Titre: Title:	Nouvelles coupes pour le problème de tournées de véhicule avec demandes stochastiques
Auteur: Author:	Alexandre Leuliet
Date:	2014
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Leuliet, A. (2014). Nouvelles coupes pour le problème de tournées de véhicule avec demandes stochastiques [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1603/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/1603/
Directeurs de recherche: Advisors:	Guy Desaulniers, Walter Rei, & Ola Jabali
Programme: Program:	Génie industriel

UNIVERSITÉ DE MONTRÉAL

NOUVELLES COUPES POUR LE PROBLÈME DE TOURNÉES DE VÉHICULE AVEC DEMANDES STOCHASTIQUES

ALEXANDRE LEULIET
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
(GÉNIE INDUSTRIEL)
DÉCEMBRE 2014

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

NOUVELLES COUPES POUR LE PROBLÈME DE TOURNÉES DE VÉHICULE AVEC DEMANDES STOCHASTIQUES

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

M. GENDREAU Michel, Ph. D., président
M. DESAULNIERS Guy, Ph. D., membre et directeur de recherche
M. REI Walter, Ph. D., membre et codirecteur de recherche
Mme JABALI Ola, Ph. D., membre et codirectrice de recherche
M. ERRICO Fausto, Ph. D., membre

REMERCIEMENTS

Je tiens tout d'abord à remercier mon directeur de recherche Guy Desaulniers qui m'a fait confiance en acceptant de me donner la charge de ce projet de recherche. Il a su trouver un juste équilibre en me laissant mon autonomie de travail d'une part, tout en assurant un suivi régulier d'autre part. Sa pédagogie et sa rigueur m'ont également été d'une grande aide.

Je remercie également mes deux co-directeurs Ola Jabali et Walter Rei qui ont tous les deux apporté leurs expertises dans le domaine étudié. La parfaite connaissance du sujet de la part d'Ola et la précision et rapidité de ses réponses m'ont été d'une aide précieuse. Je retiens de Walter sa créativité dans les idées proposées lors des différentes réunions et la pertinence de ses conseils.

Une grande partie du travail nécessaire à la production de ce mémoire a été consacrée au développement informatique des solutions théoriques envisagées. C'est pourquoi l'aide apportée par Serge Bisaillon a été d'une importance fondamentale. J'ai particulièrement apprécié sa rapidité de compréhension des problèmes rencontrés, son champ de compétence dans un domaine aussi vaste que l'informatique tout comme sa compréhension parfaite des concepts mathématiques manipulés.

Enfin, je tiens à remercier ma famille et mes amis, qu'ils soient à Montréal ou non, pour leur présence tout au long de ce projet. Ils ont chacun contribué à faire de ces deux années de recherche une expérience agréable et enrichissante.

RÉSUMÉ

Dans ce mémoire, on se propose de résoudre le problème de tournées de véhicules avec demandes stochastiques à l'aide de l'algorithme L-shaped en nombres entiers. Des coupes de type LBF (Lower Bounding Functionals) sont générées pour accélérer la résolution.

Le problème est semblable aux problèmes de tournées de véhicules classiques mais considère que les demandes des clients sont des variables aléatoires dont la valeur n'est révélée qu'au moment de leur visite. Ces variables suivent une loi Normale, sont indépendantes et identiquement distribuées.

Le problème est formulé comme un programme stochastique en deux étapes en nombres entiers. Les variables de décisions de première étape servent à définir des routes a priori qui devront minimiser le coût total espéré de parcours. Chaque fois qu'un véhicule arrive chez un client, si la demande ne peut pas être servie en totalité alors on dit qu'un échec apparaît. Dans une telle situation, les décisions de seconde étape sont prises et consistent à effectuer un aller-retour entre le client courant et le dépôt pour se recharger ou se décharger en marchandise. Ce recours est communément appelé le recours simple.

Une étape préliminaire à la résolution consiste à relaxer le modèle en enlevant temporairement les contraintes de capacité et d'élimination de sous-tours ainsi que les contraintes d'intégrité. La fonction de recours est remplacée par une variable réelle positive représentant une borne inférieure sur le coût de recours espéré. L'intégrité des variables est retrouvée grâce à une procédure d'énumération implicite. Les contraintes de capacité et de sous-tours violées sont ajoutées dynamiquement au modèle chaque fois qu'elles sont identifiées. Des coupes d'optimalité sont générées et assurent la convergence de l'algorithme vers la solution optimale.

Pour accélérer le processus de résolution par rapport à ce qui existe déjà dans la littérature, on développe deux nouvelles familles de coupes LBF. Les premières sont basées sur l'identification de chaînes dans les solutions intermédiaires rencontrées aux différents nœuds de l'arbre d'énumération. Ces coupes sont ajoutées chaque fois qu'on en identifie une qui est violée par la solution courante. Elles imposent à la borne inférieure sur le coût de recours de prendre une certaine valeur si la solution courante visite les clients des chaînes dans le même ordre. Les deuxièmes sont basées sur des ensembles non structurés de nœuds. Elles sont générées de façon semblable aux coupes LBF précédentes et présentent l'avantage d'être

actives sur un plus grand ensemble de solutions entières et fractionnaires. La contrepartie à cela est la faiblesse de la borne associée.

Pour identifier les coupes violées, on a développé des algorithmes de séparation heuristiques basés sur un même principe. Celui-ci préconise leur construction en augmentant la taille de l'ensemble des variables concernées de façon itérative jusqu'à ce que la borne de la coupe associée soit suffisante pour constituer une coupe violée.

Enfin, on produit des tests numériques sur un ensemble riche d'instances qui prouvent l'efficacité de nos travaux puisque nous résolvons 13 nouvelles instances de la littérature en moins de 10000 secondes. De plus, on réduit les temps de calcul des instances résolues de 30% en moyenne.

ABSTRACT

In this master's thesis, we intend to solve the vehicle routing problem with stochastic demands by means of an integer L-shaped algorithm. We use lower bounding functionals (LBF) to speed up the resolution process.

The problem is similar to classic vehicle routing problems, except that customer demands are random variables which values are only revealed when they are visited. These variables follow a normal distribution and are independantly and equally distributed.

The problem is formulated as an integer two-stages stochastic programming model. The first stage decision variables are used to define a priori routes designed to minimize the total expected recourse cost. Each time a vehicle reaches a client, if his demand can't be fully served then a failure is said to occur. In such a situation, second-stage decisions are made and consist in returning to the depot to load or unload goods and going back to the current client to resume the planned route. This recourse action is commonly referred to as the simple recourse.

A preliminary step to the resolution consists in relaxing the model by temporarily removing the capacity and subtour elimination constraints, as well as the integrality requirements. The recourse function is replaced by a positive real variable representing a lower bound on the expected recourse cost. Integrality is recovered by means of a branch-and-cut framework. Capacity and subtour elimination constraints are added dynamically as long as they are found to be violated. Optimality cuts are generated and guarentee the algorithm convergence toward the optimal solution.

To speed up the resolution process from what already exists in the literature, we develop two new families of LBF cuts. The first ones are based on the identification of chains within the temporary solutions met in the different nodes of the branching tree. These cuts are added each time a violated one is identified. They force the lower bound on the expected recourse cost to take a certain value if the current solution visits the chain's clients in the same order. The second ones are based on unstructured vertex sets. They are generated similarly to the previous cuts and have the advantage of being active on a wider range of fractionnal and integer solutions. The trade-off is the relative weakness of the lower bound associated.

To identify violated cuts, we developped heuristic separation procedures based on one principle. Their purpose is to construct cuts by increasing iteratively the size of the involved variables set until the associated lower bound is enough to form a violated cut.

Finally, we produce numerical experiments involving a rich set of instances which prove the efficiency of our work as 13 new instances from the literature could be solved to optimality in less than 10000 seconds. Moreover, we reduce the computing times of the solved instances by 30% on average.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

LISTE DES SIGLES ET ABRÉVIATIONS

BC	Branch and Cut
CVRP	Capacitated Vehicle Routing Problem
EV	Expected Value Solution
EEV	Expected result of using Expected Value
LBF	Lower Bounding Functionals
m-TSP	m Traveling Salesman Problem
PC	Problème Courant
PTSP	Probabilistic Traveling Salesman Problem
RV	Recourse Problem
SVRP	Stochastic Vehicle Routing Problem
UVS	Unstructured Vertex Sets
VRP	Vehicle Routing Problem
VRPSD	Vehicle Routing Problem with Stochastic Demands
VRPTW	Vehicle Routing Problem with Time Windows
VSS	Value of the Stochastic Solution

CHAPITRE 1

INTRODUCTION

1.1 Contexte et définitions

Les outils mathématiques de l'optimisation et de la recherche opérationnelle ont permis de résoudre un grand nombre de problèmes industriels et sont, de nos jours, essentiels pour la plupart des entreprises. Les problèmes de transport et de la logistique associée constituent une part importante des travaux de ce domaine dont l'apport permet notamment aux entreprises bien connues comme Fedex ou encore la Poste d'assigner des itinéraires à leur flotte de véhicules tout en minimisant leurs coûts d'opération. Ces coûts correspondent souvent à des coûts de transport associés à la consommation de carburant et à l'usure du véhicule et peuvent donc en première approximation être considérés proportionnels à la distance parcourue. En plus des compagnies spécialisées dans le transport, on peut noter que pour n'importe quel produit, les coûts de transport constitueront forcément une partie non négligeable du coût total de fabrication. Cela rend donc indispensable l'étude de ce problème.

Le problème de tournées de véhicules (VRP pour Vehicle Routing Problem en anglais) est le problème de base qui étudie ce phénomène. Il modélise la situation dans laquelle une flotte de véhicules doit effectuer des livraisons ou des collectes chez des clients. Le nombre de véhicules peut être fixé ou variable, le but étant de visiter l'ensemble des clients en minimisant la distance totale. Les routes (ou tournées) ainsi définies vont devoir respecter les contraintes classiques du problème :

- Chaque client est visité une et une seule fois.
- Chaque route commence et se termine au dépôt.

Si les clients ont chacun une demande fixée et que les véhicules ont une capacité de stockage maximum, on parle alors du CVRP (Capacitated VRP). Le problème aura donc comme contrainte supplémentaire de limiter la demande cumulée de la tournée à la capacité du véhicule qui le parcourt. Une autre variante du problème consiste à considérer une fenêtre de temps pour chaque client pendant laquelle il doit être visité. On parle alors de VRPTW (VRP with time windows).

Bien que les algorithmes existant pour ces problèmes aient une grande utilité, ils supposent

que les paramètres sont déterministes et entièrement connus avant la résolution. En réalité, plusieurs paramètres peuvent être incertains ou variables et donc être considérés comme stochastiques. L'ensemble des problèmes de tournées de véhicules avec aspect stochastique forme la classe des SVRP (Stochastic Vehicle Routing Problem). Le paramètre stochastique le plus naturel à imaginer est peut-être le temps de trajet entre les différents clients, qui sera soumis aux aléas du trafic routier, des intempéries, des travaux, etc. Il sera d'autant plus utile de prendre en compte cet aspect aléatoire si des fenêtres de temps doivent être respectées. Les autres paramètres aléatoires classiques peuvent être la présence des clients ou leurs demandes, comme le montre l'étude de Gendreau *et al.* (1996a).

On étudie dans ce mémoire le VRP avec demandes stochastiques ou encore VRPSD (VRP with stochastic demands). C'est ce dernier qui a été le plus étudié parmi les variantes du SVRP. Il a l'avantage de prendre en compte un phénomène présent dans beaucoup d'applications. On pense notamment au ramassage des ordures dans les villes. La quantité de déchet de chaque habitant varie de semaine en semaine et ne pas prendre ce paramètre en compte peut donner de mauvaises solutions, comme cela est évoqué par Louveaux (1998). Il en est de même pour les transferts d'argent liquide entre banques, ou encore les livraisons de boisson ou nourriture à différents commerces.

On suppose donc que l'on connait les lois de probabilité des demandes de chaque client. Il est usuel pour le VRPSD de considérer des lois de probabilité qui comportent peu de paramètres comme la loi Normale ou celle de Poisson. On se limitera ici à la loi Normale, pour laquelle le problème est plus difficile à résoudre que pour la loi de Poisson qui ne comporte qu'un seul et même paramètre qui est égal à son espérance ainsi qu'à sa variance. Une autre raison à cela est que des variables aléatoires suivant des lois Normales sont souvent utilisées pour formuler des phénomènes stochastiques. On suppose de plus que ces variables aléatoires représentant la demande des clients sont indépendantes et identiquement distribuées. Cela peut éloigner le problème de la réalité, mais semble être une alternative satisfaisante pour un bon nombre d'applications. En prenant en compte cet aspect stochastique du problème, on cherchera maintenant à construire des routes pour lesquelles l'espérance de la demande cumulée ne dépasse pas la capacité du véhicule, celle-ci étant supposée identique pour chacun des véhicules. Ainsi, lorsque la valeur des demandes aléatoires sera révélée, il pourra arriver certains jours que les demandes soient telles que le véhicule soit plein avant d'avoir terminé sa tournée. On nomme un tel évènement un échec. La contrainte selon laquelle l'espérance de la demande cumulée d'une route ne doit pas dépasser la capacité du véhicule pourrait être enlevée, mais elle permet d'éviter des tournées qui échouent systématiquement alors que

d'autres véhicules sont sous-utilisés.

Un point important à noter est que l'on suppose que la valeur de la demande n'est révélée qu'au moment du passage du véhicule chez le client. C'est une des raisons pour laquelle la résolution du modèle se fera à l'aide de l'optimisation dite a priori, élément important de l'optimisation stochastique. En effet, la résolution va se faire en deux étapes. Dans la première, on assigne des routes aux différents véhicules en minimisant l'espérance de la distance parcourue. Les demandes seront révélées à mesure que le véhicule parcourt les clients sur son chemin. C'est alors que sont prises les décisions de seconde étape : en cas d'échec, une stratégie de recours est implémentée. Elle consiste dans notre problème à effectuer un aller-retour entre le client qui a été partiellement servi et le dépôt. L'objectif à minimiser sera donc composé d'une partie déterministe qui est la distance totale parcourue pour effectuer les routes fixées a priori ainsi que d'une partie stochastique qui est l'espérance des coûts de recours. Cette partie stochastique s'appelle la fonction de recours. Son utilisation constitue une méthode classique pour prendre en compte l'incertitude dans un programme linéaire. D'autres stratégies de recours peuvent être choisies et seront évoquées dans la revue de littérature. Le fait d'avoir des routes définies a priori présente notamment l'avantage opérationnel pour les chauffeurs d'avoir des routes fixes auxquelles ils sont habitués. Il existe d'autres stratégies, comme la réoptimisation dynamique des routes, qu'on évoquera plus en détail au chapitre 2. Celle qu'on utilise peut donc constituer une référence avec laquelle les autres vont pouvoir se comparer.

Cette formulation du problème en deux étapes est une formulation classique de l'optimisation stochastique et peut se résoudre à l'aide de la méthode L-shaped en nombres entiers. Une étape préliminaire de cet algorithme consiste à effectuer une relaxation du modèle initial. Ainsi, l'espérance du coût de recours, qui n'est pas facilement décomposable en combinaison linéaire de variables de décision, sera modélisée par une unique variable θ réelle positive qui sera une borne inférieure sur le coût réel. Les contraintes de capacité et de sous-tours ainsi que celles d'intégrité des variables seront également relaxées. Les coupes violées seront ajoutées dynamiquement au problème et l'intégrité des variables sera récupérée à travers un algorithme d'énumération implicite et d'ajout de coupes de type Branch and Cut (BC).

1.2 Objectifs

Il est prouvé que l'algorithme de la méthode L-shaped en nombres entiers converge en un nombre fini d'itérations vers la solution optimale. Cependant, bien que le temps de résolu-

tion soit fini, il est nécessaire pour des applications pratiques que celui-ci soit raisonnable. Sa rapidité repose en grande partie sur l'existence de coupes de type LBF (Lower bounding functionnals) ajoutées dynamiquement au problème et qui auront pour but de renforcer la borne inférieure sur le coût de recours. Il existe déjà des coupes LBF efficaces qui sont basées sur l'identification de routes partielles dans les solutions intermédiaires rencontrées au cours de la résolution du problème. Ce mémoire aura pour but d'accélérer les temps de calcul de l'algorithme de résolution en proposant de nouvelles coupes LBF. Celles-ci seront basées sur l'identification de chaînes d'arêtes entières parmi les solutions intermédiaires, ainsi que d'ensembles non structurés de clients.

Pour cela, il faudra introduire l'expression littérale des contraintes que l'on ajoutera au modèle. De plus, l'identification des coupes violées par les solutions intermédiaires se fera à l'aide d'algorithmes de séparation qu'il faudra implémenter efficacement.

1.3 Plan du mémoire

Après avoir introduit le sujet, nous allons présenter dans le chapitre 2 une revue de littérature des travaux effectués dans le domaine. Celle-ci présentera les différents apports pertinents réalisés en lien avec notre sujet de recherche. On se concentrera premièrement sur les problèmes de tournées de véhicules stochastiques, avant d'insister sur le VRPSD en particulier. Puis, nous décrirons précisément les travaux de recherche en lien direct avec le cadre de ce mémoire.

Le chapitre 3 présentera le modèle mathématique choisi pour représenter la situation réelle décrite dans l'introduction. Nous exposerons dans un premier temps le formalisme utilisé dans le cadre de la programmation stochastique utilisant les fonctions de recours. Celui-ci constitue une référence du domaine et est décrit exhaustivement par Birge et Louveaux (2011). Nous montrerons ensuite comment ce formalisme peut s'appliquer à la modélisation du VRPSD.

Le chapitre 4 constituera l'apport principal de ce mémoire et traitera de la méthode de résolution du problème. Après avoir décrit dans une première section la relaxation du modèle, prérequis nécessaire pour l'utilisation de l'algorithme L-shaped en nombres entiers qui sera également présenté, nous détaillerons le calcul de la borne inférieure générale sur le coût de recours. Enfin, nous présenterons les coupes LBF les plus efficaces de la littérature et décriront leur principe avant d'expliciter la forme des nouvelles coupes que l'on introduit dans nos travaux. Nous mentionnerons leurs expressions littérales et leur mode de construction en

précisant les motivations des différents choix effectués.

Les différents tests numériques attestant des bonnes performances de nos travaux seront présentés dans le chapitre 5. Nous avons utilisé pour cela les instances de référence de la littérature pour comparer l'apport des coupes que l'on a développées. Les temps de calcul et le nombre d'instances résolues à l'optimalité font notamment partie des paramètres que l'on a relevés.

Finalement, le chapitre 6 résumera les différents apports de ce mémoire, identifiera les limites de la solution proposée et évoquera quelques axes de recherche qu'il serait intéressant d'étudier pour poursuivre et améliorer nos travaux.

CHAPITRE 2

REVUE DE LITTÉRATURE

Dans ce chapitre, nous allons présenter les différents travaux pertinents en lien avec le problème étudié. Nous commencerons par faire un tour d'horizon non exhaustif des problèmes de tournées de véhicules stochastiques, de leurs différentes versions ainsi que de leurs différentes méthodes de résolution. Dans un second temps nous nous concentrerons plus particulièrement sur les travaux de recherche qui ont un lien étroit avec ce mémoire.

2.1 Les problèmes de tournées de véhicules stochastiques

Les problèmes de transport constituent une partie importante des problèmes d'optimisation, et leur représentant phare, le VRP, a été très largement étudié. Depuis sa première apparition dans la littérature, via les travaux de Dantzig et Ramser (1959), ce problème NP-complet a constitué un défi pour bon nombre de chercheurs, comme le décrit exhaustivement Laporte (2009) dans son bilan sur les travaux effectués depuis cinquante ans dans le domaine. L'état de l'art permet actuellement de résoudre des problèmes dont la taille atteint 200 clients et 17 véhicules, comme cela est décrit par Baldacci *et al.* (2011).

Très tôt, on a cherché à intégrer des éléments stochastiques aux programmes linéaires classiques déterministes. En effet, bien que la recherche opérationnelle puisse attirer pour son simple intérêt mathématique, il faut garder à l'esprit qu'elle doit avoir une application utile pour le monde réel et l'industrie. Il faut donc pouvoir fournir des solutions efficaces et robustes quand la variance et l'incertitude de certains paramètres deviennent importants et méritent d'être pris en compte, ce qui est souvent le cas dans la réalité. C'est pourquoi on a intégré des paramètres aléatoires dans les modèles de programmation linéaire. Dantzig (1955) introduit ainsi la programmation stochastique pour laquelle il définit les processus de décision en plusieurs étapes. Il a été montré par Louveaux (1998) qu'utiliser des algorithmes déterministes dans lesquels les paramètres aléatoires étaient remplacés par leur espérance donnait de moins bonnes solutions que les modèles prévus pour prendre en compte l'aspect stochastique. Cet apport est décrit dans Birge et Louveaux (2011) comme la valeur de la solution stochastique. Typiquement, les paramètres pouvant varier dans le VRP peuvent être les temps de service et de trajets, la présence des clients ou la valeur de leurs demandes, comme cela est évoqué dans l'article de synthèse de Gendreau *et al.* (1996a).

Il est nécessaire de prendre en compte le caractère incertain des différentes durées présentes dans le problème, notamment si des contraintes précises sur les temps des tournées sont imposées. L'article de Laporte *et al.* (1992) décrit trois modèles visant à modéliser les temps de service et de parcours stochastiques. Les routes dont le temps total dépasse un certain seuil sont pénalisées. D'autres travaux prenant en compte des temps aléatoires ont été réalisés, comme ceux de Lambert *et al.* (1993) dont l'application concerne les trajets entre banques pour le déplacement des billets. Dans certaines applications, des fenêtres de temps dans lesquelles les clients demandent à être servis doivent être respectées. Ceci a été étudié par Jula *et al.* (2006) qui présentent une méthode heuristique basée sur la programmation dynamique utilisée pour résoudre le problème, sachant que les temps de parcours et de services sont stochastiques.

La présence des clients lors des tournées est également un paramètre pouvant être incertain. Le problème probabiliste du commis voyageur (PTSP pour probabilistic traveling salesman problem) étudié par Jaillet et Odoni (1988) propose de déterminer, de façon a priori, une tournée sur un ensemble de clients, sachant que seul un sous-ensemble sera visité avec une probabilité donnée. Le but est de minimiser l'espérance de la distance totale parcourue. Gendreau *et al.* (1995) considèrent également la présence des clients comme aléatoire, de même que leurs demandes. Modéliser les clients avec une présence stochastique peut être utile dans les applications pour lesquelles on souhaite servir un ensemble de clients de manière régulière, sans changer l'ordre de visite et en minimisant la distance moyenne parcourue. D'un point de vue opérationnel, garder le même ordre de visite peut faciliter le travail des chauffeurs.

Le VRPSD, dont les paramètres stochastiques sont les demandes des clients, a été la variante stochastique du VRP la plus étudiée. Comme décrit en introduction de ce mémoire, il existe de nombreuses applications pratiques pour ce problème. Pour un recensement complet des travaux effectués, on peut se référer à Gendreau *et al.* (1996a) et Cordeau *et al.* (2006). Le problème a été mentionné pour la première fois dans Tillman (1969). Il existe depuis deux approches principales pour résoudre ce problème, que ce soit de façon exacte ou heuristique. La première est une approche statique. Elle consiste à prendre des décisions a priori et prévoir des décisions de recours quand l'information stochastique est révélée : on utilise pour cela la programmation stochastique. La deuxième vision est une approche dynamique. Chaque décision est prise en considérant l'état du système, l'information déjà révélée ainsi que les prévisions futures à disposition : on utilise cette fois la programmation dynamique

stochastique. En général, les algorithmes de cette approche sont des heuristiques car il est difficile de résoudre les problèmes de manière exacte dans des temps convenables.

Comme le décrit Psaraftis (1995), les avancées technologiques dans les services de localisation et de communication permettent d'exploiter l'information en temps réel. Ainsi, les politiques de réoptimisation de la tournée prévue initialement se sont développées, comme c'est le cas dans Secomandi (2000) et Secomandi (2001). Tatarakis et Minis (2009) utilisent également la programmation dynamique pour déterminer une politique optimale indiquant le moment pour lequel il faut rentrer au dépôt, dans le cadre du VRPSD avec deux produits. Ils prouvent que la politique à suivre dépend d'un seuil, résultat étendu à un nombre quelconque de produits par Pandelis *et al.* (2012). Même si l'état de l'art concernant la méthode de réoptimisation permet de résoudre des instances impliquant jusqu'à 100 clients dans Secomandi et Margot (2009), il faut noter qu'on peut gérer uniquement les situations impliquant un seul véhicule. Dans le cas contraire, la taille de l'ensemble des états possibles du système devient trop grande et impossible à gérer dans des temps raisonnables. Ainsi, pour considérer tout de même des instances de plusieurs véhicules, Secomandi et Margot (2009) et Novoa et Storer (2009) partent du principe que l'affectation des clients aux différents véhicules a déjà été faite.

Il existe deux façons d'introduire des éléments stochastiques dans les modèles de programmation stochastique, à savoir les contraintes probabilistes et les fonctions de recours. Les deux façons fournissent une solution a priori qui peut être déterminée de manière exacte ou heuristique selon les algorithmes de résolution utilisés.

La philosophie des modèles à contraintes probabilistes est d'assurer la faisabilité des différents trajets avec une certaine probabilité. Il est en effet possible de traduire ce besoin de faisabilité sous forme d'inégalités mathématiques quand les distributions le permettent. En général, c'est le cas pour les distributions comportant peu de paramètres, comme la loi Normale ou de Poisson, qui ont deux et un seul paramètres, respectivement. Golden et Yee (1979) ont étudié le VRPSDPC (VRPSD with probabilistic constraints) dans lequel des contraintes probabilistes sont introduites pour plusieurs distributions des demandes. Le cas des demandes corrélées a également été étudié. À cette époque, on décrivait encore le VRPSD comme le stochastic VRP, signe qu'il était la variante la plus étudiée des VRP stochastiques. Le problème était résolu à l'aide d'une heuristique. Stewart et Golden (1983) décrivent eux aussi un modèle avec contraintes probabilistes pour le VRPSD dans lequel un échec apparaît quand la capacité d'un véhicule est insuffisante pour la demande totale d'une route. Laporte *et al.* (1989) montrent que les modèles à contraintes probabilistes peuvent se résoudre rela-

tivement facilement car ils peuvent être modélisés comme un VRP déterministe dans lequel des contraintes sont ajoutées et ainsi bénéficier de l'état de l'art très développé des VRP classiques.

L'impact de l'incertitude sur les demandes peut également être intégré dans l'objectif du modèle à l'aide d'une fonction de recours. C'est d'ailleurs ce que nous ferons dans ce mémoire. Pour le détail du formalisme utilisé, on peut se référer à Birge et Louveaux (2011). L'idée principale est qu'au lieu d'être déterministe, l'objectif va cette fois être l'espérance du coût engendré par les décisions de première étape. On choisit à l'avance le recours, c'est-à-dire la manière dont vont être prises les décisions de seconde étape quand l'information sur les demandes sera révélée. Plusieurs choix de recours peuvent être faits. Dans ce mémoire, on utilisera le recours traditionnel. Il consiste à effectuer un aller-retour entre le client courant et le dépôt chaque fois que la quantité de produits résiduelle du véhicule est insuffisante pour servir la demande du client. Les distributions des demandes que l'on va considérer étant continues, on n'aura jamais la situation dans laquelle la demande a été servie exactement à un client avant d'atteindre la capacité du véhicule. Ainsi, le trajet client-dépôt commencera et se terminera toujours au client courant et ne finira pas au client suivant de la tournée, ce qui pourrait être le cas pour des demandes discrètes. Ce recours a l'avantage d'être facile à mettre en place d'un point de vue opérationnel, et permet aux chauffeurs de s'habituer aux trajets qui leur sont assignés, car leurs tournées ne changent pas, contrairement aux méthodes de réoptimisation. Ce choix de recours a été fait par Gendreau *et al.* (1996b). L'heuristique de recherche taboue permet ici de trouver des solutions entières rapidement. Le recours traditionnel est utilisé également dans les travaux de Chepuri et Homem-de Mello (2005) et Goodson *et al.* (2012) notamment.

Si des méthodes de résolution différentes peuvent être utilisées pour le recours traditionnel, il existe également des recours différents. Parmi les méthodes différentes, mais qui utilisent le même recours, on a notamment Christiansen et Lysgaard (2007) qui utilise un Branch and Cut and Price. L'algorithme, qui supporte plusieurs véhicules et qui fournit une solution exacte, tire parti de la génération de colonnes qui permet de calculer l'espérance du coût de recours directement, sans attendre de solutions entières. On s'affranchit ainsi du besoin de calculer des bornes inférieures pour les solutions non entières. Ces travaux ont été repris et améliorés par Gauvin *et al.* (2014).

Parmi les recours différents, il y a celui de Yang *et al.* (2000). Les routes sont déterminées à l'avance, mais les décisions de recours se prennent dynamiquement. Chaque fois qu'on quitte

un client, on a la possibilité de faire un détour préventif au dépôt visant à diminuer le coût moyen de la tournée. Cet aller-retour a lieu lorsqu'ayant servi un client, la capacité résiduelle est inférieure à un certain seuil associé au client suivant. Ceci peut être avantageux dans les situations où il ne reste plus beaucoup de marchandise dans le véhicule et le client auquel on se trouve est plus proche du dépôt que le client suivant. Dans les travaux de Hvattum *et al.* (2006) et Hvattum *et al.* (2007), les clients peuvent appeler pendant la journée pour faire des commandes, ce qui rend incertain leurs localisations et leurs demandes. Ainsi, le problème est modélisé en plusieurs étapes et les variables de décision de recours d'une certaine période servent à modifier les routes planifiées dans les périodes précédentes en tenant compte des nouvelles demandes révélées. Un autre recours est utilisé dans Ak et Erera (2007). Dans un premier temps, on détermine les routes. On forme ensuite des paires de routes, l'une est de type I et l'autre de type II. Sur la route de type I, lorsque la capacité du véhicule n'est plus suffisante, il rentre simplement au dépôt pour finir sa tournée. Ses clients sont alors assignés à la route de type II. Celle-ci visite ses clients dans l'ordre prévu puis visite ceux ajoutés du type I à la fin dans le même ordre. Quand le véhicule est plein, on repasse au recours traditionnel. La recherche taboue est utilisée pour trouver de bonnes solutions rapidement. Les paires de routes ont également été utilisées dans Lei *et al.* (2012). Cette fois, la demande d'un client peut être partagée entre un véhicule et son partenaire pour éviter à l'un ou l'autre de devoir rentrer au dépôt.

2.2 Cadre du mémoire

Après avoir discuté de différents moyens de modéliser et de résoudre le VRPSD, on va maintenant se concentrer sur les choix faits dans le cadre de ce mémoire. Comme on l'a évoqué dans l'introduction, on modélise le problème en deux étapes. La solution a priori de première étape devra minimiser l'espérance du coût total, et le recours traditionnel sera utilisé quand un échec apparaîtra. La résolution se fera à l'aide de l'algorithme du L-shaped en nombres entiers, et des coupes de type LBF (Lower Bounding Functionals) seront utilisées pour accélérer la résolution.

Le concept de la solution a priori a été introduit dans Bertsimas (1988), Jaillet (1988) et Bertsimas *et al.* (1990). Bertsimas (1988) est un des premiers mémoires consacrés au VRPSD. Il apporte une série de bornes, de résultats asymptotiques et de propriétés pour les problèmes avec des demandes unitaires. Bertsimas *et al.* (1990) montrent que pour des noeuds uniformément distribués sur une surface, la qualité des solutions a priori obtenues est proche de

celles que fournissent les stratégies de réoptimisation.

La méthode L-shaped en nombres entiers permet de résoudre de façon exacte et en un nombre fini d'opérations les programmes stochastiques en nombres entiers. Elle est une extension de l'algorithme L-shaped de Van Slyke et Wets (1969) qui est lui-même une application aux programmes stochastiques de la décomposition de Benders (1962). La méthode L-shaped en nombres entiers est décrite dans Laporte et Louveaux (1993). On dit qu'un programme stochastique est en nombres entiers s'il contient des variables entières dans la première ou la deuxième étape. Dans ce mémoire, nous utiliserons une formulation du VRPSD dans laquelle la première étape reprend essentiellement les mêmes éléments que les formulations classiques à deux indices pour le CVRP. Ainsi, chaque variable indique le nombre de véhicules circulant entre deux noeuds du graphe, ces noeuds pouvant être des clients ou le dépôt. L'algorithme proposé permet de résoudre des modèles avec des variables de première étape binaires, les variables de seconde étape ainsi que les vecteurs aléatoires pouvant être discrets ou continus. L'algorithme se différencie de la méthode L-shaped de Van Slyke et Wets (1969) car il introduit une procédure BC permettant de prendre en compte les variables binaires de première étape. La procédure BC est différente des procédures BC habituelles, car on n'élague pas forcément un noeud quand une solution entière a été trouvée. Il faut en plus qu'on ait approximé correctement la fonction de recours $Q(x)$, voir Laporte et Louveaux (1993) pour le formalisme utilisé.

L'utilisation des coupes de type LBF permet d'accélérer la résolution du problème en fournissant une meilleure approximation de la fonction de recours pour un grand nombre de solutions. Ainsi, moins de coupes d'optimalité sont nécessaires, les bornes inférieures des noeuds de branchement sont plus élevées et un nombre restreint de solutions doit être énuméré avant de trouver la solution optimale. Selon Birge et Louveaux (2011), les coupes LBF sont simplement des inégalités valides qui s'appliquent à $Q(x)$. Il n'y a pas de conditions particulières devant être remplies, car la convergence est déjà assurée par les coupes d'optimalité. C'est sur ces inégalités que nous allons travailler dans ce mémoire.

Ce type de coupe est utilisé dans la littérature pour divers problèmes et pas uniquement pour le VRPSD. Laporte *et al.* (1994) résolvent par exemple le PTSP pour lequel la présence des clients est incertaine et est révélée après qu'une solution a priori soit définie. Quand les présences sont révélées, on suit le parcours défini en sautant simplement les clients absents. Le but est de minimiser l'espérance de la distance parcourue. Le problème est modélisé comme un programme stochastique en nombres entiers et à deux étapes. Une procédure BC

complétée par l'utilisation de LBF pour borner le coût de recours est utilisée pour résoudre le problème. Denton et Gupta (2003) adaptent la méthode L-shaped et utilisent des LBF également pour résoudre un problème de détermination d'heures de rendez-vous dans un contexte pour lequel les durées sont aléatoires, comme les rendez-vous que l'on peut prendre chez son médecin. Louveaux et Schyns (2004) résolvent le problème des m commis voyageurs (m-TSP) dans lequel les temps de trajet et/ou de service sont stochastiques, une pénalité étant encourue chaque fois que les durées des trajets sont trop longues. La méthode L-shaped en nombres entiers est utilisée explicitement et les coupes LBF valides pour des solutions fractionnaires sont utilisées pour borner le coût de recours. On peut aussi utiliser les coupes LBF pour résoudre la variante du VRPSD dans laquelle les produits à livrer ont deux dimensions, hauteur et largeur. C'est ce que font Côté *et al.* (2013), qui utilisent des coupes LBF pour accélérer la méthode L-shaped en nombres entiers.

Ce mémoire a pour but d'améliorer l'état actuel des techniques de résolution du VRPSD tout en se servant de plusieurs avancées déjà effectuées. L'apport principal se situera essentiellement sur le développement de nouvelles coupes LBF. Il est toutefois important de bien situer les contributions ayant déjà fait partie de précédentes recherches.

Un premier apport concerne la façon dont sera calculé le coût espéré d'une route. Dror et Trudeau (1986) sont les premiers à montrer que l'espérance du coût d'une route dépend de son orientation. En effet, étant donné qu'un échec a plus de chance d'arriver en fin de tournée, les distances entre les derniers clients et le dépôt vont influencer le coût total. De même, la concentration de la demande à un endroit n'aura pas le même impact si elle se trouve en début ou en fin de tournée. L'orientation choisie sera donc celle qui minimise les deux valeurs.

La méthode L-shaped en nombres entiers a été appliquée pour la première fois au VRPSD dans Gendreau *et al.* (1995). En plus des demandes, la présence des clients était alors elle aussi aléatoire. Aucune coupe de type LBF n'était utilisée. C'est la première fois que des solutions optimales exactes ont été trouvées pour ce problème. Il est également observé que les problèmes pour lesquels la présence des clients est déterministe sont plus faciles à résoudre.

Ces travaux ont été repris par Hjorring et Holt (1999) qui ont utilisé le même modèle à deux indices avec le recours traditionnel et la méthode L-shaped en nombres entiers pour résoudre. Cette fois, seule la demande des clients est incertaine. De plus, on ne considère que les instances comportant un seul véhicule. Les coupes d'optimalité générales qu'ils introduisent sont les premières coupes LBF utilisées pour le VRPSD. Elles viennent compléter

les coupes d'optimalité spécifiques essentielles pour assurer la convergence de l'algorithme. Elles se différencient cependant de ces dernières en étant actives sur un grand nombre de solutions fractionnaires et entières, contrairement aux coupes d'optimalité spécifiques qui ne sont actives que pour une seule solution entière. Les coupes LBF ainsi ajoutées sont basées sur l'identification des routes partielles dans les solutions fractionnaires intermédiaires. Ce concept de route partielle a également été introduit dans ces travaux : il s'agit en fait de parties de sous-graphes partiels induits par les variables positives et connectées d'une solution fractionnaire. Ces parties qu'on appelle routes partielles sont reliées au dépôt par un ensemble d'arêtes dont le flot est entier, pouvant potentiellement être vide, mais pour lesquelles il existe un ensemble de clients non structurés, dont l'ordre n'a pas encore été établi et qui sont donc liés entre eux par des arêtes dont le flot est fractionnaire.

Laporte *et al.* (2002) ont continué ces travaux en reprenant toujours le même modèle et la même méthode de résolution. La condition selon laquelle l'espérance de la demande totale d'une route ne doit pas dépasser la capacité des véhicules a été ajoutée. Ceci permet une meilleure stabilité des solutions trouvées et permet d'éviter des situations dans lesquelles un véhicule est sous-utilisé tandis qu'un autre échoue quasi systématiquement. La borne générale L active sur l'ensemble des solutions du problème a été améliorée. Le détail du calcul est explicité pour les distributions Normales et de Poisson. De plus, le calcul est valable pour des instances à plusieurs véhicules, ce qui n'était pas le cas dans Hjorring et Holt (1999). L'utilisation des coupes LBF a également été étendue pour le cas de plusieurs véhicules. Enfin, il faut noter qu'à partir de cet article, les coupes d'optimalité présentes et indispensables dans l'algorithme L-shaped en nombres entiers ont changé de forme. La variable θ modélisant une borne inférieure sur le coût de recours n'est plus utilisée. Cette nouvelle forme assigne un coefficient 1 à chaque variable présente dans la coupe, ce qui la rend plus stable d'un point de vue numérique par rapport à la forme employée par Laporte et Louveaux (1993), Gendreau *et al.* (1995) et Hjorring et Holt (1999).

Avec toujours le même modèle et la même méthode de résolution, Jabali *et al.* (2014) ont encore amélioré l'algorithme et proposent les plus récents travaux effectués sur le VRPSD basés sur ce type de méthode de résolution. La principale modification s'est faite au niveau de la forme des coupes LBF ajoutées. En exploitant davantage l'information fournie par les routes partielles, l'article présente trois formes génériques que l'on cherche à identifier pour ajouter les coupes correspondantes. Ces dernières se sont révélées être particulièrement efficaces puisqu'elles ont permis de résoudre des instances comportant 4 véhicules et 60 clients ou 2 véhicules et 80 clients. Elles sont ainsi les plus efficaces de la littérature à ce jour pour

cette méthode de résolution. C'est principalement par rapport à ces coupes que l'on comparera les performances de l'apport de ce mémoire. L'article fournit également un algorithme de séparation exacte permettant d'identifier les coupes. Un point important à observer et qui a été gardé en tête pour les travaux de ce mémoire est la conjecture selon laquelle des coupes actives sur un très grand nombre de solutions fractionnaires et entières, mais qui apportent une borne faible sur le coût de recours semblent plus efficaces que des coupes produisant une meilleure borne, mais actives sur un nombre restreint de solutions.

CHAPITRE 3

PRÉSENTATION DU MODÈLE MATHÉMATIQUE

Dans ce chapitre, nous allons dans un premier temps présenter le formalisme habituel utilisé pour la programmation stochastique. Ceci permettra de comprendre la logique de la formulation de notre modèle qui sera introduit dans une seconde partie.

3.1 Formalisme de la programmation stochastique

La programmation stochastique a été introduite par Dantzig (1955) et Beale (1955). Elle permet d'appréhender l'incertitude présente dans certains problèmes. Il existe deux manières de prendre en compte l'incertitude dans un modèle de programmation stochastique :

La première est la modélisation avec contraintes probabilistes. La philosophie de cette approche est d'assurer qu'un ou plusieurs évènements ne se produisent qu'à une certaine probabilité, fixée généralement à une petite valeur. Pour notre VRPSD par exemple, cela se traduirait par dire qu'une situation d'échec ne peut se produire qu'avec une probabilité de $0,05$. Pour assurer ceci, des contraintes dites probabilistes sont ajoutées au modèle. Ce concept permet donc de quantifier les risques relatifs à une décision et comme on l'a décrit dans le chapitre précédent, plusieurs travaux concernant le VRPSD avec contraintes probabilistes ont été réalisés.

La deuxième manière permet d'obtenir une solution optimale pour un problème dont l'objectif est la maximisation ou minimisation d'une espérance mathématique. Elle maximise ou minimise ainsi cet objectif pour l'ensemble des scénarios possibles de la situation. Cette approche se prête bien au VRPSD car pour des tournées quotidiennes, des valeurs différentes seront prises chaque jour par les demandes des clients et il faudra minimiser la moyenne de la distance parcourue prise sur l'ensemble de ces journées. Elle permet également de gérer l'incertitude et de prévoir des décisions de recours en cas de situation défavorable. Dans un cadre linéaire, on appelle ces programmes des programmes linéaires stochastiques avec recours. Cette approche suppose un processus de décision séquentiel. Des premières décisions sont prises, arrive ensuite la révélation de certaines variables aléatoires, des décisions correctives sont prises à leur tour puis d'autres variables aléatoires sont révélées, etc.

C'est cette approche que nous choisirons pour modéliser le VRPSD. Pour ce dernier, seules deux étapes suffisent. La première consiste à confectionner des routes a priori qui minimiseront la distance parcourue en moyenne, puis, à mesure que les demandes sont révélées en visitant chaque client, une décision de recours peut avoir lieu et consiste en un aller-retour entre le dépôt et le client qui n'a pas pu être servi entièrement.

Pour un cas comme celui-ci, on distingue ainsi les variables de première étape de celles de seconde étape. On note $x \in \mathbb{R}^{n_1}$ le vecteur des décisions de première étape, Ω l'ensemble des évènements aléatoires pouvant avoir lieu entre les deux étapes, $\omega \in \Omega$ un évènement aléatoire qui se produit entre les deux étapes, $\xi(\omega)$ le vecteur aléatoire représentant les valeurs prises par les paramètres incertains du problème, et enfin $y(x, \omega) \in \mathbb{R}^{n_2}$ le vecteur des décisions de deuxième étape. Pour schématiser, on a la séquence suivante :

$$x \rightarrow \xi(\omega) \rightarrow y(x, \omega) \tag{3.1}$$

La formulation générale du problème est la suivante :

$$\min_{x \in \mathbb{R}^{n1}} c^T x + \mathbb{E}_\xi[Q(x, \xi(\omega))] \tag{3.2}$$

sujet à :

$$Ax = b \tag{3.3}$$

$$x \geq 0 \tag{3.4}$$

avec $A \in \mathbb{R}^{m_1 \times n_1}$, la matrice des contraintes de première étape, $c \in \mathbb{R}^{n_1}$ le vecteur de coût de première étape et $b \in \mathbb{R}^{m_1}$. Comme discuté plus haut, l'objectif est composé d'une partie déterministe et d'une partie stochastique qui est l'espérance prise sur ξ de $Q(x, \xi(\omega))$, le coût de recours faisant suite à la décision de première étape x et à l'évènement aléatoire ω. Pour se ramener à une formulation en programme déterministe équivalent, et sachant que le second terme de l'objectif est une espérance prise sur l'ensemble des valeurs possibles de ξ et que donc il ne dépend que de x, on agrège ce second terme en une fonction Q définit comme suit :

$$Q(x) = \mathbb{E}_\xi[Q(x, \xi(\omega))] \tag{3.5}$$

avec

$$Q(x, \xi(\omega)) = \min_{y \in \mathbb{R}^{n_2}} \{q^T(\omega) y \mid Wy = h(\omega) - T(\omega)x,\ y \geq 0\} \tag{3.6}$$

Le vecteur de coût $q \in \mathbb{R}^{n_2}$, le vecteur $h \in \mathbb{R}^{m_2}$ et la matrice $T \in \mathbb{R}^{m_2 \times n_1}$ sont des éléments aléatoires et dépendent de ω. En revanche, la matrice $W \in \mathbb{R}^{m_2 \times n_2}$ est fixe car, on se situe dans un cadre de programmation avec recours fixe.

Le calcul de la valeur de $Q(x, \xi(\omega))$ revient en fait à résoudre le problème de la seconde étape. Pour une décision x et un évènement aléatoire ω, on cherche la valeur de y qui ajuste au mieux la solution établie en première étape. Pour le VRPSD, on dit que le recours est complet, ce qui signifie que, quelle que soit la valeur prise par le vecteur x dans la première étape, le problème de deuxième étape sera toujours réalisable.

Le choix qui a été fait de modéliser notre problème comme un modèle stochastique présente l'avantage de fournir des solutions dont la valeur de l'objectif sera relativement basse. Cette valeur représente le coût moyen subi chaque jours pour lesquels des tournées ont lieu. Comme cela est présenté par Birge et Louveaux (2011), cette valeur se nomme RP (Recourse Problem). Une autre façon de prendre en compte l'aspect incertain du problème est de fixer la valeur des demandes des clients à leurs espérances. On peut donc résoudre un problème de type CVRP déterministe dont la valeur optimale de l'objectif sera notée EV (Expected Value solution). En injectant cette solution dans le modèle stochastique, on obtient une nouvelle valeur pour l'objectif stochastique qui est dénotée EEV (Expected result of using Expected Value). La différence entre EEV et RP représente le gain obtenu en modélisant le problème grâce à la programmation stochastique. Ce gain s'appelle VSS (Value of the Stochastic Solution) et on a :

$$VSS = EEV - RP \geq 0 \tag{3.7}$$

Ceci est dû au fait que EEV est la valeur de l'objectif d'un problème pour lequel on a choisi une solution réalisable alors que RP est la valeur optimale de ce même problème.

On peut noter cependant que les modèles stochastiques à deux étapes sont plus difficiles à résoudre que les modèles déterministes. Pour notre problème, la difficulté est accentuée par le fait que les variables de première étape devront prendre des valeurs entières. En revanche, nous ne nous situerons pas dans le cas le plus difficile à résoudre pour lequel les variables de seconde étape doivent être entières avec une fonction objectif non convexe.

3.2 Formulation du VRPSD comme un programme linéaire stochastique avec recours

Maintenant que l'on a introduit la formulation générale d'un programme stochastique avec recours, appliquons à présent ce principe à notre problème.

On définit ainsi le graphe complet non orienté $G = (V, E)$, avec $V = (v_1, \ldots, v_n)$ l'ensemble des nœuds et $E = \{(v_i, v_j) | v_i, v_j \in V^2, i < j\}$ l'ensemble des arêtes. Le nœud v_1 représente le dépôt à partir duquel m véhicules de capacité D vont commencer et terminer leur tournée. Les autres nœuds v_i tels que $i > 1$ représentent les clients à desservir. Leur demande est représentée par la variable aléatoire ξ_i qui suit une loi normale d'espérance μ_i et de variance σ_i^2 tronquée en 0 et en D. On suppose également que les variables ξ_i sont indépendantes. Les arêtes $(v_i, v_j) \in E$ représentent un chemin possible entre deux clients ayant un coût c_{ij}. On suppose que les coûts sont symétriques et, pour deux clients i et j tels que $i < j$, on définit seulement c_{ij}, i.e., que le parcours de l'arête du client j vers le client i aura le même coût et on ne définira pas de coût c_{ji}. On définit les variables entières à deux indices x_{ij} indiquant le nombre de passages sur l'arête (v_i, v_j). Si $i > 0$, c'est-à-dire si l'arête concerne deux clients, alors x_{ij} peut prendre la valeur 0 si elle n'est empruntée par aucun véhicule, ou 1 dans le cas contraire. Les variables x_{1j} peuvent prendre la valeur de 2 dans le cas où un véhicule effectue un aller-retour entre le dépôt et le client j. Enfin, on note $Q(x)$ l'espérance du coût de recours associé aux routes définies par le vecteur x.

Dans notre problème, on peut associer une variable y_i binaire de seconde étape à chaque client. Celle-ci prendra la valeur de 1 si un aller-retour doit être fait entre lui-même et le dépôt et 0 dans le cas contraire. Chaque fois que le véhicule n'a plus assez de stock pour servir la demande du client, cet aller-retour sera nécessaire pour se recharger et revenir avec un stock de quantité D. En fait, le problème d'optimisation de seconde étape sera trivial : connaissant la solution x et la valeur que prennent les demandes, il sera facile de déterminer la valeur des variables y_i. C'est pourquoi, par la suite, on n'explicitera pas le problème d'optimisation de seconde étape avec son objectif et ses contraintes. Cela est généralement utile pour déterminer l'expression littérale de $Q(x)$, mais, comme on le verra par la suite, on peut déterminer cette expression sans se baser sur la formulation du problème de seconde étape.

Le modèle est le suivant :

$$\min_x \sum_{i<j} c_{ij} x_{ij} + Q(x) \tag{3.8}$$

sujet à :

$$\sum_{j=2}^{n} x_{1j} = 2m \tag{3.9}$$

$$\sum_{i<k} x_{ik} + \sum_{k<j} x_{kj} = 2, \qquad (k = 2, \ldots, n) \tag{3.10}$$

$$\sum_{v_i, v_j \in S^2} x_{ij} \leq |S| - \left\lceil \frac{\sum_{v_i \in S} \mu_i}{D} \right\rceil, \qquad (S \subset V \setminus \{v_1\},\ 2 \leq |S| \leq n-2) \tag{3.11}$$

$$0 \leq x_{ij} \leq 1, \qquad (2 \leq i \leq j \leq n) \tag{3.12}$$

$$0 \leq x_{1j} \leq 2, \qquad (2 \leq j \leq n) \tag{3.13}$$

$$x_{ij} \in \mathbb{N}, \qquad (1 \leq i \leq j \leq n) \tag{3.14}$$

La contrainte (3.9) impose le degré du nœud représentant le dépôt. Cela oblige les m véhicules à commencer puis finir leur tournée au dépôt. Les contraintes (3.10) fixent les degrés des nœuds représentant les clients. Celui-ci est égal à 2 ce qui assure la conservation du flot de véhicules à chaque client. Les contraintes (3.11) sont nécessaires pour empêcher l'apparition de sous-tours dans la solution. Elles assurent également que sur chaque route, l'espérance de la demande cumulée ne dépasse pas la capacité du véhicule. Enfin, les contraintes (3.12) à (3.14) concernent l'intégrité et les bornes des variables x_{ij}.

$Q(x)$ ne peut malheureusement pas se décomposer en combinaison linéaire des variables x_{ij}, ce qui rendrait le problème facile à résoudre. En revanche, étant donné une solution réalisable x, il est possible de calculer facilement la valeur de Q. Pour cela, on utilise le fait que l'espérance du coût de recours est décomposable pour chacune des m routes isolées. Soit donc la route k, avec $k \in [\![1; m]\!]$, composée de la suite de clients suivante : $(v_{i_1} = v_1, v_{i_2}, \ldots, v_{i_t}, v_{i_{t+1}} = v_1)$. Pour chaque client, il peut y avoir au maximum un seul recours prenant la forme d'un aller-retour entre celui-ci et le dépôt. Cet aller-retour peut avoir lieu dans plusieurs scénarios possibles. Soit c'est le premier à apparaitre sur la route et tous les clients précédents ont pu être servis sans avoir besoin de recharger, soit il y a déjà eu des recours avant d'arriver au client courant, sachant que le nombre de recours précédents est borné par le nombre de clients précédents puisqu'il y a au maximum un recours par client. Ainsi, si l'aller-retour ayant lieu chez un client est le l^e depuis le début de la route, on appellera ce dernier le l^e échec. De plus, le coût de recours pour une route isolée dépendra de son orientation $\delta \in [\![1; 2]\!]$. En effet, si pour une orientation la demande est concentrée vers le début du parcours sans pour autant dépasser la capacité, alors les allers-retours auront tendance à se faire juste après ses clients et donc pourront être plus éloignés que pour les

premiers clients. En revanche, dans l'autre orientation de ce cas de figure fictif, la demande sera concentrée vers la fin du parcours et le ou les recours se feront à des clients situés à une petite distance du dépôt. Il existe bien sûr une multitude de cas de figure, l'essentiel est de prendre en compte le fait que l'espérance du coût de recours dépendra de l'orientation de la route. D'un point de vue opérationnel, il faudra choisir l'orientation qui minimise les deux valeurs. En notant $Q^{k,\delta}$ l'espérance du coût de recours pour la route k dans l'orientation δ, on a donc :

$$Q(x) = \sum_{k=1}^{m} \min\{Q^{k,1}, Q^{k,2}\} \tag{3.15}$$

où

$$Q^{k,1} = \sum_{j=2}^{t} 2c_{1i_j} \sum_{l=1}^{j-1} P\left(\sum_{s=1}^{j-1} \xi_{i_s} \leq lD \leq \sum_{s=1}^{j} \xi_{i_s}\right) \tag{3.16}$$

pour chaque route k définie par la suite de clients $(v_{i_1} = v_1, v_{i_2}, \ldots, v_{i_t}, v_{i_{t+1}} = v_1)$.

Le calcul de $Q^{k,2}$ se fait de manière similaire en renversant la suite de clients. On observe donc bien que la somme est décomposable une première fois en énumérant les recours possibles à chaque client, puis pour chacun d'entre eux en analysant quels sont les différents scénarios qui peuvent amener un tel recours, c'est-à-dire l'ensemble des l^e échecs possibles.

Pour le calcul de $P(\sum_{s=1}^{j-1} \xi_{i_s} \leq lD \leq \sum_{s=1}^{j} \xi_{i_s})$, on pose trois évènements. Notons A l'évènement $\sum_{s=1}^{j-1} \xi_{i_s} \leq lD$, B l'évènement $\sum_{s=1}^{j-1} \xi_{i_s} \leq lD \leq \sum_{s=1}^{j} \xi_{i_s}$ et enfin C l'évènement $\sum_{s=1}^{j-1} \xi_{i_s} \leq \sum_{s=1}^{j} \xi_{i_s} \leq lD$. On a donc $A = B \cup C$ avec $B \cap C = \varnothing$. De plus, sachant que les variables aléatoires ξ_{i_s} prennent des valeurs positives, alors l'évènement C peut se simplifier en $\sum_{s=1}^{j} \xi_{i_s} \leq lD$. Ainsi, $P(A) = P(B \cup C) = P(B) + P(C) \Rightarrow P(B) = P(A) - P(C)$.

$$\Rightarrow P\left(\sum_{s=1}^{j-1} \xi_{i_s} \leq lD \leq \sum_{s=1}^{j} \xi_{i_s}\right) = P\left(\sum_{s=1}^{j-1} \xi_{i_s} \leq lD\right) - P\left(\sum_{s=1}^{j} \xi_{i_s} \leq lD\right) \tag{3.17}$$

Les variables ξ_{i_s} suivent une loi normale telle que $\xi_{i_s} \sim \mathcal{N}(\mu_{i_s}, \sigma_{i_s}^2)$. On a donc $\sum_{s=1}^{j-1} \xi_{i_s} \sim \mathcal{N}(\mu_{j-1}, \sigma_{j-1}^2)$ avec $\mu_{j-1} = \sum_{s=1}^{j-1} \mu_{i_s}$ et $\sigma_{j-1}^2 = \sum_{s=1}^{j-1} \sigma_{i_s}^2$. On définit de même μ_j et σ_j tels que $\sum_{s=1}^{j} \xi_{i_s} \sim \mathcal{N}(\mu_j, \sigma_j^2)$. En notant $z_{j-1} = \frac{\sum_{s=1}^{j-1} \xi_{i_s} - \mu_{j-1}}{\sigma_{j-1}}$ et $z_j = \frac{\sum_{s=1}^{j} \xi_{i_s} - \mu_j}{\sigma_j}$, on a alors :

$$P\left(\sum_{s=1}^{j-1} \xi_{i_s} \leq lD \leq \sum_{s=1}^{j} \xi_{i_s}\right) = P\left(z_{j-1} \leq \frac{lD - \mu_{j-1}}{\sigma_{j-1}}\right) - P\left(z_j \leq \frac{lD - \mu_j}{\sigma_j}\right) \tag{3.18}$$

On peut donc facilement calculer la probabilité du l^e échec au client v_{i_j} à l'aide de tables de valeurs de la loi normale.

Néanmoins, il faut noter que ce raisonnement s'applique pour des lois de probabilité non tronquées. Or, on a précisé que les distributions utilisées pour les demandes des clients l'étaient en 0 et en D. En fait, comme nous le verrons dans le chapitre 5 dans lequel nous présenterons les résultats numériques de nos expériences, nos instances seront telles que l'espérance des demandes individuelles sera nettement en dessous de la capacité D. Cela se traduit par un coefficient de remplissage des véhicules $\bar{f}$ jamais supérieur à 1. On détaillera son principe dans le chapitre 5. Ce choix a été fait pour éviter de rendre le problème irréalisable. Ainsi, comme les demandes ξ_i sont telles que $P(\xi_i \geq D) \simeq 0$, les calculs de probabilité présentés ci-dessus sont justes en première approximation.

CHAPITRE 4

MÉTHODE DE RÉSOLUTION

Dans ce chapitre, nous présenterons dans un premier temps la relaxation du modèle. Celle-ci est nécessaire pour faciliter la résolution. Puis, nous décrirons comment le modèle relaxé sera modifié pour assurer la faisabilité des solutions trouvées par rapport au modèle initial. La méthode utilisée pour cela est l'algorithme L-shaped en nombres entiers. La méthode de calcul de la borne générale utilisée sera ensuite décrite. La dernière section présentera enfin les trois types de coupes LBF utilisés pour accélérer la résolution.

4.1 Relaxation du modèle

Comme on l'a vu, le terme $Q(x)$ de l'objectif ne peut pas être décomposé en combinaison linéaire des variables x_{ij}. C'est pourquoi une étape préliminaire à la résolution du problème est de le rendre résoluble par un algorithme classique de la programmation linéaire comme l'algorithme du simplexe. Celui-ci n'acceptant que des termes linéaires, on va remplacer le terme $Q(x)$ par une variable θ réelle positive qui jouera le rôle d'une borne inférieure sur le coût de recours pour une solution entière réalisable x. Laisser la borne inférieure $\theta \geq 0$ est correct d'un point de vue mathématique, mais comme l'ont montré les travaux de Laporte *et al.* (2002), il est intéressant pour améliorer la résolution du problème d'avoir une borne inférieure de qualité sur un maximum de solutions réalisables. Ils ont donc proposé une borne inférieure générale $L > 0$ qui sera active pour l'ensemble des solutions réalisables. On discutera du calcul de cette borne dans la section 4.3.

De plus, les contraintes d'élimination de sous-tours et de capacité (3.11) sont très nombreuses, car il existe un nombre important de sous-ensembles de clients S. C'est pourquoi elles seront en premier lieu supprimées, puis ajoutées dynamiquement à l'aide d'un algorithme de séparation permettant d'identifier les contraintes violées par la solution courante. Enfin, comme il est courant pour les programmes linéaires en nombres entiers, les contraintes d'intégrité (3.14) seront elles aussi relaxées du modèle. On obtient donc le problème initial suivant :

$$\min_{x} \sum_{i<j} c_{ij} x_{ij} + \theta \tag{4.1}$$

sujet à :

$$\theta \geq L \tag{4.2}$$

$$\sum_{j=2}^{n} x_{1j} = 2m \tag{4.3}$$

$$\sum_{i<k} x_{ik} + \sum_{k<j} x_{kj} = 2, \qquad (k = 2, \ldots, n) \tag{4.4}$$

$$0 \leq x_{ij} \leq 1 \qquad (2 \leq i \leq j \leq n) \tag{4.5}$$

$$0 \leq x_{1j} \leq 2 \qquad (2 \leq j \leq n) \tag{4.6}$$

4.2 Algorithme L-shaped en nombres entiers

L'algorithme L-shaped en nombres entiers part de la relaxation du problème évoquée dans la section 4.1 puis atteint l'optimalité pour le modèle complet en suivant une méthode BC. L'algorithme est décrit par les étapes suivantes, dans lesquelles on fait référence au PC signifiant Problème Courant :

Étape 0 : Calculer L et mettre le compteur d'itérations ν à 0. Définir le PC comme la relaxation du VRPSD complet dans lequel les contraintes (3.11) et (3.14) sont supprimées, le terme $Q(x)$ de l'objectif est remplacé par la variable θ qui est soumise à la contrainte $\theta \geq L$. La meilleure solution connue est $\bar{z} := \infty$. À ce stade, le seul nœud actif (i.e., non évalué) est le PC initial.

Étape 1 : Sélectionner un nœud parmi la liste des nœuds actifs. S'il n'y en a pas, arrêter.

Étape 2 : Itérer le compteur $\nu := \nu+1$ et résoudre le PC. Soit (x^ν, θ^ν) sa solution optimale.

Étape 3 : Rechercher les contraintes d'élimination de sous-tours violées par la solution courante et les générer en conséquence. Des coupes LBF violées peuvent également être générées. Si une contrainte violée est trouvée, l'ajouter au PC et retourner à l'étape 2. Sinon, si $c^T x^\nu + \theta^\nu \geq \bar{z}$, élaguer le nœud courant et retourner à l'étape 1.

Étape 4 : Si la solution n'est pas entière, alors brancher sur une variable fractionnaire. Ajouter les sous-problèmes à la liste des nœuds actifs et retourner à l'étape 1.

Étape 5 : Calculer $Q(x^\nu)$ et fixer $z^\nu := c^T x^\nu + Q(x^\nu)$. Si $z^\nu < \bar{z}$, alors fixer $\bar{z} := z^\nu$.

Étape 6 : Si $\theta^\nu \geq Q(x^\nu)$, alors élaguer le nœud courant et retourner à l'étape 1. Sinon, ajouter la coupe d'optimalité définie par :

$$\sum_{1<i<j \mid x_{ij}^\nu=1} x_{ij} \leq \sum_{1<i<j} x_{ij}^\nu - 1 \tag{4.7}$$

et retourner à l'étape 2.

À l'étape 3, l'ajout de coupes violées de type LBF ou de sous-tours permet d'augmenter la valeur de la borne inférieure au nœud courant. Plus cette augmentation se fera efficacement et rapidement, plus le problème se résoudra rapidement. C'est pourquoi le concept des coupes LBFs que nous détaillerons dans la section 4.4 est déterminant dans l'accélération de l'algorithme. Le fait d'avoir une solution dont l'objectif $cx^\nu + \theta^\nu$ est élevé rapidement va alors permettre la validation de la condition $c^T x^\nu + \theta^\nu \geq \bar{z}$ pour un grand nombre de nœuds ce qui réduira la taille de l'arbre à explorer. Le fait d'élaguer le nœud correspondant quand cette condition est remplie vient du fait que l'objectif est calculé pour un problème relaxé et est donc censé être une valeur inférieure ou égale au coût optimal réel. Or, si dans cette partie de l'arbre d'énumération, cette valeur est supérieure à la borne supérieure courante, alors il sera inutile d'explorer plus profondément le nœud sachant qu'on ne trouvera jamais un objectif inférieur à la borne supérieure, ce qui améliorerait la solution courante.

À l'étape 4, les règles de branchement peuvent être diverses. Celle qu'on emploie principalement est la règle qui consiste à brancher sur l'arête fractionnaire la plus proche du dépôt. En effet, les échecs ont tendance à se produire à la fin d'une tournée quand le véhicule s'approche du dépôt. Ainsi, la borne inférieure du sous-problème dans lequel l'arête sera empruntée aura tendance à augmenter ce qui rend cette règle de branchement efficace.

L'étape 5 consiste à mettre à jour la meilleure borne supérieure. Elle compare l'objectif de l'itération ν avec la borne supérieure actuelle $\bar{z}$. Il est important de noter que contrairement aux méthodes BC classiques, l'algorithme L-shaped en nombres entiers ne compare pas l'objectif du PC $c^T x^\nu + \theta^\nu$ avec la borne $\bar{z}$ car celui-ci n'est qu'une approximation inférieure du coût réel espéré de la solution x^ν. Pour remédier à cela, il faut calculer $Q(x^\nu)$ et remplacer θ^ν dans l'objectif.

Enfin, à l'étape 6, on ajoute une coupe d'optimalité (4.7) pour être sûr de trouver la meilleure solution entière de ce nœud de l'arbre. En effet, on dispose pour l'instant de la solution entière x^ν qui minimise la valeur de $c^T x^\nu + \theta^\nu$ en respectant toutes les contraintes

du problème, dont celles d'intégrité. Or, si $\theta^{\nu} < Q(x^{\nu})$, le coût de recours espéré a été « mal » approximé. Il existe donc peut-être dans cette partie de l'arbre de branchement une autre solution $x^{\nu'}$ entière qui a une valeur $c^T x^{\nu'} + \theta^{\nu'}$ plus élevée mais pour laquelle $c^T x^{\nu'} + Q(x^{\nu'}) < c^T x^{\nu} + Q(x^{\nu})$. Il est donc nécessaire d'imposer la coupe d'optimalité pour chercher les autres solutions en résolvant à nouveau le problème tout en empêchant d'obtenir la même solution.

On peut noter que la forme que prennent les coupes d'optimalité n'a pas toujours été celle qu'on utilise ici. Pour une solution entière x^{ν} qui a un coût de recours espéré $Q(x^{\nu})$, Gendreau *et al.* (1995) et Hjorring et Holt (1999) ajoutaient une coupe d'optimalité de la forme :

$$\Big(Q(x^{\nu}) - L\Big)\Bigg(\sum_{1<i<j \mid x^{\nu}_{ij}=1} x_{ij} - \Big(\sum_{1<i<j} x^{\nu}_{ij} - 1\Big)\Bigg) + L \leq \theta \tag{4.8}$$

Cette forme a été proposée pour la première fois par Laporte et Louveaux (1993). Ce n'est qu'à partir des travaux de Laporte *et al.* (2002) que la forme (4.7) présentée à l'étape 6 de notre algorithme a été choisie. Ceci se justifie alors par leur meilleure stabilité numérique, car tous les coefficients de la coupe sont égaux à 1.

On peut enfin remarquer qu'il est possible de remplacer le "1" de la partie droite des coupes d'optimalité (4.7) par "2", chose que l'on fera dans l'implémentation informatique de l'algorithme. En effet, toutes les solutions entières différentes de la solution x^{ν} auront au minimum deux arêtes de différentes. Pour comprendre cela, considérons le graphe représentant les m routes d'une solution entière. Le degré de chaque nœud du graphe étant fixé, lorsqu'on enlève une arête, les degrés des deux nœuds concernés sont insuffisants et tous les autres nœuds ont leur degré saturé : il faut donc au minimum retirer une deuxième arête pour que les contraintes sur les routes soient respectées.

4.3 Calcul de la borne inférieure générale

Les travaux de Laporte *et al.* (2002) ont montré qu'il était possible de calculer une borne efficace pour le VRPSD avec plusieurs véhicules, améliorant ainsi le premier développement d'une borne L générale de Hjorring et Holt (1999) dont la méthode permettait uniquement de traiter le cas d'un seul véhicule.

La méthode consiste dans un premier temps à renommer les clients par ordre croissant de leur distance au dépôt, le client v_2 étant le plus près puis le client v_3 etc. Pour le calcul,

on suppose que l'on connaît deux choses : Le nombre de routes à planifier ainsi que les clients à desservir et surtout la distribution de leur demande totale cumulée d'espérance μ_T et de variance σ_T. Ces paramètres sont dans tous les cas connus avant la résolution du problème. L'idée est d'évaluer une borne inférieure sur le coût qu'entraînerait un unique échec sur chacune des m routes, supposant en plus que ce dernier se fait au niveau de l'un des m clients les plus proches. Pour cela, la méthode consiste à partir de la demande totale et de sa distribution, puis de la diviser en la répartissant parmi les m routes de telle sorte que cette borne soit minimale tout en respectant des contraintes de faisabilité : en notant, respectivement, μ_k et σ_k^2 l'espérance et la variance de la demande totale de la route k, on a :

$$L = \min_{\mu,\sigma} 2 \sum_{k=1}^{m} c_{1(k+1)} \left(1 - F_{\mathcal{N}}\Big(\frac{D - \mu_k}{\sigma_k}\Big) \right) \tag{4.9}$$

sujet à :

$$\sum_{k=1}^{m} \mu_k = \mu_T \tag{4.10}$$

$$\sum_{k=1}^{m} \sigma_k^2 = \sigma_T^2 \tag{4.11}$$

$$0 \leq \mu_k \leq D \qquad (k = 1, \ldots, m) \tag{4.12}$$

$F_{\mathcal{N}}$ est la fonction de répartition de la loi normale centrée réduite. La valeur de $1 - F_{\mathcal{N}}\Big(\frac{D-\mu_k}{\sigma_k}\Big)$ représente donc la probabilité qu'il y ait un échec sur la route k. La contrainte (4.12) vient du fait que l'on cherche à construire des routes dont l'espérance de la demande cumulée ne dépasse pas la capacité des véhicules.

L est une borne inférieure pour plusieurs raisons. Premièrement, on ne considère que la probabilité d'observer un unique échec sur chacune des routes. Or, il existe une probabilité non nulle que pour certaines routes, on observe plus d'un échec. Deuxièmement, on fait l'hypothèse que ces échecs se font au niveau des m clients les plus proches du dépôt ce qui est le cas extrême qui minimise l'impact du recours. Enfin, la division de la demande totale se fait de manière continue dans le sens où on découpe arbitrairement l'espérance μ_T sans se soucier de savoir si les espérances μ_k ainsi déterminées correspondent effectivement à une somme exacte des espérances de certains clients. La résolution exacte du problème (4.9)-(4.12) utilisant les conditions de Karush-Kuhn-Tucker est décrite dans Laporte *et al.* (2002) et c'est celle que nous utiliserons.

4.4 Coupes LBF

Dans cette section, nous décrirons le principe général sur lequel reposent les coupes LBF. Nous donnerons ensuite les trois formes de coupes efficaces qui ont été développées : les coupes LBF basées sur des routes partielles, introduites par Jabali *et al.* (2014), celles basées sur des chaînes, puis celles basées sur des ensembles non structurés de noeuds. Les deux dernières formes constituent l'apport principal de ce mémoire.

4.4.1 Principe général

On a vu qu'il était important à l'étape 3 d'avoir une valeur de l'objectif élevée pour permettre à l'algorithme d'éliminer rapidement des nœuds de l'arbre et ainsi réduire les temps de calcul. Les contraintes ajoutées jouent ici un rôle essentiel pour renforcer la relaxation. Les contraintes d'éliminations de sous-tours sont en cela utiles, car elles réduisent l'espace des solutions et ont donc tendance à entraîner une augmentation de l'objectif.

La convergence vers l'optimalité est assurée par les coupes d'optimalité (4.7) ajoutées à l'étape 6. Même si l'objectif à minimiser de la relaxation n'est qu'une approximation du coût espéré réel, il est quand même possible grâce à ces coupes de déterminer la solution optimale. Par contre, le fait d'avoir systématiquement recours à ces dernières, parce que le coût de recours espéré θ a été mal approximé, peut entraîner un nombre d'itérations élevé avec un modèle relaxé qui contient de plus en plus de contraintes et qui est donc de plus en plus difficile à résoudre. Pour cela, il est intéressant de disposer de coupes qui, pour une solution entière x, donnent une valeur de θ la plus proche possible du coût de recours réel $Q(x)$.

Des coupes jouant le rôle de coupes d'optimalité ont été introduites par Gendreau *et al.* (1995). Elles ont, comme c'est le cas pour les coupes d'optimalité en général, l'inconvénient de ne toucher qu'une seule solution entière bien qu'elles fournissent une valeur exacte pour le coût de recours. Pour bien comprendre cet inconvénient, observons la situation suivante : soit z^* la valeur optimale du problème. On suppose que l'on utilise que des coupes d'optimalité pour pallier au fait que θ n'est qu'une borne inférieure sur $Q(x)$. Dans ce cas, il faudrait en générer une pour chaque solution entière x pour laquelle $c^T x + L < z^*$, étant donné que θ prendrait la valeur L tant qu'une coupe d'optimalité n'a pas été générée. L'algorithme L-shaped en nombres entiers doit, dans cette situation, visiter chacune de ces solutions réalisables x. Ainsi, si aucune coupe n'est générée pour fournir une meilleure approximation sur $Q(x)$ que L, le modèle relaxé s'encombre rapidement de coupes d'optimalité et le nombre d'itérations de l'algorithme nécessaire à la résolution augmente.

Il est donc essentiel, pour accélérer l'algorithme, de générer des coupes qui seront actives sur plusieurs solutions et qui fourniront une approximation acceptable de θ. Ces coupes sont dites de type LBF et sont définies dans Birge et Louveaux (2011).

Le principe de base a été introduit par Hjorring et Holt (1999). Dans leurs travaux, on fait la distinction entre une coupe d'optimalité spécifique, qui fournit une approximation exacte de $Q(x)$ pour une seule solution entière x, et une coupe d'optimalité générale, qui fournit une approximation de $Q(x)$ de qualité, mais pas forcément exacte, pour un ensemble de solutions réalisables. Cela se traduit mathématiquement par la définition d'un sous-ensemble de paires d'indices $[i, j]$ qu'on note S_p, avec p l'indice d'un de ces sous-ensembles parmi tout ceux qui existent. Il est en effet possible de donner un tel indice à chacun d'eux, car ils existent en nombre fini. On suppose qu'on dispose d'une borne inférieure θ_p sur $Q(x)$, valide si $x_{ij} = 1 \; \forall [i, j] \in S_p$ et x_{ij} quelconque pour $[i, j] \notin S_p$, en respectant bien sûr les contraintes du problème. Il est alors correct d'ajouter l'inégalité valide suivante au modèle :

$$(\theta_p - L)(\sum_{[i,j] \in S_p} x_{ij} - (|S_p| - 1)) + L \leq \theta \tag{4.13}$$

La partie de gauche est égale à θ_p uniquement si l'ensemble des variables x_{ij} telles que $[i, j] \in S_p$ sont égales à 1. Sinon, si au moins l'une d'elles vaut 0, alors la partie de gauche est inférieure ou égale à L. La coupe est donc valide.

La difficulté consiste à déterminer judicieusement des sous-ensembles S_p qui rendront efficaces les coupes du type (4.13). Jusqu'ici, les coupes LBF sont basées sur l'identification dans la solution de routes partielles. Ce concept a été introduit par Hjorring et Holt (1999) et était utilisé alors pour traiter le VRPSD avec un seul véhicule. Une route partielle, de façon générale, peut se définir comme un ensemble de clients affectés à une route. Pour une partie de ces clients, on connaît l'ordre dans lequel ils seront visités, et pour d'autres non. Plus formellement, Hjorring et Holt (1999) définissent un type de route partielle comme étant la séquence de clients $(v_1, v_{i_1}, \ldots, v_{i_s}, \ldots, v_{i_e}, \ldots, v_{i_n}, v_1)$. Les premiers clients de la route et les derniers sont donc ordonnés et suivent un séquencement connu, tandis que les clients $(v_{i_{s+1}}, \ldots, v_{i_{e-1}})$ sont désordonnés. Les routes partielles sont une première proposition de sous-ensemble S_p, et la définition de Hjorring et Holt (1999) est elle-même une première proposition de routes partielles. Ce concept a été généralisé ensuite par Jabali *et al.* (2014) et a permis d'implémenter des coupes LBF efficaces dont on décrit le principe dans la partie suivante.

4.4.2 Coupes LBF basées sur les routes partielles

Les coupes générées dans les travaux de Jabali *et al.* (2014) sont les plus efficaces de la littérature à ce jour, et permettent de résoudre dans des temps raisonnables des instances du VRPSD comprenant jusqu'à 4 véhicules pour 60 clients ou 2 véhicules pour 80 clients. C'est principalement par rapport à ces coupes que l'on comparera les performances de l'apport de ce mémoire.

Le principe est d'exploiter au maximum l'information apportée par les routes partielles rencontrées dans les solutions intermédiaires. Ainsi, plutôt que de considérer un unique ensemble de clients non ordonnés liant un début et une fin de route ordonnée, on identifie cette fois l'ensemble des éléments ordonnés et non ordonnés qui sont reliés par des nœuds d'articulation pour former une route partielle complète. Les éléments ordonnés sont des chaînes d'arêtes, les éléments non ordonnés sont aussi appelés des composants non structurés.

Pour avoir une idée visuelle du type de routes partielles recherchées dans les travaux de Jabali *et al.* (2014), on peut observer l'exemple de la figure 4.1. Les nœuds colorés en gris foncé représentent les nœuds d'articulation.

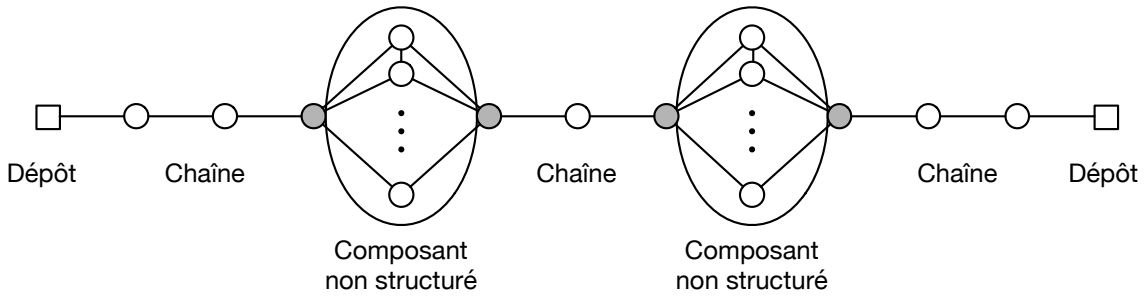

Figure 4.1 Exemple de route partielle

Si f routes partielles sont identifiées dans une solution intermédiaire, avec $1 \leq f \leq m$, on calcule pour chacune d'elles une borne inférieure P_h sur le coût de recours moyen qu'une solution entière doit subir si une de ses m route emprunte les mêmes clients en respectant le même ordre pour les clients des différentes chaînes présentes dans la route partielle h. Pour cela, on forme une route fictive dans laquelle les composants non structurés sont remplacés par un unique client dont la demande est la somme des demandes des clients du composant. Sa distance avec le dépôt correspond à la distance minimale entre les clients du composant non structuré et le dépôt. Les autres clients, qui appartiennent à des chaînes, sont laissés tels

quels. Le calcul est ensuite effectué comme pour celui d'une route classique en respectant les formules (3.15) et (3.16). Si $f < m$, on calcule en plus un terme P_{f+1} basé sur le même principe que pour le calcul de la borne générale du modèle L. Celui-ci fournit une borne inférieure sur le coût de recours moyen induit par le parcours des clients ne faisant pas partie des routes partielles par les $m - f$ clients restants. On a ainsi une borne P telle que :

$$P = \sum_{h=1}^{f+1} P_h \tag{4.14}$$

Notons b le nombre de chaînes et $b-1$ le nombre de composants non structurés d'une route partielle h. Notons également S_h^r l'ensemble des clients de la r^e chaîne de la route partielle h. Cet ensemble peut être composé d'un seul client, et s'exprime par $S_h^r = \{v_1^{rh}, \ldots, v_{l_{rh}}^{rh}\}$. Par soucis de simplicité, $v_{l_{rh}}^{rh}$ sera remplacé par $v_{l_{rh}}$ dans tout ce qui suit. On note enfin U_r^h l'ensemble des clients du r^e composant non structuré de la route partielle h et $R_h = (\bigcup_{r=1}^{b} S_h^r) \cup (\bigcup_{r=1}^{b-1} U_h^r)$. L'inégalité valide qui est alors ajoutée est :

$$(P - L)\Big(\sum_{h=1}^{r} W_h(x) - r + 1\Big) + L \leq \theta \tag{4.15}$$

où :

$$\begin{aligned} W_h(x) = &\sum_{r=1}^{b} \sum_{\substack{(v_i,v_j)\in S_h^r \\ v_i \neq v_1}} 3x_{ij} + \sum_{(v_1,v_j)\in S_h^1} x_{1j} + \sum_{(v_1,v_j)\in S_h^b} x_{1j} + \sum_{r=1}^{b-1} \sum_{(v_i,v_j)\in U_h^r} 3x_{ij} \\ &+ \sum_{r=1}^{b-1} \sum_{\substack{v_j \in U_h^r \\ v_{l^{rh}} \neq v_1}} 3x_{l^{rh},j} + \sum_{r=2}^{b} \sum_{\substack{v_j \in U_h^{r-1} \\ v_{1^{rh}} \neq v_1}} 3x_{1^{rh},j} + \sum_{\substack{v_j \in U_h^1 \\ v_{l1h} \neq v_1}} x_{l^{1h},j} + \sum_{\substack{v_j \in U_h^{b-1} \\ v_{1^{bh}} = v_1 \\ v_{l^{b-1,h}} \neq v_1}} x_{1^{bh},j} \\ &- (3|R_h| - 5) \end{aligned} \tag{4.16}$$

On ne justifiera pas ici la validité de l'expression (4.15), ceci a été fait dans Jabali *et al.* (2014).

4.4.3 Coupes LBF basées sur des chaînes

Dans cette partie consacrée aux coupes LBF basées sur des chaînes, nous commencerons par introduire le principe général sur lequel repose ce type de coupes. Dans la section suivante, nous expliquerons comment est calculée la borne associée, puis nous discuterons des différentes expressions littérales que les coupes peuvent prendre avant de présenter celle que l'on retiendra. Finalement, nous détaillerons la procédure de séparation qui permet d'ajouter

des coupes efficaces.

Principe général

Les routes partielles identifiées ont l'avantage de fournir des coupes qui appliquent une borne inférieure de qualité sur la variable θ, elle-même une borne inférieure sur le coût de recours réel $Q(x)$. Cependant, pour être identifiées dans une solution intermédiaire, elles supposent l'existence d'une telle route partielle. Or, la relaxation du problème permet des solutions fractionnaires qui n'auront pas forcément cette forme. De plus, une observation importante des travaux de (Jabali *et al.* (2014)) est la conjecture selon laquelle les coupes LBF les plus efficaces sont celles qui sont actives sur le plus grand nombre de solutions fractionnaires ou entières possibles, quitte à diminuer la valeur de la borne associée.

C'est d'après ces observations qu'il est paru nécessaire, ou du moins envisageable, d'imaginer d'autres types de coupes basées sur le principe des coupes LBF. Ceci a également été motivé par le degré de liberté laissé dans leur conception, tel qu'on l'a décrit dans la section 4.4.1.

On se propose donc d'implémenter de nouvelles coupes, basées cette fois sur l'identification de chaînes d'arêtes entières au sein des solutions intermédiaires. Pour reprendre les notations de la section 4.4.1, l'ensemble S_p sera donc un ensemble de variables représentant des arêtes liées entre elles. Ce sous-ensemble est moins restrictif que ceux en lien avec les routes partielles, car les chaînes n'ont pas l'obligation de commencer et de se terminer au dépôt. Cela présente l'avantage de toucher un grand nombre de solutions puisque toutes les variables n'appartenant pas à une des arêtes de S_p pourront prendre une valeur quelconque, entière ou fractionnaire, tout en laissant la coupe active. De plus, un deuxième avantage est qu'il sera possible d'identifier un grand nombre de coupes potentielles grâce à l'algorithme de séparation dont on discutera dans les sections suivantes. En effet, après observation du comportement de l'algorithme et de ses différentes solutions intermédiaires, on remarque qu'il est très commun que des arêtes prennent des valeurs entières et soient liées entre elles, bien que la solution globale soit fractionnaire.

Calcul de la borne

Le calcul de la borne est en fait séparable pour chaque chaîne. L'idée est de prévoir le coût espéré qu'on sera certain de subir si on emprunte l'ensemble des arêtes concernées par la coupe. La manière dont on les emprunte ne doit pas avoir d'importance.

Supposons que l'on dispose de H chaînes pour une coupe. On va détailler le calcul de l'apport de la chaîne $h \in [\![1; H]\!]$. On suppose qu'elle est composée de t_h arêtes et on note sa suite de clients $(v_{1(h)}, v_{2(h)}, \ldots, v_{t_h+1(h)})$. L'apport de la chaîne h pour la borne totale sera noté P^h. Pour envisager le cas le plus défavorable et s'assurer que l'on calcule bien une borne inférieure, on va partir du principe que le véhicule arrive en début de chaîne avec la totalité de sa capacité. Bien sûr, les deux sens de parcours seront pris en compte et la valeur minimale sera conservée. On note ainsi $P^h = \min\{P^{h,1}, P^{h,2}\}$, où $P^{h,1}$ et $P^{h,2}$ sont les bornes obtenues dans chaque sens de parcours de la chaîne h.

Comme pour le calcul de $Q(x)$ pour une solution x entière, le calcul de P^h va consister à parcourir les clients de la chaîne un à un en envisageant à chaque fois les différents scénarios pouvant mener à un échec à son niveau. Ses différents scénarios vont prendre la forme des l^e échecs. On a ainsi :

$$P^{h,1} = \sum_{i=2}^{t_h+1} 2c_{1i(h)} \sum_{l=1}^{i-1} P\left(\sum_{s=1}^{i-1} \xi_{s(h)} \leq lD < \sum_{s=1}^{i} \xi_{s(h)} \right) \tag{4.17}$$

Le calcul de $P^{h,2}$ se fait de manière semblable en renversant la liste des clients. La somme commence à l'indice du client $2(h)$ car la distribution des demandes est tronquée en D. La probabilité d'observer un échec au niveau du client $1(h)$ est donc nulle.

Pour calculer P, la borne globale de la coupe, on somme les apports des différentes chaînes et on a donc :

$$P = \sum_{h=1}^{H} P^h \tag{4.18}$$

En fait, sommer un à un les apports revient une nouvelle fois à considérer un cas défavorable, ce qu'on est forcé de faire pour s'assurer que l'on calcule bien une borne inférieure. En effet, cela suppose que chaque chaîne est parcourue dans le sens le plus défavorable et qu'en plus le véhicule qui les parcourt arrive avec la totalité de sa capacité.

Il faut se rappeler que les coupes qu'on ajoute, dont l'expression littérale sera exprimée dans la section ci-dessous, doivent être valides uniquement pour des solutions entières. La borne qu'elles apportent sur le coût de recours des solutions fractionnaires n'a pas de sens réel, bien qu'elle soit utile pour éliminer les solutions fractionnaires qu'il faudra dans tous les cas empêcher d'apparaître, au moins avec un branchement. Les solutions entières du problème forment m routes. Il se peut donc très bien que pour une de ces solutions, les chaînes

d'une coupe soient empruntées par le même véhicule. Ainsi, cela force certaines chaînes à être parcourues dans un sens fixe, ce qui peut entraîner le coût maximal des deux sens de parcours. De plus, dans ce cas de figure, les véhicules arrivent en début d'une chaîne avec une capacité résiduelle inférieure à la capacité totale, ce qui augmente la probabilité de subir un échec au niveau de la chaîne. Ceci souligne bien le fait que le coût calculé est une borne inférieure, et qu'il suffit bien de sommer l'apport de chaque chaîne pour déterminer la borne globale P. En réalité, le coût réel subi par l'emprunt des chaînes par une solution entière pourra être supérieur.

Les H chaînes devront forcément appartenir à l'une des m routes. On peut donc être sûr que plusieurs chaînes seront parcourues par le même véhicule si la coupe comporte plus de chaînes qu'il n'y a de véhicules dans le problème, c'est-à-dire si $H > m$. Une idée pour améliorer la borne P serait donc d'énumérer les différentes associations de chaînes, de calculer la valeur améliorée de P et de garder la valeur minimale pour conserver la validité de la borne inférieure. Par exemple, si $H = m+1$, on sait qu'il y aura au minimum une association de deux chaînes parcourues par le même véhicule. On peut donc énumérer tous les regroupements de deux chaînes, en prenant soin de prendre en compte les deux liaisons possibles, pour former m chaînes et calculer la valeur de P correspondante. Cependant, en pratique, on n'observe malheureusement aucune coupe pour laquelle $H > m$ dans les instances que l'on utilise pour ce mémoire et qui sont issues de la littérature.

Modification du calcul de la borne

Dans le calcul qu'on vient de détailler, on part de l'hypothèse fondamentale selon laquelle le coût minimum engendré par le parcours d'une chaîne est celui subi quand le véhicule arrive en début de chaîne avec toute sa capacité. En effet, il peut paraître intuitif d'affirmer cela car une capacité pleine assure que la probabilité d'avoir au moins un échec est minimale. Cependant, les échecs peuvent avoir lieu aux différents clients et si le véhicule a toute sa capacité, la probabilité d'échec est plus élevée vers les derniers clients de la chaîne. Or, si les premiers clients ont une distance au dépôt plus faible, on pourrait peut-être diminuer le coût en déplaçant la probabilité d'échec vers le début de la chaîne, ce qui est possible si la capacité du véhicule en entrée n'est pas pleine.

Ainsi, pour s'assurer de la validité de la borne inférieure qu'on calcule, on modifie l'ex-

pression (4.17). En étudiant toujours la chaîne $(v_{1(h)}, v_{2(h)}, \ldots, v_{t_h+1(h)})$, on définit :

$$P_{i,l}(\xi) = P\left(\xi + \sum_{s=1}^{i-1} \xi_{s(h)} \leq lD < \xi + \sum_{s=1}^{i} \xi_{s(h)}\right) \tag{4.19}$$

En notant $\xi_0 \sim \mathcal{N}(1, S_0)$, où S_0 est défini en annexe, le calcul de $P^{h,1}$ devient :

$$P^{h,1} = \sum_{i=2}^{t_h+1} 2c_{1i(h)} \left[\min \left[P_{i,1}(0), P_{i,1}(\xi_0) \right] + \sum_{l=2}^{i-1} P_{i,l}(0) \right] \tag{4.20}$$

La preuve et le détail de ce résultat sont décrits en annexe.

Expression des coupes

On va d'abord présenter le cas général pour lequel il faut déterminer l'expression de la coupe ajoutée pour n'importe quel ensemble de chaînes. La coupe générique (4.13) présentée par Hjorring et Holt (1999) pourrait à première vue convenir. En effet, en notant $S_p = \{[i, j]|$ arête $[i, j]$ fait partie de la coupe$\}$, alors il suffit d'ajouter la coupe suivante au modèle pour imposer à θ de prendre la valeur P si la solution courante emprunte les arêtes de la coupe :

$$(P - L)(\sum_{[i,j] \in S_p} x_{ij} - (|S_p| - 1)) + L \leq \theta \tag{4.21}$$

Cependant, pour assurer la validité d'une coupe, il faut qu'elle soit active sur la solution entière qu'elle concerne et applique la borne spécialement calculée pour ce cas de figure, mais également qu'elle soit redondante pour les autres solutions entières. Nous allons voir à travers un exemple que d'autres solutions entières peuvent être concernées par la coupe (4.21) alors que la borne P n'a pas été calculée pour celles-ci.

Supposons qu'une coupe soit ajoutée et concerne une seule chaîne impliquant les arêtes de $S_p = \{[1, 2], [2, 3], [3, 4]\}$. La chaîne est illustrée à la figure 4.2.

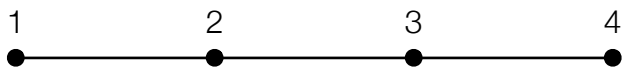

Figure 4.2 Exemple de chaîne

Elle sera ainsi active si $\sum_{[i,j] \in S_p} x_{ij} = 3$. Or, une autre solution entière donne le même

résultat. Cela est rendu possible, car les arêtes liées au dépôt peuvent prendre la valeur de 2 dans le cas d'un aller-retour. Les valeurs des variables de ce cas de figure sont indiquées à la figure 4.3 (le flot des différentes arêtes est représenté sur celles-ci).

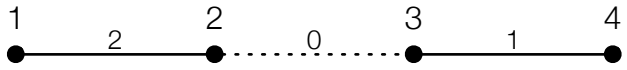

Figure 4.3 Configuration qui active la coupe

En fait, chaque fois qu'une arête liée au dépôt est présente dans la coupe, on aura ce cas de figure qui pourra apparaître et activer de façon non souhaitée la coupe. Pour pallier cet obstacle, la solution est d'affecter un coefficient 2 aux variables non liées au dépôt et un coefficient 1 pour celles qui y sont liées. De cette manière, on accorde plus de poids aux variables concernant les arêtes non liées au dépôt. Ainsi, quand l'une d'elles ne sera plus empruntée au profit d'une autre liée au dépôt, la coupe cessera d'être active. On définit donc les deux ensembles $S_{p_1} = \{[1,i] \mid \text{Arête } [1,i] \text{ fait partie de la coupe}\}$ et $S_{p_2} = \{[i,j] \mid i \neq 1 \text{ et l'arête } [i,j] \text{ fait partie de la coupe}\}$. L'expression de la coupe modifiée est la suivante :

$$(P-L)(\sum_{[1,i]\in S_{p_1}} x_{1i} + \sum_{[i,j]\in S_{p_2}} 2x_{ij} - (|S_{p_1}| + 2|S_{p_2}| - 1)) + L \leq \theta \tag{4.22}$$

Grâce à cette expression modifiée, lorsque le flot d'une arête liée au dépôt gagne une valeur de 1 à cause d'un aller-retour au détriment du flot de l'arête suivante qui perd cette valeur, l'expression $\sum_{[1,i]\in S_{p_1}} x_{1i} + \sum_{[i,j]\in S_{p_2}} 2x_{ij}$ perd également la valeur de 1, ce qui rend la coupe inactive, contrairement à l'expression $\sum_{[i,j]\in S_p} x_{ij}$ dont la valeur restait inchangée. C'est également dans ce souci d'éviter les exceptions liées en général au dépôt qui activent les coupes de façon intempestive que l'expression des coupes décrites dans Jabali *et al.* (2014) contient elle aussi des coefficients différents selon les types d'arêtes rencontrés.

L'expression (4.22) permet d'ajouter des coupes en lien avec n'importe quelle configuration de chaînes. La validité mathématique sera toujours respectée sous cette forme. Néanmoins, il faut garder en tête que les coupes que l'on cherche à développer doivent être efficaces et qu'il serait avantageux pour la coupe de toucher le plus grand nombre de solutions.

Une première remarque ayant son importance est qu'il n'est pas forcément nécessaire de

prendre en compte le dépôt dans les chaînes que l'on recherche, malgré tout ce dont on vient de parler. En effet, le parcourir en début ou en fin de chaîne ne change pas la valeur de P puisque la demande à servir en parcourant la chaîne est la même avec ou sans le dépôt.

Ceci présente un double avantage. Le premier est que l'expression de la coupe comporte une variable en moins, ce qui la rend active sur un plus grand nombre de solutions tout en conservant la même valeur pour la borne P, car la variable liant le dépôt au premier client de la chaîne pourra prendre n'importe quelle valeur tout en laissant la coupe active.

Deuxièmement, cela va permettre de s'affranchir des différents coefficients que l'on a introduit dans l'expression (4.22) et d'affecter un coefficient de 1 à toutes les variables. Les laisser tels quels conserverait la validité de la coupe, mais les remplacer par 1 permet à celle-ci d'être active sur plus de solutions fractionnaires.

Pour illustrer cela, prenons l'exemple simple suivant : on dispose d'une coupe qui concerne une chaîne dont les arêtes sont $S_p = \{[2,3],[3,4],[4,5]\}$ et dont la borne est P. Si on applique la forme générale de (4.22), l'expression de la coupe est alors :

$$\Big(P - L\Big)\Big(2(x_{23} + x_{34} + x_{45}) - 5\Big) + L \leq \theta$$

Comme on a nécessairement $P > L$ lors de l'ajout d'une coupe, alors celle-ci impose à θ de prendre une valeur supérieure ou égale à L si et seulement si :

$$2(x_{23} + x_{34} + x_{45}) - 5 \geq 0 \Leftrightarrow x_{23} + x_{34} + x_{45} \in [2.5; 3]$$

Si on affecte des coefficients 1 aux variables, la coupe est alors :

$$\Big(P - L\Big)\Big(x_{23} + x_{34} + x_{45} - 2\Big) + L \leq \theta$$

et le membre de gauche est supérieur ou égal à L si et seulement si :

$$x_{23} + x_{34} + x_{45} - 2 \geq 0 \Leftrightarrow x_{23} + x_{34} + x_{45} \in [2; 3]$$

On remarque donc que la coupe avec les coefficients 1 est moins restrictive et s'active pour un plus grand nombre de solutions fractionnaires, ce qui est utile pour l'accélération de la résolution.

Les coupes basées sur des chaînes que nous introduisons ne prennent donc pas en compte le dépôt. Cela permet d'étendre l'ensemble des solutions fractionnaires touchées en enlevant une variable dans l'expression de la coupe et en ayant la possibilité d'affecter des coefficients avantageux. La coupe finale est donc :

$$(P - L)(\sum_{[i,j]\in S_p} x_{ij} - (|S_p| - 1)) + L \leq \theta \tag{4.23}$$

avec

$$S_p = \{[i,j] \mid i \neq 1 \text{ et l'arête } [i,j] \text{ fait partie de la coupe}\} \tag{4.24}$$

Méthode de séparation

Nous allons maintenant décrire la manière dont sont ajoutées les coupes au modèle à partir des solutions intermédiaires rencontrées dans l'algorithme. La méthode que l'on va décrire a été pensée pour trouver des coupes violées par la solution courante x qui sont actives sur le plus grand nombre possible de solutions fractionnaires et entières. Cela se caractérise principalement par la recherche de coupes comportant le plus petit nombre de variables tout en assurant d'avoir une borne P qui sera utile pour la progression de l'algorithme.

On effectue une première étape qui consiste à identifier l'ensemble des chaînes dont les variables représentant les arêtes sont entières. Celle-ci consiste à identifier les variables x_{ij} de la solution courante x pour lesquelles $1 - \epsilon \leq x_{ij} \leq 1$, avec $\epsilon = 10^{-10}$. On ne cherche pas les variables entières valant 2 car un aller-retour entre un client et le dépôt n'entraîne aucun recours. Chaque fois qu'une variable entière est identifiée, on met à jour une liste de type *client* : {clients reliés par une arête avec un flot entier}. Il est ensuite possible de parcourir cette liste pour former les chaînes. On peut ensuite initialiser l'algorithme de séparation.

Soit :

- (x, θ) la solution courante
- P la valeur de la borne apportée par la coupe courante
- M l'ensemble des H chaînes présentes dans la solution x. La ligne h de M représente les variables contenues dans la chaîne h qui possède t_h arêtes :

$$M = \begin{Bmatrix} \{x_{1(1)2(1)}, & \cdots & , x_{t_1(1)t_1+1(1)}\} \\ \vdots & & \vdots \\ \{x_{1(H)2(H)}, & \cdots & , x_{t_H(H)t_H+1(H)}\} \end{Bmatrix}$$

- h l'indice de la chaîne sur laquelle on travaille
- CC la chaîne courante que l'on essaye d'ajouter à la coupe
- A l'ensemble des variables contenues dans la coupe courante
- $P(CC)$ la borne apportée par la chaîne CC, le calcul se fait en considérant les deux sens de parcours et en utilisant la formule (4.20)

L'algorithme est le suivant (voir aussi le diagramme à la figure 4.4) :

1. Initialisation

- $P := 0$
- $A := \varnothing$
- $h := 1$
- $CC := \{x_{1(1)2(1)}\}$

2. Apport de la prochaine arête

- Supprimer la première arête de la chaîne h
- Si la chaîne h est vide
  - Aller à l'étape 3
- Sinon
  - Ajouter le premier élément de la chaîne h à la fin de CC
  - Calculer $P(CC)$
  - Si $P(CC) + P > \theta + 1$
    - Aller à l'étape 4
  - Sinon
    - Revenir à l'étape 2

3. Fin de la chaîne h

- Calculer $P(CC)$
- Si $P(CC) > 0.01$
  - Mettre les éléments de CC dans A
- $h := h + 1$
- Si $h \leq H$
  - Ré-initialiser $CC := \{\}$
  - $CC := \{x_{1(h)2(h)}\}$
  - Aller à l'étape 2
- Sinon, arrêter

4. Ajout de la coupe

- $P := P + P(CC)$
- Mettre les éléments de CC dans A
- Ajouter la coupe contenant les variables de A et la borne P au modèle en respectant l'expression (4.23)
- Si M est vide
  - Arrêter
- Sinon
  - Renommer les éléments restants de M de sorte que M prenne sa forme initiale
  - Revenir à l'étape 1

L'idée principale est de construire de manière itérative des chaînes de plus en plus grandes. Dès qu'une ou plusieurs chaînes apportent une borne P suffisante, on ajoute directement la coupe sans attendre l'apport des arêtes restantes qui pourraient augmenter de nouveau la borne. C'est ce qu'on teste avec la condition $P(CC) + P > \theta + 1$ de l'étape 2 : si l'apport de la chaîne courante additionné à l'apport des chaînes prises en compte précédemment est suffisant, alors on se dirige vers l'étape 4 qui ajoute la coupe au modèle.

Le seuil $\theta + 1$ assure que la coupe ajoutée soit bien violée par la solution courante. En toute généralité, il suffit que ce seuil soit strictement supérieur à θ. Ici, le seuil $\theta + 1$ a été choisi de manière empirique. On a remarqué d'après les différents tests effectués que cette valeur donnait de bons résultats. Ces derniers seront présentés dans le chapitre 5.

Une fois qu'une coupe est ajoutée à l'étape 4, l'algorithme ne s'arrête pas forcément. S'il reste des chaînes dans M qu'on n'a pas encore étudiées, alors on reprend la procédure depuis le début pour chercher d'autres coupes. Ceci permet d'augmenter rapidement la borne du prochain nœud de l'algorithme L-shaped en nombres entiers car, grâce aux différentes coupes ajoutées au noeud courant, plusieurs formes de solutions seront évitées pour les prochaines résolutions. Si elles sont empruntées, alors la variable θ prendra la valeur de la borne P calculée.

Quand une chaîne h de M a été étudiée entièrement, mais que la borne totale n'est pas suffisante, on vérifie que son apport est suffisant, puis, on passe à la chaîne $h + 1$ si cela est possible pour continuer la recherche. L'apport d'une chaîne courante CC pour la coupe globale est jugé suffisant si sa borne $P(CC)$ est telle que $P(CC) > 0.01$. Encore une fois, la valeur 0.01 est une valeur empirique qui s'est révélée être efficace lors des différents tests. Cette condition à respecter permet de ne pas encombrer la coupe avec un ensemble de va-

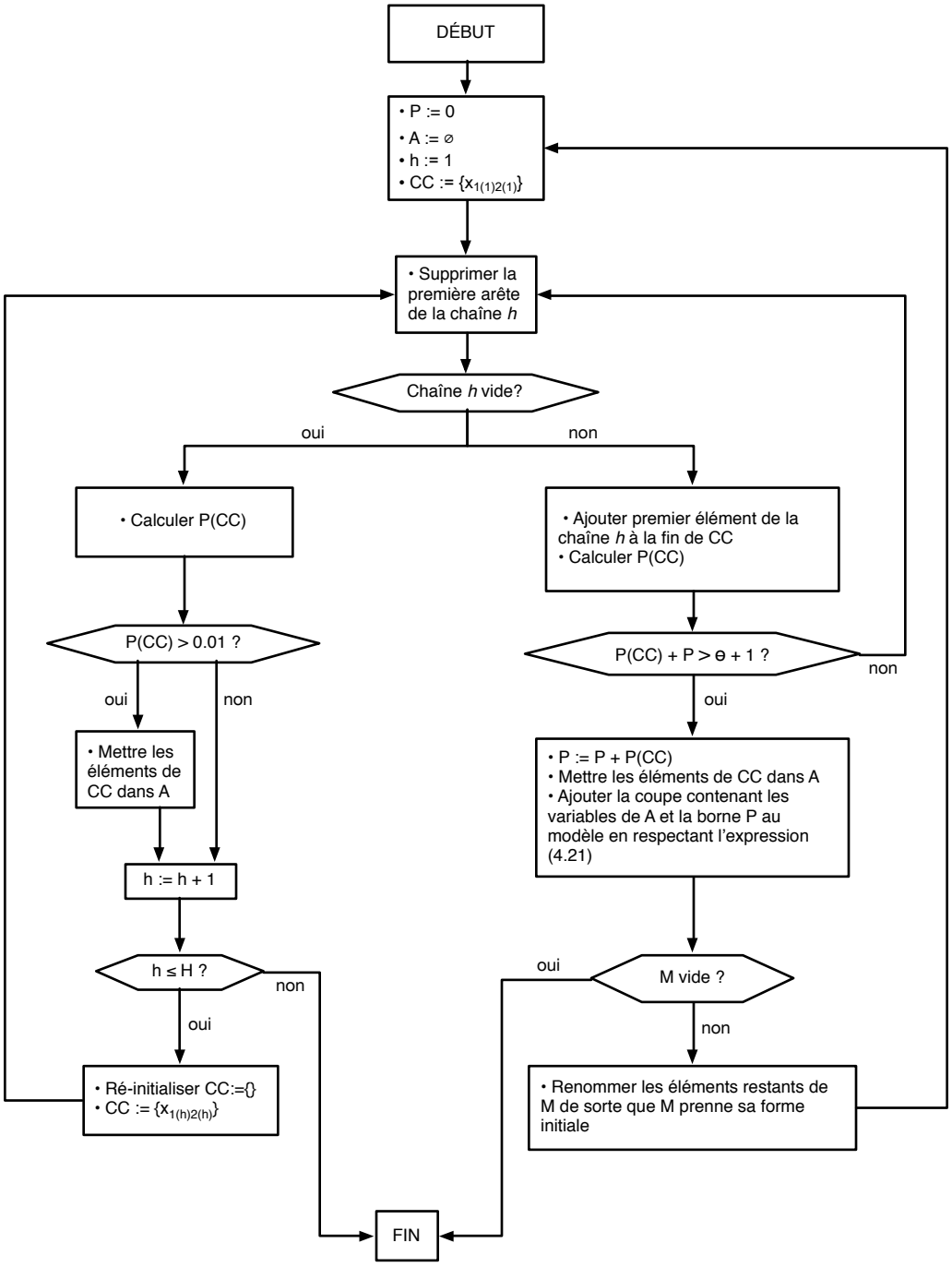

Figure 4.4 Diagramme de l'algorithme de séparation des coupes basées sur des chaînes

riables si elles n'apportent qu'une valeur très petite à la borne globale P.

Enfin, il existe deux manières de terminer la procédure de séparation. La première est celle de l'étape 3. Elle apparaît quand on a fini d'étudier la dernière chaîne de M sans que la borne globale P soit suffisante. La deuxième façon d'arrêter la procédure est celle de l'étape 4 qui a lieu quand on ajoute une coupe au modèle et qu'il ne reste plus de chaînes dans M à étudier.

4.4.4 Coupes LBF basées sur des ensembles non structurés de nœuds

Dans cette section, nous allons premièrement décrire le principe général sur lequel repose les coupes. Nous présenterons ensuite le calcul de la borne associée avant d'aborder l'expression littérale qu'on ajoute au modèle. Enfin, nous détaillerons les deux méthodes de séparation employées.

Principe général

Le modèle de coupes LBF basées sur des chaînes offre la possibilité intéressante de fournir des coupes violées relativement fréquemment qui renforcent la relaxation du modèle. Ceci est rendu possible par la bonne qualité de la borne fournie ce qui remplit souvent la condition d'ajout. Cependant, un compromis entre le nombre de solutions fractionnaires et entières actives et la qualité de la borne doit être fait pour les coupes LBF. Durant les simulations effectuées, on se rend compte qu'après l'ajout d'une coupe sensée être active sur des chaînes de la solution x^ν, la solution suivante $x^{\nu+1}$ emprunte souvent les chaînes de la coupe mais en changeant l'ordre de visite de quelques clients. Ceci donne lieu à la génération d'autres coupes actives sur ces nouvelles chaînes, et le processus continue jusqu'à ce que changer l'ordre des clients devient plus coûteux que de subir la borne d'une coupe LBF. Le modèle s'encombre ainsi rapidement de coupes ce qui ralentit les différentes résolutions.

Il semblerait donc intéressant de pouvoir ajouter une coupe qui resterait active si l'ordre de visite change d'une solution à l'autre. C'est d'après ce constat que nous nous sommes proposé d'implémenter des coupes basées sur l'identification d'ensembles non structurés de nœuds. D'un point de vue qualitatif, le rôle de ces ensembles doit être de permettre d'activer une coupe quel que soit l'ordre de visite des clients qui les composent.

Plus précisémment, il s'agit d'identifier un ensemble S_p d'arêtes $[i, j]$ qui remplit les conditions évoquées. Supposons que l'on dispose de H ensembles non structurés de clients de ce type. Soit h l'indice de l'un d'entre eux nommé $C_h = \{v_{1(h)}, v_{2(h)}, \ldots, v_{t_h+1(h)}\}$. Alors, dans le graphe du problème $G = (V, E)$, l'ensemble d'arêtes $E(C_h)$ composé des arêtes de E ayant

leurs deux extrémités dans C_h permet aux clients de C_h de former une clique. L'ensemble S_p sera alors $S_p = \bigcup_{h=1}^{H} E(C_h)$. En effet, une solution entière visitant les clients de C_h dans un ordre quelconque prendra toujours la forme d'une chaîne d'arêtes de $E(C_h)$.

Le nombre de routes pouvant visiter ces ensembles C_h peut être calculé. Si on énumère l'ensemble des ordres possibles que peuvent prendre les t_h+1 clients de C_h, alors on disposera de tous les ordres de parcours en double puisque le sens n'a pas d'importance. Ce nombre est égal à $(t_h + 1)!$, et donc nos ensembles non structurés de nœuds permettront d'activer la coupe pour $\frac{(t_h+1)!}{2}$ configurations. On se rend compte de l'efficacité théorique des coupes qu'on cherche à implémenter quand on compare ce nombre aux coupes LBF basées sur des chaînes pour lesquelles un seul ordre de parcours active la coupe.

Calcul de la borne

Comme pour la borne des coupes LBF basées sur les chaînes, il nous faut déterminer P, la valeur minimale du coût engendré par la visite des clients des ensembles C_h. Pour cela, on va de nouveau considérer que le coût est décomposable pour chaque ensemble C_h. Ceci se justifie par le fait qu'il faut se situer dans le cas le plus défavorable dans lequel un véhicule visite un ensemble C_h en ayant une quantité de marchandise égale à sa capacité. On ne se soucie donc pas de savoir si les ensembles de clients sont visités sur la même route et on peut simplement sommer les coûts séparément.

On renomme les clients de C_h dans l'ordre croissant de leur distance avec le dépôt. Soit donc $C_h = \{v_{1(h)}, v_{2(h)}, \ldots, v_{t_h+1(h)}\}$ avec $v_{1(h)}$ le client le plus près du dépôt, $v_{2(h)}$ le second plus près, etc. Comme on ne connaît pas l'ordre de visite des clients, on va travailler avec la demande totale de l'ensemble de clients qu'on dénote $\xi_T(h)$. L'idée est d'envisager les différents scénarios dans lesquels on peut observer un ou plusieurs échecs. Ceux-ci prennent la forme suivante : la première situation est celle dans laquelle il apparaît exactement un échec quand un véhicule visite l'ensemble des clients de C_h. On suppose qu'il aura lieu au client le plus proche $v_{1(h)}$ pour assurer à la borne calculée qu'elle soit bien une borne inférieure. La deuxième situation est celle dans laquelle il apparaît exactement deux échecs, le premier ayant lieu au niveau du client $v_{1(h)}$, le deuxième au niveau de $v_{2(h)}$, etc, jusqu'à l'apparition $t_h + 1(h)$ échecs ayant lieu au niveau de tous les clients en suivant l'ordre établi dans C_h.

L'expression de la borne est donc la suivante :

$$P = \sum_{i=1}^{t_h+1} \sum_{l=1}^{i} c_{1l(h)} P\Big(lD \leq \xi_T(h) \leq (l+1)D\Big) \tag{4.25}$$

Contrairement au calcul de la borne des coupes LBF basées sur les chaînes d'arêtes, on n'a pas à considérer de sens de parcours puisque la borne doit être valide pour un ordre quelconque. C'est pour cela que la borne (4.25) est inférieure à la borne (4.17).

Expression des coupes

La première chose à noter est que la recherche des ensembles non structurés de noeuds ne prendra pas en compte les arêtes liées au dépôt. On observe facilement dans la formule (4.25) que la présence du dépôt dans l'ensemble C_h n'influence pas la valeur de P car la demande totale n'est pas augmentée et la distance entre le dépôt et lui-même est nulle. De plus, cela permet de s'affranchir des différents coefficients que l'on doit imposer aux variables de la coupe pour éviter l'activation de celle-ci sur des solutions non prévues dans le calcul de P. Comme discuté pour le calcul de la borne des coupes LBF basées sur les chaînes, pouvoir affecter les mêmes coefficients aux variables d'une coupe permet son activation sur un plus grand nombre de solutions fractionnaires.

Il faut maintenant introduire une fonction qui permet d'activer la coupe pour toutes les configurations de routes visitant les clients d'un ensemble C_h. Les variables doivent être celles représentant les arêtes liant les clients de l'ensemble. Le rôle de cette fonction doit également être d'empêcher l'activation intempestive de la coupe sur d'autres solutions non prévues. Définissons ainsi la fonction F_h suivante :

$$F_h(x) = \sum_{i \in C_h} \sum_{\substack{j \in C_h \\ i<j}} x_{ij} \tag{4.26}$$

S'il y a t_h+1 clients dans C_h, alors une route visitant l'ensemble de ses clients empruntera t_h arêtes ce qui sera le résultat de $F_h(x)$ si la solution x possède cette route. Soit donc :

$$W_h(x) = F_h(x) - t_h + 1 \tag{4.27}$$

La fonction W_h est alors négative sur toute solution entière x qui ne visite pas l'ensemble C_h en une seule fois. En revanche, pour toutes les solutions entières pour lesquelles c'est le cas, $W_h(x) = 1$. On peut visualiser le cas d'une telle solution à la figure 4.5. L'ensemble C_h est rempli de gris et est délimité par une frontière fictive. Les traits pleins représentent des arêtes dont le flot est entier.

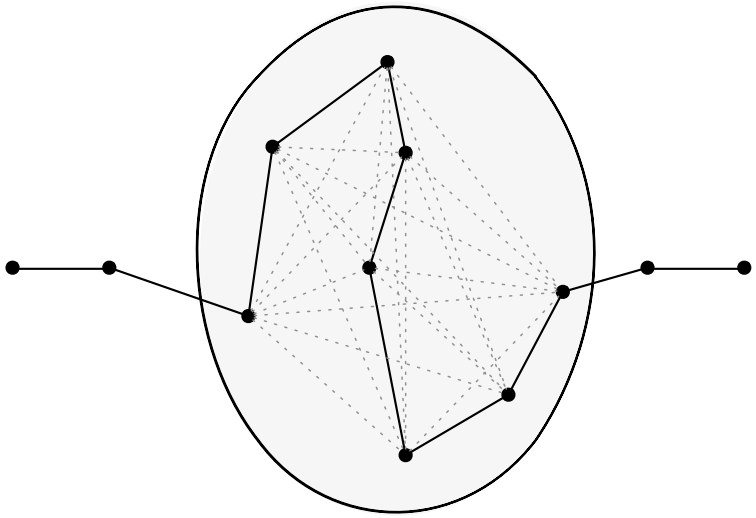

Figure 4.5 Solution entière qui traverse l'ensemble C_h

En supposant qu'il y a H ensembles non structurés de noeuds, la forme littérale finale des coupes est donc semblable à (4.23) :

$$\Big(P - L\Big)\Big(\sum_{h=1}^{H} W_h(x) - (H-1)\Big) + L \leq \theta \tag{4.28}$$

qui peut encore se réécrire : $\Big(P - L\Big)\Big(\sum\limits_{h=1}^{H}\sum\limits_{i\in C_h}\sum\limits_{\substack{j\in C_h \\ i<j}} x_{ij} - (\sum\limits_{h=1}^{H} t_h - 1)\Big) + L \leq \theta.$

On remarque que $F_h(x)$ représente la somme du flot total sur les arêtes de la clique C_h. Ainsi, le fait d'avoir choisi de supprimer l'apport du dépôt des ensembles permet de conserver la même borne P tout en supprimant $|C_h|$ variables dans (4.28) puisqu'on évite toutes les arêtes liant le dépôt aux clients.

Les coupes basées sur des chaînes de la section 4.4.3 pouvaient être actives sur des solutions fractionnaires de deux manières : Premièrement en permettant aux arêtes non prises en compte dans la coupe de prendre n'importe quelle valeur, deuxièmement en laissant active la borne si les chaînes sont traversées par un flot proche de leur valeur entière. Pour les coupes de cette section, on bénéficie toujours du premier cas de figure puisque les variables non prises en compte dans les coupes peuvent toujours prendre une valeur quelconque. En revanche, le deuxième cas de figure s'améliore nettement ici : le flot traversant la clique composée des clients d'un ensemble C_h peut prendre la valeur t_h qui active la borne pour

des flots fractionnaires. C'est un des points forts de ce type de coupe dont les performances seront discutées dans le chapitre 5. À la figure 4.6, on peut observer un exemple de flot fractionnaire traversant un ensemble non structuré de nœuds. Les arêtes avec un flot entier sont à nouveau représentées par un trait noir plein, les arêtes avec un flot fractionnaire sont représentées par un trait pointillé épais noir. Enfin, les arêtes de la clique C_h ayant un flot nul sont représentées par un trait pointillé fin en gris.

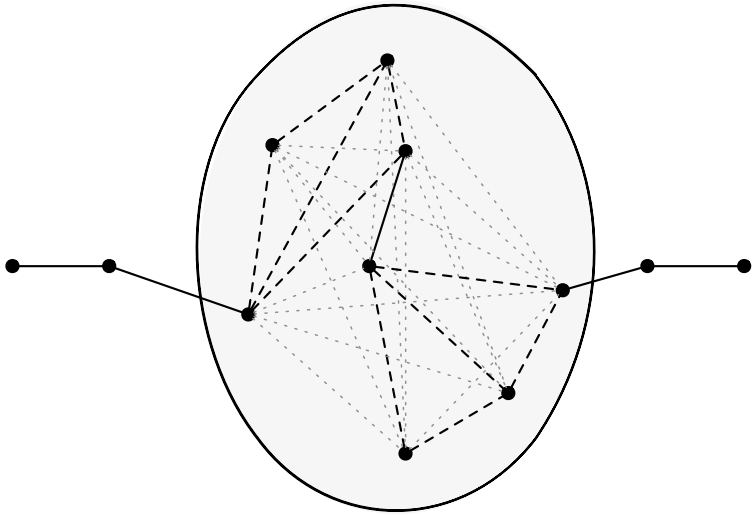

Figure 4.6 Solution fractionnaire qui traverse l'ensemble C_h

Il existe une multitude de configurations de flots fractionnaires qui activent la coupe. Cela rend la coupe robuste par rapport aux différentes solutions fractionnaires rencontrées dans la résolution du problème. Un nombre restreint de coupes sera donc suffisant pour fournir une approximation du coût de recours qui ne sera pas simplement égal à la borne générale L. Cette quantité limitée de coupes permet au modèle de ne pas trop s'encombrer et permet à l'algorithme du simplexe de trouver rapidement la solution des modèles relaxés aux différents nœuds de branchement.

La conception des ensembles de nœuds non structurés est nettement inspirée de ceux mis en place par Jabali *et al.* (2014), eux-mêmes inspirés des travaux de Hjorring et Holt (1999) et Laporte *et al.* (2002). Le flot les traversant avait alors la contrainte d'avoir un nœud d'entrée et de sortie fixé. De plus, ils devaient appartenir à une route partielle, qui est moins fréquemment identifiable dans une solution intermédiaire que nos ensembles indépendants. Les ensembles non structurés de nœuds de ce mémoire se différencient dans le sens où ils

peuvent être identifiés dans une partie du graphe de la solution sans avoir de contraintes sur les nœuds d'entrée et de sortie, ce qui laisse plus de liberté dans leur création. La contrepartie de ces avantages est la perte de qualité de la borne P apportée. C'est pourquoi il serait avantageux de combiner les coupes LBF basées sur des ensembles non structurés de nœuds avec d'autres coupes touchant un nombre restreint de solutions, mais apportant une borne P élevée, comme les coupes de la section 4.4.3. Nous montrerons les avantages de combiner nos deux ensembles de coupes dans le chapitre 5.

Les deux méthodes de séparation

Pour identifier des ensembles de type C_h qui fournissent une coupe violée par la solution courante, on va utiliser deux heuristiques qui fonctionnent sur un principe semblable à l'algorithme de séparation des coupes LBF basées sur des chaînes.

La première consiste à modifier légèrement cet algorithme de séparation en modifiant le calcul de la borne P qui se fera cette fois grâce à la formule (4.25). Les différents ensembles C_h vont donc se construire itérativement à partir des chaînes identifiées dans la solution courante. En effet, celles-ci peuvent être vues comme une réalisation d'une route à travers un ensemble non structuré de nœuds parmi toutes les routes possibles. La coupe, si elle est identifiée sera donc violée par la solution courante et, on l'espère, sera active sur les solutions suivantes de l'algorithme L-shaped en nombres entiers ce qui augmentera la borne inférieure globale du problème.

Ce choix d'algorithme de séparation heuristique permet de bénéficier des mêmes avantages que l'algorithme de séparation des coupes basées sur des chaînes : produire des coupes comportant le plus petit nombre de variables possible tout en restant efficace. De plus, nous avons pu adapter rapidement l'implémentation informatique du premier algorithme pour écrire le second. En fait, le principe de l'algorithme de séparation des coupes basées sur des chaînes peut facilement se généraliser pour générer d'autres coupes de type LBF. Ici, on applique ce principe sous la forme suivante.

Soit :
- (x, θ) la solution courante
- P la valeur de la borne apportée par la coupe courante
- M l'ensemble des H chaînes présentes dans la solution x. La ligne h de M représente

les clients contenus dans la chaîne h qui possède t_h arêtes :

$$M = \left\{ \begin{array}{cccc} \{v_{1(1)}, & v_{2(1)}, & \cdots & , v_{t_{1+1}(1)}\} \\ \vdots & & & \vdots \\ \{v_{1(H)}, & v_{2(H)}, & \cdots & , v_{t_{H+1}(H)}\} \end{array} \right\}$$

- h l'indice de la chaîne sur laquelle on travaille
- EC l'ensemble non structuré de noeuds courant que l'on essaye d'ajouter à la coupe
- A l'ensemble des variables contenues dans la coupe courante
- $P(EC)$ la borne apportée par l'ensemble courant EC, le calcul se fait en utilisant la formule (4.25)

L'algorithme est le suivant :

1. Initialisation

- $P := 0$
- $A := \varnothing$
- $h := 1$
- $EC := \{v_{1(1)}\}$

2. Apport du prochain client

- Supprimer le premier client de la chaîne h
- Si la chaîne h est vide
  - Aller à l'étape 3
- Sinon
  - Ajouter le premier client de la chaîne h à l'ensemble EC
  - Calculer $P(EC)$
  - Si $P(EC) + P > \theta + 1$
    - Aller à l'étape 4
  - Sinon
    - Revenir à l'étape 2

3. Fin de la chaîne h

- Calculer $P(EC)$
- Si $P(EC) > 0.01$
  - Considérer EC comme une clique et mettre les variables représentant ses arêtes dans A

- $h := h + 1$
- Si $h \leq H$
  - Réinitialiser $EC := \{\}$
  - $EC := \{v_{1(h)}\}$
  - Aller à l'étape 2
- Sinon, arrêter

4. Ajout de la coupe

- $P := P + P(EC)$
- Considérer EC comme une clique et mettre les variables représentant ses arêtes dans A
- Ajouter la coupe contenant les variables de A et la borne P au modèle en respectant l'expression (4.28)
- Si M est vide
  - Arrêter
- Sinon
  - Renommer les éléments restants de M de sorte que M prenne sa forme initiale
  - Revenir à l'étape 1

De même que pour les coupes basées sur des chaînes, on cherche à ajouter une coupe tant qu'il reste des clients qu'on n'a pas encore étudiés. On essaye ainsi d'exploiter au maximum l'information disponible à chaque nœud de branchement de l'algorithme L-shaped en nombres entiers. Encore une fois, les conditions $P + P(EC) > 0.01$ et $P + P(EC) > \theta + 1$ ont été choisies de façon empirique.

L'algorithme ci-dessus présente de nombreux avantages dont on a déjà discuté, mais possède la limite de ne s'intéresser qu'aux chaînes de la solution courante dont les clients sont vus comme des ensembles non structurés de nœuds. Seule la forme entière du flot traversant ces ensembles de clients est donc étudiée. Cependant, on observe expérimentalement que peu de coupes peuvent être identifiées. Ceci est principalement dû au fait que la condition $P > \theta + 1$ est difficilement remplie à cause de la relative faiblesse de la borne apportée.

En réalité, il existe une multitude de flots fractionnaires visitant des ensembles non structurés de clients dont la valeur est suffisante pour qu'une coupe soit active. Un exemple a été explicité dans la figure 4.6. Si de tels flots sont identifiés, on espère pouvoir trouver davantage de coupes que l'algorithme de séparation ci-dessus. Cela va permettre d'augmenter

plus rapidement la valeur de la borne inférieure globale de l'algorithme L-shaped en nombres entiers sur la valeur optimale du problème.

On va donc diversifier notre recherche de coupes et étendre les éléments considérés aux cliques dont le flot total parcourant ses arêtes est égal au nombre de ses clients moins un. Les clients d'une chaîne font donc partie de ce cas de figure, mais il existe plusieurs autres configurations.

Une première phase consiste à identifier de façon séparée les chaînes de la solution et les autres éléments non structurés de nœuds. Ces autres éléments seront obtenus en utilisant l'algorithme exact de séparation de Jabali *et al.* (2014). Cet algorithme fournit les ensembles UVS (Unstructured Vertex Sets) de la solution courante. Ils servent à la construction des routes partielles utilisées dans leurs travaux et sont définis de la manière suivante : soit $S = \{v_{1(S)}, \ldots, v_{t_S(S)}\}$ l'ensemble de ses clients, alors il existe au moins une arête de $E(S)$ ayant un flot fractionnaire, et $\sum_{v_i, v_j \in S} x_{ij} = t_S - 1$.

Un ensemble UVS seul peut donc être un parfait candidat pour les éléments non structurés de nœuds recherchés dans nos travaux. Néanmoins, il est important de pouvoir identifier des ensembles les plus grands possible pour la génération de coupes violées. On peut donc également considérer les liens qu'il peut y avoir entre les chaînes et les ensembles UVS, comme le font Jabali *et al.* (2014) pour construire leurs routes partielles. Quand il existe un tel lien entre une chaîne et un ensemble UVS S au niveau d'un client v_i, faisant donc partie de la chaîne, on dit d'un tel client qu'il est un nœud d'articulation. Ce terme, emprunté de la théorie des graphes, signifie que supprimer ce nœud et les arêtes qui lui sont liées sépare le graphe initial en deux ensembles connexes. On a également la relation $\sum_{v_j \in S} x_{ij} = 1$, assurant que le flot circulant dans la chaîne est entièrement conservé dans S au niveau du nœud d'articulation v_i.

Les nœuds des séquences alternées de chaînes et d'ensembles UVS présentes dans une solution courante constituent un ensemble non structuré pouvant donner une coupe violée par la solution courante. En effet, pour qu'un tel ensemble de t clients fournisse une coupe violée, il faut que la somme totale des flots des arêtes qui lient tous ses clients soit égale à $t-1$, autrement dit le nombre de clients moins un. C'est le cas pour les chaînes prises séparément, de même que pour les ensembles UVS. Cela reste le cas pour l'ensemble de la séquence puisque la somme du flot des arêtes liant les nœuds d'articulation aux ensembles UVS vaut 1.

Explicitons maintenant le deuxième algorithme de séparation heuristique utilisé, soit :

- (x, θ) la solution courante
- P la valeur de la borne apportée par la coupe courante
- M l'ensemble des H séquences alternées de chaînes et d'ensembles UVS liés de la solution x. Les éléments $v_{i(h)}$ de la ligne h de M sont soit des clients, soit des ensembles UVS composés de plusieurs clients. Chaque séquence h est composée de t_h éléments :

$$M = \begin{Bmatrix} \{v_{1(1)}, & v_{2(1)}, & \cdots & , v_{t_1(1)}\} \\ \vdots & & & \vdots \\ \{v_{1(H)}, & v_{2(H)}, & \cdots & , v_{t_H(H)}\} \end{Bmatrix}$$

- h l'indice de la séquence sur laquelle on travaille
- EC l'ensemble non structuré de noeuds courant que l'on essaye d'ajouter à la coupe
- A l'ensemble des variables contenues dans la coupe courante
- $P(EC)$ la borne apportée par l'ensemble courant EC, le calcul se fait en utilisant la formule (4.25)

1. Initialisation

- $P := 0$
- $A := \varnothing$
- $h := 1$
- $EC := \{v_{1(1)}\}$

2. Apport du prochain élément

- Supprimer le premier élément de la séquence h
- Si la séquence h est vide
  - Aller à l'étape 3
- Sinon
  - Si le premier élément de la séquence h est un ensemble UVS
    - Ajouter tous les clients de l'ensemble UVS à l'ensemble EC
  - Sinon
    - Ajouter le premier client de la séquence h à l'ensemble EC
  - Calculer $P(EC)$
  - Si $P(EC) + P > \theta + 1$
    - Aller à l'étape 4
  - Sinon
    - Revenir à l'étape 2

3. Fin de la séquence h

- Calculer $P(EC)$
- Si $P(EC) > 0.01$
  - Considérer EC comme une clique et mettre les variables représentant ses arêtes dans A
- $h := h + 1$
- Si $h \leq H$
  - Ré-initialiser $EC := \{\}$
  - Si $v_{1(h)}$ est un ensemble UVS
    - Ajouter tous les clients de l'ensemble UVS à l'ensemble EC
  - Sinon
    - $EC := \{v_{1(h)}\}$
  - Aller à l'étape 2
- Sinon, arrêter

4. Ajout de la coupe

- $P := P + P(EC)$
- Considérer EC comme une clique et mettre les variables représentant ses arêtes dans A
- Ajouter la coupe contenant les variables de A et la borne P au modèle en respectant l'expression (4.28)
- Si M est vide
  - Arrêter
- Sinon
  - Renommer les éléments restants de M de sorte que M prenne sa forme initiale
  - Revenir à l'étape 1

Chaque fois qu'on rencontre un élément qui est un ensemble UVS, on est obligé d'ajouter tous ses clients à l'ensemble courant EC. Dans le cas contraire, on ne serait pas assuré que la coupe ajoutée soit violée par la solution courante, bien qu'elle soit valide pour le modèle. En effet, pour un ensemble UVS composé de tous ses clients, on est sûr que le flot total des arêtes les liant est égal à ce nombre de clients moins un. Ce n'est plus forcément le cas pour un sous-ensemble de clients de l'ensemble UVS.

En revanche, on peut toujours tester l'ajout des clients des chaînes un à un comme on le faisait déjà pour les algorithmes de séparation précédent.

CHAPITRE 5

ANALYSE DES RÉSULTATS

Nous présentons dans ce chapitre les performances pratiques des développements théoriques que l'on a décrits. On précise dans un premier temps les paramètres généraux utilisés pour les différentes simulations. Puis, on expose les temps de calcul obtenus grâce à l'utilisation de nos différentes coupes, le nombre d'instances pouvant être résolues, le gap d'optimalité pour les instances non résolues, le nombre de nœuds de branchement et de coupes nécessaires à la résolution des problèmes et enfin le nombre d'instances résolues pour différents gaps.

5.1 Paramètres généraux

Le code informatique de l'algorithme L-shaped en nombres entiers intégrant les nouvelles coupes LBF de ce mémoire a été écrit en C++ avec la librairie de CPLEX 12.3. Tous les résultats présentés ci-dessous ont été obtenus grâce à des ordinateurs du GERAD. Chacun d'eux possède deux processeurs Intel(R) Xeon(R) X5675 à 3.07GHz et 96 Go de mémoire vive. Ces mêmes machines ont été utilisées par Jabali *et al.* (2014), mais nous avons refait les simulations afin de comparer dans des conditions semblables les performances de leurs travaux avec l'apport de ce mémoire.

Nous avons utilisé la base du code de Jabali *et al.* (2014), dans lequel la procédure BC a été implémentée à l'aide de la librairie OOBB développée au CIRRELT. La séparation des contraintes de capacité et d'élimination de sous-tours (3.11) a été réalisée grâce à la librairie CVRPSEP de Lysgaard *et al.* (2004).

Les paramètres suivants ont été choisis pour réaliser l'ensemble des calculs pour lesquels nous présenterons les résultats obtenus dans les sections suivantes :

- Le temps de calcul maximum a été fixé à 10000 secondes. Au-delà de cette limite, le calcul est arrêté.
- La stratégie d'exploration *meilleur d'abord* de l'arbre d'énumération a été employée pour sélectionner le prochain nœud à évaluer.
- Les branchements se font sur les variables fractionnaires associées aux arêtes les plus courtes, c'est-à-dire celles dont la distance entre les deux nœuds est la plus petite.

- Le gap à partir duquel on considère que l'instance est résolue à l'optimalité est $0,0001\%$

Les instances testées sont celles utilisées dans la littérature. Elles sont tirées des travaux de Jabali *et al.* (2014) et avaient déjà été utilisées par Laporte *et al.* (2002). La localisation des n nœuds du graphe a été générée aléatoirement dans une surface $[0,100]^2$ suivant une distribution uniforme. Cinq obstacles sur lesquelles les clients ne peuvent pas se situer sont placés dans la surface $[20,80]^2$, chacun d'eux ayant une largeur de 4 et une hauteur de 25, recouvrant ainsi 5% de la surface totale.

Les demandes ξ_i suivent une loi normale $\mathcal{N}(\mu_i, \sigma_i^2)$ tronquée en 0 et en D, la capacité des véhicules. Pour chaque instance, on définit le coefficient de remplissage moyen des véhicules par :

$$\bar{f} = \frac{\sum_{i=2}^{n} \mu_i}{mD} \tag{5.1}$$

Il donne une approximation du rapport entre la quantité de produit livrée par rapport à la capacité des différents véhicules.

Les tailles de problème que l'on a considérées sont résumées dans le tableau 5.1. Pour chaque taille, on considère plusieurs variations du paramètre $\bar{f}$:

Tableau 5.1 Instances testées

m	n	$\bar{f}$
2	60, 70, 80, 90	90%, 95%
3	40, 45	80%
3	50, 60, 70, 80	80%, 85%, 90%, 95%
4	40, 50	75%, 80%, 85%, 90%, 95%
4	45	80%, 85%, 90%, 95%
4	60	80%, 82%, 85%

On dispose de 481 instances en tout.

Les tableaux de résultats que l'on présentera ci-dessous compareront les performances de différentes configurations de l'algorithme L-shaped en nombres entiers appliqué au VRPSD qui font appel à différents types de coupes LBF. Chaque type sera désigné par un sigle :

- LBF_1, LBF_2, LBF_3 : les trois types de coupes LBF de Jabali *et al.* (2014) basées sur les routes partielles. Les LBF_3 étaient les plus efficaces.

- LBF_a : les coupes LBF de ce mémoire basées sur l'identification de chaînes.
- LBF_b : les coupes LBF de ce mémoire basées sur l'identification d'ensembles non structurés de nœuds. Le premier algorithme de séparation présenté a été utilisé, on étudie donc uniquement les chaînes des solutions intermédiaires pour ajouter une coupe.
- LBF_c : de mêmes que LBF_b mais en utilisant le deuxième algorithme de séparation, qui étend l'étude des chaînes à celle des ensembles UVS des solutions intermédiaires.

Les formes de routes partielles recherchées dans Jabali *et al.* (2014) sont présentées dans les figures 5.1 à 5.3.

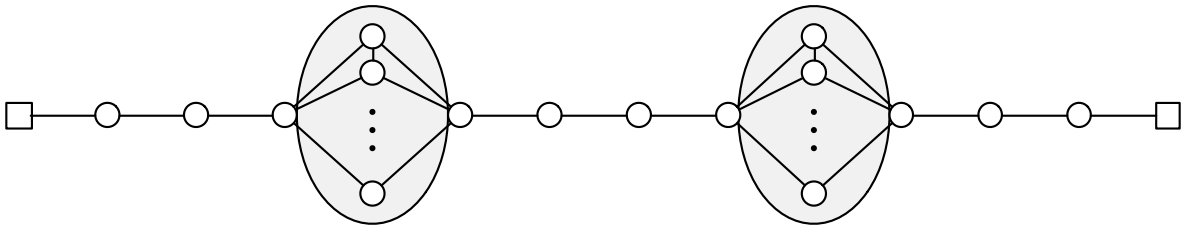

Figure 5.1 LBF_1

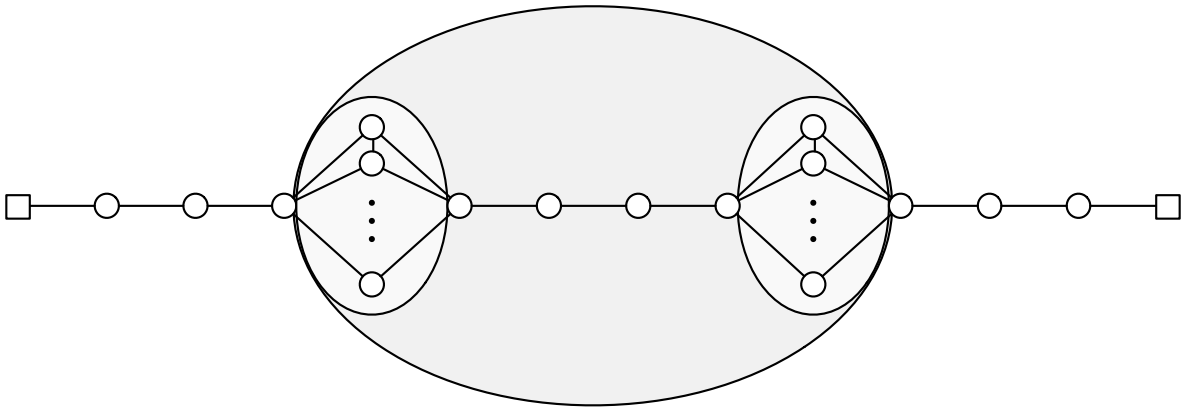

Figure 5.2 LBF_2

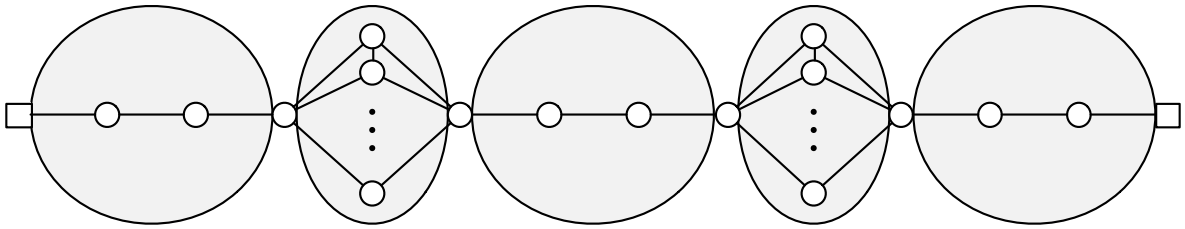

Figure 5.3 LBF_3

Chacune des formes de coupes LBF basées sur les routes partielles présentées ici sont de moins en moins spécifiques et touchent de plus en plus de solutions. En contrepartie, la borne associée est donc de moins en moins élevée. Les coupes LBF_1 considèrent chaque élément comme ce qu'il est vraiment, c'est-à-dire soit une chaîne, soit un ensemble UVS. Les coupes LBF_2 considèrent que la route partielle est composée d'une séquence chaîne-UVS-chaîne. Les deux chaînes sont liées au dépôt, peuvent éventuellement être vides et tous les autres nœuds sont insérés dans un ensemble UVS central. Dans les coupes LBF_3, chaque élément, qu'il soit une chaîne ou un ensemble UVS, est vu comme un ensemble UVS. Les éléments sont reliés entre eux par des nœuds d'articulation.

5.2 Amélioration des temps de calcul

Le tableau 5.2 compare les temps de résolution de l'algorithme L-shaped en nombres entiers couplé avec différentes coupes LBF. Chaque instance présentée dans le tableau a été résolue à l'optimalité par au moins un type de coupe. Le temps affecté aux instances non résolues est de 10000 secondes. Avant ce mémoire, les meilleures performances venaient des coupes LBF_3 pour lesquelles les temps de calcul de l'algorithme étaient en moyenne de 2615,02 secondes pour l'ensemble de ces instances. On remarque donc que l'utilisation des coupes LBF_b et LBF_c seules améliore déjà les performances puisqu'ils résolvent ces mêmes instances avec un temps moyen de 2268,91 secondes et 2253,39 secondes. Ceci constitue une amélioration de $13,24\%$ et $13,83\%$, respectivement, en terme de réduction de temps de calcul. On constate néanmoins qu'il n'y a pas d'amélioration notable entre les coupes LBF_c et LBF_b. Nous avions émis l'hypothèse qu'en modifiant l'algorithme de séparation des coupes LBF_b de sorte à élargir les possibilités d'ajouter une coupe violée à chaque nœud de l'arbre d'énumération, nous pourrions ajouter des coupes plus tôt dans l'algorithme et ainsi accélérer le processus de résolution. Ceci n'est en fait pas le cas en moyenne. En revanche, il est intéressant de disposer de ces deux configurations puisqu'il peut y avoir des écarts im-

portants pour certaines tailles d'instances. Par exemple, pour les instances comportant 60 clients avec 2 véhicules et un coefficient de remplissage de 90%, les coupes LBF_c permettent une diminution des temps de calcul de l'algorithme de $34,1\%$ par rapport aux coupes LBF_b. La situation inverse se produit pour les instances comportant 40 clients avec 4 véhicules et $\bar{f} = 85\%$, pour lesquelles les coupes LBF_b permettent de réduire les temps de résolution de $24,21\%$ par rapport aux coupes LBF_c.

Les meilleures performances viennent de la combinaison des coupes LBF_c et LBF_a, qui ont des résultats semblables à la combinaison LBF_b et LBF_a. La moyenne de temps de résolution obtenue est de $1922,49$ ce qui réduit de $26,5\%$ les temps de calcul de l'algorithme avec les coupes LBF_3, de $45,2\%$ ceux des coupes LBF_2 et enfin de $52,6\%$ ceux des coupes LBF_1. Les temps de calcul de l'algorithme avec les coupes LBF_a seules n'ont pas été présentés, car ils n'étaient pas satisfaisants, bien qu'ils soient meilleurs que ceux des coupes LBF_1 et LBF_2. En revanche, leur combinaison avec les coupes LBF_b et LBF_c est particulièrement efficace. Ces coupes sont restrictives sur le nombre de solutions actives, mais compensent la faiblesse principale des coupes LBF_b et LBF_c : leur borne trop peu élevée pour fournir des coupes violées fréquemment. Les coupes développées dans ce mémoire sont donc complémentaires.

On observe enfin qu'il n'existe pas de tendance particulière d'amélioration en fonction de la taille des problèmes. On ne peut pas prévoir si les performances de la combinaison LBF_c et LBF_a seront d'autant plus élevées que la taille des problèmes augmente. On peut observer cela à travers les figures 5.4 et 5.5 qui présentent l'évolution du pourcentage d'amélioration des temps de calcul en fonction de la taille des problèmes. Les temps comparés sont ceux de la combinaison de coupes LBF_c et LBF_a par rapport aux coupes LBF_3. On a fait une première agrégation des résultats pour observer uniquement l'évolution en fonction du nombre de véhicules, on fait de même pour le nombre de clients.

Tableau 5.2 Temps de calcul (en secondes)

m	**n**	$\bar{f}$	$\mathbf{LBF}_1$	$\mathbf{LBF}_2$	$\mathbf{LBF}_b$	$\mathbf{LBF}_c$	$\mathbf{LBF}_{a+b}$	$\mathbf{LBF}_{a+c}$	$\mathbf{LBF}_3$	**instances**
2	**60**	**0.9**	6528.5	5200.9	1444.5	951.9	457.32	584.80	2312.6	10
2	**60**	**0.95**	4014	3795.3	3279.9	3575.7	1053.85	989.39	1369.7	4
2	**70**	**0.9**	3251.9	1746.4	1275.5	1244.3	1278.12	1296.79	1147.3	10
2	**70**	**0.95**	9596.4	5713.1	5874.5	5061.9	4724.48	4368.42	4600.1	1
2	**80**	**0.9**	6088.8	4378.3	2498.6	2468.8	2642.66	2445.81	2515.3	10
2	**90**	**0.9**	8411.7	7618.8	3543.9	3167.7	4094.49	4226.69	4328.0	8
3	**40**	**0.8**	1473.2	1506.75	1069.5	1076.7	526.47	707.66	1339.3	10
3	**45**	**0.8**	1146.8	1151.5	13.3	12.5	11.26	10.48	1106.6	10
3	**50**	**0.8**	1642.4	949.2	1715.3	1786.2	620.27	997.27	746.7	18
3	**50**	**0.85**	2592.5	2581.1	3347.7	2872.6	3351.19	2574.66	3361.0	6
3	**50**	**0.9**	1596.9	824.0	10000	10000	10000	10000	3355.2	1
3	**60**	**0.8**	2432.8	2427.9	469.1	427.8	398.45	462.87	1855.4	17
3	**60**	**0.85**	3691.6	3347.1	1927.3	1599.6	998.65	958.41	2396.5	5
3	**60**	**0.9**	4833.2	4520.4	5102.2	5437.1	4552.03	4522.35	6582.3	6
3	**70**	**0.8**	4350.9	3212.9	394.0	428.0	433.46	450.27	2980.9	7
3	**70**	**0.85**	2770.7	2227.0	1979.5	2556.5	1763.53	1758.01	1225.5	8
3	**80**	**0.8**	6196.1	5944.5	2713.3	2676.4	2803.42	2925.78	2894.7	12
3	**80**	**0.85**	6075.2	5102.2	3574.9	4068.0	3107.42	3377.74	3088.1	5
4	**40**	**0.75**	159.8	160.0	435.4	461.8	448.74	432.31	121.6	3
4	**40**	**0.8**	2563.3	2582.8	462.2	596.6	487.13	420.19	2120.64	4
4	**40**	**0.85**	5219.2	5214.0	2086.0	2752.3	1771.54	1872.64	4949.3	4
4	**45**	**0.8**	10000	10000	10000	10000	10000	7155.78	10000	1
4	**45**	**0.85**	6653.3	6337.7	4765.3	4545.9	4126.37	3664.75	5862.0	7
4	**50**	**0.75**	3962.0	3382.4	4885.9	4993.9	4191.63	4201.94	3189.2	8
4	**50**	**0.8**	4681.3	4635.8	5177.7	5170.7	4748.18	3940.21	3388.3	4
4	**50**	**0.85**	10000	10000	8555.3	6235.0	10000	9866.35	10000.5	1
4	**60**	**0.82**	7381.1	7429.9	1142.5	1155.5	1328.27	1430.93	6687.3	3
		Moyenne	**4051.73**	**3504.29**	**2268.91**	**2253.39**	**1930.62**	**1922.49**	**2615.02**	**183**

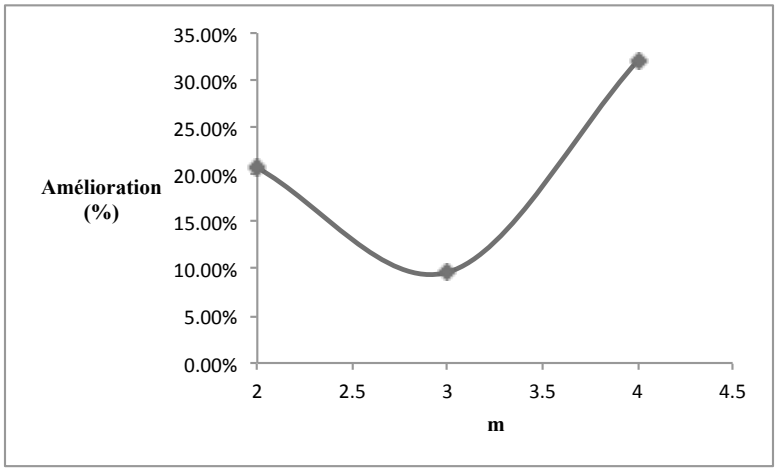

Figure 5.4 Pourcentage d'amélioration des temps de calcul en fonction du nombre de véhicules

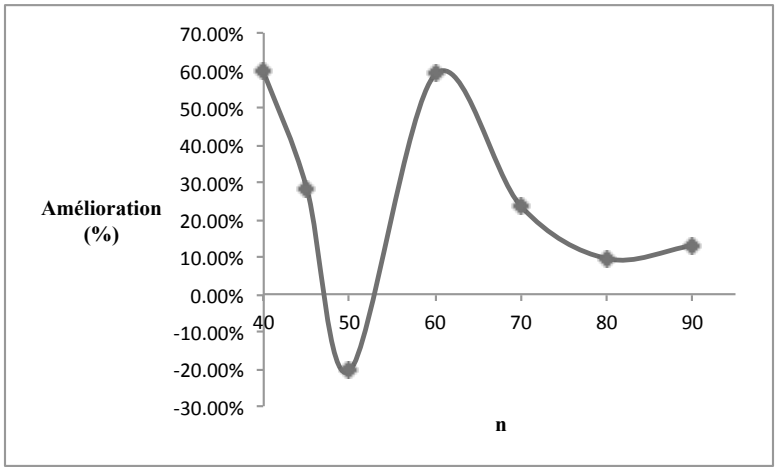

Figure 5.5 Pourcentage d'amélioration des temps de calcul en fonction du nombre de clients

5.3 Nombre d'instances résolues

Le tableau 5.3 présente le nombre d'instances résolues à l'optimalité par rapport au nombre d'instances testées pour les différents paramètres. Sur les 481 instances, l'algorithme utilisant les coupes LBF_3 de Jabali *et al.* (2014) parvient à résoudre de façon optimale 158 instances. De nouveau, on observe que les coupes LBF_b et LBF_c utilisées seules permettent à l'algorithme d'améliorer ce chiffre en résolvant respectivement 160 et 162 instances.

Tableau 5.3 Instances résolues

m	**n**	$\bar{\mathbf{f}}$	LBF_1	LBF_2	LBF_b	LBF_c	LBF_{a+b}	LBF_{a+c}	LBF_3	instances
2	**60**	**0.9**	4	6	9	10	10	10	10	10
2	**60**	**0.95**	3	3	4	4	4	4	4	10
2	**70**	**0.9**	8	9	9	9	9	9	10	10
2	**70**	**0.95**	1	1	1	1	1	1	1	10
2	**80**	**0.9**	5	7	9	9	8	9	10	10
2	**80**	**0.95**	0	0	0	0	0	0	0	10
2	**90**	**0.9**	2	3	8	8	7	7	7	10
2	**90**	**0.95**	0	0	0	0	0	0	0	10
3	**40**	**0.8**	9	9	9	9	10	10	9	10
3	**45**	**0.8**	9	9	10	10	10	10	9	10
3	**50**	**0.8**	16	17	16	16	17	18	18	20
3	**50**	**0.85**	5	5	4	5	4	5	4	9
3	**50**	**0.9**	1	1	0	0	0	0	1	10
3	**50**	**0.95**	0	0	0	0	0	0	0	16
3	**60**	**0.8**	13	13	17	17	17	17	14	20
3	**60**	**0.85**	4	4	5	5	5	5	4	8
3	**60**	**0.9**	4	4	4	4	4	5	4	18
3	**60**	**0.95**	0	0	0	0	0	0	0	10
3	**70**	**0.8**	4	5	7	7	7	7	5	10
3	**70**	**0.85**	7	7	7	7	7	7	7	9
3	**70**	**0.9**	0	0	0	0	0	0	0	9
3	**70**	**0.95**	0	0	0	0	0	0	0	19
3	**80**	**0.8**	6	7	10	10	10	10	11	19
3	**80**	**0.85**	3	3	4	3	5	5	4	10
3	**80**	**0.9**	0	0	0	0	0	0	0	10
3	**80**	**0.95**	0	0	0	0	0	0	0	9
4	**40**	**0.75**	3	3	3	3	3	3	3	8
4	**40**	**0.8**	3	3	4	4	4	4	4	10
4	**40**	**0.85**	2	3	4	4	4	4	3	13
4	**40**	**0.9**	0	0	0	0	0	0	0	15
4	**40**	**0.95**	0	0	0	0	0	0	0	10
4	**45**	**0.8**	0	0	0	0	0	1	0	8
4	**45**	**0.85**	4	4	5	5	6	7	5	10
4	**45**	**0.9**	0	0	0	0	0	0	0	12
4	**45**	**0.95**	0	0	0	0	0	0	0	7
4	**50**	**0.75**	5	6	5	5	5	5	7	9
4	**50**	**0.8**	3	3	2	3	3	4	3	7
4	**50**	**0.85**	0	0	1	1	1	1	0	9
4	**50**	**0.9**	0	0	0	0	0	0	0	18
4	**50**	**0.95**	0	0	0	0	0	0	0	10
4	**60**	**0.8**	0	0	0	0	0	0	0	9
4	**60**	**0.82**	1	1	3	3	3	3	1	10
4	**60**	**0.85**	0	0	0	0	0	0	0	10
		Total	**125**	**136**	**160**	**162**	**164**	**171**	**158**	**481**

Les meilleurs résultats sont ceux obtenus par la combinaison de coupes LBF_c et LBF_a, qui permettent d'obtenir une solution optimale pour 171 instances dans le temps imparti. La taille des instances résolues a pu être augmentée grâce aux coupes de ce mémoire. En effet, on a pu résoudre des instances dont la taille était telle qu'elles n'avaient jamais été résolues dans le temps imparti. C'est le cas pour les instances de 45 clients avec 4 véhicules et un coefficient de remplissage de 80% mais aussi pour celles comportant 50 clients avec 4 véhicules et un coefficient de remplissage de 85%.

5.4 Gap d'optimalité des instances non résolues

Le tableau 5.4 présente les valeurs des gaps d'optimalité pour les instances qui n'ont été résolues par aucune des configurations de coupes. En notant $\bar{z}$ la valeur de l'objectif de la meilleure solution entière connue et $\underline{z}$ la borne inférieure courante du problème, alors ce gap vaut : $\frac{\bar{z}-\underline{z}}{\underline{z}}$. De plus, une instance est prise en compte dans ce tableau si pour chacune des configurations, l'algorithme a eu le temps de trouver au moins une solution entière, de manière à ce qu'on ne prenne pas en compte les gaps infinis.

On remarque une nouvelle fois que les coupes de ce mémoire améliorent les performances de l'algorithme L-shaped en nombres entiers puisqu'on obtient des gaps plus faibles que pour les coupes LBF_3 qui est de $1,86\%$ en moyenne. Cette fois, ce sont les coupes LBF_b et LBF_c qui apportent les meilleurs résultats avec un gap de $1,51\%$ chacun. Ceci indique que si les combinaisons de coupes LBF_a avec les coupes LBF_b et LBF_c résolvent à l'optimalité le plus grand nombre d'instances, la variabilité de la distribution des gaps est plus élevée que pour les coupes LBF_b et LBF_c seules. Ces combinaisons sont tout de même satisfaisantes, les gaps étant respectivement de $1,65\%$ et $1,66\%$.

5.5 Nombre de coupes et de nœuds de branchement nécessaires à la résolution

Le tableau 5.5 permet d'observer le nombre de nœuds de branchement nécessaires pour atteindre la solution optimale. Chaque instance présentée a été résolue à l'optimalité, d'où leur nombre réduit par rapport aux autres tableaux de résultats. Il ressort clairement que l'utilisation de nos coupes permet de réduire le nombre de nœuds de branchement ce qui diminue le nombre d'opérations à effectuer pour terminer la résolution.

De plus, comme le montre le tableau 5.6, pour les coupes LBF_b et LBF_c seules, un nombre

Tableau 5.4 Gap des instances non résolues (en %)

m	**n**	$\bar{\mathbf{f}}$	LBF_1	LBF_2	LBF_b	LBF_c	LBF_{a+b}	LBF_{a+c}	LBF_3	instances
2	**60**	**0.95**	1.22	1.00	0.54	0.54	0.34	0.34	0.45	6
2	**70**	**0.95**	1.53	1.35	0.73	0.73	0.68	0.67	0.65	9
2	**80**	**0.95**	1.26	1.03	0.78	0.82	0.77	0.79	0.88	10
2	**90**	**0.9**	1.09	1.04	0.40	0.40	0.47	0.45	1.39	2
2	**90**	**0.95**	1.60	1.35	0.67	0.67	0.91	0.90	0.72	8
3	**50**	**0.8**	0.18	0.19	0.43	0.44	0.28	0.20	0.23	2
3	**50**	**0.85**	1.39	0.78	0.35	0.38	0.29	0.28	0.15	3
3	**50**	**0.9**	0.90	0.78	0.81	0.80	0.71	0.75	0.66	8
3	**50**	**0.95**	1.77	1.74	1.73	1.74	1.45	1.43	1.68	16
3	**60**	**0.8**	1.65	1.33	1.14	1.04	0.83	0.95	1.38	3
3	**60**	**0.85**	0.98	0.72	0.55	0.58	0.79	0.78	0.52	3
3	**60**	**0.9**	1.38	1.35	0.99	1.05	1.11	1.08	1.53	12
3	**60**	**0.95**	2.23	2.17	1.87	1.89	1.91	1.92	1.98	10
3	**70**	**0.8**	1.22	1.20	1.04	1.21	1.42	1.29	1.11	3
3	**70**	**0.9**	1.62	1.87	1.79	1.35	1.69	2.02	2.29	5
3	**70**	**0.95**	2.39	2.17	1.50	1.51	1.98	1.93	1.63	18
3	**80**	**0.8**	0.73	0.67	0.47	0.47	0.58	0.57	0.91	5
3	**80**	**0.85**	0.67	0.62	0.33	0.34	0.34	0.40	1.35	5
3	**80**	**0.9**	1.38	2.12	0.93	0.94	0.95	0.93	1.06	7
3	**80**	**0.95**	1.69	1.64	1.64	1.63	1.63	1.62	1.54	6
4	**40**	**0.75**	1.40	1.23	3.33	2.72	3.76	3.54	1.21	5
4	**40**	**0.8**	1.77	1.80	1.06	1.05	1.59	1.34	1.65	6
4	**40**	**0.85**	2.25	2.42	1.86	1.78	1.82	2.14	4.97	8
4	**40**	**0.9**	3.14	3.21	2.29	2.26	2.38	2.42	3.17	9
4	**40**	**0.95**	4.06	4.05	3.52	3.53	3.67	3.77	3.73	10
4	**45**	**0.8**	2.74	2.52	2.22	2.37	2.98	2.85	2.14	7
4	**45**	**0.85**	2.13	2.11	1.05	1.06	1.50	1.71	2.14	3
4	**45**	**0.9**	2.81	2.62	2.13	2.14	2.53	2.37	3.72	7
4	**45**	**0.95**	2.97	2.95	2.66	2.65	2.56	2.55	4.02	7
4	**50**	**0.75**	1.60	1.67	5.40	5.54	2.63	2.65	1.95	1
4	**50**	**0.8**	2.29	2.27	1.13	1.50	4.95	5.22	1.88	3
4	**50**	**0.85**	2.01	2.28	0.50	0.50	0.47	0.50	0.76	4
4	**50**	**0.9**	3.21	3.23	1.71	1.90	2.08	1.97	3.25	6
4	**50**	**0.95**	3.07	3.07	3.07	3.07	3.12	3.16	3.16	10
4	**60**	**0.8**	1.47	1.44	1.14	1.27	1.19	1.26	1.29	8
4	**60**	**0.82**	0.95	0.94	0.65	0.66	0.68	0.79	1.07	4
4	**60**	**0.85**	1.55	1.57	0.76	0.77	1.23	0.90	0.98	4
		Moyenne	**1.96**	**1.91**	**1.51**	**1.51**	**1.65**	**1.66**	**1.86**	**243**

restreint de coupes suffit maintenant à atteindre l'optimalité, si l'on compare ce chiffre avec ceux des anciennes coupes LBF. Ainsi, on profite de deux phénomèmes qui accélèrent la

résolution : d'une part on a besoin de moins de nœuds de branchement pour résoudre les instances, et d'autre part, la résolution des sous-problèmes de chaque nœud se fait plus rapidement car le modèle est moins encombré par des contraintes de type LBF.

On observe également que le peu de coupes LBF_b et LBF_c peut être compensé efficacement par l'apport complémentaire des LBF_a, puisque ces dernières sont générées en grande quantité pour les combinaisons de coupes LBF_{a+b} et LBF_{a+c}.

Tableau 5.5 Nombre de nœuds de branchement

m	**n**	$\bar{\mathbf{f}}$	LBF_1	LBF_2	LBF_b	LBF_c	LBF_{a+b}	LBF_{a+c}	LBF_3	instances
2	**60**	**0.9**	5702	3184	2387	2450	1637	1655	1372	4
2	**60**	**0.95**	10490	9694	9847	9561	8680	8244	9806	3
2	**70**	**0.9**	3119	2985	3024	2594	3740	3552	2331	7
2	**70**	**0.95**	11251	9919	22608	18841	18427	17497	19563	1
2	**80**	**0.9**	2234	1811	1590	1627	1739	1744	1402	4
2	**90**	**0.9**	4076	3274	1602	1690	2040	1930	1517	2
3	**40**	**0.8**	7174	6904	3168	3244	3361	3286	6124	9
3	**45**	**0.8**	4036	3989	920	885	1000	902	3484	9
3	**50**	**0.8**	7397	5953	2253	2223	2617	2411	2380	15
3	**50**	**0.85**	3016	2922	1104	1312	1481	1387	2027	4
3	**60**	**0.8**	2945	2703	1705	1766	1602	1619	2129	13
3	**60**	**0.85**	3459	3036	1430	1648	1930	1730	2694	3
3	**60**	**0.9**	5290	5659	3748	3747	5687	5282	7715	2
3	**70**	**0.8**	2130	2002	1020	1041	1024	976	1279	4
3	**70**	**0.85**	8674	8346	6505	7340	6636	5971	7202	7
3	**80**	**0.8**	5420	5498	2371	2201	2785	2612	4181	4
3	**80**	**0.85**	7200	6416	2250	2096	3145	2879	2728	3
4	**40**	**0.75**	5021	5062	4971	4957	5101	4955	4614	3
4	**40**	**0.8**	5476	6058	2108	2218	2410	2019	5354	3
4	**40**	**0.85**	21653	22002	7565	7114	6836	9047	15318	2
4	**45**	**0.85**	21679	21685	10410	9864	9638	9477	19246	3
4	**50**	**0.75**	6570	5725	5449	5393	5069	4963	8208	4
4	**50**	**0.8**	15197	15314	5798	5708	5304	5530	13531	2
4	**60**	**0.82**	20694	20694	2495	2343	3099	2115	3547	1
		Moyenne	**6405**	**5940**	**3310**	**3284**	**3401**	**3268**	**4670**	**112**

5.6 Nombre d'instances résolues pour différents gaps

Le tableau 5.7 présente le pourcentage d'instances dont la résolution atteint différents gaps. Par exemple, les coupes LBF_c permettent à $98,13\%$ des instances d'atteindre un gap inférieur à 5% au bout de 10000 secondes. Les coupes de ce mémoire sont meilleures que les

Tableau 5.6 Nombre de coupes ajoutées

m	**n**	$\bar{\mathbf{f}}$	LBF_1	LBF_2	LBF_b	LBF_c	LBF_a+LBF_b	LBF_a+LBF_c	LBF_3	instances
2	**60**	**0.9**	2753	1706	5	7	206 + 4	218 + 7	59	4
2	**60**	**0.95**	1690	1123	1	2	717 + 2	654 + 3	17	3
2	**70**	**0.9**	2632	1801	10	8	234 + 10	212 + 11	55	7
2	**70**	**0.95**	7354	5742	1	2	1587 + 5	1556 + 4	7	1
2	**80**	**0.9**	1992	924	2	3	133 + 3	104 + 6	34	4
2	**90**	**0.9**	6290	3321	3	4	135 + 4	122 + 5	97	2
3	**40**	**0.8**	110	49	25	25	75 + 18	72 + 20	16	9
3	**45**	**0.8**	139	149	23	22	65 + 17	55 + 21	25	9
3	**50**	**0.8**	305	260	9	9	136 + 11	120 + 12	57	15
3	**50**	**0.85**	1262	1023	29	30	116 + 24	113 + 34	106	4
3	**60**	**0.8**	353	280	10	10	28 + 8	23 + 9	56	13
3	**60**	**0.85**	797	674	17	22	112 + 14	112 + 22	105	3
3	**60**	**0.9**	292	179	8	7	141 + 12	151 + 14	95	2
3	**70**	**0.8**	268	203	5	6	29 + 5	18 + 7	11	4
3	**70**	**0.85**	2878	2101	25	30	171 + 20	162 + 30	240	7
3	**80**	**0.8**	639	561	5	7	66 + 6	57 + 7	86	4
3	**80**	**0.85**	3369	2299	16	14	135 + 12	129 + 15	116	3
4	**40**	**0.75**	274	260	39	42	150 + 34	118 + 30	73	3
4	**40**	**0.8**	201	241	27	27	230 + 26	202 + 29	30	3
4	**40**	**0.85**	247	247	16	14	84 + 4	56 + 10	395	2
4	**45**	**0.85**	2518	2349	130	145	219 + 89	203 + 107	324	3
4	**50**	**0.75**	866	761	47	51	129 + 35	101 + 37	92	4
4	**50**	**0.8**	416	406	45	48	70 + 27	70 + 38	91	2
4	**60**	**0.82**	84	84	22	15	58 + 8	46 + 14	17	1
		Moyenne	**1157**	**824**	**19**	**21**	**147+15**	**133+19**	**80**	**112**

coupes LBF_3 de Jabali *et al.* (2014) pour toutes les limites de gap. On remarque de nouveau que si les combinaisons de coupes permettent de résoudre le plus d'instances de façon optimale, ce sont les coupes LBF_b et LBF_c utilisées seules qui induisent les meilleurs pourcentages de réussite pour les différentes limites de gaps.

Tableau 5.7 Nombre d'instances résolues pour différents gaps

	$\mathbf{LBF_1}$	$\mathbf{LBF_2}$	$\mathbf{LBF_b}$	$\mathbf{LBF_c}$	$\mathbf{LBF_{a+b}}$	$\mathbf{LBF_{a+c}}$	$\mathbf{LBF_3}$
$= 0\%$	25.99%	28.27%	33.26%	33.68%	34.10%	35.55%	32.85%
$\leq 1\%$	50.94%	53.01%	61.54%	63.20%	58.42%	58.63%	56.96%
$\leq 3\%$	80.87%	81.70%	88.98%	89.19%	86.69%	86.28%	79.63%
$\leq 5\%$	94.39%	94.80%	97.71%	98.13%	95.43%	94.80%	86.90%
$\leq 7\%$	98.13%	98.54%	98.75%	99.17%	97.92%	97.51%	88.98%

CHAPITRE 6

CONCLUSION

6.1 Synthèse des travaux

Les travaux effectués dans ce mémoire apportent différentes contributions :

- On a introduit le concept des coupes LBF basées sur des chaînes d'arêtes identifiées dans les solutions intermédiaires. L'expression mathématique des coupes a été explicitée et un algorithme de séparation efficace a été décrit.
- On a également mis en place le concept des coupes LBF basées sur l'identification d'ensembles non structurés de nœuds. De même que pour les coupes LBF basées sur des chaînes, on a explicité leur expression mathématique et développé cette fois deux algorithmes de séparation efficaces.
- On a introduit un principe général qu'on a utilisé pour chacun des algorithmes de séparation. Celui-ci préconise une construction de coupes en agrandissant la taille des ensembles de nœuds concernés de manière itérative. On arrête la construction dès que la borne est suffisante pour produire une coupe violée par la solution courante. Ceci permet de construire des coupes qui comportent le moins de variables possible et donc qui sont actives sur le plus grand nombre de solutions entières et fractionnaires.
- On a montré qu'il était préférable d'ajouter des coupes qui ne concernent pas les arêtes liées au dépôt. Ceci permet de conserver la même borne inférieure, d'enlever des variables de la coupe et de s'affranchir de différents coefficients. On produit des coupes qui sont donc actives sur plus de solutions fractionnaires, ce qui accélère la résolution.
- On a apporté une confirmation à la conjecture de Jabali *et al.* (2014) selon laquelle les coupes LBF les plus efficaces pour le VRPSD sont celles qui sont actives sur le plus grand nombre possible de solutions entières et fractionnaires. Ceci a été montré par l'efficacité non contestable des coupes LBF basées sur des ensembles non structurés de nœuds.
- On a réalisé des tests numériques sur un ensemble très riche d'instances de référence de la littérature. Tous ces tests ont confirmé les bonnes performances de nos modifications de l'algorithme L-shaped en nombres entiers.
- Ces tests ont confirmé l'importance de pouvoir combiner plusieurs types de coupes

ayant un rôle complémentaire. Choisir d'ajouter des coupes qui sont actives sur un grand nombre de solutions peut avoir un prix : peu de coupes peuvent être générées, car la borne associée est souvent trop faible pour produire une coupe violée par la solution courante. Il peut donc être avantageux de les combiner avec des coupes plus spécifiques qui, elles, peuvent être identifiées plus facilement.

Tous ces éléments permettent de résoudre des instances dont la taille atteint 90 clients et 2 véhicules, 80 clients et 3 véhicules ou encore 60 clients et 4 véhicules. On a pu résoudre 13 nouvelles instances de la littérature en moins de 10000 secondes. Enfin, on réduit les temps de calcul des instances résolues de 30% en moyenne.

6.2 Limitations de la solution proposée

Bien que les temps de calcul des instances résolues soient diminués et qu'on arrive à résoudre 13 instances de plus que les travaux de Jabali *et al.* (2014), ce mémoire n'est que la continuité des avancées du domaine déjà mises en place par Gendreau *et al.* (1995), Hjorring et Holt (1999), Laporte *et al.* (2002) et Jabali *et al.* (2014). En effet, les instances les plus difficiles à résoudre ne peuvent toujours être traitées. C'est le cas de certaines instances comportant 4 véhicules, qui avaient déjà posé problème dans les travaux précédents.

On ne dispose pas d'instances de plus grande taille que celles dont disposaient Jabali *et al.* (2014). On ne peut donc pas savoir savoir si notre algorithme permet d'étendre l'ensemble des instances du VRPSD pouvant être résolu.

Enfin, cela aurait été un bon point s'il était possible d'affirmer que nos coupes permettent à l'algorithme L-shaped en nombres entiers de se comporter de mieux en mieux à mesure que la taille des problèmes augmente. Comme on l'a remarqué lors de la présentation des résultats, aucune tendance particulière n'a pu être dégagée concernant la performance de l'algorithme en fonction de la taille et complexité des instances.

6.3 Améliorations futures

Une première amélioration serait de produire des algorithmes de séparation plus performants. Les algorithmes que nous avons mis en place sont des heuristiques efficaces, mais qui ne produisent pas forcément les coupes les plus utiles possible pour la résolution. Ceci provient de différentes raisons : les premiers nœuds que l'on choisit comme candidats potentiels pour la coupe sont sélectionnés de façon arbitraire. En fait, plusieurs choix sont faits

arbitrairement, comme la façon dont sont découpées les chaînes de la solution courante pour produire différentes coupes. Il serait donc judicieux de réfléchir à une manière plus intelligente d'implémenter de tels algorithmes. Il faudrait tout de même garder à l'esprit que l'ajout de coupes qui comportent le moins de variables possible mais qui touchent le plus de solutions entières ou fractionnaires est un principe qui s'est révélé efficace lors des différents tests numériques.

Toutes les coupes LBF de Jabali *et al.* (2014) prennent en compte le dépôt dans leurs processus de séparation. Ceci à donné lieu à des expressions de coupes complexes comportant plusieurs coefficients et prenant en compte plusieurs exceptions. Or, on a vu que le laisser de côté pouvait fournir des coupes plus efficaces en termes de performance et dont l'expression est allégée de certains coefficients et comporte moins de variables. Il pourrait donc être intéressant de tester ce principe sur leurs coupes, en prenant soin de simplifier leur expression de façon à garder leur validité.

L'idée consistant à déterminer d'autres sous-ensembles de variables pouvant être utilisés pour générer des coupes est également une voie vers laquelle on peut se diriger. Ce mémoire a déjà suivi cette direction en partant des apports des travaux précédents et cela a confirmé l'utilité de la démarche. Il semble que les nouveaux types d'agrégations les plus efficaces sont ceux qui permettent aux coupes d'être actives sur le plus grand nombre de solutions entières et fractionnaires possible tout en garantissant une borne assez élevée pour que les solutions intermédiaires violent ces nouvelles contraintes.

En suivant l'idée que l'on vient d'évoquer, on pourrait générer de nouvelles familles de coupes LBF basées sur des ensembles non structurés de nœuds qui ne seraient plus affectés à un seul, mais à plusieurs véhicules. En effet, nos coupes LBF basées sur des ensembles non structurés de nœuds ont l'avantage d'être robustes face au changement d'ordre de parcours des clients faisant partie des ensembles. Cependant, l'observation du comportement de l'algorithme et des profils des solutions intermédiaires montre qu'il arrive assez fréquemment qu'un des clients faisant partie d'un ensemble non structuré d'une coupe soit affecté à une autre route, ce qui rend inactive la coupe en question et qui ralentit le processus global de résolution. Ainsi, éviter ce genre de situation pourrait s'avérer être efficace. Il faudra tout de même noter que la borne associée à cette nouvelle famille de coupes LBF risque d'être peu élevée.

Enfin, on pourrait utiliser les nouvelles coupes LBF que l'on a développées pour la réso-

lution d'autres problèmes d'optimisation stochastique qui utilisent l'algorithme L-shaped en nombres entiers pour la résolution. Il pourrait alors s'agir d'autres versions du VRPSD avec un recours différent ou prenant en compte d'autres contraintes. En fait, tous les problèmes d'optimisation stochastique dans lesquelles des véhicules ou des entités semblables doivent parcourir un graphe en respectant certaines contraintes peuvent être des candidats potentiels à l'utilisation de nos coupes LBF.

RÉFÉRENCES

AK, A. et ERERA, A. L. (2007). A paired-vehicle recourse strategy for the vehicle-routing problem with stochastic demands. *Transportation Science*, 41, 222–237.

BALDACCI, R., MINGOZZI, A. et ROBERTI, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59, 1269–1283.

BEALE, E. M. (1955). On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society. Series B (Methodological)*, 173–184.

BENDERS, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4, 238–252.

BERTSIMAS, D. (1988). *Probabilistic combinatorial optimization problems*. Thèse de doctorat, Massachusetts Institute of Technology.

BERTSIMAS, D. J., JAILLET, P. et ODONI, A. R. (1990). A priori optimization. *Operations Research*, 38, 1019–1033.

BIRGE, J. R. et LOUVEAUX, F. (2011). *Introduction to Stochastic Programming*. Springer, New York.

CHEPURI, K. et HOMEM-DE MELLO, T. (2005). Solving the vehicle routing problem with stochastic demands using the cross-entropy method. *Annals of Operations Research*, 134, 153–181.

CHRISTIANSEN, C. H. et LYSGAARD, J. (2007). A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research Letters*, 35, 773–781.

CORDEAU, J.-F., LAPORTE, G., SAVELSBERGH, M. W. et VIGO, D. (2006). Vehicle routing. *Transportation, Handbooks in Operations Research and Management Science*, 14, 367–428.

CÔTÉ, J.-F., GENDREAU, M. et POTVIN, J.-Y. (2013). The vehicle routing problem with stochastic two-dimensional items. *CIRRELT-2013-84*.

DANTZIG, G. B. (1955). Linear programming under uncertainty. *Management Science*, 1, 197–206.

DANTZIG, G. B. et RAMSER, J. H. (1959). The truck dispatching problem. *Management Science*, 6, 80–91.

DENTON, B. et GUPTA, D. (2003). A sequential bounding approach for optimal appointment scheduling. *IIE Transactions*, 35, 1003–1016.

DROR, M. et TRUDEAU, P. (1986). Stochastic vehicle routing with modified savings algorithm. *European Journal of Operational Research*, 23, 228–235.

GAUVIN, C., DESAULNIERS, G. et GENDREAU, M. (2014). A branch-cut-and-price algorithm for the vehicle routing problem with stochastic demands. *Computers & Operations Research*, 50, 141–153.

GENDREAU, M., LAPORTE, G. et SÉGUIN, R. (1995). An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Science*, 29, 143–155.

GENDREAU, M., LAPORTE, G. et SÉGUIN, R. (1996a). Stochastic vehicle routing. *European Journal of Operational Research*, 88, 3–12.

GENDREAU, M., LAPORTE, G. et SÉGUIN, R. (1996b). A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research*, 44, 469–477.

GOLDEN, B. L. et YEE, J. R. (1979). A framework for probabilistic vehicle routing. *AIIE Transactions*, 11, 109–112.

GOODSON, J. C., OHLMANN, J. W. et THOMAS, B. W. (2012). Cyclic-order neighborhoods with application to the vehicle routing problem with stochastic demand. *European Journal of Operational Research*, 217, 312–323.

HJORRING, C. et HOLT, J. (1999). New optimality cuts for a single-vehicle stochastic routing problem. *Annals of Operations Research*, 86, 569–584.

HVATTUM, L. M., LøKKETANGEN, A. et LAPORTE, G. (2006). Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic. *Transportation Science*, 40, 421–438.

HVATTUM, L. M., LøKKETANGEN, A. et LAPORTE, G. (2007). A branch-and-regret heuristic for stochastic and dynamic vehicle routing problems. *Networks*, 49, 330–340.

JABALI, O., REI, W., GENDREAU, M. et LAPORTE, G. (2014). Partial-route inequalities for the multi-vehicle routing problem with stochastic demands. *Discrete Applied Mathematics*, 177, 121–136.

JAILLET, P. (1988). A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research*, 36, 929–936.

JAILLET, P. et ODONI, A. (1988). The probabilistic vehicle routing problem. *Vehicle routing : Methods and studies. North Holland, Amsterdam.*

JULA, H., DESSOUKY, M. et IOANNOU, P. A. (2006). Truck route planning in nonstationary stochastic networks with time windows at customer locations. *IEEE Transactions on Intelligent Transportation Systems*, 7, 51–62.

LAMBERT, V., LAPORTE, G. et LOUVEAUX, F. (1993). Designing collection routes through bank branches. *Computers & Operations Research*, 20, 783–791.

LAPORTE, G. (2009). Fifty years of vehicle routing. *Transportation Science*, 43, 408–416.

LAPORTE, G., LOUVEAUX, F. et MERCURE, H. (1989). Models and exact solutions for a class of stochastic location-routing problems. *European Journal of Operational Research*, 39, 71–78.

LAPORTE, G., LOUVEAUX, F. et MERCURE, H. (1992). The vehicle routing problem with stochastic travel times. *Transportation Science*, 26, 161–170.

LAPORTE, G. et LOUVEAUX, F. V. (1993). The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13, 133–142.

LAPORTE, G., LOUVEAUX, F. V. et MERCURE, H. (1994). A priori optimization of the probabilistic traveling salesman problem. *Operations Research*, 42, 543–549.

LAPORTE, G., LOUVEAUX, F. V. et VAN HAMME, L. (2002). An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research*, 50, 415–423.

LEI, H., LAPORTE, G. et GUO, B. (2012). The vehicle routing problem with stochastic demands and split deliveries. *INFOR : Information Systems and Operational Research*, 50, 59–71.

LOUVEAUX, F. (1998). An introduction to stochastic transportation models. *Operations Research and Decision Aid Methodologies in Traffic and Transportation Management*, Springer. 244–263.

LOUVEAUX, F. et SCHYNS, M. (2004). Solving the m-tsp problem with stochastic or time dependent demands. *Proceedings of TRISTAN V (Triennial Symposium on Transportation Analysis)*. Le Gosier, Guadeloupe.

LYSGAARD, J., LETCHFORD, A. N. et EGLESE, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100, 423–445.

NOVOA, C. et STORER, R. (2009). An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European Journal of Operational Research*, 196, 509–515.

PANDELIS, D. G., KYRIAKIDIS, E. et DIMITRAKOS, T. (2012). Single vehicle routing problems with a predefined customer sequence, compartmentalized load and stochastic demands. *European Journal of Operational Research*, 217, 324–332.

PSARAFTIS, H. N. (1995). Dynamic vehicle routing : Status and prospects. *Annals of Operations Research*, 61, 143–164.

SECOMANDI, N. (2000). Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Computers & Operations Research*, 27, 1201–1225.

SECOMANDI, N. (2001). A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research*, 49, 796–802.

SECOMANDI, N. et MARGOT, F. (2009). Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Operations Research*, 57, 214–230.

STEWART, W. R. et GOLDEN, B. L. (1983). Stochastic vehicle routing : A comprehensive approach. *European Journal of Operational Research*, 14, 371–385.

TATARAKIS, A. et MINIS, I. (2009). Stochastic single vehicle routing with a predefined customer sequence and multiple depot returns. *European Journal of Operational Research*, 197, 557–571.

TILLMAN, F. A. (1969). The multiple terminal delivery problem with probabilistic demands. *Transportation Science*, 3, 192–204.

VAN SLYKE, R. M. et WETS, R. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17, 638–663.

YANG, W.-H., MATHUR, K. et BALLOU, R. H. (2000). Stochastic vehicle routing problem with restocking. *Transportation Science*, 34, 99–112.

ANNEXE A

MODIFICATION DE LA BORNE ASSOCIÉE AUX COUPES BASÉES SUR DES CHAÎNES

Dans tout ce qui suit, on étudie le coût d'une chaîne $(v_1, v_2, \ldots, v_j)$ que l'on parcourt en allant de v_1 à v_j. Dans le mémoire, on considère qu'en calculant le coût de recours moyen d'un véhicule qui parcourt la chaîne en arrivant avec toute sa capacité, alors on obtient une borne inférieure sur le coût de recours subi par un véhicule dont la route contient la chaîne. Il faut noter premièrement que si cette route démarre au client v_1 puis que d'autres clients sont ajoutés après v_j, alors son coût sera forcément supérieur, car son expression sera composée des mêmes termes que pour la chaîne auquels seront ajoutés d'autres termes positifs. Il faut donc étudier l'impact de l'ajout de clients en amont sur le coût de recours. On note $d_0 \sim \mathcal{N}(\mu_0, \sigma_0)$, la demande cumulée des clients ajoutés avant la chaîne. On va donc étudier $c_{v_1 \to v_j}(\mu_0, \sigma_0)$, le coût de recours de la chaîne $(v_1, \ldots, v_j)$ parcourue de v_1 à v_j sachant que la demande cumulée des clients en amont est d_0.

Dans une première section, on va étudier le signe de la différence de deux densités de probabilité de loi Normale de paramètres (μ_1, σ_1) et (μ_2, σ_2), avec $\mu_1 > \mu_2$ et $\sigma_1 > \sigma_2$. Ceci nous permettra de montrer que $\frac{\partial}{\partial \mu_0} c_{v_1 \to v_j}(\mu_0, \sigma_0) \geq 0$. Nous étudierons ensuite le comportement de $\frac{\partial}{\partial \sigma_0} c_{v_1 \to v_j}(\mu_0, \sigma_0)$ afin de définir un test garantissant la validité de la borne inférieure que l'on calcule pour les chaînes.

A.1 Étude préliminaire

On étudie les deux distributions suivantes :

$$f_1(x) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} \tag{A.1}$$

$$f_2(x) = \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} \tag{A.2}$$

Où :

$$\mu_1, \mu_2, \sigma_1, \sigma_2, D \geq 0 \tag{A.3}$$

$$\mu_1, \mu_2 \leq D \tag{A.4}$$

$$\mu_1 > \mu_2,\ \sigma_1 > \sigma_2 \tag{A.5}$$

$$\sigma_1 \leq \mu_1,\ \sigma_2 \leq \mu_2 \tag{A.6}$$

$$\mu_1 = \mu_2 + \mu \tag{A.7}$$

$$\sigma_1^2 = \sigma_2^2 + \sigma^2 \tag{A.8}$$

$$\mu \geq \sigma \tag{A.9}$$

Les conditions (A.7), (A.8) et (A.9) limitent l'étude mais seront remplies quand on appliquera les résultats fournis plus tard. Le but est de montrer que $f_1(D) - f_2(D) \geq 0$. On peut observer la situation à la figure A.1, la courbe la plus aplatie étant f_1.

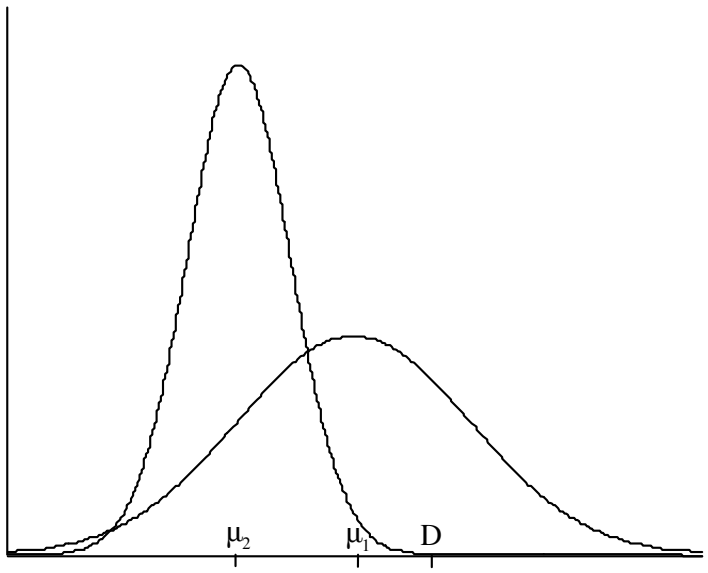

Figure A.1 Densités de probabilité

On va montrer pour cela que l'équation $f_1(x) = f_2(x)$ admet deux solutions x_1 et x_2, que $f_1(x) - f_2(x)$ prend des valeurs négatives sur $[x_1; x_2]$ et des valeurs positives sur $\mathbb{R}\backslash[x_1; x_2]$

et que $D \in \mathbb{R}\backslash[x_1; x_2]$:

$$f_1(x) = f_2(x) \tag{A.10}$$

$$\Leftrightarrow \frac{\sigma_1}{\sigma_2} = \exp\left[\frac{1}{2}\left(\left(\frac{x-\mu_2}{\sigma_2}\right)^2 - \left(\frac{x-\mu_1}{\sigma_1}\right)^2\right)\right] \tag{A.11}$$

$$\Leftrightarrow \left(\frac{x-\mu_2}{\sigma_2}\right)^2 - \left(\frac{x-\mu_1}{\sigma_1}\right)^2 = 2\ln\left(\frac{\sigma_1}{\sigma_2}\right) \tag{A.12}$$

On obtient donc un polynôme de degré 2 qui admet deux racines si et seulement si le discriminant est strictement positif. Après simplification, cette condition est remplie si et seulement si :

$$\sigma_1^2\sigma_2^2\left[(\mu_1-\mu_2)^2 + 2(\sigma_1^2-\sigma_2^2)\ln\left(\frac{\sigma_1}{\sigma_2}\right)\right] > 0 \tag{A.13}$$

Ce qui est vrai car $\sigma_1 > \sigma_2$. Après calcul, on obtient les deux racines suivantes :

$$x_1 = \frac{\left(\mu_2\sigma_1^2 - \mu_1\sigma_2^2\right) - \sigma_1\sigma_2\left(\left(\mu_1-\mu_2\right)^2 + 2\left(\sigma_1^2-\sigma_2^2\right)\ln(\frac{\sigma_1}{\sigma_2})\right)^{\frac{1}{2}}}{\sigma_1^2-\sigma_2^2} \tag{A.14}$$

$$x_2 = \frac{\left(\mu_2\sigma_1^2 - \mu_1\sigma_2^2\right) + \sigma_1\sigma_2\left(\left(\mu_1-\mu_2\right)^2 + 2\left(\sigma_1^2-\sigma_2^2\right)\ln(\frac{\sigma_1}{\sigma_2})\right)^{\frac{1}{2}}}{\sigma_1^2-\sigma_2^2} \tag{A.15}$$

Montrons que $x_1 \leq \mu_2$:

$$x_1 \leq \mu_2 \tag{A.16}$$

$$\Leftrightarrow \mu_2\sigma_1^2 - \mu_1\sigma_2^2 - \sigma_1\sigma_2\left(\left(\mu_1-\mu_2\right)^2 + 2\left(\sigma_1^2-\sigma_2^2\right)\ln(\frac{\sigma_1}{\sigma_2})\right)^{\frac{1}{2}} \leq \mu_2\sigma_1^2 - \mu_2\sigma_2^2 \tag{A.17}$$

$$\Leftrightarrow -\sigma_1\sigma_2\left(\left(\mu_1-\mu_2\right)^2 + 2\left(\sigma_1^2-\sigma_2^2\right)\ln(\frac{\sigma_1}{\sigma_2})\right)^{\frac{1}{2}} \leq \sigma_2^2(\mu_1-\mu_2) \tag{A.18}$$

Ce qui est vrai car $\mu_1 \geq \mu_2$. On est donc assuré que la racine x_1 est inférieure ou égale à μ_2. Montrons maintenant que x_2, la racine la plus grande, se situe entre μ_2 et μ_1, ce qui semble évident sur la figure A.1. Notons $F(x) = f_1(x) - f_2(x)$.

$$F(\mu_2) = \frac{1}{\sqrt{2\pi}}\left(\frac{1}{\sigma_1}e^{-\frac{(\mu_2-\mu_1)^2}{2\sigma_1^2}} - \frac{1}{\sigma_2}\right) \tag{A.19}$$

Ce terme est strictement négatif car $\frac{1}{\sigma_2} > \frac{1}{\sigma_1} > \frac{1}{\sigma_1} e^{-\frac{(\mu_2-\mu_1)^2}{2\sigma_1^2}}$.

$$F(\mu_1) = \frac{1}{\sqrt{2\pi}}\left(\frac{1}{\sigma_1} - \frac{1}{\sigma_2} e^{-\frac{(\mu_1-\mu_2)^2}{2\sigma_2^2}}\right) > 0 \tag{A.20}$$

$$\Leftrightarrow \frac{1}{\sigma_1^2} > \frac{1}{\sigma_2^2} e^{-\left(\frac{\mu_1-\mu_2}{\sigma_2}\right)^2} \tag{A.21}$$

$$\Leftrightarrow \sigma_2^2 e^{\left(\frac{\mu_1-\mu_2}{\sigma_2}\right)^2} > \sigma_1^2 = \sigma_2^2 + \sigma^2 \tag{A.22}$$

$$\Leftrightarrow \exp\left(\frac{\mu^2}{\sigma_2^2}\right) > 1 + \frac{\sigma^2}{\sigma_2^2} \tag{A.23}$$

Or :

$$\exp\left(\frac{\mu^2}{\sigma_2^2}\right) \geq \exp\left(\frac{\sigma^2}{\sigma_2^2}\right) > 1 + \frac{\sigma^2}{\sigma_2^2} \tag{A.24}$$

car $e^x - (1+x) > 0 \; \forall x \in \mathbb{R}^*$. Ainsi, $F(\mu_1) > 0$ et donc comme F est continue, $x_2 \in [\mu_2; \mu_1]$ et F est positive en dehors de l'intervalle $[x_1; x_2]$. On a donc :

$$F(lD) \geq 0, \; \forall l \in \mathbb{N}^* \tag{A.25}$$

car $\mu_1 \leq D$. Ceci constitue le résultat important que l'on utilisera dans les sections suivantes.

A.2 Évolution de $c_{v_1 \to v_j}(\mu_0, \sigma_0)$ en fonction de μ_0

En notant $d_i \sim \mathcal{N}(\mu_i, \sigma_i)$ la demande cumulée au client i, on a :

$$c_{v_1 \to v_j}(\mu_0, \sigma_0) = \sum_{i=1}^{j} 2c_{d,i} \sum_{l=1}^{+\infty} P(d_{i-1} \leq lD \leq d_i) \tag{A.26}$$

où $c_{d,i}$ est la distance du client i au dépôt.

$$\Rightarrow \frac{\partial}{\partial \mu_0} c_{v_1 \to v_j}(\mu_0, \sigma_0) = \sum_{i=1}^{j} 2c_{d,i} \sum_{l=1}^{+\infty} \frac{\partial}{\partial \mu_0} P(d_{i-1} \leq lD \leq d_i) \tag{A.27}$$

Étudions donc $\frac{\partial}{\partial \mu_0} P(d_{i-1} \leq lD \leq d_i) = \frac{\partial}{\partial \mu_0} P(d_{i-1} \leq lD) - P(d_i \leq lD)$. On a :

$$P(d_i \leq lD) = P(z \leq \frac{lD - \mu_i}{\sigma_i}) \text{ où } z \sim \mathcal{N}(0,1) \tag{A.28}$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\frac{lD-\mu_i}{\sigma_i}} e^{-\frac{t^2}{2}} dt \tag{A.29}$$

$$\Rightarrow \frac{\partial}{\partial \mu_0} P(d_i \leq lD) = \frac{1}{\sqrt{2\pi}} \times \Big(\frac{-1}{\sigma_i}\Big) e^{-\frac{(lD-\mu_i)^2}{2\sigma_i^2}} \tag{A.30}$$

$$= -\frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(lD-\mu_i)^2}{2\sigma_i^2}} \tag{A.31}$$

$$\Rightarrow \frac{\partial}{\partial \mu_0} P(d_{i-1} \leq lD \leq d_i) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(lD-\mu_i)^2}{2\sigma_i^2}} - \frac{1}{\sqrt{2\pi}\sigma_{i-1}} e^{-\frac{(lD-\mu_{i-1})^2}{2\sigma_{i-1}^2}} \tag{A.32}$$

On a montré dans la section précédente que ce terme était positif. On a donc :

$$\frac{\partial}{\partial \mu_0} c_{v_1 \to v_j}(\mu_0, \sigma_0) \geq 0 \tag{A.33}$$

A.3 Évolution de $c_{v_1 \to v_j}(\mu_0, \sigma_0)$ en fonction de σ_0

Étudions les termes $P(d_{i-1} \leq lD \leq d_i)$ séparément. Notons $d_{i-1} \sim \mathcal{N}(\mu_{i-1}, \sigma_{i-1})$ avec $\mu_{i-1} = \mu'_{i-1} + \mu_0$, $\sigma_{i-1}^2 = \sigma_{i-1}'^2 + \sigma_0^2$. On note de même $d_i \sim \mathcal{N}(\mu_i, \sigma_i)$ avec $\mu_i = \mu'_i + \mu_0$, $\sigma_i^2 = \sigma_i'^2 + \sigma_0^2$.

$$P(d_i \leq lD) = P(z \leq \frac{lD - \mu_i}{\sigma_i}) \tag{A.34}$$

$$= P(z \leq \frac{lD - \mu_i}{\sqrt{\sigma_i'^2 + \sigma_0^2}}) \tag{A.35}$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\frac{lD-\mu_i}{\sqrt{\sigma_i'^2+\sigma_0^2}}} e^{-\frac{t^2}{2}} dt \tag{A.36}$$

$$\Rightarrow \frac{\partial}{\partial \sigma_0} P(d_i \leq lD) = \frac{1}{\sqrt{2\pi}} \times \frac{(lD - \mu_i) 2\sigma_0 \times (-\frac{1}{2})}{(\sigma_i'^2 + \sigma_0^2)^{\frac{3}{2}}} e^{-\frac{(lD-\mu_i)^2}{2\sigma_i^2}} \tag{A.37}$$

$$= -\frac{\sigma_0 (lD - \mu_i)}{\sqrt{2\pi}\sigma_i^3} e^{-\frac{(lD-\mu_i)^2}{2\sigma_i^2}} \tag{A.38}$$

$$\Rightarrow \frac{\partial}{\partial \sigma_0} P(d_{i-1} \leq lD \leq d_i) = \frac{\sigma_0}{\sqrt{2\pi}} \left[\frac{(lD - \mu_i)}{\sigma_i^3} e^{-\frac{(lD-\mu_i)^2}{2\sigma_i^2}} - \frac{(lD - \mu_{i-1})}{\sigma_{i-1}^3} e^{-\frac{(lD-\mu_{i-1})^2}{2\sigma_{i-1}^2}} \right] \tag{A.39}$$

Si $lD - \mu_i = 0$, alors $\frac{\partial}{\partial \sigma_0} P(d_{i-1} \leq lD \leq d_i) \leq 0$, sinon :

$$\frac{\partial}{\partial \sigma_0} P(d_{i-1} \leq lD \leq d_i) = \frac{\sigma_0}{\sqrt{2\pi}} \left[\frac{1}{(lD-\mu_i)^2} \frac{(lD-\mu_i)^3}{\sigma_i^3} e^{-\frac{(lD-\mu_i)^2}{2\sigma_i^2}} \right. \tag{A.40}$$

$$\left. - \frac{1}{(lD-\mu_{i-1})^2} \frac{(lD-\mu_{i-1})^3}{\sigma_{i-1}^3} e^{-\frac{(lD-\mu_{i-1})^2}{2\sigma_{i-1}^2}} \right] \tag{A.41}$$

$$= \frac{\sigma_0}{\sqrt{2\pi}} \left[\frac{1}{(lD-\mu_i)^2} g\Big(\frac{lD-\mu_i}{\sigma_i}\Big) \right. \tag{A.42}$$

$$\left. - \frac{1}{(lD-\mu_{i-1})^2} g\Big(\frac{lD-\mu_{i-1}}{\sigma_{i-1}}\Big) \right] \tag{A.43}$$

avec g définie par $g(x) = x^3 e^{-\frac{x^2}{2}}$.

$$\frac{1}{(lD-\mu_i)^2} g\Big(\frac{lD-\mu_i}{\sigma_i}\Big) - \frac{1}{(lD-\mu_{i-1})^2} g\Big(\frac{lD-\mu_{i-1}}{\sigma_{i-1}}\Big) \geq \tag{A.44}$$

$$\frac{1}{(lD-\mu_{i-1})^2} \left[g\Big(\frac{lD-\mu_i}{\sigma_i}\Big) - g\Big(\frac{lD-\mu_{i-1}}{\sigma_{i-1}}\Big) \right] \tag{A.45}$$

On va essayer de trouver une condition pour que $g\Big(\frac{lD-\mu_i}{\sigma_i}\Big) \geq g\Big(\frac{lD-\mu_{i-1}}{\sigma_{i-1}}\Big)$. Comme $\frac{lD-\mu_i}{\sigma_i} \leq \frac{lD-\mu_{i-1}}{\sigma_{i-1}}$, il suffit de montrer que g est décroissante sur $[\frac{lD-\mu_i}{\sigma_i}; \frac{lD-\mu_{i-1}}{\sigma_{i-1}}]$.

$$g'(x) = x^2 e^{-\frac{x^2}{2}} (3 - x^2) \leq 0 \Leftrightarrow x \in]-\infty; -\sqrt{3}] \cup [\sqrt{3}; +\infty[\tag{A.46}$$

Ainsi, si $lD \geq \sqrt{3}\sigma_i + \mu_i$, alors $\frac{\partial}{\partial \sigma_0} P(d_{i-1} \leq lD \leq d_i) \geq 0$. Ceci est vérifié pour les termes avec $l \geq 2$.

On peut montrer que sous certaines conditions, alors $\frac{\partial}{\partial \sigma_0} P(d_{i-1} \leq D \leq d_i) \leq 0$:

$$\frac{\partial}{\partial \sigma_0} P(d_{i-1} \leq D \leq d_i) = \frac{\sigma_0}{\sqrt{2\pi}} \left[\frac{1}{\sigma_i^2} \frac{(D-\mu_i)}{\sigma_i} e^{-\frac{(D-\mu_i)^2}{2\sigma_i^2}} - \frac{1}{\sigma_{i-1}^2} \frac{(D-\mu_{i-1})}{\sigma_{i-1}} e^{-\frac{(D-\mu_{i-1})^2}{2\sigma_{i-1}^2}} \right] \tag{A.47}$$

$$= \frac{\sigma_0}{\sqrt{2\pi}} \left[\frac{1}{\sigma_i^2} f\left(\frac{D-\mu_i}{\sigma_i}\right) - \frac{1}{\sigma_{i-1}^2} f\left(\frac{D-\mu_{i-1}}{\sigma_{i-1}}\right) \right] \text{ avec } f(x) = x e^{-\frac{x^2}{2}} \tag{A.48}$$

$$\leq \frac{\sigma_0}{\sqrt{2\pi}\sigma_{i-1}^2} \left[f\left(\frac{D-\mu_i}{\sigma_i}\right) - f\left(\frac{D-\mu_{i-1}}{\sigma_{i-1}}\right) \right] \tag{A.49}$$

Le terme (A.49) est négatif ou nul si et seulement si :

$$f\left(\frac{D-\mu_i}{\sigma_i}\right) \leq f\left(\frac{D-\mu_{i-1}}{\sigma_{i-1}}\right) \tag{A.50}$$

Comme $\frac{D-\mu_i}{\sigma_i} \leq \frac{D-\mu_{i-1}}{\sigma_{i-1}}$, il suffit de montrer que f est croissante sur $\left[\frac{D-\mu_i}{\sigma_i}; \frac{D-\mu_{i-1}}{\sigma_{i-1}}\right]$:

$$f'(x) = e^{-\frac{x^2}{2}}(1-x^2) \geq 0 \Leftrightarrow x \in [-1;1] \tag{A.51}$$

Dans notre problème, $x \geq 0$, on obtient donc la condition $\frac{D-\mu_{i-1}}{\sigma_{i-1}} \leq 1$. Ainsi :

$$\text{si } D \leq \mu_{i-1} + \sigma_{i-1} \text{ alors } \frac{\partial}{\partial \sigma_0} P(d_{i-1} \leq D \leq d_i) \leq 0 \tag{A.52}$$

Essayons de dresser le tableau de variation de $P(d_{i-1} \leq D \leq d_i)$ en fonction de σ_0. La demande cumulée d_0 ne peut pas dépasser un certain seuil. En effet, l'espérance de la demande cumulée d'une route ne peut pas dépasser la capacité D. De plus, pour chaque client, l'écart-type σ_i est tel que $\sigma_i = \alpha\mu_i$ avec α pouvant valoir $1/3, 1/4$ ou 0.4 dépendammant des instances. Ainsi, une borne supérieure sur la valeur que peut prendre σ_0 est $\alpha(D - \sum_{i=1}^{j} \mu_i)$. De plus, si $D \geq \sqrt{3}\sigma_i + \mu_i$, alors $P(d_{i-1} \leq D \leq d_i)$ est croissant, c'est-à-dire si :

$$\frac{(D-\mu_i)^2 - 3\sigma_i'^2}{3} \geq \sigma_0^2 \tag{A.53}$$

Ce seuil pour σ_0 est dénoté S_i^1. De même, le seuil pour σ_0 à partir duquel on sait que $P(d_{i-1} \leq D \leq d_i)$ est décroissant est respecté si $D - \mu_{i-1} \leq \sigma_{i-1} \Leftrightarrow (D-\mu_{i-1})^2 - \sigma_{i-1}'^2 \leq \sigma_0^2$. On dénote ce seuil S_i^2. Le tableau de variation de $P(d_{i-1} \leq D \leq d_i)$ ressemble donc au tableau A.1, où $S_0 = \alpha(D - \sum_{i=1}^{j} \mu_i)$.

σ_0	0	S_i^1	S_i^2	S_0
$P(d_{i-1} \leq D \leq d_i)$		↗	↘	

Tableau A.1 Tableau de variation de $P(d_{i-1} \leq D \leq d_i)$ en fonction de σ_0

On est sûr que $S_i^1 \leq S_i^2$ car $P(d_{i-1} \leq D \leq d_i)$ ne peut pas être à la fois croissant et décroissant sur un intervalle. En revanche, il est difficile de situer les termes S_i^1 et S_i^2 par rapport à $\alpha(D - \sum_{i=1}^{j} \mu_i)$. De plus, on ne sait pas prédire le comportement de $P(d_{i-1} \leq D \leq d_i)$ sur $[S_i^1; S_i^2]$.

On va montrer que la dérivée de $P(d_{i-1} \leq D \leq d_i)$ s'annule en un seul point, ce qui assurera le fait qu'il ne peut pas y avoir de minimum local entre S_i^1 et S_i^2. La dérivée de $P(d_{i-1} \leq D \leq d_i)$ par rapport à σ_0 s'annule si et seulement si :

$$\frac{(D-\mu_i)}{\sigma_i^3} e^{-\frac{(D-\mu_i)^2}{2\sigma_i^2}} = \frac{(D-\mu_{i-1})}{\sigma_{i-1}^3} e^{-\frac{(D-\mu_{i-1})^2}{2\sigma_{i-1}^2}} \tag{A.54}$$

$$\Leftrightarrow \frac{(D-\mu_i)}{(D-\mu_{i-1})} \frac{\sigma_{i-1}^3}{\sigma_i^3} \exp\left(\frac{(D-\mu_{i-1})^2}{2\sigma_{i-1}^2} - \frac{(D-\mu_i)^2}{2\sigma_i^2}\right) = 1 \tag{A.55}$$

$$\Leftrightarrow \frac{(D-\mu_i)}{(D-\mu_{i-1})} \frac{(\sigma_{i-1}'^2 + \sigma_0^2)^{\frac{3}{2}}}{(\sigma_i'^2 + \sigma_0^2)^{\frac{3}{2}}} \exp\left(\frac{(D-\mu_{i-1})^2}{2(\sigma_{i-1}'^2 + \sigma_0^2)} - \frac{(D-\mu_i)^2}{2(\sigma_i'^2 + \sigma_0^2)}\right) = 1 \tag{A.56}$$

Montrons que la fonction h définie par :

$$h(\sigma_0) = \frac{(\sigma_{i-1}'^2 + \sigma_0^2)^{\frac{3}{2}}}{(\sigma_i'^2 + \sigma_0^2)^{\frac{3}{2}}} \exp\left(\frac{(D-\mu_{i-1})^2}{2(\sigma_{i-1}'^2 + \sigma_0^2)} - \frac{(D-\mu_i)^2}{2(\sigma_i'^2 + \sigma_0^2)}\right) \tag{A.57}$$

est strictement monotone, ce qui assurera l'unicité de la solution de l'équation (A.56). Après simplification, le calcul de la dérivée de h par rapport à σ_0 donne :

$$\exp\left(\frac{(D-\mu_{i-1})^2}{2(\sigma_{i-1}'^2 + \sigma_0^2)} - \frac{(D-\mu_i)^2}{2(\sigma_i'^2 + \sigma_0^2)}\right) \frac{\sigma_0 \sigma_{i-1}}{\sigma_i^5} \left[\sigma_i^2 \left(3 - \frac{(D-\mu_{i-1})^2}{\sigma_{i-1}^2}\right) - \sigma_{i-1}^2 \left(3 - \frac{(D-\mu_i)^2}{\sigma_i^2}\right)\right] \tag{A.58}$$

Montrons par l'absurde que ce terme est strictement négatif. Supposons donc que :

$$\sigma_i^2 \left(3 - \frac{(D-\mu_{i-1})^2}{\sigma_{i-1}^2}\right) \geq \sigma_{i-1}^2 \left(3 - \frac{(D-\mu_i)^2}{\sigma_i^2}\right) \tag{A.59}$$

$$\Leftrightarrow 3(\sigma_i^2 - \sigma_{i-1}^2) \geq \frac{\sigma_i^2}{\sigma_{i-1}^2}(D-\mu_{i-1})^2 - \frac{\sigma_{i-1}^2}{\sigma_i^2}(D-\mu_i)^2 \tag{A.60}$$

Or, on a les relations suivantes :

$$\mu_i = \mu_{i-1} + \mu_{v_i} \tag{A.61}$$

$$\sigma_i^2 = \sigma_{i-1}^2 + \sigma_{v_i}^2 \tag{A.62}$$

On a donc :

$$3\sigma_{v_i}^2 \geq \frac{\sigma_{i-1}^2 + \sigma_{v_i}^2}{\sigma_{i-1}^2}(D - \mu_{i-1})^2 - \frac{\sigma_i^2 - \sigma_{v_i}^2}{\sigma_i^2}(D - \mu_i)^2 \tag{A.63}$$

$$\Leftrightarrow 3\sigma_{v_i}^2 \geq (D - \mu_{i-1})^2 - (D - \mu_i)^2 + \frac{\sigma_{v_i}^2}{\sigma_{i-1}^2}(D - \mu_{i-1})^2 + \frac{\sigma_{v_i}^2}{\sigma_i^2}(D - \mu_i)^2 \tag{A.64}$$

Or :

$$(D - \mu_{i-1})^2 - (D - \mu_i)^2 = (D - \mu_{i-1})^2 - (D - \mu_{i-1} - \mu_{v_i})^2 \tag{A.65}$$

$$= 2(D - \mu_{i-1})\mu_{v_i} - \mu_{v_i}^2 \tag{A.66}$$

$$= \mu_{v_i}(2(D - \mu_{i-1}) - \mu_{v_i}) \tag{A.67}$$

$$= \mu_{v_i}(2D - \mu_{i-1} - \mu_{i-1} - \mu_{v_i}) \tag{A.68}$$

$$= \mu_{v_i}(2D - \mu_{i-1} - \mu_i) \tag{A.69}$$

L'équation (A.64) se ré-écrit donc :

$$3\sigma_{v_i}^2 + \mu_{v_i}(\mu_{i-1} + \mu_i - 2D) \geq \frac{\sigma_{v_i}^2}{\sigma_{i-1}^2}(D - \mu_{i-1})^2 + \frac{\sigma_{v_i}^2}{\sigma_i^2}(D - \mu_i)^2 \tag{A.70}$$

Avec :

$$3\sigma_{v_i}^2 + \mu_{v_i}(\mu_{i-1} + \mu_i - 2D) = 3\alpha^2\mu_{v_i}^2 + \mu_{v_i}(\mu_{i-1} + \mu_i - 2D) \tag{A.71}$$

$$= \mu_{vi}(3\alpha^2\mu_{v_i} + \mu_{i-1} + \mu_i - 2D) \tag{A.72}$$

$$\leq \mu_{vi}(3(0.4)^2\mu_{v_i} + \mu_{i-1} + \mu_i - 2D) \tag{A.73}$$

$$\leq \mu_{vi}(\mu_{v_i} + \mu_{i-1} + \mu_i - 2D) \tag{A.74}$$

$$\leq 2\mu_{vi}(\mu_i - D) \tag{A.75}$$

$$\leq 0 \tag{A.76}$$

Or, le terme de droite de l'inégalité (A.70) est strictement positif car $D > \mu_{i-1}$. On vient de montrer que le terme de gauche est négatif ou nul, ce qui est impossible. L'hypothèse de base était donc fausse et on peut donc conclure que $h'(\sigma_0) < 0$ ce qui montre que la dérivée de $P(d_{i-1} \leq D \leq d_i)$ s'annule en un seul point.

A.4 Test à inclure

Afin d'assurer la validité de la borne inférieure calculée pour les chaînes identifiées, on va implémenter un test qui assurera qu'on a effectivement calculé une borne inférieure.

Dans l'expression de $c_{v_1 \to v_j}(\mu_0, \sigma_0) = \sum_{i=1}^{j} 2c_{d,i} \sum_{l=1}^{+\infty} P(d_{i-1} \leq lD \leq d_i)$, tous les termes pour lesquels $l \geq 2$ sont croissants en fonction de μ_0 avec σ_0 fixe, et en fonction de σ_0 avec μ_0 fixe. On est donc assuré d'avoir la valeur minimale en les calculant pour $\mu_0 = 0$ et $\sigma_0 = 0$. Il suffit donc de comparer les termes pour lesquels $l = 1$.

Pour une valeur de σ_0 fixe, on a vu que $P(d_{i-1} \leq D \leq d_i)$ était croissant en fonction de μ_0. Choisir $\mu_0 = 0$ nous assure donc d'avoir la valeur minimale. En revanche, d'après le tableau de variation en fonction de σ_0 de la section précédente, on observe qu'il peut y avoir deux candidats qui minimisent la valeur de $P(d_{i-1} \leq D \leq d_i)$. Il s'agit de la valeur prise en $\sigma_0 = 0$ que l'on a choisie pour calculer notre borne, et la valeur prise en $\sigma_0 = \alpha(D - \sum_{i=1}^{j} \mu_i)$. Notons en revanche que choisir une valeur de σ_0 strictement positive implique qu'on a ajouté au moins un client en amont de la chaîne. L'espérance de chaque client étant au minimum de 1, si on calcule la valeur de $P(d_{i-1} \leq D \leq d_i)$ en $\sigma_0 = \alpha(D - \sum_{i=1}^{j} \mu_i)$, il faudra également imposer $\mu_0 = 1$.

Le test consiste donc à comparer la valeur des termes $P(d_{i-1} \leq D \leq d_i)$ en supposant $\mu_0 = \sigma_0 = 0$, et celle de $P(d_{i-1} \leq D \leq d_i)$ en supposant $\mu_0 = 1$ et $\sigma_0 = \alpha(D - \sum_{i=1}^{j} \mu_i)$. Si la valeur de la deuxième expression est supérieure ou égale à celle de la première, alors notre borne sera valide. Concrètement, en notant $P_{i,l}(\xi) = P\Big(\xi + \sum_{s=1}^{i-1} \xi_{v_s} \leq lD < \xi + \sum_{s=1}^{i} \xi_{v_s}\Big)$, le calcul de la borne d'une chaîne $(v_1, \ldots, v_j)$ parcourue de v_1 à v_j a comme expression modifiée :

$$P = \sum_{i=2}^{j} 2c_{1v_i} \left[\min \left[P_{i,1}(0), P_{i,1}(\xi_0) \right] + \sum_{l=2}^{i-1} P_{i,l}(0) \right] \tag{A.77}$$

où le terme 0 désigne une demande en amont nulle et $\xi_0 \sim \mathcal{N}(1, S_0)$.