PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Optimisation de tournées de véhicules avec contrainte de fragilité
Auteur: Author:	Clément Altman
Date:	2017
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Altman, C. (2017). Optimisation de tournées de véhicules avec contrainte de fragilité [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/2556/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/2556/
Directeurs de recherche: Advisors:	Guy Desaulniers, & Fausto Errico
Programme: Program:	Maîtrise recherche en mathématiques appliquées

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

OPTIMISATION DE TOURNÉES DE VÉHICULES AVEC CONTRAINTE DE FRAGILITÉ

CLÉMENT ALTMAN
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)
AVRIL 2017

© Clément Altman, 2017.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

OPTIMISATION DE TOURNÉES DE VÉHICULES AVEC CONTRAINTE DE FRAGILITÉ

présenté par : ALTMAN Clément
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. ERRICO Fausto, Ph. D., membre et directeur de recherche
Mme CHERKESLY Marilène, Ph. D., membre

DÉDICACE

A Laurence, Paul, Cherif et Mohammed pour notre ambiance de bureau.

A Adrien, Mathieu, Lucie, Greta, Luciano, Paul, Giulia, Chris, Arnaud pour ces moments au GERAD et en dehors.

A Guy Desaulniers et Fausto Errico pour leur patience, leur soutien et tout ce qu'ils m'ont appris.

A mes parents, pour le cliché.

REMERCIEMENTS

Je remercie mes directeurs pour leur appui financier tout au long de ma maîtrise.

Je tiens à remercier toutes les personnes au GERAD, au département et à l'Ecole Polytechnique de Montréal qui m'ont permis d'accomplir cette maîtrise que ce soit par leur aide matériel, les opportunités d'expériences qu'ils m'ont données, les conseils apportés ou à avoir fait de mon passage à Montréal un moment important dans ma vie.

Des remerciements sans limite pour François Lessard et Benoit Rochefort qui, au delà de leur aide indispensable à ma maîtrise, m'ont permis de maîtriser un savoir informatique des plus utile et transmis une envie pour développer mes compétences dans ce domaine.

RÉSUMÉ

Dans le cadre du transport maritime, les marchandises sont empilées verticalement selon leur ordre de collecte ou de livraison. La nature des marchandises peut être très différente, par exemple dans le cas du ravitaillement des villes dans le nord du Canada, il faut amener des voitures, de l'essence, des médicaments ou du matériel de construction. Ces différentes marchandises n'ont pas la même masse et il faut donc faire attention à la manière dont sont superposées ces marchandises. Des barils d'essence transportés sur une voiture risquent de rendre la voiture inutilisable et les bidons d'essence aussi par la même occasion. Ces problèmes de marchandises légères et lourdes ne pouvant être superposées se retrouvent également lors du chargement de remorques à partir de chariots élévateurs amenant des palettes.

Ce mémoire cherche à améliorer la construction des tournées de véhicules en tenant compte de cette contrainte de fragilité. En partant du problème connu de tournées de véhicules avec fenêtres de temps et de capacité, ce mémoire va développer des méthodes pour respecter la contrainte de fragilité et l'intégrer aux algorithmes de type *Branch-Price-and-Cut* existants le plus efficacement possible. La plupart des modifications se feront sur les algorithmes d'étiquetage durant la génération des routes. Il convient que les algorithmes doivent être capables de résoudre la plupart des instances classiques pour que la solution proposée soit applicable dans la vie réelle. À cette fin, des tests seront menés sur des instances classiques et interprétés.

Les méthodes proposées dans les chapitres 4 et 5 permettent de générer des routes respectant la contrainte de fragilité en regardant uniquement la séquence d'entrée des marchandises et sans s'intéresser aux dispositions possibles dans la cale. Ces résultats théoriques sont ensuite intégrés à un algorithme existant afin de construire des algorithmes respectant la contrainte de fragilité. Le chapitre 4 se concentre sur le cas des piles de hauteur 2 tandis que le chapitre 5 s'intéresse au cas général, sans connaitre la hauteur des piles a priori. Les résultats théoriques permettent d'obtenir des algorithmes efficaces comme le montrent les résultats expérimentaux obtenus dans les chapitres 4 et 5. Ces résultats expérimentaux sont des tests effectués avec nos algorithmes et une analyse des résultats selon différents paramètres. Ces analyses mettront en valeur l'intérêt et les spécificités de notre problème. Des comparaisons seront également effectuées avec d'autres algorithmes pour montrer la pertinence de notre recherche

ABSTRACT

In maritime merchandise transport, goods are stocked vertically according to their pick-up or delivery order. The nature of the goods can differ and sometimes heavy items cannot be put above light items. For instance, if you are carrying iron and porcelain, you do not want the porcelain to be under the iron. Building routes that respect this condition is not an easy task and an algorithm that would take care of this fragility constraint would be very useful.

This research aims at developping methods to avoid violating the fragility constraint for new routes to include in the problem. Starting from the classic vehicle routing problem, we will develop Branch-Price-and-Cut algorithms that respect time windows and capacity constraints but also respect the fragility one. Such algorithms should be efficient to solve a majority of the instances in a reasonable amount of time.

Theoretical results from chapters 4 and 5 would let us build route without a fragility problem just by knowing the order of the goods collected. Chapter 4 focus on vehicles with two floors while chapter 5 takes the number of floor as a parameter. With those results, we build algorithms that respect the fragility constraint and solve our route-building problem. To show the efficiency of our algorithms, we will conduct some tests and analyse the results according to different parameters. This analysis will show the strength of our results and the specificity of our problem.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

VRPTW	Vehicle Routing Problem with Time Windows
VRPTWFC	Vehicle Routing Problem with Time Windows and Fragilitiy Constraint
ESPPRC	Elementary Shortest Path Problem with Ressource Constraint
BPC	*Branch-Price-and-Cut*

CHAPITRE 1 INTRODUCTION

Le problème de tournées de véhicules avec fenêtres de temps (VRPTW pour *Vehicle Routing Problem with Time Windows*) est un problème majeur de la recherche opérationnelle. Il consiste à déterminer un ensemble de routes permettant de collecter ou de livrer des marchandises à des clients en respectant des fenêtres de temps et des contraintes de capacité. Les clients sont généralement visités une et une seule fois par un véhicule, leur demande doit être satisfaite par cette seule visite. Le VRPTW a été très étudié et de nombreux algorithmes de résolution ont été proposés mais la contrainte de fragilité est encore peu étudiée.

En effet, dans le transport maritime, les marchandises collectées ou livrées sont parfois de natures différentes et la superposition de ces marchandises peut entrainer une dégradation de la qualité de la marchandise inférieure (de la porcelaine stockée sous du titane, par exemple). Les compagnies doivent construire leurs tournées en respectant des fenêtres de temps (dans le nord du Canada le gel et le dégel des glaces), de capacité (les bateaux ne peuvent pas transporter une quantité infinie de marchandises) et de fragilité. Ce problème de fragilité se retrouve également dans les entrepôts où la marchandise est superposée par palette sur une remorque de transport. Les palettes sont amenées individuellement par des chariots élévateurs et sont généralement superposées par 2 ou 3 dans la remorque de transport.

Il est possible d'éviter ce genre de problème en utilisant des véhicules plus grand ou en interdisant de traiter des marchandises de natures différentes mais ces solutions ne sont pas toujours réalisables. Il est donc important de proposer une solution où la capacité des véhicules est fixe et qui minimise le coût total tout en respectant la contrainte de fragilité. Ce mémoire propose deux algorithmes de type *Branch-Price-and-Cut* (BPC) pour le problème de tournées de véhicules avec fenêtres de temps et contrainte de fragilité (VRPTWFC pour *Vehicle Routing Problem with Time Windows and Fragility Constraint*).

1.1 Définitions et concepts de base

Dans notre modélisation du VRPTWFC, des véhicules situés à un dépôt sont utilisés pour collecter des marchandises chez des clients avant de revenir au dépôt. Les clients ne peuvent être visités que pendant des intervalles de temps fixes, mais il est possible d'arriver avant le début de l'intervalle et d'attendre. Tous les véhicules sont considérés identiques et ont

une capacité limitée. Les véhicules sont constituées de piles verticales toutes identiques, la capacité totale étant répartie de manière égale entre chaque pile.

Un client est caractérisé par une localisation précise, une fenêtre de temps et un nombre entier de marchandises à collecter (les marchandises sont emballées dans des contenants de même dimension comme des palettes ou des conteneurs). Les marchandises collectées à un client peuvent être réparties dans les différentes piles d'un même véhicule mais une unité de marchandise ne peut pas être divisée. Par exemple, si un client possède deux marchandises à collecter, soit ces deux marchandises sont mises dans une même pile, soit une première marchandise est mise dans une première pile et la seconde dans une seconde pile. Par contre, il est interdit de mettre une demie marchandise dans une première pile et une marchandise et demie dans une deuxième pile. Un client ne peut être visité que par un seul véhicule qui doit impérativement collecter toutes les marchandises. Pour un client, toutes les marchandises sont de même nature.

Une autre caractéristique des véhicules est la manière de remplir les piles. Les marchandises «tombent» dans une pile, c'est-à-dire qu'une marchandise mise dans une pile se retrouve forcément juste au dessus de la marchandise placée précédemment dans la même pile. Il ne peut donc pas y avoir de «trou» dans une pile et les marchandises sont réparties selon leur ordre de collecte (voir figure 1.1). De plus une fois qu'une marchandise a été placée, elle ne peut plus être déplacée même à l'intérieur de la pile, sa place est figée pour le restant de la route.

Les marchandises collectées peuvent être de deux types : lourdes (H pour *heavy*) ou légères (L pour *light*). La contrainte de fragilité interdit d'avoir une marchandise légère située en dessous d'une marchandise lourde dans une même pile. Une telle situation est dite problématique et une route avec une situation problématique est donc interdite (voir figures 1.2 et 1.3).

H	H	
H		L
	L	L

Figure 1.1 Répartition à «trous»

Dans cette modélisation, et dans la suite de ce mémoire, nous nous intéressons uniquement à une collecte de marchandises et ne parlons jamais du cas de la livraison. Tous les résul-

H		
H	H	L
H	L	L

Figure 1.2 Situation problématique

H		
H	L	L
H	H	L

Figure 1.3 Répartition respectant la contrainte de fragilité

tats sont équivalents, la livraison fonctionne exactement comme une collecte en inversant le concept des marchandises lourdes et légères.

1.2 Éléments de la problématique

La difficulté liée à la contrainte de fragilité est de construire une solution ne contenant pas de route avec des situations problématiques. La découpe en pile du véhicule et les possibilités de répartition des marchandises collectées donnent un grand nombre de possibilités pour disposer les marchandises. Pour une même séquence de clients, il y a plusieurs possibilités de répartir les marchandises collectées. La difficulté consiste à trouver parmi toutes ces possibilités, au moins une qui respecte la contrainte de fragilité. De plus, s'il n'y a aucune possibilité de répartir les marchandises en respectant la contrainte de fragilité, il faudra, a priori, tester toutes les répartitions possibles pour le prouver ce qui est coûteux en temps. Il y a donc un réel intérêt à détecter rapidement si une route ne peut pas respecter la contrainte de fragilité ou à pouvoir construire de manière certaine des routes la respectant.

Une des méthodes les plus efficace pour résoudre le VRPTW est l'utilisation d'algorithme de type BPC qui nécessite de générer des routes respectant les contraintes du problème associées aux routes. Dans les algorithmes de type BPC appliqués au problème du VRPTWFC, nous allons devoir générer des routes respectant la contrainte de fragilité. Le temps passé à générer ces routes est déterminant dans la performance de l'algorithme, d'où la nécessité de posséder une procédure générant des routes respectant les fenêtres de temps, de capacité et de fragilité en un temps raisonnable. Les algorithmes à étiquettes permettent actuellement de générer des routes respectant les fenêtres de temps et de capacité de manière efficace ; il est intéressant d'essayer de les adapter afin d'intégrer la contrainte de fragilité sans trop

ralentir la vitesse de génération des routes.

Pour les algorithmes à étiquettes, des méthodes d'accélération ont été développées telles que l'étiquetage bidirectionnel. Ces méthodes réduisent grandement le nombre d'étiquettes générées mais rien n'indique qu'elles soient compatibles avec la contrainte de fragilité. Il est donc important d'en tenir compte lors de la construction de nouveaux algorithmes et voir s'il est possible de les intégrer dans les algorithmes développés.

1.3 Objectifs de recherche

Le but de la recherche est de produire deux algorithmes de type BPC permettant de résoudre le VRPTWFC : le premier dans le cas où les piles sont de hauteur 2 et le second dans le cas général, où la taille des piles peut être plus grande que 2. Ces deux algorithmes devront permettre d'utiliser la plupart des stratégies d'accélération des algorithmes à étiquettes, notamment l'étiquetage bidirectionnel.

Afin de tester la performance de ces algorithmes, cette recherche va également proposer une construction d'instances impliquant des marchandises lourdes et des marchandises légères. Ces instances sont basées sur les instances de *Solomon (1987)* pour le problème de tournées de véhicules avec fenêtre de temps. Pour montrer l'efficience de l'algorithme proposé dans le cas des piles de hauteur 2, la performance de notre algorithme sera comparée à celle d'un algorithme plus basique dans sa gestion de la contrainte de fragilité.

Il y a peu de littérature concernant le VPRTWFC, le problème ayant été relativement peu abordé jusqu'à récemment. Nos travaux proposent donc des résultats qui n'ont pour l'instant, à notre connaissance, pas été publié. Ce mémoire proposent des méthodes permettant aux algorithmes de type BPC de résoudre le VRPTWFC de manière efficace, sans lesquelles ces algorithmes seraient inutilisables.

1.4 Plan du mémoire

Le mémoire se découpe en cinq parties. Le chapitre 2 est une revue de littérature qui recense les principaux travaux sur les problèmes de tournées de véhicules et les méthodes de réso-

lution. Le chapitre 3 est une définition générale du problème de VRPTWFC ainsi qu'une présentation d'un algorithme de type BPC pour le VRPTW qui sera modifié dans le cadre de cette recherche. Le chapitre 4 se concentre sur le cas des piles de hauteur 2 ; on y présente un résultat théorique, l'algorithme qui en découle et ses performances expérimentales. Le cinquième et dernier chapitre considère le cas général avec, à nouveau, un résultat théorique, l'algorithme qui en découle et ses performances expérimentales. Enfin, la sixième partie est la conclusion de cette recherche reprenant les résultats importants ainsi que l'originalité des méthodes développées.

CHAPITRE 2 REVUE DE LITTÉRATURE

Les problèmes de tournées de véhicule ont été étudiés maintes et maintes fois et plusieurs méthodes de résolution (algorithme de type BPC, metaheuristique, ...) ont été développées. Ils ont d'ailleurs fait l'objet de plusieurs livres tels que *Golden et al. (2010); Toth and Vigo (2014)*. Dans cette revue de littérature, nous allons présenter les meilleurs algorithmes de résolution connus pour le VRPTW.

2.1 Problème de tournées de véhicules avec fenêtres de temps

Le VRPTW est une extension du problème de tournées de véhicules avec capacité et est étudié comme problème mathématiques depuis les années 1970. Les méthodes de résolution se rangent en deux grandes catégories : les méthodes exactes se concentrent sur la minimisation de la distance totale parcourue et les méthodes heuristiques cherchent en premier à minimiser le nombre de véhicules utilisés puis la distance totale parcourue. Les articles de synthèse de *Vidal et al. (2013); Archetti and Speranza (2014); Desaulniers et al. (2014)* récapitule les algorithmes exacts et heuristiques récents pour le VRPTW.

2.1.1 Algorithmes heuristiques

Les méthodes heuristiques cherchent à trouver une bonne solution, généralement pas la solution optimale, soit en construisant une solution, soit en se déplaçant à partir d'une solution connue. Ces déplacements peuvent se faire dans un voisinage restreint (*2-opt, échange croisé, ...*), (voir *Bräysy and Gendreau (2005)*) ou dans un voisinage large, voir (*Ahuja et al. (2002)*). A ces déplacements s'ajoute une stratégie de recherche basée sur une ou plusieurs solutions (*single trajectory search* et *population-based search*). Dans le cas des métaheuristiques, certains déplacements peuvent amener à des solutions non-réalisables ou n'améliorant pas la solution courante mais la succession de ces déplacements va permettre d'arriver à une solution réalisable meilleure. En combinant plusieurs déplacements avec une stratégie de recherche, un algorithme heuristique est formé ; ainsi de nombreux algorithmes heuristiques ont été développés. Chacun a permis d'améliorer les résolutions des instances de *Solomon (1987)* et de *Gehring and Homberger (2001)*. Le meilleur algorithme présentement est celui de *Vidal et al. (2013)* et les comparaisons entre les algorithmes peuvent être retrouvées dans, par exemple, *Desaulniers et al. (2014)*.

Cette recherche se focalisant sur les algorithmes exacts, les algorithmes heuristiques ne seront plus mentionnés dans la suite de ce mémoire.

2.1.2 Algorithmes exacts

Actuellement, il existe trois grandes familles d'algorithmes pour résoudre le VRPTW, les deux premières reposent sur un arbre de branchement tandis que la troisième réduit le nombre de variables avant de résoudre directement le problème.

La première famille est composée d'algorithmes de type *Branch-and-Cut* (*Padberg and Rinaldi (1987)*) qui combinent énumération implicite et ajout dynamique de plans coupants. A chaque noeud, la relaxation linéaire est résolu et des plans coupants sont ajoutés pour renforcer la relaxation linéaire avant de la résoudre à nouveau. S'il n'est plus possible d'ajouter des plans coupants, un branchement est effectué sur les valeurs fractionnaires de la solution. Ces algorithmes reposent sur les coupes exploitées pour renforcer la relaxation linéaire, introduites au fur et à mesure par *Bard et al. (2002)* puis *Lysgaard (2006)* et enfin *Kallehauge et al. (2007)*.

La seconde famille fut introduite par *Desrochers et al. (1992)* et correspond à des algorithmes de type BPC. Ces algorithmes consistent en un arbre de branchement où les relaxations linéaires sont résolues par la méthode de génération de colonnes, auxquelles sont ajoutés des plans coupants. La génération de colonnes permet de résoudre un problème dit "maitre" sans énumérer explicitement toutes ses variables. A chaque itération de l'algorithme, un problème maitre dit "restreint" car il ne comprend qu'un sous-ensemble des variables du problème maitre est résolu. A partir des variables duales de la solution obtenue, de nouvelles variables de coût réduit négatif sont ajoutées à l'aide d'un sous-problème puis le problème maitre "restreint" est résolu à nouveau. S'il n'est pas possible de générer de nouvelles variables alors la solution obtenue pour le problème maitre "restreint" est optimale pour le problème "maitre". Dans notre cas, la difficulté de cette méthode est de générer de nouvelles variables et à cette fin différentes stratégies ont été développées (voir section 2.2).

La dernière famille, plus récente et introduite par *Baldacci et al. (2012)*, est basée sur une énumération de routes. Cette méthode consiste à fixer un grand nombre de variables à 0 en utilisant des bornes trouvées sur les problèmes primal et dual. La borne duale est trouvée par génération de colonnes combinée à une méthode d'ascension duale tandis que la borne

supérieure est trouvée par une heuristique. En utilisant ces bornes, toutes les routes à coûts réduits plus petit que le saut entre les bornes sont énumérées par un algorithme de programmation dynamique. Enfin un programme en nombre entier contenant toutes les routes énumérées est résolu de manière exacte avec un solveur de type CPLEX ou GUROBI, par exemple. La méthode fonctionne bien si le saut entre les bornes est suffisamment faible.

Ces trois familles ont chacune donné des algorithmes pour résoudre la plupart des instance de *Solomon (1987)*. Le meilleur d'entre eux est actuellement l'algorithme de type BPC développé par *Pecin et al. (2016)* qui résout les 56 instances de *Solomon (1987)* et 51 des instances à 200 clients de *Gehring and Homberger (2001)*.

2.2 Algorithme de type *Branch-and-Price-and-Cut* pour le VRPTW

Les algorithmes proposés dans la suite de ce mémoire sont de type *Branch-and-Price-and-Cut*. Nous passons donc en revue les différentes stratégies utilisées actuellement dans ces algorithmes pour le VRPTW.

2.2.1 Recherche du plus court chemin élémentaire

La recherche du plus court chemin élémentaire avec contraintes de ressource (ESPPRC pour Elementary Shortest Path Problem with Resource Constraints) dans le sous-problème est *NP-difficile* et généralement résolue par programmation dynamique (*Irnich and Desaulniers (2005)*). Dans un tel algorithme, des étiquettes sont propagées jusqu'à chaque noeud représentant les différents chemins pour arriver à ce noeud. Afin d'éviter la multiplication des étiquettes, des règles de dominance sont appliquées (*Desaulniers et al. (1998)*).

Afin d'accélérer ce type d'algorithme, *Righini and Salani (2008)* proposent un algorithme bidirectionnel qui propagent les étiquettes dans les deux sens depuis le noeud source et depuis le noeud puits. Les étiquettes avant et arrière sont ensuite fusionnées pour former une route, ce qui permet de réduire grandement le nombre d'étiquettes générées. A cela, *Desaulniers et al. (2008)* ont ajouté une heuristique de recherche tabou permettant de trouver très rapidement en un certain nombre d'itérations des colonnes intéressantes à partir des colonnes déjà utilisées.

Pour réduire le temps de calcul dédié au sous-problème, une stratégie consiste à relâcher la

contrainte d'élémentarité pour ainsi permettre de générer des routes avec des cycles. Pour éliminer ces cycles, *Irnich and Villeneuve (2006)* proposent d'interdire les cycles de longueur inférieure à k. *Desaulniers et al. (2008)* proposent une relaxation partielle de la contrainte d'élémentarité. Enfin, *Baldacci et al. (2012)* introduisent les *ng-routes* qui interdisent, en quelque sorte, les cycles entre clients proches.

2.2.2 Stabilisation des variables duales

La génération de colonnes permet d'approcher rapidement la valeur optimale du problème maitre mais peut être lente à l'atteindre. En effet, proche de la valeur optimale, il y a souvent de la dégénérescence et il devient de plus en plus difficile de générer de nouvelles variables avec un coût réduit négatif qui permettent d'améliorer la valeur de l'objectif. Pour éviter ce problème, il est possible de stabiliser les variables duales et atteindre l'objectif plus rapidement. Différentes méthodes ont été proposées par *Rousseau et al. (2007)* avec une méthode de points intérieurs, *Kallehauge et al. (2006)* avec une relaxation de type Lagrangienne et *Benchimol et al. (2012)* qui couplent la stabilisation duale avec une méthode d'agrégation dynamique des contraintes.

2.2.3 Plans coupants

Les premières coupes introduites furent les coupes de capacité par *Laporte et al. (1986)* suivies par les coupes 2-chemins par *Kohl et al. (1999)*. A ces premières inégalités, *Jepsen et al. (2008); Pecin et al. (2016)* ont rajouté les inégalités correspondant aux coupes de Chvàtal-Gomory, permettant une meilleure borne inférieure dans l'arbre de branchement.

2.2.4 Stratégie de branchement

Trois principales stratégies de branchement sont utilisées dans la génération des colonnes. les deux premières consistent à brancher sur le flot total d'un arc entre deux clients (*Desaulniers et al. (1998)*) ou sur le flot sortant d'un sous-ensemble de clients (*Pecin et al. (2016)*). La troisième option est de brancher sur le nombre de véhicules utilisés (*Desrochers et al. (1992)*).

2.3 Problème d'empilement et contrainte de fragilité

De nombreux articles modélisent la disposition des marchandises dans le véhicule pendant la construction d'une route. Cette modélisation permet de respecter une ou des contraintes liées aux marchandises collectées dans le problème considéré. Par exemple, dans *Cherkesly et al. (2015)*, une contrainte de premier entré dernier sorti est intégré en ajoutant des ressources pour préciser si une marchandise est au sommet d'une pile ou non. Ainsi, les marchandises qui ont le droit d'être livrées sont connues. Un autre exemple peut être trouvé dans *Schyns and Lurkin (2013)* où une distance doit être respectée entre des marchandises dangeureuses et le reste des marchandises.

Les contraintes de fragilité ont été étudiées, surtout d'un point de vue heuristique, une étude récapitulative peut être trouvée dans *Pollaris et al. (2015)*. Les premiers à s'être intéressés à la contrainte de fragilité telle que nous la décrivons sont *Gendreau et al. (2006)* qui ajoutent la contrainte de fragilité à un problème de collecte avec capacité et de disposition en trois dimensions (3L-CVRP). Leur méthode de résolution est une recherche tabou, la contrainte de fragilité étant traitée en interdisant des mouvements qui conduiraient à une route ne la respectant. Ils résolvent des instances allant jusqu'à 100 clients. Ces résultats ont été améliorés par *Tarantilis et al. (2009); Fuellerer et al. (2010)* mais toujours par des méthodes heuristiques. Un article récent (*Chabot et al. (2016)*) sur le transport de marchandises dans un entrepôt considère une seule pile. Les marchandises "fragiles" ne peuvent pas être en dessous d'un volume trop important de marchandises. Les auteurs proposent un algorithme exact de type BPC où la contrainte de fragilité est traitée en rajoutant une coupe dès qu'une route ne la respecte pas. Notre travail se démarque car le contrainte de fragilité est traitée lors de la construction des routes au lieu d'être vérifiée a posteriori.

CHAPITRE 3 FORMULATION MATHÉMATIQUE ET ALGORITHME BPC GÉNÉRIQUE

Dans ce chapitre, nous introduisons une formulation mathématique pour le VRPTWFC. Ensuite, nous présentons un algorithme BPC générique pour résoudre le VRPTW qui sera modifié dans les prochains chapitres pour traiter deux cas du VRPTWFC.

3.1 Notations

Soit n le nombre de clients et $N = \{1, 2, ..., n\}$ l'ensemble des clients. Le VRPTWFC est représenté par un graphe $G = (\bar{N}, A)$ où $\bar{N} = \{0\} \cup N \cup \{n + 1\}$ correspond à l'ensemble des nœuds, 0 et $n + 1$ représentant les dépôts initial et final. L'ensemble A des arcs entre les clients et également les arcs qui vont du dépôt initial aux clients et des clients au dépôt final.

À chaque nœud $i \in \bar{N}$ est associé un nombre de marchandises à collecter $q_i \in \mathbb{N}$ ($q_0 = q_{n+1} = 0$) et une fenêtre de temps $[a_i, b_i]$, où a_i correspond à la borne inférieure à partir de laquelle le service peut commencer et b_i à la borne supérieure à partir de laquelle le service peut être commencé. On a toujours $a_i \leq b_i$. De plus, l'ensemble des clients N est séparé en deux sous-ensembles distincts H et L ($N = H \cup L$ et $H \cap L = \varnothing$). H correspond à l'ensemble des nœuds dont les marchandises sont «lourdes» et L à l'ensemble des nœuds dont les marchandises sont «légères». A priori, un nœud ne peut être à la fois des marchandises lourdes et légères, mais il est toujours possible de dédoubler un nœud pour représenter une telle situation.

A chaque arc $(i, j) \in A$ est associé un coût non-nul c_{ij} et un temps de trajet t_{ij}. L'inégalité triangulaire est supposée respectée tant pour les coûts que pour les temps de trajet.

Le nombre de véhicules à disposition est illimité avec une capacité de Q répartie selon m piles de hauteur k. On définit $M = \{1, ..., m\}$ l'ensemble des piles et $K = \{1, ..., k\}$ l'ensemble des étages. Les marchandises collectées chez un client peuvent être réparties dans les différentes piles mais il ne peut pas y avoir d'espace libre entre les marchandises d'une même pile ou avec le bas de la pile.

3.2 Une formulation mathématique

3.2.1 Problème maître

Soit Ω l'ensemble des routes respectant les fenêtres de temps et les contraintes de capacité et de fragilité. Pour toute route $r \in \Omega$, le coût total de la route, correspondant à la somme du coût des arcs parcourus, est noté c_r. Soit a_{ir} un paramètre représentant le nombre de fois où le client i est visité dans la route r (0 ou 1 si la route est élémentaire). La variable y_r est égale à 1 si la route r fait partie de la solution choisie et 0 sinon. Le problème du VRPTWFC peut s'écrire :

$$\text{minimiser} \qquad \sum_{r\in\Omega} c_r y_r, \tag{3.1}$$

$$\text{sujet à :} \qquad \sum_{r\in\Omega} a_{ir} y_r = 1, \qquad \forall i \in N, \tag{3.2}$$

$$y_r \in \{0,1\}, \qquad \forall r \in \Omega. \tag{3.3}$$

La fonction objectif (3.1) minimise le coût total tandis que les contraintes (3.2) assurent que chaque client est visité exactement une fois. La difficulté de cette formulation est le très grand nombre de variables dans Ω qui rend la résolution de la relaxation linéaire complexe. Pour contourner cette difficulté, nous allons utiliser une méthode de génération de colonnes. Au lieu de résoudre directement la relaxation linéaire avec toutes les variables, une première relaxation est résolue avec un nombre réduit de variables. A partir de cette première solution, en utilisant les variables duales obtenues par les contraintes (3.2), de nouvelles variables avec un coût réduit négatif sont générées. Ces nouvelles variables sont ajoutées aux précédentes et la relaxation linéaire est résolue à nouveau. Ce procédé est itéré jusqu'à ne plus trouver de nouvelle variable avec un coût réduit négatif. La dernière solution obtenue pour la relaxation linéaire réduite est la solution optimale pour la relaxation linéaire avec toutes les variables. Si ce n'était pas la solution optimale, il existerait une variable avec un coût réduit négatif.

Dans la génération de colonnes, le problème (3.1)-(3.3) est appelé problème maître et sa version réduite est appelé problème maître restreint. Trouver des variables à coût réduit négatif est appelé sous-problème et correspond dans notre cas à la recherche d'un plus court chemin élémentaire avec fenêtres de temps et contraintes de capacité et de fragilité.

3.2.2 Sous-problème

Le coût réduit d'un arc c_{ij} est obtenu à partir des variables duales $\alpha_i \in N$ associées aux contraintes (3.2). Pour un arc $(i,j) \in A$, on a :

$$\bar{c}_{ij} = \begin{cases} c_{ij} - \alpha_i, & \forall i \in N, \\ c_{ij}, & \text{sinon} \end{cases} \tag{3.4}$$

Pour chaque client $i \in N$, T_i est la variable indiquant le moment de début de service et pour chaque emplacement de la cale défini par sa position $(m,k) \in M \times K$, $z_{i,mk}$ la variable qui vaut 1 si une marchandise du client i est mise dans l'emplacement (m,k), 0 sinon. Pour chaque arc $(i,j) \in A$, x_{ij} est la variable binaire égale à 1 si l'arc est utilisé dans la route, 0 sinon. Le sous-problème peut être modélisé de la manière suivante :

$$\text{minimiser} \quad \sum_{(i,j)\in A} \bar{c}_{ij} x_{ij}, \tag{3.5}$$

$$\text{sujet à} \quad \sum_{j\in N} x_{oj} = 1, \tag{3.6}$$

$$\sum_{j\in \bar{N}} x_{ji} - \sum_{j\in \bar{N}} x_{ij} = 0, \qquad \forall i \in N, \tag{3.7}$$

$$\sum_{j\in N} x_{j,n+1} = 1, \tag{3.8}$$

$$T_0 = 0, \tag{3.9}$$

$$T_j \geq (T_i + t_{ij}) x_{ij}, \qquad \forall i,j \in \bar{N}, \tag{3.10}$$

$$a_i \leq T_i \leq b_i, \qquad \forall i \in N, \tag{3.11}$$

$$\sum_{i\in N} z_{i,\alpha\beta} \leq 1, \qquad \forall (\alpha,\beta) \in M \times K, \tag{3.12}$$

$$\sum_{(\alpha,\beta)\in M\times K} z_{i,\alpha\beta} = q_i \sum_{j\in \bar{N}} x_{ij}, \qquad \forall i \in N, \tag{3.13}$$

$$\sum_{i\in N} z_{i,\alpha\beta} \geq \sum_{j\in N} z_{j,\alpha(\beta+1)}, \qquad \forall (\alpha,\beta) \in M \times K\backslash\{k\}, \tag{3.14}$$

$$\sum_{i\in N} T_i z_{i,\alpha\beta} \leq T + \sum_{j\in N} (T_j - T) z_{j,\alpha(\beta+1)}, \qquad \forall (\alpha,\beta) \in M \times K\backslash\{k\}, \tag{3.15}$$

$$\sum_{i\in L} z_{i,\alpha\beta} + \sum_{j\in H} z_{j,\alpha(\beta+1)} \leq 1, \qquad \forall (\alpha,\beta) \in M \times K\backslash\{k\}, \tag{3.16}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A, \tag{3.17}$$

$$z_{i,\alpha\beta} \in \{0,1\}, \qquad \forall i \in N, \forall (\alpha,\beta) \in M \times K. \tag{3.18}$$

Dans ce modèle, T est une constante plus grande que tous les temps du problème. Par exemple, $T \geq \underset{i \in N}{max}\, b_i$.

La fonction objectif (3.5) minimise la somme des coûts réduits. Les contraintes (3.6)–(3.8) assurent la cohérence de la route ; les contraintes (3.7) assurent la conservation du flot et les contraintes (3.6) et (3.8) assurent le départ et le retour du dépôt. Les contraintes (3.10) et (3.11) permettent de calculer les temps de début de visite des noeuds tout en s'assurant qu'ils respectent les fenêtres de temps. Les contraintes (3.12) - (3.16) vérifient le chargement des marchandises. Les contraintes (3.12) vérifient qu'il n'y a pas plus d'une marchandise par emplacement. Les contraintes (3.13) assurent que pour chaque client visité, toutes ses marchandises sont collectées. Ensuite, les inégalités (3.14) vérifient qu'une pile est bien remplie du bas vers le haut. De plus, les contraintes (3.15) imposent que, dans chaque pile, les marchandises soient disposées selon leur ordre de collecte. Finalement, les contraintes (3.16) assurent que la contrainte de fragilité est respectée, i.e., qu'une marchandise lourde ne se trouve pas au dessus d'une marchandise légère. Les contraintes (3.17) et (3.18) assurent l'intégrité des variables.

Les contraintes (3.10) et (3.15) ne sont pas linéaires mais peuvent se linéariser de manière classique par une approche grand-M. Comme nous ne nous servons pas directement de cette formulation, nous omettons de présenter cette transformation.

3.3 Algorithme général pour le VRPTW

Cette section présente l'algorithme BPC utilisé dans le cas du VRPTW. Le traitement de la contrainte de fragilité sera discutée dans les chapitres suivants et les modifications apportées à l'algorithme seront décrites à ce moment. Un algorithme BPC consiste en un arbre de branchement où les relaxations linéaires sont résolues par la méthode de génération de colonnes. L'algorithme va donc alterner entre le problème maître et le sous-problème. Pour renforcer les relaxations linéaires des plans coupants sont ajoutées stratégiquement. L'exploration de l'arbre de branchement peut suivre différentes stratégies selon le but recherché.

3.3.1 Sous-problème

Le sous-problème est un problème de recherche de plus court chemin élémentaire avec des contraintes de capacité et de fenêtres de temps. Ce problème est résolu avec un algorithme

de programmation dynamique utilisant des étiquettes. Une étiquette contient des informations sur un chemin partiel issu du nœud origine et s'arrêtant à un nœud η. L'algorithme propage des étiquettes jusqu'à atteindre le nœud destination en utilisant des fonctions de prolongation. Pour ne pas avoir à considérer toutes les étiquettes, un critère de dominance est appliqué régulièrement à chaque nœud.

Le sous-problème peut être relâché en autorisant les cycles dans les routes. Ces cycles sont ensuite éliminés dans l'arbre de branchement en ajoutant des décisions de branchement.

Algorithme à étiquettes

Les éléments d'une étiquette sont appelés les composantes de l'étiquette ; dans notre cas, une étiquette E possède les composantes suivantes :

- $\eta(E)$, le nœud final du chemin partiel représenté par l'étiquette E ;
- $t(E)$, l'heure de début de service au nœud $\eta(E)$;
- $c(E)$, la somme des couts réduits des arcs parcourus ;
- $U(E)$, l'ensemble des clients non-atteignables ;
- $l(E)$, la charge totale collectée.

Un client est dit non-atteignable s'il a déjà été visité, si en y allant directement après $\eta(E)$, on arriverait après la borne supérieure de la fenêtre de temps ou si la collecte de ces marchandises dépasserait la capacité du véhicule. Si $R(E) = (0, i_1, i_2, ..., \eta(E))$ est le chemin partiel représenté par l'étiquette E alors :

$U(E) = \{i \in N \cap R(E)\} \cup \{i \in N | t(E) + t_{\eta(E),i} > b_i\} \cup \{i \in N | l(E) + q_i > Q\}$

L'étiquette initiale E_0 est de la forme suivante :

$$\eta(E_0) = 0, \tag{3.19}$$
$$t(E_0) = 0, \tag{3.20}$$
$$c(E_0) = 0, \tag{3.21}$$
$$U(E_0) = \varnothing, \tag{3.22}$$
$$l(E_0) = 0. \tag{3.23}$$

Étant donné une étiquette E, une extension le long d'un arc $(\eta(E), j) \in A$ est légitime si et seulement si $j \in (N \backslash U(E)) \cup \{n+1\}$.

Pour une étiquette E et un arc $(\eta(E), j)$ légitime, la nouvelle étiquette E' est obtenue par les fonctions de prolongation suivantes :

$$\eta(E') = j, \tag{3.24}$$
$$t(E') = \max\{t(E) + t_{\eta(E),j}, a_j\}, \tag{3.25}$$
$$c(E') = c(E) + \bar{c}_{\eta(E),j}, \tag{3.26}$$
$$U(E') = U(E) \cup \{j\} \cup \{i \in N | t(E') + t_{ji} > b_i\} \cup \{i \in N | l(E') + q_i > Q\}, \tag{3.27}$$
$$l(E') = l(E) + q_j. \tag{3.28}$$

Dominance

Afin de réduire le nombre d'étiquettes à propager, une règle de dominance est appliquée pour réduire le nombre d'étiquettes. Une étiquette E_1 domine une étiquette E_2 si et seulement si :

$$\eta(E_1) = \eta(E_2), \tag{3.29}$$
$$t(E_1) \leq t(E_2), \tag{3.30}$$
$$c(E_1) \leq c(E_2), \tag{3.31}$$
$$U(E_1) \subseteq U(E_2), \tag{3.32}$$
$$l(E_1) \leq l(E_2). \tag{3.33}$$

Ces conditions assurent un critère de dominance valide si l'inégalité triangulaire est vérifiée pour les temps de trajet.

Relaxation

Il est possible d'enlever la contrainte de chemin élémentaire et donc d'autoriser les cycles dans les routes. Dans l'algorithme à étiquettes, il suffit de modifier la composante $U(E)$ pour ne pas tenir compte des clients déjà visités. Pour une étiquette E, $U(E)$ devient :

$U(E) = \{i \in N | t(E) + t_{\eta(E),i} > b_i\} \cup \{i \in N | l(E) + q_i > Q\}$.

Les autres composantes de l'étiquette ne sont pas modifiées, les règles de prolongation et de dominance restent les mêmes.

Cette redéfinition des ensembles $U(E)$ correspond à une relaxation totale de la contrainte

d'intégrité du problème. Dans ce cas, les ensembles U(E) sont inutiles et la dominance peut se faire sans comparer les ensembles U(E) car la condition est toujours vérifiée. Dans les *ng-routes* de *Baldacci et al. (2012)* les cycles sont autorisés uniquement s'ils contiennent des clients «éloignés» entre-eux. A chaque client $i \in N$ est associé un voisinage N_i contenant i et ses voisins «proches». Une *ng-route* peut contenir un cycle $i - ... - j - ... - i$ uniquement si un des clients j vérifie $i \notin N_j$. Ainsi, les routes contenant des cycles sont obligés de faire des longs détours et ont donc peu de chance d'être dans la solution optimale du problème maître.

Étiquetage bidirectionnel

Pour accélérer la propagation des étiquettes, il peut être intéressant de propager les étiquettes depuis le nœud $n+1$ également. Il y a alors deux types d'étiquettes, les étiquettes avant venant du nœud 0 et les étiquettes arrière venant du nœud n+1. L'étiquette arrière initial E_{n+1} a les composantes suivantes :

$$\eta(E_{n+1}) = n + 1, \tag{3.34}$$
$$t(E_{n+1}) = t_{max}, \tag{3.35}$$
$$c(E_{n+1}) = 0, \tag{3.36}$$
$$U(E_{n+1}) = \varnothing, \tag{3.37}$$
$$l(E_{n+1}) = 0. \tag{3.38}$$

t_{max} correspond au temps d'arrivée le plus grand possible.

Les fonctions de prolongation selon un arc (i, j), parcouru en mode arrière de j vers i, transformant une étiquette E_j en une étiquette E_i sont :

$$\eta(E_i) = i, \tag{3.39}$$
$$t(E_i) = min\{b_i, t(E_j) - t_{ij}\}, \tag{3.40}$$
$$c(E_i) = c(E_j) + \bar{c}_{ij}, \tag{3.41}$$
$$U(E_i) = U(E_j) \cup \{i\} \cup \{l \in N | t(E_i) - t_{l,i} < a_l\} \cup \{l \in N | l(E_i) + q_i + q_l > Q\}, \tag{3.42}$$
$$l(E_i) = l(E_j) + q_j. \tag{3.43}$$

Les fonctions de prolongation arrière sont très proches des fonctions de prolongation avant, la principale différence concerne la quantité de marchandises collectée puisque dans le mode

arrière la charge du nœud courant n'est pas comptée pour permettre la fusion.

Il est également possible de faire de la dominance entre deux étiquettes arrière E_1 et E_2, les règles étant très proches du mode avant. L'étiquette E_1 domine l'étiquette E_2 si et seulement si :

$$\eta(E_1) = \eta(E_2), \tag{3.44}$$
$$t(E_1) \geq t(E_2), \tag{3.45}$$
$$c(E_1) \leq c(E_2), \tag{3.46}$$
$$U(E_1) \subseteq U(E_2), \tag{3.47}$$
$$l(E_1) \leq l(E_2). \tag{3.48}$$
$$\tag{3.49}$$

Enfin, une étiquette avant E_f et une étiquette arrière E_b peuvent être fusionnées si les conditions suivantes sont vérifiées :

$$\eta(E_f) = \eta(E_b), \tag{3.50}$$
$$t(E_f) \leq t(E_b), \tag{3.51}$$
$$l(E_f) + l(E_b) \leq Q. \tag{3.52}$$

En mode bidirectionnel, les étiquettes ne sont pas prolongées jusqu'aux nœuds 0 ou n+1 mais jusqu'à ce que la ressource $t(E)$ atteigne $\frac{T}{2}$ sans le dépasser.

3.3.2 Recherche TABU

La méthode de recherche TABU, proposée par *Desaulniers et al. (2008)*, permet de générer efficacement de nouvelles routes. La recherche TABU consiste à chercher une nouvelle route à ajouter au problème maître à partir d'une route déjà utilisée. En effectuant des retraits ou des insertions de clients dans la route, une nouvelle route est créée et si son coût réduit est négatif, elle est ajoutée au problème maître. Pour ne pas revenir à la route initiale, les mouvements inverses sont enregistrés et interdits. La recherche TABU est effectuée à intervalle régulier pour ajouter de nouvelles routes au problème maître.

3.3.3 Plans coupants

La première famille de coupes est celle des coupes 2-chemins pour les problèmes de tournées de véhicules présentée dans *Kohl et al.(1999)*. Soit $N_S \subseteq N$ un sous-ensemble de nœuds ne pouvant être desservi par un seul véhicule à cause des fenêtres de temps et soit $\delta(N_S) = \{(i,j) \in A | i \in N_S, j \in \bar{N} \backslash N_S\}$ l'ensemble des arcs sortant de N_S. L'inégalité

$$\sum_{r \in \Omega} \sum_{(i,j) \in \delta(N_S)} b_{ij}^r y_r \geq 2, \tag{3.53}$$

où b_{ij}^r est le nombre de fois où l'arc (i,j) est parcouru dans la route r, est une coupe valide. Trouver des ensembles N_S n'est pas forcément facile car il faut résoudre un problème de voyageur de commerce qui est *NP-difficile*. Une heuristique gloutonne permet de trouver certains de ces ensembles. Ces coupes entrainent de nouvelles variables duales qui sont ensuite ajoutées dans le sous-problème.

La deuxième famille de coupes correspond à un sous-ensemble d'inégalités de rang 1 de Chvátal-Gomory. Elles furent introduites par *Jepsen et al. (2008)* et sont définies de la manière suivante :

$$\sum_{r \in \Omega} \left\lfloor \frac{1}{\chi} \sum_{i \in N_S} a_{ir} \right\rfloor y_r = \left\lceil \frac{|N_S|}{\chi} \right\rceil, \forall N_S \subset N, 2 \leq \chi \leq |N_S|. \tag{3.54}$$

Comme *Jepsen et al. (2008)*, nous nous focalisons sur les sous-ensembles de trois clients car ils sont plus faciles à identifier. Les inégalités ci-dessus deviennent

$$\sum_{r \in \Omega_S} y_r \leq 1, \forall N_S \subset N, |N_S| = 3, \tag{3.55}$$

où Ω_S correspond aux sous-ensembles de routes visitant au moins deux clients de N_S. Les variables duales issues de ces coupes ne peuvent pas être directement ajoutées aux coûts des arcs mais doivent être retirées aux routes visitant au moins deux clients des sous-ensembles considérées respectivement. L'algorithme avec étiquettes doit alors être modifié en conséquence (voir *Jepsen et al. (2008)*).

3.3.4 Branchement et stratégies d'exploration

Dans un algorithme de type *Branch-Price-and-Cut*, le branchement doit permettre d'obtenir une solution entière. Nous considérons trois possibilités de branchement. La première qui n'est pas suffisante pour assurer une solution entière est le branchement sur le nombre de véhicules utilisés. Si le nombre de véhicules obtenus dans la solution est fractionnaire, on peut créer deux branches :

$$\sum_{r\in\Omega} y_r \leq \left\lfloor \sum_{r\in\Omega} \tilde{y}_r \right\rfloor \tag{3.56}$$

$$\sum_{r\in\Omega} y_r \geq \left\lceil \sum_{r\in\Omega} \tilde{y}_r \right\rceil \tag{3.57}$$

où $(\tilde{y}_1, ..., \tilde{y}_{|\Omega|})$ est la solution fractionnaire obtenue.

Il est également possible de brancher sur le flot d'un arc $(i, j) \in A$. Il y a deux branches possibles :

$$x_{ij} = 0, \tag{3.58}$$

$$x_{ij} = 1. \tag{3.59}$$

Sur la branche $x_{ij} = 0$, l'arc (i, j) est retiré du sous-problème, tandis que sur l'autre, tous les autres arcs partant de i ou arrivant à j sont retirés. Ce branchement est suffisant pour obtenir une solution entière.

Le dernier branchement se fait sur le flot sortant d'un sous-ensemble de nœuds N_S. Le sous-ensemble N_S est choisi de tel sorte que le flot sortant $f(N_S) = \sum_{r\in\Omega} \sum_{(i,j)\in\delta(N_S)} b^r_{ij}\tilde{y}_r$ est le plus éloigné possible d'un entier. Deux branches sont crées :

$$\sum_{(i,j)\in\delta(N_S)} b^r_{ij} y_r \leq \lfloor f(N_S) \rfloor \tag{3.60}$$

$$\sum_{(i,j)\in\delta(N_S)} b^r_{ij} y_r \geq \lceil f(N_S) \rceil \tag{3.61}$$

Chacune de ces contraintes étant dans le problème maître, elles vont ajouter des variables duales qui vont ensuite être intégrées dans le sous-problème.

La stratégie d'exploration choisie est la stratégie de *meilleur d'abord*.

CHAPITRE 4 CAS DES PILES DE TAILLE 2

Dans ce chapitre, nous allons nous intéresser au cas particulier où les piles sont de taille 2, c'est-à-dire que dans une pile, il ne peut y avoir que deux marchandises l'une au dessus de l'autre. Dans cette recherche, ce cas a été le premier étudié et sa résolution a permis de résoudre le cas général par la suite. En premier, nous présentons le résultat théorique puis son intégration dans l'algorithme présenté au chapitre précédent et enfin les résultats expérimentaux obtenus avec cet algorithme.

4.1 Un résultat théorique

Ici, l'espace de chargement d'un véhicule est constitué de m piles de hauteur 2, la capacité de stockage vaut donc $Q = 2m$. La contrainte de fragilité interdit d'avoir une marchandise lourde au-dessus d'une marchandise légère. Il y a donc trois configurations possibles telles qu'illustrées à la figure 4.1.

Considérons une route respectant les fenêtres de temps et de capacité. À cette route est associée une séquence de marchandises. La question est de savoir si, connaissant la séquence, une répartition des marchandises dans le véhicule satisfait la contrainte de fragilité. Cette répartition doit respecter la séquence, i.e., une fois qu'une marchandise est positionnée, elle ne peut plus être déplacée.

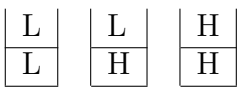

Figure 4.1 Superpositions possibles pour $k = 2$

La réponse à cette question est donnée dans la proposition 4.1.1.

Proposition 4.1.1 *Pour une séquence de marchandises lourdes et légères, il existe toujours une répartition respectant cette séquence et satisfaisant la contrainte de fragilité à l'exception des séquences atteignant la capacité de stockage Q du véhicule en chargeant un nombre impair de marchandises légères pour commencer puis uniquement des marchandises lourdes ensuite.*

Preuve par récurrence sur le nombre m de piles :

- Nous voulons prouver la proposition $\mathbb{P}(m)$ suivante pour tout $m \in \mathbb{N}^*$: «Dans un véhicule à m piles, pour une séquence de marchandises de volume inférieur ou égal à $2m$, cette séquence est admissible sauf si elle est de taille $2m$ et qu'elle débute par un nombre impair de marchandises légères puis uniquement des marchandises lourdes».
- $\mathbb{P}(1)$ est vraie.
- Supposons que $\mathbb{P}(z)$ soit vraie pour un certain $z \in \mathbb{N}^*$. Pour toute séquence de marchandises de volume inférieur ou égal à $2(z+1)$ et n'étant pas dans la situation interdite de la proposition, on distingue les trois cas suivants :
  - S'il y a au moins deux marchandises légères dans la séquence, les deux premières sont retirées. L'hypothèse de récurrence appliquée à la nouvelle séquence donne une répartition satisfaisante sur z piles. A cette répartition, les deux premières marchandises légères sont mises dans la $z+1^{ème}$ pile pour obtenir une répartition satisfaisant la contrainte de fragilité. L'hypothèse de récurrence peut être appliquée car la parité du nombre de marchandises légères n'a pas été changée.
  - S'il y a une seule marchandise légère dans la séquence, celle-ci n'est pas en première position car sinon il s'agit de la situation interdite. En enlevant la première marchandise lourde et la marchandise légère de la séquence, l'hypothèse de récurrence peut être appliquée. A la répartition obtenue, les deux marchandises enlevées sont ajoutées en les mettant dans la $z+1^{ème}$ pile.
  - S'il n'y a que des marchandises lourdes, le résultat est évident.

  Dans tous les cas, $\mathbb{P}(z+1)$ est vraie.
- Par récurrence, la proposition est vraie quel que soit le nombre de piles. □

Si la séquence contient un nombre impair de marchandises légères pour commencer puis uniquement des marchandises lourdes, au moins une marchandise légère se retrouve en bas d'une pile sans marchandise au-dessus d'elle. Si, en plus, la capacité du véhicule est atteinte, il y aura forcément une marchandise au-dessus de cette marchandise légère. Cette marchandise sera forcément lourde, il n'est donc pas possible de trouver une répartition satisfaisant la contrainte de fragilité pour de telles séquences.

L'idée pour obtenir ce résultat est de regarder la disposition de deux marchandises légères ou de deux marchandises lourdes quand deux piles sont vides.

Pour deux marchandises légères, les deux configurations illustrées à la figure 4.2 sont possibles. Dans la première disposition, après avoir placé les deux marchandises légères verticalement, la pile restante peut recevoir deux marchandises légères, deux marchandises lourdes ou une marchandise lourde suivie d'une marchandise légère. Dans la seconde disposition, après avoir placé les deux marchandises légères horizontalement, seules deux marchandises légères peuvent être placées au-dessus. Il est donc préférable de toujours superposer les marchandises légères pour mieux gérer les marchandises suivantes.

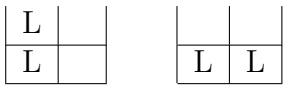

Figure 4.2 Dispositions pour deux marchandises légères

Pour deux marchandises lourdes, les configurations possibles sont données à la figure 4.3. Dans la première disposition, après avoir placé les deux marchandises lourdes verticalement, la pile restante peut recevoir deux marchandises légères, deux marchandises lourdes ou une marchandises lourde suivie d'une marchandise légère. Dans la seconde disposition, après avoir placé les deux marchandises lourdes horizontalement, toutes les combinaisons de deux marchandises peuvent être placées. Il est donc préférable de toujours aligner horizontalement les marchandises lourdes pour mieux gérer les marchandises suivantes.

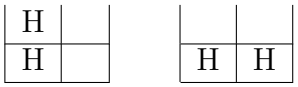

Figure 4.3 Dispositions pour deux marchandises lourdes

Dans le cas d'une marchandise légère et d'une marchandise lourde, il n'y a pas, a priori, de meilleure disposition des marchandises. Cependant, s'il n'y a plus de marchandise légère à venir alors il vaut mieux superposer la marchandise légère sur la marchandise lourde pour éviter que cette marchandise légère soit écrasée.

Afin de trouver une répartition satisfaisante, les marchandises légères sont superposées et les marchandises lourdes sont alignées au fond du véhicule. La dernière marchandise légère doit

être placée si possible dans le haut du véhicule. Si ce n'est pas possible alors il y a un risque qu'il n'existe pas de répartition satisfaisant la contrainte de fragilité si la séquence atteint la capacité du véhicule.

4.2 Implémentation dans l'algorithme de *Branch-Price-and-Cut*

Le résultat précédent nous donne un critère rapide pour savoir si une route respecte ou non la contrainte de fragilité. La contrainte de fragilité va donc être intégrée dans le sous-problème.

4.2.1 Nouvelle composante des étiquettes

Pour savoir si une route respecte ou non la contrainte de fragilité, il faut connaitre tous les clients. Rappelons qu'une étiquette représente une route partielle qui ne connait pas tous les clients. Cependant si dans cette route partielle, un client lourd a été visité avant un client léger, l'étiquette en question ne pourra donner que des routes satisfaisant la contrainte de fragilité. Pour une étiquette E, une composante $ITAP(E)$ est ajoutée pour savoir si les routes issues de cette étiquette risquent de ne pas respecter la contrainte de fragilité.

4.2.2 Fonctions de prolongation

Pour l'étiquette initiale E_0, $ITAP(E_0)$ est initialisée à 1.

En prolongeant une étiquette E_i le long d'un arc $(i,j) \in A$, la composante $ITAP(E_j)$ de la nouvelle étiquette E_j est obtenue de la manière suivante :

$$ITAP(E_j) = \begin{cases} ITAP(E_i) & si \begin{cases} i = 0,\ j \in L, \\ i \in L,\ j \in L, \\ i \in H,\ j \in H, \\ i \in L,\ j \in H,\ l(E_i)\, mod(2) = 1, \\ i \in H,\ j = n+1,\ l(E_i) = Q, \end{cases} \\ 0 & si \begin{cases} i = 0,\ j \in H, \\ i \in H,\ j \in L, \\ i \in L,\ j = n+1, \\ i \in L,\ j \in H,\ l(E_i)\, mod(2) = 0, \\ i \in H,\ j = n+1,\ l(E_i) < Q. \end{cases} \end{cases} \tag{4.1}$$

Pour faciliter la lecture de l'équation ci-dessus, le tableau 4.1 présente les différents cas possibles pour $ITAP(E_j)$.

Tableau 4.1 Table de prolongation pour ITAP selon l'arc (i, j)

	$j \in L$	$j \in H$	$j = n + 1$
$i = 0$	$ITAP(E_i)$	0	impossible
$i \in L$	$ITAP(E_i)$	$ITAP(E_i)$ si $l(E_i)\, mod(2) = 1$ 0 sinon	0
$i \in H$	0	$ITAP(E_i)$	$ITAP(E_i)$ si $l(E_i) = Q$ 0 sinon

En pratique, voici comment ITAP évolue sur une route :

- ITAP démarre à 1,
- Si la route contient un client lourd suivi d'un client léger alors ITAP vaudra 0 à la fin.
- Si la route ne contient que des clients légers, ITAP sera mise à 0 à la dernière prolongation.
- Si la route ne contient que des clients lourds, ITAP est mise à 0 à la première prolongation.
- Si la route contient uniquement des clients légers en premier puis uniquement des clients lourds ensuite, ITAP vaudra 1 uniquement si Q marchandises sont collectées et un nombre impair de marchandises légères sont collectées en premier.

Si $ITAP$ vaut 1 au nœud $n+1$, la route considérée ne respecte pas la contrainte de fragilité.

Quand une étiquette a été prolongée jusqu'au nœud $n + 1$, si $ITAP$ vaut 1 alors l'étiquette est rejetée et la route qu'elle formait n'est pas ajoutée au problème maitre.

Cette fonction de prolongation et le critère de rejet ci-dessus permettent de construire des routes respectant la contrainte de fragilité. Nous disposons donc d'un algorithme capable de construire des routes respectant les fenêtres de temps et les contraintes de capacité et de fragilité.

4.2.3 Dominance

Cette nouvelle composante $ITAP$ permet une dominance entre deux étiquettes E_1 et E_2. Le critère de dominance du chapitre 3 est repris en lui ajoutant une nouvelle ligne. E_1 domine E_2 si :

$$\eta(E_1) = \eta(E_2), \tag{4.2}$$
$$t(E_1) \leq t(E_2), \tag{4.3}$$
$$c(E_1) \leq c(E_2), \tag{4.4}$$
$$U(E_1) \subseteq U(E_2), \tag{4.5}$$
$$l(E_1) \leq l(E_2), \tag{4.6}$$
$$ITAP(E_1) \leq ITAP(E_2). \tag{4.7}$$

Ce critère de dominance reste valide pour les fenêtres de temps et la contrainte de capacité. Il est également valide pour la contrainte de fragilité selon les deux valeurs possibles pour $ITAP(E_1)$:

- Si $ITAP(E_1) = 0$ la dominance est évidente car $ITAP$ est décroissante.
- Si $ITAP(E_1) = 1$ alors $ITAP(E_2) = 1$. Prenons une extension valide pour E_2, si $l(E_1) < l(E_2)$, cette extension est forcément valide pour E_1. Si $l(E_1) = l(E_2)$, alors la composante $ITAP$ aura toujours la même valeur sur l'extension que l'on parte de E_1 ou de E_2 car seule la composante l a un impact sur la valeur de $ITAP$.

Dans les deux cas, toute extension valide pour E_2 est également valide pour E_1, le critère de dominance est donc valide.

4.2.4 Étiquetage Bidirectionnel

Il est possible de conserver le caractère bidirectionnel de l'algorithme du chapitre 3 en rajoutant la contrainte de fragilité avec la composante $ITAP$. La prolongation de la composante se fait de manière similaire en arrière mais il est important de se rappeler que la composante l en mode arrière ne prend pas en compte la quantité de marchandises collectée au nœud courant. La fonction de prolongation est donc la suivante, en parcourant l'arc $(i, j) \in A$ (de j vers i), la composante $ITAP(E_i)$ est obtenue :

$$ITAP(E_i) = \begin{cases} ITAP(E_j) & si \begin{cases} j = n+1,\ i \in H, \\ j \in H,\ i \in H, \\ j \in L,\ i \in L, \\ j \in H,\ i \in L,\ l(E_i)\, mod(2) = 1, \\ j \in L,\ i = 0,\ l(E_i) = Q, \end{cases} \\ 0 & si \begin{cases} j = n+1,\ i \in L, \\ j \in L,\ i \in H, \\ j \in H,\ i = 0, \\ j \in H,\ i \in L,\ l(E_i)\, mod(2) = 0, \\ j \in L,\ i = 0,\ l(E_i) < Q, \end{cases} \end{cases} \tag{4.8}$$

$ITAP(E_{n+1})$ est initialisée à 1. La prolongation en arrière est très proche de la prolongation en avant mais il faut faire attention au fait que la marchandise du noeud où on se situe n'est pas compté.

La fusion entre une étiquette avant E_f et une étiquette arrière E_b est possible si :

$$\eta(E_f) = \eta(E_b), \tag{4.9}$$
$$t(E_f) \leq t(E_b), \tag{4.10}$$
$$l(E_f) + l(E_b) \leq Q, \tag{4.11}$$
$$ITAP(E_f) + ITAP(E_b) < 2\ si\ \ l(E_f) + l(E_b) = Q. \tag{4.12}$$

On vérifie qu'on est au même nœud et que les fenêtre de temps et la contrainte de capacité sont respectées. Pour la contrainte de fragilité, si les deux étiquettes présentent un risque, on regarde si la capacité maximum du véhicule est atteinte. Si elle est atteinte, la fusion est impossible, dans tous les autres cas, la fusion est possible.

4.2.5 Recherche TABU

Le mouvement de retrait d'un client est toujours possible puisque la capacité du véhicule n'est plus atteinte. L'ajout d'un client ne pose aucun problème si la capacité du véhicule n'est pas atteinte et seules les routes composées de clients légers puis uniquement de clients

lourds peuvent poser problème (un des types de client peut manquer). Sur de telles routes, si l'insertion du client se fait parmi des clients de même type, légers ou lourds, ou à la séparation entre les clients légers et lourds et le nombre de marchandises légères de la nouvelle route est impair, la route n'est pas valable, elle l'est dans le cas contraire.

4.3 Expérimentations numériques

Dans cette section, nous comparons les résultats obtenus avec l'algorithme décrit précédemment avec ceux obtenus par un algorithme plus basique. Nous nous intéressons également à l'influence de certains paramètres dans les instances. Avant de présenter les résultats numériques, nous décrivons les instances utilisées pour les tests et l'algorithme basique.

4.3.1 Instances

Nous avons développé des instances comportant des clients lourds et des clients légers. Ces instances sont basées sur les instances de Solomon (1987) et plus précisément les instances R101 à R112. De ces instances, nous reprenons les positions des clients, les fenêtres de temps, les temps de parcours et les quantités de marchandises à collecter. Pour chaque client, une information est ajoutée pour savoir si les marchandises de ce client sont lourdes ou légères. Lors de la construction d'une instance, nous tirons au hasard pour chaque client la nature de ses marchandises avec une probabilité r que les marchandises de ce client soient légères et $1-r$ que les marchandises de ce client soient lourdes. Les probabilités considérées sont 0,25 ; 0,5 et 0,75.

Certaines instances de Solomon étant difficiles à résoudre avec 100 clients, les tests seront également menés avec des instances réduites aux 50 ou 75 premiers clients. De plus, la capacité de stockage est un facteur important pour avoir des routes problématiques. Nous considérons donc trois capacités possibles : 200 (la capacité originale de ces instances), 100 et 50. Ainsi pour une même instance de Solomon, il y a trois possibilités pour le nombre de clients, trois possibilités pour la probabilité de clients légers et trois possibilités pour la capacité ; il y a donc au total 27 combinaisons possibles. Une instance est caractérisée par un quadruplet de la forme $(R1m, n, r, Q)$; $R1m$ étant l'instance considérée avec m variant de 01 à 12, n le nombre de clients considérés, r la probabilité de clients légers et Q la capacité de stockage.

De plus, la répartition des clients lourds et légers étant aléatoire, pour chaque quadruplet,

nous générons 10 ou 12 instances indépendantes où la répartition des clients lourds et légers est différente mais dans les mêmes proportions.

Enfin, dans le cas $k = 2$, nous nous sommes aperçus que les quantités proposées ne permettent pas souvent d'atteindre la pleine capacité des véhicules et, par conséquent, les routes sont rarement problématiques. Afin de simuler un environnement dans lequel les routes sont plus souvent problématiques, nous avons réduit les quantités de marchandises collectées et la capacité de chaque véhicule. Pour un quadruplet, sa version réduite s'obtient en divisant la capacité des véhicules par 5 et en remplaçant les quantités de marchandises par 1 si la quantité originale est impaire, par 2 sinon. Pour chaque instance, il y a donc deux versions : classique et réduite.

4.3.2 Algorithme semi-naïf

Afin de tester la force du résultat théorique et l'algorithme qui en est sorti, nous avons voulu le comparer à une approche plus naïve pour résoudre le VRPTWFC qui consiste à créer une étiquette par disposition possible des marchandises collectées. Par exemple, si le premier client visité compte deux marchandises lourdes, il y deux manières possibles de répartir ces marchandises lourdes : soit en les superposant, soit en les alignant horizontalement. Lors de la prolongation, deux étiquettes seront créées pour représenter les deux dispositions possibles. Dans un tel algorithme, le nombre d'étiquettes devient rapidement très grand et il devient impossible de trouver des routes à coût réduit négatif en un temps raisonnable.

Toutefois, les remarques du résultat théorique peuvent être également utilisées ici afin de réduire le nombre d'étiquettes créées. Lorsqu'on collecte des marchandises lourdes, celles-ci vont d'abord être mises à l'étage inférieure puis ensuite sur des marchandises lourdes. Lorsqu'on collecte des marchandises légères, celles-ci vont d'abord être empilées les unes sur les autres ou mises au-dessus de marchandises légères. Le seul cas problématique va être si on dispose d'une marchandise légère et qu'on ne peut pas la placer au-dessus d'une autre marchandise légère. Dans ce cas, il y a deux possibilités : soit on la place dans une pile vide sans marchandise au-dessus d'elle, soit on la place dans une pile contenant une marchandise lourde (sous réserve que ces deux possibilités existent). Il est impossible de savoir a priori laquelle des deux dispositions est la plus intéressante. Dans cette situation uniquement, deux étiquettes seront créées, au lieu de créer des étiquettes pour chaque disposition possible, d'où le nom d'algorithme semi-naïf.

Fonctions de prolongation

Pour implémenter notre algorithme, nous devons rajouter trois composantes aux étiquettes présentées dans le chapitre 3 : p_L, p_H et p_F qui comptent respectivement le nombre de piles contenant uniquement une marchandise légère, uniquement une marchandise lourde et au moins une marchandise. Ces trois ressources sont initialisées à 0.

Depuis une étiquette E_i, en parcourant l'arc $(i, j) \in A$, les prolongations s'effectuent de la manière suivante :

— Si $j \in L$:

$$\begin{aligned} p_L(E_j) &= 0 \\ p_H(E_j) &= p_H(E_i) \\ p_F(E_j) &= p_F(E_i) + min\{m - p_F(E_i); \left\lfloor \frac{q_j - p_L(E_i)}{2} \right\rfloor\} \end{aligned}$$

Si $p_L(E_i) + 2\, min\{m - p_L(E_i); \left\lfloor \frac{q_j - p_L(E_i)}{2} \right\rfloor\} = q_j$ alors la prolongation est finie. Sinon, il reste $\alpha = q_j - (p_L(E_i) + 2\, min\{m - p_L(E_i); \left\lfloor \frac{q_j - p_L(E_i)}{2} \right\rfloor\})$ marchandises à placer. Deux étiquettes E_{j1} et E_{j2}, égales à E_j, sont créées et prolongées de la manière suivante :

$$\begin{aligned} p_L(E_{j1}) &= 1 \\ p_H(E_{j1}) &= p_H(E_j) - \alpha + 1 \\ p_F(E_{j1}) &= p_F(E_j) + 1 \end{aligned}$$

Si $p_F(E_{j1}) > m$ alors l'étiquette E_{j1} n'est pas considérée. Dans cette étiquette une marchandise légère est placée dans une pile vide. Pour l'étiquette E_{j2}, on calcule

$$\begin{aligned} p_L(E_{j2}) &= 0 \\ p_H(E_{j2}) &= p_H(E_j) - \alpha \\ p_F(E_{j2}) &= p_F(E_j) \end{aligned}$$

Si $p_H(E_{j2}) < 0$ alors l'étiquette E_{j2} n'est pas considérée. Dans cette étiquette, toutes les marchandises légères restantes sont placées au dessus de marchandises lourdes.

L'étiquette intermédiaire E_j est abandonnée.

— Si $j \in H$:

$$
\begin{aligned}
p_L(E_j) &= p_L(E_i) \\
p_H(E_j) &= p_H(E_i) + 2\,min\{m - p_F(E_i); q_j\} - q_j \\
p_F(E_j) &= p_F(E_i) + min\{m - p_F(E_i); q_j\}
\end{aligned}
$$

Si $p_H(E_j) < 0$ alors la prolongation n'est pas valide.

Le critère de dominance appliqué pour comparer deux étiquettes E_1 et E_2 est le suivant :

$$\eta(E_1) = \eta(E_2), \tag{4.13}$$
$$t(E_1) \leq t(E_2), \tag{4.14}$$
$$c(E_1) \leq c(E_2), \tag{4.15}$$
$$U(E_1) \subseteq U(E_2), \tag{4.16}$$
$$l(E_1) < l(E_2) \;||\; (l(E_1) = l(E_2)) \wedge (p_F(E_1) \leq p_F(E_2)) \tag{4.17}$$

Cependant, il n'est plus possible de conserver la prolongation bidirectionnelle car les fusions ne sont plus réalisables. Lors des fusions, il est parfois nécessaire de déplacer certaines marchandises (étaler des piles de marchandises légères par exemple), ce qui n'est pas possible car la position de chaque marchandise n'est pas connue.

4.3.3 Résultats globaux

Nos algorithmes ont été testés sur toutes les instances de Solomon avec les 50, 75 ou 100 premiers clients. Les capacités ont varié entre 50, 100 et 200 et le ratio de clients lourds et de clients légers ont été de 0,25 ; 0,5 et 0,75. Pour compenser la génération aléatoire des clients lourds et des clients légers, 12 instances avec les mêmes paramètres mais pas la même répartition de clients lourds et légers ont été testées.

Pour l'analyse, on donne la priorité aux nombre d'instances résolues en moins d'une heure de recherche dans l'arbre de branchement au détriment du temps moyen sur l'ensemble des instances testées. Les recherches dans l'arbre de branchement sont arrêtées après une heure

de calcul.

Chaque instance a été testée par trois algorithmes : notre algorithme avec la recherche bidirectionnelle, notre algorithme sans la recherche bidirectionnelle et l'algorithme semi-naïf. Les résultats pour chaque instance sont regroupés dans les tableaux 4.2 et 4.3.

Tableau 4.2 Nombre d'instances «classiques» résolues

	R101	R102	R103	R104	R105	R106	
Bidirectionnel	324	324	323	287	324	324	
Monodirectionnel	324	324	306	256	324	295	
Semi-naïf	324	324	290	214	324	288	
	R107	R108	R109	R110	R111	R112	Total
Bidirectionnel	288	279	266	268	314	242	3563
Monodirectionnel	254	203	241	210	294	213	3244
Semi-naïf	252	195	211	202	268	209	3101

Tableau 4.3 Nombre d'instances «réduites» résolues

	R101	R102	R103	R104	R105	R106	
Bidirectionnel	324	324	322	288	324	320	
Monodirectionnel	324	324	314	272	324	319	
Semi-naïf	324	324	296	252	324	305	
	R107	R108	R109	R110	R111	R112	Total
Bidirectionnel	288	288	301	286	290	219	3574
Monodirectionnel	216	150	281	197	291	171	3183
Semi-naïf	216	118	220	128	288	148	2943

Au total, notre algorithme en mode bidirectionnel a résolu 7137 instances tandis que l'algorithme semi-naïf n'en a résolu que 6044 sur un total de 7776 instances proposées. Notre algorithme est donc globalement plus efficace qu'un algorithme basique et, en particulier, les instances plus difficiles comme R104, R108 ou R112. Enfin si on ne regarde que les instances à 100 clients, les plus difficiles à résoudre, notre algorithme est bien plus performant que l'algorithme semi-naïf, en résolvant 1973 instances contre 1313 pour le second.

Notre algorithme en mode monodirectionnel permet de voir l'impact de la proposition 4.4.1 qui permet de ne pas multiplier le nombre d'étiquettes mais également la propagation bidirectionnelle. Notre algorithme en mode monodirectionnel résout plus d'instances que le semi-naïf (6427 contre 6044) mais l'écart est plus grand avec notre algorithme en mode bidirectionnel. De plus, si on regarde les instances difficiles à 100 clients, notre algorithme en

mode monodirectionnel a des performances proches de l'algorithme semi-naïf. Le caractère bidirectionnel de notre algorithme est donc important.

4.3.4 Statistiques sur le déroulement de l'algorithme et les solutions

Les principales statistiques sur le déroulement de l'algorithme et les solutions sont présentées dans les tableaux 4.4 et 4.5. Les caractéristiques sont données uniquement en mode bidirectionnel et les valeurs moyennes des statistiques sont issues uniquement des instances résolues.

Les statistiques sont les suivantes :

- n est le nombre de clients considérés ;
- $\#opt$ est le nombre d'instances résolues à l'optimalité sur un total de 1296 instances testées ;
- $t(s)$ est le temps CPU moyen en secondes de résolution des instances ;
- $gap(\%)$ le saut moyen d'intégrité exprimé en % ;
- $\#N$ est le nombre moyen de nœuds dans l'arbre de branchement ;
- $\#cc$ est le nombre moyen de coupes de capacité générées ;
- $\#2pc$ est le nombre moyen de coupes de 2-chemins générées ;
- $\#src$ est le nombre moyen de coupes de sous-ensemble générées ;
- $tSP(\%)$ est le pourcentage moyen du temps passé à résoudre le sous-problème par rapport au temps total ;
- $\#C$ est le nombre moyen de clients par véhicule ;
- $VP(\%)$ est le pourcentage de véhicules remplis à leur capacité maximale.

Dans ces tableaux, le temps de résolution augmente avec le nombre de clients considérés étant donné que les instances sont de plus en plus difficiles à résoudre. Les gaps sont peu élevés ce qui se reflète dans le nombre de nœuds de branchement faible. Peu de coupes de capacité sont générées, surtout des coupes de 2-chemins et en majorité des coupes de sous-ensemble. La proportion de temps consacrée à résoudre le sous-problème est toujours majoritaire, le sous-problème est donc bien la plus grande difficulté pour notre algorithme. Le pourcentage de véhicules pleins n'est pas très élevé car la contrainte de fragilité à tendance a interdire les véhicules pleins. Comme il y a peu de véhicules pleins, le nombre de clients par route est faible. On peut en déduire que les fenêtres de temps sont, en général, plus restrictives que la capacité des véhicules.

Tableau 4.4 Statistiques de l'algorithme VRPTWFC pour les instances classiques

n	#opt	t(s)	gap(%)	#N	#cc	#2pc	#src	tSP(%)	#C	VP(%)
50	1296	65	0.93	23.0	0.8	6.3	21.2	82	4.9	6.6
75	1277	237	0.65	122.2	0.9	10.4	93.3	79	5.1	14
100	990	769	0.51	238.7	1.0	11.9	183	76	5.4	19

Tableau 4.5 Statistiques de l'algorithme VRPTWFC pour les instances réduites

n	#opt	t(s)	gap(%)	#N	#cc	#2pc	#src	tSP(%)	#C	VP(%)
50	1296	20	0.76	3.6	0.4	2.5	6.0	82	6.0	16
75	1295	165	0.63	24.8	0.5	4.1	31.8	80	6.5	23
100	983	639	0.50	37.5	0.5	4.1	49.3	78	7.2	11

4.3.5 Influence du ratio lourds/légers

Il est intéressant de se demander si selon le ratio de clients lourds et de clients légers, notre algorithme en mode bidirectionnel est plus ou moins efficace. On regarde dans le tableau 4.6 le nombre d'instances «classiques» résolues selon le ratio, les instances R101, R102, R105 et R106 étant toutes résolues à l'optimalité, elles sont retirées.

Tableau 4.6 Influence du ratio lourds/légers

r	R103	R104	R107	R108	R109	R110	R111	R112	total
25 %	108	96	96	96	102	95	96	73	762
50 %	107	96	96	96	99	96	97	74	761
75 %	107	96	96	96	100	95	97	72	759

Le nombre d'instances résolues est presque indépendant du ratio choisi. Les légères différences peuvent s'expliquer par le fait que les répartitions étant aléatoires, la difficulté n'est pas équitablement répartie. En plus, si on prend chaque instance individuellement, le résultat reste vrai.

4.3.6 Influence de la capacité

Enfin, il est intéressant de se demander le rôle de la capacité dans notre problème. En effet, si la capacité est trop grande, aucune route ne posera problème et notre algorithme est aussi efficace que l'algorithme général pour le VRPTW. On regarde donc le nombre d'instances résolues par notre algorithme en mode bidirectionnel selon la capacité considérée au tableau 4.7.

Tableau 4.7 Influence de la capacité (instances classiques)

cap	R103	R104	R107	R108	R109	R110	R111	R112	total
50	107	108	108	100	103	88	107	98	819
100	108	71	72	71	55	72	99	72	620
200	108	108	108	108	108	108	108	72	828

Il apparait clairement que la capacité a une influence importante sur le nombre d'instances résolues. Si la capacité est trop faible ou trop grande le même nombre d'instances est résolu. La résolution ne dépend donc pas du problème de superposition lourds/légers mais de la difficulté intrinsèque des instances. Par contre, si la capacité est ajustée, le nombre d'instance résolues est plus faible car les problèmes de superposition lourds/légers apparaissent et ralentissent la génération de routes.

Des résultats similaires sont obtenus pour les instances réduites comme rapportées au tableau 4.8. Cette fois-ci, c'est pour la capacité égale à 50 qu'il y a le moins d'instances résolues.

Tableau 4.8 Influence de la capacité (instances réduites)

cap	R103	R104	R107	R108	R109	R110	R111	R112	total
50	106	72	72	72	85	71	74	75	623
100	108	108	108	108	108	107	108	72	827
200	108	108	108	108	108	108	108	72	828

Cette capacité critique s'explique du fait de la nécessité d'atteindre exactement la capacité du véhicule pour avoir des problèmes de superposition lourds/légers. Si la capacité est trop grande alors aucune ou peu de routes ne l'atteignent et peuvent poser des problèmes de superposition. Si la capacité est trop faible, le nombre de routes possibles est faible et la combinatoire des quantités à collecter fait que peu de routes posent problème. L'importance de la combinatoire s'observe également quand on compare le nombre d'instances résolues entre les instances classiques et réduites (voir tableau 4.9)

Le nombre d'instances résolues par notre algorithme en mode bidirectionnel est à peu près le même pour les instances classiques et réduites mais est plus faible pour les instances réduites pour l'algorithme semi-naïf. Dans le cas des instances réduites, la combinatoire permet d'avoir plus de routes atteignant la capacité du véhicule et donc avec potentiellement un problème de superposition. Il est donc plus difficile de générer des routes à coût réduit négatif et résoudre le problème à l'optimalité est donc plus difficile.

Tableau 4.9 Influence de la combinatoire

	Bididrectionnel	Monodirectionnel	naïf
instances classiques	3563	3244	3101
instances réduites	3574	3183	2943

Pour appuyer ces résultats, les tableaux 4.10 et 4.11 donnent les statistiques sur le déroulement de l'algorithme et les solutions en fonction de la capacité. Les statistiques sont les mêmes qu'en 4.3.4. Comme prévu, le nombre d'instances résolues est plus faible pour une capacité de 100 dans les instances classiques et de 10 pour les instances réduites. Logiquement les temps moyens de calculs sont les plus longs pour ces instances. Le pourcentage de véhicules pleins est le plus grand pour les capacités les plus faibles mais il ne s'effondre pas autant pour les instances de capacité de 100 que pour les autres instances. Ces résultats confirment ce qui a été dit plus haut, il existe une capacité critique pour laquelle la difficulté de résolution est maximale.

Tableau 4.10 Statistiques de l'algorithme VRPTWFC pour les instances classiques selon les capacités

cap	#opt	t(s)	gap(%)	#N	#cc	#2pc	#src	tSP(%)	#C	VP(%)
50	1251	341	0.68	280.6	1.3	19.8	194.8	76	3	26
100	1052	459	0.84	54.8	1.2	4.7	58.2	81	5.6	11
200	1260	189	0.65	10.7	0.3	2.7	18.3	81	6.9	0.8

Tableau 4.11 Statistiques de l'algorithme VRPTWFC pour les instances réduites selon les capacités

cap	#opt	t(s)	gap(%)	#N	#cc	#2pc	#src	tSP(%)	#C	VP(%)
10	1055	302	0.67	42.6	1.4	5.1	47.3	81	5.6	58
20	1259	215	0.64	11.3	0.14	2.8	18.8	80	6.9	0.2
40	1260	220	0.64	11.5	0.03	2.8	18.9	80	6.9	0.2

4.3.7 Comparaison avec le VRPTW

Une autre chose intéressante à regarder est le nombre d'instances résolues pour lesquelles la solution optimale obtenue est différente de celle obtenue avec un algorithme résolvant le VPRTW de manière exacte. Ces instances sont celles où les tournées obtenues sans tenir compte de la contrainte de fragilité ne la respectent pas.

Pour R101, 72 instances classiques et 19 instances réduites sur un total de 648 instances poseraient des problèmes de superposition si elles étaient résolues avec un algorithme de VRPTW, soit 14% de toutes les instances. Pour R102, 39 instances classiques et 23 instances réduites soit 10% des instances. De manière globale, il semble que, pour 10% des instances, la solution obtenue pour le VRPTWFC soit différente de la solution obtenue pour le VRPTW.

Toutefois, ce résultat n'est pas à prendre directement comme le fait que seulement 10% des instances sont plus dures à résoudre. Dans nos algorithmes proposés, de nombreuses routes sont générées et aucune de ces routes ne violera la contrainte de fragilité. Un algorithme de VRPTW peut arriver à une solution respectant la contrainte de fragilité mais il aura peut être généré des routes ne la respectant pas pendant son processus. De plus, il ne faut pas confondre la proposition «*La solution optimale est la même pour le VRPTW et le VRPTWFC*» avec «*Aucune des routes possibles dans cette instance ne pose un problème de superposition lourd/léger*». La première est plus un résultat lié au hasard de la génération des clients lourds et des clients légers tandis que la seconde est liée à une composition particulière de l'instance (uniquement des quantités de marchandises paires collectées, par exemple). Enfin, il est important de noter que nous cherchons à résoudre de manière optimale notre problème et si la solution obtenue par le VRPTW ne respecte pas la contrainte de fragilité, il n'est pas facile de trouver la solution optimale à partir de la solution obtenue. Dans le contexte d'une compagnie de transport maritime, si une tournée sur dix dégrade la marchandise, sa réputation risque de devenir négative très rapidement.

Deux autres données intéressantes sont l'évolution du temps de calcul et du coût des solutions selon que les instances soient résolues avec le VRPTW ou le VRPTWFC. Pour les instances avec 50 clients, toutes les instances ont été résolues avec le VRPTW et le VRPTWFC. Les résultats du tableau 4.12 montrent une augmentation en coût très faible et une augmentation en temps légère, inférieure à 50%, dans tous les cas. Cependant, parmi les instances considérées, la solution obtenue pour le VRPTWFC est parfois la même que celle obtenue pour le VRPTW. Il convient de regarder la différence de coût uniquement avec les instances dont

les solutions pour le VRPTW et le VRPTWFC sont différentes, présentées dans le tableau 4.13. L'augmentation en coût est en moyenne faible, toujours inférieure à 0.1%, le surcoût pour respecter la contrainte de fragilité est faible alors que le risque pris si les marchandises sont abimées peut être grand.

Tableau 4.12 Comparaison VRPTW et VRPTWFC

instances	classiques		réduites	
	t(s)	coûts	t(s)	coûts
VRPTW	45	1396.59	16	1272.66
VRPTWFC	64	1396.79	20	1272.82

Tableau 4.13 Augmentation absolue moyenne du coût liée à la contrainte de fragilité

instances	n=50	n=75	n=100
classiques	3.4	2.8	4.1
réduites	4.0	1.3	0.9

CHAPITRE 5 CAS GÉNÉRAL

Dans ce chapitre, nous étudions le cas où la hauteur des piles des véhicules est considérée comme une donnée du problème. Suite à la présentation d'un résultat théorique, nous proposons un algorithme de résolution et le testons sur quelques instances.

5.1 Un résultat théorique

Ici, un véhicule est constitué de m piles de hauteur k, la capacité du véhicule est donc $Q = m \times k$. La contrainte de fragilité interdit d'avoir une marchandise légère en-dessous d'une marchandise lourde. On a toujours $k \geq 2$.

La question est toujours de savoir s'il est possible de trouver une répartition satisfaisante dans le véhicule connaissant la séquence de marchandises.

On définit : $F(x) = \begin{cases} x \ mod[k], & si \ x \ mod[k] \ \neq \ 0, \\ k, & sinon. \end{cases}$

L'idée du résultat est la suivante : après avoir collecté des marchandises légères et possiblement des marchandises lourdes, les marchandises collectées sont réparties telles que toutes les piles contenant au moins une marchandise légère soient pleines. En effet, si toutes les piles contenant des marchandises légères sont pleines, alors il ne pourra jamais y avoir de problème de superposition. Si jamais il n'est pas possible de compléter toutes les piles contenant des marchandises légères, une pile incomplète va apparaitre (s'il y en a plusieurs, il suffit de les rassembler pour n'en avoir qu'une seule). Il y a alors un risque de collecter trop de marchandises lourdes et de devoir en mettre dans cette pile incomplète provoquant un problème de superposition. Dans ce cas, toutes les marchandises lourdes collectées précédemment sont utilisées pour compléter la pile incomplète avec des marchandises légères. Il est donc facile de savoir le volume minimum de marchandises lourdes à collecter pour qu'il y ait un problème de superposition.

Si jamais il y a une pile incomplète, mais qu'on collecte ensuite des marchandises légères, ces marchandises sont mises dans la pile incomplète jusqu'à la compléter. Après que la pile soit complétée, le risque d'avoir une nouvelle pile incomplète revient et il faut recommencer le même raisonnement.

Proposition 5.1.1 *Pour une séquence de marchandises lourdes et légères, il existe toujours une répartition de la séquence respectant la contrainte de fragilité à l'exception du cas suivant : du début de la séquence jusqu'à un client i, ont été collectées*

- *a_H marchandises lourdes*
- *a_L marchandises légères*

tel que $a_H + F(a_L) < k$ puis, après ce client i sont collectées strictement plus que $Q - (a_H + a_L) - k + (a_H + a_L) mod[k]$ marchandises lourdes.

La condition $a_H + F(a_L) < k$ indique qu'après le client i, il y aura une pile contenant des marchandises légères partiellement remplie et la quantité $Q-(a_H+a_L)-k+(a_H+a_L)mod[k]$ l'espace libre avant de devoir mettre une marchandise dans la colonne partiellement remplie.

Preuve par récurrence sur le nombre m de piles, avec $k \in \mathbb{N}$ fixé :

- Nous voulons prouver la proposition $\mathbb{P}(m)$ suivante pour tout $m \in \mathbb{N}^*$: «Dans un véhicule avec m piles de hauteur k, une séquence d'entrée de marchandises de volume inférieur à $m \times k$ est admissible sauf si elle ne respecte pas le critère énoncé ci-dessus».

- $\mathbb{P}(1)$ est vraie :
  - $F(a_L) = a_L$ car $a_L \neq 0$ et $k \geq 2$. Si $a_H + a_L < k$ il y a une marchandise légère en haut d'une pile qui n'est pas pleine.
  - Si $a_H + a_L < k$, $Q - (a_H + a_L) - k + (a_H + a_L) mod[k] = 0$. On ne peut collecter aucune marchandise lourde car il n'y a qu'une seule pile, la marchandise lourde viendrait donc directement écraser la marchandise légère.
- Supposons que $\mathbb{P}(m)$ soit vraie pour un certain $m \in \mathbb{N}^*$. Pour toute séquence d'entrée de marchandises de volume inférieur à $(m+1) \times k$ et n'étant pas la situation interdite de la proposition, on distingue les trois cas suivants :
  - S'il y a plus de k marchandises légères dans la séquence, on retire les k premières dans l'ordre d'entrée. On applique l'hypothèse de récurrence avec m piles sur la nouvelle séquence pour obtenir une répartition satisfaisante. On reprend la séquence initiale, à la répartition satisfaisante précédente, on ajoute que les k premières marchandises légères qu'on avait retirées en les mettant dans la $m + 1^{ème}$ pile. On peut appliquer l'hypothèse de récurrence car on n'a pas changé le modulo

k du nombre de marchandises légères.

— S'il y a strictement moins de k marchandises légères dans la séquence, disons $f > 0$, il y a au moins $k - f$ marchandises lourdes avant la première marchandise légère, sinon on se retrouverait dans la situation interdite. On retire les $k - f$ premières marchandises lourdes et les f marchandises légères de la séquence, on applique l'hypothèse de récurrence avec m piles. On obtient une répartition satisfaisante à laquelle on ajoute les k marchandises précédemment retirées en les mettant dans la $m + 1^{ème}$ pile.

— S'il n'y a que des marchandises lourdes, le résultat est évident.

Dans tous les cas, $\mathbb{P}(m + 1)$ est vraie.

— Par récurrence, le résultat est vrai quel que soit le nombre de piles m et la hauteur k de ces piles. □

Ce résultat concerne l'ensemble de la séquence, il assure donc une répartition respectant la contrainte de fragilité sans avoir à déplacer les marchandises durant la collecte. Toutefois, il n'est pas possible de connaitre la position des marchandises pendant la construction de la route car il faut connaitre l'ensemble des marchandises collectées pour connaitre la répartition finale.

Ce résultat permet de déterminer si une route respecte ou non la contrainte de fragilité. Son implémentation dans l'algorithme du chapitre 3, présentée dans la section suivante, va permettre de résoudre le VRPTWFC efficacement.

5.2 Implémentation dans l'algorithme BPC

Le résultat de la proposition 5.1.1 permet de déterminer si une route respecte la contrainte de fragilité. Il va donc être intégré au sous-problème.

5.2.1 Nouvelles composantes des étiquettes

Trois nouvelles composantes vont être ajoutées aux étiquettes pour résoudre le sous-problème. La première, nommée $ITAP$ comme dans le chapitre précédent, indique s'il y a une pile incomplète avec des marchandises légères et, si oui, combien d'espaces sont vides dans cette pile. Les deux autres composantes, notées a_H et a_L, comptent respectivement les nombres de marchandises lourdes et légères collectées.

5.2.2 Fonctions de prolongation

Pour l'étiquette initiale E_0, ces trois composantes sont initialisées à 0.

Le long d'un arc $(i, j) \in A$, les composantes sont prolongées de la manière suivante :

$$a_H(E_j) = \begin{cases} a_H(E_i) + q_j, & si\ j \in H; \\ a_H(E_i), & sinon, \end{cases} \tag{5.1}$$

$$a_L(E_j) = \begin{cases} a_L(E_i) + q_j, & si\ j \in L, \\ a_L(E_i), & sinon, \end{cases} \tag{5.2}$$

$$ITAP(E_j) = \begin{cases} ITAP(E_i), & si\ j \in H \cup \{n+1\}, \\ ITAP(E_i) - q_j, & si\ (ITAP(E_i) - q_j \geq 0) \wedge (j \in L), \\ \max\{0; k - (a_H(E_j) + F(a_L(E_j)))\}, & sinon. \end{cases} \tag{5.3}$$

Pour résumer, ITAP n'est modifié que lors d'une visite à un client léger. Si une pile est incomplète, les marchandises collectées servent à compléter cette pile partiellement (premier cas) ou totalement (second cas). Si la pile est complétée totalement, il faut placer les marchandises légères restantes à collecter, soit en remplacent des marchandises lourdes de la pile précédemment incomplète soit en formant une nouvelle pile avec les marchandises lourdes collectées. Ces deux possibilités sont résumées dans la formule $max\{0; k - (a_H(E_j) + F(a_L(E_j)))\}$ qui dispose toutes les marchandises légères à nouveau.

Pour qu'une prolongation soit légitime, il faut vérifier :

$$l(E_j) + ITAP(E_j) \leq Q. \tag{5.4}$$

5.2.3 Dominance

Pour qu'une étiquette E_1 domine une étiquette E_2, il faut en premier vérifier les critères du chapitre 3, (3.28) à (3.31) mais pas nécessairement (3.32), puis vérifier l'un des trois critères suivants.

Critère 1

$l(E_2) + ITAP(E_2) - (l(E_1) + ITAP(E_1)) \geq k$

Si ce critère est vérifié, l'inégalité suivante est vraie : $Q - (l(E_1) + ITAP(E_1)) - (Q - (l(E_2) + ITAP(E_2))) \geq k$. Toute extension légitime pour le chemin va placer des marchandises dans l'espace libre $(Q - (l(E_2) + ITAP(E_2)))$ en respectant la contrainte de fragilité. Ces marchandises peuvent être placées exactement dans la même disposition dans l'espace libre $Q - (l(E_1) + ITAP(E_1))$ et il reste au moins k espaces libres accessibles. Ces k espaces libres peuvent être regroupés sans violer la contrainte de fragilité et libérer ainsi une pile en entier dans la prolongation de l'étiquette E_1. Dans cette pile, les marchandises placées dans l'espace libre $ITAP(E_2)$ peuvent être placées sans problème de superposition. La prolongation est donc légitime pour l'étiquette E_1, ce critère de dominance est valide.

Critère 2

$l(E_2) - l(E_1) \geq k - 1$

Toute extension du chemin 2 pour le chemin 1 donnera un total de marchandises collectées inférieur ou égal à $Q - k + 1$. En faisant uniquement des piles de marchandises lourdes et des piles de marchandises légères, il n'y aura aucune pile avec les deux types de marchandises. Comme il ne peut donc pas y avoir de problème de superposition, ce critère de dominance est valide.

Critère 3

- $l(E_2) \geq l(E_1)$
- $l(E_2) + ITAP(E_2) \geq l(E_1) + ITAP(E_1)$
- $a_H(E_1) \geq k - 1$ **OU** $F(a_L(E_1) \leq F(a_L(E_2))$

Vérifier ces trois conditions nous assure que toute extension du chemin 2 est légitime pour le chemin 1. En effet, supposons une extension légitime pour le chemin 2. Si cette extension ne comporte que des marchandises légères, la première condition nous assure que l'extension est aussi légitime pour le chemin 1. Si cette extension ne comporte que des marchandises lourdes, la deuxième condition nous assure que cette extension est légitime pour le chemin 1. Sinon, la troisième condition nous assure que, si une nouvelle pile de marchandises légères est commencée pour le chemin 1, alors soit cette pile est forcément pleine ($a_{H1} \geq k - 1$) ou soit une nouvelle pile a été également commencée pour le chemin 2 et avec la seconde condition, il n'y aura pas de problème de superposition. Ce critère de dominance est donc valide.

Stratégie

Les deux premiers critères ne sont pas nécessaires pour avoir la dominance, le troisième l'est peut-être mais ce résultat n'a pas été prouvé.

Les deux premiers critères étant plus faciles à vérifier que la troisième, ils sont donc vérifiées en premier. Le troisième critère permet une dominance entre des chemins proches en terme de quantités de marchandises collectées.

5.2.4 Étiquetage bidirectionnel

Le mode arrière s'interprète comme une livraison au lieu d'une collecte et il suffit donc d'inverser les rôles des marchandises légères et lourdes pour la prolongation des étiquettes. Une ressource $JTAP$ similaire à $ITAP$ est créée. a_H, a_L et $JTAP$ sont initialisées à 0. Pour une prolongation arrière le long d'un arc $(i, j) \in A$ (de j vers i) :

$$a_H(E_i) = \begin{cases} a_H(E_j) + q_j, & si\ j \in H, \\ a_H(E_j), & sinon, \end{cases} \tag{5.5}$$

$$a_L(E_i) = \begin{cases} a_L(E_j) + q_j, & si\ j \in L, \\ a_L(E_j), & sinon, \end{cases} \tag{5.6}$$

$$JTAP(E_i) = \begin{cases} JTAP(E_j), & si\ j \in L \cup \{0\}, \\ JTAP(E_j) - q_j, & si\ (JTAP(E_j) - q_j \geq 0) \wedge (j \in H), \\ \max\{0; k - (a_L(E_j) + F(a_H(E_j)))\}, & sinon. \end{cases} \tag{5.7}$$

Là encore, il faut tenir compte qu'on collecte la marchandise du nœud de départ et non du nœud d'arrivée lors de la prolongation.

En étiquetage avant, jusqu'à maintenant, la répartition des marchandises lourdes qui n'était pas en dessous de marchandises légères n'était pas considérée. À partir de maintenant, ces marchandises sont empilées verticalement pour occuper le moins de piles possibles. La même règle est appliquée pour l'étiquetage arrière avec les marchandises légères respectivement. Cette règle ne change rien aux résultats précédents et ne donne pas une «meilleure» répartition, elle permet uniquement de trouver les critères de fusion.

Dans cette configuration, les étiquettes avant et arrière ont au maximum deux piles incomplètes. Une mixte pouvant contenir des marchandises lourdes et des marchandises légères et une non-mixte contenant uniquement des marchandises lourdes en avant et uniquement des marchandises légères en arrière.

Pour fusionner deux étiquettes avant E_f et arrière E_b, il faut vérifier les critères (3.50)–(3.52).

Ensuite on calcule $A = l(E_f) + ITAP(E_f) + k - F(l(E_f) + ITAP(E_f)) + l(E_b) + JTAP(E_b) + k - F(l(E_b) + JTAP(E_b))$. En divisant par k, $l(E_f) + ITAP(E_f) + k - F(l(E_f) + ITAP(E_f))$ correspond au nombre de piles utilisées dans l'étiquette avant et $l(E_b) + JTAP(E_b) + k - F(l(E_b) + JTAP(E_b))$ correspond au nombre de piles utilisées dans l'étiquette arrière. A divisé par k correspond donc au nombre de piles utilisées par les deux étiquettes et ne peut prendre que des valeurs entières multiples de k. La fusion va essayer de répartir ces différentes piles pour trouver une répartition satisfaisante. La fusion peut se faire sans réarrangement (voir figure 5.1) ou avec (voir figure 5.2). La difficulté est de savoir si les réarrangements sont légitimes ou si la fusion est impossible. Le tableau 5.1 récapitule les différents critères de fusion.

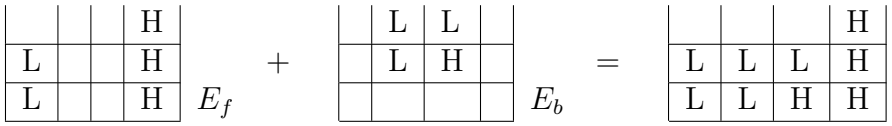

Figure 5.1 Fusion entre E_f et E_b sans réarrangement

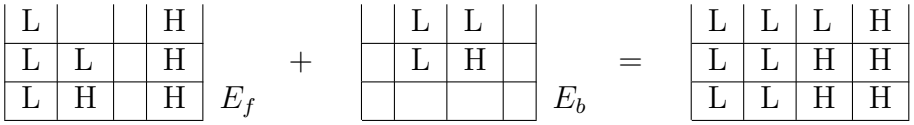

Figure 5.2 Fusion entre E_f et E_b avec réarrangements

Si $A \leq Q$ alors on peut réaliser la fusion car il y a suffisamment de piles dans le véhicule pour disposer les piles des étiquettes avant et arrière sans que celles-ci ne se superposent.

Si $A = Q + k$, il y a trois cas possibles :

— Si dans l'étiquette arrière, il y a une pile avec uniquement des marchandises légères et dans l'étiquette avant une pile avec uniquement des marchandises lourdes, la fusion est possible. En effet en retirant la pile de marchandises légères de l'étiquette arrière, les deux étiquettes peuvent être fusionnées car il y a suffisamment de piles dans le véhicule pour disposer les piles des étiquettes avant et arrière sans que celles-ci ne se superposent. Ensuite un maximum des marchandises lourdes de l'étiquette avant sont passées dans la pile mixte de l'étiquette arrière. A ce moment soit la pile incomplète non-mixte de l'étiquette avant est vide pour mettre la pile retirée, soit la pile mixte de l'étiquette arrière est pleine et les marchandises légères retirées peuvent être placées dans les espaces libérées légitimement car elles sont légères et au dessus de marchandises collectées précédemment (il y a de la place d'après la condition (3.50)). Les deux conditions à vérifier sont :
  — $a_L(E_b) > k - F(a_H(E_b)) - JTAP(E_b)$
  — $a_H(E_f) > k - F(a_L(E_f)) - ITAP(E_f)$

— S'il n'y a pas de pile avec uniquement des marchandises légères dans l'étiquette arrière ($a_L(E_b) = k - F(a_H(E_b)) - JTAP(E_b)$) alors il faut que les marchandises lourdes de la pile avec le moins de marchandises lourdes de l'étiquette arrière puissent aller dans la pile avec le moins de marchandises lourdes de l'étiquette avant, soit $k - F(a_H(E_b)) \geq F(l(E_f) + ITAP(E_f))$. Cette règle fonctionne aussi s'il n'y a pas de pile incomplète avec uniquement des marchandises lourdes dans l'étiquette avant ($F(l(E_f) + ITAP(E_f)) = k$).

— S'il n'y a pas de pile avec uniquement des marchandises lourdes dans l'étiquette

avant ($a_H(E_f) = k - F(a_L(E_f)) - ITAP(E_f)$) alors il faut que les marchandises légères de la pile avec le moins de marchandises légères de l'étiquette avant puissent aller dans la pile avec le moins de marchandises légères de l'étiquette arrière, soit $k - F(a_L(E_f)) \geq F(l(E_b) + JTAP(E_b))$. Cette règle fonctionne aussi s'il n'y a pas de pile incomplète avec uniquement des marchandises légères dans l'étiquette arrière ($F(l(E_b) + JTAP(E_b)) = k$).

Si $A = Q + 2k$, alors la stratégie est la suivante :

— Si l'étiquette arrière ne possède pas de pile avec uniquement des marchandises légères ($a_L(E_b) = k - JTAP(E_b) - F(a_H(E_b))$) alors la fusion est impossible. En effet, dans ce cas :

$$\begin{aligned} & l(E_b) + JTAP(E_b) = k + a_H(E_b) - F(a_H(E_b)) \\ \Leftrightarrow\ & F(l(E_b) + JTAP(E_b)) = k \end{aligned}$$

En substituant ce résultat dans l'expression de A, on obtient :

$$\begin{aligned} & Q - l(E_f) - ITAP(E_f) = a_H(E_b) - F(a_H(E_b)) - F(l(E_f) + ITAP(E_f)) \\ \Rightarrow\ & Q - l(E_f) - ITAP(E_f) < a_H(E_b) \end{aligned}$$

Or $Q - l(E_f) - ITAP(E_f)$ est le nombre maximum d'items lourds qu'on peut collecter à partir de la fin de l'étiquette avant, la fusion est donc impossible. Par symétrie, si l'étiquette avant ne contient pas de pile avec uniquement des marchandises lourdes ($a_H(E_f) = k - ITAP(E_f) - F(a_L(E_f))$), la fusion est impossible.

— Si l'étiquette avant contient au moins une pile avec uniquement des lourds et l'étiquette arrière une pile avec uniquement des marchandises légères, la pile contenant le moins de marchandises lourdes exclusivement de l'étiquette avant ($F(l(E_f) + ITAP(E_f))$ marchandises lourdes) et la pile contenant le moins de marchandises légères exclusivement de l'étiquette arrière ($F(l(E_b) + JTAP(E_b))$ marchandises légères) sont retirées pour pouvoir fusionner les deux étiquettes. Les marchandises légères retirées de l'étiquette arrière peuvent être placées dans la pile mixte de l'étiquette avant, en premier dans les espaces libres ($ITAP(E_f)$) puis à la place des marchandises lourdes de l'étiquette avant restantes ($a_H(E_f) - F(l(E_f) + ITAP(E_f))$). Ces marchandises lourdes sont déplacées dans la pile mixte de l'étiquette arrière. Si ce dernier mouvement n'est pas possible car la pile mixte est pleine, alors le critère (3.50) n'est pas vérifié. La place disponible est donc $ITAP(E_f) + a_H(E_f) - F(l(E_f) + ITAP(E_f))$. Par symé-

trie, la place disponible pour les marchandises lourdes retirées de l'étiquette avant est $JTAP(E_b) + a_L(E_b) - F(l(E_b) + JTAP(E_b))$. Les conditions à vérifier pour la fusion sont donc :

— $ITAP(E_f) + a_H(E_f) \geq F(l(E_f) + ITAP(E_f)) + F(l(E_b) + JTAP(E_b))$
— $JTAP(E_b) + a_L(E_b) \geq F(l(E_f) + ITAP(E_f)) + F(l(E_b) + JTAP(E_b))$

Si $A = Q + 3k$, alors la fusion n'est pas possible :

$$l(E_f) + l(E_b) \leq Q$$
$$\Leftrightarrow\ k \leq ITAP(E_f) - F(l(E_f) + ITAP(E_f)) + JTAP(E_b) - F(l(E_b) + JTAP(E_b))$$

Comme $F(l(E_f) + ITAP(E_f))$, $F(l(E_b) + JTAP(E_b)) \geq 1$:

$$ITAP(E_f) + JTAP(E_b) \geq k + 2$$

et $ITAP(E_f) \leq k - 1$, d'où $ITAP(E_f) > 0$ et $JTAP(E_b) > 0$. De même, $F(l(E_f) + ITAP(E_f)) < k$ et $F(l(E_b) + JTAP(E_b)) < k$. Il y a donc quatre piles incomplètes. en retirant ces quatre piles, les deux étiquettes peuvent fusionner et il reste une pile vide. Les quatre piles incomplètes doivent rentrer dans cette pile vide car toutes les autres piles sont pleines. C'est impossible car les marchandises légères de l'étiquette avant sont forcément en dessous des marchandises lourdes de l'étiquette arrière.

Si $A \geq Q + 4k$, la fusion n'est pas possible car en retirant les quatre piles partiellement remplies, la fusion est possible mais il n'y a plus de place pour placer les marchandises retirées.

Le tableau 5.1 résume les différents cas possibles. En implémentant tous ces cas possibles, un algorithme bidirectionnel est obtenu pour le VRPTWFC sans connaitre a priori la hauteur des piles.

Tableau 5.1 Table de fusion des étiquettes

A	Résultat
$\leq Q$	La fusion est toujours possible
$= Q + k$	Si $a_L(E_b) = k - F(a_H(E_b)) - JTAP(E_b)$, la fusion est possible si $k - F(a_H(E_b)) \geq F(l(E_f) + ITAP(E_f))$ Si $a_H(E_f) = k - F(a_L(E_f)) - ITAP(E_f)$, la fusion est possible si $k - F(a_L(E_f)) \geq F(l(E_b) + JTAP(E_b))$ Sinon il faut vérifier : $a_L(E_b) > k - F(a_H(E_b)) - JTAP(E_b)$ $a_H(E_f) > k - F(a_L(E_f)) - ITAP(E_f)$ Sinon la fusion est impossible.
$= Q + 2k$	Il faut vérifier : $a_L(E_b) > k - JTAP(E_b) - F(a_H(E_b))$ $a_H(E_f) > k - ITAP(E_f) - F(a_L(E_f))$ $ITAP(E_f) + a_H(E_f) \geq F(l(E_f) + ITAP(E_f)) + F(l(E_b) + JTAP(E_b))$ $JTAP(E_b) + a_L(E_b) \geq F(l(E_f) + ITAP(E_f)) + F(l(E_b) + JTAP(E_b))$ Sinon la fusion est impossible
$\geq Q + 3k$	La fusion est impossible

5.2.5 Recherche TABU

Pour la recherche TABU, le mouvement qui consiste à retirer un client est toujours légitime mais l'ajout d'un client peut être problématique. Le mieux serait de parcourir la nouvelle route et de vérifier qu'il n'y a jamais une situation interdite mais ce procédé risque d'être coûteux en temps. En connaissant les valeurs finales des composantes associées à une étiquette finale E, nous avons trouvé le critère suivant :

— Si le client i est un client lourd ($i \in H$) et si $l(E)+ITAP(E)+q_i < Q$ alors la nouvelle route respecte la contrainte de fragilité.

5.3 Résultats expérimentaux

Les tests ayant été menés uniquement avec notre algorithme bidirectionnel, il n'est pas possible de le comparer avec un autre algorithme pour mesurer sa performance. Nous avons mené les tests avec les instances de Solomon de R101 à R112, en ne faisant varier que le nombre de clients entre 50, 75 et 100. La probabilité pour un client d'être léger est de $0,5$ et la capacité du véhicule est fixé à 120. Pour un même quadruplet, 10 instances sont créées pour

compenser l'aléatoire lors du tirage des clients lourds et des clients légers. Chaque instance est testée avec les quatre hauteurs de piles suivantes : 2, 3, 4, et 5. La capacité étant fixée, le nombre de piles varient en fonction de la hauteur. Dans ce chapitre, nous ne considérons que les instances classiques, les instances réduites ne sont pas générées.

5.3.1 Résultats globaux

Les performances globales de l'algorithme sont résumées dans le tableau 5.2 selon le nombre de clients. Ce tableau montre que notre algorithme est globalement efficace pour les instances à 50 ou à 75 clients sauf les plus difficiles R104 et R108. Par contre, avec 100 clients, seules les instances simples sont résolues. De plus, à part pour R108, le résultat est assez binaire, soit les 40 instances sont résolues, soit aucune ne l'est, les performances de l'algorithme dépendent plus du nombre de clients que de la hauteur des piles. Cet algorithme est moins performant que celui présenté au chapitre précédent car la taille des piles n'est pas connue à l'avance. Or la taille des piles est un élément important lors de la construction des routes. En effet selon la taille des piles, une route peut être interdite à différents moments. Si la taille des piles est de 2, c'est la dernière marchandise collectée qui détermine si la route est interdite ou non. Si la taille des piles est plus grande, la route peut être interdite plus en amont.

Tableau 5.2 Nombre d'instances résolues selon le nombre de clients

	R101	R102	R103	R104	R105	R106	
n=50	40	40	40	40	40	40	
n=75	40	40	40	0	40	40	
n=100	40	40	40	0	40	0	
	R107	R108	R109	R110	R111	R112	Total
n=50	40	1	40	40	40	40	441
n=75	40	2	40	40	40	40	402
n=100	40	0	40	40	0	0	280

5.3.2 Statistiques sur le déroulement de l'algorithme et les solutions

Les principales statistiques de notre algorithme sont présentées dans le tableau 5.3. Les statistiques sont les mêmes qu'au chapitre précédent et les valeurs moyennes sont issues uniquement des instances résolues.

Dans ce tableau, le temps moyen de résolution n'augmente pas quand on passe de 50 à 75 clients. Ce résultat peut s'expliquer par le fait que les instances R104 ne sont plus résolues

pour $n = 75$ et qu'elles étaient très longues à résoudre pour $n = 50$. Le temps de résolution augmente quand $n = 100$. Les écarts d'intégrité sont peu élevés ce qui se reflète dans le nombre de nœuds de branchement faible. Peu de coupes de capacité sont générées, surtout des coupes de 2-chemins et en majorité des coupes de sous-ensemble. La proportion de temps consacrée à résoudre le sous-problème est toujours majoritaire autour de 90%, le sous-problème est donc bien la plus grande difficulté pour notre algorithme. Le pourcentage de véhicules pleins n'est pas très élevé car la contrainte de fragilité à tendance à interdire les véhicules pleins. Comme il y a peu de véhicules pleins, le nombre de clients par route est faible.

Tableau 5.3 Statistiques de l'algorithme pour le VRPTWFC

k	n	#opt	t(s)	gap(%)	#N	#cc	#2pc	#src	tSP(%)	#C	VP(%)
2	50	110	185	0.93	22.2	1.0	7.8	19.7	91	5.7	5
	75	101	171	0.76	8.2	1.2	2.3	19.0	92	6.1	9
	100	60	337	0.49	25.0	0.8	2.2	40.7	89	6.0	6
3	50	110	189	0.93	21.9	1.0	7.4	19.4	92	5.7	5
	75	101	183	0.77	8.6	1.3	2.8	19.4	91	6.1	9
	100	60	350	0.49	25.6	0.8	2.1	41.2	89	6.0	6
4	50	111	206	0.94	22.0	1.0	8.0	20.5	92	5.7	5
	75	100	162	0.76	9.4	1.3	2.2	19.7	92	6.1	9
	100	60	346	0.49	24.4	0.9	2.4	40.5	89	6.0	6
5	50	110	181	0.93	20.9	1.0	7.7	19.1	92	5.7	5
	75	100	164	0.76	8.9	1.3	2.3	19.1	91	6.1	9
	100	60	355	0.49	25.5	0.9	2.3	41.3	89	6.0	6

5.3.3 Influence de la hauteur des piles

Mis à part pour R108, la hauteur des piles ne change pas le nombre d'instances résolues, l'influence de ce paramètre semble donc faible. Pour obtenir un résultat, nous allons regarder, dans le tableau 5.4, le temps moyen de résolution des 10 instances selon chaque hauteur de pile pour les instances à 75 clients. Nous ne nous intéressons pas aux résultats pour les instances de R101 à R104 car les temps de calcul sont tous égaux.

Les résultats ont été arrondis au dixième de seconde, sauf si la limite de temps était atteinte (dans ce cas, le temps de calcul prend la valeur 3600). Dans le tableau, les temps moyens de calcul sont plus longs si la hauteur des piles est grande : R107 et R108 ont les plus longs temps de calcul pour une hauteur de 4 et R108 à R112 ont les plus longs temps de calcul pour une hauteur de 5. À l'opposé, pour une hauteur de pile 2, on retrouve les temps moyens de calcul les plus faibles pour les instances R107 à R112. Notre algorithme a plus de difficultés

Tableau 5.4 Influence de la hauteur des piles

k	R105	R106	R107	R108	R109	R110	R111	R112
2	3.6	17.2	198.2	3561.7	141.1	141.4	344.1	532.0
3	3.4	17.1	210.4	3590.3	148.8	167.1	348.6	573.4
4	3.5	16.4	307.3	3600.0	147.4	160.0	396.5	569.4
5	3.6	15.8	208.2	3600.0	153.4	189.2	434.77	609.7

à résoudre les instances quand les piles sont hautes. Ce résultat semble logique car lorsque les piles sont hautes, une route peut être interdite avec moins de marchandises collectées que lorsque les piles sont basses.

On peut également regarder le nombre d'instances dont la solution est différente de celle obtenue pour le VRPTW, selon la hauteur et l'augmentation moyenne de coût induite. Les tableaux 5.5 et 5.6 présentent ces informations selon le nombre de clients considérés sur l'ensemble des instances résolues (voir tableau 5.2). Le nombre d'instances problématiques augmentent avec la hauteur des piles. Ce résultat peut s'expliquer par le fait que plus les piles sont hautes, moins il y a de piles, le risque de superposition lourd/léger augmente donc. Cependant cette augmentation du nombre d'instances problématiques ne se traduit pas forcément par une augmentation plus importante du coût de la solution comme le montre le tableau 5.6.

Tableau 5.5 Nombre d'instances problématiques selon la hauteur des piles

k	2	3	4	5
n=50	3	5	4	6
n=75	3	4	9	10
n=100	3	1	2	3

Tableau 5.6 Augmentation absolue du coût selon la hauteur des piles

k	2	3	4	5
n=50	1.6	0.4	1.8	0.7
n=75	1.0	3.1	1.9	2.2
n=100	0.7	2.1	1.1	1.4

5.3.4 Comparaison avec le VRPTW

Pour comparer avec le VRPTW, on regarde les instances à 75 clients en enlevant les instances R104 et R108 car elles ne sont pas résolues entièrement. Le tableau 5.7 présente le temps moyen de calcul et le coût moyen selon que les instances soient résolues avec le VRPTW ou le VRPTWFC. L'augmentation en temps reste inférieure à 50% et l'augmentation en coût est extrêmement limitée.

Tableau 5.7 Comparaison VRPTW et VRPTWFC

	t(s)	coût
VRPTW	105	1292.72
VRPTWFC	154	1292.76

Si on ne regarde que les instances où la solution est différente entre le VRPTW et VRPTWFC, l'augmentation de coût est de 2.1 unités, soit encore une augmentation très faible par rapport au coût de la solution du VRPTW.

CHAPITRE 6 CONCLUSION

Dans ce dernier chapitre, nous revenons sur les travaux présentés dans ce mémoire, les résultats théoriques, les algorithmes proposés et enfin les résultats expérimentaux obtenus. Ce chapitre sera également l'occasion d'évoquer les limites de nos travaux et également les développements futurs possibles.

6.1 Synthèse des travaux

Dans ce mémoire, nous avons développé un modèle de collecte de marchandises lourdes et légères stockées dans des piles tel qu'une marchandise lourde ne se retrouve pas sur une marchandise légère. Dans ce modèle, les marchandises collectées auprès d'un client peuvent être placées dans différentes piles mais elles ne peuvent pas être déplacées après leur collecte. La difficulté de ce problème est le nombre de possibilités de placement qui rend difficile l'utilisation d'algorithmes à étiquettes. Le résultat principal de ce mémoire est d'avoir trouvé des méthodes pour propager les étiquettes avec une multiplication limitée selon les différentes dispositions des marchandises, l'une pour des piles de taille 2, l'autre sans connaitre la taille des piles a priori. Ainsi l'ajout de la contrainte de fragilité a pu être traité sans multiplication des étiquettes, seule la vitesse de propagation des étiquettes a été ralentie. De plus, ces méthodes de propagation ont permis de conserver la propagation bidirectionnelle qui réduit considérablement le temps passé dans le sous-problème.

Ces méthodes ont été mises en place dans un algorithme BPC afin de les tester sur les instances de Solomon modifiées pour comporter des marchandises lourdes et légères. Les résultats obtenus par nos algorithmes sur ces instances ont démontré la force des méthodes proposées, notamment pour le cas des piles de taille 2. Dans ce cas, nous avons pu comparer notre algorithme à une version plus naïve multipliant les étiquettes et constaté la pertinence de notre approche.

6.2 Limitations de la solution proposée

La première critique de notre modèle est sa simplicité dans la disposition des marchandises. Toutes les marchandises collectées sont de même volume et les collectes chez chaque client

peuvent être réparties entre les différentes piles. Ces deux simplifications sont rarement appliquées dans des problèmes de collecte à piles multiples et, au lieu d'être une contrainte, elles aident à résoudre le problème. De plus, si on retire l'une ou l'autre de ces simplifications, le problème devient plus complexe et les méthodes proposées dans ce mémoire ne sont plus applicables.

Cette recherche s'est focalisée sur l'intégration de la contrainte de fragilité dans un algorithme à étiquettes mais cette contrainte peut avoir d'autres impacts. Des coupes peuvent être rajoutées grâce à cette contrainte pour renforcer les relaxations linéaires. De la même manière, les recherches TABU peuvent être améliorées, rendant la génération de nouvelles routes plus efficace.

Enfin, la question sur le nombre d'instances pour lesquelles la contrainte de fragilité est une contrainte réelle limite l'impact de la solution proposée de ce mémoire. Pour certaines instances, la solution sans prendre en compte la contrainte de fragilité est également la solution en tenant compte de la contrainte de fragilité. Cette solution étant plus facile à obtenir, notre travail peut paraître d'une utilité très relative. Toutefois, résoudre une instance sans tenir compte de la contrainte de fragilité puis vérifier ensuite si la solution obtenue respecte la contrainte de fragilité peut être risqué si la solution obtenue ne respecte pas la contrainte de fragilité. Dans ce cas, il n'est pas possible de corriger la solution obtenue sans perdre l'optimalité. Notre algorithme est sûrement un peu moins rapide que la méthode proposée ci-dessus mais il a l'avantage de nous garantir l'optimalité quelle que soit la difficulté de l'instance.

6.3 Améliorations futures

Comme écrit dans la section précédente, il est sûrement possible d'améliorer les algorithmes proposés en améliorant les coupes utilisées pour tenir compte de la contrainte de fragilité ou en améliorant la recherche TABU. De plus, dans le cas des piles de taille 2, une recherche sur le nombre de clients avec un nombre pair de marchandises à collecter devrait permettre d'améliorer la construction des solutions.

Il faudrait aussi étudier les autres méthodes de recherche exactes (*Branch-and-Cut* et énumération implicite) et comment intégrer la contrainte de fragilité dans ces algorithmes. Il serait

ainsi possible de comparer les trois méthodes et d'obtenir la plus efficace pour résoudre ce problème.

Il est possible de complexifier la répartition en imposant que les marchandises collectées chez un même client soient toutes dans la même pile. Une contrainte de ce type obligerait à revoir le respect de la contrainte de manière importante.

Enfin, dans cette recherche, nous ne nous sommes intéressées qu'au problème de cueillette mais le problème de cueillette et livraison est également très intéressant. Il pose plus de difficultés car il faut s'intéresser au chargement au début et à la fin de chaque route mais aussi à certains moments à l'intérieur de chaque route. De plus dans ce cas, la découpe en pile de la cale devient plus contraignante.

RÉFÉRENCES

R. K. Ahuja, Ö. Ergun, J. B. Orlin, et A. P. Punnen, “A survey of very large-scale neighborhood search techniques”, *Discrete Applied Mathematics*, vol. 123, no. 1, pp. 75–102, 2002.

C. Archetti et M. G. Speranza, “A survey on matheuristics for routing problems”, *EURO Journal on Computational Optimization*, vol. 2, no. 4, pp. 223–246, 2014.

R. Baldacci, A. Mingozzi, et R. Roberti, “Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints”, *European Journal of Operational Research*, vol. 218, no. 1, pp. 1–6, 2012.

J. F. Bard, G. Kontoravdis, et G. Yu, “A branch-and-cut procedure for the vehicle routing problem with time windows”, *Transportation Science*, vol. 36, no. 2, pp. 250–269, 2002.

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.

O. Bräysy et M. Gendreau, “Vehicle routing problem with time windows, part I : Route construction and local search algorithms”, *Transportation Science*, vol. 39, no. 1, pp. 104–118, 2005.

T. Chabot, R. Lahyani, L. C. Coelho, et J. Renaud, “Order picking problems under weight, fragility and category constraints”, *International Journal of Production Research*, pp. 1–19, 2016.

M. Cherkesly, G. Desaulniers, et G. Laporte, “Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and last-in-first-out loading”, *Transportation Science*, vol. 49, no. 4, pp. 752–766, Nov. 2015.

G. Desaulniers, O. Madsen, et S. Ropke, “The vehicle routing problem with time windows”, dans *Vehicle routing : Problems, methods, and applications*, série MOS-SIAM Series on Optimization, P. Toth et D. Vigo, éds. SIAM, 2014, pp. 119 –159.

G. Desaulniers, J. Desrosiers, M. M. Solomon, F. Soumis, et D. Villeneuve, “A unified framework for deterministic time constrained vehicle routing and crew scheduling problems”, dans *Fleet Management and Logistics*, T. Crainic et G. Laporte, éds. Springer, 1998, pp. 57–93.

G. Desaulniers, F. Lessard, et A. Hadjar, “Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows”, *Transportation Science*, vol. 42, no. 3, pp. 387–404, 2008.

M. Desrochers, J. Desrosiers, et M. Solomon, “A new optimization algorithm for the vehicle routing problem with time windows”, *Operations Research*, vol. 40, no. 2, pp. 342–354, 1992.

G. Fuellerer, K. F. Doerner, R. F. Hartl, et M. Iori, “Metaheuristics for vehicle routing problems with three-dimensional loading constraints”, *European Journal of Operational Research*, vol. 201, no. 3, pp. 751–759, 2010.

H. Gehring et J. Homberger, “A parallel two-phase metaheuristic for routing problems with time windows”, *Asia-Pacific Journal of Operational Research*, vol. 18, no. 1, p. 35, 2001.

M. Gendreau, M. Iori, G. Laporte, et S. Martello, “A tabu search algorithm for a routing and container loading problem”, *Transportation Science*, vol. 40, no. 3, pp. 342–350, 2006.

B. Golden, S. Raghavan, et E. Wasil, *The Vehicle Routing Problem : Latest Advances and New Challenges*, série Operations Research/Computer Science Interfaces Series. Springer US, 2010.

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

S. Irnich et D. Villeneuve, “The shortest-path problem with resource constraints and k-cycle elimination for $k \geq 3$”, *INFORMS Journal on Computing*, vol. 18, no. 3, pp. 391–406, 2006.

M. Jepsen, B. Petersen, S. Spoorendonk, et D. Pisinger, “Subset-row inequalities applied to the vehicle-routing problem with time windows”, *Operations Research*, vol. 56, no. 2, pp. 497–511, 2008.

B. Kallehauge, J. Larsen, et O. B. Madsen, “Lagrangian duality applied to the vehicle routing problem with time windows”, *Computers & Operations Research*, vol. 33, no. 5, pp. 1464–1487, 2006.

B. Kallehauge, N. Boland, et O. B. Madsen, “Path inequalities for the vehicle routing problem with time windows”, *Networks*, vol. 49, no. 4, pp. 273–293, 2007.

N. Kohl, J. Desrosiers, O. B. Madsen, M. M. Solomon, et F. Soumis, “2-path cuts for the vehicle routing problem with time windows”, *Transportation Science*, vol. 33, no. 1, pp. 101–116, 1999.

G. Laporte, H. Mercure, et Y. Nobert, “An exact algorithm for the asymmetrical capacitated vehicle routing problem”, *Networks*, vol. 16, no. 1, pp. 33–46, 1986.

J. Lysgaard, “Reachability cuts for the vehicle routing problem with time windows”, *European Journal of Operational Research*, vol. 175, no. 1, pp. 210–223, 2006.

M. Padberg et G. Rinaldi, “Optimization of a 532-city symmetric traveling salesman problem by branch and cut”, *Operations Research Letters*, vol. 6, no. 1, pp. 1–7, 1987.

D. G. Pecin, C. Contardo, G. Desaulniers, et E. Uchoa, “New enhancements for the exact solution of the vehicle routing problem with time windows”, *INFORMS Journal on Computing*, Jan. 2016.

H. Pollaris, K. Braekers, A. Caris, G. K. Janssens, et S. Limbourg, “Vehicle routing problems with loading constraints : state-of-the-art and future directions”, *OR Spectrum*, vol. 37, no. 2, pp. 297–330, 2015.

G. Righini et M. Salani, “New dynamic programming algorithms for the resource constrained elementary shortest path problem”, *Networks*, vol. 51, no. 3, pp. 155–170, 2008.

L.-M. Rousseau, M. Gendreau, et D. Feillet, “Interior point stabilization for column generation”, *Operations Research Letters*, vol. 35, no. 5, pp. 660–668, 2007.

M. Schyns et V. Lurkin, “The airline container loading problem with pickup & delivery and multi doors”, dans *53rd AGIFORS Annual Proceedings 2013 : Annual Symposium and Study Group Meeting*. AGIFORS, 2013.

M. M. Solomon, “Algorithms for the vehicle routing and scheduling problems with time window constraints”, *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987.

C. D. Tarantilis, E. E. Zachariadis, et C. T. Kiranoudis, “A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 255–271, 2009.

P. Toth et D. Vigo, *Vehicle Routing : Problems, Methods, and Applications, Second Edition*, série MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, 2014.

T. Vidal, T. G. Crainic, M. Gendreau, et C. Prins, “Heuristics for multi-attribute vehicle routing problems : A survey and synthesis”, *European Journal of Operational Research*, vol. 231, no. 1, pp. 1–21, 2013.