y_p + \sum_{v\in\mathcal{V}} (\delta_v^+ z_v^+ + \delta_v^- z_v^-) \tag{4.26}$$

$$s.c. \quad \sum_{d\in\mathcal{D}} \sum_{p\in\Omega^d} a_{vp} y_p + z_v^+ - z_v^- = 1 \qquad \forall v \in \mathcal{V} \tag{4.27}$$

$$\sum_{p\in\Omega^d} y_p \leq r_d \qquad \forall d \in \mathcal{D} \tag{4.28}$$

$$0 \leq z_v^+ \leq \epsilon_v^+ \qquad \forall v \in \mathcal{V} \tag{4.29}$$

$$0 \leq z_v^- \leq \epsilon_v^- \qquad \forall v \in \mathcal{V} \tag{4.30}$$

$$y_p \in \{0,1\} \qquad \forall p \in \Omega^d,\ \forall d \in \mathcal{D} \tag{4.31}$$

Dans la version heuristique de notre algorithme, on résout (PM_{pert}^{rl}) jusqu'à un certain stade dans l'arbre de branchement, où l'on revient définitivement au problème (PM^{rl}). Ce changement s'effectue lorsque la méthode de fixation des colonnes est jugée trop risquée, c'est-à-dire si la meilleure colonne (itinéraire) a une valeur fractionnaire en dessous d'un certain seuil.

Lors des tests numériques réalisés, on a remarqué que les perturbations étaient en général enlevées dans l'un des deux ou trois derniers nœuds de l'arbre de branchement.

4.5 Comparaison numérique des différentes stratégies exactes et heuristiques

Nous avons testé numériquement les stratégies d'accélération présentées aux sections 4.3 et 4.4 et nous proposons de comparer les résultats obtenus avec ceux présentés à la section 4.2. On analyse d'abord les performances de l'algorithme heuristique puis celles de l'élimination d'arcs. On discutera finalement de la possibilité de combiner ces deux méthodes.

4.5.1 Algorithme heuristique

On analyse maintenant les performances de notre version heuristique de l'algorithme de génération de colonnes. Cette heuristique applique les deux stratégies décrites précédemment, à savoir :

- Le branchement par fixation de colonne : à chaque nœud de l'arbre de branchement, on fixe exactement une colonne (ou itinéraire), on choisit l'une de celles dont la valeur fractionnaire est maximale. C'est la seule méthode de branchement utilisée.
- La perturbation des contraintes de partitionnement : à chaque nœud, on résout (PM^{rl}_{pert}) au lieu de (PM^{rl}) avec ϵ^+_v, et ϵ^-_v aléatoirement choisis dans l'intervalle $[0, 0.01]$. On prend $\delta^+_v = \delta^-_v = 1$. La gestion des perturbations est implémentée dans GENCOL comme une méthode de branchement, ainsi, après chaque relaxation linéaire résolue dans l'arbre, on évalue la possibilité de retirer les perturbations. Lorsque la valeur de la meilleure colonne fixable est en dessous d'un certain seuil, on enlève toutes les perturbations et on résout (PM^{rl})

On donne aux tableaux 4.3 et 4.4 les résultats obtenus avec cette heuristique, présentés de la même manière qu'aux tableaux 4.1 et 4.2. On a analysé cette fois-ci des problèmes de 200, 300 et 500 voyages.

Les temps de résolution sont réduits de façon très importante entre les versions exacte et heuristique de l'algorithme. Pour les problèmes de 300 voyages, la résolution des réseaux **mix*** et **src*_c3** est en moyenne plus de 200 fois plus rapide avec l'heuristique. En particulier, on passe de 404 nœuds dans l'arbre de branchement à 26 pour le réseau **mix***, soit environ 15 fois moins. Cette réduction est essentiellement due à la nouvelle méthode de branchement. De plus, étant donné qu'on fixe un itinéraire à chaque nœud, la taille du problème dans les

Tableau 4.3 Comparaison de 5 réseaux de modélisation du MDVSP. Résultats en moyenne sur 10 instances de 200, 300 et 500 voyages avec l'algorithme heuristique.

	Coût de la solution			**Temps CPU (s)**					
type	**Entière**	**Relax.**	**bus.**	**Total**	**Nd0**	**S-P**	**Nb. Nds**	**arcs**	**R**
200 voyages									
cnx	24279.5	24165.5	23.5	2.9	2.2	1.2	24.5	21186.3	10
mix	24279.5	24165.5	23.5	2.0	1.4	0.4	24.0	2703.4	10
mix*	24279.5	24165.5	23.5	1.9	1.4	0.2	24.5	2050.7	10
tsn	24276.4	24162.5	23.5	2.4	1.6	0.7	24.6	2394.7	10
tsn*	24276.4	24162.5	23.5	2.2	1.5	0.5	24.7	1718.9	10
300 voyages									
cnx	26055,4	25942,1	24,9	25,1	17,4	12,6	26,5	50168,3	10
mix	26055,4	25942,1	24,9	14,5	8,9	1,8	25,8	4102,1	10
mix*	26055,4	25942,1	24,9	13,0	8,6	0,9	26,2	3125,3	10
ret	26050,1	25936,9	24,9	16,8	9,5	3,0	26,4	3609,3	10
ret*	26050,1	25936,9	24,9	14,4	9,2	1,8	26,0	2597,8	10
500 voyages									
cnx	35797,5	35641,5	34,4	205,2	127,6	111,6	35,2	138304,7	10
mix	35797,5	35641,5	34,4	102,7	59,9	7,4	34,9	8579,4	10
mix*	35797,5	35641,5	34,4	96,6	58,7	3,8	35,2	6949,3	10
ret	35796,9	35640,8	34,4	111,8	63,5	11,4	34,9	6084,0	10
ret*	35796,8	35640,8	34,4	105,1	62,5	7,4	34,6	4438,9	10

nœuds subséquents est réduite du nombre de voyages couverts par cet itinéraire. Ainsi, la complexité des relaxations linéaires à résoudre décroit rapidement.

La valeur de la première relaxation linéaire donnée est ici (PM^{rl}_{pert}) et donc il est normal que la différence avec la meilleure valeur entière trouvée soit importante. En revanche, en comparant avec les valeurs obtenues à la section 4.2 pour les instances de 300 voyages, on voit que les valeurs trouvées pour le MDVSP classique sont en fait optimales. Pour le MDVSP avec décalage d'horaires, elles sont en moyenne à 0.1 unités de l'optimum (pour le réseau **src*_c3**).

Cet algorithme heuristique donne donc des solutions quasi-optimales avec des temps de calcul fortement réduits. On est en particulier assuré que le nombre de nœuds dans l'arbre est

limité par le nombre de véhicules nécessaires et donc d'obtenir une solution en un temps maîtrisé. Cependant, il n'est pas exclu que certaines fixations de colonnes soient de mauvais choix et conduisent à des solutions éloignées de l'optimum. Par exemple, pour les instances de 500 voyages, on peut remarquer que le réseau **src*_c3** donne 0.1 autobus de plus que les autres réseaux. En analysant le détail des résultats, on s'est rendu compte que la solution d'une des instances contient un autobus de plus que celles obtenues avec les autres réseaux. Dans certains cas, il est donc possible qu'on s'éloigne significativement de l'optimum.

Si le nombre de véhicules disponibles dans chaque dépôt est très proche du nombre de véhicules nécessaires, autrement dit si les contraintes (4.3) sont serrées, il est théoriquement possible que, suite à de "mauvais" choix de colonnes fixées, l'algorithme termine sans aucune solution. Dans les instances traitées, le nombre d'autobus disponibles à chaque dépôt est suffisamment large et ce cas ne s'est pas présenté.

Enfin, on analyse l'effet de la perturbation des contraintes dans notre algorithme heuristique. On compare au tableau 4.5 les temps de calcul avec et sans perturbations des contraintes pour des problèmes de 500 voyages modélisés à l'aide du réseau **src*_3c**. Le temps de calcul est donc réduit en moyenne d'un facteur 4 pour le premier nœud et 8 pour le temps CPU total. On peut remarquer qu'on a pourtant plus de nœuds dans le cas avec perturbations, ce qui peut s'expliquer par le fait que la solution reste longtemps perturbée et donc, il faut résoudre plus de nœuds avant de trouver une solution entière. Excepté le fait que, dans une instance, on ait du ajouter un autobus (donc $\frac{1000}{10}$ dans la valeur moyenne de la solution), les valeurs des solutions trouvées avec et sans perturbations sont quasiment identiques. La perturbation des contraintes joue donc un rôle clé pour notre heuristique.

4.5.2 Méthode d'élimination de variable

Dans cette section, on illustre l'efficacité de la méthode d'élimination d'arcs présentée en 4.3. On a ici utilisé la méthode bidirectionnelle, i.e. la plus précise. L'objectif est d'accélérer l'algorithme exact de génération de colonnes, en gardant l'assurance d'obtenir une solution entière.

On a vu à la section 4.2 que le gap d'intégrité, i.e. la différence entre la valeur optimale et la valeur de la première relaxation linéaire, était très souvent nul ou proche de zéro. On utilise donc cette caractéristique pour éliminer très rapidement les arcs : au lieu de chercher une borne heuristique z_E, on fait une hypothèse sur le gap δ entre la valeur de la relaxation et celle du problème en nombres entiers comme décrit à la section 4.3.3.

Tableau 4.4 Comparaison de 5 réseaux de modélisation du MDVSP avec décalage d'horaires. Résultats en moyenne sur 10 instances de 200, 300 et 500 voyages avec l'algorithme heuristique.

	Coût de la solution			**Temps CPU (s)**			**Nb.**		
type	**Entière**	**Relax.**	**bus.**	**Total**	**Nd0**	**S-P**	**Nds**	**arcs**	**R**
200 voyages									
cnx_c3	22478.1	22371.6	21.8	29.2	22.7	27.2	22.4	21186.3	10
src*_c3	22478.1	22371.6	21.8	2.8	2.0	0.9	22.7	2703.4	10
src2*_c3	22478.0	22371.6	21.8	3.4	2.2	1.4	22.7	2050.7	10
mix*_c3	22478.1	22371.6	21.8	3.0	2.2	1.1	22.8	2394.7	10
tsn*_c3	22477.0	22370.7	21.8	3.6	2.4	1.5	22.8	1718.9	10
300 voyages									
cnx_c3	24394,3	24289,1	23,4	217,5	136,3	202,7	24,5	444219,7	10
src*_c3	24394,3	24289,1	23,4	18,2	11,1	3,6	24,3	6519,7	10
src2*_c3	24394,3	24289,1	23,4	20,8	11,9	5,6	24,5	8019,7	10
mix*_c3	24394,5	24289,1	23,4	18,3	11,7	4,0	25,0	19390,7	10
ret*_c3	24392,7	24287,5	23,4	22,2	12,2	6,1	24,5	7425,3	10
500 voyages									
cnx_c3	32945,4	32799,7	31,8	1249,4	675,8	1139,5	31,2	1232535,7	10
src*_c3	33045,8	32799,7	31,9	133,3	74,6	16,9	32,5	12560,8	10
src2*_c3	32945,2	32799,7	31,8	130,3	76,9	22,8	31,8	15060,8	10
mix*_c3	33047,2	32799,7	31,9	133,8	84,0	26,3	32,6	47869,3	10
ret*_c3	32944,8	32799,4	31,8	143,5	81,1	26,3	32,0	12647,1	10

Au cours des tests, on a remarqué que c'est après le premier nœud que la grande majorité des arcs sont éliminés. Il est apparu cependant que le fait d'essayer d'éliminer des arcs après chaque nœud de l'arbre de branchement était significativement plus efficace que de réaliser l'élimination seulement après le premier nœud. Dans les résultats présentés, on a donc toujours appliqué l'élimination d'arcs après chaque relaxation linéaire.

Aux tableaux 4.6 et 4.7, on étudie l'influence de la méthode d'élimination d'arcs sur des instances de 300 et 500 voyages. Pour cela, on compare les résultats obtenus avec l'algorithme exact de génération de colonnes (noté dans les tableaux **exact**), ceux obtenus lorsqu'on ajoute l'élimination d'arcs avec δ très petit (0.01 ou un peu plus si on ne trouve pas de solution) (**exact_elim**). Enfin, on prend une valeur de δ plus large (en général $\delta = 2$) dans le test

Tableau 4.5 Résolution heuristique avec et sans perturbation. Résultats en moyenne sur 10 instances de 500 voyages.

type	Coût de la solution Entière	Relax.	bus.	Temps CPU (s) Total	Nd0	S-P	Nb. Nds
500 voyages - réseau src*_c3							
Sans perturbations	32945.6	32944.8	31.8	1087.8	321.9	135.9	25.5
Avec perturbations	33045.8	32799.7	31.9	133.3	74.6	16.9	32.5

noté **exact_elimL**.

Dans ces tableaux, on donne la valeur optimale entière obtenue et celle de la première relaxation linéaire. On donne ensuite la valeur entière supposée, c'est-à-dire la borne $z_{LP} + \delta$. On reporte aussi le pourcentage d'arcs éliminés après la première relaxation linéaire et les temps CPU consacrés, dans l'ordre, à l'élimination d'arcs, au sous-problème et total.

Tableau 4.6 Résultats avec élimination d'arcs par analyse des coûts réduits ; en moyenne sur 5 tests de 300 voyages.

	Coût Solution Ent.	Relax.	arcs $z_{LP} + \delta$	élim (Nd0)	Temps CPU(s) élim.	S-P	Total
sans décalage (**mix***)							
exact	25745,1	25745,1				149,7	**2450,5**
exact_elim	25745,1	25745,1	25745,1	69,3%	5,5	5,1	**26,7**
exact_elimL	25745,1	25745,1	25746,6	62,2%	6,2	20,3	**176,3**
avec décalage (**src*_3c**)							
exact	24382	24381,9				213,0	**1473,7**
exact_elim	24382	24381,9	24382,5	72,6%	6,9	27,0	**128,8**
exact_elimL	24382	24381,9	24384,2	57,1%	12,1	110,3	**487,6**

Pour les problèmes de 500 voyages du tableau 4.7, on a ajouté la colonne **R** qui donne le nombre d'instances résolues (parmi 4) dans le temps imparti (2h=7200s ici).

Lorsqu'on prend la meilleure borne possible (en général $\delta = 0.01$), c'est-à-dire l'algorithme **exact_elim**, on peut donc éliminer environ de 70% des arcs dès le premier noeud. Cette élimination a un impact très intéressant sur le temps de calcul, puisque celui-ci est divisé par

Tableau 4.7 Résultats avec élimination d'arcs par analyse des coûts réduits ; en moyenne sur 4 tests de 500 voyages.

	Coût Solution		Coût Ent.	N0 arcs	Temps CPU(s)			
	Ent.	Relax.	supposé	élim	élim.	S-P	Total	R
sans décalage (**mix***)								
exact	40766,8	36196,9				192,8	**7200,0**	0
exact_elim	36196,9	36196,9	36196,9	75,4%	29,0	20,0	**198,7**	4
exact_elimL	36196,9	36196,9	36198,7	64,0%	55,7	427,7	**4520,0**	4
avec décalage (**src*_3c**)								
exact	39730,8	33433,4				510,0	**7200,0**	0
exact_elim	33179,2	33433,4	33434,0	74,1%	217,9	286,1	**1392,2**	3
exact_elimL	34197,8	33433,4	33435,7	60,8%	278,6	979,4	**6576,5**	1

un facteur 10 environ pour les problèmes de 300 voyages.

Le test **exact_elimL** montre que si on augmente (même faiblement) la valeur de la meilleure solution entière connue (ou supposée), les performances sont bien moins bonnes. L'algorithme avec élimination d'arcs est donc particulièrement adapté pour les instances ayant un très petit gap d'intégrité ou si l'on est capable d'avoir de très bonnes solutions heuristiquement.

4.5.3 Intégration de l'élimination d'arcs à la version heuristique de l'algorithme

Dans les tests réalisés aux deux sections précédentes, nous avons reporté une réduction importante du temps de calcul, que ce soit pour la méthode exacte d'élimination d'arcs ou pour la méthode de branchement par fixation de colonne conjuguée aux perturbations des contraintes. On peut donc se demander s'il est possible de réduire encore davantage le temps de calcul en combinant ces deux méthodes.

On voudrait donc diminuer le temps de calcul de l'heuristique utilisée à la section 4.5.1 en éliminant des arcs au fur et à mesure. Cependant, on fait face à deux obstacles.

Premièrement, lorsqu'on perturbe le problème, on obtient la valeur optimale de (PM^{rl}_{pert}) qui est en général assez éloignée de la solution optimale. Le gap entre z_E et z^{pert}_{LP} est grand et donc il permet d'éliminer peu d'arcs. Les arcs éliminés sont peu susceptibles d'être choisis et donc cela n'améliore pas le temps de résolution. Plutôt que de chercher une solution entière

ou de l'estimer, on peut aussi directement éliminer tous les arcs dont le coût réduit dépasse un certain seuil δ. Cela revient à prendre une borne $z_E = z_{LP}^{pert} + \delta$ même si celle-ci est plus petite que la solution entière attendue.

Ensuite, éliminer des arcs alors qu'on utilise la méthode de fixation de colonne est risqué. En effet, lorsqu'on fixe des colonnes, c'est une décision incertaine et irréversible. Si le choix de colonne n'est pas optimal, l'algorithme peut cependant trouver une solution très proche de l'optimum en utilisant des arc non-optimaux [3]. Or l'élimination d'arcs réduit l'espace des solutions et peut, en particulier, éliminer des arcs non-optimaux qui seraient pertinents dans l'algorithme heuristique. De ce fait, un algorithme qui combine la fixation de colonne à l'élimination d'arcs a des chances de terminer sans aucune solution ou avec une solution très éloignée de l'optimum.

Le problème est donc le suivant : pour que l'élimination d'arcs soit efficace, il faut éliminer le plus d'arcs possibles mais si on en élimine trop, la méthode de fixation des colonnes devient risquée. On a cependant essayé plusieurs techniques afin de trouver une combinaison intéressante entre l'algorithme heuristique et l'élimination d'arcs, en faisant en particulier varier la valeur de δ utilisée.

On donne au tableau 4.8 les résultats obtenus avec les algorithmes suivants :

- **exact** : algorithme exact de génération de colonnes
- **exact_elim** : **exact** avec l'élimination d'arcs
- **heur_cfix** : algorithme heurisitque de génération de colonnes avec fixation d'une colonne à chaque noeud
- **heur_cfix_elim** : **heur_cfix** avec élimination des arcs
- **heur_cfix_pert** : **heur_cfix** avec perturbation des contraintes.
- **heur_cfix_pert_elim** : on a ajouté l'élimination des arcs à l'heuristique **heur_cfix_pert**. On élimine tous les arcs dont le coût réduit dépasse un certain seuil δ.

On voudrait donc trouver un algorithme plus efficace que notre heuristique **heur_cfix_pert**. Le tableau 4.8 détaille le coût de la solution entière trouvée, la valeur de la première relaxation, le temps CPU total en secondes, celui consacré à résoudre le nœud 0, le pourcentage d'arcs

3. On entend par arcs non-optimaux, des arcs qui ne se situent dans aucune des solutions optimales entières.

éliminés après la première relaxation et le nombre d'instances résolues (sur 5).

Tableau 4.8 Comparaison de 6 algorithmes. Résultats en moyenne sur 5 instances de 300 voyages.

	Coût de la solution		**Temps CPU(s)**		**Nd0 arcs**	
	Entière	**Relax.**	**Total**	**Nd0**	**élim.**	**R**
exact	24382,0	24381,9	1481,5	22,2		5
exact_elim	24382,0	24381,9	146,6	22,0	72%	5
heur_cfix	24382,1	24381,9	55,7	22,2		5
heur_cfix_elim	24382,5	24381,9	26,2	22,2	72%	5
heur_cfix_pert	24382,0	24271,2	16,6	10,6		5
heur_cfix_elim_pert	24470,6	24271,2	15,2	11,5	65%	4

L'algorithme qui combine l'heuristique à l'élimination d'arcs **heur_cfix_elim_pert** est donc très légèrement plus rapide que l'heuristique **heur_cfix_pert** mais l'une des instances n'est pas résolue. Différents tests additionnels (entre autres en interdisant d'éliminer certains types d'arc) ont été réalisés mais sans succès. Par conséquent, cette approche n'a pas été conservée.

CHAPITRE 5 CONTRÔLE DE LA QUALITÉ DES HORAIRES DE DÉPART ET EXTENSIONS

Nous avons exploré dans les deux chapitres précédents différents réseaux et algorithmes pour résoudre efficacement le MDVSP avec et sans décalage d'horaires. La question se pose ensuite de savoir combien de voyages il est nécessaire de décaler et si ces décalages diminuent la qualité de la grille horaire initiale. La section 5.1 propose une méthode de contrôle de la qualité des horaires qui n'augmente pas la complexité des algorithmes. Cette méthode met en lumière certaines des propriétés du sous-réseau de choix de copie introduit à la section 3.3.2. On présente ensuite les résultats des tests effectués avec cette méthode. On analyse également l'influence du nombre de copies permises pour chaque voyage. Enfin, on propose à la section 5.3 un moyen de contrôler le nombre de lignes empruntées par itinéraire.

5.1 Contrôle de la qualité des horaires de départ

La *grille horaire* initiale, ou l'ensemble des horaires de départ initialement prévus, est supposée être de qualité satisfaisante. La qualité des horaires de départ peut être définie par différents critères, par exemple, son adéquation avec les trajets des utilisateurs, les espacements réguliers entre les trajets d'une même ligne, la pertinence des correspondances réalisables,... On voudrait donc que le décalage d'horaires introduit ne réduise pas, ou le moins possible, la qualité de la grille horaire.

Dans la suite, nous nous appuyons sur les trois critères suivants pour évaluer la qualité des horaires de départ :

- le nombre de voyages décalés par rapport à l'horaire initial. On veut en effet éviter de modifier l'heure de départ de nombreux voyages.
- l'espacement entre les voyages consécutifs d'une même ligne. En général, les voyages d'une même ligne respectent une fréquence constante durant chaque période de la journée (heures creuses/heures de pointe). Dans la mesure du possible, on souhaite donc que les espacements initiaux entre les trajets d'une même ligne soient préservés.
- la qualité des correspondances entre certains voyages. La qualité d'un réseau d'autobus repose aussi sur la facilité de connexion entre plusieurs lignes. On souhaite que la correspondance soit possible et pratique pour les utilisateurs entre certaines paires de voyages.

Il appartient au décideur (compagnie d'autobus, commune,...) d'établir l'importance qu'il accorde à chacun de ces critères. Le modèle que nous avons développé est relativement large et pourrait prendre en compte d'autres critères sur la qualité des horaires.

Cette section est structurée de la manière suivante : on présente d'abord le modèle de contrôle des horaires, puis un algorithme en deux phases pour le résoudre. On donne pour finir deux idées de prolongement de l'algorithme en deux phases.

5.1.1 Modélisation du problème de contrôle de la qualité des horaires de départ

On propose ici un modèle de contrôle de la qualité des horaires selon les trois critères énoncés précédemment. Dans un souci de lisibilité des modèles, on présente d'abord un modèle de contrôle des deux premiers critères puis l'extension de ce modèle pour prendre en compte le respect des correspondances.

Contrôle du nombre de voyages décalés et des espacements entre les voyages consécutifs

Soit $v \in \mathcal{V}$, l'ensemble des copies M_v peut être représenté par la liste des écarts croissants à l'heure de départ de la copie initiale (en minutes), par exemple $\{-2, 0, 2\}$. Afin de modéliser facilement le problème de contrôle des horaires de départ, on voudrait représenter l'ensemble des décalages permis par une liste d'entiers consécutifs. On notera cette nouvelle représentation M'_v. Pour ce faire, on définit δ_t, le pas de temps auquel on s'autorise à choisir les horaires. δ_t doit être choisi tel que tous les écarts entre les copies de M_v soient des multiples de δ_t. Dans l' exemple $M_v = \{-2, 0, 2\}$, on peut prendre $\delta_t = 1$ si on autorise les décalages de 1 minute ou, $\delta_t = 2$ si on autorise seulement les décalages de deux minutes, c'est-à-dire ceux correspondant aux copies. On permet ainsi d'adopter une discrétisation temporelle plus fine dans le choix des horaires que celle prise pour définir les copies de M_v. On verra dans la section 5.1.2 comment les graphes avec sous-réseaux de choix de copie supportent ces deux discrétisations.

M'_v est alors défini comme l'ensemble des décalages permis en unités de temps δ_t minutes. Cela assure que M'_v est une liste d'entiers consécutifs. Par exemple, toujours pour des copies décalées de $-2,\ 0$ ou 2 minutes, on aura $M'_v = \{-2, -1, 0, 1, 2\}$ si on a $\delta_t = 1$ et $M'_v = \{-1, 0, 1\}$ si on choisit $\delta_t = 2$. On définit ensuite la variable $x_v \in M'_v$ égale au décalage choisi pour le

voyage v.

Si on note lb_v et $ub_v \in M'_v$ les valeurs du décalage minimal et maximal permis pour v, alors on peut représenter linéairement la contrainte de domaine $x_v \in M'_v$:

$$x_v \in M'_v \Leftrightarrow \begin{cases} lb_v \leq x_v \leq ub_v \\ x_v \in \mathbb{N} \end{cases} \tag{5.1}$$

Ainsi, si le voyage n'est pas décalé, on a $x_v = 0$. On propose un modèle linéaire qui pénalise les écarts à la copie initiale en norme 1. Pour cela, on introduit, pour chaque voyage $v \in \mathcal{V}$, des variables d'écart s_v^+, s_v^- pénalisées avec un coût α_{dec} dans l'objectif.

On définit $\mathcal{E}$ l'ensemble des paires de voyages consécutifs pour lesquelles on veut contrôler l'espacement. Si l'espacement entre deux voyages $(v, v') \in \mathcal{E}$ n'est pas modifié, alors on aura $x_v - x'_v = 0$. On introduit $p_{vv'}^+$ et $p_{vv'}^-$ les variables d'écart à l'espacement optimal en norme 1. Ces variables sont pénalisées par α_{esp} dans l'objectif.

On définit le modèle de contrôle des horaires de départ (Q) ainsi :

$$\min \ \alpha_{dec} \sum_{v \in \mathcal{V}} (s_v^+ + s_v^-) + \alpha_{esp} \sum_{(v,v') \in \mathcal{E}} (p_{vv'}^+ + p_{vv'}^-) \tag{5.2}$$

Sous les contraintes

$$lb_v \leq x_v \qquad \forall v \in \mathcal{V} \tag{5.3}$$
$$x_v \leq ub_v \qquad \forall v \in \mathcal{V} \tag{5.4}$$
$$x_v = s_v^+ - s_v^- \qquad \forall v \in \mathcal{V}, \tag{5.5}$$
$$x_v - x_{v'} = p_{vv'}^+ - p_{vv}^- \qquad \forall (v, v') \in \mathcal{E} \tag{5.6}$$
$$x_v \in \mathbb{N} \qquad \forall v \in \mathcal{V} \tag{5.7}$$
$$p_{vv'}^-, \ p_{vv'}^+ \geq 0 \qquad \forall (v, v') \in \mathcal{E} \tag{5.8}$$
$$s_v^-, \ s_v^+ \geq 0 \qquad \forall v \in \mathcal{V} \tag{5.9}$$

Le modèle (Q) pénalise le décalage des voyages et les écarts à l'espacement optimal (5.2) sous les contraintes de choix de copie (5.3) et (5.4). Les équations (5.5) définissent les écarts à la copie initiale et (5.6) les écarts à l'espacement optimal entre deux voyages consécutifs.

La solution optimale du problème (Q) est évidemment $x_v = 0,\ \forall v \in \mathcal{V}$ et de valeur optimale nulle. Dans la section 5.1.2, on modifiera les contraintes (5.3) et (5.4) en fonction de la solution du MDVSP avec décalage d'horaires. On continuera d'appeler ce problème (Q).

Remarque 4. *On a supposé dans le modèle (Q) que, dans la grille horaire initiale, les espacements entre les autobus d'une même ligne étaient optimaux. Si ce n'est pas le cas, on peut simplement ajouter un paramètre donnant l'espacement optimal. Soit $(v, v') \in \mathcal{E}$, une paire de voyages dont on contrôle l'espacement. On note $e^*_{vv'}$ l'écart optimal entre v et v' en unités de temps δ_t. Les contraintes* (5.6) *deviennent alors*

$$x_v - x_{v'} - e^*_{vv'} = p^+_{vv'} - p^-_{vv} \quad \forall (v, v') \in \mathcal{V}$$

Le paramètre $e^*_{vv'}$ est donc le nombre d'unités de temps (δ_t) d'avance que v doit idéalement avoir sur v'.

Remarque 5. *Si l'on juge que décaler certains voyages dégrade plus la qualité de la grille horaire que d'autres, on peut remplacer le coefficient α_{dec} par des pénalités α^v_{dec} spécifiques pour chaque voyage. Il en va de même pour les espacements et le coefficient α_{esp}.*

Respect des correspondances entre certains voyages

On voudrait, de plus, pouvoir contrôler la qualité des correspondances entre certaines paires de voyages. On note $\mathcal{C}$ l'ensemble des correspondances souhaitées. Soit $c = (v, v') \in \mathcal{C}$, on suppose que l'heure de départ de ces deux voyages peut être modifiée.

À nouveau, le décalage relatif entre les voyages v et v' s'écrit $x_v - x_{v'}$. On suppose connaître la valeur optimale Δ^*_c de ce décalage relatif. Si $x_v - x_{v'} = \Delta^*_c$, le temps de correspondance entre les autobus v et v' est idéal pour les passagers. Pour illustrer la signification du paramètre Δ^*_c, on détaille les deux cas suivants :

- Si dans la grille horaire initiale, la correspondance c est déjà parfaitement respectée, alors on aura $\Delta^*_c = 0$.
- Si, au contraire, il serait préférable que le voyage v' soit retardé de δ_t minutes par rapport au voyage v, alors on aurait $\Delta^*_c = -1$.

Donc, si la correspondance c est parfaitement respectée, on aura $x_v - x_{v'} - \Delta^*_c = 0$. On pénalise les écarts de $x_v - x_{v'} - \Delta^*_c$ à 0 à l'aide d'une fonction linéaire par morceaux f_c

illustrée à la figure 5.1. Grâce à la forme de cette fonction, on peut classer et pénaliser la qualité d'une correspondance selon les trois cas suivants :

1. si la correspondance est optimale ($x_v - x_{v'} - \Delta_c^* = 0$), alors la pénalité est nulle.
2. si la correspondance est ratée ($x_v - x_{v'} - \Delta_c^*$ dépasse l'intervalle de faisabilité $[-l_c^-, l_c^+]$), alors on paye une pénalité P_c.
3. si, enfin, la correspondance est de qualité moyenne, i.e. le risque est grand que les passagers ne puissent pas "attraper" leur correspondance ou bien le temps d'attente entre les deux trajets est un peu long ($x_v - x_{v'} - \Delta_c^* \neq 0 \in [-l_c^-, l_c^+]$), alors on pénalise linéairement le temps d'écart à 0.

Pour définir précisément f_c, on a besoin d'introduire les paramètres suivants :

$[-L_c^-, L_c^+]$	: l'étendue maximale des valeurs de $x_v - x_{v'} - \Delta_c^*$
$[-l_c^-, l_c^+]$	: l'étendue des valeurs de $x_v - x_{v'} - \Delta_c^*$ pour lesquelles la correspondance est réalisable
P_c :	: la pénalité pour une correspondance infaisable
$-\alpha_c^-$, α_c^+	: les pentes de pénalisation des écarts à 0 (donc de l'optimum) de $x_v - x_{v'} - \Delta_c^*$

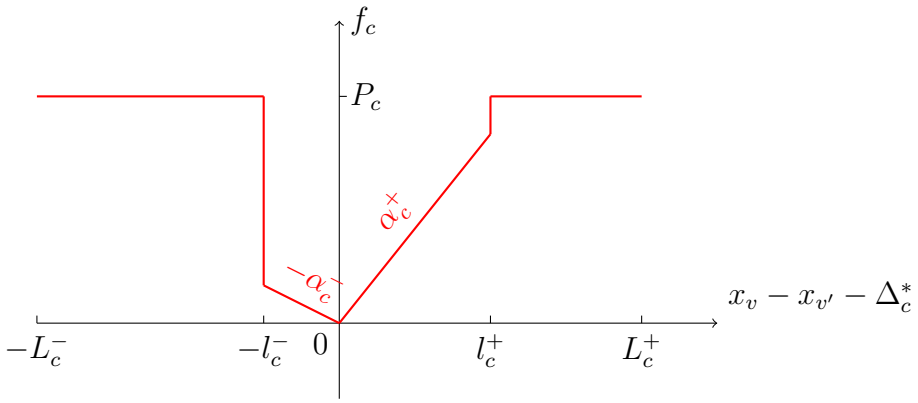

Figure 5.1 Schéma de la fonction f_c : pénalisation des écarts au temps de correspondance idéal

Pour chaque $c = (v, v') \in \mathcal{C}$, il faut maintenant ajouter dans l'objectif du problème (Q) le terme

$$\alpha_{cor} f_c(x_v - x_{v'} - \Delta_c^*)$$

avec α_{cor} le poids relatif accordé aux correspondances (à comparer avec α_{dec} et α_{esp}). Afin de modéliser linéairement $f_c(x_v - x_{v'} - \Delta_c^*)$, il est nécessaire d'introduire de nouvelles variables

d'écart et les contraintes adéquates.

Variables

p_c^+ : la valeur absolue de $x_v - x_{v'} - \Delta_c^*$ si ce terme est positif

p_c^- : la valeur absolue de $x_v - x_{v'} - \Delta_c^*$ si ce terme est négatif

y_c^+ : la variable binaire égale à 1 si et seulement si $x_v - x_{v'} - \Delta_c^* > l_c^+$ et donc la correspondance est infaisable

y_c^- : la variable binaire égale à 1 si et seulement si $x_v - x_{v'} - \Delta_c^* < l_c^-$ et donc la correspondance est infaisable

q_c^+ : la valeur absolue de $x_v - x_{v'} - \Delta_c^*$ si ce terme est dans $[0, l_c^+]$, 0 sinon

q_c^- : la valeur absolue de $x_v - x_{v'} - \Delta_c^*$ si ce terme est dans $[-l_c^-, 0]$, 0 sinon

Objectif Pour chaque $c = (v, v') \in \mathcal{C}$, on ajoute à l'objectif de (Q), le terme suivant

$$\alpha_{cor}\, f_c(x_v - x_{v'} - \Delta_c^*) = \alpha_{cor}\left(\alpha_c^+ q_c^+ + \alpha_c^- q_c^- + P_c(y_c^+ + y_c^-)\right) \tag{5.10}$$

Contraintes Pour chaque $c = (v, v') \in \mathcal{C}$ on ajoute à (Q) les contraintes suivantes, où L_c^+, L_c^- ainsi que $(L_c^+ - l_c^+)$ et $(L_c^- - l_c^-)$ font office de "grand M" :

$$x_v - x_{v'} - \Delta_c^* = p_c^+ - p_c^- \qquad \text{définition de } p_c^+, p_c^- \tag{5.11}$$

$$q_c^+ \geq p_c^+ - y_c^+ L_c^+ \qquad \text{définition de } q_c^+ \tag{5.12}$$

$$q_c^- \geq p_c^- - y_c^- L_c^- \qquad \text{définition de } q_c^- \tag{5.13}$$

$$p_c^+ - l_c^+ \leq y_c^+ (L_c^+ - l_c^+) \qquad \text{définition de } y_c^+ \tag{5.14}$$

$$p_c^- - l_c^- \leq y_c^- (L_c^- - l_c^-) \qquad \text{définition de } y_c^- \tag{5.15}$$

$$p_c^-,\ p_c^+ \geq 0 \tag{5.16}$$

$$q_c^-,\ q_c^+ \geq 0 \tag{5.17}$$

$$y_c^-,\ y_c^+ \in \{0, 1\} \tag{5.18}$$

$$x_v \in M_v',\ x_{v'} \in M_{v'}' \tag{5.19}$$

On a donc construit un modèle linéaire pour maximiser la qualité des correspondances. Ce modèle permet de classer le décalage relatif entre deux voyages en plusieurs régions (correspondance idéale, correcte ou ratée) qui sont pénalisées différemment.

Remarque 6. *On a traité ici le cas d'une correspondance entre deux voyages qui peuvent tous les deux être décalés. Il peut également exister le cas (plus facile) où l'on souhaite coordonner un trajet d'autobus v décalable avec un événement extérieur (par exemple un train) ou avec*

un autre trajet dont l'horaire est fixé. Il suffit alors de poser $x'_v = 0$ *dans le modèle précédent puis de minimiser* $f_c(x_v - \Delta_c^*)$ *où* Δ_c^* *est défini comme précédemment.*

5.1.2 Algorithmes de résolution

Nous devons maintenant faire le lien entre le problème de MDVSP avec décalage d'horaires (PM) et le problème de contrôle des horaires de départ (Q). On remarque que choisir un itinéraire p qui couvre le voyage v a une influence sur la valeur de x_v. Dans certains réseaux (voir la section 3.3.1), p impose une unique valeur pour x_v. En revanche, le sous-réseau de choix de copie (voir la section 3.3.2) détient la propriété suivante.

Propriété 1. *Un chemin dans le graphe avec sous-réseau de choix de copie ne fixe pas nécessairement l'heure de départ des voyages qu'il couvre.*

Considérons, par exemple, le sous-réseau associé à un voyage v représenté par trois copies a, b et c. Soit un chemin p (voir les arcs en gras de la figure 5.2) qui entre dans le sous-réseau par le nœud représentant la copie a et qui en sort par la copie b. Alors les copies a et b sont compatibles avec le chemin p. On peut donc choisir pour v l'heure de départ de la copie a ou celle de b, ou n'importe quelle heure de départ comprise entre celle de a et celle de b. Si le chemin p est choisi, on aura donc l'encadrement de x_v suivant $d_v^a \leq x_v \leq d_v^b$ où d_v^a et d_v^b sont les décalages des copies a et b (dans M'_v).

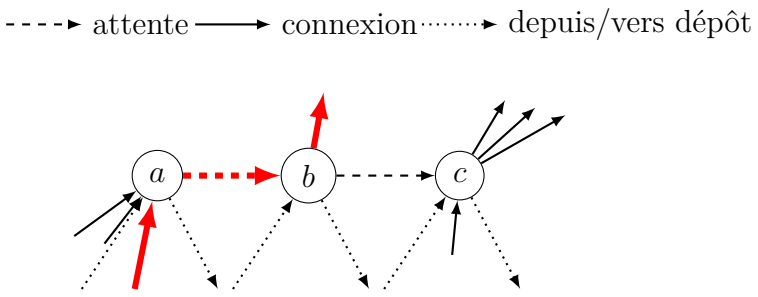

Figure 5.2 Illustration de la propriété 1

Dans la suite, on dira qu'un arc ou un chemin *entre dans le sous-réseau* par la copie u lorsqu'on emprunte un arc de connexion ou en provenance du dépôt dont la seconde extrémité est le nœud représentant u. On ne tient donc pas compte des arcs attente qui relient les copies entre elles, et que l'on considère comme des arcs "à l'intérieur" du sous-réseau. Il en va de même lorsqu'on dit qu'on *sort du sous-réseau* par une certaine copie.

Ainsi, choisir un chemin implique un encadrement sur la valeur de x_v. A partir de cette remarque, on propose les deux méthodes de résolution décrites ci-après : la première consiste à ajouter le modèle (Q) au problème maître (PM) et ainsi résoudre un problème multi-objectif minimisant à la fois le coût des itinéraires et les pénalités sur la qualité des horaires. La seconde décompose le problème en deux phases et s'avère beaucoup plus efficace que la première.

Ajout direct des contraintes de contrôle des horaires dans le problème maître. Une solution pour prendre en compte la qualité des horaires dans la résolution du MDVSP consiste simplement à ajouter les contraintes et la fonction objectif de (Q) au problème (PM). Pour ce faire, il faut lier le choix des itinéraires p et les contraintes d'encadrement de x_v.

On note pour chaque copie $u \in M_v$, $d_v^u \in M'_v$ le décalage (en unité δ_t minute) qui lui est associé. On introduit de plus pour chaque voyage $v \in \mathcal{V}$ et chaque copie $u \in M_v$, les variables binaires x_u^e et x_u^s : x_u^e est égale à 1 si on entre dans le sous-réseau par la copie u et à 0 sinon. Symétriquement x_u^s vaut 1 si on sort du sous-réseau par la copie u.

Dans le problème (Q), l'encadrement maximal $lb_v \leq x_v \leq ub_v$ devient

$$\sum_{u \in M_v} d_v^u x_u^e \leq x_v \leq \sum_{u \in M_v} d_v^u x_u^s. \tag{5.20}$$

Enfin, pour faire le lien entre les variables x_u^s, x_u^e et les variables y_p, on introduit les paramètres a_p^{ue} et a_p^{us} égaux à 1 si le chemin p entre dans le (respectivement sort du) sous-réseau du voyage v par la copie u.

On a alors

$$\begin{aligned} x_u^e &= \sum_{p \in \Omega} a_p^{ue} y_p \\ x_u^s &= \sum_{p \in \Omega} a_p^{us} y_p. \end{aligned} \tag{5.21}$$

On peut donc définir le problème (PMQ) : le MDVSP avec décalage contrôlé des horaires de départ (par souci de lisibilité, on n'a pas ajouté ici le contrôle des correspondances).

$$\min_{y,x,s,p} \sum_{d\in\mathcal{D}} \sum_{p\in\Omega^d} c_p y_p + \alpha_{dec} \sum_{v\in\mathcal{V}} (s_v^+ + s_v^-) + \alpha_{esp} \sum_{(v,v')\in\mathcal{E}} (p_{vv'}^+ + p_{vv'}^-) \tag{5.22}$$

sous les contraintes (5.23)

$$\sum_{d\in\mathcal{D}} \sum_{p\in\Omega^d} a_{vp} y_p = 1 \qquad \forall v \in \mathcal{V} \tag{5.24}$$

$$\sum_{p\in\Omega^d} y_p \leq r_d \qquad \forall d \in \mathcal{D} \tag{5.25}$$

$$\sum_{u\in M_v} d_v^u \sum_{p\in\Omega} a_p^{ue} y_p \leq x_v \qquad \forall v \in \mathcal{V} \tag{5.26}$$

$$\sum_{u\in M_v} d_v^u \sum_{p\in\Omega} a_p^{us} y_p \geq x_v \qquad \forall v \in \mathcal{V} \tag{5.27}$$

$$x_v = s_v^+ - s_v^- \qquad \forall v \in \mathcal{V}, \tag{5.28}$$

$$x_v - x_{v'} = p_{vv'}^+ - p_{vv}^- \qquad \forall (v,v') \in \mathcal{E} \tag{5.29}$$

$$y_p \in \{0,1\} \qquad \forall p \in \Omega \tag{5.30}$$

$$x_v \in \mathbb{N} \qquad \forall v \in \mathcal{V} \tag{5.31}$$

$$s_v^-,\ s_v^+ \geq 0 \qquad \forall v \in \mathcal{V} \tag{5.32}$$

$$p_{vv'}^-,\ p_{vv'}^+ \geq 0 \qquad \forall (v,v') \in \mathcal{E} \tag{5.33}$$

L'objectif est donc de minimiser une somme pondérée du coût des itinéraires et des pénalités sur la qualité des horaires. Comme dans (PM), on a les contraintes de satisfaction de chaque voyage (5.24) et de la capacité de chaque dépôt (5.25). Les contraintes (5.26) et (5.27) donnent l'encadrement de x_v imposé par les itinéraires choisis. C'est une réécriture de (5.20) en tenant compte de (5.21). Comme dans (Q), on a ensuite les contraintes de pénalisation des écarts à la copie initiale (5.28) et des espacements modifiés (5.29).

Nous avons implémenté le modèle (PMQ) et l'avons résolu par génération de colonnes. Cependant, le fait d'avoir ajouté le modèle (Q) tend à fractionner les solutions trouvées. En effet, en prenant des combinaisons de chemins adéquatement choisis, le problème parvient à contraindre le moins possible (5.26) et (5.27), assurant par exemple qu'on puisse toujours avoir $x_v = 0$ et donc ne pas payer les pénalités de décalage ou de modification des espacements. Dans les tests réalisés, on ne pouvait résoudre exactement dans des temps raisonnables (3 heures) que des problèmes de 50 voyages ou moins. La version heuristique ne donnait pas non plus de résultats satisfaisants car elle fixait des colonnes dont la valeur fractionnaire était faible.

Algorithme en deux phases. Comme le problème (PMQ) est trop complexe à résoudre tel quel, nous proposons maintenant une deuxième approche où l'on sépare le problème en deux étapes successives :

— on résout le MDVSP avec décalage d'horaires et on trouve ainsi l'ensemble des itinéraires p optimaux.

— on choisit ensuite les meilleurs horaires possibles (au sens de (Q)) à partir des itinéraires p précédemment trouvés.

Autrement dit, on cherche d'abord les itinéraires qui minimisent les coûts (nombre de véhicules et coût des connexions), puis on trouve le meilleur horaire que ces itinéraires permettent. Lorsqu'un itinéraire est fixé, on peut savoir quels horaires sont possibles. Si p couvre le voyage v, alors on a :

$$\sum_{u \in M_v} d_v^u a_p^{ue} \leq x_v \leq \sum_{u \in M_v} d_v^u a_p^{us} \tag{5.34}$$

L'encadrement (5.34) signifie que la valeur de x_v est bornée inférieurement par l'horaire de la copie par laquelle p entre dans le sous-réseau v et supérieurement par le décalage de la copie de sortie. Lors de la seconde étape, on résout donc (Q) en remplaçant (5.3) et (5.4) par (5.34). La méthode de calcul en deux étapes permet de ne pas complexifier le problème (PM).

Cependant, il est souhaitable que les décisions prises dans la première phase influent le moins possible sur les choix d'horaires de la deuxième étape. On voudrait éviter de choisir des itinéraires p qui contraignent fortement le problème de choix d'horaires (Q), à moins qu'ils ne permettent de diminuer significativement le coût de la solution. On peut pour cela ajouter des pénalités sur certains arcs de connexion entre les sous-réseaux. Par exemple, si un itinéraire p emprunte un arc de connexion qui sort de la copie a, alors on a dans (Q) la contrainte suivante $x_v = d_v^a$. Cela limite fortement le domaine réalisable de (Q). On propose donc d'ajouter une forte pénalité L sur les arcs sortant de la copie a. On fait de même pour les arcs entrant dans le sous-réseau par la dernière copie, qui imposent $x_v = d_v^c$. Dans le cas à trois copies, on ajoute également une pénalité légère l sur les arcs entrant en b (qui empêchent de prendre la copie a) et ceux sortant par b (qui empêchent de prendre la copie c). On donne à la figure 5.3 les pénalités l et L ajoutées sur les arcs pour un sous-réseau avec trois copies.

On pénalise pareillement les arcs de connexion et ceux représentant un départ/retour au dépôt. Les seuls arcs non pénalisés sont ceux entrant dans le sous-réseau par la première copie ou sortant par la dernière copie (et les arcs attente).

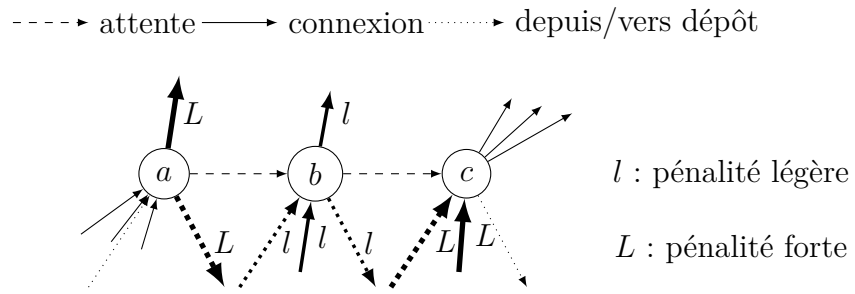

Figure 5.3 Définition des pénalités (l, L) dans un sous-réseau à trois copies.

On verra dans la section 5.2 que l'introduction des pénalités augmente la qualité des horaires obtenus lors de la seconde étape. On y analyse différents choix de paire de pénalités (l, L).

Algorithme 1 Résolution en deux phases du MDVSP avec décalage contrôlé des horaires de départ

1. Choisir les pénalités l et L et construire les graphes $\mathcal{G}^d$ (avec sous-réseaux).
2. Résoudre le MDVSP avec décalage d'horaires par génération de colonnes (exacte ou heuristique).
3. Analyser les itinéraires optimaux obtenus à l'étape 2 et en déduire les encadrements sur x_v (5.34).
4. Résoudre le problème de maximisation des horaires (Q) avec les encadrements obtenus à l'étape 3 à la place de (5.3) et (5.4).

5.1.3 Prolongements

Il existe plusieurs voies d'amélioration de cet algorithme en deux phases pour résoudre le MDVSP avec décalage contrôlé des horaires de départ. On en propose ici deux : on étudie d'abord un moyen d'élargir l'ensemble des solutions possibles dans la seconde étape, sans changer le coût des itinéraires de la première étape. Ensuite, on souligne le lien entre le choix des pénalités (l, L) et une relaxation lagrangienne du problème (PMQ).

Prise en compte des arcs de connexion multiples

On propose une manière d'élargir le domaine des solutions du problème (Q) résolu dans la seconde étape. On remarque que dans l'algorithme en deux phases, on ne permet pas de revenir sur le choix des itinéraires p fait dans la première partie. On a considéré, en particulier, que le choix des arcs était fixé et qu'il conduisait donc à un unique encadrement sur la valeur de x_v. Or, dans la construction des arcs reliant les sous-réseaux entre eux, il existe des cas dans lesquels plusieurs arcs représentent la même connexion. La figure 3.12 illustre les 5 cas

possibles pour deux voyages ayant chacun trois copies (pareillement espacées).

Prenons, par exemple (figure 5.4), le deuxième cas proposé de la figure 3.12. Dans celui-ci, une même connexion entre les voyages v et v' doit être représentée par deux arcs parallèles. En effet, si le temps minimal entre le départ de v et de v' est noté T (la durée du voyage v plus le temps minimal pour effectuer le trajet entre l'arrivée de v et le départ de v') et vérifie l'encadrement

$$t_{a'} - t_c < T \leq t_{b'} - t_c = t_{a'} - t_b$$

alors il est nécessaire d'introduire les arcs (b, a') et (c, b') pour être sûr de ne pas empêcher un enchaînement de voyages réalisables.

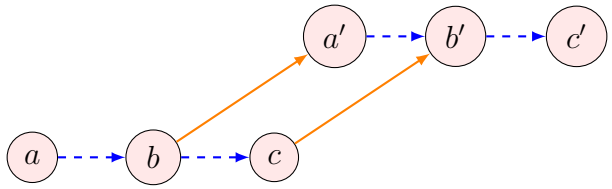

Figure 5.4 Exemple de connexion entre deux sous-réseaux représentée par plusieurs arcs

Supposons qu'un itinéraire p contienne la connexion $v - v'$ et donc passe, par exemple, par l'arc (b, a'). Dans le modèle (Q), prendre cet arc impose $x_v \leq d_v^b$ (restrictif) et $d_{v'}^a \leq x_{v'}$ (non restrictif). La question suivante se pose alors : aurait il été possible (dans la construction de p) et préférable (pour le contrôle des horaires Q) de prendre l'arc (c, b') au lieu de (b, a') ?

Lorsqu'un itinéraire p contient une connexion représentée par plusieurs arcs parallèles, on propose de relâcher le choix de l'arc fait dans la première étape et de garder seulement l'information "il faut faire une connexion directe entre v et v'". Pour s'assurer que cette connexion reste réalisable, il faut que x_v et $x_{v'}$ vérifient l'inégalité

$$x_{v'} - x_v \geq \left\lceil \frac{T - (t_{b'} - t_b)}{\delta_t} \right\rceil \tag{5.35}$$

Si on retourne à l'exemple de la figure 5.4, l'inégalité (5.35) devient $x_{v'} - x_v \geq -1$. On remplace donc les deux inégalités impliquées par le choix de l'arc (b, a') ainsi :

$$\begin{cases} x_v \leq d_v^b \\ x_{v'} \geq d_{v'}^{a'} \end{cases} \longrightarrow \begin{cases} x_v \leq d_v^c \\ x_{v'} \geq d_{v'}^{a'} \\ x_{v'} - x_v \geq -1 \end{cases}$$

Dans le cas général, on supposons qu'on a k arcs représentant la connexion entre v et v'. On les notes $(u_1, u_1'), \ldots, (u_k, u_k')$. Soit un itinéraire p qui utilise l'arc (u_i, u_i') avec $i \in \{1, .., k\}$. Alors on peut faire la transformation suivante :

$$\begin{cases} x_v \leq d_v^{u_i} \\ x_{v'} \geq d_{v'}^{u_i'} \end{cases} \longrightarrow \begin{cases} x_v \leq d_v^{u_k} \\ x_{v'} \geq d_{v'}^{u_1'} \\ x_{v'} - x_v \geq \left\lceil \dfrac{T - (t_{b'} - t_b)}{\delta_t} \right\rceil \end{cases} \tag{5.36}$$

Si la troisième équation du système de droite est vérifiée, alors les deux premières le sont aussi (sinon on aurait pu définir d'autres arcs de connexion entre v et v') donc elles sont non nécessaires.

Par rapport à l'algorithme 1 en deux phases, on a donc proposé les deux modifications suivantes :

- à l'étape 3., on distingue les arcs de connexion unique (pour lesquels on ajoute les contraintes (5.34)), des arcs de connexion multiples pour lesquels on ajoute les contraintes (5.35).
- à l'étape 4., on résout le problème (Q) avec ces contraintes.

On pourrait également reconsidérer le choix des pénalités l, L en distinguant les mêmes cas qu'à la figure 3.12. On pourrait alors choisir une pénalité pour chacun de ces cas, plutôt qu'une pénalité dépendant des arcs de sortie/entrée dans le sous-réseaux.

La prise en compte des arcs de connexion multiples décrite ici, a été implémentée dans les tests présentés à la section 5.2. Les résultats ainsi obtenus sont jusqu'à 50 % meilleurs (en terme de valeur du problème (Q)) que lorsque que les arcs multiples ne sont pas pris en compte.

Lien avec la relaxation lagrangienne

On montre dans ce paragraphe le lien que l'on peut faire entre les pénalités (l, L) et une relaxation lagrangienne du problème global d'optimisation simultanée des itinéraires d'autobus et des horaires. On avait ainsi défini à la section 5.1.1 le problème (PMQ) qui s'appuie sur la formulation en partitionnement en circuits du MDVSP (PM). On réécrit ci-dessous (PMQ) en introduisant explicitement les variables x_u^e et x_u^s pour rendre les contraintes (5.41) et (5.42) plus lisibles.

$$\min_{y,x,s,p} \sum_{d\in\mathcal{D}} \sum_{p\in\Omega^d} c_p y_p + \alpha_{dec} \sum_{v\in\mathcal{V}} (s_v^+ + s_v^-) + \alpha_{esp} \sum_{(v,v')\in\mathcal{E}} (p_{vv'}^+ + p_{vv'}^-) \tag{5.37}$$

sous les contraintes (5.38)

$$\sum_{d\in\mathcal{D}} \sum_{p\in\Omega^d} a_{vp} y_p = 1 \qquad \forall v \in \mathcal{V} \tag{5.39}$$

$$\sum_{p\in\Omega^d} y_p \leq r_d \qquad \forall d \in \mathcal{D} \tag{5.40}$$

$$\sum_{u\in M_v} d_v^u x_u^e \leq x_v \qquad \forall v \in \mathcal{V} \tag{5.41}$$

$$\sum_{u\in M_v} d_v^u x_u^s \geq x_v \qquad \forall v \in \mathcal{V} \tag{5.42}$$

$$\sum_{p\in\Omega} a_p^{ue} y_p = x_u^e \qquad \forall u \in M_v,\ \forall v \in \mathcal{V} \tag{5.43}$$

$$\sum_{p\in\Omega} a_p^{us} y_p = x_u^s \qquad \forall u \in M_v,\ \forall v \in \mathcal{V} \tag{5.44}$$

$$x_v = s_v^+ - s_v^- \qquad \forall v \in \mathcal{V}, \tag{5.45}$$

$$x_v - x_{v'} = p_{vv'}^+ - p_{vv}^- \qquad \forall (v, v') \in \mathcal{E} \tag{5.46}$$

$$y_p \in \{0, 1\} \qquad \forall p \in \Omega \tag{5.47}$$

$$x_v \in \mathbb{N} \qquad \forall v \in \mathcal{V} \tag{5.48}$$

$$s_v^-,\ s_v^+ \geq 0 \qquad \forall v \in \mathcal{V} \tag{5.49}$$

$$p_{vv'}^-,\ p_{vv'}^+ \geq 0 \qquad \forall (v, v') \in \mathcal{E} \tag{5.50}$$

On relaxe les contraintes (5.41) et (5.42), c'est-à-dire celles liant les décisions sur les itinéraires et celles sur les horaires et on les place dans l'objectif. On note la relaxation lagrangienne obtenue (PMQ^{Lagr}). Les multiplicateurs de Lagrange de ces contraintes seront respectivement notés $\lambda_v^{inf} \geq 0$ et $\lambda_v^{sup} \geq 0$. L'objectif (5.37) devient alors

$$\min_{y,x,s,p} \sum_{d\in\mathcal{D}} \sum_{p\in\Omega^d} c_p y_p + \alpha_{dec} \sum_{v\in\mathcal{V}} (s_v^+ + s_v^-) + \alpha_{esp} \sum_{(v,v')\in\mathcal{E}} (p_{vv'}^+ + p_{vv'}^-) \\ - \sum_{v\in\mathcal{V}} \lambda_v^{inf} (x_v - \sum_{u\in M_v} d_v^u x_u^e) - \sum_{v\in\mathcal{V}} \lambda_v^{sup} (\sum_{u\in M_v} d_v^u x_u^s - x_v). \tag{5.51}$$

Le fait d'avoir relaxé les contraintes (5.41) et (5.42) a pour conséquence que le problème (PMQ^{Lagr}) peut se résoudre en séparant les décisions sur les itinéraires (contraintes (5.39) et (5.40)) et les décisions sur les horaires (5.45) et (5.46).

Dans l'objectif, on a donc, entre autres, ajouter pour tout $v \in \mathcal{V}$ le terme

$$\lambda_v^{inf} \sum_{u\in M_v} d_v^u x_u^e - \lambda_v^{sup} \sum_{u\in M_v} d_v^u x_u^s$$

Cela revient à ajouter un coût $\lambda_v^{inf} d_v^u$ sur un arc entrant dans le sous-réseau par la copie u et $-\lambda_v^{sup} d_v^u$ sur un arc sortant du sous-réseau par la copie u. On voit alors le lien avec les pénalités (l, L) définies précédemment : plus le décalage d'une copie d_v^u est grand, plus la pénalité pour entrer par cette copie ($\lambda_v^{inf} d_v^u$) est grande et la pénalité pour sortir par cette copie ($-\lambda_v^{sup} d_v^u$) est petite (algébriquement).

Dans un exemple à 3 copies et $M'_v = \{-1, 0, 1\}$, on compare au tableau 5.1, les pénalités sur les arcs sortants et les coûts supplémentaires des arcs dans le modèle (PMQ^{Lagr}). On notera que les arcs sortants sont pénalisés par $[L, l, 0]$, si on pénalise par L un arc sortant de la première copie, par l un arc sortant de la deuxième et par 0, ceux sortant de la dernière. On introduit le "coût Lagrange révisé" dans lequel on a simplement ajouté le coût constant λ_v^{inf} à tous les arcs entrants dans le sous-réseau du voyage v et λ_v^{sup} à tous les arcs en sortant. Cela ne change rien à la valeur du problème, mais permet de comparer plus facilement les coûts lagrangiens et les pénalités (l, L).

On voit donc que dans le cas où $L = 2l$ et $\lambda_v^{inf} = \lambda_v^{sup} = l$ pour tout $v \in \mathcal{V}$, la première phase de l'algorithme permet, en fait, de résoudre la relaxation lagrangienne (PMQ^{Lagr}). La seconde phase trouve ensuite une solution réalisable optimale (vis-à-vis des décisions déjà prises).

Il serait donc intéressant de développer une méthode de résolution en deux phases itérative en mettant à jour les pénalités sur les arcs à la manière d'une relaxation lagrangienne.

Tableau 5.1 Comparaison entre les pénalités (l, L) et les coûts supplémentaires sur les arcs dans la relaxation lagrangienne

	pénalités (l, L)	coût Lagrange	coût Lagrange révisé
arcs sortants	$[L, l, 0]$	$[\lambda_v^{sup}, 0, -\lambda_v^{sup}]$	$[2\lambda_v^{sup}, \lambda_v^{sup}, 0]$
arcs entrants	$[0, l, L]$	$[-\lambda_v^{inf}, 0, \lambda_v^{inf}]$	$[0, \lambda_v^{inf}, 2\lambda_v^{inf}]$

5.2 Résultats numériques

On analyse maintenant les résultats numériques obtenus par l'algorithme en deux phases présenté dans la section précédente. On cherche à quantifier l'influence du choix des pénalités l, L et de l'ensemble M_v. Dans un premier temps, la seconde phase de l'algorithme a pour objectif de limiter les décalages de voyage et les modifications d'espacement. On ajoute à la section 5.2.2 le contrôle des correspondances.

5.2.1 Contrôle du nombre de voyages décalés et de la modification des espacements

Pour résoudre la première phase de l'algorithme, i.e. le MDVSP avec décalage d'horaires, on utilise l'algorithme de génération de colonnes heuristique décrit à la section 4.4. A chaque nœud de l'arbre de branchement, on fixe donc un itinéraire et les contraintes de partitionnement sont perturbées.

Pour la seconde phase de l'algorithme, i.e. la maximisation de la qualité des horaires à partir des itinéraires trouvés dans la première phase, on résout le programme en nombres entiers (Q) à l'aide de CPLEX (version 12.6.3.0). Les temps de calcul relatifs à cette phase ne seront pas donnés ici car ils sont négligeables par rapport à ceux de la première étape (de l'ordre de 0.10 seconde ou moins).

Nouveaux schémas de copies. On étudie ici l'influence du choix des copies M_v. Jusqu'à présent, tous les exemples ont été donnés avec la suite de décalages $[-2, 0, 2]$ minutes, que nous appellerons aussi *schéma de copies*. Au tableau 5.2, on introduit deux nouveaux schémas de copies qui diffèrent du précédent par le nombre de copies et l'espacement entre celles-ci.

Tableau 5.2 Définition de 3 schémas de copies.

Nom	Décalages en minutes
3_2min	$[-2, 0, 2]$
5_1min	$[-2, -1, 0, 1, 2]$
5_2min	$[-4, -2, 0, 2, 4]$

Le schéma 5_1min permet donc les mêmes décalages que 3_2min mais est discrétisé à la minute. Au contraire, 5_2min est plus large que 3_2min et il permet d'avancer ou de retarder les voyages jusqu'à quatre minutes.

Pour les deux nouveaux schémas avec 5 copies, il faut adapter les pénalités (l, L) définies à la figure 5.3 pour un cas avec 3 copies par voyage. Les pénalités sur les arcs sortant du sous-réseau peuvent être définies sous forme de liste, pour 3 copies on a choisi $[L, l, 0]$, c'est-à-dire qu'on pénalise par L les arcs sortant de la première copie, par l ceux sortant de la deuxième et qu'on ne pénalise pas les arcs sortant de la dernière copie. De même, on définit la liste des pénalités sur les arcs entrant dans le sous-réseau : $[0, l, L]$ lorsqu'on a 3 copies. Les pénalités dans un cas à 5 copies sont données dans le tableau 5.3.

Tableau 5.3 Définition des pénalités l, L dans les cas avec 3 et 5 copies

	3 copies	5 copies
arcs sortants	$[L, l, 0]$	$[2L, L, l, \frac{l}{2}, 0]$
arcs entrants	$[0, l, L]$	$[0, \frac{l}{2}, l, L, 2L]$

On remarque que le choix fait au tableau 5.3 permet d'avoir (quasiment) les mêmes pénalités sur les copies $[-2, 0, 2]$ dans les schémas 3_2min et 5_2min.

Tous les résultats de la section 5.2 sont donnés en moyenne sur 5 instances de 500 voyages. Dans chacun des tests réalisés, le même schéma de copies (3_2min, 5_1min ou 5_2min) est appliqué à tous les voyages de l'instance et le réseau utilisé est **src*_3c**. On analyse l'influence des pénalités (l, L) et du schéma de copies sur la qualité des horaires et le coût des itinéraires dans le tableau 5.4. On note, par exemple, **l1L5** le choix $l = 1, L = 5$. On donne comme référence les valeurs du problème de MDVSP (voir la première ligne) obtenus avec le

réseau **mix***.

Dans les premières colonnes du tableau 5.4, on donne les résultats de la première phase de l'algorithme (MDVSP avec décalage d'horaires) :

1. le coût total des itinéraires. Le coût des pénalités n'est pas inclus dans cette valeur.
2. le cout "opérationnel", c'est-à-dire le coût total des itinéraires moins les coûts fixes d'utilisation des véhicules (1000 par véhicule). Il s'agit donc du coût lié aux voyages haut-le-pied, aux sorties/rentrées au dépôt et aux attentes.
3. le nombre d'autobus

Dans la seconde section, on donne les résultats de la seconde phase de l'algorithme, i.e. le problème (Q). Dans les tests réalisés, on a adopté la même discrétisation pour la définition des copies et pour la résolution de (Q) ("$M_v = M'_v$"). On a donc pris $\delta_t = 2$ pour 3_2min et 5_2min et $\delta_t = 1$ pour 5_1min. Pour pouvoir comparer facilement les résultats, on a ensuite rapporté les résultats de (Q) à la minute (ce qui revient à multiplier par deux les résultats des schémas 3_2min et 5_2min). Enfin, on a choisi $\alpha_{dec} = 1,\ \alpha_{esp} = 2$ comme poids dans la fonction objectif. Le tableau 5.4 donne ensuite :

1. le *Score*, c'est-à-dire la valeur optimale de (Q), les résultats sont classés par ordre décroissant de cette valeur.
2. la somme des décalages en minute à la grille horaire initiale : $\sum_{v \in \mathcal{V}} (s_v^+ + s_v^-)$.
3. la somme des écarts aux espacements optimaux : $\sum_{(v,v') \in \mathcal{E}} (p_{vv'}^+ + p_{vv'}^-)$.

Tableau 5.4 Comparaison des coûts et de la qualité des horaires pour différents choix de pénalités et de schémas de copies. Résultats en moyenne sur 5 instances de 500 voyages.

		Coût des itinéraires			**Problème Q**		
Copie	**Pénalités**	**coût tot.**	**coût opé.**	**nb. bus**	**Score**	**Som.déc.**	**Som.esp.**
1	**MDVSP**	36441	1441	35,0			
3_2min	**l0L0**	33579,5	1179,5	32,4	1822,4	692,8	564,8
	l0,1L0,5	33635,7	1235,7	32,4	693,6	223,2	235,2
	l1L2	33834,6	1434,6	32,4	224,8	63,2	80,8
	l0,5L5	33875,8	1475,8	32,4	97,6	24,8	36,4
	l1L5	33905,5	1505,5	32,4	85,6	21,6	32,0
	l5L10	34171,4	1771,4	32,4	65,6	15,2	25,2
	l1L10	33941,0	1541,0	32,4	57,2	12,4	22,4
	l0,5L10	33922,1	1522,1	32,4	55,6	13,2	21,2
	l10L50	34232,2	1832,2	32,4	43,6	9,2	17,2
	l100L500	36469,6	1869,6	34,6	4,0	0,8	1,6
5_1min	**l0L0**	33570,2	1170,2	32,4	1736,2	697,0	519,6
	l1L2	33856,8	1456,8	32,4	119,2	31,2	44,0
	l1L5	33919,8	1519,8	32,4	52,4	13,2	19,6
	l1L10	33944,5	1544,5	32,4	40,4	10,0	15,2
	l10L50	34220,4	1820,4	32,4	34,6	8,6	13,0
	l5L50	34141,2	1741,2	32,4	34,2	10,2	12,0
5_2min	**l0L0**	29149,3	949,3	28,2	4139,2	1346,4	1396,4
	l1L5	29690,9	1490,9	28,2	614,0	197,2	208,4
	l1L10	29753,0	1553,0	28,2	563,6	180,4	191,6
	l1L50	31403,6	1603,6	29,8	256,4	72,4	92,0
	l5L50	31745,4	1945,4	29,8	230,8	68,4	81,2
	l10L50	32249,1	2049,1	30,2	177,2	51,6	62,8
	l10L100	33733,0	2133,0	31,6	85,2	20,4	32,4

Les figures 5.5 et 5.6 permettent de comparer visuellement les solutions du tableau 5.4. Chaque point du graphe correspond à une ligne du tableau. L'abscisse donne le coût des itinéraires et l'ordonnée la valeur de (Q) (le Score). Dans la figure 5.6, on a placé l'ensemble des points obtenus avec les 3 schémas de copies. Dans la figure 5.5, on représente avec une échelle plus grande les points obtenus pour 3_2min et 5_1min.

Le tableau 5.4 démontre que l'ajout des pénalités permet de diminuer considérablement la valeur optimale du problème (Q) et donc d'augmenter la qualité des horaires. Pour le schéma 3_2min, on parvient ainsi à diminuer le nombre de voyages décalés sans augmenter le nombre de véhicules mais seulement le coût opérationnel. Il est intéressant de remarquer qu'il peut être suffisant de modifier très peu de voyages (13.2 minutes de décalage, soit environ 7 voyages décalés de 2 minutes pour **l0.5L10**, par exemple) pour économiser un ou plusieurs véhicules. Il appartient ensuite au décideur de choisir quel compromis il préfère entre le coût de la solution et la qualité des horaires (voir graphique 5.5 et 5.6). La discrétisation à la minute du schéma 5_1min permet de diminuer encore un peu les pénalités sur la qualité des horaires.

Si on permet de décaler les horaires jusqu'à 4 min de retard/d'avance (schéma 5_2min), le coût des itinéraires peut encore être largement diminué. En effet, on pourrait ainsi économiser en moyenne 6.8 véhicules par rapport à la solution du MDVSP sans décalage d'horaires. La qualité des horaires s'en trouve cependant diminuée.

On analyse également la taille des réseaux et les temps de calcul pour les instances du tableau 5.4. Le nombre moyen d'arcs des réseaux est détaillé au tableau 5.5 pour chacun des schémas de copies. On distingue les arcs de connexion entre les sous-réseaux, les arcs attente entre les copies d'un même voyage, les arcs entre les voyages et le dépôt et les arcs représentant une attente au dépôt. Pour chaque instance, on rappelle qu'il y a deux graphes car deux dépots.

Le tableau 5.6 donne dans l'ordre : la meilleure solution entière obtenue (celle-ci tient compte des pénalités), la valeur de (PM^{rl}) (préalablement obtenue sans perturber les contraintes), et la différence entre ces deux valeurs que l'on appelle ici "gap". On donne ensuite le temps CPU total de résolution (en secondes), le temps nécessaire pour résoudre la première relaxation linéaire (PM^{rl}_{pert}), le temps CPU consacré au sous-problème et, enfin, le nombre de nœuds dans l'arbre de branchement.

Le tableau 5.6 montre que l'algorithme heuristique donne en un peu plus de 2 minutes en

Tableau 5.5 Nombre d'arcs pour les 3 schémas de copies. Résultats en moyenne sur 5 instances de 500 voyages.

Type d'arcs	connexion	attente	depuis/vers dépôt	attente depôt	Total
3_2min	4613.4	1000.0	6000.0	906.4	**18133.2**
5_1min	5286.0	2000.0	10000.0	1246.6	**25818.6**
5_2min	7563.2	2000.0	10000.0	1268.0	**30394.4**

moyenne des résultats très proches de l'optimum pour les schémas 3_2min et 5_1min. Quand le gap est nul, on est en effet assuré d'être à l'optimum. Dans le cas contraire, le gap donne une borne supérieure sur l'écart à la valeur optimale. Pour les instances de 5_2min avec faibles pénalités, les gaps sont plus significatifs.

De plus, on observe que, par rapport au schéma 3_2min, les temps de calcul augmentent en moyenne de 8% pour le schéma 5_1min et de 75% pour 5_2min. Pour un même schéma de copies, les temps de calcul varient en revanche peu avec le choix des pénalités l, L.

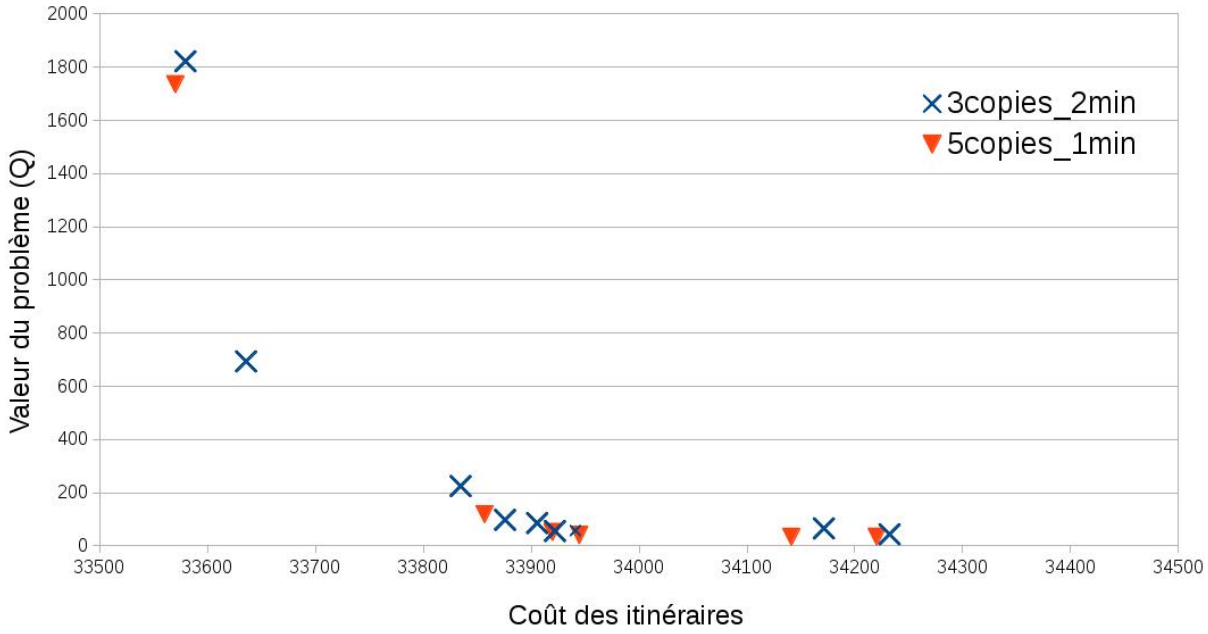

Figure 5.5 Graphique coût des itinéraires/qualité des horaires pour différents choix de pénalités et les schémas de copies 3_2min ($[-2, 0, 2]$) et 5_1min ($[-2, -1, 0, 1, 2]$).

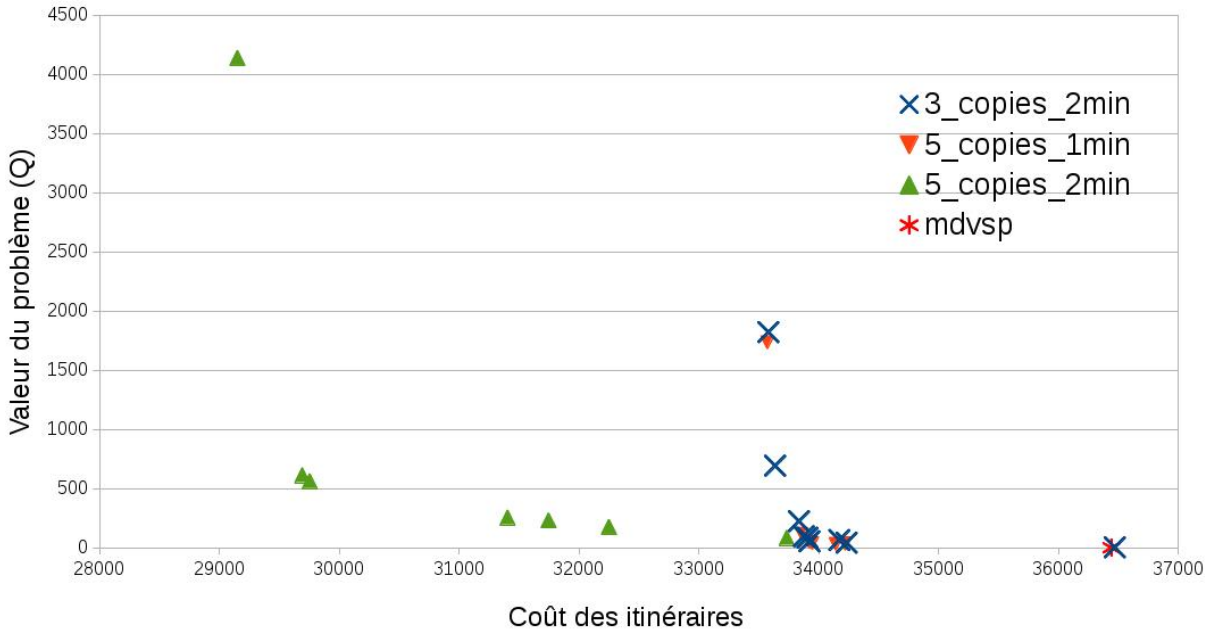

Figure 5.6 Graphique coût des itinéraires/qualité des horaires pour différents choix de pénalités et les schémas de copies 3_2min ($[-2, 0, 2]$), 5_1min ($[-2, -1, 0, 1, 2]$) et 5_2min ($[-4, -2, 0, 2, 4]$).

Tableau 5.6 Comparaison des temps de calcul pour différents choix de pénalités et de schémas de copies. Résultats en moyenne sur 5 instances de 500 voyages.

		Coût de la solution			Temps CPU (s)			Nb.
Copie	**Pénalités**	**Entière**	**Relax.**	**GAP**	**Total**	**Nœud 0**	**S-P**	**nœuds**
3_2min	**l0L0**	33579,48	33577,26	2,22	123,8	79,5	12,9	33,2
	l0,1L0,5	33731,14	33731,14	0,00	114,2	83,0	11,2	33,4
	l1L2	34053,00	34053,00	0,00	135,3	90,5	12,1	33,8
	l5L10	34443,36	34440,78	2,58	151,8	102,9	13,4	33,8
	l0,5L5	34035,40	34035,40	0,00	132,8	96,1	12,5	33,6
	l1L5	34115,92	34115,74	0,18	127,3	92,2	12,1	33,4
	l1L10	34166,44	34166,42	0,02	131,9	94,5	11,8	33,6
	l0,5L10	34086,88	34086,88	0,00	138,1	97,2	13,5	33,2
	l10L50	34898,24	34898,24	0,00	154,3	110,4	13,4	34,4
	l100L500	37849,56	37849,56	0,00	126,2	92,2	11,9	36,0
	Moyenne	**34495.9**	**34495.4**	**0.50**	**133.6**	**93.8**	**12.5**	**33.8**
5_1min	**l0L0**	33570,20	33569,25	0,95	126,8	82,6	18,7	33,4
	l1L2	34043,70	34043,59	0,11	134,2	97,8	17,4	33,4
	l1L5	34108,06	34107,99	0,07	134,1	100,6	17,8	33,8
	l1L10	34170,22	34170,22	0,00	144,8	104,2	18,4	33,4
	l10L50	35012,36	35012,36	0,00	164,2	114,3	21,4	34,0
	l5L50	34859,20	34859,11	0,09	160,0	116,5	20,1	33,8
	Moyenne	**34294.0**	**34293.8**	**0.20**	**144.0**	**102.7**	**19.0**	**33.6**
5_2min	**l0L0**	29149,28	28789,77	359,51	226,8	133,4	31,3	29,4
	l1L5	30482,38	30172,55	309,83	257,6	153,3	32,9	30,0
	l1L10	30944,64	30706,13	238,51	284,9	154,2	34,7	29,8
	l5L50	34243,40	34225,89	17,51	248,2	153,7	37,3	31,6
	l1L50	33514,64	33485,15	29,49	200,2	135,1	26,1	31,4
	l10L50	34850,08	34849,10	0,98	222,8	147,0	30,5	32,2
	l10L100	35783,96	35783,96	0,00	197,3	135,9	30,5	33,6
	Moyenne	**32709.8**	**32573.2**	**136.5**	**234.0**	**144.6**	**31.9**	**31.1**

5.2.2 Ajout des correspondances

On ajoute maintenant le contrôle des correspondances à la deuxième phase de notre algorithme.

Ne disposant pas des données réelles sur les correspondances, on construit les ensembles $\mathcal{C}$. Pour les instances de 500 voyages, on choisit 50 paires de voyages $(v, v') \in \mathcal{C}$ aléatoirement parmi les couples de voyages ayant lieu au même moment et appartenant à des lignes différentes.

Pour chaque correspondance $c \in \mathcal{C}$, on choisit aléatoirement le décalage optimal Δ_c^* suivant la distribution de probabilité du tableau 5.7. Dans la moitié des cas, $\Delta_c^* = 0$ et donc la correspondance est déjà parfaitement respectées dans la grille horaire initiale.

Tableau 5.7 Distribution de probabilité de Δ_c^*

valeur de Δ_c^*	-2	-1	0	1	2
probabilité	0.05	0.2	0.5	0.2	0.05

On choisit la même fonction de pénalisation f_c pour toutes les correspondances $c \in \mathcal{C}$, avec les paramètres suivants :

$$P_c = 1,\ \alpha_c^+ = \frac{1}{3},\ l_c^+ = 1,\ l_c^- = 0.$$

Il y a donc ici trois cas possibles, selon la valeur de $x_v - x'_v - \Delta_c^*$:

- si $x_v - x'_v - \Delta_c^* = 0$, la pénalité est nulle.
- si $x_v - x'_v - \Delta_c^* = 1$, la pénalité est égale à $\frac{1}{3}$.
- dans tous les autres cas, la correspondance est ratée et la pénalité vaut 1.

Dans le tableau 5.8, on reprend les mêmes instances et paramètres qu'au tableau 5.4 et on ajoute le contrôle des correspondances dans la seconde phase de l'algorithme. On choisit $\alpha_{dec} = 1,\ \alpha_{esp} = 2,\ \alpha_{cor} = 4$.

On ajoute la colonne "Som. cor" qui donne la somme des pénalités sur les correspondances $\sum_{c=(v,v')\in\mathcal{C}} f_c(x_v - x'_v - \Delta_c^*)$, c'est-à-dire ici 1 par correspondance manquée et $\frac{1}{3}$ par correspondance de qualité moyenne. On remarque que l'ajout des pénalités permet de diminuer cette valeur autour de 14. Dans ces exemples, on améliore même la qualité initiale de la grille horaire en ce qui concerne les correspondances (on avait, en effet, 19.1 pour le MDVSP sans

décalage d'horaires).

Tableau 5.8 Comparaison du coût des itinéraires et de la qualité des horaires avec prise en compte des correspondances, pour différents couples de pénalités (l, L). Résultats en moyenne sur 5 problèmes de 500 voyages.

	Coût des itinéraires			**Problème Q**			
Pénalités	**coût tot**	**coût opé.**	**nb. bus**	**Score**	**Som.déc.**	**Som.esp.**	**Som.cor.**
MDVSP	36441	1441	35,0	38.2	0	0	19.1
l0L0	33579,5	1179,5	32,4	2042,3	696,4	574,0	24,7
l0,1L0,5	33635,7	1235,7	32,4	852,8	230,4	247,2	16,0
l1L2	33834,6	1434,6	32,4	360,8	84,8	85,2	13,2
l0,5L5	33875,8	1475,8	32,4	235,1	44,4	41,2	13,5
l1L5	33905,5	1505,5	32,4	223,5	39,2	37,2	13,7
l5L10	34171,4	1771,4	32,4	206,9	32,8	30,8	14,5
l1L10	33941,0	1541,0	32,4	197,2	26,8	27,6	14,4
l0,5L10	33922,1	1522,1	32,4	196,3	25,6	26,4	14,7
l10L50	34232,2	1832,2	32,4	183,3	24,4	22,4	14,3
l100L500	36469,6	1869,6	34,6	145,5	14,0	6,8	14,7

Enfin, pour un choix de pénalités fixé $(l = 0.5, L = 10)$, on analyse les différentes solutions obtenues lorsqu'on fait varier les paramètres α_{dec}, α_{esp} et α_{cor} au tableau 5.9. Cela donne donc un aperçu de l'espace des solutions du problème de choix des horaires (Q) contraint par les itinéraires obtenus par le MDVSP. Les plus petites valeurs des pénalités sur le décalage des voyages (Som. dec.), la modification des espacements (Som. esp.) et les pénalités sur les correspondances (Som. cor.) ont été mises en gras. Si le respect des correspondances est un objectif majeur, $(\alpha_{cor} = 10)$, on peut augmenter le nombre de correspondances respectées, i.e. la somme des pénalités sur les correspondances diminue (meilleure valeur 4.4 pour 50 correspondances à respecter).

5.2.3 Résultats pour une instance réelle

On donne au tableau 5.10 les résultats obtenus pour un problème réel fourni par GIRO contenant 1042 voyages. On utilise l'algorithme heuristique pour le résoudre. Les résultats de la seconde phase sont encore une fois rapportés à la minute, i.e. quand on note 58 minutes de décalage, cela indique que 29 voyages ont été décalés chacun de 2 minutes.

Tableau 5.9 Influence des coefficients α sur la solution de (Q) pour les pénalités **l0,5L10**. Résultats en moyenne sur 5 problèmes de 500 voyages.

α_{dec},	α_{esp},	α_{cor}	**Som. dec.**	**Som. esp.**	**Som. cor.**
1	2	4	25,6	26,4	14,7
1	2	10	61,6	68,0	**4,4**
1	10	10	66,0	18,4	16,0
1	2	0,1	12,4	21,6	19,3
10	1	0,1	**12,0**	22,0	19,1
1	10	0,1	47,6	**14,4**	20,3

Tableau 5.10 Valeurs des solutions pour l'instance réelle de 1042 voyages

		Coût des itinéraires			**Problème** Q		
Copie	**Pénalités**	**coût tot**	**coût opé.**	**nb. bus**	**Score**	**Som.déc.**	**Som.esp.**
1	**MDVSP**	45392.8	1392.8	44	0	0	0
3_2min	**l0L0**	41926.2	926.2	41	2412	1276	568
	l0.5L10	42146.8	91146.8	41	178	58	60

En terme de temps de calcul, on résout le problème avec une seule copie en 2650 secondes et celui avec 3 copies en 4525 secondes. Pour évaluer la qualité des solutions trouvées, la seule valeur dont on dispose est la différence entre la valeur de (PM^{rl}_{pert}) et la solution entière trouvée. Ce "gap" vaut 206 (0.4%) pour le cas avec une copie et 530 (1.2%) pour le cas avec 3 copies **l0L0**. Enfin l'algorithme consacre une grande partie de son temps à résoudre la première relaxation (perturbée) : 85% du temps et 67% du temps pour les problèmes avec une et 3 copies respectivement.

5.3 Contrôle du nombre de lignes par itinéraire

On présente ici une extension du modèle de MDVSP avec ou sans décalage d'horaires pour prendre en compte une contrainte opérationnelle spécifique : la limitation du nombre de lignes couvertes par chaque itinéraire. En effet, en prévision du problème de construction des horaires de chauffeur, il est préférable d'avoir des itinéraires d'autobus qui contiennent un nombre limité de lignes différentes. Si, par exemple, on impose une limite de 2 lignes par itinéraire, alors l'enchaînement

ligne 1 - ligne 2 - ligne 2 - ligne 1 - ligne 2

serait valide (alors qu'il comporte trois changements de ligne).

5.3.1 Modélisation

La contrainte que nous voulons ajouter est une caractéristique spécifique à chaque itinéraire. On peut donc s'assurer qu'elle est vérifiée dès la construction de l'itinéraire dans le sous-problème. Cela peut être réalisé en ajoutant des ressources dans le graphe $\mathcal{G}^d$. Résoudre le sous-problème revient alors à un problème de plus court chemin avec contraintes de ressources.

On note $\mathcal{L} = \{l_1, l_2, \ldots, l_{|\mathcal{L}|}\}$ l'ensemble des lignes associées aux voyages de $\mathcal{V}$. On note de plus M la limite imposée sur le nombre de lignes par itinéraire. Lors de la construction des itinéraires dans le sous-problème, il est nécessaire de garder en mémoire le nom des lignes déjà empruntées. Pour cela, on ajoute dans le graphe $\mathcal{G}^d$ les ressources suivantes :

- $R_{l_1}, R_{l_2}, \ldots R_{l_{|\mathcal{L}|}}$: ces ressources retiennent si, oui ou non, chacune des lignes a déjà été empruntée par l'itinéraire. Ces ressources doivent, à tout moment, être contenues dans l'intervalle $[0, 1]$. Lorsqu'on entre dans le sous-réseau d'un voyage v appartenant à la ligne l, on incrémente la ressource R_l si la ligne l n'a pas déjà été visitée.
- C est une ressource qui compte le nombre de lignes utilisées jusque-là. À chaque noeud, elle doit être contenue dans $[0, M]$.

La figure 5.7 illustre le cas d'un problème avec 3 lignes et une limite de $M = 2$ lignes différentes par itinéraire. Pour simplifier, on a pris ici le cas d'un réseau de connexion avec $i, j, k \in \mathcal{V}$ trois voyages représentés par des nœuds. En dessous de ceux-ci, on note les étiquettes d'un chemin possible.

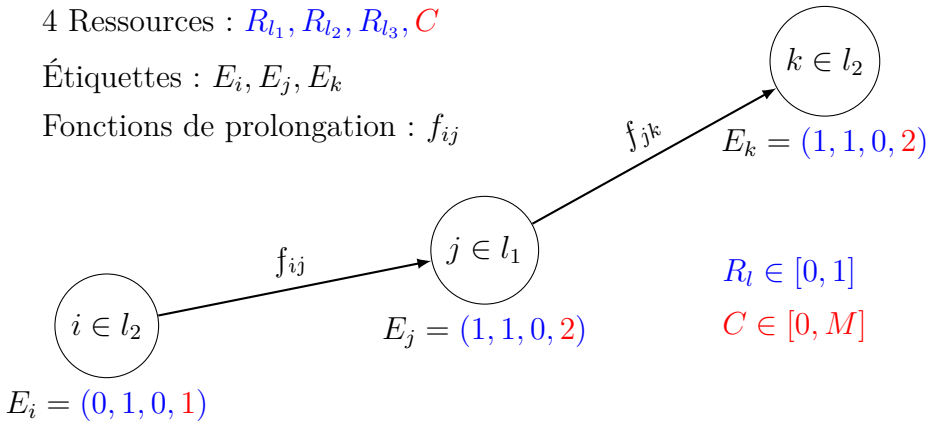

Figure 5.7 Illustration des ressources R_l et C (compteur de lignes visitées)

On définit maintenant les fonctions f_{ij} de prolongation des étiquettes sur chaque arc (i, j). On suppose que j représente un voyage de la ligne $l(j)$. On note $E_i = \left(R^i_{l_1}, \ldots, C^i\right)$ et $E_j = \left(R^j_{l_1}, \ldots, C^j\right)$ les étiquettes d'un chemin jusqu'aux nœuds i et j respectivement. Si on emprunte l'arc (i, j), alors on prolonge l'étiquette E_i ainsi : $E_j = f_{ij}(E_i)$ qui peut se décomposer ainsi :

- pour toute ligne $l \in \mathcal{L}, l \neq l(j)$: $R^j_l = R^i_l$
- pour $l = l(j)$: $R^j_{l(j)} = R^i_{l(j)} + \begin{cases} 1 \text{ si } R^i_{l(j)} = 0 \\ 0 \text{ sinon} \end{cases} \iff R^j_{l(j)} = 1$
- pour la ressource C : $C^j = C^i + \begin{cases} 1 \text{ si } R^i_{l(j)} = 0 \\ 0 \text{ sinon} \end{cases}$

Ainsi la valeur de C est toujours égale au nombre de lignes visitées, ou encore à la somme des valeurs de R_l. La propriété suivante est donc vérifiée à chaque nœud j :

$$C^j = \sum_{l \in \mathcal{L}} R^j_l. \tag{5.52}$$

Définissons maintenant la règle de dominance entre deux étiquettes E^1_j et E^2_j au nœud j provenant des chemins c_1 et c_2. E^1_j domine E^2_j si et seulement si :

$$\forall l \in \mathcal{L} : \; R^{j^1}_l \leq R^{j^2}_l. \tag{5.53}$$

On remarque que (5.53) implique aussi

$$C^{j1} \leq C^{j2}. \tag{5.54}$$

Pour assurer que cette règle de dominance est bien définie, on vérifie que (5.53) implique que toutes les extensions possibles pour le chemin c_2 le sont aussi pour le chemin c_1.

Soit une extension admissible pour le chemin c_2 et soit $v \in \mathcal{V}$ un voyage de la ligne l, le premier voyage visité dans cette extension. On distingue quatre cas :

- soit la ligne l fait partie des chemins c_1 et c_2 et donc le prolongement par v est admissible pour les deux chemins.
- soit l fait partie de c_2 mais pas de c_1 ce qui implique que $C^{j1} < C^{j2}$. Donc il est permis d'ajouter l à la liste des chemins empruntés par c_1.
- soit l ne fait partie ni de c_1 ni de c_2. Comme l'extension est possible pour c_2, on a $M - 1 \geq C^{j2} \geq C^{j1}$, donc elle est aussi possible pour c_1.
- le cas où l fait partie du chemin c_1 mais pas de c_2 est exclu par la propriété (5.53).

On remarque que l'on conserve la propriété (5.53) dans tous les cas. Et donc, par récurrence sur les voyages rencontrés par l'extension du chemin c_2, on prouve que celle-ci est aussi admissible comme extension du chemin c_1.

Pour chacune des ressources, on donne maintenant les détails des bornes et fonctions de prolongation dans le cas d'un graphe avec sous-réseaux de choix de copie. Au nœud source, toutes les ressources sont initialisées à 0. Lorsqu'on passe par le dépôt, on remet également les ressources R_l et C à 0. Il serait intéressant d'étudier le cas contraire, où on ne fait pas cette remise à zéro.

Tableau 5.11 Définition des bornes et fonctions de prolongation pour un graphe avec sous-réseaux de choix de copie

Borne sur les nœuds	$\mathbf{R_l}$	**C**
Source/Attente dépôt	$[0, 0]$	$[0, 0]$
Voyage	$[0, 1]$	$[0, M]$
Contribution sur les arcs (i, j)		
Attente (dépot/inter-copie)	0	0
Sortie du dépôt/ connexion	$f_{ij}(E_i) - E_i$	$f_{ij}(E_i) - E_i$
Retour au dépôt	-1	$-M$

5.3.2 Résultats

On donne les résultats numériques en moyenne sur 5 instances de 300 et 500 voyages dans les tableaux 5.12 et 5.13. Encore une fois, on utilise une génération de colonnes heuristique avec fixation de colonne et perturbation des contraintes de partitionnement. On donne, dans l'ordre la valeur de la solution entière, le nombre d'autobus et le gap entre la solution entière et la relaxation linéaire sans perturbations. On donne ensuite le temps total de calcul en secondes, le temps pour résoudre le premier nœud et le temps consacré à résoudre le sous-problème.

On considère quatre limites différentes sur le nombre de lignes par itinéraire : $M = 1,\ 2,\ 3$ ou $M = +\infty$. On observe que, par rapport au cas classique ($M = +\infty$), la valeur de la solution augmente très peu si on permet 2 ou 3 lignes par trajet. Pour $M = 1$, il peut être nécessaire d'ajouter un véhicule. Cela est probablement dû au fait qu'on remet à zéro le compteur de lignes dès qu'on fait une pause au dépôt. L'algorithme aura donc tendance à proposer des itinéraires qui repassent souvent par le dépôt.

Le temps CPU consacré au sous-problème (S-P) augmente légèrement avec la valeur de M ($M = +\infty$ exclu).

Tableau 5.12 Comparaison des résultats en fonction du nombre de lignes autorisés par itinéraire. Moyenne sur 5 instances de 300 voyages.

	Solution			Temps CPU (s)		
Max nb. lignes	**Sol. Ent.**	**Nb. bus**	**GAP**	**Total**	**Nœud0**	**S-P**
Sans décalage						
1	26428,1	25,2	0	11,8	7,8	1,6
2	25746,4	24,6	0	14,0	9,1	2,1
3	25745,1	24,6	0	17,7	9,9	2,9
$+\infty$	25745,1	24,6	0	1,7	8,9	15,4
Avec décalage						
1	24481,7	23,4	0	20,0	10,7	5,4
2	24384,3	23,4	0,136	21,4	11,9	6,7
3	24382,1	23,4	0,016	21,5	12,4	6,9
$+\infty$	24382,2	23,4	0,048	4,6	11,5	18,9

Tableau 5.13 Comparaison des résultats en fonction du nombre de lignes autorisées par itinéraire. Moyenne sur 5 instances de 500 voyages.

	Solution			Temps CPU (s)		
Max nb. lignes	**Sol. Ent.**	**Nb. bus**	**GAP**	**Total**	**Nœud0**	**S-P**
Sans décalage						
1	36560,7	35	0,00	96,6	52,2	8,2
2	36445,9	35	0,01	87,3	56,3	9,8
3	36442,6	35	0,00	94,4	58,4	11,7
inf	36442,2	35	0,00	102,3	55,9	6,8
Avec décalage						
1	34117,6	32,8	0,07	106,0	67,5	19,4
2	33584,2	32,4	0,18	127,4	77,0	28,1
3	33578,7	32,4	0,22	142,3	85,9	37,4
inf	33578,7	32,4	0,29	121,5	71,6	19,5

CHAPITRE 6 CONCLUSION

6.1 Synthèse du mémoire

Au cours de ce projet, nous avons étudié les problèmes de construction d'itinéraires de véhicule classiques, puis avec décalage d'horaires et enfin, avec contrôle de la qualité des horaires modifiés. Nous avons ainsi exploré les modèles et algorithmes possibles pour résoudre ces problèmes.

Cela nous a conduits à proposer pour modéliser le MDVSP, l'utilisation d'un réseau hybride entre les réseaux de connexions et espace-temps. Ce réseau de connexions mixte donne des temps de calcul équivalents, sinon meilleurs, au réseau espace-temps et conserve en grande partie la flexibilité du réseau de connexions. Il peut ainsi s'adapter facilement à de nombreuses contraintes des compagnies d'autobus.

Pour ajouter le décalage des horaires de départ, on a proposé à nouveau d'hybrider les réseaux de connexions et espace-temps, en introduisant les sous-réseaux de choix de copie. Malgré la réduction du nombre d'arcs ainsi produite, l'algorithme exact de génération de colonnes s'est avéré long. D'une part, l'arbre de branchement devient très vite grand et, d'autre part, la dégénérescence des instances rend chaque nœud de cet arbre difficile à résoudre.

Nous avons donc cherché des stratégies d'accélération de l'algorithme, quitte à perdre l'optimalité des solutions trouvées. L'heuristique qui en a résulté combine la fixation d'une colonne à chaque nœud de l'arbre et la perturbation des contraintes de partitionnement. Malgré la simplicité de la méthode de branchement utilisée, les résultats ainsi obtenus sont très proches de l'optimalité.

Enfin, s'agissant du contrôle des horaires après décalage, notre stratégie heuristique en deux phases permet de trouver rapidement de bonnes solutions sans ralentir l'algorithme de résolution du MDVSP avec décalage d'horaires. On peut ainsi obtenir un équilibre entre le coût des itinéraires d'autobus et la qualité des horaires finaux.

6.2 Prolongements possibles

Les techniques de réduction de la taille des réseaux, de perturbations des contraintes et de branchement heuristique ont permis de réduire très significativement les temps de calculs. Cependant, il serait intéressant d'étudier la taille maximale des problèmes que notre algorithme peut résoudre en des temps jugés raisonnables. Plusieurs idées de réduction du temps de calcul peuvent être explorées :

- Il est, d'une part, possible de réduire davantage la taille des réseaux et ainsi faciliter la résolution du sous-problème. Une possibilité est d'agréger les sous-réseaux de choix de copie en suivant la même idée que l'agrégation des nœuds au dépôt. Par exemple, pour deux copies consécutives reliées par un arc attente, si aucun arc de connexion ne sort de la première copie, alors on peut regrouper les deux nœuds-copies ensemble. On pourrait par la suite aussi enlever certains arcs de connexions multiples et donc réduire la symétrie du problème. Ces réductions supplémentaires pourraient s'avérer particulièrement utiles dans les cas où le nombre de copies ou de voyages est grand.
- Certaines techniques de réduction de la dégénérescence pourraient être appliquées telles que l'agrégation dynamique de contraintes ou la génération de colonnes stabilisée. Il pourrait également être profitable d'explorer d'autres stratégies heuristiques d'accélération, par exemple, en éliminant certains arcs ou en arrêtant prématurément chaque nœud de résolution.

Il serait très intéressant de mieux comprendre les interactions entre les deux phases de l'algorithme de résolution du MDVSP avec décalage contrôlé des horaires. On peut souhaiter, en effet, trouver un moyen de choisir les pénalités (l, L) qui s'adapte aux exigences du problème de détermination des horaires. Une étude plus précise du lien entre l'algorithme en deux phases et la relation lagrangienne pourrait aider en ce sens.

Enfin, une extension possible, mais complexe, de ce travail consisterait à intégrer la construction des horaires de chauffeurs au MDVSP avec décalage d'horaires.

RÉFÉRENCES

P. Benchimol, G. Desaulniers, et J. Desrosiers, “Stabilized dynamic constraint aggregation for solving set partitioning problems”, *European Journal of Operational Research*, vol. 223, no. 2, pp. 360 – 371, 2012.

A. A. Bertossi, P. Carraresi, et G. Gallo, “On some matching problems arising in vehicle scheduling models”, *Networks*, vol. 17, no. 3, pp. 271–281, 1987.

N. Bélanger, G. Desaulniers, F. Soumis, et J. Desrosiers, “Periodic airline fleet assignment with time windows, spacing constraints, and time dependent revenues”, *European Journal of Operational Research*, vol. 175, no. 3, pp. 1754 – 1766, 2006.

S. Bunte et N. Kliewer, “An overview on vehicle scheduling models”, *Public Transport*, vol. 1, no. 4, pp. 299–317, 2009.

G. Carpaneto, M. Dell’amico, M. Fischetti, et P. Toth, “A branch and bound algorithm for the multiple depot vehicle scheduling problem”, *Networks*, vol. 19, no. 5, pp. 531–548, 1989.

A. A. Ceder, “Optimal multi-vehicle type transit timetabling and vehicle scheduling”, *Procedia-Social and Behavioral Sciences*, vol. 20, pp. 19–30, 2011.

G. Desaulniers et M. D. Hickman, “Public transit”, *Handbooks in Operations Research and Management Science*, vol. 14, pp. 69–127, 2007.

G. Desaulniers, J. Lavigne, et F. Soumis, “Multi-depot vehicle scheduling problems with time windows and waiting costs”, *European Journal of Operational Research*, vol. 111, no. 3, pp. 479–494, 1998.

J. Desrosiers, Y. Dumas, M. M. Solomon, et F. Soumis, “Chapter 2 time constrained routing and scheduling”, *Handbooks in Operations Research and Management Science*, vol. 8, pp. 35 – 139, 1995.

O. du Merle, D. Villeneuve, J. Desrosiers, et P. Hansen, “Stabilized column generation”, *Discrete Mathematics*, vol. 194, no. 1–3, pp. 229 – 237, 1999.

C. Fleurent et R. Lessard, “Integrated timetabling and vehicle scheduling”, GIRO Inc.Montreal, Canada., Rapp. tech., 2009.

V. Gintner, N. Kliewer, et L. Suhl, “Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice”, *OR Spectrum*, vol. 27, no. 4, pp. 507–523, 2005.

M. Groiez, G. Desaulniers, A. Hadjar, et O. Marcotte, “Separating valid odd-cycle and odd-set inequalities for the multiple depot vehicle scheduling problem”, *EURO Journal on Computational Optimization*, vol. 1, no. 3, pp. 283–312, 2013.

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

P. C. Guedes, W. P. Lopes, L. R. Rohde, et D. Borenstein, “Simple and efficient heuristic approach for the multiple-depot vehicle scheduling problem”, *Optimization Letters*, vol. 10, no. 7, pp. 1449–1461, 2016.

V. Guihaire et J.-K. Hao, “Transit network design and scheduling : A global review”, *Transportation Research Part A : Policy and Practice*, vol. 42, no. 10, pp. 1251 – 1273, 2008.

V. Guihaire et J.-K. Hao, “Transit network timetabling and vehicle assignment for regulating authorities”, *Computers & Industrial Engineering*, vol. 59, no. 1, pp. 16 – 23, 2010.

A. Hadjar et F. Soumis, “Dynamic window reduction for the multiple depot vehicle scheduling problem with time windows”, *Computers & Operations Research*, vol. 36, no. 7, pp. 2160–2172, 2009.

A. Hadjar, O. Marcotte, et F. Soumis, “A branch-and-cut algorithm for the multiple depot vehicle scheduling problem”, *Operations Research*, vol. 54, no. 1, pp. 130–149, 2006.

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

O. J. Ibarra-Rojas, R. Giesen, et Y. A. Rios-Solis, “An integrated approach for timetabling and vehicle scheduling problems to analyze the trade-off between level of service and operating costs of transit networks”, *Transportation Research Part B : Methodological*, vol. 70, pp. 35 – 46, 2014.

S. Irnich et G. Desaulniers, “Shortest path problems with resource constraints”, dans *Column generation*, G. Desaulniers, J. Desrosiers, et M. M. Solomon, éds. Springer, 2005, pp. 33–65.

S. Irnich, G. Desaulniers, J. Desrosiers, et A. Hadjar, “Path-reduced costs for eliminating arcs in routing and scheduling”, *INFORMS Journal on Computing*, vol. 22, no. 2, pp. 297–313, 2010.

A. Kéri et K. Haase, “Simultaneous vehicle and crew scheduling with trip shifting”, dans *Operations Research Proceedings 2007*. Springer Berlin Heidelberg, 2008, pp. 467–472.

N. Kliewer, S. Bunte, et L. Suhl, “Time windows for scheduled trips in multiple depot vehicle scheduling”, dans *Proceedings of the EWGT2006 joint conferences*, 2006, pp. 340–346.

N. Kliewer, T. Mellouli, et L. Suhl, “A time–space network based exact optimization model for multi-depot bus scheduling”, *European Journal of Operational Research*, vol. 175, no. 3, pp. 1616 – 1627, 2006.

N. Kliewer, B. Amberg, et B. Amberg, “Multiple depot vehicle and crew scheduling with time windows for scheduled trips”, *Public Transport*, vol. 3, no. 3, pp. 213–244, 2012.

G. Laporte, F. A. Ortega, M. A. Pozo, et J. Puerto, “Multi-objective integration of timetables, vehicle schedules and user routings in a transit network”, *Transportation Research Part B : Methodological*, vol. 98, pp. 94 – 112, 2017.

B. Laurent et J.-K. Hao, “Iterated local search for the multiple depot vehicle scheduling problem”, *Computers & Industrial Engineering*, vol. 57, no. 1, pp. 277 – 286, 2009.

A. Löbel, “Vehicle scheduling in public transit and Lagrangean pricing”, *Management Science*, vol. 44, no. 12-part-1, pp. 1637–1649, 1998.

Z.-g. Liu et J.-s. Shen, “Regional bus operation bi-level programming model integrating timetabling and vehicle scheduling”, *Systems Engineering-Theory & Practice*, vol. 27, no. 11, pp. 135–141, 2007.

A. Oukil, H. B. Amor, J. Desrosiers, et H. El Gueddari, “Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems”, *Computers & Operations Research*, vol. 34, no. 3, pp. 817–834, 2007.

A.-S. Pepin, G. Desaulniers, A. Hertz, et D. Huisman, “A comparison of five heuristics for the multiple depot vehicle scheduling problem”, *Journal of Scheduling*, vol. 12, no. 1, pp. 17–30, 2009.

H. L. Petersen, A. Larsen, O. B. Madsen, B. Petersen, et S. Ropke, “The simultaneous vehicle scheduling and passenger service problem”, *Transportation Science*, vol. 47, no. 4,

pp. 603–616, 2012.

C. C. Ribeiro et F. Soumis, “A column generation approach to the multiple-depot vehicle scheduling problem”, *Operations Research*, vol. 42, no. 1, pp. 41–52, 1994.

A. Schöbel, “An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation”, *Transportation Research Part C : Emerging Technologies*, vol. 74, pp. 348 – 365, 2017.

V. Schmid et J. F. Ehmke, “Integrated timetabling and vehicle scheduling with balanced departure times”, *OR Spectrum*, vol. 37, no. 4, pp. 903–928, 2015.

A. Van den Heuvel, J. Van Den Akker, et M. Van Kooten, “Integrating timetabling and vehicle scheduling in public bus transportation”, *Reporte Técnico UU-CS-2008-003, Department of Information and Computing Sciences, Utrecht University, Holanda*, 2008.