UNIVERSITÉ DE MONTRÉAL

OPTIMISATION DE TOURNÉES DE SERVICE EN TEMPS RÉEL

SIXTINE BINART
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(MATHÉMATIQUES DE L'INGÉNIEUR)
MARS 2014

POLYTECHNIQUE MONTRÉAL

THÈSE

Université Lille1 Sciences et Technologies

CIRRELT

LAGIS

En vue de l'obtention du grade de

DOCTEUR DE L'UNIVERSITÉ DE LILLE 1

et de

PHILOSOPHIAE DOCTOR DE L'ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Spécialités : *Automatique et Mathématiques de l'ingénieur*

Présentée et soutenue le *28/03/2014* **par :**

Sixtine BINART

Optimisation de tournées de service en temps réel

JURY

Daniele VIGO	Professeur, Université de Bologne	Rapporteur
François LOUVEAUX	Professeur, Université de Namur	Rapporteur
Dominique FEILLET	Professeur, Ecole des Mines de Saint-Etienne	Examinateur
Gilles PESANT	Professeur, Ecole Polytechnique de Montréal	Examinateur
Louis-Martin ROUSSEAU	Professeur, Ecole Polytechnique de Montréal	Examinateur
Nicolaï CHRISTOV	Professeur, Université de Lille 1	Directeur
Michel GENDREAU	Professeur, Ecole Polytechnique de Montréal	Directeur
Frédéric SEMET	Professeur, Ecole Centrale de Lille	Co-directeur
Pierre DEJAX	Professeur, Ecole des Mines de Nantes	Membre invité

Directeurs de Thèse :

Nicolaï CHRISTOV et *Michel GENDREAU*

Unités de Recherche :

LAGIS, école doctorale EDSPI

CIRRELT, département de mathématiques et de génie industriel

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

OPTIMISATION DE TOURNÉES DE SERVICE EN TEMPS RÉEL

présentée par : BINART Sixtine
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. FEILLET Dominique, Ph.D, président
M. GENDREAU Michel, Ph.D, membre et directeur de recherche
M. CHRISTOV Nicolaï, Ph.D, membre et codirecteur de recherche
M. SEMET Frédéric, Ph.D, membre et codirecteur de recherche
M. ROUSSEAU Louis-Martin, Ph.D, membre
M. VIGO Daniele, Ph.D, membre
M. LOUVEAUX François, Ph.D, membre

DÉDICACE

À Vincent Thirion,
mon fiancé ...

REMERCIEMENTS

Je tiens tout d'abord à remercier Frédéric Semet, mon co-directeur de thèse, pour la confiance qu'il m'a accordée en me proposant cette thèse il y a trois ans, pour le soutien qu'il a su me fournir, pour tous ses conseils qui m'ont beaucoup aidé et appris, d'un point de vue scientifique comme d'un point de vue méthodologie de travail et autonomie. Merci pour toutes les discussions très intéressantes que nous avons eu et pour son soutien financier en fin de thèse. Enfin, je le remercie vivement pour les repas partagés à chacun de mes retours de Montréal, pour sa sympathie et son aptitude à ménager mes humeurs.

Je tiens également à remercier Michel Gendreau, mon directeur de thèse, pour sa confiance et pour tout ce qu'il m'a apporté, à savoir son expertise dans le domaine du stochastique, sa connaissance étendue de la littérature, et ses conseils avisés. Il m'a également beaucoup apporté d'un point de vue culturel, en m'accueillant bras ouverts à Montréal, et en me donnant de bonnes adresses et de bons conseils sur place. Un grand merci de m'avoir donné la possibilité de participer à des conférences, même si, pour certaines, je n'y ai participé que par procuration, à mon grand regret. Enfin, merci de m'avoir offert l'opportunité d'encadrer des travaux dirigés à l'Ecole Polytechnique de Montréal. J'ai ainsi pu me rendre compte que la sévérité dans la notation française n'avait pas lieu d'être au Canada.

Je tiens aussi à remercier Nicolaï Christov, mon directeur de thèse, qui m'a permis d'effectuer cette thèse et m'a offert son soutien dès le début, alors même qu'il ne me connaissait pas. Il m'a, entre autres, beaucoup assisté dans de nombreuses démarches administratives rendues compliquées par ma cotutelle de thèse. Il a toujours répondu présent et je l'en remercie.

Je tiens également à remercier Pierre Dejax, mon deuxième co-directeur de thèse, mais qui pour des raisons administratives, n'a pu apparaître qu'en tant que membre invité dans ce manuscrit alors qu'il mérite largement sa place de co-directeur. En effet, ce projet provient de lui et je n'aurais donc pas pu travailler sur ce sujet sans lui. De plus, il m'a apporté un oeil industriel sur la problématique, m'a fourni de précieux conseils et a toujours répondu à mes demandes de correction/relecture dans les temps impartis, même quand les délais impartis étaient serrés.

Un grand merci à Daniele Vigo et François Louveaux, mes rapporteurs, pour leur relecture consciencieuse de mon manuscrit et leur rapports constructifs. Un grand merci également à Louis-Martin Rousseau, examinateur, d'avoir accepté de juger mon travail et de suivre ma soutenance de thèse par visioconférence (ce qui ne fut pas sans quelques difficultés). Merci aussi à Dominique Feillet, examinateur et directeur du jury, d'avoir accepté de juger mon travail et de présider mon jury de soutenance. Merci à tous ces membres de mon jury

pour leurs questions après ma soutenance, qui m'ont permis d'entrevoir d'autres perspectives futures sur mon travail.

Je remercie également les équipes des deux laboratoires (LAGIS et CIRRELT) au sein desquels j'ai pu séjourner durant ma thèse. Je tiens à remercier tout particulièrement Serge Bisaillon du CIRRELT pour son aide et son soutien face aux multiples problèmes informatiques auxquels j'ai été confrontée. Il a toujours répondu présent et m'a permis de résoudre un nombre de problèmes incalculable. Merci à lui pour tout le temps qu'il a su me consacrer et pour ses précieux conseils. Je tiens également à remercier Daniel Charbonneau du CIRRELT, pour son aide dans mon apprentissage de Linux. Je suis arrivée au CIRRELT, n'ayant jamais travaillé sous linux et j'utilise à présent beaucoup plus la console de commandes grâce à tout ce qu'il m'a appris et à tous ses conseils. Un grand merci à Lucie-Nathalie Cournoyer pour sa sympathie et son accueil, ainsi qu'à toute l'équipe du CIRRELT. Je voudrais également remercier tout particulièrement Patrick Gallais du LAGIS pour son aide lors des problèmes d'ordinateur ou de disque dur rencontrés vers la fin de ma thèse. Il a su se montrer d'une grande aide, disponible et efficace et je l'en remercie. Un grand merci à Christine Yvoz et Brigitte Foncez du LAGIS pour leur aide dans de nombreuses démarches administratives et pour leur sympathie. Je tiens également à adresser un remerciement spécial à Bernard Szukala pour ses blagues, son humour, sa bonne humeur et nos nombreuses discussions fort sympathiques au laboratoire. Un grand merci à Mireille pour sa sympathie, ses gâteaux, ses dessins, son sourire et son soutien sans faille. Un grand merci enfin à toutes les personnes du troisième étage du bâtiment C pour leur sympathie et leur sourire. Enfin, j'adresse un remerciement spécial à Thomas Bourdeaud'huy pour m'avoir supportée comme co-bureau durant la deuxième partie de ma thèse.

Mes remerciements vont aussi à ma famille et mes amis. Un grand merci à mes parents et mes grand-parents, sans qui je n'en serai pas là aujourd'hui, qui m'ont soutenu avec affection durant ma thèse et qui se sont déplacés pour assister à ma soutenance de thèse. Un grand merci à mes soeurs pour leur soutien sans faille (en particulier lors du problème avec mon disque dur) et pour les super moments de bonheur qu'elles ont su m'offrir durant ma thèse. Merci beaucoup à la famille Thirion, qui a toujours répondu présente et qui m'a soutenue en m'offrant toute son affection ainsi que des moments de pur bonheur durant ces trois années de thèse. Ils sont également venus assister à ma soutenance de thèse (de Nancy et même de Mulhouse), pour mon plus grand bonheur. Un merci tout particulier à Solange Henry pour sa patience d'ange à relire mon manuscrit minutieusement et pour toute son affection et pour son soutien durant ma thèse. Enfin, je remercie également mes amis pour leur compréhension face à ma faible disponibilité durant ma thèse et pour les bons moments passés en leur compagnie. Je remercie également celles et ceux qui ont pu se libérer et venir assister à ma

soutenance de thèse.

Enfin, je tiens à adresser un remerciement tout particulier à mon fiancé, Vincent Thirion, pour le soutien indéfectible qu'il m'a apporté tout au long de ma thèse. A chacun de mes moments de doute (et ils ont été nombreux), il a répondu présent et a su trouver les mots justes pour me redonner la motivation nécessaire. Vincent m'a également donné de précieux conseils tout au long de cette thèse et a supporté toutes mes présentations blanches et relu tous mes documents. Il a su également me supporter durant ces trois années difficiles et a su faire des sacrifices sur certains week-ends, vacances et autres (en se montrant très compréhensif). Plus que tout, il m'a fait passer avant sa vie professionnelle, en quittant un CDD qui le passionnait pour m'accompagner au Canada, en acceptant de rester au Canada avec moi alors qu'il n'avait pas de travail là-bas (pour cause de visa) et en acceptant de prendre un poste à son retour en France dans un tout autre domaine que celui qu'il souhaitait, en attendant la fin de ma thèse. Autant dire que Vincent a fait preuve d'une grande patience et d'un amour sans faille. C'est pour toutes ces raisons que je lui dédie ce manuscrit de thèse et le travail de ces trois dernières années.

RÉSUMÉ

Les tournées de service concernent l'organisation de déplacement de personnels vers des clients afin d'effectuer différentes activités techniques ou commerciales. Ces tournées peuvent devoir répondre à des objectifs et faire face à des contraintes nombreuses et complexes. Lors de la planification et de l'exécution de tournées de service mono-période, les entreprises sont confrontées aux aléas des temps de service et de parcours. C'est pourquoi, dans cette thèse, nous nous intéressons à une variante du problème de tournées de service, dans laquelle les temps de parcours et de service sont stochastiques. Il s'agit du problème de tournées de service multi-dépôt, incluant fenêtres de temps, temps de service et de parcours stochastiques avec priorité entre les clients (distinction clients obligatoires / clients optionnels). Afin de résoudre cette problématique, nous proposons trois méthodes différentes. Dans la première méthode, nous construisons d'abord des routes contenant uniquement des clients obligatoires puis nous procédons à l'insertion des clients optionnels. La deuxième méthode est une méthode approchée basée sur la génération de colonnes consistant à générer un ensemble de routes de bonne qualité pour chaque véhicule puis à en sélectionner une par véhicule. La dernière méthode est un algorithme de branch and price basé sur la deuxième méthode. Le sous-problème consiste à générer des routes réalisables pour un véhicule donné, tandis que le problème maître permet de sélectionner des routes en s'assurant que la priorité des clients est respectée. Après chacune de ces méthodes, afin d'évaluer la qualité de ces solutions face aux aléas, nous utilisons un algorithme de programmation dynamique et procédons à un ensemble de simulations du déroulement des tournées en temps réel. Nous avons testé ces méthodes sur des problèmes dont les données sont issues du milieu industriel.

Mots-clés : Tournées de véhicules, multi-dépôt, fenêtres de temps, temps de service stochastiques, temps de parcours stochastiques, priorité entre les clients

ABSTRACT

The field service routing problem consists in assigning the visits of technicians to clients in order to satisfy their requests for service activities such as maintenance. When planning service routes, companies have to face hazardous travel and service times. Therefore, in this thesis, we deal with a variant of the single-period field service routing problem in which travel and service times are stochastic. It is the field service routing problem with multiple depots, time windows, stochastic travel and service times and priority within customers (distinguishing mandatory and optional customers). To solve this problem, we propose three different methods. In the first one, we first build routes containing only mandatory customers and then, we insert optional customers in these routes. The second one is a heuristic method based on column generation consisting in generating a set of valuable routes for each vehicle and then in selecting one route per vehicle. The last method is a branch and price algorithm, based on the second method, in which the subproblem consists in finding feasible routes for a given vehicle, whereas the master problem consists in selecting routes while ensuring that customer's priority is respected. After each of these methods, in order to evaluate the quality of these solutions regarding stochasticity, we use a dynamic programming algorithm and we proceed to a set of simulations of the real-time execution of the service activities over the period. All our experimentations have been made on problems coming from realistic data.

Keywords : Vehicle routing, multi-depot, time windows, stochastic service times, stochastic travel times, priority within customers

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

CHAPITRE 1

Introduction

Aujourd'hui, avec le développement du secteur tertiaire, nombreuses sont les entreprises confrontées à la planification de tournées de service. Cette activité consiste à organiser, sur une ou plusieurs périodes de temps, les déplacements de personnels chez des clients (industriels ou particuliers), pour effectuer des opérations techniques ou commerciales. Aussi, dans un contexte de compétitivité croissante, elles doivent faire face à une clientèle des plus exigeantes et cherchent à améliorer le niveau de service. Elles sont donc souvent amenées à respecter des heures de rendez-vous (ou des plages horaires) et doivent gérer une notion de priorité entre les clients (dépendant, par exemple, de la fidélité du client, de l'importance de la demande...). De plus, ces entreprises doivent aussi prendre en compte les variations des temps de service (durée effective du service chez les clients) et de parcours (du fait des aléas dans les transports). Si cette nécessité est une réalité, peu nombreux sont ceux qui se sont intéressés au problème de tournées de service avec temps de service et de parcours stochastiques. Le plus souvent, le problème de planification de tournées de service comporte des temps de service stochastiques ou des temps de parcours stochastiques (incorporant parfois la variation des temps de service dans celle des temps de parcours). Ces contraintes peuvent rendre les approches déterministes inapplicables ou peu performantes.

L'objectif de cette thèse est de proposer des méthodes de résolution pour le problème de tournées de service mono-période avec fenêtres de temps de visite chez les clients, plusieurs dépôts, priorité entre les clients (nous distinguons les clients "obligatoires" dont la visite est impérative et les clients "optionnels" dont la visite peut être annulée ou reportée à une période ultérieure) et temps de service et de parcours stochastiques.

Pour résoudre cette problématique, nous proposons une approche globale de résolution en deux étapes : une étape de planification et une étape d'exécution. Au cours de l'étape de planification, on construit des routes contenant des clients obligatoires et optionnels. Durant l'étape d'exécution, on utilise des outils de programmation dynamique pour déterminer la politique optimale. Après la programmation dynamique, on procède à un ensemble de simulations de l'exécution des tournées en temps réel au long de la période afin d'évaluer la qualité des solutions obtenues. Pour l'étape de planification, nous proposons trois méthodes distinctes : une heuristique basée sur la priorité des clients, consistant à établir des routes à partir des clients obligatoires puis à insérer les clients optionnels dans ces routes ; une heuristique basée sur la génération de colonnes consistant à générer un ensemble de routes de

bonne qualité pour chaque véhicule puis à en sélectionner une par véhicule ; un algorithme de branch and price dans laquelle le problème maître consiste à sélectionner des routes en s'assurant que la priorité des clients est respectée tandis que le sous-problème consiste à générer des routes réalisables. Dans chacune de ces méthodes de planification, on utilise des estimés connus a priori des temps de service et de parcours (estimés minimaux, maximaux ou modaux). Nous procédons à l'expérimentation de ces méthodes sur la base d'un ensemble de jeux de données réalistes déjà publiés et correspondant aux caractéristiques des tournées de service d'une grande entreprise.

Dans le premier chapitre, nous donnons une description détaillée de la problématique qui nous intéresse. Puis, dans le chapitre 2, nous situons le problème étudié par rapport à la litérature existante. Dans le troisième chapitre, nous prouvons que la politique optimale de notre algorithme de programmation dynamique est une politique de seuil. Enfin, dans les chapitres 4, 5 et 6, nous détaillons les trois méthodes de résolution proposées pour le problème traité.

CHAPITRE 2

Le problème de tournées de service avec temps de parcours et de service stochastiques

2.1 Description de la problématique

Avant de décrire notre problème, rappelons la définition du problème de tournées de service. Il peut être formulé comme suit. Etant donné un nombre limité de techniciens et de requêtes clients, il s'agit de trouver des routes pour desservir ces requêtes, en s'assurant que les clients sont servis dans leur fenêtre de temps par un technicien ayant les compétences requises. L'objectif est de minimiser la distance parcourue.

Le problème auquel nous nous intéressons ici est une variante du problème de tournées de service. Nous complétons sa description en précisant un certain nombre d'éléments et d'hypothèses :

- *Techniciens omniscients*
  On suppose que les techniciens sont capables de procéder à toutes les interventions. En d'autres termes, on supprime l'aspect compétences des techniciens du problème de tournées de service classique. Aussi, dans la suite, on suppose qu'à chaque technicien est associé un véhicule et on utilise le terme de véhicule plutôt que celui de technicien.

- *Priorité entre les clients*
  Dans ce problème, on distingue deux types de clients : les clients obligatoires et les clients optionnels.
  Les clients optionnels sont connus a priori et n'ont pas de plage horaire de visite imposée (nous considérons qu'ils ont une fenêtre de temps correspondant à l'horizon de temps et peuvent être reportés à une autre période en tout temps). De plus, afin de gérer une certaine notion de priorité au sein des clients optionnels, un profit leur est associé. Les client obligatoires apparaissent au cours du temps mais on suppose qu'un client obligatoire ne sera pas servi pendant la période au cours de laquelle il apparaît (s'il apparaît à la période J, il ne sera pas servi avant la période $J+1$). Tous les clients obligatoires sont donc connus a priori à chaque période. De plus, une fenêtre de temps dure est associée à chaque client obligatoire dès qu'il apparaît.

– *Fenêtres de temps dures*
  Les fenêtres de temps du dépôt (durée d'une journée de travail) et des clients obligatoires sont dures. Ce qui signifie que l'on n'autorise aucune heure supplémentaire ni aucun retard chez les clients obligatoires.

– *Aspect multi-dépôts*
  On suppose que l'on dispose, pour les véhicules, non pas d'un dépôt central mais de plusieurs dépôts. Chaque véhicule a son propre dépôt origine et dépôt destination (le plus souvent, il s'agit du domicile du technicien). Pour un véhicule donné, le dépôt origine et le dépôt destination peuvent être identiques.

– *Véhicules de capacité infinie*
  Etant donné qu'il s'agit de tournées de service et non de transport de marchandises, on supposera que la capacité des véhicules est infinie (ou, en d'autres termes, que le volume transporté par chaque véhicule n'excède pas sa capacité).

– *Temps de parcours et de service stochastiques*
  Souvent, dans les problèmes de tournées de service, des temps de parcours et de service déterministes sont considérés. Cela ne reflète pourtant pas la réalité : les temps de transport sont soumis à des aléas tels que la météorologie, le trafic, les accidents... De même, les aléas sur les temps de service ne sont pas toujours négligeables. En effet, quel que soit le type de service fourni, les temps de service sont variables (cela peut être lié à l'absence du client, à la nécessité de monter des escaliers, ainsi qu'à de nombreux autres aléas). Ces variations ne sont pas neutres. On a donc choisi de prendre en compte des temps de transport et de service stochastiques.

Une application générique de ce problème est la construction de routes de techniciens pour des opérations de maintenance et de réparation. Dans ce problème, les clients obligatoires requièrent des opérations de type réparation tandis que les clients optionnels requièrent des opérations de service (contrôle, relevé de compteur, maintenance, ...). Dans cette application, comme les véhicules servent uniquement au transport de matériel et du personnel, on peut effectivement supposer que la capacité de leur véhicule est infinie. D'autre part, les véhicules ne transportant pas de marchandises, ils peuvent donc avoir leurs propres dépôts origine et destination (typiquement les domiciles des techniciens). Enfin, comme le service fourni au client peut être de type réparation, on comprend la nécessité de considérer des temps de

service stochastiques.

2.2 Hypothèses

On suppose que tous les clients sont connus a priori (cf. plus haut, paragraphe Priorité entre les clients).
Les valeurs minimales, modales et maximales des temps de service et des vitesses sont également supposées connues a priori. En effet, on peut supposer que l'on dispose d'un historique permettant de calculer ces valeurs.
On suppose que les distances vérifient l'inégalité triangulaire et que les unités de temps sont discrètes (on travaille, par exemple, en minutes).
Aussi, on suppose que les temps de parcours et les temps de service sont indépendants.

2.3 Lois de probabilité

Pour modéliser les temps de parcours et de service stochastiques, on souhaite des lois de probabilité tronquées. En effet, les temps de parcours et de service sont bornés. Aussi, on suppose que les unités de temps sont discrètes (cf. ci-dessus). On choisit donc d'utiliser des lois de distribution triangulaires discrètes.
Pour le temps de service au client i, on utilise une loi de distribution triangulaire discrète et symétrique entre $\underline{\sigma}_i - 1$ et $\overline{\sigma}_i + 1$, où $\underline{\sigma}_i$ et $\overline{\sigma}_i$ désignent respectivement le temps de service minimal et maximal du client i.
Pour le temps de parcours unitaire δ, on utilise une loi de distribution triangulaire discrète entre $\left\lceil \frac{100}{v_{max}} \right\rceil - 1$ et $\left\lceil \frac{100}{v_{min}} \right\rceil + 1$, de mode $\left\lceil \frac{100}{v_{mode}} \right\rceil$ où v_{min} et v_{max} correspondent respectivement aux vitesses de parcours minimale et maximale, et v_{mode} désigne la vitesse de parcours modale (la plus probable). Ensuite, afin de s'assurer que les temps de parcours vérifient l'inégalité triangulaire, la probabilité du temps de parcours τ_{ij} entre les clients i et j est donnée par la formule :

$$P(\tau_{ij} = m) = P\left(\left\lceil \frac{D_{ij}\delta}{100} \right\rceil = m\right)$$

avec D_{ij} la distance euclidienne entre les clients i et j.

2.4 Instances

Afin de pouvoir tester les différentes méthodes développées dans cette thèse, nous proposons d'utiliser des instances extraites de celles de Tricoire [68] et correspondant aux caracté-

ristiques des demandes de services d'une grande entreprise. Dans sa thèse, Tricoire [68] s'est intéressé au problème de tournées de véhicules multi-dépôts, multi-périodes, avec fenêtres de temps, priorité entre les clients et points de restauration. Toutefois, il prend en compte des temps de parcours et de service déterministes. Dans ce contexte, il propose des instances avec 3 véhicules, sur un horizon de 5 jours. Dans ces instances, il associe à chaque client des coordonnées (x, y), une fenêtre de temps (e, l), une période de validité d'un ou plusieurs jours (suivant l'urgence du client). Et à chaque véhicule, il associe, chaque jour, un dépôt origine et un dépôt destination. A partir de ces instances, nous avons extrait des instances journalières avec un nombre variable de clients obligatoires (compris entre 5 et 9). Ainsi, à partir de chaque instance originale I (avec $I \in \{C1\_1; C1\_2; C1\_3; C1\_4; C1\_5\}$), nous avons extrait les instances $I\_m(n)$ (avec $m \in \{5; 6; 7; 8; 9\}$) comprenant m clients obligatoires et n clients (obligatoires et optionnels). Dans ces instances, nous avons conservé les trois véhicules ainsi que la journée de travail de huit heures (480 minutes), mais nous avons modifié les fenêtres de temps des clients obligatoires en attribuant à la première moitié des clients obligatoires le matin (fenêtre de temps $[0; 240]$) et aux autres l'après-midi (fenêtre de temps $[240; 480]$). Pour procéder à l'extraction des instances, étant donnée une instance initiale I, un nombre de clients obligatoires m et un nombre total de clients n, nous avons d'abord identifié les clients obligatoires et optionnels de cette instance. Puis, nous avons conservé les m premiers clients obligatoires et avons ajouté les $n - m$ premiers clients optionnels, n'apparaissant pas déjà dans la liste des clients obligatoires. En procédant ainsi, nous construisons l'instance $I\_m(n)$.

CHAPITRE 3

Etat de l'art

Dans le chapitre précédent, nous avons défini notre problème comme étant une variante du problème de tournées de service avec omniscience des techniciens, priorité entre les clients, plusieurs dépôts, capacité infinie, et temps de parcours et de service stochastiques. Dans ce chapitre, nous faisons un survol des principales méthodes de résolution proposées pour le problème de tournées de service de techniciens et pour des problèmes de tournées de véhicules (VRP) qui s'en rapprochent. A cet effet, on note que notre problème présente de nombreuses ressemblances avec le problème de tournées de véhicules avec fenêtres de temps (VRPTW). En effet, le VRPTW peut être formulé comme suit. Etant donné un dépôt, des véhicules de capacité limitée et des demandes clients, il s'agit de trouver des routes pour satisfaire toutes les demandes, en s'assurant que chacune de ces routes a pour origine et destination le dépôt donné et que les fenêtres de temps sont respectées. L'objectif est de minimiser la distance parcourue et parfois le nombre de véhicules utilisés. Notre problème peut donc être assimilé à un VRPTW sélectif (il n'est pas nécessaire de desservir tous les clients), avec plusieurs dépôts, capacité infinie et temps de parcours et de service stochastiques. Dans ce chapitre, nous présenterons d'abord les variantes du VRPTW et du problème de tournées de service avec priorité entre les clients ou stochasticité (car ce sont les deux spécificités de notre problème). Nous donnerons ensuite un aperçu des différentes méthodologies proposées pour résoudre ces variantes. Enfin, nous ferons une synthèse de cet état de l'art et nous introduirons l'approche de résolution globale proposée dans cette thèse.

3.1 Variantes similaires à notre problème

Ici, nous nous sommes intéressés aux variantes du problème de tournées de service et du VRPTW prenant en compte une spécificité de notre problème : priorité entre les clients ou des temps de service et/ou de parcours stochastiques. Nous les classerons suivant deux aspects majeurs du problème : la priorité entre les clients et l'aspect stochastique.

3.1.1 Priorité entre les clients

Qu'il s'agisse de planifier des tournées de service ou des tournées de véhicules, les entreprises peuvent attacher plus d'importance à certains clients qu'à d'autres pour de nombreuses

raisons (fidélité, exigences du client...). Elles peuvent donc être amenées à définir une priorité entre les clients. Ainsi, Zeimpekis *et al.* [74], Borenstein *et al.* [9], Branchini *et al.* [13], Cortés *et al.* [18] et Alsheddy et Tsang [1] associent à chaque client une priorité dès que la requête de ce dernier entre dans le système. De même, Angelelli *et al.* [3] et Hadjiconstantinou et Roberts [34] affectent à chaque client une période de validité plus ou moins grande suivant sa priorité dès son arrivée dans le système. Quant à eux, Petrakis *et al.* [54] établissent une priorité entre les clients en leur associant une pénalité de retard. Ceschia *et al.* [15] et Rasmussen *et al.* [56] définissent également une priorité entre les clients en leur associant une pénalité. Plus précisément, ils associent aux clients un coût de non desserte. Tandis que Rasmussen *et al.* [56] associent un coût de non desserte à tous les clients, Ceschia *et al.* [15] distinguent deux types de clients : les clients obligatoires et les clients optionnels, et n'associent un coût de non desserte qu'aux clients optionnels. Dans le cadre du problème du déploiement en temps réel d'une flotte d'ambulances (où les clients considérés sont en réalité des patients), Gendreau *et al.* [31] et Haghani et Yang [35] associent à chaque patient une priorité dépendant de son état de santé. Cette priorité est définie dès qu'une requête entre dans le système mais peut évoluer au cours du temps (comme la santé du patient peut se détériorer). La priorité des clients est donc dynamique. Enfin, Tricoire [68], Dugardin [25], Bostel *et al.* [10], Tricoire *et al.* [69] et Delage [21] proposent une priorité définie sur le type de requête, comme dans notre cas. Ils distinguent deux types de clients : les différables et les rendez-vous. Les requêtes différables correspondent à des opérations planifiées par l'entreprise (comme des opérations de maintenance, des relevés de compteurs...). Celles-ci sont connues a priori et peuvent être différées (elles ne sont pas obligatoires). Au contraire, les rendez-vous correspondent aux requêtes provenant des clients (le plus souvent, il s'agit de demande d'intervention suite à une panne). Ces requêtes apparaissent au cours du temps et sont obligatoires. Elles ont une fenêtre de temps qui leur est associée et on ne peut les différer.

3.1.2 Stochasticité

Dans notre problème, on a considéré deux types d'aléas : ceux sur les temps de parcours et ceux sur les temps de service. Dans la littérature sur le problème de tournées de véhicules classique et ses principales variantes, ce sont principalement les temps de parcours qui sont considérés comme stochastiques. A cet effet, diverses lois de probabilité ont été proposées pour modéliser les temps de parcours stochastiques : la loi standard [75], la loi normale [39], [61], [65] et la distribution gamma [59], [14], [64]. En 2007, Topaloglu [67] traite du VRP dans lequel les temps de parcours suivent une loi de probabilité quelconque et propose un modèle de programmation dynamique. Quant à Shen *et al.* [62], Shao *et al.* [61], Tavakkoli-Moghaddam *et al.* [65] et Zhang *et al.* [75], ils associent au dépôt une fenêtre de temps,

modélisent le problème comme un modèle avec contrainte en probabilité où la probabilité que tous les véhicules soient de retour au dépôt avant la fin de la journée est supérieure à un seuil donné, et minimisent la distance totale parcourue. Certains auteurs proposent de résoudre le VRP avec fenêtres de temps et temps de parcours stochastiques. Ainsi, Ando et Taniguchi [2], Russell et Urban [59], Jie [39], et Tas *et al.* [64] formulent le problème comme un programme à variables entières. Ils cherchent à minimiser la somme pondérée de la distance totale parcourue et des pénalités liées à la violation des fenêtres de temps. De plus, Ando et Taniguchi [2] et Russell et Urban [59] minimisent le nombre de véhicules utilisés. Pour résoudre le TSPTW avec temps de parcours stochastiques, Jula *et al.* [41] définissent un niveau de confiance associé à chaque client comme étant la probabilité d'arriver à ce noeud avant la fin de la fenêtre de temps et s'assurent que cette probabilité est suffisante tout au long de la résolution.

Dans la littérature sur le problème de tournées de véhicules classique et ses principales variantes, seul Xu [73] considère des temps de service stochastiques. Pour la modélisation des temps de service, il reste générique en ne spécifiant aucune loi de distribution. Son objectif est de minimiser l'espérance du temps passé par les requêtes dans le système (ce temps inclut le temps de service).

Dans la littérature du problème de tournées de service, la prise en compte de temps de service stochastiques est plus fréquente. Différentes lois de probabilité ont été proposées pour modéliser les aléas sur les temps de service, notamment la loi normale [48], lognormale [34], la loi de Weibull [19], [63] et la loi de distribution triangulaire [9]. De plus, certaines variantes (avec ou sans fenêtres de temps) du VRP classique ont été traitées avec aléas sur les temps de service. Ainsi, Hadjiconstantinou et Roberts [34] et Lei *et al.* [48] prennent en compte uniquement une fenêtre de temps au dépôt. Tous deux modélisent le problème comme un modèle de programmation stochastique à deux étapes avec recours. Tandis que Lei *et al.* [48] proposent un seul recours pour pénaliser l'éventuel retard au dépôt, Hadjiconstantinou et Roberts [34] proposent, qui plus est, un recours consistant à retourner au dépôt dès que la fin de la fenêtre de temps au dépôt est atteinte. Cortés *et al.* [19], Borenstein *et al.* [9], Delage [21] et Souyris *et al.* [63], quant à eux, traitent du VRPTW. Delage [21] présente un modèle de programmation stochastique à deux étapes avec recours (où le recours consiste à allouer des pénalités d'attente et de retard). Quant à Borenstein *et al.* [9], ils modélisent le VRPTW comme un problème d'ordonnancement avec contraintes de ressource. Souyris *et al.* [63] proposent un modèle d'optimisation robuste.

La stochasticité des temps de service et des temps de parcours a été introduite par Laporte *et al.* [46]. Ces derniers restent très généraux en considérant le problème de tournées de véhicules classique sans spécifier de loi de distribution. Ils proposent trois modèles distincts :

un modèle avec contrainte en probabilité ainsi que deux modèles stochastiques avec recours. Une approche inhabituelle pour les aléas sur les temps de service et de parcours est présentée par Dugardin [25] dans le cadre du problème de tournées de service : il n'anticipe pas l'incertitude en utilisant des outils d'optimisation stochastiques, mais préfère attendre que l'aléa se produise avant de réagir (en procédant à une réoptimisation déterministe). Dans le cadre du problème de tournées de véhicules, Kenyon [43], Kenyon et Morton [44], Wang et Regan [72] et Zeimpekis *et al.* [74] proposent de prendre en compte simultanément les temps de parcours et de service stochastiques. Dans ce contexte, Kenyon [43], Kenyon et Morton [44] minimisent l'heure de fin espérée tandis que Zeimpekis *et al.* [74] proposent de maximiser le nombre de clients servis et Wang et Regan [72] minimisent les coûts espérés. Quant à aux, Teng *et al.* [66] et Li *et al.* [49] formulent le problème comme un modèle de programmation stochastique à deux étapes avec recours où le recours consiste à attribuer des pénalités pour le retard, les temps de service... Li *et al.* [49] proposent aussi un modèle avec contrainte en probabilité.

3.2 Méthodes de résolution

3.2.1 Heuristiques basées sur l'affectation

Des heuristiques simples basées sur la résolution du TSP ont été proposées pour résoudre le problème de gestion de flotte de techniciens en temps réel. En 1994, Xu [73] propose une heuristique appelée « Part-TSP » consistant, pour le problème à k véhicules, à partitionner le territoire en k zones géographiques. Ensuite, il affecte un technicien à chaque zone. Dès que les zones géographiques ont été attribuées aux techniciens, un TSP est résolu pour chaque couple technicien-zone (si un technicien n'a plus de requêtes à servir à un instant donné, il se repositionne au centre de sa zone). Quant à Borenstein *et al.* [9], ils décomposent le problème en plusieurs sous-problèmes résolus à l'aide d'heuristiques. Ils partitionnent donc les clients en plusieurs groupes (correspondant à des zones géographiques) avec un algorithme de clustering : le « k-means algorithm ». Ils affectent ensuite les techniciens aux zones avec une heuristique. Puis, les frontières des zones sont rendues floues : chaque client localisé à la frontière de plusieurs zones appartiendra à toutes les zones frontalières simultanément. Enfin, ils utilisent des règles d'affectation spécifiques pour affecter les tâches aux techniciens. Plus récemment, Petrakis *et al.* [54] ont proposé des heuristiques basées sur l'insertion à moindre coût et sur un modèle d'affectation. Ils y ajoutent une méthode de post-optimisation de type recherche locale à voisinage variable.

3.2.2 Métaheuristiques

Recherche taboue

La recherche taboue [33] a été très utilisée dans la littérature pour résoudre le problème de gestion de flotte en temps réel. Dans cette méthode, on entretient une liste taboue à chaque itération pour éviter les cycles. La recherche taboue peut être résumée comme suit : à partir d'une solution initiale, on génère des solutions voisines. Si le déplacement vers la meilleure de ces solutions n'est pas un mouvement tabou, on met à jour la solution et la liste taboue. On continue jusqu'à ce qu'une condition d'arrêt soit remplie. Dans le cadre du problème de relocalisation d'ambulances, Gendreau *et al.* [31] proposent d'utiliser une recherche taboue pour calculer à l'avance des stratégies de redéploiement (afin que, lors de l'arrivée d'un appel, on ait juste à choisir une stratégie). Ainsi, quand le patient urgent appelle, l'ambulance la plus proche est envoyée sur place. Il peut s'agir d'une ambulance en route pour servir un autre client. Néanmoins, une ambulance en route pour aller servir un client sera déviée de ce client seulement si ce dernier peut être desservi par une autre ambulance dans le temps imparti. Une autre variante de la recherche taboue est utilisée par Russell et Urban [59] : il s'agit d'une recherche taboue dans laquelle on maintient un pool contenant les meilleures solutions obtenues. Ces solutions sont utilisées comme point de départ pour la recherche. Quant à Tas *et al.* [64], ils présentent une recherche taboue dans laquelle ils utilisent une mémoire à moyen terme comme mécanisme d'intensification (si la meilleure solution réalisable reste inchangée pendant un certain nombre d'itérations, elle devient la solution courante). Ceschia *et al.* [15] proposent une recherche taboue et considèrent les trois opérateurs de voisinage suivants : insertion, échange intra-route et échange inter-route. De plus, ils optent pour une liste taboue de longueur dynamique. Enfin, Shen *et al.* [62] et Li *et al.* [49] proposent une heuristique basée sur la recherche taboue. Li *et al.* [49] construisent d'abord une solution déterministe en utilisant l'algorithme de Clarke and Wright. Ensuite, ils appliquent une recherche taboue avec un voisinage basé sur le choix aléatoire entre les opérateurs : 2-opt, relocalisation et échange. Quant à Shen *et al.* [62], ils utilisent comme opérateurs de voisinage : 2-opt, relocalisation, échange, insertion d'un client absent de la solution ou échange d'un client absent de la solution avec un client ou une séquence de clients.

Algorithmes évolutionnistes

L'algorithme génétique [36] est une stratégie évolutionniste consistant à faire évoluer une population de solutions. Cette méthode procède en plusieurs itérations, chacune d'elles se décomposant en trois phases : la sélection des parents, le croisement, la mutation et la mise à jour de la population. La première phase permet, étant donnée une population de solutions,

de sélectionner les solutions parents à l'aide d'une fonction d'évaluation. Dans la phase de croisement, on procède au croisement des solutions parents précédemment sélectionnées. Les solutions ainsi obtenues subissent des mutations. On obtient alors un ensemble de solutions filles. Parmi ces solutions filles, on sélectionne les meilleures et on met à jour la population de solutions. Jie [39] applique cette méthode de résolution au problème de tournées de véhicules avec temps de parcours stochastiques.
L'algorithme mémétique [53] est un algorithme génétique hybridé avec une recherche locale (après la mutation, une recherche locale commence pour choisir la nouvelle population de parents). Comme ils considèrent un problème multi-périodes, Tricoire [68] et Bostel *et al.* [10] proposent d'utiliser un algorithme mémétique à horizon glissant. Les solutions initiales sont considérées comme vides. Les solutions filles héritent d'une ou plusieurs routes de leurs parents. Une solution fille peut contenir une route vide. Néanmoins, cet opérateur de croisement peut être problématique étant donné qu'un seul client peut être servi deux fois dans une solution fille. On considère que, lors de la construction d'une solution fille, un client déjà desservi dans la partie construite ne peut être desservi une deuxième fois ; il sera donc simplement enlevé des autres routes héritées. De plus, si une requête est desservie dans les solutions parents mais pas dans la solution fille, on procèdera à la meilleure insertion de ce client dans cette dernière. Nous avons mentionné plus haut qu'une solution fille pouvait contenir une route vide. Cette route peut servir à insérer les clients qui n'étaient pas servis dans les solutions parents (pour diversifier les solutions). Dès que les solutions filles sont construites, elles sont améliorées à l'aide d'une recherche locale et évaluées sur le critère de distance. Les meilleures d'entre elles sont sélectionnées pour constituer la nouvelle génération.
La recherche dispersée [51] est une méthode dans laquelle on gère un pool de solutions. A chaque itération, on crée de nouvelles solutions en combinant les solutions courantes, puis on améliore ces nouvelles solutions à l'aide d'heuristiques. Ensuite, on ajoute les meilleures solutions ainsi générées au pool de solutions. Zhang *et al.* [75] adaptent cette méthode à leur problème de type VRP avec temps de parcours stochastiques.

Optimisation par essaims particulaires

L'optimisation par essaims particulaires est une métaheuristique créée par Kennedy et Eberhart [42]. Elle consiste, étant donnée une population initiale de particules, à autoriser ces particules à se déplacer afin de converger vers des optima locaux. Néanmoins, cette méthode tend à rester bloquée dans un optimum local au lieu de converger vers un optimum global. C'est pourquoi Shao *et al.* [61] l'hybrident avec une recherche locale.

Recuit simulé

L'algorithme de recuit simulé [45] est une méthode inspirée de la métallurgie. Elle consiste, à chaque itération, à considérer une solution choisie aléatoirement dans le voisinage de la solution courante. La solution courante est ensuite mise à jour dans deux cas : la nouvelle solution est meilleure que l'ancienne, ou un critère probabiliste est respecté. Dans le cadre de leur problème de tournées de véhicules avec temps de service stochastiques, Tavakkoli-Moghaddam *et al.* [65] proposent d'hybrider la méthode de recuit simulé avec des opérateurs de voisinage de l'algorithme génétique : la mutation et le croisement.

3.2.3 Autres méthodes basées sur la recherche locale

Pour résoudre le problème de gestion de flotte en temps réel, Branchini *et al.* [13] proposent une recherche locale granulaire. Dans ce but, ils construisent d'abord une solution initiale, à laquelle ils appliquent une recherche locale. Comme ils veulent s'assurer la couverture de tout le territoire et équilibrer la charge de travail entre les véhicules, ils proposent de construire un ensemble de K clients (où K correspond au nombre de véhicules) dispersés géographiquement. Dès que cet ensemble est construit, ils affectent chacun de ces clients à un véhicule et insèrent les autres clients sur les routes des véhicules. Ensuite, ils appliquent une recherche locale avec seuil de granularité pour résoudre le VRP dynamique (avec de nouveaux clients apparaissant au cours du temps). En d'autres termes, ils limitent la recherche locale à une liste de clients définie par un seuil de granularité. Ces clients sont tous situés dans un rayon égal au seuil de granularité. Du fait de l'aspect dynamique du problème, la liste de clients et le seuil de granularité peuvent varier au cours du temps. Pour éviter de rester bloqués dans des optima locaux avec la recherche locale, Alsheddy et Tsang [1] proposent une recherche locale guidée. Il s'agit d'une recherche locale où l'on ajoute des pénalités dans la fonction d'évaluation afin d'éviter les optima locaux.

La recherche à voisinage variable [52] est une variante de la recherche locale très utilisée pour résoudre le problème de tournées de véhicules. Dans cette méthode, on dispose d'un ensemble de structures de voisinage et on procède itérativement. A chaque itération, on choisit une structure de voisinage différente et on choisit au hasard une solution voisine de la solution courante. Ensuite, on applique à cette solution voisine une recherche locale. Si la solution ainsi obtenue est meilleure que la solution actuelle, on met à jour cette dernière. Angelelli *et al.* [3] proposent une variante de cette méthode avec trois opérateurs de voisinage (échange, relocalisation et insertion). Ils l'intègrent dans un contexte en temps réel dans la mesure où des réoptimisations ont lieu chaque jour et à intervalles de temps réguliers durant la journée. Dans le cadre d'un Team Orienteering avec temps de parcours et de service stochastiques,

Campbell *et al.* [14] prennent en compte deux opérateurs de voisinage supplémentaires : le « 1-shift » et le « ruin and recreate ». Lei *et al.* [48] prennent en compte 6 opérateurs de voisinage : 2-opt, échange et insertion (versions inter et intra-routes). De plus, ils intègrent un mécanisme de granularité dans la recherche locale. Flatberg *et al.* [27] présentent un algorithme basé sur la recherche locale itérative. Cet algorithme est lancé à chaque fois qu'un événement (arrivée d'une nouvelle requête, mise à jour d'une requête existante, arrivée chez un client...) a lieu et gère un pool de solutions. Quand une nouvelle requête apparaît, on modifie la définition du problème et on met à jour le pool de solutions (les solutions devenues irréalisables sont réparées si possible ou enlevées du pool). La procédure suivante est itérée sur le pool : on sélectionne une solution dans le pool et on crée un scénario (contenant les éventuelles futures requêtes). Puis, on ajoute ces requêtes à celles de la solution considérée. On résout un VRP sur toutes ces requêtes (celles de la solution et celles que l'on vient d'ajouter) en utilisant une recherche locale itérative. On obtient ainsi une nouvelle solution qui prend en compte l'arrivée stochastique de nouvelles requêtes. On enlève de la solution les requêtes non apparues et on évalue la solution. Selon la qualité de la solution ainsi obtenue, on l'ajoute ou non au pool. Si on l'ajoute, on la compare à la solution courante et on met à jour cette dernière si nécessaire.

3.2.4 Simulation de Monte Carlo et modèle déterministe

Etant donné un problème avec variables aléatoires, la simulation de Monte Carlo permet de générer une loi de distribution des solution obtenues. Cette méthode consiste à échantillonner la loi de probabilité des variables aléatoires et à procéder à des simulations pour chacun de ces échantillons, obtenant ainsi un ensemble représentatif de solutions possibles (avec une probabilité associée à chacune d'elles). Dans le cadre du problème de tournées de techniciens avec temps de service stochastiques, Delage [21] combine cette méthode avec la recherche taboue : la simulation de Monte Carlo leur permet d'évaluer correctement la qualité d'une solution (en observant les retards pouvant être obtenus du fait de la stochasticité des temps de service) dans la recherche taboue. Dans cette dernière, ils construisent d'abord une solution initiale en associant à chaque client obligatoire le dépôt le plus proche. Ensuite, ils établissent les routes des véhicules (chaque route est associée à un dépôt) en triant les clients obligatoires de chaque dépôt par milieu de fenêtre de temps croissant. Puis, ils insèrent les clients différables (optionnels) jusqu'à ce que toutes les routes soient de longueur maximale. Dès que la solution initiale est construite, la recherche taboue peut commencer. Dans cette recherche taboue, les opérateurs de voisinage sont l'insertion et la suppression ; une solution est évaluée selon le nombre de requêtes satisfaites et le retard estimé par la simulation de Monte Carlo. Cette procédure de recherche taboue est relancée à chaque nouvel événement.

Quant à Kenyon [43] et Kenyon et Morton [44], ils proposent de combiner la simulation de Monte Carlo avec une méthode de résolution appelée DESVRP. Ils prennent en compte simultanément les temps de service et de parcours stochastiques. La simulation de Monte Carlo leur permet d'échantillonner les lois de probabilité pour résoudre ensuite un problème déterministe pour chacun de ces échantillons. Néanmoins, ce problème reste difficile à résoudre avec une méthode de type Branch and Bound du fait du nombre exponentiel de contraintes d'élimination de sous-tours. Ils proposent donc une méthode de type Branch and Cut.

3.2.5 Méthodes d'optimisation stochastique

D'autres méthodes d'optimisation ont été proposées pour prendre en compte les aspects stochastiques du problème. Ainsi, Teng *et al.* [66] proposent une adaptation de la méthode appelée integer L-shaped [47]. Elle consiste (pour un modèle de programmation stochastique à deux étapes avec recours) à résoudre le problème de première étape. Ce problème consiste à trouver une route maximisant le profit (desservant autant de clients que possible) tout en respectant la contrainte de retour au dépôt. Si cette route contient des sous-tours, on ajoute itérativement des contraintes d'élimination de sous-tours et on résout à nouveau. Ensuite, on ajoute les contraintes de faisabilité pour la deuxième étape puis les coupes d'optimalité.

3.2.6 Autres méthodes de résolution

Pour résoudre le problème de gestion de flotte en temps réel, avec temps de service stochastiques, Cortés *et al.* [19] proposent une méthode d'optimisation robuste. Ils optimisent donc le problème dans le pire des cas (où le pire cas correspond à la plus grande déviation totale des temps de service pour un technnicien et non aux temps de service maximaux pour chaque client). Ce problème, rendu déterministe par la formulation robuste, est ensuite résolu avec une génération de colonnes. Dans cette dernière, le problème maître consiste à sélectionner les meilleures routes parmi celles construites par le sous-problème. Tricoire [68], Bostel *et al.* [10] et Tricoire *et al.* [69] se sont intéressés à la variante déterministe et multi-périodes du problème qui nous intéresse et proposent de le résoudre sur un horizon roulant en utilisant, eux aussi, une méthode de génération de colonnes. Cette fois, le sous-problème n'est pas résolu de façon exacte mais à l'aide d'une heuristique. De surcroît, afin d'obtenir une solution entière, la génération de colonnes est intégrée à une méthode d'énumération implicite : le Branch and Price [6]. Dans le cadre de leur problème de tournées de service à domicile, Rasmussen *et al.* [56] proposent eux aussi une méthode de génération de colonnes intégrée dans un Branch and Price. Toutefois, afin de réduire les temps de calcul au niveau du sous-problème, ils y intègrent une méthode de clustering afin de réduire le graphe du sous-

problème (en attribuant un cluster de clients à chaque véhicule). De même, Souyris *et al.* [63] proposent une méthode de type Branch and Price pour le problème de tournées de service avec temps de service stochastiques, dans laquelle le sous-problème est robuste et résolu avec un solveur de programmation par contraintes.

En 1992, Laporte *et al.* [46] proposent une méthode exacte de type Branch and Cut pour résoudre un VRP avec temps de parcours et de service stochastiques. Une variante inhabituelle du Branch and Bound est introduite par Hadjiconstantinou et Roberts [34]. Il s'agit d'une méthode de Branch and Bound appliquée à deux arbres de décisions correspondant aux deux étapes du modèle de programmation stochastique avec recours. Haghani et Yang [35] et Topaloglu [67] proposent une méthode de résolution exacte dans un contexte temps réel. Ainsi, Haghani et Yang [35] optimisent exactement (avec Cplex) à chaque nouvel événement ou à intervalle de temps régulier. Quant à Topaloglu [67], il transforme le modèle de programmation dynamique en plusieurs sous-problèmes approchés (chacun de ces sous-problèmes correspondant à une étape de programmation dynamique). Ces sous-problèmes, formulés comme des programmes en nombres entiers, sont ensuite résolus exactement. Pour traiter du problème de gestion de flotte en temps réel, Zeimpekis *et al.* [74] considèrent un seul véhicule avec des aléas sur les temps de service et de parcours stochastiques. Lorsqu'un nouvel événement survient (retard d'un véhicule), le problème consiste à modifier le planning afin de servir les clients les plus importants. La méthode qu'ils proposent, pour ce faire, est de vérifier, dans un premier temps, s'il est encore possible de desservir tous les clients de la solution courante (par rapport aux fenêtres de temps). Ensuite, ils évaluent les clients en observant leur profit et leur fenêtre de temps et insèrent le meilleur client possible sur la route. Cette étape d'évaluation-insertion est répétée jusqu'à ce que tous les clients soient insérés ou qu'il ne soit plus possible d'insérer des clients sur cette route. Cette méthode est très similaire à la méthode appelée S-algorithm [70].

Enfin, Jula *et al.* [41] et Delage [21] proposent des méthodes basées sur la programmation dynamique. Tandis que Jula *et al.* [41] utilisent la programmation dynamique pour résoudre un TSPTW avec temps de parcours stochastiques, Delage [21] l'intègre dans une méthode afin de résoudre un MDVRPTW avec temps de service stochastiques et priorité entre les clients. Il distingue deux types de clients : les rendez-vous (obligatoires) et les différables (optionnels). Il suppose que chaque client obligatoire est associé à un véhicule et ne peut en changer, et que tous les clients optionnels sont stockés dans une liste. Ainsi, chaque véhicule dispose d'une liste ordonnée de clients obligatoires à desservir et d'une liste de clients optionnels à servir si possible. Comme les temps de service sont stochastiques, il propose une méthode de programmation dynamique où chaque étape correspond à la fin de service chez un client et où les décisions sont les suivantes : soit on se rend au prochain client obligatoire, soit on va

desservir un client optionnel, soit on retourne au dépôt.

3.3 Tableaux de synthèse

Dans les tableaux ci-dessous, on indique les caractéristiques prises en compte. Il est important de bien distinguer la caractéristique « fenêtre de temps » de celle intitulée « période de validité ». On les distingue comme suit :

- une fenêtre de temps est une plage horaire pour le service du client. Cette plage horaire peut durer jusqu'à une journée de travail.
- une période de validité s'étend sur plusieurs jours. De plus, c'est une sorte de fenêtre de temps dure. Un client ne peut être desservi en dehors de sa période de validité.

Ci-après donc deux tableaux récapitulatifs : le premier regroupe tous les articles que nous avons considéré qui traitent le problème en temps réel et le deuxième tous ceux qui ne traitent pas le problème en temps réel.

Légende pour les tableaux qui suivent :
(0) Les fenêtres de temps correspondent ici à l'heure de début de service au plus tôt et l'heure de fin de service au plus tard.
(1) Ici, tous les clients ont des périodes de validité ne dépassant pas deux jours.
(2) Il s'agit de l'heure à laquelle tous les véhicules sont rentrés au dépôt.
(3) Les notations correspondent à l'aspect stochastique pris en compte : TP pour temps de parcours et TS pour temps de service.
(4) Il s'agit des pénalités liées au fait de ne pas servir un client le premier jour de sa période de validité, et de l'écart entre la charge de travail quotidienne réelle et la charge de travail quotidienne moyenne (calculée avec l'historique).
(5) Il s'agit du temps moyen que les requêtes passent dans le système (temps avant le début du service + temps de service).
(6) Les différents types de modélisation sont notés : CCP pour modèle avec contrainte en probabilité, MR pour modèle robuste, MS2E pour modèle de programmation stochastique à deux étapes, P. Dyn. pour programmation dynamique et PNE pour programme en nombre entier.

	Caractéristiques génériques			Caractéristiques spécifiques							Fonction Objectif				Modélisation	Méthode de résolution
				Générales			Service clients			Temps réel							
	Problème en temps réel	Fenêtres de temps	Capacité infinie	Multi-dépôt	Retard dépôt	Compétences	Retard Clients	Période de validité	Priorité entre les clients	Aspect multi-période	Aspect stochastique	min. distance parcourue	min. nb de clients non desservis	min. retard	Autres		
Zeimpekis 2007 [2]	✓	✓							✓				✓			N/A	Méthode similaire au S algorithm
Borenstein 2009 [3]	✓	✓(0)	✓	✓		✓			✓		TS(2)		✓			N/A	Heuristique à plusieurs étapes
Branchini 2009 [4]	✓	✓					✓		✓			✓	✓	✓		N/A	Recherche Locale granulaire
Angelelli 2009 [7]	✓		✓					✓		✓		✓	✓			N/A	Recherche Locale à voisinage variable
Hadjiconstantinou 2002 [8]	✓		✓					✓	✓	✓	TS(2)	✓				MS2E(6)	Paired Tree Search Algorithm
Petrakis 2012 [9]	✓	✓	✓	✓	✓	✓	✓		✓			✓		✓		MIP	Heuristiques
Gendreau 2001 [12]	✓								✓						Max. % demande couverte en un rayon r1	N/A	Recherche Taboue Parallèle avec Mémoire Adaptative
Haghani 2007 [13]	✓					✓			✓			✓				PNE(6)	Résolution exacte
Dugardin 2006 [14]	✓	✓(0)	✓	✓		✓		✓	✓	✓			✓			N/A	Politiques de réaction
Delage 2010 [17]	✓	✓(0)	✓	✓				✓	✓		TS(2)	✓	✓			MR, MS2E(6)	Programmation dynamique ; Monte Carlo + Recherche Taboue
Topaloglu 2007 [25]	✓					✓					TP(2)	✓				P.Dyn(6)	Résolution exacte
Jula 2005 [28]	✓	✓	✓								TP, TS(2)	✓				N/A	Programmation dynamique
Kenyon 1998 [34]	✓		✓								TS(2)				min. l'heure de retour au dépôt(2)	MS2E(6)	Simulation de Monte Carlo + DESVRP

Figure 3.1 Tableau récapitulatif : Problèmes traités en temps réel

	Caractéristiques génériques			Caractéristiques spécifiques							Fonction Objectif				Modélisation	Méthode de résolution
				Générales			Service clients			Temps réel							
	Temps réel	Fenêtres de temps	Capacité infinie	Multi-dépôt	Retard dépôt	Compétences	Retard Clients	Période de validité	Priorité entre les clients	Aspect multi-période	Aspect stochastique	min. distance	min. nb de clients non desservis	min. retard	Autres		
Cortés 2010 [5]		✓	✓				✓		✓	✓		✓		✓		N/A	N/A
Alsheddy 2011 [6]		✓	✓	✓		✓			✓				✓			PNE[6]	Recherche locale guidée
Ceschia 2011[10]		✓			✓		✓	✓	✓	✓		✓		✓	min. nb veh.	non linéaire	Recherche taboue
Rasmussen 2012 [11]		✓	✓	✓		✓			✓			✓		✓	max. préf. visites	PNE[6]	Génération de colonnes
Tricoire 2006 [1], Bostel 2008 [15]		✓[0]	✓	✓		✓		✓	✓	✓		✓	✓				Algorithme mémétique et génération de colonnes sur horizon roulant
Tricoire 2011 [16]		✓[0]	✓	✓		✓		✓	✓	✓		✓	✓				Génération de colonnes
Zhang 2012 [18]					✓						TP[2]	✓			min. nb veh.	CCP[6]	Recherche dispersée
Jie 2010 [19]		✓									TP[2]	✓		✓		CCP[6]	Algorithme génétique
Shao 2010 [20]											TP[2]	✓				CCP[6]	Optimisation par essaims particulaires
Tavakkoli 2012 [21]											TP[2]	✓				CCP[6]	Recuit simulé hybride
Russell 2007 [22]		✓					✓				TP[2]	✓		✓	min. nb veh.	N/A	Recherche taboue
Campbell 2011 [23]			✓				✓				TS[2]					N/A	Recherche locale à voisinage variable
Tas 2011 [24]		✓			✓		✓				TS[2]	✓		✓		PNE[6]	Recherche taboue
Shen 2006 [26]											TP[2]	✓	✓			PNE, CCP[6]	Heuristique basée sur la rech. taboue
Ando 2006 [27]		✓			✓		✓				TP[2]	✓		✓	min. nb veh.		Algorithme génétique
Lei 2011 [29]					✓						TS[2]	✓			min. tps service	MS2E[6]	Recherche locale granulaire à voisinage variable
Cortés 2007 [30]		✓	✓				✓				TS[2]	✓		✓		MR[6]	Génération de colonnes
Souyris2012 [31]			✓				✓		✓		TS[2]	✓	✓	✓		CCP[6]	Génération de colonnes
Xu 1994 [32]											TS[2]				min. tps moy.[4]	N/A	Heuristique Part-TSP
Laporte 1992 [33]											TP, TS[2]	✓			min. nb veh.	MS2E,CCP[6]	Branch and Cut algorithm
Kenyon 2003 [35]			✓								TS, TP[2]				min. h de fin[5]	PNE[6]	Simulation de Monte Carlo + DESVRP
Wang 2001 [36]		✓									TS, TP[2]	✓				Pb d'affectation	N/A
Teng 2004 [37]					✓						TS, TP[2]		✓	✓		MS2E[6]	Integer L-shaped Method
Li 2010 [38]		✓			✓		✓				TS, TP[2]	✓		✓		MS2E,CCP[6]	Heuristique basée sur la rech. taboue

Figure 3.2 Tableau récapitulatif : Problèmes qui ne sont pas traités en temps réel

Nous pouvons observer dans ces tableaux que le problème auquel nous nous intéressons est nouveau car il prend en compte simultanément une priorité entre les clients et des temps de parcours et de service stochastiques. Pour résoudre ce problème, on propose une méthodologie en deux étapes : la planification et l'exécution. Au cours de l'étape de planification, on construit des routes avec des clients obligatoires et des clients optionnels en utilisant des estimés connus a priori. Ensuite, dans l'étape d'exécution, on utilise des outils de programmation dynamique pour déterminer la politique optimale à partir des distributions de probabilité des temps de service et de parcours. Dans le chapitre suivant, nous allons prouver que la politique optimale de notre programmation dynamique est une politique de seuil.

CHAPITRE 4

Politique optimale et politique de seuil

Pour résoudre le problème de tournées de service avec temps de parcours et de service stochastiques présenté au chapitre 1, on propose une méthodologie en deux étapes : la planification et l'exécution. Au cours de l'étape de planification, on construit des routes avec des clients obligatoires et des clients optionnels en utilisant des estimés connus a priori. Ensuite, durant l'étape d'exécution, on utilise des outils de programmation dynamique pour déterminer la politique optimale à partir des distributions de probabilité des temps de service et de parcours. Au cours du déroulement des simulations (du déroulement de l'horizon de temps), on peut alors réagir rapidement en temps réel en adoptant la décision définie par la politique optimale ainsi obtenue. Dans ce chapitre, nous allons prouver que la politique optimale de notre programmation dynamique est une politique de seuil. Dans un premier temps, nous présentons le contexte de notre preuve et les deux algorithmes de programmation dynamique proposés. Ensuite, nous donnerons les notations et hypothèses. Enfin, nous montrerons que la politique optimale pour nos deux algorithmes de programmation dynamique est une politique de preuve.

4.1 Contexte et algorithmes proposés

Supposons que l'on dispose d'un ensemble de routes prévues pour une flotte de véhicules, contenant simultanément des clients obligatoires et des clients optionnels. On définit alors la notion de segment comme étant la portion de route entre deux clients obligatoires (les dépôts sont considérés comme des clients obligatoires). A chaque segment est associée une liste de clients optionnels, qui peut être vide (cf. exemple ci-dessous). Aussi, on suppose que les segments sont rangés comme suit (cf. figure 4.1) : le segment $p+1$ a pour origine le client obligatoire $o^{p+1} = d^p$ (client obligatoire correspondant à la destination du segment p). Par exemple, prenons la route prévue $(o, 1, 2, a, 3, 4, 5, 6, b, c, 7, d)$ d'un véhicule, avec o le dépôt origine, d le dépôt destination, a, b et c trois clients obligatoires et $1, 2, 3, 4, 5$ et 6 des clients optionnels. Cette route est composée de quatre segments : le segment $[o, a]$, le segment $[a, b]$, le segment $[b, c]$ et le segment $[c, d]$. A chacun de ces segments est associée une liste ordonnée de clients optionnels : pour $[o, a]$, il s'agit de la liste $\{1, 2\}$; pour $[a, b]$, il s'agit de la liste $\{3, 4, 5\}$; pour $[b, c]$ il s'agit de la liste vide, et pour $[c, d]$ de la liste $\{7\}$.

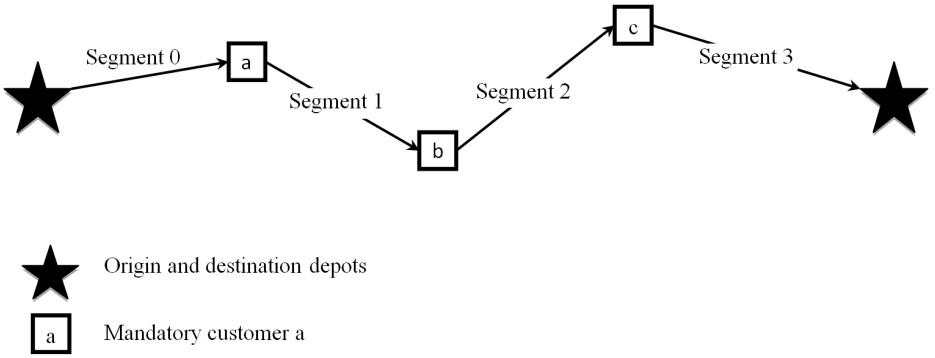


Figure 4.1 Ordre des segments pour un véhicule donné (route)

On suppose également que l'objectif est de maximiser la somme des profits associés à la desserte des clients optionnels, tout en minimisant la somme pondérée du retard chez les clients obligatoires et des temps de parcours. Pour faire face aux temps de parcours et de service stochastiques, on utilise, pour chaque véhicule de la flotte, des outils de programmation dynamique pour déterminer la politique optimale (qui servira ensuite à modifier la route prévue en temps réel). Nous supposons que chaque étape de la programmation dynamique correspond à la fin de service chez un client (obligatoire ou non) et nous proposons alors deux algorithmes de programmation dynamique.

Dans le premier algorithme, on ne considère que le segment en cours. L'objectif est de maximiser le profit espéré sur le segment courant. Soit p ce segment, v le client courant et d^p le client destination du segment p, deux choix sont alors possibles : soit on se rend directement chez le client destination du segment d^p, soit on visite un client optionnel, dont le service est prévu entre v et d^p (cf. figure 4.2).

Dans le deuxième algorithme, on considère le reste de la route. Les choix sont alors les mêmes que ci-dessus sauf que l'objectif n'est plus de maximiser le profit espéré sur le segment courant mais plutôt de maximiser le profit espéré sur le reste de la route.

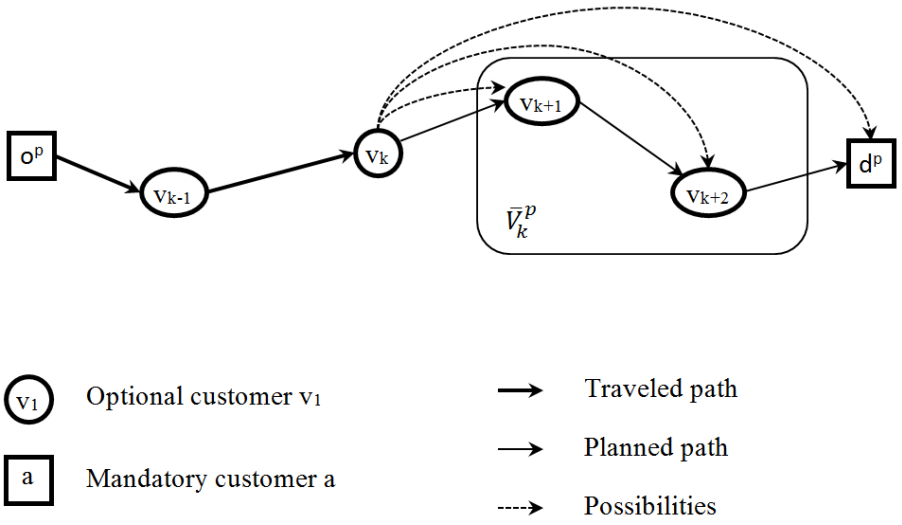


Figure 4.2 Etape k de la programmation dynamique

4.2 Notations

Soit un segment $\{o, v_1, v_2, ..., v_N, d\}$, où $\{v_1, v_2, ..., v_N\}$ est la liste ordonnée des clients optionnels associés à ce segment. On utilisera les notations suivantes :

- Paramètres généraux :

N	ensemble des clients, obligatoires ou optionnels
O	ensemble des clients optionnels
p_{v_i}	profit associé à la desserte du client optionnel v_i
α	pondération des temps de parcours dans la fonction de revenu
D_{ij}	distance entre les clients i et j
Γ_d	pénalité de retard associée au client destination du segment
$[e_d; l_d]$	fenêtre de temps du client destination du segment
$\underline{\sigma}_i$	temps de service minimal chez le client i
$\overline{\sigma}_i$	temps de service maximal chez le client i
$\underline{\tau}_{ij}$	temps de parcours minimal pour aller de i à j
$\overline{\tau}_{ij}$	temps de parcours maximal pour aller de i à j
$f(d, t, \emptyset)$	profit espéré si on débute le service du client d à t

- Paramètres à l'étape k :

t_k	heure de fin de service du client v_k
$\bar{V}_k$	ensemble des clients optionnels associés au segment situés après v_k
$f(v_k, t_k, \bar{V}_k)$	profit espéré si on finit le service du client v_k (avec $v_k \neq d$) à t_k

Pour tout client i, le temps de service, noté σ_i, suit une loi de distribution triangulaire discrète entre $\underline{\sigma_i} - 1$ et $\overline{\sigma}_i + 1$, de mode $\hat{\sigma}_i$.

Le temps de parcours unitaire, noté δ, suit une loi de distribution triangulaire discrète entre $\underline{\delta} - 1$ et $\overline{\delta} + 1$, de mode $\hat{\delta}$.

4.3 Hypothèses

On choisit Γ_d tel que : $\quad \forall i \in O, \quad \Gamma_d \geqslant \dfrac{p_i}{E(\sigma_i)}$ (H1)

On suppose que les distances vérifient l'inégalité triangulaire :

$$\forall a, b, c \in N, \quad D_{ab} + D_{bc} \geqslant D_{ac} \tag{H2}$$

On suppose que les temps de parcours sont discrets et distribués comme suit :

$$\forall i, j \in N, \quad P(\tau_{ij} = m) = P\Big(\big\lceil D_{ij}\delta \big\rceil = m\Big) \tag{H3}$$

Comme les temps de parcours sont discrets, on suppose sans perte de généralité que :

$$\forall i, j \in N, \text{ avec } i \neq j, \quad \underline{\tau_{ij}} \geqslant 1 \tag{H4}$$

On suppose que les temps de parcours et les temps de service sont indépendants. (H5)

A partir des hypothèses (H2) et (H3), on a :

$$\forall i, j \in N, E(\tau_{ij}) = \sum_{m=\underline{\tau_{ij}}}^{\overline{\tau}_{ij}} P(\tau_{ij} = m)m = \sum_{m=\underline{\delta}}^{\overline{\delta}} P(\delta = m)\big\lceil D_{ij}\delta \big\rceil$$

Ainsi,

$$\begin{aligned}
E(\tau_{ab}) + E(\tau_{bc}) &= \sum_{m=\underline{\delta}}^{\bar{\delta}} P(\delta = m)\Big(\lceil D_{ab}\delta \rceil + \lceil D_{bc}\delta \rceil\Big) \\
&\geqslant \sum_{m=\underline{\delta}}^{\bar{\delta}} P(\delta = m)\lceil (D_{ab} + D_{bc})\delta \rceil \\
&\geqslant \sum_{m=\underline{\delta}}^{\bar{\delta}} P(\delta = m)\lceil D_{ac}\delta \rceil \\
E(\tau_{ab}) + E(\tau_{bc}) &\geqslant E(\tau_{ac})
\end{aligned}$$

D'où la propriété sur les temps de parcours :

$$\forall a, b, c \in N, \quad E(\tau_{ab}) + E(\tau_{bc}) \geqslant E(\tau_{ac}) \tag{P1}$$

4.4 Algorithme 1 : preuve sur un segment

Rappelons que chaque étape de la programmation dynamique correspond à la fin de service chez un client. On dispose alors d'une liste de clients optionnels pouvant être servis avant le prochain client obligatoire. Deux options sont à envisager à l'étape k : soit on se rend directement chez le prochain client obligatoire, soit on se rend chez le client optionnel de $\bar{V}_k$ qui maximise le profit espéré.
Le premier algortihme de programmation dynamique consiste à ne considérer qu'un seul segment. En notant $R_c(v,t)$ le profit espéré lié au fait de se rendre au client c depuis le client v au temps t (heure de fin de service chez le client v), la fonction de revenu peut alors être formulée comme suit :

$$\begin{aligned}
f(v_k, t_k, \bar{V}_k) &= \max \begin{cases} R_d(v_k, t_k) \\ \max_{\bar{v}\in\bar{V}_k} R_{\bar{v}}(v_k, t_k) \end{cases} \\
&= \max \begin{cases} -\alpha E(\tau_{v_k d}) + E\big(f(d, t_k + \tau_{v_k d}, \emptyset)\big) \\ \max_{\bar{v}\in\bar{V}_k} \Big(p_{\bar{v}} - \alpha E(\tau_{v_k \bar{v}}) + E\big(f(\bar{v}, t_k + \tau_{v_k \bar{v}} + \sigma_{\bar{v}}, \bar{V}_k \backslash \{\bar{v}\})\big)\Big) \end{cases}
\end{aligned}$$

avec $f(d, t, \emptyset) = -\Gamma_d \max(t - l_d, 0)$

Montrons que, dans le cadre de cet algorithme, la politique optimale est une politique de seuil. Pour ce faire, on procèdera par induction sur les clients optionnels et on prouvera les assertions suivantes :

$\mathcal{A}_1(k)$: $\forall \bar{v} \in \bar{V}_k, \quad R_{\bar{v}}(v_k, l_d) \leqslant R_d(v_k, l_d)$
$\mathcal{A}_2(k)$: $\forall \bar{v} \in \bar{V}_k, \quad R_{\bar{v}}(v_k, t)$ est décroissante.
$\mathcal{A}_3(k)$: $\forall \bar{v} \in \bar{V}_k, \forall t, \quad R_{\bar{v}}(v_k, t+1) - R_{\bar{v}}(v_k, t) \leqslant R_d(v_k, t+1) - R_d(v_k, t)$ (en d'autres termes, $R_d(v_k, t) - R_{\bar{v}}(v_k, t)$ est croissante)
$\mathcal{A}_4(k)$: $f(v_k, t, \bar{V}_k)$ est une fonction décroissante du temps.

4.4.1 Client destination d

On se trouve chez le client destination du segment.
Le revenu se calcule alors comme suit :

$$f(d, t, \emptyset) = -\Gamma_d \max(t - l_d, 0) = \begin{cases} 0 & \text{si } t \leqslant l_d \\ -\Gamma_d(t - l_d) & \text{sinon} \end{cases}$$

Ce revenu est bien une fonction décroissante du temps. En effet, elle est décroissante sur chacun des morceaux et $\forall t > l_d, f(d, t, \emptyset) = -\Gamma_d(t - l_d) < 0 = f(d, l_d, \emptyset)$.

4.4.2 Client optionnel v_N (étape $k = N$)

Supposons qu'on se trouve au dernier client optionnel du segment, on se rend donc directement au client destination du segment :

$$f(v_N, t_N, \emptyset) = -\alpha E(\tau_{v_N d}) + \sum_{m=\underline{\tau}_{v_N d}}^{\overline{\tau}_{v_N d}} P(\tau_{v_N d} = m) f(d, t_N + m, \emptyset)$$

Ici, $\bar{V}_N$ est vide, les assertions $\mathcal{A}_1(N)$, $\mathcal{A}_2(N)$ et $\mathcal{A}_3(N)$ sont donc clairement vérifiées. De plus, comme le revenu au client destination est une fonction décroissante du temps, $f(v_N, t_N, \emptyset)$ est aussi une fonction décroissante du temps (et $\mathcal{A}_4(N)$ est vérifiée). On a bien une politique de seuil, étant donné qu'on décide de se rendre au client destination en tout temps.

4.4.3 Client optionnel v_{N-1} (étape $k = N - 1$)

Supposons maintenant que l'on se trouve à l'avant-dernier client optionnel du segment. On a deux possibilités : soit on se rend directement au client destination, soit on visite le dernier client optionnel du segment. Le choix est déterminé par l'équation :

$$f(v_{N-1}, t_{N-1}, \{v_N\}) = \max \begin{pmatrix} R_d(v_{N-1}, t_{N-1}) \\ R_{v_N}(v_{N-1}, t_{N-1}) \end{pmatrix}$$

avec

$$R_d(v_{N-1}, t_{N-1}) = -\alpha E(\tau_{v_{N-1}d}) + \sum_{m=\underline{\tau}_{v_{N-1}d}}^{\overline{\tau}_{v_{N-1}d}} P(\tau_{v_{N-1}d} = m) f(d, t_{N-1} + m, \emptyset)$$

$$R_{v_N}(v_{N-1}, t_{N-1}) = p_{v_N} - \alpha E(\tau_{v_{N-1}v_N}) + \sum_{m=\underline{\tau}_{v_{N-1}v_N}}^{\overline{\tau}_{v_{N-1}v_N}} P(\tau_{v_{N-1}v_N} = m) \sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_N} = n) f(v_N, t_{N-1} + m + n, \emptyset)$$

Ainsi, on ira directement au client destination du segment si $R_d(v_{N-1}, t) \geqslant R_{v_N}(v_{N-1}, t)$. Simplifions l'écriture de $R_d(v_{N-1}, t)$:

$$\begin{aligned} R_d(v_{N-1}, t) &= -\alpha E(\tau_{v_{N-1}d}) + \sum_{m=\underline{\tau}_{v_{N-1}d}}^{\overline{\tau}_{v_{N-1}d}} P(\tau_{v_{N-1}d} = m) f(d, t + m, \emptyset) \\ &= -\alpha E(\tau_{v_{N-1}d}) - \Gamma_d \sum_{m=\underline{\tau}_{v_{N-1}d}}^{\overline{\tau}_{v_{N-1}d}} P(\tau_{v_{N-1}d} = m) \max(t + m - l_d, 0) \end{aligned}$$

$$\boxed{R_d(v_{N-1}, t) = -\alpha E(\tau_{v_{N-1}d}) - \Gamma_d \sum_{m=\max(l_d - t + 1, \underline{\tau}_{v_{N-1}d})}^{\overline{\tau}_{v_{N-1}d}} P(\tau_{v_{N-1}d} = m)(t + m - l_d)} \tag{4.1}$$

Et celle de $R_{v_N}(v_{N-1}, t)$:

$$R_{v_N}(v_{N-1}, t) = p_{v_N} - \alpha E(\tau_{v_{N-1}v_N}) + \sum_{m=\underline{\tau}_{v_{N-1}v_N}}^{\overline{\tau}_{v_{N-1}v_N}} P(\tau_{v_{N-1}v_N} = m) \sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_N} = n) f(v_N, t + m + n, \emptyset)$$

Aussi, d'après l'hypothèse (H5), les temps de service et de parcours sont indépendants donc

$P(\tau_{ij})P(\sigma_j) = P(\tau_{ij} + \sigma_j)$. D'où :

$$\begin{aligned}
R_{v_N}(v_{N-1}, t) &= p_{v_N} - \alpha E(\tau_{v_{N-1}v_N}) \\
&\quad + \sum_{m=\underline{\tau}_{v_{N-1}v_N}+\underline{\sigma}_{v_N}}^{\overline{\tau}_{v_{N-1}v_N}+\overline{\sigma}_{v_N}} P(\tau_{v_{N-1}v_N} + \sigma_{v_n} = m) f(v_N, t+m, \emptyset) \\
&= p_{v_N} - \alpha E(\tau_{v_{N-1}v_N}) + \sum_{m=\underline{\tau}_{v_{N-1}v_N}+\underline{\sigma}_{v_N}}^{\overline{\tau}_{v_{N-1}v_N}+\overline{\sigma}_{v_N}} P(\tau_{v_{N-1}v_N} + \sigma_{v_n} = m) \Big(- \alpha E(\tau_{v_N d}) \\
&\quad + \sum_{n=\underline{\tau}_{v_N d}}^{\overline{\tau}_{v_N d}} P(\tau_{v_N d} = n) f(d, t+m+n, \emptyset) \Big)
\end{aligned}$$

$$\boxed{\begin{aligned}
R_{v_N}(v_{N-1}, t) &= p_{v_N} - \alpha E(\tau_{v_{N-1}v_N}) - \alpha E(\tau_{v_N d}) \\
&\quad + \sum_{m=\underline{\tau}_{v_{N-1}v_N}+\underline{\sigma}_{v_N}}^{\overline{\tau}_{v_{N-1}v_N}+\overline{\sigma}_{v_N}} P(\tau_{v_{N-1}v_N} + \sigma_{v_n} = m) \sum_{n=\underline{\tau}_{v_N d}}^{\overline{\tau}_{v_N d}} P(\tau_{v_N d} = n) \max(t+m+n, 0)
\end{aligned}} \quad (4.2)$$

Pour montrer que la politique optimale est bien une politique de seuil à l'étape $k = N-1$, on prouvera les assertions $\mathcal{A}_1(N-1)$, $\mathcal{A}_2(N-1)$, $\mathcal{A}_3(N-1)$ et $\mathcal{A}_4(N-1)$ puis on en conclura que l'on a bien une politique de seuil :

Preuve de l'assertion $\mathcal{A}_1(N-1)$

Pour prouver que l'assertion $\mathcal{A}_1(N-1)$ est vraie, montrons que $R_d(v_{N-1}, l_d) \geqslant R_{v_N}(v_{N-1}, l_d)$. Pour ce faire développons l'expression de $R_d(v_{N-1}, l_d)$ puis celle de $R_{v_N}(v_{N-1}, l_d)$

D'après l'équation (4.1), on sait que $R_d(v_{N-1}, l_d)$ vaut :

$$R_d(v_{N-1}, l_d) = -\alpha E(\tau_{v_{N-1}d}) - \Gamma_d \sum_{m=\max(1, \underline{\tau}_{v_{N-1}d})}^{\overline{\tau}_{v_{N-1}d}} P(\tau_{v_{N-1}d} = m) m$$

Or, d'après l'hypothèse (H4), comme $v_{N-1} \neq d$, $\max(1, \underline{\tau}_{v_{N-1}d}) = \underline{\tau}_{v_{N-1}d}$.

On obtient donc :

$$\begin{aligned} R_d(v_{N-1}, l_d) &= -\alpha E(\tau_{v_{N-1}d}) - \Gamma_d \sum_{m=\underline{\tau}_{v_{N-1}d}}^{\overline{\tau}_{v_{N-1}d}} P(\tau_{v_{N-1}d} = m)m \\ &= -\alpha E(\tau_{v_{N-1}d}) - \Gamma_d E(\tau_{v_{N-1}d}) \\ \boxed{R_d(v_{N-1}, l_d) = -(\alpha + \Gamma_d)E(\tau_{v_{N-1}d})} & \end{aligned} \tag{4.3}$$

Calculons maintenant $R_{v_N}(v_{N-1}, l_d)$, à partir de l'équation (4.2) :

$$\begin{aligned} R_{v_N}(v_{N-1}, l_d) &= p_{v_N} - \alpha\big(E(\tau_{v_{N-1}v_N}) + E(\tau_{v_Nd})\big) \\ &\quad - \Gamma_d \sum_{m=\underline{\tau}_{v_{N-1}v_N}+\underline{\sigma}_{v_N}}^{\overline{\tau}_{v_{N-1}v_N}+\overline{\sigma}_{v_N}} P(\tau_{v_{N-1}v_N} + \sigma_{v_N} = m) \sum_{n=\underline{\tau}_{v_Nd}}^{\overline{\tau}_{v_Nd}} P(\tau_{v_Nd} = n) \max(m+n, 0) \end{aligned} \tag{4.4}$$

$$\begin{aligned} &= p_{v_N} - \alpha\big(E(\tau_{v_{N-1}v_N}) + E(\tau_{v_Nd})\big) \\ &\quad - \Gamma_d \sum_{m=\underline{\tau}_{v_{N-1}v_N}+\underline{\sigma}_{v_N}}^{\overline{\tau}_{v_{N-1}v_N}+\overline{\sigma}_{v_N}} P(\tau_{v_{N-1}v_N} + \sigma_{v_N} = m) \sum_{n=\underline{\tau}_{v_Nd}}^{\overline{\tau}_{v_Nd}} P(\tau_{v_Nd} = n)(m+n) \\ &= p_{v_N} - \alpha\big(E(\tau_{v_{N-1}v_N}) + E(\tau_{v_Nd})\big) - \Gamma_d \sum_{n=\underline{\tau}_{v_Nd}}^{\overline{\tau}_{v_Nd}} P(\tau_{v_Nd} = n)n \\ &\quad - \Gamma_d \sum_{m=\underline{\tau}_{v_{N-1}v_N}+\underline{\sigma}_{v_N}}^{\overline{\tau}_{v_{N-1}v_N}+\overline{\sigma}_{v_N}} P(\tau_{v_{N-1}v_N} + \sigma_{v_N} = m)m \end{aligned} \tag{4.5}$$

Aussi, d'après l'hypothèse (H5), les temps de service et de parcours sont indépendants

donc $P(\tau_{v_{N-1}v_N} + \sigma_{v_n}) = P(\tau_{v_{N-1}v_N})P(\sigma_{v_N})$. D'où :

$$\begin{aligned}
R_{v_N}(v_{N-1}, l_d) &= p_{v_N} - \alpha\big(E(\tau_{v_{N-1}v_N}) + E(\tau_{v_Nd})\big) - \Gamma_d E(\tau_{v_Nd}) \\
&\quad - \Gamma_d \sum_{m=\underline{\tau}_{v_{N-1}v_N}}^{\overline{\tau}_{v_{N-1}v_N}} P(\tau_{v_{N-1}v_N} = m) \sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_N} = n)(m+n) \\
&= p_{v_N} - \alpha\big(E(\tau_{v_{N-1}v_N}) + E(\tau_{v_Nd})\big) - \Gamma_d E(\tau_{v_Nd}) \\
&\quad - \Gamma_d \sum_{m=\underline{\tau}_{v_{N-1}v_N}}^{\overline{\tau}_{v_{N-1}v_N}} P(\tau_{v_{N-1}v_N} = m)m - \Gamma_d \sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_N} = n)n \\
&= p_{v_N} - \alpha\big(E(\tau_{v_{N-1}v_N}) + E(\tau_{v_Nd})\big) - \Gamma_d E(\tau_{v_Nd}) \\
&\quad - \Gamma_d\big(E(\tau_{v_{N-1}v_N}) + E(\sigma_{v_N})\big) \\
&= p_{v_N} - (\alpha + \Gamma_d)\big(E(\tau_{v_{N-1}v_N}) + E(\tau_{v_Nd})\big) - \Gamma_d E(\sigma_{v_N})
\end{aligned}$$

Aussi, d'après (H1), $\Gamma_d \geqslant \dfrac{p_{v_N}}{E(\sigma_{v_N})}$ et $p_{v_N} - \Gamma_d E(\sigma_{v_N}) \leqslant 0$. D'où :

$$\begin{aligned}
R_{v_N}(v_{N-1}, l_d) &\leqslant -(\alpha + \Gamma_d)\big[\underbrace{E(\tau_{v_{N-1}v_N}) + E(\tau_{v_Nd})}_{\geqslant E(\tau_{v_{N-1}d}) \text{ (d'après (P1))}}\big] \\
&\leqslant -(\alpha + \Gamma_d)E(\tau_{v_{N-1}d}) \\
\boxed{R_{v_N}(v_{N-1}, l_d) \leqslant R_d(v_{N-1}, l_d))}&
\end{aligned}$$

Preuve de l'assertion $\mathcal{A}_2(N-1)$

Montrons que R_d et R_{v_N} sont décroissantes (assertion $\mathcal{A}_2(N-1)$).

Par définition, $R_d(v,t) = -\alpha E(\tau_{vd}) + \sum_{m=\underline{\tau}_{vd}}^{\overline{\tau}_{vd}} P(\tau_{vd} = m) f(d, t+m, \emptyset)$.

Ainsi, $R_d(v,t+1) - R_d(v,t) = \sum_{m=\underline{\tau}_{vd}}^{\overline{\tau}_{vd}} P(\tau_{vd} = m)\big(f(d,t+1+m,\emptyset) - f(d,t+m,\emptyset)\big)$.

Comme $f(d,t,\emptyset)$ est décroissante, $R_d(v,t)$ l'est également pour tout v.

De même, $R_{v_N}(v,t) = -\alpha E(\tau_{vv_N}) + \sum_{m=\underline{\tau}_{vv_N}}^{\overline{\tau}_{vv_N}} P(\tau_{vv_N} = m) f(v_N, t+m, \emptyset)$. En procédant comme pour $R_d(v,t)$, on montre que, comme $f(v_N,t,\emptyset)$ est décroissante, $R_{v_N}(v,t)$ est décroissante pour tout v.

Preuve de l'assertion $\mathcal{A}_3(N-1)$

Montrons à présent que l'assertion $\mathcal{A}_3(N-1)$ est vraie.
En notant $\Delta R_d(v_{n-1},t) = R_d(v_{N-1},t+1) - R_d(v_{N-1},t)$ et en utilisant l'équation (4.1), on obtient :

$$\Delta R_d(v_{N-1},t) = -\Gamma_d \sum_{m=\max(l_d-t,\underline{\tau}_{v_{N-1}d})}^{\overline{\tau}_{v_{N-1}d}} P(\tau_{v_{N-1}d}=m)(t+1+m-l_d)$$
$$+\Gamma_d \sum_{m=\max(l_d-t+1,\underline{\tau}_{v_{N-1}d})}^{\overline{\tau}_{v_{N-1}d}} P(\tau_{v_{N-1}d}=m)(t+m-l_d)$$

Deux cas sont à envisager :

- Soit $\underline{\tau}_{v_{N-1}d} \geqslant l_d - t + 1$ (ou encore $P(\tau_{v_{N-1}d} \geqslant l_d - t) = 1$).
  On a alors $\max(l_d - t, \underline{\tau}_{v_{N-1}d}) = \max(l_d - t + 1, \underline{\tau}_{v_{N-1}d}) = \underline{\tau}_{v_{N-1}d}$. Ce qui donne

$$\Delta R_d(v_{N-1},t) = -\Gamma_d \sum_{m=\underline{\tau}_{v_{N-1}d}}^{\overline{\tau}_{v_{N-1}d}} P(\tau_{v_{N-1}d}=m) = -\Gamma_d = -\Gamma_d P(\tau_{v_{N-1}d} \geqslant l_d - t)$$

- Soit $\underline{\tau}_{v_{N-1}d} \leqslant l_d - t$.
  On a alors : $\max(l_d - t, \underline{\tau}_{v_{N-1}d}) = l_d - t$ et $\max(l_d - t + 1, \underline{\tau}_{v_{N-1}d}) = l_d - t + 1$. Et donc

$$\Delta R_d(v_{N-1},t) = -\Gamma_d \sum_{m=l_d-t}^{\overline{\tau}_{v_{N-1}d}} P(\tau_{v_{N-1}d}=m)(t+1+m-l_d)$$
$$+\Gamma_d \sum_{m=l_d-t+1}^{\overline{\tau}_{v_{N-1}d}} P(\tau_{v_{N-1}d}=m)(t+m-l_d)$$

  Or, en $m = l_d - t$, $t + m - l_d = 0$. D'où :

$$\sum_{m=l_d-t+1}^{\overline{\tau}_{v_{N-1}d}} P(\tau_{v_{N-1}d}=m)(t+m-l_d) = \sum_{m=l_d-t}^{\overline{\tau}_{v_{N-1}d}} P(\tau_{v_{N-1}d}=m)(t+m-l_d)$$

On en déduit que :

$$\begin{aligned}\Delta R_d(v_{N-1},t) &= -\Gamma_d \sum_{m=l_d-t}^{\overline{\tau}_{v_{N-1}d}} P(\tau_{v_{N-1}d}=m)\Big((t+1+m-l_d)-(t+m-l_d)\Big)\\ &= -\Gamma_d \sum_{m=l_d-t}^{\bar{\tau}_{v_{N-1}d}} P(\tau_{v_{N-1}d}=m)\\ \boxed{\Delta R_d(v_{N-1},t) = -\Gamma_d P(\tau_{v_{N-1}d}\geqslant l_d-t)} &\end{aligned} \tag{4.6}$$

En raisonnant comme ci-dessus, à partir de l'équation (4.2), on obtient :

$$\begin{aligned}&R_{v_N}(v_{N-1},t+1)-R_{v_N}(v_{N-1},t)\\ &= -\Gamma_d \sum_{m=\underline{\tau}_{v_{N-1}v_N}+\underline{\sigma}_{v_N}}^{\overline{\tau}_{v_{N-1}v_N}+\overline{\sigma}_{v_N}} P(\tau_{v_{N-1}v_N}+\sigma_{v_n}=m)\sum_{n=\underline{\tau}_{v_Nd}}^{\overline{\tau}_{v_Nd}} P(\tau_{v_Nd}=n)\max(t+1+m+n-l_d,0)\\ &\quad+\Gamma_d \sum_{m=\underline{\tau}_{v_{N-1}v_N}+\underline{\sigma}_{v_N}}^{\overline{\tau}_{v_{N-1}v_N}+\overline{\sigma}_{v_N}} P(\tau_{v_{N-1}v_N}+\sigma_{v_n}=m)\sum_{n=\underline{\tau}_{v_Nd}}^{\overline{\tau}_{v_Nd}} P(\tau_{v_Nd}=n)\max(t+m+n-l_d,0)\\ &= -\Gamma_d\big(R_{N-1,N,d}(t+1)-R_{N-1,N,d}(t)\big)\end{aligned}$$

$$\boxed{\Delta R_{v_N}(v_{N-1},t) = -\Gamma_d\big(R_{N-1,N,d}(t+1)-R_{N-1,N,d}(t)\big)} \tag{4.7}$$

Montrons que $\forall t, R_{v_N}(v_{N-1},t+1)-R_{v_N}(v_{N-1},t) \leqslant R_d(v_{N-1},t+1)-R_d(v_{N-1},t)$.

- Soit $t \geqslant l_d$
  On a alors $P(\tau_{v_{N-1}d}\geqslant l_d-t)=1$ et $\Delta R_d(v_{N-1},t)=-\Gamma_d$. On sait aussi que $\max(t+1+m+n-l_d,0)=t+1+m+n-l_d$ et $\max(t+m+n-l_d,0)=t+m+n-l_d$, d'où :

$$R_{v_N}(v_{N-1},t+1)-R_{v_N}(v_{N-1},t)=R_d(v_{N-1},t+1)-R_d(v_{N-1},t)=-\Gamma_d$$

- Soit $t < l_d$
  Montrons qu'alors $\big(R_{N-1,N,d}(t+1)-R_{N-1,N,d}(t)\big) \geqslant P(\tau_{v_{N-1}d}\geqslant l_d-t)$. On a :

$$P(\tau_{v_{N-1}d}\geqslant l_d-t)=P(\lceil D_{v_{N-1}d}\delta\rceil \geqslant l_d-t)$$

  Soit δ_1 tel que $\lceil D_{v_{N-1}d}(\delta_1-1)\rceil < l_d-t \leqslant \lceil D_{v_{N-1}d}\delta_1\rceil$.
  Alors, $P(\tau_{v_{N-1}d}\geqslant l_d-t)=P(\delta\geqslant\delta_1)$.

Aussi, $R_{N-1,N,d}(t)$ peut s'écrire :

$$
\begin{aligned}
&R_{N-1,N,d}(t)\\
&=\sum_{m=\underline{\delta}}^{\overline{\delta}} P(\delta=m)\sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_n}=n)\max\Big(t+\lceil D_{v_{N-1}v_N}m\rceil+\lceil D_{v_Nd}m\rceil+n-l_d,0\Big)\\
&=\sum_{m=\underline{\delta}}^{\delta_1-1} P(\delta=m)\sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_n}=n)\max\Big(t+\lceil D_{v_{N-1}v_N}m\rceil+\lceil D_{v_Nd}m\rceil+n-l_d,0\Big)\\
&\quad+\sum_{m=\delta_1}^{\overline{\delta}} P(\delta=m)\sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_n}=n)\max\Big(t+\lceil D_{v_{N-1}v_N}m\rceil+\lceil D_{v_Nd}m\rceil+n-l_d,0\Big)
\end{aligned}
$$

Soit $\delta\geqslant\delta_1$. D'après les propriétés de la fonction plafond, $\lceil a\rceil+\lceil b\rceil\geqslant\lceil a+b\rceil$. D'où :

$$
\begin{aligned}
t+\lceil D_{v_{N-1}v_N}\delta\rceil+\lceil D_{v_Nd}\delta\rceil+n-l_d &\geqslant t+\lceil \underbrace{(D_{v_{N-1}v_N}+D_{v_Nd})}_{\geqslant D_{v_{N-1}d}\text{ d'après (H2)}}\delta\rceil+n-l_d\\
&\geqslant t+\lceil (D_{v_{N-1}d})\delta\rceil+n-l_d\\
&\geqslant t+\lceil (D_{v_{N-1}d})\delta_1\rceil+n-l_d\\
&\geqslant t+(l_d-t)+n-l_d=n>0
\end{aligned}
$$

Donc, $\forall\delta\geqslant\delta_1$, $t+\lceil D_{v_{N-1}v_N}\delta\rceil+\lceil D_{v_Nd}\delta\rceil+n-l_d>0$ Et :

$$
\begin{aligned}
&R_{N-1,N,d}(t)\\
&=\sum_{m=\underline{\delta}}^{\delta_1-1} P(\delta=m)\sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_n}=n)\max\Big(t+\lceil D_{v_{N-1}v_N}m\rceil+\lceil D_{v_Nd}m\rceil+n-l_d,0\Big)\\
&\quad+\sum_{m=\delta_1}^{\overline{\delta}} P(\delta=m)\sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_n}=n)\Big(t+\lceil D_{v_{N-1}v_N}m\rceil+\lceil D_{v_Nd}m\rceil+n-l_d\Big)
\end{aligned}
$$

En utilisant cette expression, recalculons $\big(R_{N-1,N,d}(t+1) - R_{N-1,N,d}(t)\big)$.

$$
\begin{aligned}
&R_{N-1,N,d}(t+1) - R_{N-1,N,d}(t) \\
&= \sum_{m=\underline{\delta}}^{\delta_1-1} P(\delta = m) \sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_n} = n) \max\Big(t+1+\big\lceil D_{v_{N-1}v_N} m\big\rceil + \big\lceil D_{v_N d} m\big\rceil + n - l_d, 0\Big) \\
&\quad + \sum_{m=\delta_1}^{\overline{\delta}} P(\delta = m) \sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_n} = n)\Big(t+1+\big\lceil D_{v_{N-1}v_N} m\big\rceil + \big\lceil D_{v_N d} m\big\rceil + n - l_d\Big) \\
&\quad - \sum_{m=\underline{\delta}}^{\delta_1-1} P(\delta = m) \sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_n} = n) \max\Big(t+\big\lceil D_{v_{N-1}v_N} m\big\rceil + \big\lceil D_{v_N d} m\big\rceil + n - l_d, 0\Big) \\
&\quad - \sum_{m=\delta_1}^{\overline{\delta}} P(\delta = m) \sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_n} = n)\Big(t+\big\lceil D_{v_{N-1}v_N} m\big\rceil + \big\lceil D_{v_N d} m\big\rceil + n - l_d\Big) \\
&= \sum_{m=\underline{\delta}}^{\delta_1-1} P(\delta = m) \sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_n} = n)\Bigg[\max\Big(t+1+\big\lceil D_{v_{N-1}v_N} m\big\rceil + \big\lceil D_{v_N d} m\big\rceil + n - l_d, 0\Big) \\
&\qquad - \max\Big(t+\big\lceil D_{v_{N-1}v_N} m\big\rceil + \big\lceil D_{v_N d} m\big\rceil + n - l_d, 0\Big)\Bigg] \\
&\quad + \sum_{m=\delta_1}^{\overline{\delta}} P(\delta = m) \sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_n} = n)\Bigg[\Big(t+1+\big\lceil D_{v_{N-1}v_N} m\big\rceil + \big\lceil D_{v_N d} m\big\rceil + n - l_d\Big) \\
&\qquad - \Big(t+\big\lceil D_{v_{N-1}v_N} m\big\rceil + \big\lceil D_{v_N d} m\big\rceil + n - l_d\Big)\Bigg] \\
&\geqslant \sum_{m=\delta_1}^{\overline{\delta}} P(\delta = m) \sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_n} = n)\Bigg[\Big(t+1+\big\lceil D_{v_{N-1}v_N} m\big\rceil + \big\lceil D_{v_N d} m\big\rceil + n - l_d\Big) \\
&\qquad - \Big(t+\big\lceil D_{v_{N-1}v_N} m\big\rceil + \big\lceil D_{v_N d} m\big\rceil + n - l_d\Big)\Bigg] \\
&\geqslant \sum_{m=\delta_1}^{\overline{\delta}} P(\delta = m) \sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_n} = n) \\
&\geqslant \sum_{m=\delta_1}^{\overline{\delta}} P(\delta = m)
\end{aligned}
$$

$$\boxed{R_{N-1,N,d}(t+1) - R_{N-1,N,d}(t) \geqslant P(\delta \geqslant \delta_1)} \tag{4.8}$$

On en conclut, d'après (4.6) et (4.7), que :

$$R_{v_N}(v_{N-1}, t+1) - R_{v_N}(v_{N-1}, t) \leqslant R_d(v_{N-1}, t+1) - R_d(v_{N-1}, t)$$

L'assertion $\mathcal{A}_3(N-1)$ est donc bien vérifiée.

Soit s tel que $R_d(v_{N-1}, s) \geqslant R_{v_N}(v_{N-1}, s)$ et soit $t \geqslant s$. D'après $\mathcal{A}_3(N-1)$, on a :

$$\begin{aligned} R_d(v_{N-1}, t) &= R_d(v_{N-1}, s) + \sum_{k=0}^{t-1} \big(R_d(v_{N-1}, s+k+1) - R_d(v_{N-1}, s+k)\big) \\ &\geqslant R_{v_N}(v_{N-1}, s) + \sum_{k=0}^{t-1} \big(R_{v_N}(v_{N-1}, s+k+1) - R_{v_N}(v_{N-1}, s+k)\big) \\ &\geqslant R_{v_N}(v_{N-1}, t) \end{aligned}$$

Donc $\forall t \geqslant s$, on vérifie bien $R_d(v_{N-1}, t) \geqslant R_{v_N}(v_{N-1}, t)$.

On a montré (*) que les fonctions de revenu R_d et R_{v_N} décroissent avec le temps (cf. section 4.4.3) et (**) que $R_d(v_{N-1}, l_d) \geqslant R_{v_N}(v_{N-1}, l_d)$ (cf. section 4.4.3). On sait aussi (***) que s'il existe un s pour lequel $R_d(v_{N-1}, s) \geqslant R_{v_N}(v_{N-1}, s)$, alors $\forall t \geqslant s, R_d(v_{N-1}, t) \geqslant R_{v_N}(v_{N-1}, t)$ (cf. ci-dessus).
Deux cas sont donc à envisager (on suppose que l'instant initial au client v_{N-1} est l'instant 0) :

- Soit $R_d(v_{N-1}, 0) \geqslant R_{v_N}(v_{N-1}, 0)$, auquel cas on préfèrera toujours aller au client destination ($s = -1$).
- Soit $R_d(v_{N-1}, 0) < R_{v_N}(v_{N-1}, 0)$. Auquel cas, d'après (*) et (**), il existe au moins un seuil en lequel $R_d(v_{N-1}, t) - R_{v_N}(v_{N-1}, t)$ change de signe. D'après (***), ce seuil s est unique et vérifie $\forall t \leqslant s$, $R_{v_N}(v_{N-1}, t) \geqslant R_d(v_{N-1}, t)$ et $\forall t > s$, $R_d(v_{N-1}, t) \geqslant R_{v_N}(v_{N-1}, t)$.

Dans tous les cas, on a bien une politique de seuil, le revenu est bien une fonction décroissante du temps ($\mathcal{A}_4(N-1)$ est vérifiée) et il existe bien un seuil unique entre v_N et d.

4.4.4 Induction sur les clients optionnels

Supposons à présent que la politique optimale aux étapes $k+1, k+2, ..., N$ soit bien une politique de seuil, et que les assertions $\mathcal{A}_1, \mathcal{A}_2$, $\mathcal{A}_3$ et $\mathcal{A}_4$ soient vraies à ces étapes.

Montrons qu'alors, ces assertions sont toujours vraies à l'étape k et que la politique optimale à l'étape k est une politique de seuil.
A l'étape k, on dispose de plusieurs possibilités : soit on va directement au client destination, soit on se rend chez un client optionnel de $\bar{V}_k$.

$$f(v_k, t_k, \bar{V}_k) = \max \begin{cases} -\alpha E(\tau_{v_k d}) + \sum_{m=\underline{\tau}_{v_k d}}^{\overline{\tau}_{v_k d}} P(\tau_{v_k d} = m) f(d, t_k + m, \emptyset)) \\ \max_{\bar{v} \in \bar{V}_k} \Big(p_{\bar{v}} - \alpha E(\tau_{v_k \bar{v}}) \\ + \sum_{m=\underline{\tau}_{v_k \bar{v}}}^{\overline{\tau}_{v_k \bar{v}}} P(\tau_{v_k \bar{v}} = m) \sum_{n=\underline{\sigma}_{\bar{v}}}^{\overline{\sigma}_{\bar{v}}} P(\sigma_{\bar{v}} = n) f(\bar{v}, t_k + m + n, \bar{V}_k \backslash \{\bar{v}\}) \Big) \end{cases}$$

Pour montrer que l'on a bien une politique de seuil à l'étape k, on procède en trois temps :

1. Montrer qu'il existe un seuil unique entre tout client optionnel de $\bar{V}_k$ et le client destination
2. Montrer qu'il existe un (ou plusieurs) seuil(s) entre toute paire de clients optionnels de $\bar{V}_k$
3. Montrer que ces seuils définissent une politique de seuil

Seuil unique entre un client optionnel et le client destination

Soient v_k le client courant et c_1 un client optionnel de $\bar{V}_k$, on notera $R_{c_1}(v_k, t)$ le profit espéré si on décide d'aller au client c_1 et $R_d(v_k, t)$ le profit espéré si on se rend au client destination au temps t.
On veut montrer ici qu'il existe un seuil s pour lequel $\forall t \leqslant s$, on va visiter le client c_1 ($R_d(v_k, t) \leqslant R_{c_1}(v_k, t)$) et $\forall t > s$, on se rend en d ($R_d(v_k, t) \geqslant R_{c_1}(v_k, t)$).
Pour ce faire, raisonnons comme précédemment : montrons les assertions $\mathcal{A}_1(k)$, $\mathcal{A}_2(k)$ et $\mathcal{A}_3(k)$.

– Montrons que l'assertion $\mathcal{A}_1(k)$ est vraie, soit $R_d(v_k, l_d) \geqslant R_{c_1}(v_k, l_d)$.
D'après l'équation (4.3), $R_d(v_k, l_d) = -(\alpha + \Gamma_d) E(\tau_{v_k d})$.

Aussi, par définition, on a :

$$\begin{aligned}R_{c_1}(v_k, l_d) &= p_{c_1} - \alpha E(\tau_{v_k c_1}) \\ &\quad + \sum_{m=\underline{\tau}_{v_k c_1}}^{\overline{\tau}_{v_k c_1}} P(\tau_{v_k c_1} = m) \sum_{n=\underline{\sigma}_{c_1}}^{\overline{\sigma}_{c_1}} P(\sigma_{c_1} = n) f(c_1, l_d + m + n) \\ &= p_{c_1} - \alpha E(\tau_{v_k c_1}) + \sum_{m=\underline{\tau}_{v_k c_1}+\underline{\sigma}_{c_1}}^{\overline{\tau}_{v_k c_1}+\overline{\sigma}_{c_1}} P(\tau_{v_k c_1} + \sigma_{c_1} = m) f(c_1, l_d + m)\end{aligned}$$

Soit k_1 l'étape associée au client c_1, comme $c_1 \in \bar{V}_k$, on sait que $k_1 \geqslant k$. Soit $v \in \bar{V}_{k_1}$. Par hypothèse de récurrence, quand on se trouve au client c_1, les assertions $\mathcal{A}_1$ et $\mathcal{A}_3$ sont vérifiées. On a donc :

$$R_d(c_1, l_d) \geqslant R_v(c_1, l_d) \text{et} R_d(c_1, t+1) - R_d(c_1, t) \geqslant R_v(c_1, t+1) - R_v(c_1, t)$$

D'où les inégalités pour tout $t \geqslant l_d$:

$$\begin{aligned}R_d(c_1, t) &= R_d(c_1, l_d) + \sum_{m=l_d}^{t-1} \big(R_d(c_1, m+1) - R_d(c_1, m)\big) \\ &\geqslant R_v(c_1, l_d) + \sum_{m=l_d}^{t-1} \big(R_v(c_1, m+1) - R_v(c_1, m)\big) \\ R_d(c_1, t) &\geqslant R_v(c_1, t)\end{aligned}$$

Ainsi, $\forall t \geqslant l_d$, on a : $\forall v \in \bar{V}_{k_1}, R_d(c_1, t) \geqslant R_v(c_1, t)$. Donc $\forall t \geqslant l_d$, $f(c_1, t) = R_d(c_1, t)$. On en déduit :

$$\begin{aligned}R_{c_1}(v_k, l_d) &= p_{c_1} - \alpha E(\tau_{v_k c_1}) + \sum_{m=\underline{\tau}_{v_k c_1}+\underline{\sigma}_{c_1}}^{\overline{\tau}_{v_k c_1}+\overline{\sigma}_{c_1}} P(\tau_{v_k c_1} + \sigma_{c_1} = m) R_d(c_1, l_d + m) \\ &= p_{c_1} - \alpha E(\tau_{v_k c_1}) - \alpha E(\tau_{c_1 d}) \\ &\quad - \Gamma_d \sum_{m=\underline{\tau}_{v_k c_1}+\underline{\sigma}_{c_1}}^{\overline{\tau}_{v_k c_1}+\overline{\sigma}_{c_1}} P(\tau_{v_k c_1} + \sigma_{c_1} = m) \sum_{n=\underline{\tau}_{c_1 d}}^{\overline{\tau}_{c_1 d}} P(\tau_{c_1 d} = n) \max(m+n, 0)\end{aligned}$$

En raisonnant comme pour l'équation (4.4), en remplaçant v_N par c_1 et v_{N-1} par v_k,

on obtient :

$$\begin{aligned} R_{c_1}(v_k, l_d) &= p_{c_1} - \alpha E(\tau_{v_k c_1}) - \alpha E(\tau_{c_1 d}) - \Gamma_d \Big[E(\tau_{v_k c_1}) + E(\tau_{c_1 d}) + E(\sigma_{c_1}) \Big] \\ &= \underbrace{p_{c_1} - \Gamma_d E(\sigma_{c_1})}_{\leqslant 0 \text{(d'après (H1))}} - (\alpha + \Gamma_d) \underbrace{\Big[E(\tau_{v_k c_1}) + E(\tau_{c_1 d}) \Big]}_{\geqslant E(\tau_{v_k d}) \text{ (d'après (P1))}} \\ &\leqslant -(\alpha + \Gamma_d) E(\tau_{v_k d}) \\ R_{c_1}(v_k, l_d) &\leqslant R_d(v_k, l_d) \end{aligned}$$

– Montrons à présent que R_d et R_{c_1} sont décroissantes (assertion $\mathcal{A}_2(k)$).
Comme précédemment, R_d est décroissante. Aussi, par hypothèse de récurrence, $f(c_1, t, \bar{V}_{c_1})$ est décroissante donc R_{c_1} est décroissante.

– Montrons désormais que l'assertion $\mathcal{A}_3(k)$ est vérifiée.
D'après l'équation (4.6), on a : $R_d(v_k, t+1) - R_d(v_k, t) = -\Gamma_d P(\tau_{v_k d} \geqslant l_d - t)$. En notant $\Delta R_{c_1}(v_k, t) = R_{c1}(v_k, t+1) - R_{c1}(v_k, t)$, on a :

$$\Delta R_{c1}(v_k, t) = \sum_{m=\underline{\tau}_{v_k c_1} + \underline{\sigma}_{c_1}}^{\overline{\tau}_{v_k c_1} + \overline{\sigma}_{c_1}} P(\tau_{v_k c_1} + \sigma_{c_1} = m) \Big(f(c_1, t+1+m) - f(c_1, t+m) \Big)$$

Soient c_m le sommet choisi en $t+m$ et c_{m+1} le sommet choisi en $t+m+1$.
Alors, $f(c_1, t+1+m) - f(c_1, t+m) = R_{c_{m+1}}(c_1, t+1+m) - R_{c_m}(c_1, t+m)$.
Aussi, $R_{c_m}(c_1, t+m) \geqslant R_{c_{m+1}}(c_1, t+m)$.
On en déduit que $f(c_1, t+1+m) - f(c_1, t+m) \leqslant R_{c_{m+1}}(c_1, t+1+m) - R_{c_{m+1}}(c_1, t+m)$

$$\Delta R_{c1}(v_k, t) \leqslant \sum_{m=\underline{\tau}_{v_k c_1} + \underline{\sigma}_{c_1}}^{\overline{\tau}_{v_k c_1} + \overline{\sigma}_{c_1}} P(\tau_{v_k c_1} + \sigma_{c_1} = m) \Big(R_{c_{m+1}}(c_1, t+1+m) - R_{c_{m+1}}(c_1, t+m) \Big)$$

Or, par hypothèse de récurrence, $\mathcal{A}_3$ est vraie aux étapes $k+1, ..., N$. Donc

$$R_{c_{m+1}}(c_1, t+1) - R_{c_{m+1}}(c_1, t) \leqslant R_d(c_1, t+1+m) - R_d(c_1, t+m)$$

D'où :

$$\begin{aligned}
\Delta R_{c1}(v_k,t) &\leqslant \sum_{m=\underline{\tau}_{v_kc_1}+\underline{\sigma}_{c_1}}^{\overline{\tau}_{v_kc_1}+\overline{\sigma}_{c_1}} P(\tau_{v_kc_1}+\sigma_{c_1}=m)\Big(R_d(c_1,t+m+1)-R_d(c_1,t+m)\Big)\\
&\leqslant \sum_{m=\underline{\tau}_{v_kc_1}+\underline{\sigma}_{c_1}}^{\overline{\tau}_{v_kc_1}+\overline{\sigma}_{c_1}} P(\tau_{v_kc_1}+\sigma_{c_1}=m)\Big(-\Gamma_{d^p}P(\tau_{c_1d}\geqslant l_d-t-m)\Big)\\
&\leqslant -\Gamma_d\sum_{m=\underline{\delta}}^{\overline{\delta}} P(\delta=m)P(\sigma_{c_1}\geqslant l_d-t-\lceil D_{v_kc_1}m\rceil+\lceil D_{c_1d}m\rceil)
\end{aligned}$$

Or, soit δ_2 tel que $P(\tau_{v_kd}\geqslant l_d-t)=P(\delta\geqslant\delta_2)$, et soit $\delta\geqslant\delta_2$, on a :

$$\begin{aligned}
\lceil D_{v_kc_1}\delta\rceil+\lceil D_{c_1d}\delta\rceil &\geqslant \lceil (D_{v_kc_1}+D_{c_1d})\delta\rceil\\
&\geqslant \lceil D_{v_kd}\delta\rceil\\
&\geqslant \lceil D_{v_kd}\delta_2\rceil\\
&\geqslant l_d-t
\end{aligned}$$

Donc :

$$\begin{aligned}
\Delta R_{c1}(v_k,t) &\leqslant -\Gamma_d\sum_{m=\underline{\delta}}^{\delta_2-1} P(\delta=m)P(\sigma_{c_1}\geqslant l_d-t-\lceil D_{v_kc_1}m\rceil+\lceil D_{c_1d}m\rceil)\\
&\quad -\Gamma_d\sum_{m=\delta_2}^{\overline{\delta}} P(\delta=m)P(\sigma_{c_1}\geqslant l_d-t-\lceil D_{v_kc_1}m\rceil+\lceil D_{c_1d}m\rceil)\\
&\leqslant -\Gamma_d\sum_{m=\delta_2}^{\overline{\delta}} P(\delta=m)P(\sigma_{c_1}\geqslant l_d-t-\lceil D_{v_kc_1}m\rceil+\lceil D_{c_1d}m\rceil)\\
&\leqslant -\Gamma_dP(\delta\geqslant\delta_2)=-\Gamma_dP(\tau_{v_kd}\geqslant l_d-t)\\
\Delta R_{c1}(v_k,t) &\leqslant R_d(v_k,t+1)-R_d(v_k,t)
\end{aligned}$$

L'assertion $\mathcal{A}_3$ est donc vérifiée à l'étape k.

On sait donc que les fonctions de revenu R_{c_1} et R_d sont décroissantes et que $R_d(v_k,l_d)\geqslant R_{c_1}(v_k,l_d)$.
De plus, comme $\mathcal{A}_3(k)$ est vraie, en procédant comme à l'étape $N-1$, on peut montrer que s'il existe s pour lequel on a $R_d(v_k,s)\geqslant R_{c_1}(v_k,s)$, alors $\forall t\geqslant s, R_d(v_k,t)\geqslant R_{c_1}(v_k,t)$. Deux cas sont donc à envisager :

– Soit $R_d(v_k,0)\geqslant R_{c_1}(v_k,0)$, auquel cas on préfèrera toujours aller au client destination

$(s = -1)$.

- Soit $R_d(v_k, 0) \leqslant R_{c_1}(v_k, 0)$, auquel cas, il existe un seuil unique s vérifiant $\forall t \leqslant s$, on choisit d'aller au client c_1 et $\forall t > s$, on choisit d'aller directement au client destination du segment.

Il existe donc bien un seuil unique entre tout client optionnel de $\bar{V}_k$ et le client destination du segment.

Seuil(s) entre deux clients optionnels

Soient c_1 et c_2 deux clients optionnels de $\bar{V}_k$, on notera $R_{c_1}(v_k, t)$ et $R_{c_2}(v_k, t)$ le profit espéré si on décide d'aller respectivement au client c_1 et c_2 au temps t.
On veut montrer ici qu'il existe un (ou plusieurs) seuil(s) $s_1, ..., s_n$ pour lesquels $\forall t \in]s_1, s_2]$, on préfère aller visiter c_1 (réciproquement c_2) et $\forall t \in]s_2, s_3]$, on préfère aller visiter c_2 (réciproquement c_1). On montre très simplement, comme ci-dessus, que R_{c_2} et R_{c_1} sont décroissantes. Deux cas sont alors à envisager :

- Soit $R_{c_1}(0) - R_{c_2}(0)$ et $R_{c_1}(l_d) - R_{c_2}(l_d)$ sont tous les deux positifs (réciproquement négatifs). Deux cas sont alors possibles. Soit $\forall t \in [0; l_d], R_{c_1}(t) - R_{c_2}(t)$ est positif (réciproquement négatif). On préfèrera donc toujours visiter c_1 (réciproquement c_2). Soit il existe au moins un intervalle sur lequel $R_{c_1}(t) - R_{c_2}(t)$ est négatif (réciproquement positif). Sur cet (ces) intervalle(s), on préfèrera donc aller en c_2 (réciproquement c_1) et hors de cet (ces) intervalle(s), on préfèrera aller en c_1 (récipro-quement c_2).
- Soit $R_{c_1}(0) - R_{c_2}(0)$ et $R_{c_1}(l_d) - R_{c_2}(l_d)$ sont de signe contraire. Auquel cas, il existe au moins un seuil s en lequel $R_{c_1}(t) - R_{c_2}(t)$ change de signe (id est, un point en lequel il devient préférable de visiter non plus c_2 mais c_1, ou l'inverse).

Il existe donc bien au moins un seuil pour chaque paire de clients optionnels de $\bar{V}_k$.

Politique de seuil

Montrons que l'on a donc bien une politique de seuil. Soit $s1$ le maximum des seuils entre les clients optionnels de $\bar{V}_k$ et le client destination (et $c1$ le client associé à ce seuil). Pour tout $t \leqslant s1$, il existe un client optionnel préférable au client destination. Par contre, si $t > s1$, on préfèrera aller directement au client destination. On a donc identifié un premier seuil $s1$.
Maintenant, soit S_2 l'ensemble des seuils s_i entre les clients optionnels de $\bar{V}_k$ et le client $c1$ pour lesquels $\forall t > s_i$, on va au client $c1$. Soit $s2$ le plus grand seuil de S_2 et $c2$ le client qui lui est associé. Pour tout $t \leqslant s2$, il existe un client optionnel préférable au client $c1$. Par contre, si $t > s2$, il est préférable d'aller au client $c1$. On a donc identifié un deuxième seuil $s2$.
On procède ainsi jusqu'à l'étape $h + 1$ pour laquelle S_{h+1} est vide. On obtient ainsi h seuils

$s1, s2, ..., sh$ et les clients associés $c1, c2, ..., ch$. La politique optimale est alors la suivante :
$\forall t \leqslant sh$, on va au client ch
$\forall t, sh < t \leqslant s(h-1)$, on va au client $c(h-1)$
...
$\forall t, s2 < t \leqslant s1$, on va au client $c1$
$\forall t > s1$, on se rend directement au client destination du segment d
On a donc bien identifié une politique de seuil à l'étape k.

Décroissance de la fonction de revenu (assertion $\mathcal{A}_4(k)$)

Montrons que la fonction de revenu est décroissante (jusqu'ici, on sait juste qu'elle est décroissante par morceaux, chaque morceau étant délimité par les seuils). Considérons deux morceaux successifs $m1 = [t1, t2]$ et $m2 = [t2 + 1, t3]$. Supposons que, sur $m1$, le revenu soit donné par $R1$ et sur $m2$ par $R2$.
Montrons que $R1(t2) \geqslant R2(t2 + 1)$, ce qui prouvera la décroissance au niveau des discontinuités.
On sait que $R2$ est décroissante donc $R2(t2) \geqslant R2(t2 + 1)$.
De plus, sur $m1 = [t1, t2]$, le revenu est donné par $R1$ donc $R1(t2) \geqslant R2(t2)$.
On en conclut que $R1(t2) \geqslant R2(t2) \geqslant R2(t2 + 1)$.
La fonction de revenu à l'étape k est donc bien décroissante sur son ensemble de définition.

4.5 Algorithme 2 : preuve sur toute la route

Nous venons de traiter la première stratégie ne considérant qu'un seul segment. Nous nous intéressons à présent à la deuxième stratégie considérant le reste de la route. Dans cette section, on réintroduit donc l'index p du segment (on notera o^p l'origine du segment, d^p la destination du segment et $\bar{V}_k^p$ l'ensemble des clients optionnels, associés au segment p, situés après v_k). On introduit également la notation V^p pour désigner l'ensemble des clients optionnels associés au segment p. On notera dans cette section $\hat{f}(v_k, t_k, \bar{V}_k^p)$ le revenu espéré si on finit le service du client v_k à t_k pour cette stratégie et $\hat{f}(d^p, t)$ le revenu espéré si on débute le service du client d^p à t. On notera également $\hat{R}_{d^p}(v_k, t)$ et $\hat{R}_{\bar{v}}(v_k, t)$ les revenus associés au fait de se rendre respectivement au client d^p et au client $\bar{v}$ depuis le client v_k à t_k. Avec ces

notations, pour cette stratégie, la fonction de revenu s'écrit alors :

$$\hat{f}(v_k, t_k, \bar{V}_k^p) = \max \begin{cases} \hat{R}_{d^p}(v_k, t_k) \\ \max_{\bar{v} \in \bar{V}_k^p} \hat{R}_{\bar{v}}(v_k, t_k) \end{cases}$$
$$= \max \begin{cases} -\alpha E(\tau_{v_k d^p}) + E[\hat{f}(d^p, t_k + \tau_{v_k d^p})] \\ \max_{\bar{v} \in \bar{V}_k^p} \left(p_{\bar{v}} - \alpha E(\tau_{v_k \bar{v}}) + E[\hat{f}(\bar{v}, t_k + \tau_{v_k \bar{v}}, \bar{V}_k^p \backslash \{\bar{v}\})] \right) \end{cases}$$

avec $\hat{f}(d^p, t) = -\Gamma_{d^p} \max(t - l_{d^p}, 0) + \hat{f}(d^p, t + \sigma_{d^p}, V^{p+1})$
En ce qui concerne les assertions, on utilisera les mêmes notations que précédemment $\mathcal{A}_1, \mathcal{A}_2$, $\mathcal{A}_3$ et $\mathcal{A}_4$ auxquelles on associera deux paramètres : le segment p et l'étape k. Pour prouver que la politique optimale est bien une politique de suil pour cette stratégie, on prouvera par induction sur les segments et les étapes que les assertions $\mathcal{A}_1, \mathcal{A}_2$, $\mathcal{A}_4$ sont vérifiées. Et on donnera un contre-exemple pour montrer que l'unicité du seuil (assertion $\mathcal{A}_3$) n'est pas vérifiée.

4.5.1 Dernier segment de route d'un véhicule

On a prouvé dans la section précédente que, dans ce cas, les assertions $\mathcal{A}_1, \mathcal{A}_2$, $\mathcal{A}_3$ et $\mathcal{A}_4$ sont vérifiées et que la politique optimale est bien une politique de seuil.

4.5.2 Induction sur les segments de route

Supposons que les assertions soient vérifiées sur les segments $p+1, p+2, ..., |P|$.
Montrons qu'alors, elles restent vraies pour le segment p et que la politique optimale est bien une politique de seuil sur ce segment.
Le fait de considérer le reste de la route et non simplement le segment en cours entraîne une différence dans l'expression de la fonction de revenu au client destination du segment p. En effet, on a :

$$\hat{f}(d^p, t) = -\Gamma_{d^p} \max(t - l_{d^p}, 0) + \sum_{m=\underline{\sigma}_{d^p}}^{\overline{\sigma}_{d^p}} P(\sigma_{d^p} = m) \hat{f}(o^{p+1}, t + m, V^{p+1})$$

Par hypothèse de récurrence, $\hat{f}(o^{p+1}, t, V^{p+1})$ est décroissante donc ce revenu est une fonction décroissante du temps.

Client optionnel v_N (étape $k = N$)

On se trouve au dernier client optionnel du segment p, on se rend donc directement au client destination :

$$\hat{f}(v_N, t_N, \bar{V}_N^p) = -\alpha E(\tau_{v_N d^p}) + \sum_{m=\underline{\tau}_{v_N d^p}}^{\overline{\tau}_{v_N d^p}} P(\tau_{v_N d^p} = m)\hat{f}(d^p, t_N + m)$$

Ici, $\bar{V}_N^p$ est vide, les assertions $\mathcal{A}_1(N,p)$, $\mathcal{A}_2(N,p)$ et $\mathcal{A}_3(N,p)$ sont donc clairement vérifiées. De plus, comme $\hat{f}(d^p, t)$ est décroissante, $\hat{f}(v_N, t, \bar{V}_N^p)$ l'est également (et $\mathcal{A}_4(N,p)$ est vérifiée). On a bien une politique de seuil (on décide de se rendre au client destination en tout temps).

Client optionnel v_{N-1} (étape $k = N-1$)

On se trouve à l'avant dernier client optionnel du segment. On a deux possibilités : soit on se rend directement au client destination du segment, soit on visite le dernier client optionnel du segment. Le choix est déterminé par l'équation :

$$\hat{f}(v_{N-1}, t_{N-1}, \bar{V}_{N-1}^p) = \max \begin{pmatrix} \hat{R}_{d^p}(v_{N-1}, t_{N-1}) \\ \hat{R}_{v_N}(v_{N-1}, t_{N-1}) \end{pmatrix}$$

avec

$$\hat{R}_{d^p}(v_{N-1}, t_{N-1}) = -\alpha E(\tau_{v_{N-1} d^p}) + \sum_{m=\underline{\tau}_{v_{N-1} d^p}}^{\overline{\tau}_{v_{N-1} d^p}} P(\tau_{v_{N-1} d^p} = m)\hat{f}(d^p, t_{N-1} + m)$$

$$\begin{aligned}\hat{R}_{v_N}(v_{N-1}, t_{N-1}) = p_{v_N} &- \alpha E(\tau_{v_{N-1} v_N}) \\ &+ \sum_{m=\underline{\tau}_{v_{N-1} v_N}}^{\overline{\tau}_{v_{N-1} v_N}} P(\tau_{v_{N-1} v_N} = m) \sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_N} = n)\hat{f}(v_N, t_{N-1} + m + n, \emptyset)\end{aligned}$$

Pour montrer qu'il existe un (ou plusieurs) seuil(s) au-delà duquel on va directement au client destination et en deçà duquel on va au dernier client optionnel, on montre les assertions $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_4$ et on donne un contre-exemple pour l'assertion $\mathcal{A}_3$ (unicité du seuil).

Preuve de l'assertion $\mathcal{A}_1(N-1,p)$

Montrons l'assertion $\mathcal{A}_1(N-1,p)$: $\hat{R}_{d^p}(v_{N-1}, l_{d^p}) \geqslant \hat{R}_{v_N}(v_{N-1}, l_{d^p})$.
On sait que $\hat{R}_{d^p}(v_{N-1}, l_{d^p})$ vaut :

$$\begin{aligned}
\hat{R}_{d^p}(v_{N-1}, l_{d^p}) &= -\alpha E(\tau_{v_{N-1}d^p}) + \sum_{m=\underline{\tau}_{v_{N-1}d^p}}^{\overline{\tau}_{v_{N-1}d^p}} P(\tau_{v_{N-1}d^p} = m)\hat{f}(d^p, l_{d^p} + m) \\
&= -\alpha E(\tau_{v_{N-1}d^p}) - \Gamma_{d^p} \sum_{m=\underline{\tau}_{v_{N-1}d^p}}^{\overline{\tau}_{v_{N-1}d^p}} P(\tau_{v_{N-1}d^p} = m)\max(m, 0) \\
&\quad + \sum_{m=\underline{\tau}_{v_{N-1}d^p}}^{\overline{\tau}_{v_{N-1}d^p}} P(\tau_{v_{N-1}d^p} = m) \sum_{n=\underline{\sigma}_{d^p}}^{\overline{\sigma}_{d^p}} P(\sigma_{d^p} = n)\hat{f}(o^{p+1}, t + m + n, V^{p+1}) \\
&= -(\alpha + \Gamma_{d^p})E(\tau_{v_{N-1}d^p}) \\
&\quad + \sum_{m=\underline{\tau}_{v_{N-1}d^p}}^{\overline{\tau}_{v_{N-1}d^p}} P(\tau_{v_{N-1}d^p} = m) \sum_{n=\underline{\sigma}_{d^p}}^{\overline{\sigma}_{d^p}} P(\sigma_{d^p} = n)\hat{f}(o^{p+1}, l_{d^p} + m + n, V^{p+1}) \\
&= -(\alpha + \Gamma_{d^p})E(\tau_{v_{N-1}d^p}) \\
&\quad + \sum_{m=\underline{\delta}}^{\overline{\delta}} P(\delta = m) \sum_{n=\underline{\sigma}_{d^p}}^{\overline{\sigma}_{d^p}} P(\sigma_{d^p} = n)\hat{f}(o^{p+1}, l_{d^p} + \lceil D_{v_{N-1}d^p}\delta \rceil + n, V^{p+1}) \quad (4.9)
\end{aligned}$$

Calculons à présent $\hat{R}_{v_N}(v_{N-1}, l_{d^p})$:

$$\begin{aligned}
&\hat{R}_{v_N}(v_{N-1}, l_{d^p}) \\
&= p_{v_N} - \alpha E(\tau_{v_{N-1}v_N}) + \sum_{m=\underline{\tau}_{v_{N-1}v_N} + \underline{\sigma}_{v_N}}^{\overline{\tau}_{v_{N-1}v_N} + \overline{\sigma}_{v_N}} P(\tau_{v_{N-1}v_N} + \sigma_{v_N} = m)\hat{f}(v_N, l_{d^p} + m, \emptyset) \\
&= p_{v_N} - \alpha \underbrace{\big(E(\tau_{v_{N-1}v_N}) + E(\tau_{v_N d^p})\big)}_{\geqslant E(\tau_{v_{N-1}d^p}) \text{ (cf. (P1))}} \\
&\quad + \sum_{m=\underline{\tau}_{v_{N-1}v_N} + \underline{\sigma}_{v_N}}^{\overline{\tau}_{v_{N-1}v_N} + \overline{\sigma}_{v_N}} P(\tau_{v_{N-1}v_N} + \sigma_{v_N} = m) \sum_{n=\underline{\tau}_{v_N d^p}}^{\overline{\tau}_{v_N d^p}} P(\tau_{v_N d^p} = n)\hat{f}(d^p, l_{d^p} + m + n) \quad (4.10)
\end{aligned}$$

$$\begin{aligned}
&\hat{R}_{v_N}(v_{N-1}, l_{d^p}) \\
&\leqslant p_{v_N} - \alpha E(\tau_{v_{N-1}d^p}) \\
&\quad + \sum_{m=\underline{\tau}_{v_{N-1}v_N}+\underline{\sigma}_{v_N}}^{\overline{\tau}_{v_{N-1}v_N}+\overline{\sigma}_{v_N}} P(\tau_{v_{N-1}v_N} + \sigma_{v_N} = m) \sum_{n=\underline{\tau}_{v_N d^p}}^{\overline{\tau}_{v_N d^p}} P(\tau_{v_N d^p} = n)\hat{f}(d^p, l_{d^p} + m + n)
\end{aligned}$$

Développons l'expression de $\hat{f}(d^p, l_{d^p} + m + n)$.
Soit $m + n \in [\underline{\tau}_{v_{N-1}v_N} + \underline{\sigma}_{v_N} + \underline{\tau}_{v_N d^p}; \overline{\tau}_{v_{N-1}v_N} + \overline{\sigma}_{v_N} + \overline{\tau}_{v_N d^p}]$, on a $m + n > 0$ d'après (H5). D'où :

$$\begin{aligned}
\hat{f}(d^p, l_{d^p} + m + n) &= -\Gamma_{d^p} \max(m + n, 0) + \sum_{q=\underline{\sigma}_{d^p}}^{\overline{\sigma}_{d^p}} \hat{f}(o^{p+1}, l_d + m + n + q, V^{p+1}) \\
&= -\Gamma_{d^p}(m + n) + \sum_{q=\underline{\sigma}_{d^p}}^{\overline{\sigma}_{d^p}} \hat{f}(o^{p+1}, l_d + m + n + q, V^{p+1}) \qquad (4.11)
\end{aligned}$$

Comme les temps de service et de parcours sont indépendants,

$$\begin{aligned}
&\sum_{m=\underline{\tau}_{v_{N-1}v_N}+\underline{\sigma}_{v_N}}^{\overline{\tau}_{v_{N-1}v_N}+\overline{\sigma}_{v_N}} P(\tau_{v_{N-1}v_N} + \sigma_{v_N} = m) \sum_{n=\underline{\tau}_{v_N d^p}}^{\overline{\tau}_{v_N d^p}} P(\tau_{v_N d^p} = n)(m + n) \\
&= \sum_{m=\underline{\tau}_{v_{N-1}v_N}}^{\overline{\tau}_{v_{N-1}v_N}} P(\tau_{v_{N-1}v_N} = m) \sum_{n=\underline{\sigma}_{v_N}}^{\overline{\sigma}_{v_N}} P(\sigma_{v_N} = m) \sum_{q=\underline{\tau}_{v_N d^p}}^{\overline{\tau}_{v_N d^p}} P(\tau_{v_N d^p} = q)(m + n + q) \\
&= E(\tau_{v_{N-1}v_N}) + E(\sigma_{v_N}) + E(\tau_{v_N d^p}) \qquad (4.12)
\end{aligned}$$

En utilisant les équations (4.11) et (4.12), on obtient :

$$\hat{R}_{v_N}(v_{N-1}, l_{d^p}) \leqslant p_{v_N} - \alpha E(\tau_{v_{N-1}d^p}) - \Gamma_{d^p}[E(\tau_{v_{N-1}v_N}) + E(\tau_{v_N d^p}) + E(\sigma_{v_N})] + A \qquad (4.13)$$

Avec :

$$A = \sum_{m=\underline{\tau}_{v_{N-1}v_N}+\underline{\sigma}_{v_N}}^{\overline{\tau}_{v_{N-1}v_N}+\overline{\sigma}_{v_N}} P(\tau_{v_{N-1}v_N} + \sigma_{v_N} = m) \sum_{n=\underline{\tau}_{v_N d^p}}^{\overline{\tau}_{v_N d^p}} P(\tau_{v_N d^p} = n) \sum_{q=\underline{\sigma}_{d^p}}^{\overline{\sigma}_{d^p}} P(\sigma_{d^p} = q)\hat{f}(o^{p+1}, l_{d^p} + m + n + q, V^{p+1})$$

$$\hat{R}_{v_N}(v_{N-1}, l_{d^p}) \leqslant \underbrace{p_{v_N} - \Gamma_{d^p} E(\sigma_{v_N})}_{\leqslant 0 \text{ d'après (H1)}} - \alpha E(\tau_{v_{N-1}d^p}) - \Gamma_{d^p} \underbrace{\left(E(\tau_{v_{N-1}v_N}) + E(\tau_{v_N d^p})\right)}_{\geqslant E(\tau_{v_{N-1}d^p}) \text{ (cf. (P1))}} + A$$

$$\hat{R}_{v_N}(v_{N-1}, l_{d^p}) \leqslant -(\alpha + \Gamma_{d^p}) E(\tau_{v_{N-1}d^p}) + A$$

Afin de pouvoir comparer $\hat{R}_{v_N}(v_{N-1}, l_{d^p})$ et $\hat{R}_{d^p}(v_{N-1}, l_{d^p})$, examinons le terme A. on a :

$$A = \sum_{m=\underline{\delta}}^{\overline{\delta}} P(\delta = m) \sum_{l=\underline{\sigma}_{d^p}}^{\overline{\sigma}_{d^p}} P(\sigma_{v_N} = l) \sum_{n=\underline{\sigma}_{d^p}}^{\overline{\sigma}_{d^p}} P(\sigma_{d^p} = n) \hat{f}(o^{p+1}, l_{d^p} + \lceil D_{v_{N-1}v_N} m \rceil + l + \lceil D_{v_N d^p} m \rceil + n, V^{p+1})$$

De plus, on sait que :

$$\begin{aligned} \lceil D_{v_{N-1}v_N} m \rceil + \lceil D_{v_N d^p} m \rceil &\geqslant \lceil (D_{v_{N-1}v_N} + D_{v_N d^p}) m \rceil && \text{propriété de la fonction plafond} \\ &\geqslant \lceil D_{v_{N-1}d^p} m \rceil && \text{d'après (H2)} \end{aligned}$$

Comme la fonction $\hat{f}(o^{p+1}, t, V^{p+1})$ est décroissante (par hypothèse de récurrence), on a :
$\hat{f}(o^{p+1}, l_{d^p} + \lceil D_{v_{N-1}v_N} m \rceil + \lceil D_{v_N d^p} m \rceil + n, V^{p+1}) \leqslant \hat{f}(o^{p+1}, l_{d^p} + \lceil D_{v_{N-1}d^p} m \rceil + n, V^{p+1})$
On en conclut que $\hat{R}_{v_N}(v_{N-1}, l_{d^p}) \leqslant \hat{R}_{d^p}(v_{N-1}, l_{d^p})$.

Preuve de l'assertion $\mathcal{A}_2(N-1, p)$

Montrons que $\hat{R}_{d^p}$ et $\hat{R}_{v_N}$ sont décroissantes (assertion $\mathcal{A}_2(N-1, p)$) Comme $\hat{f}(d^p, t)$ et $\hat{f}(v_N, t, \emptyset)$ sont décroissantes, $\hat{R}_{d^p}$ et $\hat{R}_{v_N}$ sont clairement décroissantes.

Preuve de l'assertion $\mathcal{A}_4(N-1, p)$

Montrons que $\hat{f}(v_{N-1}, t, \bar{V}^p_{N-1})$ est décroissante (assertion $\mathcal{A}_4(N-1, p)$). On sait d'ores et déjà que la fonction est décroissante par morceaux comme $\hat{R}_{d^p}$ et $\hat{R}_{v_N}$ sont décroissantes. En procédant comme dans le paragraphe §3.4.4, on montre la décroissance aux discontinuités. On en conclut que $\hat{f}(v_{N-1}, t, \bar{V}^p_{N-1})$ est bien décroissante sur son ensemble de définition.

Preuve de l'assertion $\mathcal{A}_3(N-1, p)$

Pour ce qui est de l'unicité du seuil (assertion $\mathcal{A}_3(N-1, p)$), montrons avec un contre-exemple que la fonction $\hat{R}_{d^p} - \hat{R}_{v_N}$ n'est pas croissante.
Le contre-exemple est représenté graphiquement ci-dessous. Dans cet exemple, on choisit $\alpha = 1$, $\Gamma_{d^p} = 5000$, $\underline{\delta} = 5$, $\overline{\delta} = 13$ et $\hat{\delta} = 7$.
Pour les clients optionnels, on pose $p_i = 20$, $\underline{\sigma}_i = 15$, $\overline{\sigma}_i = 30$ et $\hat{\sigma}_i = 22$.
Pour les clients obligatoires, on pose $\underline{\sigma}_i = 30$ et $\overline{\sigma}_i = 60$ et $\hat{\sigma}_i = 35$.

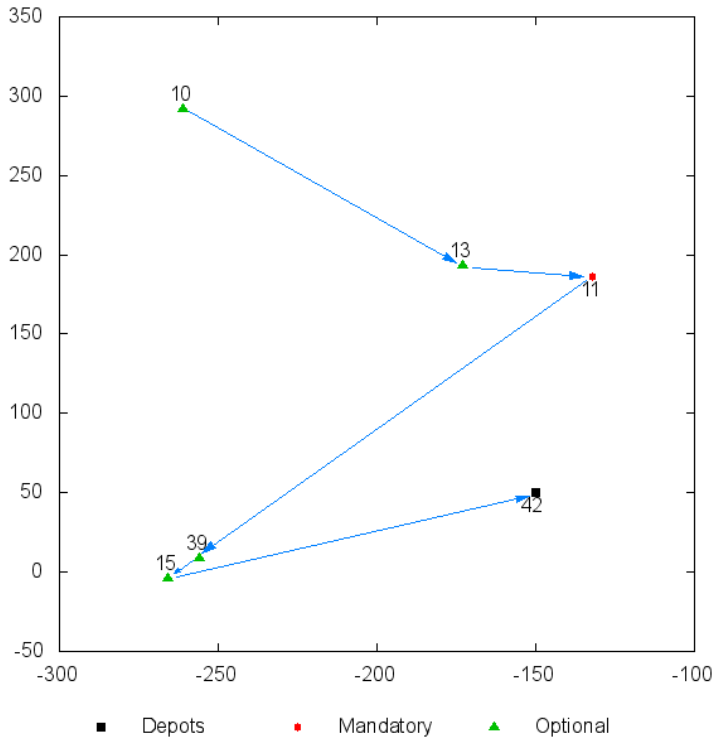


Figure 4.3 Contre-exemple graphique

On se trouve au client 10 ($v_{N-1} = 10$, $v_N = 13$ et $d^p = 11$).
En calculant, avec la programmation dynamique, les fonctions de revenu des clients 11, 39, 15 et 42 en fonction du temps, on calcule $\hat{R}_{11}(10,t)$ et $\hat{R}_{13}(10,t)$ comme suit :

$$\hat{R}_{11}(10,t) = -\alpha E(\tau_{10;11}) + \sum_{m=\underline{\tau}_{10;11}}^{\overline{\tau}_{10;11}} \hat{f}(11, t+m, \emptyset)$$

$$\hat{R}_{13}(10,t) = p_{13} - \alpha\big(E(\tau_{10;13}) + E(\tau_{13;11})\big) + \sum_{m=\underline{\tau}_{10;13;11}}^{\overline{\tau}_{10;13;11}} \hat{f}(11, t+m, \emptyset)$$

On obtient ainsi, en $t = 340$, $\hat{R}_{11}(10,t) = 94.04$ et $\hat{R}_{13}(10,t) = 93.44$.
Donc on préfère se rendre au client destination du segment ($d^p = 11$) en $t = 340$.
En $t = 341$, on a : $\hat{R}_{11}(10,t) = 89.51$ et $\hat{R}_{13}(10,t) = 90.01$.
On préfère donc se rendre au client optionnel 13 en $t = 341$.
Ainsi, en posant $s = 340$, on a : $\hat{R}_{d^p}(10,s) \geqslant \hat{R}_{13}(10,s)$. Par contre, on a également : $\hat{R}_{d^p}(10,s+1) \leqslant \hat{R}_{13}(10,s+1)$.
On a donc $\hat{R}_{d^p}(10,s) - \hat{R}_{v_N}(10,s) \geqslant 0 \geqslant \hat{R}_{d^p}(10,s+1) - \hat{R}_{v_N}(10,s+1)$. En d'autres termes,

$\hat{R}_{d^p} - \hat{R}_{v_N}$ n'est pas croissante. L'unicité du seuil n'est donc pas vérifiée.

Induction sur les clients optionnels

Supposons à présent que la politique optimale aux étapes $k+1, k+2, ..., N$ soit bien une politique de seuil, et que les assertions $\mathcal{A}_1, \mathcal{A}_2$ et $\mathcal{A}_4$ soient vraies à ces étapes (on a vu que $\mathcal{A}_3$ n'était pas vérifiée).
Montrons qu'alors, ces assertions sont toujours vraies à l'étape k et que la politique optimale à l'étape k est une politique de seuil.
A l'étape k, on dispose de plusieurs possibilités : soit on va au client destination du segment d^p, soit on se rend chez un client optionnel de $\bar{V}_k^p$.

$$\hat{f}(v_k, t_k, \bar{V}_k^p) = \max \left\{ \begin{array}{l} -\alpha E(\tau_{v_k d^p}) + \displaystyle\sum_{m=\underline{\tau}_{v_k d^p}}^{\overline{\tau}_{v_k d^p}} P(\tau_{v_k d^p} = m) \hat{f}(d^p, t_k + m) \\ \\ \displaystyle\max_{\bar{v} \in \bar{V}_k^p} \left(p_{\bar{v}} - \alpha E(\tau_{v_k \bar{v}}) \right. \\ \left. + \displaystyle\sum_{m=\underline{\tau}_{v_k \bar{v}}}^{\overline{\tau}_{v_k \bar{v}}} P(\tau_{v_k \bar{v}} = m) \sum_{n=\underline{\sigma}_{\bar{v}}}^{\overline{\sigma}_{\bar{v}}} P(\sigma_{\bar{v}} = n) \hat{f}(\bar{v}, t_k + m + n, \bar{V}_k \backslash \{\bar{v}\}) \right) \end{array} \right.$$

Pour montrer que l'on a bien une politique de seuil à l'étape k, on procède en trois temps :

1. Montrer qu'il existe un (ou plusieurs) seuil(s) entre tout client optionnel de $\bar{V}_k^p$ et d^p
2. Montrer qu'il existe un (ou plusieurs) seuil(s) entre toute paire de clients optionnels de $\bar{V}_k^p$
3. Montrer que ces seuils définissent une politique de seuil

Seuil(s) entre un client optionnel et le client destination

Soient v le client courant et c_1 un client optionnel de $\bar{V}_k^p$, on notera $\hat{R}_{c_1}(v,t)$ le profit espéré si on décide d'aller au client c_1 et $\hat{R}_{d^p}(v,t)$ le profit espéré si on se rend au client destination au temps t.
On veut montrer qu'il existe un seuil s pour lequel $\forall t \leqslant s$, on choisit de visiter le client c_1 $\left(\hat{R}_{d^p}(v,t) \leqslant \hat{R}_{c_1}(v,t)\right)$ et $\forall t > s$, on se rend au client destination $\left(\hat{R}_{d^p}(v,t) \geqslant \hat{R}_{c_1}(v,t)\right)$.
Pour ce faire, raisonnons comme précédemment : montrons les assertions $\mathcal{A}_1$ et $\mathcal{A}_2$.

Preuve de l'assertion $\mathcal{A}_1(k,p)$

Montrons que $\hat{R}_{d^p}(v, l_{d^p}) \geqslant \hat{R}_{c_1}(v, l_{d^p})$ (assertion $\mathcal{A}_1(k,p)$).

En remplaçant v_N par v dans l'équation (4.9), on obtient l'expression suivante pour $\hat{R}_{d^p}(v, l_{d^p})$:

$$\begin{aligned}\hat{R}_{d^p}(v, l_{d^p}) = &-(\alpha + \Gamma_{d^p})E(\tau_{vd^p}) \\ &+ \sum_{m=\underline{\tau}_{vd^p}}^{\overline{\tau}_{vd^p}} P(\tau_{vd^p} = m) \sum_{n=\underline{\sigma}_{d^p}}^{\overline{\sigma}_{d^p}} P(\sigma_{d^p} = n)\hat{f}(o^{p+1}, l_{d^p} + m + n, V^{p+1})\end{aligned}$$

Aussi, par définition, $\hat{R}_{c_1}(v, l_{d^p})$ vaut :

$$\hat{R}_{c_1}(v, l_{d^p}) = p_{c_1} - \alpha E(\tau_{vc_1}) + \sum_{m=\underline{\tau}_{vc_1}+\underline{\sigma}_{c_1}}^{\overline{\tau}_{vc_1}+\overline{\sigma}_{c_1}} P(\tau_{vc_1} + \sigma_{c_1} = m)\hat{f}(c_1, l_{d^p} + m, \bar{V}_k^p)$$

Par hypothèse de récurrence, quand on se trouve au client c_1, la politique optimale est une politique de seuil donc le revenu en c_1 pour $t \geqslant l_{d^p}$ correspond à celui associé au fait d'aller directement au client destination. Ainsi,

$$\begin{aligned}\hat{R}_{c_1}(v, l_{d^p}) &= p_{c_1} - \alpha E(\tau_{vc_1}) + \sum_{m=\underline{\tau}_{vc_1}+\underline{\sigma}_{c_1}}^{\overline{\tau}_{vc_1}+\overline{\sigma}_{c_1}} P(\tau_{vc_1} + \sigma_{c_1} = m)\Big[- \alpha E(\tau_{c_1 d^p}) \\ &\quad + \sum_{n=\underline{\tau}_{c_1 d^p}}^{\overline{\tau}_{c_1 d^p}} P(\tau_{c_1 d^p} = n)\hat{f}(d^p, l_{d^p} + m + n)\Big] \\ &= p_{c_1} - \alpha\big(E(\tau_{vc_1}) + E(\tau_{c_1 d^p})\big) \\ &\quad + \sum_{m=\underline{\tau}_{vc_1}+\underline{\sigma}_{c_1}}^{\overline{\tau}_{vc_1}+\overline{\sigma}_{c_1}} P(\tau_{vc_1} + \sigma_{c_1} = m) \sum_{n=\underline{\tau}_{c_1 d^p}}^{\overline{\tau}_{c_1 d^p}} P(\tau_{c_1 d^p} = n)\hat{f}(d^p, l_{d^p} + m + n)\end{aligned}$$

L'équation ci-dessus est identique à l'équation (4.10) en remplaçant v_{N-1} par v. En procédant comme pour l'équation (4.10), on montre que $\hat{R}_{c_1}(v, l_{d^p}) \leqslant \hat{R}_{d^p}(v, l_{d^p})$.

Preuve de l'assertion $\mathcal{A}_2(k,p)$

Montrons à présent que $\hat{R}_{d^p}$ et $\hat{R}_{c_1}$ sont décroissantes (assertion $\mathcal{A}_2(k,p)$).

Comme $\hat{f}(d^p, t)$ est décroissante, $\hat{R}_{d^p}$ est décroissante. Aussi, par hypothèse de récurrence, $\hat{f}(c_1, t, \bar{V}_k^p)$ est décroissante donc $\hat{R}_{c_1}$ est clairement décroissante.

On sait donc : (*) que les fonctions de revenu $\hat{R}_{c_1}$ et $\hat{R}_{d^p}$ sont décroissantes (cf. ci-dessus), (**) que $\hat{R}_{d^p}(v, l_{d^p}) \geqslant \hat{R}_{c_1}(v, l_{d^p})$ (cf. section 4.5.2) et (***) que $\hat{R}_{d^p} - \hat{R}_{c_1}$ n'est pas monotone.

Deux cas sont donc à envisager :

- Soit $\hat{R}_{d^p}(v,0) \geqslant \hat{R}_{c_1}(v,0)$. D'après (**) et (***), deux cas sont alors possibles. Soit $\forall t \in [0; l_{d^p}], \hat{R}_{d^p}(v,t) \geqslant \hat{R}_{v_N}(v,t)$. On préfèrera donc toujours visiter d^p. Soit il existe au moins un intervalle sur lequel $\hat{R}_{d^p}(v,t) < \hat{R}_{c_1}(v,t)$. Sur cet intervalle, on préfèrera donc aller en c_1 et hors de cet intervalle, on préfèrera aller en d^p.
- Soit $\hat{R}_{d^p}(v,0) \leqslant \hat{R}_{c_1}(v,0)$. Auquel cas, d'après (**), il existe au moins un seuil s (il peut y en avoir plusieurs d'après (***)) en lequel $\hat{R}_{d^p}(v,t) - \hat{R}_{c_1}(v,t)$ change de signe (id est, un point en lequel il devient préférable de visiter non plus c_1 mais d^p, ou l'inverse).

Il existe donc bien au moins un seuil entre tout client optionnel de $\bar{V}_k^p$ et le client destination du segment.

Seuil(s) entre deux clients optionnels
Même preuve que précédemment (quand on raisonnait sur un seul segment).

Politique de seuil et décroissance de la fonction de revenu ($\mathcal{A}_4(k,p)$)
Mêmes preuves que précédemment (quand on raisonnait sur un seul segment).

Ceci finalise la preuve que la politique optimale obtenue avec nos deux algorithmes de programmation dynamique est une politique de seuil.

CHAPITRE 5

Heuristique basée sur la priorité des clients

Dans la variante du problème de tournées de service considérée, tous les clients sont connus a priori et les temps de parcours et de service sont stochastiques. L'approche de résolution proposée pour résoudre ce problème est une méthode en deux étapes : la planification et l'exécution. Au cours de l'étape de planification, on construit des routes avec des clients obligatoires et des clients optionnels en utilisant des estimés connus a priori. Ensuite, dans l'étape d'exécution, on utilise des outils de programmation dynamique pour déterminer les seuils de la politique optimale à partir des distributions de probabilité des temps de service et de parcours. On dispose ainsi d'une liste de créneaux horaires et de décisions associées. Lors des simulations du déroulement de l'horizon de temps, on décide en temps réel de la suite de la route pour chaque technicien/véhicule. On peut ainsi décider de ne pas desservir un ou plusieurs clients optionnels pour être à l'heure au client obligatoire suivant (cf. chapitre précédent). L'idée dans cette approche est d'utiliser les clients optionnels comme tampon pour absorber les variations sur les temps de parcours et de service.
Dans ce chapitre, on suppose que le nombre de clients obligatoires est suffisant (si ce n'est pas le cas, on peut rendre des clients optionnels obligatoires) et on présente une heuristique basée sur la priorité des clients pour l'étape de planification. Dans cette heuristique, on utilise la priorité des clients pour décomposer, comme Delage [21], l'étape de planification en deux phases : i) construire des routes en ne considérant que les clients obligatoires (on appelle l'ensemble de ces routes le « squelette ») ; ii) insertion des clients optionnels dans le squelette. Dans ce chapitre, nous détaillons l'étape de planification dans la section 5.1, puis l'étape d'exécution dans la section 5.2 pour présenter ensuite les résultats dans la section 5.3 et conclure sur cette méthode dans la section 5.4.

5.1 Etape de planification

Dans l'étape de planification, on cherche à construire des routes optimales contenant à la fois des clients obligatoires et des clients optionnels. En supposant que les bornes sur les temps de parcours et de service sont connues (cf. chapitre 1), on procède en deux phases : d'abord, on construit un squelette de routes contenant uniquement des clients obligatoires et ensuite on insère les clients optionnels dans ce squelette. Dans la phase I (construction du squelette), on formule le problème comme un programme à variables mixtes et on le résout

de façon exacte en utilisant un algorithme de branch and bound. Dans la deuxième phase (insertion des clients optionnels), on formule le problème comme un programme en nombres entiers et on procède en deux temps : on résout d'abord le modèle avec des estimés pessimistes pour les temps de service et de parcours, en utilisant un algorithme de branch and cut puis on répare et on améliore la solution à l'aide d'une méthode heuristique. Pour l'insertion des clients optionnels, on propose également une méthode de relaxation lagrangienne suivie de l'heuristique de réparation et d'amélioration, ce qui nous permet de résoudre de plus grosses instances.

5.1.1 Phase I : Etablissement du squelette

Dans cette phase, on considère uniquement les clients obligatoires, qui possèdent une fenêtre de temps et sont tous connus a priori. Afin de construire le squelette de routes comprenant uniquement des clients obligatoires, il faut résoudre un m-TSPTW sur l'ensemble des clients obligatoires. Comme on n'autorise aucun retard chez les clients obligatoires, on considère dans cette phase que les temps de parcours et de service sont maximaux. On utilise les notations suivantes :

- Ensembles
  - M ensemble de clients obligatoires
  - O ensemble de clients optionnels
  - K ensemble de véhicules

- Paramètres
  - h horizon de temps
  - (o^k, d^k) dépôts origine et destination du véhicule k
  - $[e_i, l_i]$ fenêtre de temps du client i : le service doit commencer entre e_i et l_i (par convention, on notera $l_{o^k} = 0$, $e^k_d = 0$ et $l_{d^k} = h$)
  - p_i profit associé au service du client optionnel i
  - $\overline{\tau}_{ij}$ temps de parcours maximal entre i et j
  - $\overline{\sigma}_i$ temps de service maximal au client i
  - T grande constante de temps

- Variables
  - x^k_i booléen indiquant si le client obligatoire i est servi par le véhicule k
  - y^k_{ij} booléen indiquant si le client obligatoire i est servi juste avant le client obligatoire j par le véhicule k
  - t_i heure de début de service chez le client obligatoire i

Avec les notations mentionnées ci-dessus, on peut formuler le problème comme un programme à variables mixtes ($M4.1$) :

$$\min \sum_{k \in K} \sum_{i \in M \cup \{o^k\}} \sum_{j \in M \cup \{d^k\}} \overline{\tau}_{ij} y_{ij}^k$$

sujet à :

$$\sum_{k \in K} x_i^k = 1 \qquad \forall i \in M \tag{5.1}$$

$$\sum_{j \in M \cup \{d^k\}} y_{ij}^k = x_i^k \qquad \forall i \in M, k \in K \tag{5.2}$$

$$\sum_{i \in M \cup \{o^k\}} y_{ij}^k = x_j^k \qquad \forall j \in M, k \in K \tag{5.3}$$

$$e_i \leqslant t_i \leqslant l_i \qquad \forall i \in M \cup \{o^k; d^k\} \tag{5.4}$$

$$t_j \geqslant t_i + \overline{\sigma}_i + \overline{\tau}_{ij} + \sum_{k \in K} (y_{ij}^k - 1)T \qquad \forall i \in M \cup \{o^k\}, j \in M \cup \{d^k\} \tag{5.5}$$

$$y_{ij}^k \in \{0; 1\} \qquad \forall k \in K, i \in M \cup \{o^k\}, j \in M \cup \{d^k\}$$

$$x_i^k \in \{0; 1\} \qquad \forall i \in M, k \in K$$

$$t_i \geqslant 0 \qquad \forall i \in M \cup \{o^k; d^k\}$$

Les contraintes (5.1) expriment le fait que chaque client obligatoire doit être servi une et une seule fois. Les contraintes (5.2) et (5.3) sont des contraintes de degré entrant et sortant. Les contraintes (5.4) assurent le respect des fenêtres de temps (le service au client i doit commencer dans la fenêtre de temps $[e_i, l_i]$). Enfin, les contraintes (5.5) sont des contraintes de précédence temporelle, qui assurent l'élimination des sous-tours. Ce modèle est résolu de façon exacte à l'aide d'un solveur commercial, étant donné que le nombre de clients obligatoires est faible dans les instances considérées.

5.1.2 Phase II : Insertion des clients optionnels

Une fois le squelette de routes servant les clients obligatoires construit, on dispose pour chaque véhicule d'une liste ordonnée de clients obligatoires à desservir. Afin d'améliorer la qualité de service, on met à jour les heures de début de service au plus tôt et au plus tard, e_i et l_i, associées au client obligatoire i. Elles correspondent à l'heure de début de service respectivement dans le meilleur et le pire des cas. On pose donc $l_i = t_i$ et on calcule e_i avec les temps de parcours et de service minimaux (en s'assurant que e_i respecte la fenêtre de

temps d'origine) en fonction des routes du squelette.

Dans cette phase, on introduit le concept de *segment* comme étant une portion de route entre deux clients obligatoires successifs (le dépôt origine et le dépôt destination de chaque véhicule sont considérés comme des clients obligatoires). Le segment p a trois caractéristiques : une origine o^p, une destination d^p et une longueur Δ^p donnée par la formule $\Delta^p = l_{d^p} - e_{o^p} - \underline{\sigma}_{o^p}$ où l_{d^p} désigne l'heure de début de service au plus tard chez le client d^p, $\underline{\sigma}_{o^p}$ le temps de service minimal au client origine o^p et e_{o^p} l'heure de début de service au plus tôt chez le client o^p.

Avec ce nouveau concept de segment, le problème d'insertion des clients optionnels dans le squelette consiste à établir des routes sur chaque segment du squelette, tout en s'assurant, sur chaque segment p, que la longueur de la route n'excède pas Δ^p. Deux fonctions objectif sont considérées dans cette phase d'insertion : la maximisation du profit associé à la desserte des clients optionnels et ensuite, la minimisation du temps de parcours total. Pour traiter de ces objectifs, on utilise la méthode classique de la somme pondérée pour les combiner en un seul où α désigne le poids associé aux temps de parcours ($\alpha \in [0; +\infty]$). Pour définir ce problème, on utilise les mêmes notations que précédemment plus celles-ci :

- Ensembles
  - P ensemble de segments sur tous les véhicules. Les segments sont ordonnés comme suit : le segment $p+1$ a pour origine d^p (le client destination du segment p).

- Paramètres
  - α pondération des temps de parcours dans la fonction objectif
  - Δ^p longueur du segment p
  - $\underline{\tau}_{ij}$ temps de parcours minimal entre i et j
  - $\underline{\sigma}_i$ temps de service minimal au client i
  - $\tilde{\tau}_{ij}$ temps de parcours de référence entre i et j
  - $\tilde{\sigma}_i$ temps de service de référence au client i

- Variables
  - x_i^p booléen indiquant si le client optionnel i est servi sur le segment p
  - y_{ij}^p booléen indiquant si i est servi juste avant j sur le segment p

Avec les notations ci-dessus, le problème peut être formulé sous forme du modèle ($M4.2$) suivant :

$$\max . \sum_{p \in P} \sum_{i \in O} p_i x_i^p - \alpha \sum_{p \in P} \sum_{i \in O \cup \{o^p\}} \sum_{j \in O \cup \{d^p\}} \tilde{\tau}_{ij} y_{ij}^p$$

sujet à :

$$\sum_{p \in P} x_i^p \leqslant 1 \qquad \forall i \in O \tag{5.6}$$

$$\sum_{j \in O \cup \{d^p\}} y_{ij}^p = x_i^p \qquad \forall i \in O, p \in P \tag{5.7}$$

$$\sum_{i \in O \cup \{o^p\}} y_{ij}^p = x_j^p \qquad \forall j \in O, p \in P \tag{5.8}$$

$$\sum_{i \in O \cup \{o^p\}} \sum_{j \in O \cup \{d^p\}} \tilde{\tau}_{ij} y_{ij}^p + \sum_{i \in O} \tilde{\sigma}_i x_i^p \leqslant \Delta^p \qquad \forall p \in P \tag{5.9}$$

$$\sum_{i \in O \cup \{d^p\}} y_{o^p i}^p = 1 \qquad \forall p \in P \tag{5.10}$$

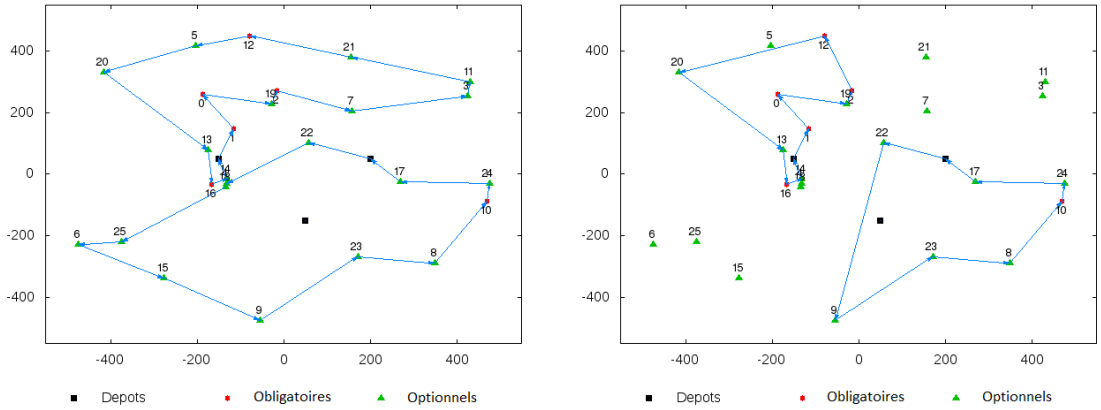
$$\sum_{i \in S} \sum_{j \in S} y_{ij}^p \leqslant \sum_{i \in S \setminus \{l\}} x_i^p \qquad \forall S \subset O, |S| \geqslant 2, \forall l \in S \tag{5.11}$$

$$x_i^p, y_{ij}^p \in \{0; 1\} \qquad \forall i \in O \cup \{o^p\}, j \in O \cup \{d^p\}, p \in P$$

Les contraintes (5.6) expriment que chaque client optionnel est servi au plus une fois. Les contraintes (5.7) et (5.8) sont des contraintes de degré entrant et sortant. Les contraintes (5.9) forcent la longueur d'une route sur un segment à être inférieure à la longueur du segment. Les contraintes (5.10) assurent que la route quitte bien le client origine du segment. Les contraintes (5.11) sont des contraintes d'élimination de sous-tours. Elles sont nécessaires dans ce modèle car les clients optionnels n'ont pas de fenêtre de temps.
Concernant les valeurs de référence pour les temps de parcours et de service $\tilde{\tau}_{ij}$ et $\tilde{\sigma}_i$, plusieurs choix sont possibles. Si on choisit les valeurs minimales (estimés optimistes), on peut obtenir des solutions avec des routes vides tandis que certaines routes servent un grand nombre de clients. C'est une solution admissible tant que l'on se place dans le cas optimiste. Mais après avoir été confrontée à la réalité (durant l'étape d'exécution), de nombreux clients restent non desservis alors même qu'une route demeure vide (cf. exemple figure 5.1).

Pour éviter cet écueil, on doit répartir la charge de travail entre les véhicules lors de l'insertion des clients optionnels. On décompose donc l'insertion des clients optionnels en deux temps :

1. Insertion des clients optionnels avec **estimation pessimiste** des temps de parcours et de service.
2. Insertion des clients optionnels avec **estimation optimiste** des temps de parcours et

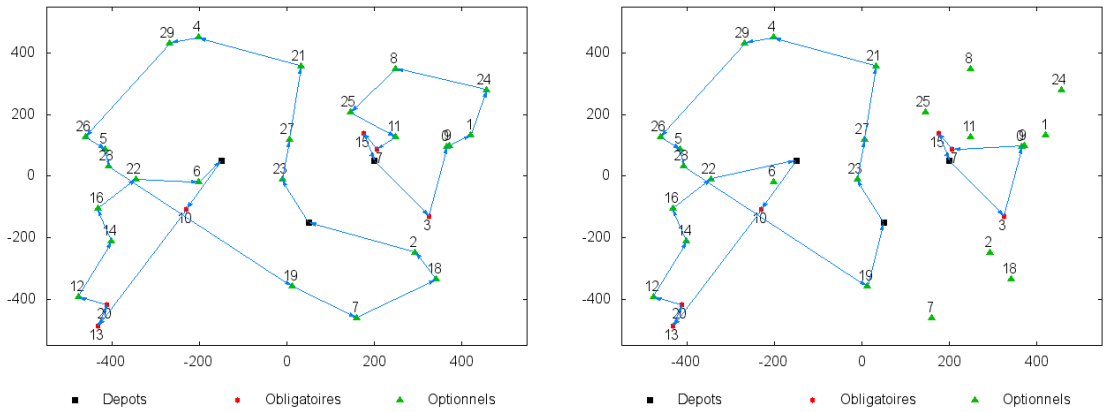


(a) Après insertion des optionnels (b) Après simulation

Figure 5.1 Insertion des clients optionnels avec estimés optimistes

de service, en gardant l'affectation des clients aux véhicules obtenue en 1.

Cette méthode fournit de bons résultats qui seront présentés section 5.3. Toutefois, comme on peut le voir figure 5.2, cette méthode tend à planifier des routes qui seront bouleversées lors de la simulation. En effet, en choisissant de procéder à l'insertion des clients optionnels en deux temps avec estimés pessimistes puis optimistes, on construit des routes sur des segments avec une contrainte de durée correspondant à la longueur de ces segments. Mais si on considère la route d'un véhicule comprenant plusieurs segments, cette route aura alors une contrainte de durée correspondant à la somme des longueurs des segments associés, i.e. $l_{d^p} - e_{o^p} - \underline{\sigma}_{o^p}$ (estimé optimiste de la longueur du segment p). La route obtenue peut donc être beaucoup plus longue que l'horizon de temps. Ainsi, la probabilité de faisabilité de certaines routes peut parfois chuter à 20%. Ce qui a pour conséquence, lorsque les temps de parcours et de service réels sont révélés, de changer complètement les routes planifiées. On choisit donc d'assurer une probabilité de faisabilité minimale. L'insertion des clients optionnels consiste, à présent, à insérer les clients optionnels avec **estimés pessimistes** des temps de service et de parcours. Ensuite, à réparer et améliorer la solution obtenue, tout en assurant que chaque route conserve une probabilité suffisante d'être réalisable (plus grande qu'un seuil donné).
On propose un algorithme de branch and cut pour l'insertion des clients optionnels en ajoutant dynamiquement les contraintes d'élimination de sous-tours (5.11). Comme l'insertion des clients optionnels avec estimés pessimistes sur des petites instances avec un algorithme de branch and cut est trop longue (cf. tableau 4.5), on propose des méthodes d'accélération pour améliorer les temps de calcul. De plus, comme le modèle d'insertion des clients optionnels dans le squelette présente une structure bloc angulaire (hormis un ensemble de contraintes liantes), et afin de résoudre des problèmes de plus grande taille, nous utilisons une méthode



(a) Après insertion des optionnels (b) Après simulation

Figure 5.2 Insertion des clients optionnels avec estimés pessimistes puis optimistes

de relaxation lagrangienne. Dans ce qui suit, nous détaillerons dans un premier temps les différentes techniques d'accélération dans la section 5.1.2, puis nous présenterons la méthode de relaxation lagrangienne dans la section 5.1.2 et enfin, décrirons les algorithmes de réparation et d'amélioration dans la section 5.1.2.

Techniques d'accélération

Comme nous l'avons mentionné plus haut, la résolution du modèle ($M4.2$) avec un algorithme de branch and cut peut s'avérer très chronophage. Afin d'améliorer les temps de calcul, on propose dans cette section différentes techniques d'amélioration : du prétraitement, une heuristique d'insertion pour construire une solution initiale, des « reachability cuts » et des inégalités d'élimination de sous-ensembles.

Prétraitement

Une méthode d'accélération très répandue est de faire du prétraitement, afin de fixer des variables avant de commencer le branch and cut. Ce prétraitement est basé sur l'idée que, lors de l'insertion des clients optionnels dans le squelette, des clients ne peuvent pas être servis sur certains segments. Afin d'éviter cela, en comparant les temps de service et de parcours maximaux avec les longueurs des segments, on obtient des conditions qui nous permettent de fixer la valeur de certaines variables ou de renforcer la formulation en ajoutant des inégalités valides :

Proposition 1. *Les clauses suivantes sont valides :*

$$Si \qquad \overline{\tau}_{o^p j} + \overline{\sigma}_j + \overline{\tau}_{j d^p} \geqslant \Delta^p, \quad alors\ x_j^p = 0 \tag{5.12}$$

$$Si \qquad \overline{\tau}_{o^p i} + \overline{\sigma}_i + \overline{\tau}_{ij} + \overline{\sigma}_j + \overline{\tau}_{j d^p} \geqslant \Delta^p, \quad alors\ y_{ij}^p = 0 \tag{5.13}$$

$$Si \qquad \left.\begin{array}{l} \overline{\tau}_{o^p i} + \overline{\sigma}_i + \overline{\tau} + \overline{\sigma}_j + \overline{\tau}_{j d^p} \geqslant \Delta^p \\ \overline{\tau}_{o^p j} + \overline{\sigma}_j + \overline{\tau} + \overline{\sigma}_i + \overline{\tau}_{i d^p} \geqslant \Delta^p \end{array}\right\}, \quad alors\ x_i^p + x_j^p \leqslant 1 \tag{5.14}$$

La clause (5.12) exprime le fait que le client i ne peut être visité sur le segment p si le temps total nécessaire pour servir uniquement i sur le segment p (temps de parcours de o^p vers i, temps de service du client i et temps de parcours de i vers d^p) dépasse la longueur Δ^p du segment p. De même, la clause (5.13) indique que le client j ne peut être servi après le client i si le temps total nécessaire pour servir uniquement les clients i et j dans cet ordre sur le segment p (temps de parcours de o^p vers i, temps de service du client i, temps de parcours de i vers j, temps de service du client j et temps de parcours de j vers d^p) dépasse la longueur Δ^p de ce segment. Enfin, la clause (5.14) est une inégalité valide basée sur la clause (5.13). Cette clause indique que si le temps total nécessaire pour visiter i et j dans cet ordre et celui nécessaire pour visiter j et i dans cet ordre excèdent la longueur Δ^p du segment p, alors i et j ne peuvent être desservis ensemble sur le segment p. En effet, en appliquant la clause (5.13) aux conditions de la clause (5.14), on obtient $y_{ij}^p = 0$ et $y_{ji}^p = 0$ (i.e. j ne peut être servi après i et i ne peut être servi après j sur le segment p). Donc on conclut que i et j ne peuvent être servis tous les deux sur le segment p.

Heuristique d'insertion

Afin de construire une solution initiale pour le branch and cut, on propose une heuristique basée sur l'insertion des clients. Cette heuristique peut être décrite comme suit. On calcule les coûts d'insertion de chaque client sur chaque segment (si un client i ne peut être inséré sur un segment p, le coût d'insertion associé c_i^p sera $+\infty$). Ensuite, on résout un problème d'affectation (affectation des clients aux segments). Etant donnée la solution de ce problème, on dispose pour chaque segment d'une route servant potentiellement un client optionnel. Donc les coûts d'insertion peuvent avoir changé pour les clients non affectés. Ce processus est ensuite réitéré jusqu'à ce qu'un des critères d'arrêt suivants soit satisfait :

i) Tous les clients optionnels ont été insérés

ii) Aucun client optionnel n'a pu être inséré à cette étape

Reachability Cuts

Afin d'accélérer l'algorithme de branch and cut, on propose de renforcer les contraintes d'élimination de sous-tours (5.11) présentes dans le modèle ($M4.2$). Elles peuvent être reformulées comme suit. Soit $\delta^-(S) = \{(i,j)|i \notin S, j \in S\}$ pour un ensemble de clients S, on a :

$$\sum_{(k,l)\in\delta^-(S)} y_{kl}^p \geqslant x_i^p \quad \forall S \subset O, |S| \geqslant 2, \forall i \in S, p \in P \tag{5.15}$$

Les contraintes (5.15) sont connues pour être équivalentes aux contraintes (5.11). Lysgaard [50] propose de renforcer ces contraintes en définissant les « reachability cuts ». En suivant l'exemple de Lysgaard [50], on définit l'ensemble d'accessibilité A_i^{p-} comme étant l'ensemble minimal d'arcs permettant d'accéder au client i depuis l'origine du segment p. Par exemple, si (o^p, a, b, i) est un chemin possible (au regard de la contrainte de durée maximale) de o^p vers i, alors $\{(o^p, a); (a, b); (b, i)\} \subset A_i^{p-}$ tant que la matrice des temps satisfait l'inégalité triangulaire. Ensuite, les contraintes (5.15) peuvent être renforcées en considérant les *reachability cuts* suivantes :

$$\sum_{(k,l)\in\delta^-(S)\cap A_i^{p-}} y_{kl}^p \geqslant x_i^p \quad \forall S \subset O, |S| \geqslant 2, \forall i \in S, p \in P \tag{5.16}$$

En d'autres termes, soit p un segment, i un client et S un ensemble de clients (avec $o^p \notin S$ et $i \in S$), si le client i est servi sur le segment p, alors il existe un chemin depuis l'origine du segment o^p vers i, et chaque arc de ce chemin appartient à A_i^{p-}. En particulier, comme $i \in S$ et $o^p \notin S$, il existe au moins un arc de A_i^{p-} entrant dans S. Ces contraintes sont clairement plus fortes que les contraintes d'élimination classiques (SEC) comme $\delta^-(S) \cap A_i^{p-} \subseteq \delta^-(S)$. Dans les contraintes (5.16), on peut voir que le nombre de termes dans le membre de gauche dépend de la taille de A_i^{p-}. Ainsi, quand l'ensemble d'accessibilité A_i^{p-} est grand, la contrainte (5.16) n'est pas très forte, comparée à la contrainte SEC associée. Pour cette raison, dans un premier temps, on choisit de générer ces nouvelles contraintes seulement sur les segments dont la longueur n'excède pas un seuil L_{max} et de générer les contraintes classiques d'élimination de sous-tours sur les autres segments. Pour séparer les reachability cuts, on détermine a priori l'ensemble d'arcs A_i^{p-} pour chaque $i \in O$. Dans la phase de prétraitement, on crée une liste C^p de clients qui peuvent être servis sur chaque segment p (relativement à la longueur des segments).

Ensuite, pour chaque client i et segment p, on parcourt la liste des arcs (j, k) tels que $j \in C^p$ et $k \in C^p$. Si le chemin (o^p, j, k, i) est réalisable sur le segment p, on ajoute les arcs correspondant à l'ensemble A_i^{p-} s'ils ne lui appartiennent pas déjà. Ensuite, pour identifier les reachability cuts violées, on considère tous les couples possibles (client i, segment p) et,

pour chacun d'entre eux, on résout un problème de flot maximum sur le graphe support $G = (\{j|j \in C^p\} \cup \{o^p\}, A_i^{p-})$.

Inégalités d'élimination de sous-ensembles

Une autre technique d'accélération consiste à utiliser une famille d'inégalités valides appelées les inégalités d'élimination de sous-ensembles. Ces inégalités proviennent des contraintes (5.9).

Proposition 2. *Soit un ensemble S de clients et un segment p, soit $L^p(S)$ la longueur du plus court chemin du noeud o^p au noeud d^p desservant tous les clients contenus dans S. Si $L^p(S) > \Delta^p$, alors l'inégalité valide d'élimination de sous-ensembles :*

$$x^p(S) \leqslant |S| - 1 \tag{5.17}$$

est valide.

Notons que pour $|S| = 2$, on a $x_i^p + x_j^p \leqslant 1$. Ces inégalités ont déjà été générées lors du prétraitement. Dans notre algorithme de branch and cut, ces inégalités sont générées pour $3 \leqslant |S| \leqslant S_{max}$.

Les contraintes (5.17) sont séparées de façon heuristique. Etant donnée une solution $(\tilde{x}, \tilde{y})$ et un segment p, on range dans un premier temps les clients i vérifiant $\tilde{x}_i^p > 0$ par ordre décroissant des $\tilde{x}_i^p$. A partir de cette liste ordonnée, on identifie les k-uplets $\{i_1, i_2, ..., i_k\}$ vérifiant $\tilde{x}_{i_1}^p + \tilde{x}_{i_2}^p + ... + \tilde{x}_{i_k}^p > k - 1$. Pour chacun de ces k-uplets, on regarde si $(o^p, i_1, i_2, ..., i_k, d^p)$ est admissible. Si tel est le cas, il n'y a pas d'inégalité de sous-ensemble violée pour ce k-uplet et ce segment. Sinon, on identifie le plus court chemin de o^p vers d^p desservant tous les clients du k-uplet. Si le plus court chemin n'est pas admissible, on a identifié une inégalité d'élimination de sous-ensemble violée de taille k avec $S = \{i_1, i_2, ..., i_k\}$.

Relaxation lagrangienne

On peut observer que le modèle d'insertion des clients optionnels dans le squelette présente une structure bloc-angulaire (avec pour seules contraintes liantes les contraintes (5.6)) et peut être décomposé en sous-problèmes (un par segment) si on ne prend pas en compte les contraintes (5.6). Une approche de résolution consiste donc à relâcher ces contraintes liantes et à leur attribuer un multiplicateur de Lagrange (i.e. une pénalité) dans la fonction objectif, obtenant ainsi la relaxation lagrangienne $R(u)$ pour une valeur donnée de $u = (u_1, u_2, ..., u_{|O|})$:

$$\max \sum_{p \in P} \sum_{i \in O} p_i x_i^p - \alpha \sum_{p \in P} \sum_{i \in O \cup \{o^p\}} \sum_{j \in O \cup \{d^p\}} \tilde{\tau}_{ij} y_{ij}^p + \sum_{i \in O} u_i (1 - \sum_{p \in P} x_i^p)$$

sujet aux contraintes (5.7) à (5.11)

Afin de pouvoir décomposer ce problème en sous-problèmes, on reformule la fonction objectif comme suit :

$$\max \sum_{p\in P}\sum_{i\in O}(p_i - u_i)x_i^p - \alpha \sum_{p\in P}\sum_{i\in O\cup\{o^p\}}\sum_{j\in O\cup\{d^p\}} \tilde{\tau}_{ij}y_{ij}^p + \sum_{i\in O} u_i$$

Comme le dernier terme $\sum_{i\in O} u_i$ est constant pour un vecteur u donné, on peut décomposer $R(u)$ en sous-problèmes. Pour un segment p, le sous-problème $R^p(u)$ s'écrit :

$$\max \sum_{i\in O}(p_i - u_i)x_i^p - \alpha \sum_{i\in O\cup\{o^p\}}\sum_{j\in O\cup\{d^p\}} \tilde{\tau}_{ij}y_{ij}^p$$

sujet à :

$$\sum_{j\in O\cup\{d^p\}} y_{ij}^p = x_i^p \qquad \forall i \in O \tag{5.18}$$

$$\sum_{i\in O\cup\{o^p\}} y_{ij}^p = x_j^p \qquad \forall j \in O \tag{5.19}$$

$$\sum_{i\in O\cup d^p} y_{o^p i}^p = 1 \tag{5.20}$$

$$\sum_{i\in O\cup\{o^p\}}\sum_{j\in O\cup\{d^p\}} \tilde{\tau}_{ij}y_{ij}^p + \sum_{i\in O}\tilde{\sigma}_i x_i^p \leqslant \Delta^p \tag{5.21}$$

$$\sum_{i\in S}\sum_{j\in S} y_{ij}^p \leqslant \sum_{i\in S\setminus\{l\}} x_i^p \qquad \forall S \subset O, |S| \geqslant 2, \forall l \in S \tag{5.22}$$

$$y_{ij}^p \in \{0;1\} \qquad \forall i \in O\cup\{o^p\}, j \in O\cup\{d^p\}$$

$$x_i^p \in \{0;1\} \qquad \forall i \in O$$

Afin de procéder à l'insertion des clients optionnels dans le squelette, comme le vecteur u peut prendre un grand nombre de valeurs, on doit résoudre le problème du dual lagrangien, qui peut être formulé comme suit :

$$\min_u \left(\max_x \sum_{p\in P}\sum_{i\in O} p_i x_i^p - \alpha \sum_{p\in P}\sum_{i\in O\cup\{o^p\}}\sum_{j\in O\cup\{d^p\}} \tilde{\tau}_{ij}y_{ij}^p + \sum_{i\in O} u_i(1 - \sum_{p\in P} x_i^p) \right)$$

Pour résoudre le dual lagrangien, on applique l'algorithme des sous-gradients. Pour cet algorithme, on a besoin d'une bonne borne inférieure (afin d'assurer la convergence de la méthode). Pour ce faire, on construit une solution réalisable en utilisant l'heuristique d'insertion présenté dans la section 5.1.2 et on améliore la solution ainsi obtenue en utilisant des opérateurs de type string exchange, arc exchange et relocalisation (en interdisant le déplacement

de tout client obligatoire). Soit $\underline{\omega}$ la valeur de cette solution (valeur de la borne inférieure), $\epsilon \in [0;2]$ et $u^0 = (0, 0, ..., 0)$, on procède comme suit à chaque itération k de l'algorithme du sous-gradient :

Iteration k :

- $u = u^k$
- Pour chaque segment $p \in P$, résoudre $R^p(u^k)$ avec estimés pessimistes pour les temps de service et de parcours, en utilisant la méthode de branch and cut décrite précédemment et les techniques d'accélération. On obtient ainsi la solution $(x_i^p)_{i \in O, p \in P}$.
- Avec les valeurs $(x_i^p)_{i \in O, p \in P}$ de la solution obtenue, calculer la valeur $z(u^k)$ de la solution de $R(u^k)$.
- Calculer la longueur de pas μ_k et la direction du pas D_i^k pour tout $i \in O$ avec les formules :

$$\mu_k = \frac{\epsilon(z(u^k) - \underline{\omega})}{\sum_{i \in O} \left(1 - \sum_{p \in P} x_i^p\right)^2} \qquad \text{et} \qquad D_i^k = 1 - \sum_{p \in P} x_i^p$$

- Calculer le nouveau vecteur u^{k+1} avec $u_i^{k+1} = \max(u_i^k + \mu_k D_i^k, 0)$.
- $k \leftarrow k + 1$

L'algorithme des sous-gradients finit soit quand le gap entre la meilleure borne supérieure $\min_k z(u^k)$ et la meilleure borne inférieure $\max_k \underline{\omega}^k$ descend en-dessous d'un certain seuil G_{end}, soit après un nombre fixé d'itérations I_{max}. On récupère alors la meilleure solution réalisable trouvée (soit la meilleure borne inférieure).

Afin d'accélérer la convergence, on propose deux modifications de la méthode : utiliser la méthode de Kiev pour le calcul des sous-gradients et calculer des bornes inférieures à chaque itération afin de remplacer $\underline{\omega}$ par la meilleure borne inférieure.

Dans la méthode de Kiev, à l'itération k, au lieu de prendre en compte seulement le sous-gradient en u^k, on propose de prendre en compte une combinaison convexe du sous-gradient en u^k et du sous-gradient en u^{k-1}. On introduit donc un coefficient $\beta \in [0;1]$ pour cette combinaison convexe, et on note $(x_i^{p(k)})_{i \in O, p \in P}$ la solution obtenue à l'itération k. A présent, les longueurs et directions des pas sont obtenues avec les formules :

$$\mu_k = \frac{\epsilon(z(u^k) - \underline{\omega})}{\sqrt{\sum_{i \in O} \left(1 - \sum_{p \in P} x_i^{p(k)}\right)^2}} \qquad \text{et} \qquad D_i^k = \frac{1 - \sum_{p \in P} \left(\beta x_i^{p(k)} + (1-\beta) x_i^{p(k-1)}\right)}{\sqrt{\sum_{i \in O} \left(1 - \sum_{p \in P} \left(\beta x_i^{p(k)} + (1-\beta) x_i^{p(k-1)}\right)\right)}}$$

Aussi, on calcule une nouvelle borne inférieure $\underline{\omega}_k$ à chaque itération. Avec l'ensemble de ces bornes inférieures, on peut alors sélectionner la meilleure borne inférieure à chaque itération

($\max_k \underline{\omega}_k$) et utiliser cette meilleure borne inférieure dans les formules ci-dessus au lieu de la borne inférieure initiale (on pose $\underline{\omega} = \max_k \underline{\omega}_k$). Pour calculer une nouvelle borne inférieure à l'itération k, on répare la solution obtenue à l'itération k en supprimant les doublons. Puis on améliore la solution en procédant à des string exchange, arc exchange et relocalisation successives (sans déplacer les clients obligatoires). On obtient ainsi une solution réalisable, i.e. une nouvelle borne inférieure comme il s'agit d'un problème de maximisation.

Cette méthode heuristique, basée sur la relaxation lagrangienne, nous permet d'obtenir une borne inférieure de qualité pour la solution au problème d'insertion des clients optionnels dans le squelette. De plus, étant donnée sa rapidité de convergence, elle rend possible la résolution d'instances de plus grande taille.

Comme nous résolvons ici le problème segment par segment avec des estimés pessimistes, après la relaxation lagrangienne, on peut obtenir, comme précédemment, une solution avec une probabilité très faible d'être réalisable. On applique donc les algorithmes de réparation et d'amélioration à la solution obtenue (cf. section 5.1.2).

Algorithmes de réparation et d'amélioration de solution

Comme nous l'avons mentionné précédemment, lors de l'insertion des clients optionnels dans le squelette, on peut obtenir des routes avec une faible probabilité de faisabilité. Dans cette phase, on essaye d'insérer des clients optionnels dans le squelette tout en assurant, sur chaque segment p, que la longueur d'une route n'excède pas la longueur du segment Δ^p. Mais la longueur d'un segment est définie par la formule $\Delta^p = l_{d^p} - e_{o^p} - \underline{\sigma}_{o^p}$. Dans le pire des cas, la longueur du segment est plutôt $l_{d^p} - l_{o^p} - \overline{\sigma}_{o^p}$. Pour cette raison, la probabilité pour une route d'être réalisable peut s'avérer très faible. Afin d'éviter cet écueil, on introduit un nouveau paramètre F comme étant le seuil de réalisabilité. On propose une procédure consistant à réparer dans un premier temps la solution (afin que chaque route ait une probabilité d'être réalisable supérieure ou égale à F). Ensuite, on améliore la solution en essayant d'insérer les clients non desservis et de repositionner les clients tout en assurant la probabilité F pour chaque route.

Une description pseudo-code de la procédure de réparation est donnée dans l'algorithme 1. Soit P_k l'ensemble des segments associés au véhicule k. Etant donnée la route prévue pour un véhicule k (et les segments associés P_k), en utilisant les distributions de probabilité des temps de service et de parcours, on peut calculer pour chaque client obligatoire les heures d'arrivée possibles ainsi que les probabilités associées. $calculerProbaFaisable(o^p, d^p)$ retourne la probabilité pour une route d'être réalisable en d^p (probabilité d'arriver avant l_{d^p}), connaissant les heures d'arrivée possibles et les probabilités associées en o^p. On note $clientPlusGrandDetour(a, b)$ la fonction qui renvoie le client situé entre a et b (a et b exclus)

générant le plus grand détour du point de vue des temps de parcours. L'algorithme de réparation consiste à enlever le client générant le plus grand détour jusqu'à ce que la probabilité de faisabilité soit suffisante.

pour chaque $k \in K$ **faire**
  pour chaque $p \in P_k$ **faire**
    $proba \leftarrow calculerProbaFaisable(o^p, d^p)$;
    tant que $proba < F$ **faire**
      $c \leftarrow clientPlusGrandDetour(o^p, d^p)$;
      enlever le client c;
      $proba \leftarrow calculerProbaFaisable(o^p, d^p)$;
    fin
  fin
fin

Algorithm 1: Réparation de la solution

La description en pseudo-code de l'algorithme d'amélioration est donnée dans l'algorithme 2. Dans cet algorithme, soient N le nombre total de clients, L une liste vide et U la liste des clients non desservis. $clientPlusGrandDetour(L)$ retourne le client n'appartenant pas à L, et générant le plus grand détour dans la solution. $meilleureInsertion(c)$ renvoie la meilleure insertion possible (du point de vue des temps de parcours) pour le client c dans la solution (une insertion étant définie par une route, une position et un profit d'insertion). Une insertion est impossible si elle génère une route avec une probabilité de réalisabilité plus petite que F ou si des fenêtres de temps ne sont pas respectées. $meilleureInsertion(U)$, avec U une liste de clients, est une méthode consistant à : récupérer la meilleure insertion possible sur toutes les routes et tous les clients de U et à procéder à cette insertion si elle présente un intérêt, jusqu'à ce qu'aucun client ne puisse plus être inséré. Dans cet algorithme, une insertion ne sera pas considérée si le seuil de réalisabilité n'est pas respecté.

Pour procéder à la réparation et à l'amélioration d'une solution, on propose deux variantes. Dans la première (que l'on appellera $V1$), on procède à une exécution de l'algorithme de réparation puis on exécute l'algorithme d'amélioration une seule fois. Dans la deuxième variante (appelée $V2$), on procède à une exécution de l'algorithme de réparation mais on exécute l'algorithme d'amélioration jusqu'à ce que la solution ne puisse plus être améliorée.

5.2 Etape d'exécution

Au début de l'étape d'exécution, on dispose, pour chaque véhicule, d'une route contenant à la fois des clients obligatoires et des clients optionnels, obtenue à la fin de l'étape de planification. Le but de l'étape d'exécution est d'ajuster les routes planifiées à la réalité.

$U \leftarrow ClientsNonDesservis()$;
$meilleureInsertion(U)$;
$L \leftarrow []$;
tant que $taille(L) < N - taille(U)$ **faire**
  $c \leftarrow clientPlusGrandDetour(L)$;
  $(bestRoute, bestPosition, bestProfit) \leftarrow meilleureInsertion(c)$;
  si $meilleurProfit > 0$ **alors**
    insérer le client c dans la route $bestRoute$ en position $bestPosition$;
    enlever le client c de la liste U;
  fin
  ajouter c dans la liste L;
fin

Algorithm 2: Amélioration de la solution

Jusqu'ici, nous avons supposé que les temps de service et de parcours étaient soit à leur borne supérieure, soit à leur borne inférieure. Dans cette étape, on prend en compte la stochasticité sur les temps de service et de parcours et on modifie le planning en temps réel pour faire face à cette stochasticité. On suppose que les segments sont rangés comme suit (cf. chapitre 3, figure 4.1) : le segment $p+1$ a pour origine le client obligatoire $o^{p+1} = d^p$ (client obligatoire correspondant à la destination du segment p).
Chaque étape de programmation dynamique correspond à la fin de service chez un client (obligatoire ou optionnel). A chaque étape, on dispose d'une liste de clients optionnels non desservis qui peuvent être servis avant le prochain client obligatoire. Dans ce contexte, deux options sont à considérer : soit le véhicule se rend directement chez le prochain client obligatoire, soit il visite le client optionnel de cette liste qui maximise le profit.

On définit les paramètres suivants :

Γ_j	pénalité de retard au client obligatoire j
l_j	fin de la fenêtre de temps pour le client obligatoire j
τ_{ij}	temps de parcours du client i au client j ($\underline{\tau}_{ij} \leqslant \tau_{ij} \leqslant \overline{\tau}_{ij}$)
σ_i	temps de service au client i ($\underline{\sigma}_i \leqslant \sigma_i \leqslant \overline{\sigma}_i$)

A l'étape k, on définit :

v_k	client où se trouve le véhicule à l'étape k
t_k	heure de fin de service au client v_k
V^p	liste ordonnée de clients optionnels associés au segment p
$\tilde{V}^p = V^p \cup o^p$	liste de clients optionnels associés au segment p + client origine du segment p
$\bar{V}_k^p$	liste de clients optionnels du segment p situés après v_k

Avec ces notations, on propose deux algorithmes distincts de programmation dynamique. Dans le premier algorithme, on considère un seul segment p tandis que l'on prend en compte le reste de la route dans le second.

- Méthode considérant un segment (OS)
  Dans le premier algorithme, quand un véhicule finit de servir un client, deux possibilités s'offrent à lui (cf. chapitre 3, figure 4.2). Soit il se rend directement au prochain client obligatoire d^p, touchant ainsi un revenu correspondant au profit espéré en d^p. Soit il visite le client optionnel $\bar{v}$ de $\bar{V}_k^p$ maximisant le profit espéré (profit asssocié à la visite du client $\bar{v}$ + profit espéré en $\bar{v}$). Dans cet algorithme, le profit espéré au client obligatoire d^p est l'opposé de la pénalité de retard chez ce client. La fonction de revenu de la première méthode peut donc s'écrire comme suit :

  $$f(v_k, t_k, \bar{V}_k^p) = \max\left(E[f(d^p, t_k + \tau_{v_k d^p}, \emptyset)], \max_{\bar{v} \in \bar{V}_k^p}(p_{\bar{v}} + E[f(\bar{v}, t_k + \tau_{v_k \bar{v}} + \sigma_{\bar{v}}, \bar{V}_k^p \backslash \{\bar{v}\})])\right)$$

  avec $f(d^p, t, \emptyset) = -\Gamma_{d^p} \max(t - l_{d^p}, 0)$

- Méthode considérant toute la route (WR)
  Dans le deuxième algorithme, le seul changement concerne le revenu associé au client obligatoire d^p au temps t : ce revenu comprend toujours l'opposé de la pénalité de retard chez ce client mais il comprend aussi le profit espéré associé à la fin de service du client d^p au temps $t+\sigma_{d^p}$. La fonction de revenu du deuxième algorithme s'écrit donc :

  $$f(v_k, t_k, \bar{V}_k^p) = \max\left(E[\hat{f}(d^p, t_k + \tau_{v_k d^p})], \max_{\bar{v} \in \bar{V}_k^p}(p_{\bar{v}} + E[f(\bar{v}, t_k + \tau_{v_k \bar{v}}, \bar{V}_k^p \backslash \{\bar{v}\})])\right)$$

  avec $\hat{f}(d^p, t) = -\Gamma_{d^p} \max(t - l_{d^p}, 0) + f(d^p, t + \sigma_{d^p}, V^{p+1})$

En résolvant ces équations de Bellman, on obtient les seuils de la politique optimale. En effet, pour chaque client, on obtient différents seuils temporels définissant des intervalles de temps. A chacun de ces intervalles est associée une décision optimale (par exemple, si $t \in [t_1, t_2[$, se rendre au client c).

5.3 Expérimentation

5.3.1 Contexte expérimental

L'étape de planification est résolue de façon exacte avec Cplex 12.4 pour des instances contenant jusqu'à 40 clients. Pour l'établissement du squelette, on utilise un algorithme de branch and bound tandis que, pour l'insertion des clients optionnels, on utilise un algorithme de branch and cut dans lequel on ajoute dynamiquement les contraintes d'élimination de sous-tours (5.11). Pour des instances plus grandes (avec 50 clients), on utilise un algorithme de branch and bound pour le squelette, mais on utilise une relaxation lagrangienne pour l'insertion des clients optionnels dans le squelette. Dans les deux cas, à la fin de l'étape de planification, on utilise des algorithmes de réparation et d'amélioration afin d'obtenir des routes avec une probabilité suffisante d'être réalisables. En ce qui concerne l'étape d'exécution, on utilise des outils de programmation dynamique pour trouver la politique optimale. On obtient ainsi, pour chaque client, une liste de seuils et de décisions associées correspondant à la politique optimale. Ensuite, on procède à 100 simulations par instance, chaque simulation consistant d'abord à générer des temps de parcours et de service stochastiques puis, à chaque fois que le service chez un client s'achève, à comparer l'heure de fin de service avec les seuils et à prendre la décision optimale déterminée par l'algorithme de programmation dynamique. Les tests ont été effectués sur une machine avec 4CPU, 2.8GHz et 30Go de RAM.

5.3.2 Réglage des paramètres

Concernant le réglage des paramètres, nous nous sommes inspirés des valeurs choisies par Tricoire [68]. Ainsi, nous considérons un horizon de temps de 8 heures, soit 480 minutes. De plus, comme il propose de considérer une vitesse moyenne de 35km/h, nous avons choisi une vitesse minimale $v_{min} = 20km/h$ et une vitesse maximale $v_{max} = 50km/h$ (observant ainsi une valeur moyenne de 35km/h), valables sur l'ensemble du réseau routier. Après avoir converti ces valeurs v_{min} et v_{max} en unités arbitraires par minute, on calcule le temps de parcours unitaire minimal $\underline{\delta} = \lceil 100/v_{max} \rceil$ et maximal $\overline{\delta} = \lceil 100/v_{min} \rceil$. Les temps de parcours minimaux et maximaux sont ensuite obtenus en utilisant les formules $\underline{\tau}_{ij} = \left\lceil D_{ij}\underline{\delta} \right\rceil$ et $\overline{\tau}_{ij} = \left\lceil D_{ij}\overline{\delta} \right\rceil$. Pour le réglage des temps de service, nous avons considéré que les temps de service

des clients optionnels étaient compris entre 15 et 30 minutes tandis que ceux des clients obligatoires étaient compris entre 30 et 60 minutes. Pour l'insertion des client optionnels, nous avons supposé que le service d'un client optionnel, quel qu'il soit, génère un profit $p_i = 100$. Afin de bien choisir la valeur du paramètre α pour la pondération des temps de parcours dans la fonction objectif de la deuxième phase, nous avons procédé à quelques tests. En choisissant $\alpha = 1$, on s'assure de préférer insérer des clients optionnels au fait de parcourir une moins grande distance. Pour la programmation dynamique, on choisit une pénalité de retard $\Gamma_{d^p} = 5000$ pour tout p. Enfin, pour les simulations, afin d'avoir des temps de parcours et de service inattendus, on diminue la vitesse minimale à 15km/h et on augmente le temps de service maximal pour les clients obligatoires à 90 min. Dans cette section, nous procédons au réglage des paramètres pour la deuxième phase de l'étape de planification (l'insertion des clients optionnels). Nous verrons d'abord le réglage des paramètres du branch and cut, puis celui des méthodes de réparation et d'amélioration, et enfin, nous règlerons les paramètres pour la relaxation lagrangienne.

Paramètres du branch and cut

Dans le cadre de l'algorithme de branch and cut pour l'insertion des clients optionnels dans le squelette des clients obligatoires, nous avons proposé des techniques d'accélération (cf. section 5.1.2) telles que les reachability cuts et les inégalités d'élimination de sous-ensembles. Pour ajuster la valeur de L_{max} (taille maximale des segments sur lesquels on génère des contraintes de type reachability cuts au lieu des contraintes classiques d'élimination de sous-tours), nous avons procédé à plusieurs tests sur des petites instances (contenant 30 clients et 3 véhicules). Les résultats obtenus en faisant varier L_{max} entre 0 et l'horizon de temps ($L_{max} = 480$) sont synthétisés dans le tableau 5.1. Contrairement à ce qui était attendu, on observe dans le tableau 5.1 que les meilleurs temps de calcul sont obtenus pour $L = 480$ (i.e. l'horizon de temps), soit lorsque l'on génère les reachability cuts sur tous les segments. En effet, section 5.1.2 du paragraphe 5.1.2, nous avons mentionné que les contraintes de type reachability cuts sont plus fortes que les contraintes d'élimination de sous-tours. Ainsi, indépendamment de la taille des segments, on préfère générer les reachability cuts plutôt que les contraintes d'élimination de sous-tours.

Pour le choix de la taille maximale S_{max} des inégalités d'élimination de sous-ensembles générées, nous avons également conduit une série de tests synthétisés dans le tableau 5.2. On observe dans ce tableau que les meilleurs temps de calcul sont obtenus pour $S_{max} = 6$, soit lorsque l'on génère des inégalités d'élimination de sous-ensembles contenant au plus 6 clients. On choisira donc $S_{max} = 6$. En effet, pour $S_{max} = 7$, on ne génère quasiment aucune inégalité

Tableau 5.1 Réglage de L_{max} (taille maximale des segments pour les reachability cuts)

Instances	Nombre de clients obligatoires	L=0	L=80	L=160	L=240	L=320	L=400	L=480
C1_1_5		4	5	4	27	14	4	4
C1_2_5		10	11	11	23	23	24	10
C1_3_5	5	39	40	40	46	46	46	29
C1_4_5		1159	1208	1190	609	607	610	263
C1_5_5		55	55	35	57	57	57	41
C1_1_6		69	69	90	97	97	97	68
C1_2_6		139	136	100	64	64	64	82
C1_3_6	6	267	259	263	206	165	166	148
C1_4_6		120	120	120	82	83	83	76
C1_5_6		1024	1033	1041	247	319	322	320
C1_1_7		1219	1230	1427	565	559	561	559
C1_2_7		26	27	23	16	19	19	18
C1_3_7	7	75	75	75	63	47	47	47
C1_4_7		96	97	77	92	92	92	47
C1_5_7		54	54	54	36	37	37	36
C1_1_8		315	317	316	115	117	117	117
C1_2_8		25	25	29	25	62	18	18
C1_3_8	8	13	13	14	24	12	12	12
C1_4_8		76	78	75	80	80	80	76
C1_5_8		23	24	23	23	31	31	31
C1_1_9		101	98	98	35	35	35	35
C1_2_9		28	27	27	25	41	41	41
C1_3_9	9	28	28	28	41	41	24	24
C1_4_9		134	132	132	110	110	80	80
C1_5_9		7	8	8	11	12	6	6

de sous-ensemble de taille 7. On passe donc plus de temps à rechercher des inégalités valides qu'à en générer, ce qui représente une perte de temps. D'autre part, étant donné que l'on ne génère quasiment aucune inégalité de sous-ensembles de taille 7, il est inutile de procéder à des tests avec $S_{max} > 7$.

Tableau 5.2 Réglage de S_{max} (taille maximale des inégalités d'élimination de sous-ensembles)

Instances	Nombre de clients obligatoires	$S_{max} = 3$	$S_{max} = 4$	$S_{max} = 5$	$S_{max} = 6$	$S_{max} = 7$
C1_1_5		4	4	4	4	4
C1_2_5		8	8	8	12	17
C1_3_5	5	27	28	27	26	28
C1_4_5		163	131	177	139	153
C1_5_5		32	34	34	34	46
C1_1_6		61	40	44	40	44
C1_2_6		65	58	59	55	60
C1_3_6	6	124	76	68	80	75
C1_4_6		69	46	45	53	59
C1_5_6		211	105	102	102	102
C1_1_7		368	145	144	144	145
C1_2_7		16	16	16	15	15
C1_3_7	7	28	21	21	21	21
C1_4_7		65	64	51	53	49
C1_5_7		52	33	33	33	33
C1_1_8		66	56	56	56	56
C1_2_8		21	28	21	23	22
C1_3_8	8	17	13	12	13	12
C1_4_8		63	46	58	60	70
C1_5_8		22	23	25	17	16
C1_1_9		31	23	23	23	23
C1_2_9		21	26	25	26	27
C1_3_9	9	28	29	33	26	27
C1_4_9		69	46	42	40	40
C1_5_9		5	7	7	7	6

Paramètres des algorithmes de réparation et d'amélioration

Dans l'étape de planification, lors de l'insertion des clients optionnels, nous avons proposé des algorithmes de réparation et d'amélioration afin de s'assurer que les routes construites aient une probabilité suffisante d'être réalisable (cf. section 5.1.2). Afin de choisir le meilleur réglage du paramètre F (seuil de faisabilité d'une route) et de la variante à utiliser, nous avons procédé à un ensemble de tests dont les résultats sont regroupés dans le tableau 5.3. On y observe que le nombre de clients desservis ainsi que la distance totale parcourue sont plus grands dans la variante V2 tandis que le retard moyen reste comparable dans les deux variantes. On choisira donc la variante V2 afin de privilégier l'insertion des clients optionnels. Quant au choix de la valeur F, on remarque que les résultats sont moins variables pour $F = 0.9$. En effet, dans certains cas, on peut avoir un nombre de clients non desservis nettement plus faible avec $F = 0.8$ et dans d'autres cas, c'est le contraire. Il en va de même pour la distance totale et le retard. Pour favoriser la qualité de la solution obtenue, on choisit donc $F = 0.9$ pour la suite des expérimentations.

Tableau 5.3 Impact de la variante et du paramètre F sur les résultats obtenus

	# de clients oblig.	30 clients								40 clients							
		variante V1				variante V2				variante V1				variante V2			
		F = 0.8		F = 0.9		F = 0.8		F = 0.9		F = 0.8		F = 0.9		F = 0.8		F = 0.9	
		WR	OS	WR	OS	WR	OS	WR	OS	WR	OS	WR	OS	WR	OS	WR	OS
nb moyen de clients non desservis	5	5.82	3.72	5.98	3.93	4.03	3.53	5.33	3.36	10.91	10.13	10.87	10.09	9.74	9.73	9.66	9.66
	6	8.48	8.09	5.53	5.54	8.44	8.00	5.59	5.55	13.49	12.73	12.96	12.96	12.54	12.53	12.69	12.68
	7	6.47	5.84	9.24	8.61	6.41	5.65	5.52	5.51	13.84	13.11	13.28	13.28	13.00	13.02	13.28	13.28
	8	6.24	6.32	7.57	6.17	6.86	6.23	7.55	6.13	13.43	13.42	13.32	13.32	13.40	13.38	13.21	13.22
	9	6.65	6.65	7.08	6.42	6.59	6.59	7.03	6.36	14.93	14.41	14.24	14.33	13.94	14.02	14.51	14.00
distance parcou-rue en moyenne	5	198	209	199	210	206	210	199	211	189	191	189	191	196	196	193	194
	6	211	214	212	212	212	214	212	212	200	202	202	202	205	204	204	204
	7	200	203	201	204	204	207	209	209	191	194	198	198	195	196	198	198
	8	212	213	195	204	209	211	195	204	197	198	198	198	198	198	198	198
	9	202	202	197	202	202	202	197	201	188	189	193	191	192	192	189	191
retard moyen	5	2.03	2.06	2.01	2.05	2.03	2.06	1.99	2.03	1.82	1.85	1.79	1.80	1.80	1.85	1.77	1.81
	6	43.07	43.07	3.09	3.10	43.07	43.12	3.09	3.15	2.59	2.62	2.65	2.65	2.61	2.64	2.65	2.65
	7	2.68	2.76	2.68	2.75	2.71	2.79	2.70	2.77	2.52	2.54	2.41	2.45	2.52	2.54	2.41	2.45
	8	4.76	3.48	3.14	3.24	3.51	3.58	3.18	3.24	3.63	3.67	3.62	3.61	3.59	3.69	3.60	3.61
	9	4.82	4.79	4.65	4.65	4.92	4.89	4.62	4.63	6.69	5.61	6.71	5.60	6.71	5.57	6.75	5.59

Légende :

Variante V1 = une réparation et une amélioration

Variante V2 = une réparation et autant d'améliorations que possible

F = seuil pour la probabilité de faisabilité

WR = stratégie de programmation dynamique considérant toute la route

OS = stratégie de programmation dynamique considérant un seul segment

Paramètres de la relaxation lagrangienne

Dans l'étape de planification, nous avons proposé une méthode heuristique, basée sur la relaxation lagrangienne, pour résoudre le problème d'insertion des clients optionnels dans le squelette (cf.section 5.1.2). Cette méthode heuristique fait appel à l'algorithme du sous-gradient dans lequel interviennent deux paramètres : ϵ et β. Pour le premier paramètre, on choisit $\epsilon = 1$. Pour le réglage de β, nous avons réalisé des tests de convergence en procédant à 100 itérations de l'algorithme du sous-gradient. La moyenne des résultats obtenus sur l'ensemble des instances à 30 clients en faisant varier β entre 0 et 1 avec un pas de 0.1 nous a permis de tracer le graphique 5.3 (pour plus de lisibilité, nous n'avons retenu que quelques unes de ces courbes). Dans ce graphique, on a choisi comme origine de l'axe des ordonnées la moyenne des solutions optimales des instances à 30 clients.

Dans ce graphique, on peut constater que la courbe pour $\beta = 1$ (donc sans la méthode de Kiev) présente une convergence comparable à celle obtenue avec toute autre valeur de β. On choisira donc de ne pas prendre en compte la méthode de Kiev (ou $\beta = 1$). Aussi, on constate que, dès 50 itérations, on atteint le palier de convergence. Comme on applique un algorithme de réparation et d'amélioration après la relaxation lagrangienne, on choisit donc de se limiter à $I_{max} = 50$ itérations. On choisit également d'arrêter l'algorithme si le gap entre la meilleure borne supérieure et la meilleure borne inférieure est inférieur à $G_{end} = 2\%$.

Les résultats obtenus en comparant les différentes variantes de l'algorithme du sous-gradient sont synthétisés dans le tableau 5.4. Dans ce tableau, les en-têtes de colonnes sont les suivants : SG : méthode du sous-gradient classique ; LB : calcul de bornes inférieures à chaque itération ; K : méthode de Kiev. Aussi, dans les deux premières variantes, on fixe le nombre d'itérations à 100 tandis que, dans la dernière variante, on pose $I_{max} = 50$ et $G_{end} = 2\%$.

Pour l'analyse des résultats, on notera SG la première variante (colonne intitulée SG, nb iter = 100), SG, LB la deuxième variante (SG, LB, nb iter = 100) et SG, LB, I_{max} la dernière variante ($SG, LB, I_{max} = 50, G_{end} = 2\%$). On peut constater dans le tableau 5.4 que, pour la variante classique du sous-gradient (SG), le gap entre la meilleure borne supérieure et la meilleure borne inférieure peut aller jusque 30%, tandis qu'il est en-dessous de 8% avec l'ajout des bornes inférieures (SG, LB). De même, le gap entre la meilleure borne supérieure et la solution optimale pouvait atteindre 11% et se situe désormais sous la barre des 8%. Enfin, on remarque que la borne inférieure fournie par la variante SG, LB est à moins de 3% et même, très souvent, à 0% de la solution optimale. Par contre, les temps de calcul de cette variante sont nettement plus élevés qu'avec la variante classique du sous-gradient. D'où l'intérêt de la troisième variante (avec G_{end} et I_{max}), dans laquelle on résout toutes les instances en moins de 8 minutes. De plus, dans cette variante, les gaps sont légèrement détériorés mais on constate tout de même que les bornes inférieures fournies dans cette variante restent à moins de 3.5%

de la solution optimale et, pour la plupart, se trouvent même à moins de 1% de l'optimalité. Ce qui valide notre choix de la troisième variante : sous-gradient avec bornes inférieures, $G_{end} = 2\%$ et $I_{max} = 50$ itérations.

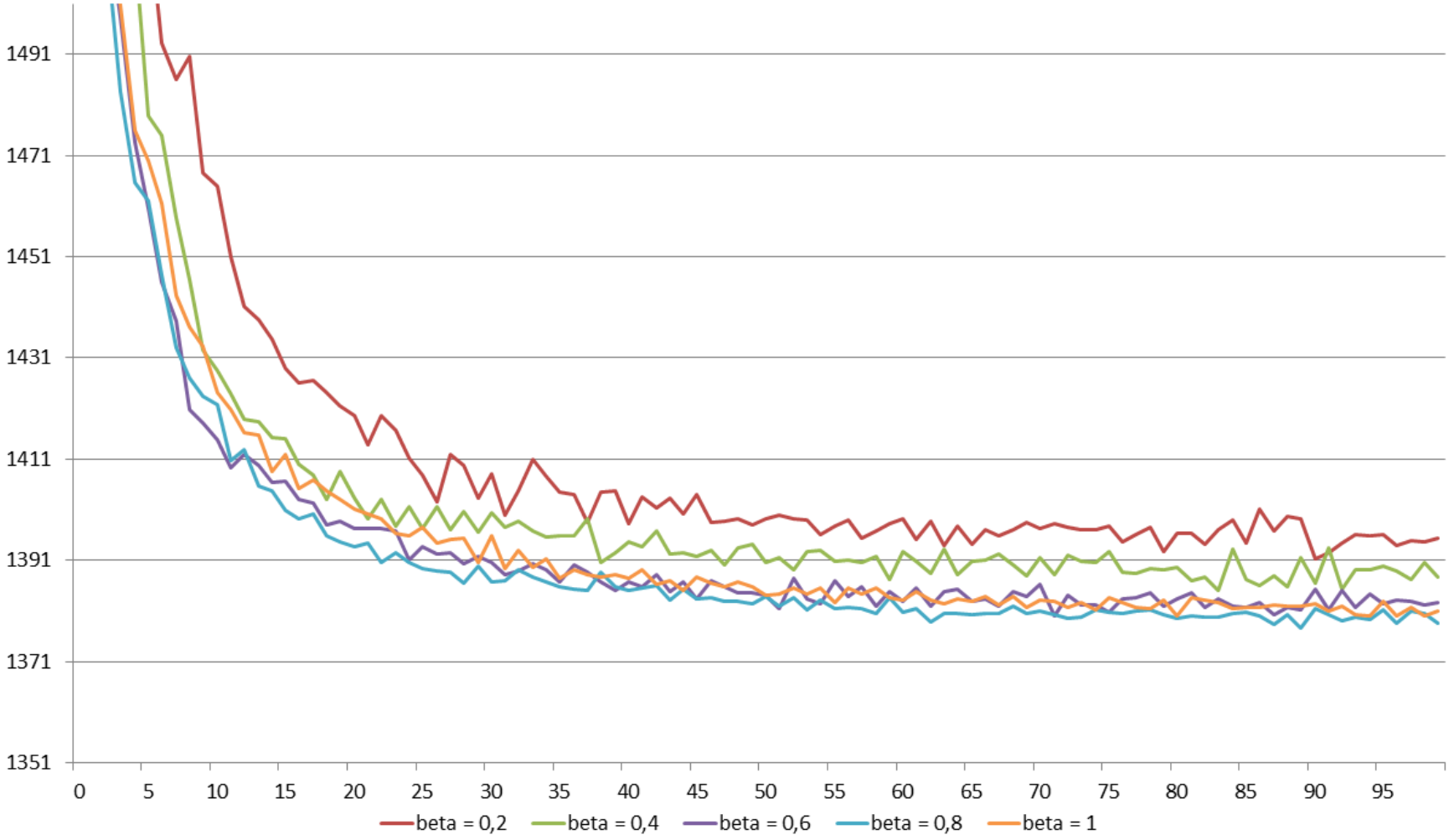


Figure 5.3 Influence du paramètre β sur la convergence de la méthode du sous-gradient

Tableau 5.4 Convergence de la méthode du sous-gradient suivant la variante choisie

Instance	SG (nb iter = 100)				SG, LB (nb iter = 100)				SG, LB ($I_{max} = 50$, $G_{end} = 2\%$)				
	Temps	Gap opt	Gap LB	Gap LB-opt	Temps	Gap opt	Gap LB	Gap LB-opt	Temps	nb Iter	Gap opt	Gap LB	Gap LB-opt
C1_1_5	120	2,03%	12,91%	9,63%	510	0,05%	0,05%	0,00%	294	41	1,87%	1,87%	0,00%
C1_2_5	263	0,39%	3,51%	3,01%	447	0,45%	0,45%	0,00%	64	12	1,95%	1,95%	0,00%
C1_3_5	449	2,99%	6,61%	3,40%	428	1,81%	1,81%	0,00%	180	50	2,71%	2,71%	0,00%
C1_4_5	710	4,59%	13,80%	8,09%	698	2,04%	2,19%	0,15%	370	50	2,04%	2,19%	0,15%
C1_5_5	279	2,46%	9,76%	6,65%	499	0,00%	0,00%	0,00%	51	10	0,20%	1,41%	1,20%
C1_1_6	199	6,40%	16,37%	8,56%	293	2,83%	2,98%	0,15%	166	50	2,83%	2,98%	0,15%
C1_2_6	457	6,90%	18,93%	10,11%	557	2,65%	2,65%	0,00%	309	50	2,65%	2,65%	0,00%
C1_3_6	298	10,59%	29,54%	14,63%	514	3,68%	3,68%	0,00%	277	50	3,68%	3,68%	0,00%
C1_4_6	468	6,10%	16,08%	8,60%	571	2,58%	2,74%	0,16%	304	50	2,58%	2,74%	0,16%
C1_5_6	178	9,03%	26,26%	13,65%	427	3,10%	3,71%	0,59%	252	50	3,96%	4,58%	0,59%
C1_1_7	272	0,29%	1,77%	1,45%	267	0,10%	0,10%	0,00%	25	9	0,87%	0,87%	0,00%
C1_2_7	174	6,93%	25,82%	15,01%	429	0,20%	0,20%	0,00%	59	12	1,77%	1,77%	0,00%
C1_3_7	59	6,96%	22,12%	12,41%	224	3,07%	3,07%	0,00%	132	50	3,14%	3,14%	0,00%
C1_4_7	324	6,18%	17,67%	9,76%	445	3,42%	6,36%	2,77%	246	50	3,66%	6,61%	2,77%
C1_5_7	129	8,35%	25,26%	13,51%	341	0,00%	0,15%	0,15%	49	14	1,21%	1,75%	0,53%
C1_1_8	167	8,82%	9,82%	0,92%	245	7,90%	7,90%	0,00%	141	50	7,90%	7,90%	0,00%
C1_2_8	197	3,59%	11,04%	6,70%	373	0,48%	0,48%	0,00%	128	27	1,24%	1,88%	0,62%
C1_3_8	89	4,25%	9,02%	4,38%	186	2,29%	2,36%	0,07%	125	50	3,17%	3,45%	0,27%
C1_4_8	259	4,97%	14,35%	8,20%	388	1,82%	1,82%	0,00%	75	19	1,82%	1,82%	0,00%
C1_5_8	230	1,43%	3,95%	2,43%	350	0,50%	0,57%	0,07%	124	30	1,50%	1,57%	0,07%
C1_1_9	82	10,17%	21,31%	9,19%	152	4,99%	4,99%	0,00%	86	50	5,28%	5,28%	0,00%
C1_2_9	125	11,38%	29,66%	14,10%	329	0,40%	0,40%	0,00%	85	23	1,68%	1,68%	0,00%
C1_3_9	242	1,85%	4,49%	2,52%	279	0,00%	0,00%	0,00%	97	32	0,93%	1,10%	0,17%
C1_4_9	200	3,53%	12,19%	7,71%	284	1,67%	1,67%	0,00%	163	50	1,67%	5,19%	3,35%
C1_5_9	238	0,00%	2,30%	2,25%	378	0,00%	0,00%	0,00%	64	16	1,53%	1,53%	0,00%

5.3.3 Résultats de l'étape de planification

Dans le tableau 5.5 sont reportés les temps de calcul moyens de l'étape de planification dans laquelle l'insertion des clients optionnels consiste en un seul temps (celui avec les estimés pessimistes). On a limité les temps de calcul à 2 heures (soit 7200s). Si le problème n'a pas pu être résolu dans ce délai, on remplace le temps de calcul par un « - ». De nombreuses variantes de l'algorithme de branch and cut sont considérées. Les en-têtes de colonnes sont les suivants : # résolus : nombre d'instances résolues à optimalité en moins de 2 heures ; Temps moyen : moyenne des temps de calcul sur les instances résolues à optimalité ; B&C : branch and cut (avec une priorité de branchement sur les variables x) ; P : Prétraitement sur les variables ; H : Heuristique pour générer une solution initiale ; RC : Reachability cuts ; SEI : Inégalités d'élimination de sous-ensembles. Dans le tableau 5.5, on peut observer une

Tableau 5.5 Etape de planification : Temps de calcul moyens (en secondes)

# de clients	# d'obligatoires	# d'instances	B&C		B&C, P		B&C, P, H		B&C, P, H, RC		B&C, P, H, RC, SEI	
			# résolus	Temps moyen	# résolus	Temps moyen	# résolus	Temps moyen	# résolus	Temps moyen	# résolus	Temps moyen
30	5	5	5	1813	5	237	5	253	5	68	5	43
	6	5	4	2797	5	288	5	324	5	137	5	66
	7	5	2	1118	5	408	5	294	5	138	5	53
	8	5	4	1689	5	109	5	90	5	50	5	34
	9	5	5	1613	5	69	5	60	5	37	5	24
40	5	5	0	-	0	-	0	-	1	3869	3	3277
	6	5	0	-	3	1952	3	1720	4	1601	4	962
	7	5	0	-	1	519	1	493	3	3736	4	1451
	8	5	0	-	1	556	2	3428	2	1196	5	1880
	9	5	0	-	3	2250	3	1988	3	1201	5	1053

nette diminution des temps de calcul à l'aide des différentes techniques d'accélération. Nous pouvons résoudre toutes les instances avec 30 clients en un temps moyen d'une minute et la plupart des instances à 40 clients en moins de 2 heures lorsque toutes les techniques d'accélération sont mises en oeuvre.

Les résultats obtenus en comparant les méthodes de relaxation Lagrangienne et de branch and cut sont synthétisés dans le tableau 5.6. Les en-têtes de colonnes sont les suivants : # résolus : nombre d'instances résolues ; Gap moyen opt : gap moyen entre la meilleure borne supérieure $\overline{z}$ et la valeur optimale z^* (gap $= \dfrac{\overline{z} - z^*}{z^*}$) ; Gap moyen LB : gap moyen entre $\overline{z}$ et la meilleure borne inférieure $\underline{z}$ (gap $= \dfrac{\overline{z} - \underline{z}}{\underline{z}}$) ; Gap moyen LB - opt : gap moyen entre $\underline{z}$ et z^* (gap $= \dfrac{z^* - \underline{z}}{z^*}$). Pour la méthode basée sur la relaxation lagrangienne, toutes les instances

sont résolues dans le temps imparti. Toutefois, les instances ne sont pas résolues à l'optimalité, étant donné qu'il s'agit d'une heuristique. On n'indiquera pas pour cette méthode le nombre d'instances résolues à optimalité, étant donné que celui-ci est nul (sauf cas exceptionnel).

Tableau 5.6 Etape de planification : Branch and cut versus Relaxation Lagrangienne

# de clients	# d'obligatoires	# d'instances	Branch and cut			Relaxation lagrangienne			
			# résolus	Gap moyen LB	Temps moyen	Gap moyen opt.	Gap moyen LB	Gap moyen LB-opt.	Temps moyen
30	5	5	5	0.0%	43	1.75%	2.03%	0.27%	192
	6	5	5	0.0%	66	3.14%	3.32%	0.18%	262
	7	5	5	0.0%	53	2.13%	2.83%	0.66%	102
	8	5	5	0.0%	34	3.13%	3.32%	0.19%	119
	9	5	5	0.0%	24	2.22%	2.95%	0.70%	99
40	5	5	3	1.8%	3277	1.98%	2.64%	0.46%	1211
	6	5	4	2.0%	962	1.99%	3.09%	0.52%	1106
	7	5	4	0.8%	1451	1.60%	2.81%	0.27%	572
	8	5	5	0.0%	1880	2.09%	2.42%	0.32%	532
	9	5	5	0.0%	1053	1.35%	2.43%	1.04%	507
50	5	5	0	4.04%	7200	-	1.44%	-	1227
	6	5	1	4.99%	5960	0.33%	1.63%	0.00%	3064
	7	5	1	4.03%	6820	0.88%	2.57%	0.07%	2844
	8	5	0	7.50%	7200	-	2.40%	-	2323
	9	5	2	2.78%	5365	0.54%	2.13%	0.46%	1408

Dans le tableau 5.6, on peut observer que la méthode de relaxation lagrangienne est très efficace sur toutes tailles d'instances. Ainsi, sur les instances à 30 et 40 clients, la meilleure solution obtenue est à moins de 1% de la solution optimale. Et cette méthode nous permet de résoudre les instances avec 50 clients et 3 véhicules en moins de 3100 secondes (environ 50 minutes). Pour la suite des résultats, nous utiliserons donc la méthode exacte pour les instances à 30 et 40 clients et la méthode de relaxation lagrangienne pour celles à 50 clients.

Afin de tester notre méthode d'insertion des clients optionnels, nous avons procédé à des tests sur les instances du Team Orienteering Problem proposées par Chao *et al.* [16], et avons comparé nos résultats à ceux obtenus par Boussier *et al.* [11] avec une méthode de branch and price. Les résultats obtenus en comparant ces deux méthodes sont regroupés dans le tableau 5.7. Dans ce tableau, on peut constater, en moyenne, que l'algorithme de branch and price proposé est plus performant que notre algorithme de branch and cut. La raison en est la suivante : dans le problème de team orienteering, on considère un seul dépôt et tous les véhicules partent de ce dépôt pour y retourner. Cela génère une certaine symétrie au niveau du problème, qui n'apparaît pas dans notre problème multi-dépôt. Dans notre algorithme de branch and cut, afin de parer un minimum à cette symétrie, nous ajoutons des contraintes du type $\sum_{i \in O} x_i^p \leqslant \sum_{i \in O} x_i^{p+1}, \forall p \in P$. Mais celles-ci s'avèrent insuffisantes pour diminuer suffisamment la symétrie du problème. Toutefois, avec notre branch and cut, nous avons tout

Tableau 5.7 Comparaison du Branch and cut et du Branch and price sur les instances du Team Orienteering Problem

Type d'instance	Instance	Nombre d'instances	Branch and cut			Branch and price		
			Nb résolus	Valeur moyenne	CPU moyen	Nb résolus	Valeur moyenne	CPU moyen
p1	p1_2	18	18	140,0	13,2	15	116,0	127,5
	p1_3	18	18	111,1	75,0	18	111,1	2,1
	p1_4	18	18	84,2	120,2	18	84,2	0,1
p2	p2_2	11	11	190,5	0,4	11	190,5	0,1
	p2_3	11	11	136,4	0,2	11	136,4	0,1
	p2_4	11	11	94,5	0,0	11	94,5	0,0
p3	p3_2	20	20	496,0	16,1	12	357,5	283,4
	p3_3	20	20	411,5	263,2	18	375,0	98,5
	p3_4	20	18	305,6	476,8	20	336,5	0,9
p4	p4_2	20	3	332,7	1688,3	5	429,6	996,6
	p4_3	20	4	141,3	1263,8	9	422,7	643,6
	p4_4	20	6	90,8	330,2	11	344,3	65,8
p5	p5_2	26	8	162,5	674,8	11	275,5	470,2
	p5_3	26	10	155,0	771,5	16	369,1	161,7
	p5_4	26	10	100,0	288,7	23	476,2	103,7
p6	p6_2	14	6	190,0	194,0	9	385,3	157,1
	p6_3	14	9	194,7	914,7	13	404,8	683,0
	p6_4	14	10	36,6	162,8	14	255,0	0,6
p7	p7_2	20	7	217,1	1448,9	6	177,0	9,3
	p7_3	20	7	143,1	1153,9	9	213,3	379,6
	p7_4	20	7	94,1	474,6	12	240,2	71,0

de même pu résoudre quelques instances restées ouvertes avec le branch and price de Boussier *et al.* [11]. Les résultats obtenus sur ces instances ouvertes sont regroupés dans le tableau 5.8. On peut constater que, lorsque les problèmes restent de taille réduite, on peut résoudre toutes les instances même si le branch and price est souvent plus rapide.

Tableau 5.8 Instances ouvertes du Team Orienteering Problem résolues à l'optimalité

Type d'instance	Nb de clients	Instance	Valeur	CPU
p1	32	p1_2_p	245	23
		p1_2_q	265	15
		p1_2_r	275	50
p3	33	p3_2_l	590	24
		p3_2_m	620	31
		p3_2_n	660	34
		p3_2_o	690	15
		p3_2_p	720	15
		p3_2_q	760	25
		p3_2_r	790	33
		p3_2_s	800	21
		p3_3_s	720	836
		p3_3_t	760	1416
p7	102	p7_2_g	459	5812

5.3.4 Résultats de l'étape d'exécution

Dans cette section, nous présentons les résultats obtenus à la fin de la méthode (après la programmation dynamique et après simulations). Nous noterons WR la stratégie de programmation dynamique consistant à considérer toute la route et OS celle consistant à ne considérer qu'un seul segment. Dans les tableaux 5.9 et 5.10, nous présentons les résultats de l'étape d'exécution avec les deux stratégies de programmation dynamique. Ces résultats ont été obtenus en utilisant la meilleure solution obtenue lors de l'étape de planification (même si celle-ci n'était pas optimale) avec une insertion des clients optionnels en deux temps (estimés pessimistes puis optimistes). Nous n'avons pas reporté les temps de calcul associés à l'étape d'exécution car ils sont de quelques secondes.

Tableau 5.9 Valeurs moyennes après insertion des clients optionnels en 2 temps et simulation sur des instances à 30 clients

# clients obligatoires	Avant simulation		Après simulation					
	# moy. non servis	distance moyenne	# moy. non servis		distance moyenne		retard moyen	
			WR	OS	WR	OS	WR	OS
5	0	215	6.9	4.6	208	220	9.5	9.5
6	0	224	9.8	7.2	213	225	14.6	14.7
7	0	224	10.2	7.8	212	226	11.1	11.9
8	0	219	9.6	8.4	227	233	13.8	13.9
9	0	223	10.3	8.2	227	239	18.1	46.7

Tableau 5.10 Valeurs moyennes après insertion des clients optionnels en 2 temps et simulation sur des instances à 40 clients

# clients obligatoires	Avant simulation		Après simulation					
	# moy. non servis	distance moyenne	# moy. non servis		distance moyenne		retard moyen	
			WR	OS	WR	OS	WR	OS
5	4.8	213	11.1	11.1	176	177	6.9	6.9
6	1.6	249	13.2	12.9	188	189	12.2	12.0
7	1.4	246	17.1	13.6	162	176	16.4	13.1
8	0.0	247	16.0	14.7	168	172	16.9	15.6
9	0.0	248	14.2	13.5	179	189	15.4	14.7

Dans les tableaux 5.9 et 5.10, on observe que le nombre de clients devenus non desservis durant la simulation est très élevé (5 à 10 clients en moyenne sur les instances à 30 clients et 11 à 17 clients en moyenne sur les instances à 40 clients.). Ces solutions ne sont pas admissibles. Les résultats obtenus avec l'insertion des clients optionnels avec estimés pessimistes (à l'aide du branch and cut pour 30 et 40 clients, et à l'aide de la relaxation lagrangienne pour 50 clients) puis réparation et amélioration de la solution pour la variante $V2$, et $F = 0.9$, sur des instances contenant jusqu'à 50 clients sont synthétisés dans les tableaux 5.11, 5.12 et 5.13.

Tableau 5.11 Valeurs moyennes après insertion des clients optionnels avec estimés pessimistes et simulation sur des instances à 30 clients

# clients obligatoires	Avant simulation		Après simulation					
	# moy. non servis	distance moyenne	# moy. non servis		distance moyenne		retard moyen	
			WR	OS	WR	OS	WR	OS
5	1.4	220	5.3	3.4	199	211	2.0	2.0
6	2.8	232	5.6	5.6	212	212	3.1	3.2
7	2.4	230	5.5	5.5	209	209	2.7	2.8
8	3.4	216	7.6	6.1	195	204	3.2	3.2
9	3.0	217	7.0	6.4	197	201	4.6	4.6

Tableau 5.12 Valeurs moyennes après insertion des clients optionnels avec estimés pessimistes et simulation sur des instances à 40 clients

# clients obligatoires	Avant simulation		Après simulation					
	# moy. non servis	distance moyenne	# moy. non servis		distance moyenne		retard moyen	
			WR	OS	WR	OS	WR	OS
5	7.2	202	9.7	9.7	193	194	1.8	1.8
6	10.2	217	12.7	12.7	204	204	2.7	2.7
7	10.0	211	13.3	13.3	198	198	2.4	2.5
8	10.0	210	13.2	13.2	198	198	3.6	3.6
9	10.0	206	14.5	14.0	189	191	6.8	5.6

Dans les tableaux 5.11 à 5.13, on observe que le nombre de clients devenus non desservis durant les simulations est nettement plus raisonnable ici (3 à 4 clients en moyenne sur toutes les instances). En comparant les stratégies de programmation dynamique, on peut constater que la stratégie consistant à considérer un segment s'avère plus effective du point de vue du nombre de clients desservis. Quant au retard moyen, il garde des valeurs comparables, quelle que soit la stratégie. En ce qui concerne la distance moyenne, en calculant le ratio distance/nombre de clients visités, on peut comparer ces valeurs. Elles restent quasiment identiques d'une stratégie à l'autre.

Tableau 5.13 Valeurs moyennes après insertion des clients optionnels avec relaxation lagrangienne et simulation sur des instances à 50 clients

# clients obligatoires	Avant simulation		Après simulation					
	# moy. non servis	distance moyenne	# moy. non servis		distance moyenne		retard moyen	
			WR	OS	WR	OS	WR	OS
5	15.4	192	17.8	17.9	185	185	2.0	2.0
6	17.6	197	21.0	20.5	189	189	3.6	3.6
7	18	192	21.8	21.2	184	185	3.6	3.6
8	18.6	200	22.1	22.2	187	187	4.3	4.4
9	19.2	188	22.7	22.8	180	180	5.7	5.9

5.3.5 Représentation graphique des résultats obtenus

Intéressons-nous à présent à la représentation graphique des résultats obtenus après chaque étape de la méthode de résolution sur une instance contenant 40 clients et 3 véhicules : C1_5_8(40).

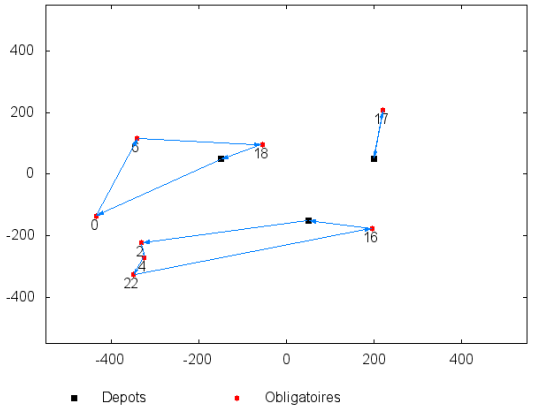


Figure 5.4 Après établissement du squelette

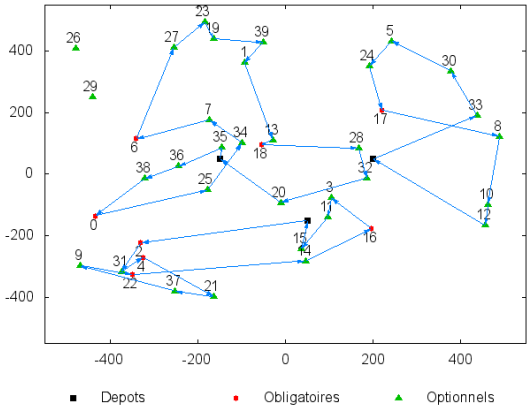


Figure 5.5 Après insertion des clients optionnels

Dans les graphiques 5.4 à 5.8, on peut observer les différentes étapes de notre méthode de résolution sur l'instance C1_5_8(40), contenant 8 clients obligatoires. Tout d'abord, on construit un squelette de routes en ne considérant que ces 8 clients (figure 5.4). Ensuite, figure 5.5, on insère les clients optionnels dans le squelette avec estimés pessimistes des temps de service et de parcours. On peut observer que la tournée en haut à gauche est bien remplie et ne peut desservir 2 clients optionnels. Après réparation de la solution figure 5.6, la tournée située en haut à gauche est clairement allégée et 9 clients de plus sur cette tournée deviennent non desservis. Ensuite, on améliore la solution en gardant une probabilité de faisabilité suffisante (figure 5.7). Enfin, on utilise l'algorithme de programmation dynamique

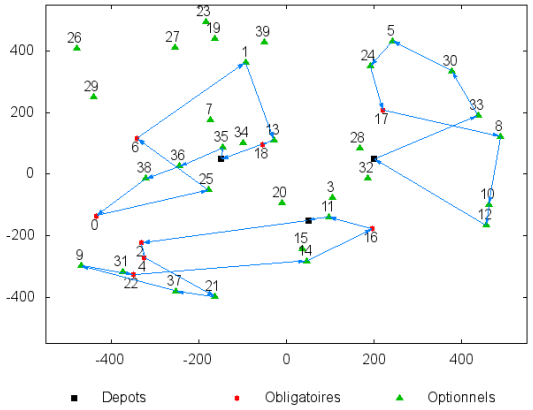


Figure 5.6 Après réparation de la solution

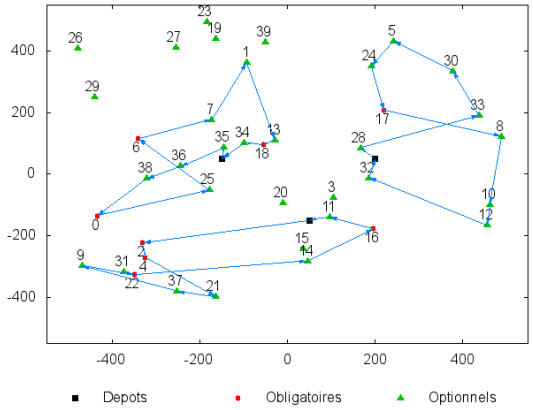


Figure 5.7 Après amélioration de la solution

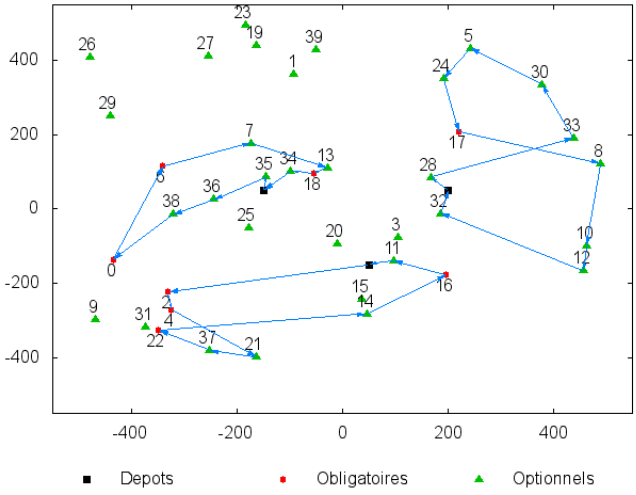


Figure 5.8 Après programmation dynamique

et on procède aux simulations. On obtient ainsi les routes réelles figure 5.8, qui ne sont pas très différentes des routes planifiées. Finalement, seuls 4 clients optionnels ont été annulés.

5.4 Conclusion

Dans ce chapitre, nous avons présenté une heuristique basée sur la priorité des clients pour l'étape de planification. Cette heuristique procède en deux phases : la construction de routes ne contenant que des clients obligatoires (ou construction du squelette) puis l'insertion des clients optionnels dans le squelette. Nous avons également proposé des méthodes d'accélération pour l'insertion des clients optionnels, qui nous ont permis de résoudre des instances

contenant jusqu'à 40 clients avec 3 véhicules en moins de 2 heures. En utilisant la relaxation lagrangienne et ces techniques d'accélération, nous avons également pu résoudre des instances comportant 50 clients et 3 véhicules en moins d'une heure. Dans les résultats obtenus après l'étape de planification, nous avons pu remarquer que la stratégie de programmation dynamique consistant à ne considérer qu'un seul segment s'avère plus efficace du point de vue du nombre de clients desservis tandis que les deux stratégies se valent quant à la distance totale parcourue et au retard (tableaux 5.9 à 5.13). On pourrait donc préférer la stratégie considérant un seul segment mais du point de vue opérationnel, il semble préférable de considérer le reste de la route et non segment par segment. Afin de prendre cette décision, il faudrait effectuer des tests sur un horizon de temps multi-période afin de juger de la pertinence de ces deux stratégies. La méthode que nous avons proposée permet donc de résoudre des instances de taille raisonnable en un temps acceptable (en effet, pour 3 véhicules, on ne peut espérer desservir plus de 50 clients en une journée). Toutefois, la méthode que nous avons proposée présente un inconvénient non négligeable : elle présuppose l'existence d'un nombre suffisant de clients obligatoires (sinon, l'établissement du squelette sur les clients obligatoires devient inutile). Or, d'un point de vue pratique, dans l'application à laquelle nous nous intéressons (où les clients obligatoires correspondent à des clients chez lesquels une opération de réparation est planifiée), on peut espérer ne pas avoir à traiter un trop grand nombre de pannes au quotidien. Une piste intéressante serait donc d'intégrer les deux phases de planification en une seule, i.e. à construire directement des routes contenant des clients optionnels et des clients obligatoires. C'est pourquoi, dans le chapitre suivant, nous allons présenter une heuristique, basée sur la génération de colonnes, capable de construire des routes avec les deux types de clients en une seule étape.

CHAPITRE 6

Heuristique basée sur la génération de colonnes

Dans le chapitre précédent, nous avons proposé une heuristique basée sur la priorité des clients pour l'étape de planification. Cette heuristique procède en deux temps : création de routes à partir des clients obligatoires puis insertion des clients optionnels dans ces routes. Toutefois, cette méthodologie présente l'inconvénient de nécessiter un nombre suffisant de clients obligatoires (afin de justifier de la construction du squelette des clients obligatoires). Dans ce chapitre, nous proposons une heuristique basée sur la génération de colonnes permettant de construire directement des routes contenant simultanément des clients obligatoires et des clients optionnels. Cette méthode vise à générer un ensemble de routes diversifiées et de bonne qualité pour chacun des véhicules, puis, à sélectionner une route pour chaque véhicule en résolvant un programme linéaire en nombres entiers. Dans les sections suivantes, nous détaillerons en premier lieu la méthode utilisée pour générer des routes puis nous expliquerons la sélection exacte des routes pour chaque véhicule. Ensuite, nous présenterons les résultats expérimentaux et enfin nous conclurons.

6.1 Génération de routes

Lors de la génération de routes, on considère les véhicules un par un. Soient N l'ensemble des clients, M l'ensemble des clients obligatoires, O l'ensemble des clients optionnels et P_{ij} l'ensemble de chemins de $i \in M$ à $j \in M$ (desservant des clients optionnels). Soient o^k et d^k l'origine et la destination du véhicule considéré, on note $M^o = M \cup \{o^k\}$ et $M^d = M \cup \{d^k\}$. Afin de conserver la notion de priorité entre les clients obligatoires et les clients optionnels, on attribue aux clients obligatoires un profit p_i supérieur à celui des clients optionnels. Il s'agit alors de générer des routes maximisant le profit total associé aux visites des clients, tout en minimisant la distance totale parcourue et en respectant les contraintes de fenêtres de temps associées à la destination du véhicule et aux clients obligatoires. Notons $\hat{\tau}_{lm}$ et $\overline{\tau}_{lm}$ les temps de parcours modaux et maximaux de l à m, $\hat{\sigma}_i$ et $\overline{\sigma}_i$ les temps de service modaux et maximaux au client i, $\hat{\tau}_p$ et $\hat{\sigma}_p$ les temps modaux de parcours et de service associés au chemin p, $[e_i, l_i]$ la fenêtre de temps du client i, T l'horizon de temps et α la pondération des temps de parcours dans la fonction objectif. On utilise les variables décisionnelles :

x_l variable booléenne indiquant si le client l est servi

y_{ij} variable booléenne indiquant si le client obligatoire j suit le client obligatoire i (quand on ne considère que les clients obligatoires de la route)

z_{lm} variable booléenne indiquant si l'arc (l, m) est utilisé $(l, m \in N)$

t_i heure de début de service modale chez le client obligatoire i

$\bar{t}_i$ heure de début de service au plus tard chez le client obligatoire i (quand on ne considère que les clients obligatoires de la route)

Avec les variables définies ci-dessus, on peut formuler le problème de génération de la meilleure route pour le véhicule k comme suit :

$$\max . \sum_{i \in N} p_i x_i - \alpha \sum_{l \in N \cup \{o^k\}} \sum_{m \in N \cup \{d^k\}} \hat{\tau}_{lm} z_{lm}$$

sujet à :

$$\sum_{m \in N \cup \{d^k\}} z_{o^k m} = 1 \tag{6.1}$$

$$\sum_{l \in N \cup \{o^k\}} z_{lm} = x_m \qquad \forall m \in N \tag{6.2}$$

$$\sum_{m \in N \cup \{d^k\}} z_{lm} = x_l \qquad \forall l \in N \tag{6.3}$$

$$\sum_{(l,m) \in p} z_{lm} \leqslant |p| - 1 + y_{ij} \quad \forall i \in M^o, j \in M^d, p \in P_{ij} \tag{6.4}$$

$$t_i + (\hat{\tau}_p + \hat{\sigma}_p) \left(\sum_{(l,m) \in p} z_{lm} - |p| + 1 \right) + (y_{ij} - 1)T \leqslant t_j \qquad \forall i \in M^o, j \in M^d, p \in P_{ij} \tag{6.5}$$

$$\bar{t}_i + \overline{\sigma}_i + \overline{\tau}_{ij} + (y_{ij} - 1)T \leqslant \bar{t}_j \qquad \forall i \in M^o, j \in M^d \tag{6.6}$$

$$t_i + \hat{\sigma}_i + \hat{\tau}_{ij} + (y_{ij} - 1)T \leqslant t_j \qquad \forall i \in M^o, j \in M^d \tag{6.7}$$

$$e_i \leqslant \bar{t}_i \leqslant l_i \qquad \forall i \in M \cup \{o^k; d^k\} \quad (6.8)$$

$$e_i \leqslant t_i \leqslant l_i \qquad \forall i \in M \cup \{o^k; d^k\} \quad (6.9)$$

$$x_l, z_{lm} \in \{0; 1\} \qquad \forall l, m \in N \quad (6.10)$$

$$y_{ij} \in \{0; 1\} \qquad \forall i \in M^o, j \in M^d \quad (6.11)$$

$$\bar{t}_i, t_i \geqslant 0 \qquad \forall i \in M \cup \{o^k; d^k\} \quad (6.12)$$

La contrainte (6.1) assure que la route du véhicule quitte l'origine (éventuellement pour aller directement au dépôt destination). Les contraintes (6.2) et (6.3) sont des contraintes de degrés entrant et sortant. Les contraintes (6.4) spécifient que si on emprunte le chemin p allant de i à j, alors le client obligatoire j suit le client obligatoire i (quand on ne considère que les clients obligatoires). Les contraintes (6.5), quant à elles, indiquent que, si on utilise un chemin p pour aller de i à j, alors, on doit s'assurer de commencer le service au client obligatoire j après avoir quitté le client obligatoire i et après avoir desservi tous les clients optionnels du chemin p. Les contraintes (6.6) et (6.7) sont des contraintes de précédence temporelle entre les clients obligatoires (avec des estimés modaux et maximaux). Enfin, les contraintes (6.8) et (6.9) assurent le respect des fenêtres de temps des clients obligatoires.

Le problème, pour un véhicule donné, peut donc être assimilé à un « Orienteering Problem with Time Windows » (OPTW) dans lequel le temps de parcours total est minimisé et auquel on ajoute les contraintes (6.4) à (6.6) et (6.8). Comme l'ont montré Bramel et Simchi-Levi [12], ce problème peut être reformulé comme un « Elementary Shortest Path Problem with Resource Constraints » (ESPPRC). Comme nous souhaitons générer un ensemble de routes diversifiées de très bonne qualité, l'idéal est d'utiliser une méthode exacte de type programmation dynamique. On peut alors, à la fin de l'algorithme de programmation dynamique sélectionner les meilleures routes (selon un critère donné). Dans ce qui suit, nous présenterons un bref état de l'art sur les méthodes de résolution proposées pour l'ESPPRC et nous expliquerons la méthode retenue. Nous détaillerons ensuite l'algorithme de Righini et Salani [58]. Puis, nous présenterons sa version améliorée telle qu'implantée par Salani (communications privées) et enfin nous proposerons différentes variantes de cet algorithme, adaptées à la résolution de notre problème.

6.1.1 Etat de l'art des méthodes de résolution de l'ESPPRC

Pour résoudre l'ESPPRC, différentes méthodes ont été proposées dans la littérature dans le cadre du problème de tournées de véhicules avec fenêtres de temps. On distinguera trois catégories : les algorithmes de programmation dynamique exacte, les algorithmes de pro-

grammation dynamique exacte avec relaxation des contraintes d'élémentarité et les méthodes heuristiques de résolution de sous-problèmes. Le premier algorithme de programmation dynamique exacte pour résoudre un ESPPRC est proposé par Feillet *et al.* [26]. Dans cette méthode, ils étendent l'algorithme d'étiquetage de Desrochers [23] proposé pour résoudre le SPPRC (plus court chemin avec contraintes de ressources) au cas élémentaire. Righini et Salani [57] proposent une variante bidirectionnelle bornée de l'algorithme de Feillet, afin d'accélérer la résolution du problème (cf. description détaillée dans le chapitre précédent).
La contrainte sur l'élémentarité d'un chemin étant à l'origine de nombreux états non dominés, plusieurs méthodes ont été proposées, basées sur la relaxation des contraintes d'élémentarité. Ainsi, Boland *et al.* [8] et Righini et Salani [58] proposent une méthode appelée « decremental state space relaxation » (DSSR). Dans cette méthode, ils relâchent entièrement les contraintes d'élémentarité lors de la génération des chemins (se ramenant ainsi à un problème de plus court chemin avec contraintes de ressources) et ajoutent itérativement des contraintes empêchant la formation de cycles jusqu'à obtention de chemins élémentaires. De même, Baldacci *et al.* [5] proposent une méthode appelée ng-route relaxation. Dans cette méthode, ils introduisent une notion d'élémentarité partielle d'un chemin en interdisant l'extension d'un chemin à un sous-ensemble des clients visités par ce chemin. Ce sous-ensemble est défini à partir d'une notion de voisinage. Afin de gagner encore en efficacité, Pinto [55] intègre la DSSR dans la ng-route relaxation en augmentant itérativement la taille des voisinages de la ng-route relaxation jusqu'à obtenir des routes ng-réalisables.
Les méthodes proposées ci-dessus sont des algorithmes exacts de programmation dynamique. Ces méthodes pouvant s'avérer très chronophages, Desaulniers *et al.* [22] proposent deux stratégies heuristiques de programmation dynamique afin d'accélérer la résolution de l'ESPPRC. La première stratégie est une « limited discrepancy search » (LDS) : étant donné un paramètre D_{max}, elle consiste à ne garder pour chaque client que ses D_{max} successeurs les plus proches et ses D_{max} prédécesseurs les plus proches (la distance correspondant aux coûts réduits associés aux arcs). La deuxième stratégie est une stratégie de dominance aggrégée : étant donné un paramètre R_{max}, elle consiste à n'utiliser dans les tests de dominance que R_{max} ressources.

6.1.2 Méthode de résolution de l'ESPPRC retenue

Dans la littérature (cf. ci-dessus), plusieurs techniques consistant à relâcher les contraintes d'élémentarité d'un chemin ont été proposées pour accélérer la programmation dynamique : la DSSR [58] et les ng-routes [5]. La DSSR (decremental state space relaxation) consiste à relaxer la contrainte d'élémentarité d'un chemin lors de son extension (et donc à résoudre un

SPPRC au lieu d'un ESPPRC) puis à forcer l'élémentarité des arcs de la solution optimale tant que celle-ci contient des cycles. Dans leur article en 2009, Righini et Salani [58] testent cette méthode sur un ensemble d'instances dont celles de Cordeau *et al.* [17] et y indiquent qu'en deux heures, ils obtiennent des solutions à plus de 50% de gap de l'optimal sur les instances avec fenêtres de temps larges (pr11 à pr20). Or, dans ces instances, les fenêtres de temps ne sont pas aussi larges que dans notre problème. Nous n'avons donc pas conservé cette méthodologie pour nos tests.
La méthode des ng-routes consiste aussi à relaxer la contrainte d'élémentarité d'un chemin, sans pour autant réduire le problème à un SPPRC. Il s'agit, pour un chemin p, de s'interdire non pas tous les clients déjà visités mais un sous-ensemble de ces clients. Soit N_i l'ensemble des Δ_{NG} plus proches voisins du client i (i inclus), et $p = (i_1, i_2, .., i_k)$ un chemin, on définit l'ensemble des clients interdits $\Pi(p) = \{i_r : i_r \in \cap_{s=r+1}^{k} N_{i_s}, r = 1, ..., k-1\} \cup \{i_k\}$. Nous avons testé cette méthodologie, combinée à l'élimination des 2-cycles. Toutefois, lors des tests, quelle que soit la valeur du paramètre Δ_{NG}, nous avons obtenu des routes non élémentaires. Nous n'avons donc pas retenu cette méthode pour nos expérimentations.
Afin de procéder à une génération d'un ensemble de routes de très bonne qualité, nous avons choisi de nous baser sur l'algorithme de programmation dynamique bi-directionnelle bornée proposé par Righini et Salani [58].

6.1.3 Algorithme de programmation dynamique bidirectionnelle bornée de Salani

Dans l'algorithme de programmation dynamique bidirectionnelle bornée proposé par Righini et Salani [58], on considère deux directions : une direction « forward » (depuis l'origine vers la destination) et une direction « backward » (depuis la destination vers l'origine). On considère également une ressource critique. La programmation dynamique bidirectionnelle bornée consiste alors à construire des chemins dans chacune des deux directions à l'aide de la programmation dynamique classique puis à procéder à la jonction des chemins forward et backward. De plus, lors de la construction des chemins, on doit s'assurer, dans chacune des directions, que la ressource critique utilisée par un chemin n'excède pas la moitié de la quantité disponible de cette ressource. Une description en pseudo-code de cet algorithme est présentée dans l'algorithme 3.

Pour un véhicule donné, soient s l'origine et t la destination de ce véhicule. Avant de démarrer la programmation dynamique, Righini et Salani [58] procèdent à un tri des clients par ordre d'extension avec la fonction $triClientsOrdreExtension()$. Ils trient les clients par ordre croissant de début de fenêtre de temps, et pour un même début de fenêtre de temps, par ordre décroissant de fin de fenêtre de temps. Ensuite, ils associent à chaque chemin un

```
L ← triClientsOrdreExtension();
// Extension forward
Γ_fw ← [∅, ∅, ...., ∅];
Γ_fw[s] ← {(0, 0, 0, 0, s)};
tant que fini ≠ vrai faire
    fini ← vrai;
    pour i = 0 à taille(L) faire
        pour chaque li ∈ Γ_fw[L[i]] faire
            pour chaque j ∈ L tel que extensionFwdPossible(li, j) faire
                si extendFwd(li, j) alors
                    fini ← faux;
                fin
            fin
        fin
    fin
fin
// Extension backward
Γ_bw ← [∅, ∅, ...., ∅];
Γ_bw[t] ← {(0, 0, 0, 0, t)};
tant que fini ≠ vrai faire
    fini ← vrai;
    pour i = taille(L) à 0 faire
        pour chaque li ∈ Γ_bw[L[i]] faire
            pour chaque j ∈ L tel que extensionBwdPossible(li, j) faire
                si extendBwd(li, j) alors
                    fini ← faux;
                fin
            fin
        fin
    fin
fin
// Jonction des labels
pour chaque li ∈ Γ_fw faire
    pour chaque lj ∈ Γ_bw faire
        joindre(li, lj);
    fin
fin
```

Algorithm 3: Programmation dynamique bornée bidirectionnelle (Righini et Salani [58])

label permettant de caractériser ce chemin (cf. description plus loin) et créent des matrices pour contenir les labels forward et backward, respectivement Γ_{fw} et Γ_{bw} ($\Gamma_{fw}[i]$ contient tous les labels forward finissant au client i). Ils initialisent un label forward pour le client s et un label backward pour le client t. Une fois l'initialisation terminée, ils procèdent dans un premier temps à l'extension forward. Puis, ils considèrent les clients un à un par ordre d'extension et essayent d'étendre forward tous les labels associés à ce client. L'extension forward s'arrête quand tous les labels ajoutés sont dominés. Pour l'extension forward, ils disposent de deux fonctions : $extensionFwdPossible(li, j)$ qui indique si l'extension forward du label li au client j est possible et une fonction $extendFwd(li, j)$ qui étend le label li au client j en forward et ajoute le label ainsi obtenu à la liste $\Gamma_{fw}[j]$ si ce label n'est pas dominé. $extendFwd(li, j)$ retourne vrai si la liste des labels forward a été modifiée. Pour l'extension backward, ils considèrent cette fois les clients par ordre inverse d'extension et procèdent comme ci-dessus mais en extension backward. Les fonctions utilisées sont identiques à celles ci-dessus : $extensionBwdPossible(li, j)$ indique si l'extension backward du label li au client j est possible, et $extendBwd(li, j)$ étend le label li au client j en backward, ajoute le label ainsi obtenu à la liste $\Gamma_{bw}[j]$ si ce label n'est pas dominé et retourne vrai si la liste des labels backward a été modifiée. Enfin, une fois les extensions forward et backward terminées, ils procèdent à la jonction des labels forward et backward à l'aide de la fonction $joindre(li, lj)$ qui joint le label forward li au label backward lj si cette jonction est possible. Cet algorithme fournit donc un ensemble de routes obtenues par jonction de chemins forward et de chemins backward.

6.1.4 Algorithme de programmation dynamique implanté par Salani

Depuis son article en 2009, Salani a amélioré son code et a proposé une nouvelle variante de son algorithme de programmation dynamique (communications privées avec Salani) pour résoudre spécifiquement un OPTW. Dans cette variante, il construit des chemins forward et des chemins backward en limitant la durée d'un chemin dans une direction à la moitié de la durée totale autorisée. Puis, il procède à la jonction des chemins forward et backward. Le fonctionnement de cette méthode est identique à celui de Righini et Salani [58]. La seule différence réside au niveau des labels (cf. plus loin). Pour cet algorithme, Salani associe à chaque chemin p un label (S, t, r, u, i) où S est le vecteur de visite des clients ($S[k] = 1$ si le client k est visité par le chemin p, $S[k] = 2$ si le client k est inaccessible pour le chemin p et $S[k] = 0$ sinon), t le temps consommé sur le chemin p (en utilisant les temps de service et les distances comme temps de parcours), r le profit associé à la visite des clients, u le nombre de clients inaccessibles et i désigne le dernier client visité par le chemin. On remarquera que, pour un label forward, le temps consommé correspond au temps qui s'est écoulé entre

le départ de l'origine et le début de service au client courant. Pour un label backward, le temps consommé représente plutôt le temps écoulé entre la fin de service au client courant et l'arrivée à destination. Pour ces labels, il définit des règles d'extension, de dominance ainsi que de jonction décrites ci-après. En notant N l'ensemble de clients, d le dépôt destination, σ_i la durée de service au client i, τ_{ij} le temps de parcours de i à j, p_i le profit associé à la visite du client i et $[e_i, l_i]$ la fenêtre de temps du client i, il définit la durée totale autorisée T comme étant l'heure maximale d'arrivée au dépôt, soit $T = max_i(l_i + \sigma_i + \tau_{id})$.

Dans la première version de Righini et Salani [58] décrite au paragraphe 6.1.3, ils utilisaient les labels (S, t, r, i). Ils ne prenaient pas en compte le nombre de clients inaccessibles u. Cette valeur aide à augmenter le nombre de labels dominés et accélère l'algorithme.

Règles d'extension

Etendre un label revient à ajouter un client au chemin qui lui est associé. L'extension du label (S, t, r, u, i) au client j génère le label (S', t', r', u', j). En posant $e = e_j$ si on est en forward et $e = T - l_j - \sigma_j$ si on est en backward, on a :

$$\begin{cases} S' = S \text{ et } S'[j] = 1 \\ t' = \max(e, t + \sigma_i + \tau_{ij}) \\ r' = r + p_j \\ u' = u + 1 + |U'_j| \text{ où } U'_j = \{k \in N | S[k] = 0 \text{ et } t' + \sigma_j + \tau_{jk} > l_k\} \text{ en forward} \\ \qquad\qquad\qquad \text{et } U'_j = \{k \in N | S[k] = 0 \text{ et } t' + \sigma_j + \tau_{jk} > T - e_k - \sigma_k\} \text{ en backward} \end{cases}$$

Le nouveau label visite les mêmes clients que l'ancien label, plus le client j. En forward, le temps consommé par le nouveau label correspond soit à l'heure d'arrivée au plus tôt au client j, soit au temps consommé par l'ancien label auquel on ajoute le temps consommé pour desservir i et celui consommé pour se rendre au client j. En backward, le temps consommé par le nouveau label correspond soit au temps consommé entre la fin de service au plus tard au client j et l'arrivée à destination (à $t = T$), soit au temps consommé par l'ancien label auquel on ajoute le temps consommé pour servir i et celui consommé pour aller de j à i (temps de parcours symétriques). Le revenu associé au nouveau label correspond au revenu de l'ancien label auquel on ajoute le profit associé au client j. Enfin, le nombre de clients inaccessibles pour le nouveau label correspond au nombre de clients inaccessibles associés à l'ancien label auquel on ajoute le nombre de clients devenus inaccessibles avec la visite du client j.

En posant $l = l_j$ en forward et $l = T - e_j - \sigma_j$ en backward, l'extension sera réalisable si les conditions suivantes sont vérifiées :

$$\begin{cases} S[j] = 0 \\ t' \leqslant l \\ t' \leqslant T/2 \quad \text{si c'est un label backward} \\ t \leqslant T/2 \quad \text{si c'est un label forward} \end{cases}$$

Ici, on s'assure de ne pas visiter deux fois le même client, de respecter la fenêtre de temps au client j. On s'assure également qu'un label backward respecte la contrainte de durée. Par contre, on autorise un label forward à déborder d'un client sur cette même contrainte. En d'autres termes, si la durée d'un chemin forward ne dépasse pas $T/2$, on s'autorise l'extension du label forward associé (même si le label forward ainsi généré ne respecte pas la contrainte de durée).

Règles de dominance

Pour limiter le nombre de labels considérés à chaque étape de la programmation dynamique, on utilise des règles de dominance. Le label (S_1, t_1, r_1, u_1, i) domine le label (S_2, t_2, r_2, u_2, i) si :

$$\begin{cases} S_1 \leqslant S_2 \\ t_1 \leqslant t_2 \\ r_1 \geqslant r_2 \\ u_1 \leqslant u_2 \\ \text{une de ces inégalités est stricte} \end{cases}$$

Les labels dominés ne seront pas conservés dans la programmation dynamique.

Règles de jonction

La jonction d'un label forward $(S^{fw}, t^{fw}, r^{fw}, u^{fw}, i)$ et d'un label backward $(S^{bw}, t^{bw}, r^{bw}, u^{bw}, i)$ n'est possible que si les conditions suivantes sont vérifiées :

$$\begin{cases} S^{fw}[i] + S^{bw}[i] \leqslant 1 & \forall i \in N \\ t^{fw} + \sigma_i + \tau_{ij} + t^{bw} \leqslant T \end{cases}$$

6.1.5 Variantes proposées

Dans le cadre du problème que nous traitons, nous devons apporter quelques modifications à l'algorithme de programmation dynamique bidirectionnel implanté par Salani décrit ci-dessus (cf. section 6.1.4). On doit notamment gérer une ressource supplémentaire liée aux

clients obligatoires et à la stochasticité des temps de service et de parcours : la ressource $\bar{t}$ indiquant la durée du label dans le pire cas (quand on ne considère que les clients obligatoires). De plus, dans notre problème, comme les temps de service et de parcours sont stochastiques, on utilisera leur valeur modale pour la ressource t. Enfin, étant donné que nos temps de parcours sont différents de nos distances, on ajoute une ressource de distance d dans nos labels. Comme nos clients obligatoires sont les seuls à avoir des fenêtres de temps serrées, la distance est nécessaire au niveau des critères de dominance. En effet, sans ce critère de distance, on aurait du mal à éliminer des chemins dominés au niveau des clients obligatoires. On associe ainsi à chaque chemin p un nouveau label $(S, t, \bar{t}, r, d, u, i)$ où S est le vecteur de visite des clients (défini comme précédemment), t le temps consommé sur le chemin p (en utilisant les valeurs modales des temps de service et de parcours), $\bar{t}$ le temps consommé par les clients obligatoires sur le chemin p (en utilisant les temps de service et de parcours maximaux), r le profit associé à la visite des clients, d la distance parcourue, u le nombre de clients inaccessibles et i désigne le dernier client visité par le chemin. Pour ces labels, nous définissons des règles d'extension, de dominance ainsi que de jonction ci-après. Puis, nous présentons la méthodologie.

Règles d'extension

Etendre un label revient à ajouter un client au chemin qui lui est associé. L'extension du label $(S, t, \bar{t}, r, d, u, i)$ au client j génère le label $(S', t', \bar{t}', r', d', u', j)$. En notant D_{ij} la distance de i à j et en posant $e = e_j$ si on est en forward et $e = T - l_j - \hat{\sigma}_j$ si on est en backward, on a :

$$\begin{cases} S' = S \text{ et } S'[j] = 1 \\ t' = \max(e, t + \hat{\sigma}_i + \hat{\tau}_{ij}) \\ r' = r + p_j \\ d' = d + D_{ij} \\ u' = u + 1 + |U'_j| \text{ où } U'_j = \{k \in N | S[k] = 0 \text{ et } t' + \sigma_j + \tau_{jk} > l_k\} \text{ en forward} \\ \qquad\qquad\qquad\qquad \text{et } U'_j = \{k \in N | S[k] = 0 \text{ et } t' + \sigma_j + \tau_{jk} > T - e_k - \sigma_k\} \text{ en backward} \end{cases}$$

De plus, suivant le caractère obligatoire (ou optionnel) du client j, en notant m le dernier client obligatoire du label avant extension, on posera :

$$\bar{t}' = \begin{cases} \bar{t} & \text{si } j \text{ est optionnel} \\ \bar{t} + \bar{\sigma}_m + \bar{\tau}_{mj} & \text{sinon} \end{cases}$$

Ces règles d'extension sont similaires à celles présentées section 6.1.4. On modifie la règle d'extension pour la ressource temporelle t en prenant en compte les valeurs modales des

temps de service et de parcours. Aussi, on ajoute deux règles associées aux deux nouvelles ressources : la distance et les temps consommés par les clients obligatoires. La distance associée au nouveau label correspond à la distance de l'ancien label à laquelle on ajoute la distance pour aller de i à j. Le temps consommé par les clients obligatoires associé au nouveau label dépend de la nature du client j. Si le client j est optionnel, le temps consommé par les clients obligatoires reste inchangé dans le nouveau label. Si, au contraire, le client j est obligatoire, le temps consommé par les clients obligatoires est incrémenté du temps de service maximal au dernier client obligatoire m de l'ancien label et du temps de parcours maximal de m à j.

En posant $l = l_j$ en forward et $l = T - e_j - \hat{\sigma}_j$ en backward, l'extension sera réalisable si les conditions suivantes sont vérifiées :

$$\begin{cases} S[j] = 0 \\ t' \leqslant l \\ t' \leqslant T/2 & \text{si c'est un label backward} \\ t \leqslant T/2 & \text{si c'est un label forward} \end{cases}$$

Si le client j est un client obligatoire, les deux conditions suivantes doivent également être vérifiées :

$$\begin{cases} \bar{t}' \leqslant T/2 \\ \bar{t}' \leqslant l \end{cases}$$

Ici, on s'assure de ne pas visiter deux fois le même client, de respecter la fenêtre de temps au client j. On s'assure également du respect de la contrainte de durée pour un label backward tandis qu'on autorise un label forward à déborder d'un client sur cette même contrainte. De plus, si le client j est obligatoire, on doit s'assurer que le chemin ne contenant que les clients obligatoires est réalisable dans le pire des cas (avec les estimés maximaux). On doit donc s'assurer du respect de la borne sur la longueur d'un chemin et de la fenêtre de temps au client j avec les estimés pessimistes.

Règles de dominance

Pour limiter le nombre de labels considérés à chaque étape de la programmation dynamique, on utilise des règles de dominance. Le label $(S_1, t_1, \bar{t}_1, r_1, d_1, u_1, i)$ domine le label $(S_2, t_2, \bar{t}_2, r_2, d_2, u_2, i)$ si :

$$\begin{cases} S_1 \leqslant S_2 \\ t_1 \leqslant t_2 \\ \bar{t}_1 \leqslant \bar{t}_2 \\ r_1 \geqslant r_2 \\ d_1 \leqslant d_2 \\ u_1 \leqslant u_2 \\ \text{une de ces inégalités est stricte} \end{cases}$$

On retrouve ici les mêmes règles de dominance que dans la section 6.1.4. On ajoute juste deux règles de dominance au niveau de la nouvelle ressource de distance et au niveau de la ressource de temps consommé par les clients obligatoires. Les labels dominés ne seront pas conservés dans la programmation dynamique.

Règles de jonction

La jonction d'un label forward $(S^{fw}, t^{fw}, \bar{t}^{fw}, r^{fw}, d^{fw}, u^{fw}, i)$ et d'un label backward $(S^{bw}, t^{bw}, \bar{t}^{bw}, r^{bw}, d^{bw}, u^{bw}, i)$ n'est possible que si les conditions suivantes sont vérifiées :

$$\begin{cases} S^{fw}[i] + S^{bw}[i] \leqslant 1 & \forall i \in N \\ t^{fw} + \hat{\sigma}_i + \hat{\tau}_{ij} + t^{bw} \leqslant T \\ \bar{t}^{fw} + \bar{\sigma}_{m^{fw}} + \bar{\tau}_{m^{fw}m^{bw}} + \bar{t}^{bw} \leqslant T \end{cases}$$

où m^{fw} et m^{bw} représentent respectivement le dernier client obligatoire du label forward et du label backward.

Ci-dessus, nous avons présenté notre adaptation de l'algorithme bidirectionnel borné implanté par Salani dans le cadre de notre problème. On notera cet algorithme (BD, DA, VV) pour indiquer qu'il s'agit de l'algorithme bidirectionnel (BD), avec débordement autorisé d'un label forward sur la borne (DA) dans lequel S désigne le vecteur de visite (VV). Nous avons également proposé 3 variantes de cet algorithme. Dans la première variante (BD, DI, VV), on interdit le débordement d'un label forward sur la contrainte de durée. Ainsi, dans cette variante (BD, DI, VV), quel que soit le label $(S, t, \bar{t}, r, d, u, i)$, forward ou backward, on a $t \leqslant T/2$. Dans la deuxième variante (BD, DI, NV), on interdit également ce débordement d'un label forward et on modifie les labels en ajoutant une ressource s indiquant le nombre de clients visités. On modifie ensuite les règles de dominance en remplaçant la règle $S_1 \leqslant S_2$ par la règle $s_1 \leqslant s_2$. Cela ne remet pas en question l'élémentarité des chemins car on stocke tout de même dans chaque label le chemin associé. Par contre, cela modifie les règles de dominance car on pourra comparer deux labels ne visitant pas du tout les mêmes clients. Ainsi, on augmente fortement le nombre de labels dominés mais on perd également l'optimalité de

la solution. Dans la dernière variante (UD, DI, NV), on interdit le débordement d'un label forward, on utilise le label modifié avec la ressource s indiquant le nombre de clients visités et on procède à l'extension dans une seule direction (algorithme de programmation dynamique classique, noté UD).
Parmi ces différentes variantes, on en distingue de deux types : les variantes exactes (BD, DA, VV) et (BD, DI, VV) et les variantes approchées (BD, DI, NV) et (UD, DI, NV). Au vu des résultats obtenus avec les variantes approchées, nous avons choisi de procéder à une postoptimisation de type 2-Opt (échange d'arcs intra-route). Cette méthode permet d'améliorer la qualité des solutions obtenues du point de vue de la distance parcourue et donc des temps de parcours. Les performances de ces différentes variantes seront comparées dans la section résultats de ce chapitre.

6.1.6 Sélection des routes

Après avoir généré des routes, nous disposons pour chaque véhicule d'un ensemble de routes diversifiées et de bonne qualité. Pour chaque véhicule, cet ensemble de routes est obtenu par jonction de chaque chemin forward avec chaque chemin backward. Il peut donc contenir plusieurs exemplaires d'une même route comme nos clients ne sont pas très contraints (par exemple, la route (o ;1 ;2 ;3 ;4 ;d) peut être obtenue en joignant le chemin forward (o ;1 ;2) et le chemin backward (3 ;4 ;d) ou en joignant le chemin forward (o ;1 ;2 ;3) avec le chemin backward (4 ;d)). On ne procède pas à l'élimination de ces doublons car cette opération est coûteuse en temps de calcul tandis que Cplex élimine très rapidement les doublons lors de la sélection des routes. A partir de ces ensembles de routes, il nous faut à présent procéder à la sélection d'une route par véhicule, en s'assurant de desservir exactement une fois chaque client obligatoire, au plus une fois chaque client optionnel. L'objectif est de maximiser le profit total associé à la desserte des clients optionnels tout en minimisant le temps de parcours total. Notons Ω^k l'ensemble des routes générées pour le véhicule k. Pour une route r, on note p_r le profit associé, $\hat{\tau}_r$ le temps de parcours modal sur cette route et δ_{ir} le booléen indiquant si le client i est desservi sur la route r. En introduisant les variables booléennes x_r^k indiquant si la route r est sélectionnée pour le véhicule k, le problème de sélection des routes peut s'écrire :

$$\max \sum_{k \in K} \sum_{r \in \Omega^k} (p_r - \alpha \hat{\tau}_r) x_r^k$$

sujet à :

$$\sum_{k\in K}\sum_{r\in\Omega^k}\delta_{ir}x_r^k = 1 \qquad \forall i \in M \quad (6.13)$$

$$\sum_{k\in K}\sum_{r\in\Omega^k}\delta_{ir}x_r^k \leqslant 1 \qquad \forall c \in O \quad (6.14)$$

$$\sum_{r\in\Omega^k} x_r^k = 1 \qquad \forall k \in K \quad (6.15)$$

$$x_r^k \in \{0;1\}$$

Les contraintes (6.13) forcent chaque client obligatoire à être desservi une et une seule fois, les contraintes (6.14) indiquent que chaque client optionnel est desservi au plus une fois et les contraintes (6.15) forcent la desserte d'une et une seule route par véhicule. Nous choisissons de résoudre ce programme linéaire de façon exacte à l'aide d'un solveur commercial.

6.2 Expérimentation

Nous procédons aux expérimentations sur des instances de la littérature ainsi que sur les instances décrites chapitre 1, en utilisant Cplex 12.4, pour procéder à la sélection des routes. Les tests ont été effectués sur une machine avec 4CPU, 2.8GHz et 30Go de RAM. Dans cette section, nous présentons d'abord les résultats obtenus sur les instances de la littérature puis ceux sur les instances décrites chapitre 1.

6.2.1 Résultats sur les instances de la littérature

Afin de valider la méthodologie proposée dans ce chapitre et de comparer les résultats obtenus par les quatre variantes de l'algorithme de programmation dynamique (BD, DA, VV), (BD, DI, VV), (BD, DI, NV) et (UD, DI, NV), nous avons procédé à des tests sur des instances de la littérature. Tout d'abord, pour valider la méthodologie de génération de routes, nous l'avons testée sur des instances du OPTW. Ensuite, pour valider la méthodologie dans sa globalité, nous avons procédé à des tests sur des instances du TOPTW. Dans cette section, nous présenterons d'abord les résultats sur les instances du OPTW.

Résultats sur les instances du OPTW

Afin de pouvoir comparer les quatre variantes de l'algorithme de programmation dynamique (BD, DA, VV), (BD, DI, VV), (BD, DI, NV) et (UD, DI, NV), nous les avons testées sur les instances proposées par Righini et Salani [58]. Pour ces tests, comme il s'agit

de résoudre un problème de type OPTW, nous avons utilisé les mêmes labels que dans le code de Salani, mais avec nos variantes codées en `C++`. Nous avons également exécuté le code de Salani sur notre machine afin d'obtenir des temps de calcul comparables. On notera que les résultats obtenus avec le code de Salani sur les instances de type OPTW sont nouveaux, puisqu'ils n'ont fait l'objet d'aucune publication jusqu'à présent, et sont meilleurs que ceux proposés dans Righini et Salani [58]. Les résultats de ces tests sur les instances à 50, 100 clients et 100 clients avec fenêtres de temps larges sont regroupés dans les tableaux 6.1, 6.2 et 6.3. Les en-têtes de colonnes dans ces tableaux sont : Opt : valeur de la solution optimale, BKS : valeur de la meilleure solution connue, CPU : temps de résolution en secondes, Profit : profit collecté, # vis. : nombre de clients visités et Gap : gap entre la solution obtenue et la solution optimale (ou la meilleure solution connue). Dans ces tableaux, on se limite à deux heures de résolution (7200 secondes). Si la résolution n'a pu se terminer en le temps imparti, on inscrira un « - » dans la case correspondante. En ce qui concerne les solutions optimales et les meilleures solutions connues sur ces instances, nous avons utilisé celles obtenues par Hu et Lim [37]. De plus, on indique la valeur en gras s'il s'agit de la solution optimale.

Tableau 6.1 Comparaison sur les instances de type OPTW à 50 clients

Instance	Opt.	Méthodes exactes									Méthodes approchées							
		Code de Salani			BD, DA, VV			BD, DI, VV			BD, DI, NV				UD, DI, NV			
		CPU	Profit	# vis.	CPU	Profit	# vis.	CPU	Profit	# vis.	CPU	Profit	# vis.	Gap	CPU	Profit	# vis.	Gap
c101	**270**	0	270	10	0	270	10	0	270	10	0	**270**	10	0,00%	0,01	**270**	9	0,00%
c102	**300**	3,49	300	11	3,42	300	11	1,23	300	11	0,01	**300**	11	0,00%	0,03	**300**	10	0,00%
c103	**320**		-			-		3437,07	320	11	0,04	300	10	6,25%	0,16	**320**	10	0,00%
c104	**340**		-			-			-		0,09	330	11	2,94%	0,44	**340**	10	0,00%
c105	**300**	0	300	11	0	300	11	0,01	300	11	0	**300**	11	0,00%	0,01	**300**	10	0,00%
c106	**280**	0	280	10	0	280	10	0	280	10	0	**280**	10	0,00%	0,01	**280**	9	0,00%
c107	**310**	0,01	310	11	0,01	310	11	0,01	310	11	0,01	**310**	11	0,00%	0,01	**310**	9	0,00%
c108	**320**	0,01	320	11	0,01	320	11	0,01	320	11	0,01	**320**	11	0,00%	0,03	**320**	10	0,00%
c109	**340**	0,15	340	11	0,15	340	11	0,14	340	11	0,02	**340**	11	0,00%	0,03	**340**	10	0,00%
r101	**126**	0	126	5	0	126	5	0	126	5	0	**126**	5	0,00%	0	**126**	4	0,00%
r102	**198**	0,16	198	9	0,16	198	9	0,06	198	9	0,01	182	8	8,08%	0,03	195	8	1,52%
r103	**214**	11,29	214	10	11,13	214	10	0,92	214	10	0,02	202	10	5,61%	0,08	208	8	2,80%
r104	**227**	1607,81	227	10	1602,63	227	10	24,12	227	10	0,05	225	10	0,88%	0,27	223	9	1,76%
r105	**159**	0	159	6	0	159	6	0,04	159	6	0	**159**	6	0,00%	0,01	**159**	5	0,00%
r106	**208**	0,2	208	10	0,2	208	10	0,08	208	10	0,01	203	8	2,40%	0,03	203	7	2,40%
r107	**220**	10,43	220	10	10,29	220	10	1	220	10	0,03	216	10	1,82%	0,10	210	8	4,55%
r108	**227**	1371,02	227	10	1372,44	227	10	21,94	227	10	0,05	225	10	0,88%	0,22	223	9	1,76%
r109	**192**	0,01	192	8	0,01	192	8	0,01	192	8	0,01	**192**	8	0,00%	0,03	**192**	7	0,00%
r110	**208**	0,06	208	9	0,06	208	9	0,03	208	9	0,01	**208**	9	0,00%	0,06	**208**	8	0,00%
r111	**223**	0,78	223	9	0,78	223	9	0,23	223	9	0,02	207	9	7,17%	0,09	211	8	5,38%
r112	**226**	2,65	226	10	2,67	226	10	0,55	226	10	0,03	225	10	0,44%	0,09	210	8	7,08%
rc101	**180**	0	180	7	0	180	7	0	180	9	0	**180**	9	0,00%	0,01	**180**	8	0,00%
rc102	**230**	0,02	230	9	0,02	230	9	0,02	230	10	0,01	**230**	9	0,00%	0,01	**230**	8	0,00%
rc103	**240**	0,17	240	9	0,18	240	9	0,08	240	9	0,01	**240**	9	0,00%	0,03	**240**	8	0,00%
rc104	**270**	3,96	270	10	3,96	270	10	0,77	270	10	0,02	260	10	3,70%	0,05	260	9	3,70%
rc105	**210**	0,02	210	9	0,02	210	9	0,01	210	9	0,01	**210**	9	0,00%	0,01	**210**	8	0,00%
rc106	**210**	0,01	210	8	0,01	210	8	0,01	210	8	0	**210**	8	0,00%	0,01	**210**	7	0,00%
rc107	**240**	0,18	240	10	0,18	240	10	0,08	240	10	0,01	230	8	4,17%	0,03	230	7	4,17%
rc108	**250**	1,7	250	9	1,71	250	9	0,59	250	9	0,02	**250**	9	0,00%	0,04	240	8	4,00%

Tableau 6.2 Comparaison sur les instances de type OPTW à 100 clients

Instance	Opt.	Méthodes exactes									Méthodes approchées							
		Code de Salani			BD, DA, VV			BD, DI, VV			BD, DI, NV				UD, DI, NV			
		CPU	Profit	# vis.	CPU	Profit	# vis.	CPU	Profit	# vis.	CPU	Profit	# vis.	Gap	CPU	Profit	# vis.	Gap
c101	**320**	0,02	320	10	0,02	320	10	0,03	320	10	0,01	**320**	10	0,00%	0,05	**320**	9	0,00%
c102	**360**		-			-			-		0,09	**360**	11	0,00%	0,29	**360**	10	0,00%
c103	**400**		-			-			-		0,26	380	10	5,00%	1,19	390	10	2,50%
c104	**420**		-			-			-		0,5	380	10	9,52%	2,12	410	10	2,38%
c105	**340**	0,03	340	10	0,03	340	10	0,04	340	10	0,03	**340**	10	0,00%	0,18	**340**	9	0,00%
c106	**340**	0,05	340	10	0,05	340	10	0,06	340	10	0,04	**340**	10	0,00%	0,14	**340**	9	0,00%
c107	**370**	0,06	370	11	0,06	370	11	0,07	370	11	0,04	**370**	11	0,00%	0,11	**370**	10	0,00%
c108	**370**	0,11	370	11	0,11	370	11	0,13	370	11	0,07	**370**	11	0,00%	0,18	**370**	10	0,00%
c109	**380**	1,49	380	11	1,46	380	11	1,55	380	11	0,14	**380**	11	0,00%	0,33	**380**	10	0,00%
r101	**198**	0,01	198	9	0,01	198	9	0,01	198	9	0,01	**198**	9	0,00%	0,03	**198**	8	0,00%
r102	**286**	529,28	286	11	512,15	286	11	335,95	286	11	0,07	267	11	6,64%	0,34	**286**	10	0,00%
r103	**293**		-			-			-		0,2	290	11	1,02%	0,97	292	10	0,34%
r104	303		-			-			-		0,4	299	12	1,32%	2,03	303	11	0,00%
r105	**247**	0,04	247	11	0,04	247	11	0,06	247	11	0,02	**247**	11	0,00%	0,1	**247**	10	0,00%
r106	**293**	289,26	293	11	280,37	293	11	199,51	293	11	0,13	282	11	3,75%	0,51	**293**	10	0,00%
r107	**299**		-			-			-		0,27	292	11	2,34%	1,12	292	12	2,34%
r108	308		-			-			-		0,47	305	12	0,97%	2,14	303	11	1,62%
r109	**277**	0,2	277	12	0,21	277	12	0,23	277	12	0,07	275	12	0,72%	0,33	275	11	0,72%
r110	**284**	3,33	284	13	3,34	284	13	3,15	284	13	0,16	281	11	1,06%	1,21	282	11	0,70%
r111	**297**	496,88	297	12	476,79	297	12	330,19	297	12	0,25	289	11	2,69%	0,96	294	11	1,01%
r112	**298**		-			-		3632,62	298	12	0,33	289	12	3,02%	1,41	287	11	3,69%
rc101	**219**	0,02	219	9	0,02	219	9	0,03	219	9	0,01	**219**	11	0,00%	0,06	**219**	8	0,00%
rc102	**266**	1,56	266	10	1,59	266	10	1,49	266	10	0,06	**266**	10	0,00%	0,41	**266**	9	0,00%
rc103	**266**	236,36	266	10	235,25	266	10	135,54	266	10	0,14	**266**	10	0,00%	1,04	**266**	9	0,00%
rc104	**301**		-			-			-		0,26	**301**	11	0,00%	2,24	**301**	10	0,00%
rc105	**244**	0,21	244	12	0,21	244	12	0,23	244	12	0,04	241	11	1,23%	0,22	**244**	10	0,00%
rc106	**252**	0,13	252	11	0,17	252	11	0,15	252	11	0,05	**252**	11	0,00%	0,31	246	9	2,38%
rc107	**277**	3,24	277	10	3,36	277	10	2,89	277	10	0,12	**277**	10	0,00%	0,61	**277**	9	0,00%
rc108	**298**	52	298	11	52,51	298	11	29,69	298	11	0,21	287	11	3,69%	0,91	278	9	6,71%

Tableau 6.3 Comparaison sur les instances de type OPTW à 100 clients (fenêtres de temps larges)

Instance	BKS	Méthodes exactes									Méthodes approchées							
		Code de Salani			BD, DA, VV			BD, DI, VV			BD, DI, NV				UD, DI, NV			
		CPU	Profit	# vis.	CPU	Profit	# vis.	CPU	Profit	# vis.	CPU	Profit	# vis.	Gap	CPU	Profit	# vis.	Gap
c201	870	0,04	870	30	0,09	870	30	0,03	870	30	0,04	**870**	30	0,00%	1,32	**870**	29	0,00%
c202	930		-			-			-		1,00	880	31	5,38%	16,43	930	31	0,00%
c203	960		-			-			-		3,61	850	29	11,46%	44,43	940	30	2,08%
c204	980		-			-			-		7,41	800	26	18,37%	386,29	960	31	2,04%
c205	910	0,18	910	31	0,21	910	31	0,11	910	31	0,12	**910**	31	0,00%	5,04	900	30	1,10%
c206	930	1,05	930	32	1,13	930	32	0,74	930	32	0,29	**930**	32	0,00%	6,30	920	30	1,08%
c207	930	19,54	930	31	13,52	930	31	10,96	930	31	0,40	920	30	1,08%	5,50	910	30	2,15%
c208	950	712,23	950	31	556,74	950	31	531,54	950	31	0,40	940	31	1,05%	8,52	940	30	1,05%
r201	797	8,61	797	38	7,37	797	38	5,06	797	38	0,80	790	38	0,88%	50,79	793	37	0,50%
r202	929		-			-			-		18,88	837	42	9,90%	163,18	884	43	4,84%
r203	1021		-			-			-		276,96	849	40	16,85%	2874,33	966	46	5,39%
r204	1086		-			-			-		1427,75	772	34	28,91%	5358,00	1019	49	6,17%
r205	953		-			-			-		5,18	933	44	2,10%	174,74	908	41	4,72%
r206	1029		-			-			-		63,12	867	40	15,74%	661,86	976	46	5,15%
r207	1072		-			-			-		378,71	857	39	20,06%	4460,81	1016	50	5,22%
r208	1112		-			-			-		1261,57	809	36	27,25%	4751,50	1055	52	5,13%
r209	950		-			-			-		20,06	825	39	13,16%	449,80	903	43	4,95%
r210	987		-			-			-		104,78	811	37	17,83%	528,59	942	46	4,56%
r211	1046		-			-			-		35,83	980	45	6,31%	449,11	991	46	5,26%
rc201	795	1,53	795	33	1,58	795	33	0,77	795	33	0,61	785	33	1,26%	29,93	785	32	1,26%
rc202	936		-			-			-		8,25	866	35	7,48%	314,39	895	38	4,38%
rc203	1003		-			-			-		138,64	863	37	13,96%	5746,75	955	40	4,79%
rc204	1140		-			-			-		204,01	1043	42	8,51%	7200,00	1047	44	8,16%
rc205	859		-			-			-		4,62	840	37	2,21%	290,15	849	36	1,16%
rc206	895		-			-			-		3,44	862	35	3,69%	405,35	864	34	3,46%
rc207	983		-			-			-		13,76	927	42	5,70%	589,84	908	38	7,63%
rc208	1053		-			-			-		20,97	1017	41	3,42%	565,38	1028	42	2,37%

Dans les tableaux 6.1, 6.2 et 6.3, on observe une diminution des temps de calcul entre la variante (BD, DA, VV) et la version de Salani qui sont pourtant les mêmes au niveau du pseudo-code. Cette diminution est liée aux différences d'implémentation (ordre des tests dans l'algorithme, code en `C++` et non en `C`...). On observe également que la variante (BD, DI, VV) consistant à limiter l'extension d'un label forward à $T/2$ donne de meilleurs temps de résolution que la variante (BD, DA, VV). Elle permet de résoudre deux instances en moins de 2 heures qu'il n'était pas possible de résoudre auparavant. De plus, on remarquera dans le tableau 6.3 que l'optimalité de 7 meilleures solutions connues est prouvée par notre méthode ainsi que par le code de Salani (sur les instances c201, c205, c206, c207, c208, r201 et rc201).

En ce qui concerne les méthodes approchées, on observe que la variante (BD, DI, NV) est nettement plus rapide que toutes les autres variantes (exactes ou approchées) et qu'elle fournit des solutions situées à moins de 10% de la solution optimale sur les instances avec fenêtres de temps serrées. Toutefois, dès que l'on traite des instances avec fenêtres de temps larges (instances de type C2, R2, RC2), les solutions qu'elle fournit peuvent se trouver à 30% de la solution optimale, ce qui n'est pas acceptable. En effet, dans la variante (BD, DI, NV), on procède à une programmation dynamique forward, une programmation dynamique backward avec pour critère le nombre de clients visités et non le vecteur de visite. Ainsi, sur des instances aux fenêtres de temps larges, on peut avoir de très bonnes solutions en forward et de très bonnes solutions en backward mais lors de la jonction, il y a une forte probabilité que ces solutions aient des clients en commun et ne puissent être jointes. On peut donc obtenir des solutions de qualité très médiocre. Pour remédier à ce problème, dans la variante (UD, DI, NV), on procède comme dans la variante (BD, DI, NV) mais dans une seule direction. Il s'agit d'une programmation dynamique classique et non bidirectionnelle bornée. On observe que cette variante (UD, DI, NV) fournit des solutions de meilleure qualité sur l'ensemble des instances avec un gap à optimalité ne dépassant pas 8,2%. Toutefois, si cette variante fournit de très bonnes solutions, elle requiert des temps de calcul nettement plus élevés que la variante (BD, DI, NV).

Résultats sur les instances du TOPTW

Avant de tester et valider la méthode proposée dans ce chapitre, nous avons testé les variantes approchées sur des instances de la littérature afin de connaître le gap entre les solutions obtenues par notre méthode et les solutions optimales. Nous avons donc choisi d'utiliser les instances de Vansteenwegen *et al.* [71] comme il s'agit d'instances du TOPTW pour lesquelles les solutions optimales sont connues. Plus précisément, nous avons choisi de procéder à des tests sur les instances de Vansteenwegen *et al.* [71] de type r2, c2 et rc2 (avec fenêtres de temps lagres) car ce sont les instances qui se rapprochent le plus de nos instances

(décrites au chapitre 1). Dans ces instances de Vansteenwegen *et al.* [71], plusieurs véhicules et un unique dépôt sont considérés. En utilisant notre méthodologie (construction des routes à l'aide de la programmation dynamique puis sélection exacte d'une route par véhicule) telle quelle, on obtient des solutions de piètre qualité du fait de la symétrie du problème. Nous avons donc ajouté une phase de génération d'une partition de l'ensemble des sommets associés aux clients en amont. Ainsi, on procède à plusieurs partitionnements en un nombre de sous-ensembles égal au nombre de véhicules utilisés dans la solution optimale. Pour chacun de ces sous-ensembles, on procède à la génération de routes de notre méthode. Ensuite, on sélectionne une route par véhicule parmi toutes les routes de tous les sous-ensembles. Les résultats obtenus en appliquant cette méthodologie avec les variantes (BD, DI, NV) et (UD, DI, NV) sont synthétisés dans le Tableau 6.4. Dans ce tableau, les en-têtes de colonnes sont les suivants : Nb veh : nombre de véhicules utilisés dans la solution optimale, Opt : valeur de la solution optimale (profit collecté), CPU : temps de calcul en secondes, Valeur : valeur de la solution obtenue, Gap : gap entre la solution obtenue et la solution optimale. Si la génération de routes n'a pu terminer en moins de 2 heures sur une instance, on mettra un « - » dans la case correspondante.

Tableau 6.4 Comparaison sur les instances de Vansteenwegen *et al.* [71]

Instance	Nb veh	Opt	Variante BD, DI, NV			Variante UD, DI, NV		
			CPU	Valeur	Gap	CPU	Valeur	Gap
c201	4	1810	0,07	1810	0,00%	1,38	1810	0,00%
c202	4	1810	0,54	1810	0,00%	5,38	1790	1,10%
c203	4	1810	0,94	1810	0,00%	6,90	1800	0,55%
c204	4	1810	1,62	1680	7,18%	4,95	1800	0,55%
c205	4	1810	0,37	1810	0,00%	1,70	1810	0,00%
c206	4	1810	0,20	1810	0,00%	2,51	1810	0,00%
c207	4	1810	0,25	1810	0,00%	2,72	1810	0,00%
c208	4	1810	0,24	1810	0,00%	2,30	1810	0,00%
r201	4	1458	0,81	1385	5,01%	53,98	1384	5,08%
r202	3	1458	3,33	1354	7,13%	248,69	1383	5,14%
r203	3	1458	21,98	1353	7,20%	1168,54	1410	3,29%
r204	2	1458	743,02	1140	21,81%		-	
r205	3	1458	1,92	1413	3,09%	603,49	1398	4,12%
r206	3	1458	10,47	1373	5,83%	714,92	1428	2,06%
r207	2	1458	291,05	1224	16,05%		-	
r208	2	1458	758,26	1186	18,66%		-	
r209	3	1458	6,13	1381	5,28%	609,40	1427	2,13%
r210	3	1458	15,74	1332	8,64%	831,68	1421	2,54%
r211	2	1458	10,49	1313	9,95%	4307,51	1321	9,40%
rc201	4	1724	0,38	1695	1,68%	12,81	1684	2,32%
rc202	3	1724	2,01	1619	6,09%	196,40	1640	4,87%
rc203	3	1724	11,90	1605	6,90%	470,05	1677	2,73%
rc204	3	1724	26,07	1592	7,66%	403,16	1716	0,46%
rc205	4	1724	0,62	1692	1,86%	22,76	1701	1,33%
rc206	3	1724	1,48	1679	2,61%	226,77	1663	3,54%
rc207	3	1724	3,19	1640	4,87%	291,88	1645	4,58%
rc208	3	1724	9,60	1670	3,13%	216,75	1680	2,55%

Dans ce tableau, on observe sur 3 instances que les gaps à optimalité peuvent atteindre

20%. Pour y remédier, nous aurions pu développer encore la méthodologie. Toutefois, notre objectif est avant tout de valider notre méthodologie sur les instances de la littérature. Nous pouvons d'ailleurs observer, sur le reste des instances (comprenant 3 à 4 véhicules), que le gap à optimalité ne dépasse pas 10%. Si les temps de calcul sont nettement plus faibles dans la variante bidirectionnelle, les gaps à optimalité sont nettement plus élevés. En effet, quand on procède à la programmation dynamique bidirectionnelle, étant donné que les fenêtres de temps sont larges (instances de type c2, r2 et rc2) et qu'un véhicule peut desservir jusqu'à 50 clients sur une tournée, on peut construire des chemins très longs en forward et en backward et souvent, ces chemins ne peuvent être joints car ils ont un ou plusieurs clients en commun. Toutefois, il peut également arriver que les solutions en bidirectionnel soient meilleures que celles dans une direction. En effet, en bidirectionnel, on génère plus de chemins (donc plus de diversité), ce qui aide lors de la sélection exacte des routes.

6.2.2 Résultats sur les instances du chapitre 1

Après avoir procédé aux tests et à la validation de l'heuristique basée sur la génération de colonnes sur les instances de la littérature, nous procédons à présent aux tests sur les instances décrites au chapitre 1. Dans cette section, nous procèderons d'abord au réglage des paramètres puis nous présenterons les résultats de la méthode de ce chapitre sur les instances du chapitre 1 et enfin, nous comparerons la méthode de ce chapitre avec la méthode du chapitre précédent.

Réglage des paramètres

Concernant le réglage des paramètres, nous utilisons le même réglage de paramètres que dans les chapitres précédents : horizon de temps $T = 480$ minutes, vitesse minimale $v_{min} = 20km/h$ et vitesse modale $v_{mod} = 40km/h$. Après avoir converti ces valeurs v_{min} et v_{mod} en unités arbitraires par minute, on calcule le temps de parcours unitaire minimal $\underline{\delta} = \lceil 100/v_{max} \rceil$ et modal $\hat{\delta} = \lceil 100/v_{mod} \rceil$. Les temps de parcours minimaux et modaux sont ensuite obtenus en utilisant les formules $\underline{\tau}_{ij} = \left\lceil D_{ij}\underline{\delta} \right\rceil$ et $\hat{\tau}_{ij} = \left\lceil D_{ij}\hat{\delta} \right\rceil$. Pour le réglage des temps de service, nous avons choisi des temps de service minimaux et modaux respectivement de 15 et 22 minutes pour les clients optionnels et de 30 et 35 minutes pour les clients obligatoires. Comme précédemment, nous avons supposé que le service d'un client optionnel, quel qu'il soit, génère un profit $p_o = 100$ et nous avons choisi $\alpha = 1$. Pour le choix du profit associé aux clients obligatoires p_m (utilisé durant la génération de routes), nous avons procédé à une série de tests des variantes approchées (UD, DI, NV) et (BD, DI, NV) sur les instances à 40 clients dont les résultats sont synthétisés dans les tableaux 6.5 et 6.6. Afin d'accorder la

priorité aux clients obligatoires par rapport aux clients optionnels, on choisit de faire varier le profit des clients obligatoires entre 200 et 500. Les en-têtes de colonnes dans ces tableaux sont : CPU : temps de résolution en secondes, # ND : nombre de clients optionnels non desservis et Dist. : distance totale parcourue en kilomètres.

Tableau 6.5 Influence du profit des clients obligatoires, variante (UD, DI, NV)

Nb de clients	Nb de clients oblig.	Profit = 200			Profit = 300			Profit = 400			Profit = 500		
		CPU	# ND	Dist.	CPU	# ND	Dist.	CPU	# ND	Dist.	CPU	# ND	Dist.
30	5	1,1	0,2	214,01	1,3	0,2	219,04	1,3	0,2	218,98	1,1	0,2	219,07
	6	1,4	0,4	207,00	1,6	0,4	207,00	1,7	0,2	212,31	1,7	0,2	212,31
	7	1,2	0,2	222,57	1,4	0,2	222,57	1,4	0,2	221,22	1,6	0,2	222,53
	8	1,3	0,6	217,62	1,6	0,6	218,86	1,7	0,6	216,19	1,7	0,6	216,19
	9	1,2	0,8	218,35	1,5	0,8	216,19	1,6	0,8	215,63	1,6	0,8	221,29
40	5	2,8	3,6	223,80	3,0	3,6	225,75	2,8	3,8	222,27	3,3	3,8	222,27
	6	3,5	3,2	228,96	4,5	3,0	236,90	4,4	3,0	236,90	4,8	3,0	236,89
	7	4,5	4,2	223,44	5,0	4,0	224,93	5,5	4,2	225,61	5,5	4,2	225,86
	8	3,7	3,6	226,63	4,8	3,4	236,65	4,9	3,4	236,65	4,6	3,6	232,82
	9	3,6	4,0	234,51	4,1	4,0	233,28	4,4	4,2	234,54	4,4	4,4	232,61
50	5	4,3	7,6	232,75	5,3	8,2	218,78	5,6	8,2	226,48	6,2	8,6	223,80
	6	5,9	8,0	235,26	7,9	7,8	235,51	8,3	7,8	235,51	8,4	7,8	235,51
	7	7,3	9,0	221,51	9,6	9,2	220,29	10,1	9,4	226,32	10,3	9,8	221,33
	8	7,0	7,6	232,06	8,9	7,6	231,35	9,6	8,2	224,43	10,4	8,4	241,88
	9	7,6	9,4	227,00	9,1	9,4	226,27	9,7	10,0	220,24	10,1	10,4	220,60

Tableau 6.6 Influence du profit des clients obligatoires, variante (BD, DI, NV)

Nb de clients	Nb de clients oblig.	Profit = 200			Profit = 300			Profit = 400			Profit = 500		
		CPU	# ND	Dist.	CPU	# ND	Dist.	CPU	# ND	Dist.	CPU	# ND	Dist.
30	5	22,5	0,0	199,88	21,6	0,0	199,88	23,1	0,0	199,74	23,5	0,0	199,88
	6	27,2	0,0	213,70	37,1	0,0	214,48	30,4	0,0	214,48	31,0	0,0	214,48
	7	28,1	0,0	219,29	33,5	0,0	219,29	37,1	0,0	219,29	35,1	0,0	219,29
	8	21,9	0,0	214,55	24,8	0,0	214,50	25,9	0,0	214,50	26,2	0,0	214,50
	9	23,4	0,0	216,61	29,0	0,0	216,63	27,7	0,0	216,63	25,0	0,0	216,63
40	5	80,4	1,6	230,88	96,8	1,4	238,37	88,1	1,4	238,37	104,2	1,4	238,37
	6	93,1	1,8	249,82	119,9	1,8	249,91	117,2	1,8	249,91	117,2	1,8	249,91
	7	104,6	2,6	238,15	122,4	2,2	254,18	112,5	2,2	254,18	122,5	2,2	254,18
	8	77,6	2,0	242,84	97,6	2,0	242,00	94,1	2,0	242,00	98,0	2,0	242,00
	9	107,8	2,6	238,42	120,3	2,4	239,64	120,2	2,4	239,59	111,5	2,4	239,59
50	5	173,3	8,0	228,42	184,1	8,0	226,26	173,9	8,0	226,26	184,1	8,0	226,26
	6	192,9	8,6	221,53	212,0	8,6	221,88	181,5	8,6	221,88	184,6	8,6	221,53
	7	198,5	9,2	224,26	240,1	9,2	221,45	235,3	9,2	221,80	233,9	9,2	221,80
	8	189,5	9,2	214,84	218,1	9,2	214,84	201,6	9,2	214,84	219,0	9,2	214,84
	9	180,2	9,4	220,79	261,2	9,4	220,86	245,2	9,4	220,86	237,2	9,4	220,86

Dans ces tableaux, on observe, pour la variante (BD, DI, NV), que le nombre de clients non desservis sur les instances à 40 clients est plus élevé lorsque le profit associé aux clients obligatoires vaut 200. On observe également que, pour un nombre de clients non desservis identique, la distance totale parcourue est plus élevée pour un profit de 200 associé aux clients obligatoires. Quant aux temps de calcul, ils ont tendance à augmenter avec le profit (mais les valeurs restent comparables). On évitera donc de choisir la valeur $p_m = 200$. En ce

qui concerne la variante (UD, DI, NV), on observe que le nombre de clients non desservis atteint le plus souvent sa valeur minimale pour $p_m = 300$. Comme précédemment, les temps de calcul augmentent avec le profit mais gardent des valeurs comparables. Pour la suite des expérimentations, on choisira donc d'attribuer un profit $p_m = 300$ aux clients obligatoires.

Résultats de la méthode

Dans cette section, nous avons testé les trois variantes (BD, DI, VV), (BD, DI, NV) et (UD, DI, NV) sur les instances vues au chapitre 1. Toutefois, la variante exacte (BD, DI, VV) (avec vecteur de visite) générant trop de labels, on ne peut procéder à la sélection exacte d'une route par véhicule parmi ces labels. On présentera donc uniquement les solutions obtenues à l'aide des variantes (BD, DI, NV) et (UD, DI, NV) (suivies d'une méthode de postoptimisation de type 2-Opt, comme indiqué précédemment). Afin d'obtenir des résultats complets, nous avons procédé, après l'étape de planification, à l'étape d'exécution (i.e. la programmation dynamique), décrite au chapitre 4, à la fin de la méthode. Puis, nous avons réalisé 100 simulations par instance. Les résultats obtenus pour ces deux variantes sont synthétisés dans les tableaux 6.7 et 6.8. Les en-têtes de ces tableaux sont les suivants : CPU : temps de calcul en secondes, CPU : temps de la génération de routes en secondes, # moy. non servis : nombre moyen de clients non servis, WR : stratégie de programmation dynamique considérant toute la route et OS : stratégie de programmation dynamique considérant un seul segment de route (cf. chapitre précédent). Dans ce tableau, le nombre indiqué de clients non servis après simulation comprend le nombre de clients non servis avant simulation plus le nombre de clients devenus non desservis durant la simulation.

Tableau 6.7 Résultats avant/après simulation pour la variante (BD, DI, NV)

Nb de clients	Nb de clients oblig.	Avant simulation				Après simulation					
		CPU	CPU Gen	# moy. non servis	distance moy.	# moy. non servis		distance moy.		retard moy.	
						WR	OS	WR	OS	WR	OS
30	5	21,5	0,8	0,0	200	1,2	0,7	197	199	0,6	7,4
	6	37,1	0,8	0,0	214	3,8	1,9	199	208	1,1	19,2
	7	33,5	0,8	0,0	219	4,0	2,8	206	213	2,1	30,8
	8	24,8	0,8	0,0	214	3,2	2,2	201	208	2,4	29,3
	9	29,0	0,8	0,0	217	4,5	2,0	203	214	4,8	62,1
40	5	96,8	2,1	1,4	238	7,8	6,4	206	220	1,8	44,7
	6	119,9	2,3	1,8	250	9,9	7,9	214	229	2,2	59,1
	7	122,4	2,4	2,2	254	11,7	8,8	221	241	2,7	78,2
	8	97,6	2,4	2,0	242	11,1	7,9	201	232	2,7	93,0
	9	120,3	2,6	2,4	240	12,3	7,8	198	232	8,2	135,0
50	5	184,1	4,2	8,0	226	16,3	14,6	195	214	1,8	68,9
	6	212,0	4,6	8,6	222	17,5	15,5	193	208	1,5	48,9
	7	240,1	4,9	9,2	221	19,1	16,7	194	208	2,9	62,8
	8	218,1	5,4	9,2	215	19,7	16,4	188	205	3,9	82,2
	9	261,2	5,7	9,4	221	21,3	16,8	187	210	3,5	102,6

Tableau 6.8 Résultats avant/après simulation pour la variante (UD, DI, NV)

Nb de clients	Nb de clients oblig.	Avant simulation				Après simulation					
		CPU	CPU Gen	# moy. non servis	distance moy.	# moy. non servis		distance moy.		retard moy.	
						WR	OS	WR	OS	WR	OS
30	5	1,3	0,6	0,2	219	6,2	3,6	183	198	1,3	40,1
	6	1,6	0,8	0,4	207	5,9	4,8	180	185	1,8	21,4
	7	1,4	0,9	0,2	223	7,4	4,8	190	200	3,1	57,0
	8	1,6	0,9	0,6	219	6,3	5,6	181	181	1,3	8,4
	9	1,5	0,8	0,8	216	6,2	4,8	185	188	2,7	32,2
40	5	3,0	1,8	3,6	226	9,6	8,4	194	199	1,3	21,4
	6	4,5	2,7	3,0	237	11,3	10,0	187	190	1,9	29,4
	7	5,0	3,1	4,0	225	11,7	10,7	179	182	2,7	25,7
	8	4,8	3,0	3,4	237	13,0	9,5	187	197	4,1	88,4
	9	4,1	3,1	4,0	233	12,6	10,5	179	186	6,1	56,3
50	5	5,3	4,1	8,2	219	16,8	14,5	184	192	1,6	57,9
	6	7,9	6,3	7,8	236	18,1	15,8	181	187	2,5	57,6
	7	9,6	6,9	9,2	220	19,6	17,3	173	180	2,2	51,1
	8	8,9	7,8	7,6	231	19,9	17,7	180	188	3,6	51,9
	9	9,1	7,6	9,4	226	21,2	16,6	182	197	6,1	125,7

Dans ces tableaux, on observe une cohérence des résultats obtenus : le nombre de clients non desservis n'augmente pas beaucoup au cours des simulations. Ce qui signifie que les routes ne sont pas bouleversées au cours des simulations. On constate aussi que le nombre moyen de clients desservis est compris entre 25 et 32 clients. En ce qui concerne la distance, on observe que, pour un même nombre de clients desservis, la distance moyenne parcourue diminue, pour un retard similaire quand la taille des instances augmente. En effet, plus les instances contiennent de clients, meilleures sont les routes construites (car il y a plus de choix possibles).

En comparant les deux stratégies de programmation dynamique (WR et OS), on remarque que la stratégie ne considérant qu'un segment est préférable en ce qui concerne le nombre de clients non desservis (comme au chapitre précédent) mais on préfère la stratégie considérant toute la route du point de vue de la distance parcourue (pour comparer la distance parcourue avec un nombre de clients visités différent, on calcule le ratio distance/nombre de clients visités et on compare le ratio obtenu pour les deux stratégies).

En comparant les deux méthodes, on observe que le nombre de clients non servis avant simulation est plus faible avec la variante bidirectionnelle. Il en est de même après simulation, quelle que soit la stratégie. Toutefois, on constate également que la différence entre le nombre de clients non desservis pour la méthode bidirectionnelle et celui obtenu pour la variante unidirectionnelle diminue quand la taille des instances augmente (elle est quasiment nulle pour les instances de taille 50). En ce qui concerne les temps de calcul, ils sont plus élevés pour la variante bidirectionnelle mais restent inférieurs à 5 minutes. En effet, dans la variante bidirectionnelle, on génère un plus grand nombre de routes et la sélection exacte nécessite donc

plus de temps. En comparant les solutions obtenues après simulation par les deux méthodes, on constate que le retard moyen et la distance moyenne (ratio distance parcourue / nombre de clients visités) sont plus élevés dans la méthode bidirectionnelle. Toutefois, les différences au niveau de la distance et du retard pour la stratégie WR restent faibles. De plus, comme nous l'avons mentionné plus haut, le nombre de clients desservis par la méthode bidirectionnelle est nettement supérieur. C'est donc cette variante approchée que nous retiendrons et que nous comparerons avec l'heuristique basée sur la priorité des clients présentée au chapitre précédent.

6.2.3 Comparaison de cette méthode avec la méthode précédente

Les résultats avant simulation (30, 40 et 50 clients) obtenus avec la méthode de ce chapitre et celle du chapitre précédent sont synthétisés dans le Tableau 6.9. Dans ce tableau, on

Tableau 6.9 Comparaison des 2 méthodes avant simulation

# clients	# clients obligatoires	Heuristique basée sur la priorité des clients			Heuristique basée sur la génération de colonnes		
		CPU (s)	non servis	Distance (km)	CPU (s)	non servis	Distance (km)
30	5	43	1,4	220	22	0,0	200
	6	66	2,8	232	37	0,0	214
	7	53	2,4	230	34	0,0	219
	8	34	3,4	216	25	0,0	214
	9	24	3	217	29	0,0	217
40	5	3277	7,2	202	97	1,4	238
	6	962	10,2	217	120	1,8	250
	7	1451	10	211	122	2,2	254
	8	1880	10	210	98	2,0	242
	9	1053	10	206	120	2,4	240
50	5	1227	15,4	192	184	8,0	226
	6	3064	17,6	197	212	8,6	222
	7	2844	18	192	240	9,2	221
	8	2323	18,6	200	218	9,2	215
	9	1408	19,2	188	261	9,4	221

observe, avant simulation, que la méthode présentée dans ce chapitre est nettement plus rapide que l'heuristique basée sur la priorité des clients. En effet, elle permet de résoudre toutes les instances en moins de 5 minutes, alors que cela pouvait prendre jusqu'à une heure précédemment. En ce qui concerne la qualité des solutions, on peut également constater que le nombre de clients non desservis avant simulation est nettement moins élevé dans la méthode présentée ici que dans l'heuristique basée sur la priorité des clients. Tandis que l'on desservait 30 clients en moyenne dans l'heuristique basée sur la priorité des clients, quelle que soit la taille des instances, on dessert à présent 30 clients pour les instances à 30 clients, 38 clients pour les instances à 40 clients et 40 clients pour les instances à 50 clients (soit 10 clients de plus que précédemment). Sur les instances de taille 30, les solutions obtenues avec la méthode de ce chapitre sont de meilleure qualité : elles desservent plus de clients en parcourant moins

de distance que l'heuristique basée sur la priorité des clients. Cela se justifie par le fait que, dans l'heuristique basée sur la priorité des clients, on fixait le squelette des clients avant d'insérer les clients optionnels tandis qu'à présent, on construit directement des routes avec des clients obligatoires et optionnels, ce qui peut donc conduire à des solutions de meilleure qualité. Pour les autres instances, on ne peut pas comparer la qualité globale des solutions obtenues par les deux méthodes étant donné que, dans l'heuristique basée sur la priorité des clients, on s'assure que la solution obtenue avant simulation soit réalisable dans 90% des cas (ce qui implique forcément un nombre de clients non desservis plus élevé avant simulation). Pour juger de la qualité des solutions, il nous faut donc comparer les résultats obtenus après simulation. Ces résultats obtenus après simulation, avec les stratégies de programmation dynamique mentionnées au chapitre 4, sont regroupés dans les tableaux 6.10 et 6.11.

Tableau 6.10 Comparaison des 2 méthodes après simulation, stratégie WR

# clients	# clients obligatoires	Heuristique basée sur la priorité des clients			Heuristique basée sur la génération de colonnes		
		non servis	Distance (km)	retard	non servis	Distance (km)	retard
30	5	5,3	199	2	1,2	197	0,6
	6	5,6	212	3,1	3,8	199	1,1
	7	5,5	209	2,7	4,0	206	2,1
	8	7,6	195	3,2	3,2	201	2,4
	9	7	197	4,6	4,5	203	4,8
40	5	9,7	193	1,8	7,8	206	1,8
	6	12,7	204	2,7	9,9	214	2,2
	7	13,3	198	2,4	11,7	221	2,7
	8	13,2	198	3,6	11,1	201	2,7
	9	14,5	189	6,8	12,3	198	8,2
50	5	17,9	185	2	16,3	195	1,8
	6	21,1	189	3,6	17,5	193	1,5
	7	22,2	184	3,6	19,1	194	2,9
	8	22,1	187	4,3	19,7	188	3,9
	9	22,8	180	5,7	21,3	187	3,5

Tableau 6.11 Comparaison des 2 méthodes après simulation, stratégie OS

# clients	# clients obligatoires	Heuristique basée sur la priorité des clients			Heuristique basée sur la génération de colonnes		
		non servis	Distance (km)	retard	non servis	Distance (km)	retard
30	5	3,4	211	2	0,7	199	7,4
	6	5,6	212	3,2	1,9	208	19,2
	7	5,5	209	2,8	2,8	213	30,8
	8	6,1	204	3,2	2,2	208	29,3
	9	6,4	201	4,6	2,0	214	62,1
40	5	9,7	194	1,8	6,4	220	44,7
	6	12,7	204	2,7	7,9	229	59,1
	7	13,3	198	2,5	8,8	241	78,2
	8	13,2	198	3,6	7,9	232	93,0
	9	14	191	5,6	7,8	232	135,0
50	5	17,9	185	2	14,6	214	68,9
	6	20,5	189	3,6	15,5	208	48,9
	7	21,5	185	3,6	16,7	208	62,8
	8	22,1	187	4,4	16,4	205	82,2
	9	22,7	180	5,9	16,8	210	102,6

On observe dans ces tableaux que le nombre de clients non desservis est nettement plus élevé dans l'heuristique basée sur la priorité des clients que dans la méthode de ce chapitre, quelle que soit la stratégie de programmation dynamique. D'autre part, le retard est légèrement moins élevé dans la méthode approchée quand on considère la stratégie WR tandis qu'il est nettement plus élevé dans la stratégie OS. En effet, dans la stratégie consistant à considérer un seul segment, étant donné que la route est réalisable dans 90% des cas dans l'heuristique basée sur la priorité des clients, on ne peut pas avoir de retard moyen très élevé. Par contre, dans la méthode de ce chapitre, rien ne garantit cette réalisabilité. Ainsi, quand on considère la stratégie WR, on se prémunit contre des retards aux clients obligatoires car on prend les décisions en observant toute la route (et donc tous les clients obligatoires qui suivent). Par contre, dans la stratégie ne considérant qu'un segment à la fois, on se prémunit uniquement contre un éventuel retard au prochain client obligatoire mais pas aux clients obligatoires suivants. On peut donc obtenir des retards très élevés. Quelle que soit la stratégie en tout cas, on observe sur les instances à 30 clients que les solutions obtenues sont de meilleure qualité avec la méthode de ce chapitre (nombre de clients non desservis et distance parcourue moins élevée). Seul le retard pourrait prêter à discussion mais il ne dépasse pas 7 minutes par client obligatoire dans la stratégie WR. Comme on préfère desservir un maximum de clients, on préfèrera donc la méthode de ce chapitre.

6.3 Conclusion

Dans ce chapitre nous avons proposé une heuristique basée sur la génération de colonnes consistant à générer, pour chaque véhicule, des routes à l'aide d'une méthode de type pro-

grammation dynamique puis à sélectionner de façon exacte une route par véhicule en résolvant un programme linéaire en nombre entiers avec Cplex. Nous avons proposé plusieurs variantes de programmation dynamique pour la génération de routes : une variante exacte (bi-directionnelle, avec vecteur de visite) et deux variantes approchées (bidirectionnelle et unidirectionnelle avec nombre de visités). Etant donné que nos instances ne sont pas très contraintes (fenêtres de temps larges pour tous les clients), la méthode exacte ne permet pas de résoudre nos instances en moins de 2 heures. Par contre, elle nous a permis d'établir l'optimalité de plusieurs solutions approchées de la littérature (sur des instances de type OPTW). Quant aux variantes approchées, nous avons pu constater, dans les résultats expérimentaux, que la variante approchée bidirectionnelle était nettement plus rapide et fournissait de meilleures solutions que la variante approchée unidirectionnelle. En effet, cette méthode permet d'obtenir de bonnes solutions sur toutes les instances en moins de 5 minutes. Ensuite, nous avons pu comparer cette méthode avec l'heuristique basée sur la priorité des clients proposée au chapitre précédent. Nous avons alors constaté que la méthode de ce chapitre fournissait de meilleures solutions en moins de temps que la méthode précédente.

Toutefois, si cette variante approchée s'avère efficace, elle reste une méthode approchée. Etant données sa rapidité et son efficacité, il serait intéressant de proposer une méthode exacte basée sur la variante approchée bidirectionnelle, avec nombre de visités, de ce chapitre. C'est pourquoi, dans le chapitre suivant, nous proposons un algorithme de branch and price qui intègre la méthode approchée de ce chapitre.

CHAPITRE 7

Algorithme de branch and price

Dans le chapitre précédent, nous avons présenté une heuristique basée sur la génération de colonnes pour résoudre notre variante du problème de tournées de service. Cette méthode s'est montrée très efficace sur nos instances avec des temps de calcul de moins de 5 minutes. Toutefois, cette méthode, tout comme celle présentée au chapitre 4, reste une méthode approchée. Dans ce chapitre, nous proposons une méthode exacte pour l'étape de planification. Comme cette étape correspond à un problème de tournées de véhicules multi-dépôts, avec fenêtres de temps et priorité entre les clients (MDVRPTW avec priorité), on ne peut espérer résoudre ce problème en utilisant les méthodes standards d'énumération implicite. Toutefois, en formulant ce problème comme un problème de partitionnement, il nous devient possible de résoudre ce problème de façon exacte avec un algorithme de branch and price. Dans ce chapitre, nous rappellerons le principe du branch and price. Puis, nous reformulerons notre problématique sous forme d'un problème maître et d'un sous-problème. Nous détaillerons ensuite les composantes de la méthode de branch and price utilisées. Enfin, nous présenterons les résultats obtenus et nous conclurons sur cette méthode.

7.1 Principe des algorithmes de branch and price

La méthode de branch and price, introduite par Johnson [40], implémentée par Desrochers et Soumis [24] et baptisée par Savelsbergh [60] et Barnhart *et al.* [6], est une recherche arborescente (branch and bound) dans laquelle on ajoute de nouvelles colonnes à chaque noeud. L'idée, dans cette méthode, est de résoudre la relaxation linéaire du problème restreint à un sous-ensemble de variables (le nombre total de variables étant trop élevé pour une résolution directe) puis à brancher pour obtenir une solution entière. A chaque noeud de l'arbre, on procède à une génération de colonnes, c'est-à-dire à l'extension du sous-ensemble de variables (on résout un problème de pricing). La génération de colonnes, introduite par Gilmore et Gomory [32], est basée sur la décomposition du problème initial en deux problèmes : le problème maître et le sous-problème. Le problème maître correspond au problème initial dans lequel on se limite à un sous-ensemble de variables. Le sous-problème (ou problème de pricing), quant à lui, permet de générer de nouvelles variables, à ajouter au problème maître. Une itération de la génération de colonnes consiste alors à résoudre le problème maître puis à résoudre le sous-problème pour ajouter de nouvelles variables présentant un coût réduit intéressant dans

le problème maître (coût réduit négatif pour un problème de minimisation, positif pour un problème de maximisation). La génération de colonnes finit lorsqu'il n'y a plus de variables intéressantes à ajouter. La solution est alors optimale pour la relaxation linéaire du problème initial. La méthode est schématisée figure 7.1 (figure extraite du mémoire de thèse de Tricoire [68]).

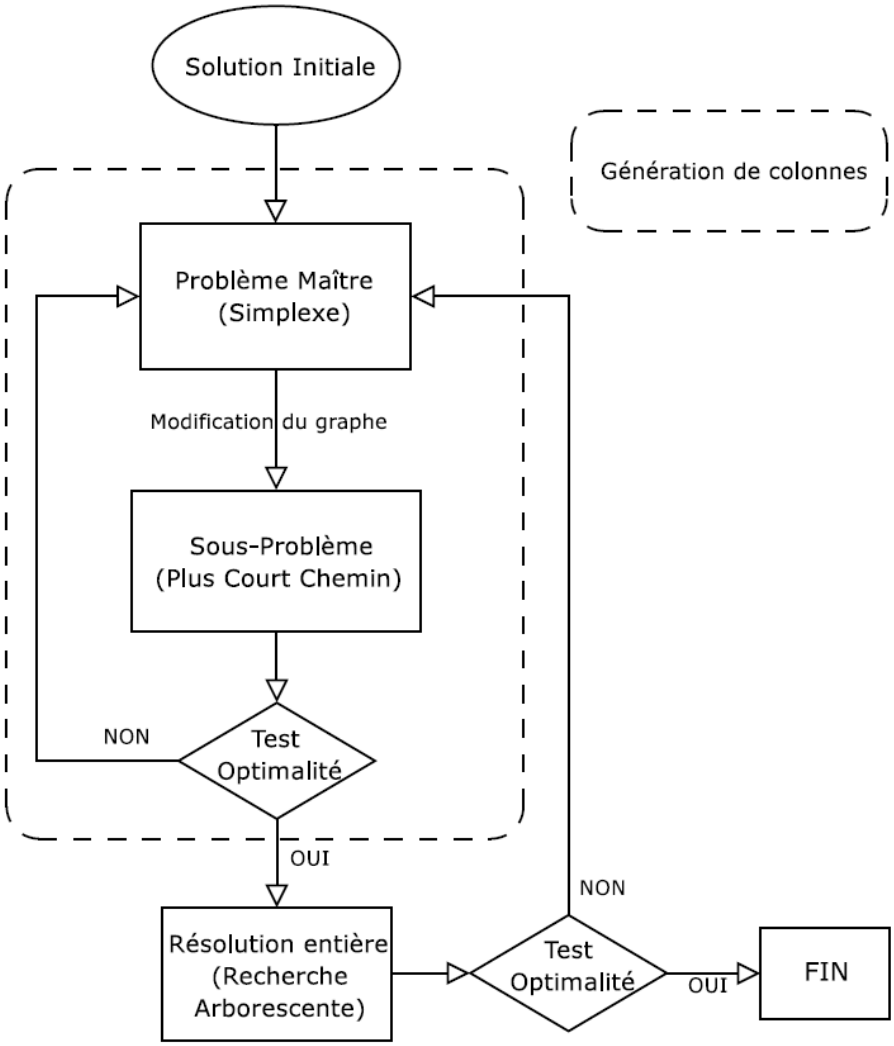


Figure 7.1 Schéma de la méthode de branch and price (Tricoire [68])

Considérons l'exemple du problème de tournées de véhicules classique et formulons le problème maître et le sous-problème associés. Soient N l'ensemble de clients, Ω un sous-ensemble de routes réalisables, c_r le coût de la route r, δ_{ir} un booléen indiquant si le client i appartient à la route r et K le nombre de véhicules. On suppose que les coûts entre les clients satisfont l'inégalité triangulaire. En posant x_r la variable booléenne indiquant si la

route r est choisie, on peut formuler le problème de tournées de véhicules classique comme le problème de partitionnement suivant :

$$\min \sum_{r \in \Omega} c_r x_r$$

sujet à :

$$\sum_{r \in \Omega} \delta_{ir} x_r \geqslant 1 \qquad \forall i \in N \tag{7.1}$$

$$\sum_{r \in \Omega} x_r \leqslant K \tag{7.2}$$

$$x_r \in \mathbb{N} \tag{7.3}$$

$$r \in \Omega \tag{7.4}$$

Les contraintes (7.1) assurent la visite de chaque client. La contrainte (7.2) veille au respect du nombre de véhicules. Dans la contrainte (7.3), on peut constater que les variables ne sont plus booléennes mais entières. En effet, comme on minimise les coûts, il n'est pas nécessaire d'imposer $x_r \in \{0; 1\}$. La relaxation linéaire de ce problème est appelée problème maître. Il est défini sur un ensemble de routes réalisables Ω obtenues en résolvant le sous-problème. Dans le sous-problème, l'objectif est d'identifier des routes réalisables de coût réduit négatif (car il s'agit d'un problème de minimisation). Soient π_i les variables duales associées aux contraintes (7.1) et ϵ la variable duale associée à la contrainte (7.2), on peut formuler la fonction objectif du sous-problème comme suit :

$$\min c_r - \sum_{i \in N} \pi_i - \epsilon$$

Les contraintes du sous-problème sont les contraintes de faisabilité d'une route du problème considéré (contraintes de degré entrant et sortant, fenêtres de temps, longueur d'une route...).

7.2 Formulation

Dans la section précédente, nous avons rappelé les principes du branch and price. Pour pouvoir appliquer cette méthodologie à notre problème, il est nécessaire de reformuler celui-ci sous forme d'un problème maître et d'un sous-problème. Dans notre problème, chaque véhicule dispose de son dépôt origine et de son dépôt destination. On aura donc un ensemble de routes réalisables par véhicule (et un sous-problème par véhicule). Dans ce qui suit, nous présenterons d'abord la formulation du problème maître puis nous donnerons une formulation du sous-problème pour un véhicule donné.

7.2.1 Problème maître

Le problème maître consiste à affecter une route réalisable à chaque véhicule en s'assurant que chaque client obligatoire est servi une fois et que chaque client optionnel est servi au plus une fois. Soient M l'ensemble des clients obligatoires, O l'ensemble des clients optionnels et Ω^k l'ensemble de routes réalisables pour le véhicule k. Notons p_r le revenu total de la route r (profit - temps de parcours modal), δ_{ir} un booléen indiquant si le client $i \in M$ appartient à la route r et δ_{cr} un booléen indiquant si le client $c \in O$ appartient à la route r. En utilisant les variables décisionnelles booléennes w_r^k indiquant si la route $r \in \Omega^k$ est utilisée par le véhicule k, on formule le problème maître comme suit :

$$\max . \sum_{k \in K} \sum_{r \in \Omega^k} p_r w_r^k$$

sujet à :

		Var. duales	
$\sum_{k \in K} \sum_{r \in \Omega^k} \delta_{ir} w_r^k = 1$	$\forall i \in M$	β_i	(7.5)
$\sum_{k \in K} \sum_{r \in \Omega^k} \delta_{cr} w_r^k \leqslant 1$	$\forall c \in O$	γ_c	(7.6)
$\sum_{r \in \Omega^k} w_r^k = 1$	$\forall k \in K$	ϵ_k	(7.7)
$w_r^k \in \{0;1\}$	$\forall k \in K, r \in \Omega^k$		

Les contraintes (7.5) assurent que chaque client obligatoire est servi une et une seule fois. Les contraintes (7.6) imposent que chaque client optionnel soit servi au plus une fois. Enfin, les contraintes (7.7) interdisent l'affectation de plusieurs routes à un même véhicule. Dans ce modèle, on associe aux contraintes (7.5) les variables duales β_i, aux contraintes (7.6) les variables duales γ_c et aux contraintes (7.7) les variables duales ϵ_k. Etant donnés les types de contraintes, les variables γ_c sont positives tandis que les variables β_i et ϵ_k sont de signe inconnu.

Dans le cadre de la génération de colonnes, on reformule les contraintes de partitionnement (pour éviter d'avoir des variables duales non bornées). Pour ce faire, on remplace les contraintes (7.5) par des contraintes de recouvrement (afin de desservir au moins une fois chaque client obligatoire) et les contraintes (7.7) du problème maître par des contraintes de *packing* (pour que nos variables primales soient bornées et pour avoir au plus une route par

véhicule). On obtient les contraintes :

$$\sum_{k \in K} \sum_{r \in \Omega^k} \delta_{ir} w_r^k \geqslant 1 \qquad \forall i \in M \tag{7.8}$$

$$\sum_{r \in \Omega^k} w_r^k \leqslant 1 \qquad \forall k \in K \tag{7.9}$$

Avec ces nouvelles contraintes, les variables duales β_i sont à présent de signe négatif et les variables duales ϵ_k de signe positif.

7.2.2 Sous-problème pour le véhicule k

Le sous-problème associé au véhicule k vise à construire des routes réalisables pour ce véhicule. Ces routes contiennent des clients obligatoires et des clients optionnels. Afin de s'assurer le respect des fenêtres de temps des clients obligatoires dans le pire des cas (quand on prend en compte uniquement les clients obligatoires) et dans le cas modal (quand on prend en compte les clients obligatoires et les clients optionnels), on prend en compte deux types de ressources temporelles : des heures de service modales et des heures de service au plus tard. Soient β_i, γ_c et ϵ_k les variables duales associées respectivement aux contraintes (7.8), (7.6) et (7.9). Soient $M' = M \cup \{o^k; d^k\}$ l'ensemble de clients obligatoires plus l'origine et la destination du véhicule, $M^o = M \cup \{o^k\}$ l'ensemble de clients obligatoires plus l'origine du véhicule, $M^d = M \cup \{d^k\}$ l'ensemble de clients obligatoires plus la destination du véhicule, $N = M \cup O$ l'ensemble des clients (obligatoires ou non) et P_{ij} l'ensemble de chemins de $i \in M$ à $j \in M$ (desservant des clients optionnels). Par exemple, $p = \{(i, k_1); (k_1, k_2); ...; (k_{n-1}, k_n); (k_n, j)\}$ et $|p| = n + 1$. Notons α la pondération des temps de parcours dans la fonction objectif, p_c le profit associé à la desserte du client optionnel $c \in O$, $\hat{\tau}_{lm}$ le temps de parcours modal de l à m et $\hat{\tau}_p$ la longueur modale du chemin $p \in P_{ij}$ (comprend les temps de service modaux et les temps de parcours modaux). On utilise les variables décisionnelles suivantes :

- x_l variable booléenne indiquant si le client l est servi
- y_{ij} variable booléenne indiquant si le client obligatoire j suit le client obligatoire i (quand on ne considère que les clients obligatoires de la route)
- z_{lm} variable booléenne indiquant si l'arc (l, m) est utilisé $(l, m \in N)$
- $\hat{t}_i$ heure de début de service modale chez le client obligatoire i
- $\bar{t}_i$ heure de début de service au plus tard chez le client obligatoire i (quand on ne considère que les clients obligatoires de la route)

Modèle

$$\max . \sum_{i \in M} (-\beta_i) x_i + \sum_{c \in O} (p_c - \gamma_c) x_c - \alpha \sum_{l \in N \cup \{o^k\}} \sum_{m \in N \cup \{d^k\}} \hat{\tau}_{lm} z_{lm} - \epsilon_k$$

sujet aux contraintes (6.1) à (6.12) (cf. chapitre 5, section 6.1)

On observe dans cette formulation que les contraintes (6.1) à (6.4) sont des contraintes de tournée tandis que les autres contraintes sont des contraintes temporelles. Ce problème peut donc être assimilé à un problème de plus court chemin élémentaire avec contraintes de ressources. Afin de retrouver une formulation similaire à celle du problème du plus court chemin, reformulons la fonction objectif du sous-problème. Dans un premier temps, reformulons la fonction objectif sous forme d'un problème de minimisation :

$$\min . \sum_{i \in M} \beta_i x_i + \sum_{c \in O} (\gamma_c - p_c) x_c + \alpha \sum_{l \in N \cup \{o^k\}} \sum_{m \in N \cup \{d^k\}} \hat{\tau}_{lm} z_{lm} + \epsilon_k$$

En utilisant la contrainte (6.3), on a :

$$\sum_{i \in M} \beta_i x_i = \sum_{i \in M} \sum_{j \in N \cup \{d^k\}} \beta_i z_{ij}$$
$$\sum_{c \in O} (\gamma_c - p_c) x_c = \sum_{c \in O} \sum_{j \in N \cup \{d^k\}} (\gamma_c - p_c) z_{cj}$$

De même, en utilisant la contrainte (6.1), on a :

$$\epsilon_k = \sum_{m \in N \cup \{d^k\}} \epsilon_k z_{o^k m}$$

On peut donc reformuler la fonction objectif comme suit :

$$\min . \sum_{i \in M} \sum_{j \in N \cup \{d^k\}} \beta_i z_{ij} + \sum_{c \in O} \sum_{j \in N \cup \{d^k\}} (\gamma_c - p_c) z_{cj} + \alpha \sum_{l \in N \cup \{o^k\}} \sum_{m \in N \cup \{d^k\}} \hat{\tau}_{lm} z_{lm} + \sum_{m \in N \cup \{d^k\}} \epsilon_k z_{o^k m}$$

En regroupant les variables z_{lm} suivant que l est un client obligatoire, optionnel ou bien le dépôt origine, on obtient :

$$\min . \sum_{i \in M} \sum_{j \in N \cup \{d^k\}} (\alpha \hat{\tau}_{ij} + \beta_i) z_{ij} + \sum_{c \in O} \sum_{j \in N \cup \{d^k\}} (\alpha \hat{\tau}_{cj} + \gamma_c - p_c) z_{cj} + \sum_{m \in N \cup \{d^k\}} (\alpha \hat{\tau}_{o^k m} + \epsilon_k) z_{o^k m}$$

Avec cette fonction objectif du type $\min \sum_{i,j \in N} c_{ij} x_{ij}$, on retrouve bien une formulation similaire à celle d'un problème de plus court chemin. Les coûts modifiés c_{ij} associés aux arcs (i, j) sont répertoriés en trois catégories suivant le type du client i :

– Soit i est un client obligatoire ($i \in M$) et $c_{ij} = \alpha\hat{\tau}_{ij} + \beta_i$.
– Soit i est un client optionnel ($i \in O$) et $c_{ij} = \alpha\hat{\tau}_{ij} - p_i + \gamma_i$.
– Soit i est le dépôt origine du véhicule considéré ($i = o^k$) et $c_{ij} = \alpha\hat{\tau}_{ij} + \epsilon_k$.

Avec ces coûts modifiés, on retrouve bien une formulation des sous-problèmes de type ESPPRC (elementary shortest path problem with resource constraints) où les ressources correspondent à un temps modal et un temps maximal consommé.

7.3 Composantes de l'algorithme de branch and price

Dans l'algorithme de branch and price, trois composantes majeures peuvent impacter l'efficacité de la méthode. Il s'agit de la construction de la solution initiale, de la méthode de résolution du sous-problème et de la stratégie de branchement. Nous détaillerons donc ces trois composantes dans cette section.

7.3.1 Construction de la solution initiale

L'algorithme de branch and price est basé sur la résolution du problème sur un sous-ensemble de routes de bonne qualité puis sur l'ajout de routes au fur et à mesure. Etant donné que, dans notre problème, nous devons assurer la couverture des clients obligatoires sans desservir plus d'une fois chaque client optionnel, il nous faut un ensemble de routes initial de bonne qualité mais aussi et surtout diversifié. Nous choisissons donc d'utiliser l'heuristique basée sur la génération de colonnes (variante (BD, DI, NV)) décrite au chapitre précédent pour construire la solution initiale.
Pour les instances de taille 40 et plus, l'ensemble de colonnes initiales par véhicule ainsi généré étant très grand, nous limitons cet ensemble, pour chaque véhicule, aux colonnes avec un écart de profit de moins de Δp_{max} de la meilleure route. En effet, le profit associé à une route étant principalement guidé par le profit associé aux clients optionnels et obligatoires, on choisit de se limiter aux routes ayant un écart de profit de moins de Δp_{max} et non un gap en %.

7.3.2 Résolution du sous-problème

Dans la section précédente, nous avons ramené la formulation des sous-problèmes à des ESPPRC. Nous avons présenté, au chapitre précédent, un état de l'art des méthodes de résolution de l'ESPPRC (cf. section 6.1.1). Dans cette section, nous ferons un bref état de l'art sur l'intégration des méthodes de résolution de l'ESPPRC dans la génération de colonnes puis nous justifierons nos choix pour la résolution du sous-problème.

Etat de l'art des méthodes de résolution de l'ESPPRC dans la génération de colonnes

Dans la génération de colonnes, les algorithmes exacts de programmation dynamique peuvent s'avérer très chronophages, d'autant plus qu'ils sont utilisés à maintes reprises pour générer de nouvelles colonnes. Desaulniers *et al.* [22] proposent donc une stratégie heuristique pour générer de nouvelles colonnes sans avoir à résoudre un ESPPRC. Cette stratégie est une méthode de recherche taboue basée sur l'insertion et la suppression d'un client dans une route. Pour générer de nouvelles colonnes de coût réduit négatif, ils appliquent un algorithme de recherche taboue aux routes de la solution courante du problème maître .
Comme nous l'avons mentionné à plusieurs reprises, la résolution exacte d'un ESPPRC nécessite des temps de calcul importants. Il faut donc éviter le plus possible d'y avoir recours. Souvent, la génération de colonnes se décompose en deux phases : une première phase où l'on génère des colonnes avec une méthode approchée puis une deuxième phase où les colonnes sont générées de façon exacte (afin de garantir l'optimalité de la solution). Les méthodes approchées utilisées dans cette approche en deux phases sont la LDS (Boussier *et al.* [11] et Tricoire [68]) et la programmation dynamique avec dominance aggrégée (Jepsen *et al.* [38], Bettinelli *et al.* [7] et Dayarian *et al.* [20]). Les méthodes exactes utilisées dans la deuxième phase sont l'algorithme de Feillet *et al.* [26] (Boussier *et al.* [11], Tricoire [68] et Jepsen *et al.* [38]), l'algorithme de programmation dynamique bidirectionnelle bornée de Salani (Bettinelli *et al.* [7]) et la ng-DSSR route relaxation (Dayarian *et al.* [20]). Desaulniers *et al.* [22] et Gauvin *et al.* [29] proposent de prendre en compte plusieurs méthodes approchées. Ainsi, à chaque itération de l'algorithme de génération de colonnes, Desaulniers *et al.* [22] appliquent les méthodes suivantes l'une après l'autre, par ordre de rapidité : la méthode de recherche taboue puis l'algorithme de limited discrepancy search puis la méthode de dominance aggrégée et enfin la programmation dynamique exacte. Dès que l'une d'elles génère des colonnes de coût réduit négatif, ils passent à l'itération suivante. Si aucune ne génère de colonnes, la solution est optimale. Gauvin *et al.* [29] procèdent de même avec la méthode de recherche taboue, puis un algorithme de programmation dynamique bidirectionnelle bornée avec ng-route relaxation et dominance aggrégée.

Méthodes choisies pour la résolution du sous-problème

Comme nous l'avons mentionné au chapitre précédent, la relaxation de la contrainte d'élémentarité lors de la résolution du sous-problème ne nous aide pas, car les contraintes portant sur les clients ne sont pas assez fortes. Nous ne retiendrons donc pas les méthodes de type

NG-route ou DSSR. Nous choisirons comme méthode exacte la variante exacte (BD, DI, VV) de programmation dynamique bidirectionnelle bornée (cf. chapitre précédent). En ce qui concerne les méthodes approchées pour générer de nouvelles colonnes, nous disposons déjà d'une méthode très efficace présentée au chapitre précédent : la variante (BD, DI, NV) de programmation dynamique bidirectionnelle bornée. Cette méthode peut être assimilée à une méthode de type dominance aggrégée car on relâche la ressource vecteur de visites dans les tests de dominance. Comme cette méthode permet de générer un nombre important de routes rapidement, nous n'utiliserons pas de recherche taboue. Par contre, nous proposons également d'intégrer une méthode approchée de type *limited discrepancy search* lors de la génération de colonnes. Rappelons que cette méthode consiste à ne garder, pour chaque sommet, que les D_{max} plus proches prédécesseurs et successeurs (au vu des coûts réduits) dans le graphe du sous-problème. Or, dans notre problème, les clients ne sont pas très contraints au niveau des fenêtres de temps, ce qui rend le graphe initial complet (à quelques arcs près). Le fait d'identifier, à chaque itération, les D_{max} plus proches voisins dans le graphe est donc très coûteux en temps de calcul (pour chaque client, tri des voisins puis élimination des mauvais arcs). Pour éviter cela, on propose de construire un graphe restreint a priori en gardant pour chaque sommet ses D_{max} plus proches voisins (en terme de distance/temps de parcours). Une fois ce graphe construit, on associe à chaque itération une valeur aux arcs de ce graphe correspondant aux coûts modifiés de l'itération et on résout un ESPPRC sur ce graphe. Nous proposons deux heuristiques de type limited discrepancy search : la première consiste à résoudre avec la variante approchée (BD, DI, NV) l'ESPPRC sur le graphe restreint et la deuxième consiste à résoudre exactement l'ESPPRC sur ce même graphe (variante (BD, DI, VV)).

Nous disposons ainsi de quatre méthodes pour générer de nouvelles colonnes. Les trois premières sont des méthodes approchées : la variante approchée (BD, DI, NV), la variante approchée (BD, DI, NV) avec discrepancy search, la variante exacte (BD, DI, VV) avec discrepancy search. Quant à la dernière, il s'agit d'une méthode exacte : variante (BD, DI, VV) de l'algorithme de programmation dynamique. Dans l'algorithme de génération de colonnes, on distingue deux phases : une phase dans laquelle les nouvelles colonnes sont générées à l'aide d'heuristiques suivie d'une phase dans laquelle les colonnes sont générées de façon exacte. Dans la première phase (génération heuristique de nouvelles colonnes), comme l'ont proposé Desaulniers *et al.* [22] et Gauvin *et al.* [29], à chaque itération, nous appliquons les méthodes approchées par ordre de rapidité décroissante. Nous utilisons d'abord la variante approchée (BD, DI, NV) avec discrepancy. Si celle-ci ne permet pas de générer de colonnes, nous utilisons la variante approchée (BD, DI, NV) simple. Si aucune colonne n'a pu être générée jusqu'ici, on applique la variante exacte (BD, DI, VV) avec discrepancy search. Enfin,

dans la deuxième phase de la génération de colonnes, on procède à la génération exacte de nouvelles colonnes à l'aide de la variante (BD, DI, VV). Aussi, dans ces deux phases, comme le nombre de colonnes générées par nos méthodes est élevé, on introduit un paramètre N_{max} et, à chaque itération de la génération de colonnes, on ajoute seulement les N_{max} meilleures colonnes pour chaque véhicule.

7.3.3 Stratégie de branchement

Comme nous l'avons mentionné précédemment, après résolution de la génération de colonnes, on peut obtenir une solution fractionnaire. Il est donc nécessaire d'intégrer la génération de colonnes dans une méthode de recherche arborescente de type branch and price (branch and bound avec possibilité d'ajouter de nouvelles colonnes à chaque noeud).

Etat de l'art des stratégies de branchement

Dans le problème de tournées de véhicules classique, la stratégie de branchement la plus fréquemment utilisée ([28], [68], [22], [7], [55] et [29]) est le branchement sur les variables de flot (si on note x_{ij}^k la variable indiquant le flot traversant l'arc (i, j) pour le véhicule k, cela revient à brancher sur $\sum_k x_{ij}^k$). Toutefois, si cette stratégie est souvent utilisée, elle n'est pas très efficace quand le graphe est dense.
Une autre stratégie, introduite par Augerat [4], reprise par Jepsen *et al.* [38] et Gauvin *et al.* [29], consiste à brancher sur le nombre d'arcs adjacents à un ensemble de clients. Il s'agit d'identifier un ensemble de clients pour lequel le nombre d'arcs adjacents est dans l'intervalle $]2; 4[$ et à forcer d'un côté ce nombre à être égal à 2, et de l'autre côté, à être égal à 4.
Dans le cadre du problème de Team Orienteering, Boussier *et al.* [11] proposent de brancher sur le nombre de visites d'un client. Cette stratégie de branchement est spécifique au problème de team orienteering où les clients peuvent être desservis une fois ou ne pas être desservis.
Pour son problème de tournées de véhicules multi-dépôt avec fenêtres de temps, Dayarian *et al.* [20] proposent deux stratégies de branchement. La première est un branchement sur les fenêtres de temps introduit par Gélinas *et al.* [30] consistant à diviser la fenêtre de temps en deux intervalles correspondant aux noeuds fils. La deuxième stratégie consiste à brancher sur l'affectation des usines aux producteurs (si le flot traversant un arc producteur-usine est fractionnaire, on branche).

Stratégie de branchement choisie

Dans notre problème, les clients (optionnels ou obligatoires) étant faiblement contraints au niveau des fenêtres de temps, le graphe support est donc très dense. Les stratégies de

branchement consistant à brancher sur les variables de flot ou sur les fenêtres de temps ne sont donc pas judicieuses. Aussi, comme notre problème comporte des clients optionnels qui sont desservis au plus une fois, nous choisissons de nous inspirer de la stratégie de branchement proposée par Boussier *et al.* [11]. Toutefois, le nombre de visites d'un client (obligatoire ou optionnel) étant souvent entier tandis que le nombre de visites d'un client par un véhicule est décimal, nous choisissons de brancher sur le nombre de visites d'un client par un véhicule. Nous pouvons donc brancher aussi bien sur le nombre de visites d'un client optionnel que sur celui d'un client obligatoire.

Pour choisir sur quel couple (client, véhicule) brancher, nous proposons deux méthodes. La première méthode procède par véhicule et s'arrête dès qu'un couple (client,véhicule) est identifié. Pour chaque véhicule, on recherche d'abord parmi les clients obligatoires le couple (client, véhicule) avec le nombre de visites le plus fractionnaire. Si aucun couple n'a pu être identifié pour le branchement, on procède à cette recherche parmi les clients optionnels. Si aucun couple n'a été identifié pour le branchement, on passe au véhicule suivant. Dans la deuxième méthode, on procède par catégorie de clients. On recherche d'abord parmi les clients obligatoires et l'ensemble des véhicules, le couple (client, véhicule) avec le nombre de visites le plus fractionnaire. Si aucun couple n'a pu être identifié, on procède de même au niveau des clients optionnels.

Notre stratégie de branchement consiste, dans la première branche, à forcer la desserte du client par le véhicule concerné et, dans la deuxième branche, à interdire la desserte du client par le véhicule concerné.
Pour forcer la desserte du client i par le véhicule k au niveau du problème maître, on rajoute une contrainte du type $\sum_{r\in\Omega^k} \delta_{ir} x_r^k \geqslant 1$. On choisit volontairement une contrainte de type recouvrement plutôt qu'une contrainte de partitionnement afin de générer une variable duale de signe connu. Cette contrainte génère une variable duale et donc une modification du coût réduit au niveau du sous-problème correspondant, mais ne garantit pas la génération de routes contenant le client i pour le véhicule k. C'est pourquoi, afin de garantir la génération de routes contenant le client i, on modifie les coûts réduits associés aux arcs sortants de ce client en posant pour tout j, $\hat{c}_{ij} = \hat{c}_{ij} - M$ avec M suffisamment grand. On veut s'assurer que les clients optionnels forcés deviennent plus importants que les clients obligatoires.
Pour interdire la desserte du client i par le véhicule k au niveau du problème maître, on rajoute une contrainte du type $\sum_{r\in\Omega^k} \delta_{ir} x_r^k \leqslant 0$. Comme ci-dessus, la contrainte génère une modification des coûts réduits au niveau du sous-problème concerné sans garantir la génération de routes ne contenant pas le client i. On interdit donc la desserte du client i au niveau du

sous-problème en associant à tous les arcs ayant pour extrémité le client i un coût infini. Pour l'exploration de l'arbre de branchement, on parcourt l'arbre en profondeur en commençant par le côté où les clients sont forcés (afin d'obtenir rapidement de bonnes bornes pour élaguer la suite de l'arbre).

Etant donnée notre stratégie de branchement consistant à interdire/forcer la desserte d'un client par un véhicule, on peut se retrouver avec un squelette de clients obligatoires non réalisable. En effet, quand on interdit la desserte d'un client obligatoire par un véhicule, on impose implicitement la desserte de ce client obligatoire par un autre véhicule. Inversement, quand on impose la desserte d'un client obligatoire par un véhicule, on empêche potentiellement la desserte de certains clients obligatoires par ce véhicule et donc on impose la desserte de ces clients obligatoires par d'autres véhicules. Dans les deux cas, on impose la desserte d'un ensemble de clients obligatoires par un ensemble de véhicules sans vérifier si le squelette reste réalisable avec ces nouvelles contraintes. On peut donc se retrouver avec un squelette de clients obligatoires non réalisable. Pour éviter cet écueil, on vérifie après le branchement si le squelette est réalisable. Si ce n'est pas le cas, on passe directement au noeud suivant. Par contre, lors du branchement sur la desserte d'un client optionnel, d'après notre stratégie d'identification du couple sur lequel brancher, cela signifie que les clients obligatoires sont tous desservis une fois par un véhicule (sinon, on aurait branché sur un client obligatoire) et donc que le squelette est fixé. Dans ce cas, il n'y a donc pas de problème de réalisabilité du squelette.

7.4 Expérimentation

Nous procédons aux expérimentations sur les instances décrites chapitre 1, en utilisant Cplex 12.4, pour procéder à la résolution du problème maître. Les tests ont été effectués sur une machine avec 4CPU, 2.8GHz et 30Go de RAM.

7.4.1 Réglage des paramètres

Concernant le réglage des paramètres, nous utilisons le même réglage de paramètres que dans les chapitres précédents : horizon de temps $T = 480$ minutes, vitesse minimale $v_{min} = 20km/h$ et vitesse modale $v_{mod} = 40km/h$. Après avoir converti ces valeurs v_{min} et v_{mod} en unités arbitraires par minute, on calcule le temps de parcours unitaire minimal $\underline{\delta} = \lceil 100/v_{max} \rceil$ et modal $\hat{\delta} = \lceil 100/v_{mod} \rceil$. Les temps de parcours minimaux et modaux sont ensuite obtenus en utilisant les formules $\underline{\tau}_{ij} = \left\lceil D_{ij}\underline{\delta} \right\rceil$ et $\hat{\tau}_{ij} = \left\lceil D_{ij}\hat{\delta} \right\rceil$. Pour le réglage des temps de service, nous avons choisi des temps de service minimaux et modaux respective-

ment de 15 et 22 minutes pour les clients optionnels et de 30 et 35 minutes pour les clients obligatoires. Comme précédemment, nous avons supposé que le service d'un client optionnel, quel qu'il soit, génère un profit $p_o = 100$ et nous avons choisi $\alpha = 1$. Pour le profit associé aux clients obligatoires p_m, nous avons repris la valeur choisie à la section 5.2 du chapitre précédent, à savoir $p_m = 300$. Pour la génération des colonnes initiales sur les instances de taille supérieure à 40, comme nous l'avons mentionné section 7.3.1, nous nous limitons, pour un véhicule donné, aux routes présentant un écart de profit de moins de Δp_{max} de la meilleure route. Afin d'obtenir des ensembles de routes de taille comparable à ceux obtenus sur les instances à 30 clients, on choisit $\Delta p_{max} = 500$. A chaque itération de la génération de colonnes, comme nous l'avons mentionné section 7.3.2, nous n'ajoutons pas toutes les colonnes générées mais seulement N_{max} colonnes par véhicule. Pour ce paramètre, on a choisi $N_{max} = 30$ (suite à des tests préliminaires). Finalement, pour la méthode de type *limited discrepancy search*, le graphe initial étant quasiment complet, nous avons choisi de nous limiter aux $D_{max} = 10$ plus proches voisins.

7.4.2 Calibrage de la méthode

Avant de présenter les résultats obtenus avec la méthode de ce chapitre, nous devons choisir la méthode de résolution du sous-problème. Dans la section 7.3.2, nous avons proposé plusieurs méthodes pour générer de nouvelles colonnes : la variante heuristique (BD, DI, NV) sur le graphe restreint (que l'on notera HeurDisc), la variante approchée (BD, DI, NV) (que l'on notera HeurSimple), la variante exacte (BD, DI, VV) sur le graphe restreint (que l'on notera ExactDisc) et la variante exacte (BD, DI, VV) (que l'on notera Exact). Ces méthodes étant classées par ordre de rapidité décroissantes, nous les appliquons dans cet ordre. Nous avons testé sur les instances à 30 clients différentes variantes : la variante (HeurDisc, HeurSimple, ExactDisc, Exact), la variante (HeurSimple, ExactDisc, Exact) et la variante (HeurSimple, Exact). Lors de ces tests, nous avons pu observer que la méthode HeurDisc entraînait des temps de calcul plus élevés et augmentait le nombre d'itérations nécessaires pour atteindre l'optimalité. Nous n'avons donc pas retenu cette variante dans nos tableaux de résultats. Lors des tests des autres variantes, nous avons observé le célèbre « Tailing effect » de la génération de colonnes. Nous avons donc décidé de proposer une méthodologie pour y remédier : au bout de I_{max} itérations heuristiques de la génération de colonnes durant lesquelles la solution ne change pas, on passe à la génération exacte des routes. On propose donc deux variantes supplémentaires, à savoir : la variante (HeurSimple, ExactDisc, Exact, I_{max}) et la variante (HeurSimple, Exact, I_{max}). Dans le tableau 7.1, nous reportons donc les résultats obtenus avec les quatre variantes mentionnées ci-dessus en posant $I_{max} = 50$ (au vu des résultats obtenus sans I_{max}). Les en-têtes de colonnes sont les suivants : CPU :

temps de calcul en minutes ; # ité. à opt. : nombre d'itérations avant obtention de la solution optimale ; # ité. heur. : nombre d'itérations heuristiques ; # ité. exactes : nombre d'itérations exactes. Comme précédemment, dans ce tableau, nous limitons les temps de calcul à 4 heures (si les temps de calcul dépassent cette durée, on inscrit un « - » dans la case correspondante). Aussi, pour les instances sur lesquelles un branchement est nécessaire, on n'indique pas le nombre d'itérations avant optimalité (on met un « - » dans la case correspondante). L'introduction du paramètre $I_{max} = 50$ ne modifiant pas le nombre d'itérations nécessaires pour atteindre l'optimalité (qui est d'au plus 51), nous n'indiquerons pas cette valeur pour les variantes (HeurSimple, ExactDisc, Exact, I_{max}) et (HeurSimple, Exact, I_{max}).

Tableau 7.1 Comparaison des méthodes de résolution de l'ESPPRC

# de clients oblig.	Instances	HeurSimple, Exact				HeurSimple, ExactLDS, Exact				HeurSimple, ExactLDS, Exact, $I_{max} = 50$			HeurSimple, Exact, $I_{max} = 50$		
		CPU (min)	# d'ité. à opt	# d'ité. heur	# d'ité. exactes	CPU (min)	# d'ité. à opt	# d'ité. heur	# d'ité. exactes	CPU (min)	# d'ité. heur	# d'ité. exactes	CPU (min)	# d'ité. heur	# d'ité. exactes
5	C1_1_5	16	51	49	18	11	49	65	1	11	65	1	16	49	18
	C1_2_5	66	25	23	32	87	23	114	19	82	75	33	66	23	32
	C1_3_5	27	0	80	61	46	0	173	27	24	50	33	24	50	33
	C1_4_5	137	-	169	69	156	-	321	33	181	287	106	150	163	120
	C1_5_5	49	52	80	72	33	51	122	17	34	103	29	43	79	71
6	C1_1_6	25	11	42	32	25	10	108	1	17	62	2	22	41	31
	C1_2_6	83	-	76	73	128	-	237	55	136	165	109	83	76	73
	C1_3_6	139	7	177	210	100	6	412	25	30	58	27	27	58	27
	C1_4_6	231	-	189	173	226	-	382	72	201	360	64	235	189	174
	C1_5_6	28	15	13	47	31	13	125	1	31	65	34	28	13	47
7	C1_1_7	8	13	34	3	15	12	58	5	15	58	5	8	34	3
	C1_2_7	22	7	39	15	39	6	123	13	22	58	7	22	39	15
	C1_3_7	73	2	156	71	84	2	306	36	43	53	55	40	53	55
	C1_4_7	239	-	145	110	224	-	348	57	224	348	57	239	145	110
	C1_5_7	32	0	72	33	37	0	159	1	27	50	35	26	50	35
8	C1_1_8	42	-	48	66	49	-	93	55	49	93	55	42	48	66
	C1_2_8	17	16	51	4	31	15	109	2	40	67	28	17	51	4
	C1_3_8	39	20	95	41	59	19	268	1	54	71	98	51	71	98
	C1_4_8			-				-			-			-	
	C1_5_8	25	15	43	29	25	14	99	1	18	66	1	23	42	28
9	C1_1_9	17	16	68	15	21	15	80	26	17	67	14	15	67	14
	C1_2_9	42	-	44	56	61	-	156	18	60	156	18	42	44	56
	C1_3_9	16	24	48	13	18	23	53	16	18	53	16	16	48	13
	C1_4_9	13	22	20	31	26	20	109	14	16	72	5	13	20	31
	C1_5_9	63	-	67	85	75	-	210	33	72	157	56	63	67	85

Dans le tableau 7.1, on observe que les meilleurs temps de calcul sont obtenus avec la variante (HeurSimple, Exact, $I_{max} = 50$). En particulier, sur l'instance C1_3_6(30), cette variante permet de diviser les temps de calcul par 4. On constate également que le nombre d'itérations nécessaire pour atteindre la solution optimale reste similaire (diminution de 0 à 2 itérations) dans la variante avec ExactLDS. Aussi, on remarque dans la variante (HeurSimple, ExactLDS, Exact) que la solution optimale est obtenue durant les itérations heuristiques et que le nombre d'itérations exactes est souvent nettement plus faible dans cette variante. Pour la suite des expérimentations, nous utiliserons la variante (HeurSimple, Exact, $I_{max} = 50$) pour ses temps de calcul.

Une autre composante importante de l'algorithme de branch and price est la stratégie d'identification du couple (client,véhicule) sur lequel brancher. Nous en avions proposé deux (cf. section 7.3.3 : on notera la première (par véhicule) $V1$ et la deuxième (par catégories de clients) $V2$. Les résultats obtenus sur les instances à 40 clients sont synthétisés dans le tableau 7.2. Dans ce tableau, on reporte pour chaque instance les temps de calcul en minutes (colonne CPU). On limite les temps de calcul à 2 heures et on indique la meilleure solution trouvée dans le temps imparti.

Tableau 7.2 Comparaison des stratégies d'identification pour le branchement

	Branchement V1			Branchement V2		
	CPU	Non visités	Distance	CPU	Non visités	Distance
C1_1_5	24	0	245,45	23	0	245,447
C1_2_5	43	0	226,78	48	0	226,78
C1_3_5	34	0	241,33	120	2	226,454
C1_4_5	120	0	249,52	59	0	249,52
C1_5_5	5	0	220,26	6	0	220,263
C1_1_6	29	0	249,99	20	0	249,991
C1_2_6	120	0	250,82	120	1	265,534
C1_3_6	70	0	253,04	120	1	266,634
C1_4_6	103	0	243,19	120	1	252,538
C1_5_6	17	0	221,96	17	0	221,961
C1_1_7	4	0	254,78	4	0	254,781
C1_2_7	42	0	235,10	40	0	235,1
C1_3_7	120	1	271,14	120	1	253,496
C1_4_7	37	0	256,07	23	0	256,002
C1_5_7	12	0	230,89	12	0	230,894
C1_1_8	120	1	222,79	120	1	230,149
C1_2_8	68	0	243,44	113	0	246,748
C1_3_8	5	0	250,09	5	0	250,086
C1_4_8	72	0	251,89	120	0	239,686
C1_5_8	17	0	215,16	18	0	215,157
C1_1_9	120	1	237,41	120	1	244,444
C1_2_9	120	2	244,21	120	1	235,903
C1_3_9	120	2	253,90	52	1	245,891
C1_4_9	114	1	228,60	120	1	228,6
C1_5_9	9	0	214,75	9	0	214,745

Dans le tableau 7.2, on observe que la stratégie V1 permet de n'avoir que 7 instances nécessitant 2 heures de calcul tandis que la stratégie V2 présente 10 instances nécessitant 2

heures de calcul. De plus, dans l'ensemble le nombre de clients non servis dans la stratégie V2 est souvent plus élevé que dans la stratégie V1. Nous choisissons donc, pour la suite des expérimentations, d'identifier sur quel couple (client,véhicule) brancher à l'aide de la stratégie V1 (procédant par véhicule).

7.4.3 Résultats de la méthode

Après avoir calibré notre méthode (cf. ci-dessus), nous procédons à présent aux tests de l'algorithme de branch and price sur les instances du chapitre 1. Pour les instances contenant 40 clients et plus, une itération exacte de génération de colonnes prend plus d'une heure, nous générons donc uniquement des colonnes à l'aide de méthodes approchées (HeurSimple, $I_{max} = 50$). Les solutions obtenues sur ces instances ne sont donc pas optimales. Les résultats obtenus avant et après simulation sur les instances à 30, 40 et 50 clients sont synthétisés dans le tableau 7.3. Les temps de calcul indiqués dans ce tableau sont en minutes. On indique en gras les valeurs correspondant à la solution optimale (instances à 30 clients). Les temps de calcul sont limités à 4 heures pour les instances à 30 clients (résolution exacte) et à 2 heures pour les autres instances (résolution approchée).

Tableau 7.3 Résultats avant/après simulation pour l'algorithme de branch and price

Nb de clients	Nb de clients oblig.	Avant simulation			Après simulation					
		CPU	# moy. non servis	distance moy.	# moy. non servis		distance moy.		retard moy.	
					WR	OS	WR	OS	WR	OS
30 clients	5	**60**	**0,0**	**196**	2,3	1,9	187	189	0,2	5,8
	6	**79**	**0,0**	**199**	4,2	2,5	181	190	1,2	36,3
	7	**67**	**0,0**	**202**	4,3	2,4	183	193	1,6	25,8
	8	**33**	**0,0**	**201**	4,7	3,7	180	188	1,8	30,8
	9	**30**	**0,0**	**205**	4,9	3,0	184	198	3,6	51,0
40 clients	5	45	0,0	237	7,7	6,3	202	216	1,3	47,4
	6	68	0,0	244	8,7	7,0	196	211	1,6	44,0
	7	43	0,2	250	10,8	7,8	207	226	3,1	80,0
	8	56	0,2	237	10,5	6,7	194	216	2,7	116,7
	9	97	1,2	236	11,8	8,0	201	222	7,1	101,5
50 clients	5	120	7,2	215	16,0	14,6	191	200	1,6	49,1
	6	97	7,4	218	17,5	15,2	192	200	1,1	43,7
	7	120	8,6	218	19,0	15,8	192	204	3,2	68,4
	8	97	8,2	212	19,2	16,6	182	198	2,8	64,4
	9	98	8,8	213	20,8	16,1	186	202	3,9	100,3

Dans le tableau 7.3, on observe, comme au chapitre précédent, la cohérence des résultats obtenus : le nombre de clients devenus non desservis durant la simulation n'est pas très élevé. En comparant les stratégies de programmation dynamique, on observe, comme au chapitre précédent, que la stratégie considérant un seul segment est préférable pour le nombre de clients desservis et que la stratégie considérant toute la route est préférable pour le retard et la distance parcourue. Les différences au niveau du retard sont vraiment importantes, avec

un retard pouvant aller jusqu'à 2 heures dans la stratégie considérant un seul segment (contre seulement 7 minutes avec l'autre stratégie). On constate également dans ce tableau que l'on nécessite une heure de calcul en moyenne pour obtenir la solution optimale sur les instances à 30 clients.

7.4.4 Comparaison de cette méthode avec les méthodes précédentes

Après avoir observé les résultats obtenus avec la méthode de ce chapitre, comparons cette méthode avec les deux méthodes proposées précédemment (méthode approchée basée sur la priorité et méthode approchée basée sur la génération de colonnes). Les résultats avant simulation (30, 40 et 50 clients) obtenus avec les trois méthodes sont synthétisés dans le Tableau 7.4. On indique en gras les résultats correspondant aux solutions optimales (obtenus avec l'algorithme de branch and price sur les instances à 30 clients).

Tableau 7.4 Comparaison des 3 méthodes avant simulation

# clients	# clients obligatoires	Heuristique basée sur la priorité			Heuristique basée sur la génération de colonnes			Branch and price		
		CPU (s)	non servis	Distance (km)	CPU (s)	non servis	Distance (km)	CPU (s)	non servis	Distance (km)
30	5	43	1,4	220	22	0,0	200	**3588**	**0,0**	**196**
	6	66	2,8	232	37	0,0	214	**4740**	**0,0**	**199**
	7	53	2,4	230	34	0,0	219	**4020**	**0,0**	**202**
	8	34	3,4	216	25	0,0	214	**1995**	**0,0**	**201**
	9	24	3	217	29	0,0	217	**1788**	**0,0**	**205**
40	5	3277	7,2	202	97	1,4	238	2652	0,0	237
	6	962	10,2	217	120	1,8	250	3828	0,0	244
	7	1451	10	211	122	2,2	254	2580	0,2	250
	8	1880	10	210	98	2,0	242	3408	0,2	237
	9	1053	10	206	120	2,4	240	5325	1,2	236
50	5	1227	15,4	192	184	8,0	226	7200	7,2	215
	6	3064	17,6	197	212	8,6	222	5832	7,4	218
	7	2844	18	192	240	9,2	221	7200	8,6	218
	8	2323	18,6	200	218	9,2	215	5568	8,2	212
	9	1408	19,2	188	261	9,4	221	5856	8,8	213

Dans le tableau 7.4, on constate que les temps de résolution sont nettement plus élevés dans l'algorithme de branch and price. Etant donné le nombre très élevé de clients non desservis avant simulation dans l'heuristique basée sur la priorité des clients (rappelons que l'on s'assure dans cette méthode que les routes soient réalisables dans 90% des cas), nous ne retiendrons pas cette méthode dans la comparaison des résultats. En comparant les résultats obtenus avec l'heuristique basée sur la génération de colonnes et ceux obtenus avec l'algorithme de branch and price, on observe une nette diminution du nombre de clients non desservis dans l'algorithme de branch and price (sur les instances à 40 et 50 clients) accompagnée d'une diminution de la distance totale parcourue (sur toutes les instances). Afin de juger de la différence de qualité des solutions obtenues, il faut comparer la valeur des solutions obtenues dans les deux méthodes (où la valeur d'une solution correspond à la valeur de la

fonction objectif, à savoir, le profit des clients optionnels visités auquel on soustrait les temps de parcours modaux). Avec ces valeurs, on peut calculer le gap à optimalité sur les instances à 30 clients et le gap séparant les solutions des deux premières méthodes des solutions de l'algorithme de branch and price sur les autres instances. Ces valeurs et gaps sont reportés dans le tableau 7.5.

Tableau 7.5 Branch and price versus heuristique basée sur la génération de colonnes

# clients	# clients obligatoires	Branch and price	Heuristique basée sur la génération de colonnes	
		Valeur	Valeur	Gap
30 clients	5	**2149**	2143	0,3%
	6	**2044**	2018	1,3%
	7	**1938**	1908	1,5%
	8	**1840**	1817	1,3%
	9	**1734**	1714	1,2%
40 clients	5	3074	2931	4,7%
	6	2969	2773	6,6%
	7	2833	2625	7,3%
	8	2755	2566	6,9%
	9	2557	2431	4,8%
50 clients	5	3389	3292	3,5%
	6	3264	3139	3,8%
	7	3045	2980	2,1%
	8	2997	2891	3,5%
	9	2834	2761	2,5%

Dans le tableau 7.5, on constate que les solutions obtenues avec l'heuristique basée sur la génération de colonnes sont de très bonne qualité puisqu'elles sont situées à moins de 1.5% des solutions optimales en moyenne. Par contre, sur les instances de plus grande taille, les gaps augmentent et peuvent atteindre les 7% en moyenne, alors qu'il ne s'agit pas de gap à optimalité. Ces gaps sont plus importants sur ces instances car le nombre de clients non desservis est plus important dans l'heuristique basée sur la génération de colonnes (impactant fortement la valeur de la solution car chaque client non desservi génère une perte de 100 dans la valeur de la solution). Ainsi, même si l'algorithme de branch and price ne permet pas d'atteindre la solution optimale sur les instances de grande taille, il permet tout de même une amélioration importante de la qualité des solutions obtenues à l'aide de l'heuristique basée sur la génération de colonnes. Toutefois, pour juger de la qualité réelle des solutions, il nous faut à présent comparer les résultats obtenus après simulation. Ces résultats obtenus après simulation, avec les stratégies de programmation dynamique mentionnées au chapitre 4, sont regroupés dans les tableaux 7.6 et 7.7.

Après simulation, on constate que se baser sur les solutions fournies par l'algorithme de branch and price conduit à desservir plus de clients que l'heuristique basée sur la priorité mais moins de clients que l'heuristique basée sur la génération de colonnes sur les instances à 30 clients. Ceci s'explique sur ces petites instances par le fait que la solution optimale peut

Tableau 7.6 Comparaison des 3 méthodes après simulation, stratégie WR

# clients	# clients obligatoires	Heuristique basée sur la priorité			Heuristique basée sur la génération de colonnes			Branch and price		
		non servis	Distance (km)	retard	non servis	Distance (km)	retard	non servis	Distance (km)	retard
30	5	5,3	199	2	1,2	197	0,6	2,3	187	0,2
	6	5,6	212	3,1	3,8	199	1,1	4,2	181	1,2
	7	5,5	209	2,7	4,0	206	2,1	4,3	183	1,6
	8	7,6	195	3,2	3,2	201	2,4	4,7	180	1,8
	9	7	197	4,6	4,5	203	4,8	4,9	184	3,6
40	5	9,7	193	1,8	7,8	206	1,8	7,7	202	1,3
	6	12,7	204	2,7	9,9	214	2,2	8,7	196	1,6
	7	13,3	198	2,4	11,7	221	2,7	10,8	207	3,1
	8	13,2	198	3,6	11,1	201	2,7	10,5	194	2,7
	9	14,5	189	6,8	12,3	198	8,2	11,8	201	7,1
50	5	17,9	185	2	16,3	195	1,8	16,0	191	1,6
	6	21,1	189	3,6	17,5	193	1,5	17,5	192	1,1
	7	22,2	184	3,6	19,1	194	2,9	19,0	192	3,2
	8	22,1	187	4,3	19,7	188	3,9	19,2	182	2,8
	9	22,8	180	5,7	21,3	187	3,5	20,8	186	3,9

Tableau 7.7 Comparaison des 3 méthodes après simulation, stratégie OS

# clients	# clients obligatoires	Heuristique basée sur la priorité			Heuristique basée sur la génération de colonnes			Branch and price		
		non servis	Distance (km)	retard	non servis	Distance (km)	retard	non servis	Distance (km)	retard
30	5	3,4	211	2	0,7	199	7,4	1,9	189	5,8
	6	5,6	212	3,2	1,9	208	19,2	2,5	190	36,3
	7	5,5	209	2,8	2,8	213	30,8	2,4	193	25,8
	8	6,1	204	3,2	2,2	208	29,3	3,7	188	30,8
	9	6,4	201	4,6	2,0	214	62,1	3,0	198	51,0
40	5	9,7	194	1,8	6,4	220	44,7	6,3	216	47,4
	6	12,7	204	2,7	7,9	229	59,1	7,0	211	44,0
	7	13,3	198	2,5	8,8	241	78,2	7,8	226	80,0
	8	13,2	198	3,6	7,9	232	93,0	6,7	216	116,7
	9	14	191	5,6	7,8	232	135,0	8,0	222	101,5
50	5	17,9	185	2	14,6	214	68,9	14,6	200	49,1
	6	20,5	189	3,6	15,5	208	48,9	15,2	200	43,7
	7	21,5	185	3,6	16,7	208	62,8	15,8	204	68,4
	8	22,1	187	4,4	16,4	205	82,2	16,6	198	64,4
	9	22,7	180	5,9	16,8	210	102,6	16,1	202	100,3

être très déséquilibrée étant donné le nombre de clients considérés. On peut ainsi avoir une solution optimale dans laquelle deux véhicules desservent, à eux deux, 25 clients tandis que le dernier véhicule ne dessert que 5 clients. Lors de la simulation, les routes étant déséquilibrées, on observera alors plus de clients annulés. Sur les instances à 40 et 50 clients, par contre, on observe une diminution du nombre de clients non desservis et de la distance parcourue après simulation dans le cadre de l'algorithme de branch and price. En ce qui concerne le retard observé, il garde des valeurs similaires dans les deux méthodes.

7.5 Conclusion

Dans ce chapitre, nous avons proposé une méthode exacte : un algorithme de branch and price pour résoudre le problème de tournées de service avec priorité entre les clients et temps de parcours et de service stochastiques. Nous avons proposé plusieurs méthodes pour résoudre le sous-problème, à savoir deux variantes approchées (programmation dynamique bidirectionnelle avec nombre de clients visités et programmation dynamique bidirectionnelle avec vecteur de visites sur un graphe restreint) et une variante exacte (programmation dynamique bidirectionnelle avec vecteur de visites). Cette méthode nous a permis d'établir les solutions optimales pour toutes les instances à 30 clients. Pour les instances à 40 et 50 clients, elle nous a fourni des solutions approchées, de qualité nettement supérieures à celles fournies par l'heuristique basée sur la génération de colonnes (avec un gap pouvant atteindre 7%). En comparant cette méthode avec les deux méthodes proposées précédemment après simulation, nous avons pu constater que cette méthode fournissait des solutions desservant moins de clients que la méthode du chapitre précédent sur les instances à 30 clients. Par contre, sur les instances de taille 40 et 50, l'algorithme de branch and price permet d'obtenir des solutions de meilleure qualité du point de vue du nombre de clients desservis et de la distance parcourue. En ce qui concerne le retard, les résultats obtenus sont comparables.

CHAPITRE 8

Conclusion

Dans cette thèse, nous avons présenté une approche globale de résolution pour résoudre le problème de tournées de service avec priorité entre les clients et temps de service et de parcours stochastiques. Cette méthode consiste en deux étapes : une étape de planification et une étape d'exécution. L'étape de planification consiste à construire des routes desservant des clients optionnels et obligatoires en utilisant des estimés des temps de service et de parcours connus a priori. Durant l'étape d'exécution, on prend en compte la stochasticité des temps de service et de parcours, et on utilise des outils de programmation dynamique pour déterminer la politique optimale. Après la programmation dynamique, on procède à un ensemble de simulations de l'exécution des tournées en temps réel au long de la période afin d'évaluer la qualité des solutions obtenues. Dans le cadre de cette approche, nous avons développé trois méthodologies pour l'étape de planification. La première, une heuristique basée sur la priorité des clients, consiste à construire un squelette de clients obligatoires puis à insérer des clients optionnels dans ce squelette. La deuxième, une heuristique basée sur la génération de colonnes, consiste à générer des routes contenant des clients obligatoires et optionnels puis à sélectionner exactement une route pour chaque véhicule. La dernière méthode est un algorithme de branch and price. En ce qui concerne l'étape d'exécution, nous avons prouvé que la politique optimale de notre algorithme de programmation dynamique est une politique de seuil.

Nous avons d'abord validé nos différentes méthodes en procédant à des tests sur des instances issues de la littérature. L'heuristique basée sur la priorité des clients nous a permis de résoudre à optimalité des instances restées ouvertes de type OPTW. Grâce à l'heuristique basée sur la génération de colonnes, nous avons établi l'optimalité d'un ensemble de meilleures solutions connues sur d'autres instances de type OPTW. Après validation de ces méthodes, nous avons procédé aux expérimentations sur des instances issues de données industrielles. La première méthode présente de bons résultats, avec des temps de calcul raisonnables sur l'ensemble des instances. Toutefois, elle présente un inconvénient : elle présuppose un nombre suffisant de clients obligatoires. La deuxième méthode présente de meilleurs résultats que la première méthode (aussi bien au niveau du nombre de clients visités que de la distance parcourue). Aussi, au niveau des temps de calcul, elle permet de résoudre toutes les instances en moins de 5 minutes. Enfin, la troisième méthode nous a permis d'obtenir les solutions

optimales sur les instances à 30 clients et ainsi d'apprécier la qualité des solutions obtenues avec la deuxième méthode. Elle nous a également permis de résoudre de façon approchée les instances à 40 et 50 clients. Sur ces instances, nous avons pu observer que les solutions fournies par l'algorithme de branch and price étaient nettement meilleures que celles obtenues par l'heuristique basée sur la génération de colonnes (avant comme après simulation). Toutefois, les temps de calcul de l'algorithme de branch and price sont nettement plus élevés que ceux de l'heuristique basée sur la génération de colonnes. En ce qui concerne l'étape d'exécution, nous avons pu constater que la stratégie de programmation dynamique consistant à considérer un seul segment est préférable du point de vue du nombre de clients desservis tandis que la stratégie considérant le reste de la route est préférable pour le retard et la distance parcourue.

Parmi les trois méthodologies développées dans cette thèse pour l'étape de planification, on préfèrera donc utiliser l'heuristique de génération de colonnes si on est prêt à sacrifier la qualité des solutions pour gagner en temps de calcul et, inversement, on préfèrera utiliser l'algorithme de branch and price si on est prêt à sacrifier du temps pour gagner en qualité de la solution obtenue. Du point de vue de l'étape d'exécution, les résultats nous orientent plutôt vers la stratégie de programmation dynamique consistant à ne considérer qu'un seul segment. Toutefois, cette décision est discutable. Pour déterminer si ce choix est judicieux, il faudrait intégrer ces trois méthodologies dans un horizon multi-période afin de juger de la qualité des solutions sur un horizon roulant.

RÉFÉRENCES

[1] ALSHEDDY, A. et TSANG, E. (2011). Empowerment scheduling for a field workforce. *Journal of Scheduling*, 14, 639–654.

[2] ANDO, N. et TANIGUCHI, E. (2006). Travel Time Reliability in Vehicle Routing and Scheduling with Time Windows. *Networks and Spatial Economics*, 6, 293–311.

[3] ANGELELLI, E., BIANCHESSI, N., MANSINI, R. et SPERANZA, M. (2009). Short term strategies for a dynamic multi-period routing problem. *Transportation Research Part C : Emerging Technologies*, 17, 106–119.

[4] AUGERAT, P. (1995). *Approche polyèdrale du problème de tournées de véhicules*. Thèse de doctorat, Institut National Polytechnique de Grenoble-INPG.

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

[6] BARNHART, C., JOHNSON, E., NEMHAUSER, G., SAVELSBERGH, M. et VANCE, P. (1998). Branch-and-price : Column generation for solving huge integer programs. *Operations research*, 46, 316–329.

[7] BETTINELLI, A., CESELLI, A. et RIGHINI, G. (2011). A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows. *Transportation Research Part C : Emerging Technologies*, 19, 723–740.

[8] BOLAND, N., DETHRIDGE, J. et DUMITRESCU, I. (2006). Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, 34, 58–68.

[9] BORENSTEIN, Y., SHAH, N., TSANG, E., DORNE, R., ALSHEDDY, A. et VOUDOURIS, C. (2009). On the partitioning of dynamic workforce scheduling problems. *Journal of Scheduling*, 13, 411–425.

[10] BOSTEL, N., DEJAX, P. et GUEZ, P. (2008). Multiperiod planning and routing on a rolling horizon for field force optimization logistics. *Routing Problem : Latest Advances*.

[11] BOUSSIER, S., FEILLET, D. et GENDREAU, M. (2007). An exact algorithm for team orienteering problems. *4or*, 5, 211–230.

[12] BRAMEL, J. et SIMCHI-LEVI, D. (2001). Set-covering-based algorithms for the capacitated vrp. *The vehicle routing problem*, 9, 85–108.

[13] BRANCHINI, R., AMARAL ARMENTANO, V. et LøKKETANGEN, A. (2009). Adaptive granular local search heuristic for a dynamic vehicle routing problem. *Computers & Operations Research*, 36, 2955–2968.

[14] CAMPBELL, A., GENDREAU, M. et THOMAS, B. (2011). The orienteering problem with stochastic travel and service times. *Annals of Operations Research*, 186, 61–81.

[15] CESCHIA, S., DI GASPERO, L. et SCHAERF, A. (2011). Tabu search techniques for the heterogeneous vehicle routing problem with time windows and carrier-dependent costs. *Journal of Scheduling*, 14, 601–615.

[16] CHAO, I., GOLDEN, B., WASIL, E. *ET AL.* (1996). The team orienteering problem. *European journal of operational research*, 88, 464–474.

[17] CORDEAU, J., GENDREAU, M. et LAPORTE, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30, 105–119.

[18] CORTÉS, C., GENDREAU, M., LENG, D. et WEINTRAUB, A. (2010). A simulation-based approach for fleet design in a technician dispatch problem with stochastic demand. *Journal of the Operational Research Society*, 1–14.

[19] CORTÉS, C., ORDONEZ, F. et SOUYRIS, S. (2007). Routing technicians under stochastic service times : a robust optimization approach. *Tristan VI*. Phuket Island, Thailand, 10–15.

[20] DAYARIAN, I., CRAINIC, T. G., GENDREAU, M. et REI, W. (2013). A column generation approach for a multi-attribute vehicle routing problem. Rapport technique, Technical Report CIRRELT-2013-57, Montreal, CIRRELT.

[21] DELAGE, E. (2010). Re-optimization of technician tours in dynamic environments with stochastic service time. Rapport technique, Ecole des Mines de Nantes.

[22] DESAULNIERS, G., LESSARD, F. et HADJAR, A. (2008). Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42, 387–404.

[23] DESROCHERS, M. (1988). An algorithm for the shortest path problem with resource constraints. Rapport technique, GERAD.

[24] DESROCHERS, M. et SOUMIS, F. (1989). A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23, 1–13.

[25] DUGARDIN, F. (2006). Optimisation réactive de tournées de service en environnement dynamique. Rapport technique, Ecole des Mines de Nantes.

[26] FEILLET, D., DEJAX, P., GENDREAU, M. et GUEGUEN, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints : Application to some vehicle routing problems. *Networks*, 44, 216–229.

[27] FLATBERG, T., HASLE, G., KLOSTER, O., NILSSEN, E. et RIISE, A. (2007). Dynamic and stochastic vehicle routing in practice. V. Zeimpekis, C. Tarantilis, G. Giaglis et I. Minis, éditeurs, *Dynamic Fleet Management*, Springer, chapitre 3. 41–63.

[28] FUKASAWA, R., LONGO, H., LYSGAARD, J., DE ARAGÃO, M., REIS, M., UCHOA, E. et WERNECK, R. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, 106, 491–511.

[29] GAUVIN, C., DESAULNIERS, G. et GENDREAU, M. (2013). A branch-cut-and-price algorithm for the vehicle routing problem with stochastic demands. Rapport technique, Technical report, GERAD.

[30] GÉLINAS, S., DESROCHERS, M., DESROSIERS, J. et SOLOMON, M. M. (1995). A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research*, 61, 91–109.

[31] GENDREAU, M., LAPORTE, G. et SEMET, F. (2001). A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel Computing*, 27, 1641–1653.

[32] GILMORE, P. et GOMORY, R. (1961). A linear programming approach to the cutting-stock problem. *Operations research*, 9, 849–859.

[33] GLOVER, F., LAGUNA, M. *ET AL.* (1997). *Tabu search*, vol. 22. Springer.

[34] HADJICONSTANTINOU, E. et ROBERTS, D. (2002). Routing under uncertainty : an application in the scheduling of field service engineers. P. Toth et D. Vigo, éditeurs, *The vehicle routing problem*, SIAM, chapitre 13. 331–352.

[35] HAGHANI, A. et YANG, S. (2007). Real-time emergency response fleet deployment. V. Zeimpekis, C. Tarantilis, G. Giaglis et I. Minis, éditeurs, *Dynamic Fleet Management*, Springer, chapitre 7. 13–162.

[36] HOLLAND, J. H. (1975). *Adaptation in natural and artificial systems : An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press.

[37] HU, Q. et LIM, A. (2013). An iterative three-component heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*.

[38] JEPSEN, M., PETERSEN, B., SPOORENDONK, S. et PISINGER, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56, 497–511.

[39] JIE, G. (2010). Model and algorithm of vehicle routing problem with time windows in stochastic traffic network. *Logistics Systems and Intelligent Management, 2010*. 848–851.

[40] JOHNSON, E. (1989). Modeling and strong linear programs for mixed integer programming. *Algorithms and Model Formulations in Mathematical Programming*, Springer. 1–43.

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

[42] KENNEDY, J. et EBERHART, R. (1995). Particle Swarm Optimization.

[43] KENYON, A. (1998). *Stochastic Vehicle Routing Problems with Random Travel Times*. Thèse de doctorat, University of Texas at Austin.

[44] KENYON, A. et MORTON, D. (2003). Stochastic vehicle routing with random travel times. *Transportation Science*, 37, 69–82.

[45] KIRKPATRICK, S., JR., D. et VECCHI, M. (1983). Optimization by simmulated annealing. *science*, 220, 671–680.

[46] LAPORTE, G., LOUVEAUX, F. et MERCURE, H. (1992). The Vehicle Routing Problem with Stochastic Travel Times. *Transportation Science*, 26, 161–170.

[47] 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.

[48] LEI, H., LAPORTE, G. et GUO, B. (2011). A generalized variable neighborhood search heuristic for the capacitated vehicle routing problem with stochastic service times. *Top*, Online Fir.

[49] LI, X., TIAN, P. et LEUNG, S. (2010). Vehicle routing problems with time windows and stochastic travel and service times : Models and algorithm. *International Journal of Production Economics*, 125, 137–145.

[50] LYSGAARD, J. (2006). Reachability cuts for the vehicle routing problem with time windows. *European Journal of Operational Research*, 175, 210–223.

[51] MARTI, R., LAGUNA, M. et GLOVER, F. (2006). Principles of scatter search. *European Journal of Operational Research*, 169, 359–372.

[52] MLADENOVIĆ, N. et HANSEN, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24, 1097–1100.

[53] MOSCATO, P. et COTTA, C. (2003). A gentle introduction to memetic algorithms. *Handbook of metaheuristics*, Springer. 105–144.

[54] PETRAKIS, I., HASS, C. et BICHLER, M. (2012). On the impact of real-time information on field service scheduling. *Decision Support Systems*, 53, 282–293.

[55] PINTO, R. (2012). *Exact Algorithms for Arc and Node Routing Problems*. Thèse de doctorat, Pontifćia Universidade Católica do Rio de Janeiro.

[56] RASMUSSEN, M., JUSTESEN, T., DOHN, A. et LARSEN, J. (2012). The home care crew scheduling problem : Preference-based visit clustering and temporal dependencies. *European Journal of Operational Research*, 219, 598–610.

[57] RIGHINI, G. et SALANI, M. (2004). Dynamic programming algorithms for the elementary shortest path problem with resource constraints. *Electronic Notes in Discrete Mathematics*, 17, 247–249.

[58] RIGHINI, G. et SALANI, M. (2009). Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers & Operations Research*, 36, 1191–1203.

[59] RUSSELL, R. A. et URBAN, T. L. (2007). Vehicle routing with soft time windows and Erlang travel times. *Journal of the Operational Research Society*, 59, 1220–1228.

[60] SAVELSBERGH, M. (1997). A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 45, 831–841.

[61] SHAO, Z., GAO, S. et WANG, S. (2010). A Hybrid Particle Swarm Optimization Algorithm for Vehicle Routing Problem with. *Engineering Applications of Artificial Intelligence*, 23, 566–574.

[62] SHEN, Z., ORDÓNEZ, F. et DESSOUKY, M. (2006). The minimum unmet demand stochastic vehicle routing problem.

[63] SOUYRIS, S., CORTÉS, C., ORDÓÑEZ, F. et WEINTRAUB, A. (2012). A robust optimization approach to dispatching technicians under stochastic service times. *Optimization Letters*.

[64] TAS, D., DELLAERT, N., WOENSEL, T. et KOK, T. (2011). Vehicle Routing Problem with Stochastic Travel Times Including Soft Time Windows and Service Costs Vehicle Routing Problem with Stochastic Travel Times Including Soft Time Windows and Service Costs. *Computers & Operations Research*, 364.

[65] TAVAKKOLI-MOGHADDAM, R., ALINAGHIAN, M., SALAMAT-BAKHSH, A. et NOROUZI, N. (2012). A hybrid meta-heuristic algorithm for the vehicle routing problem with stochastic travel times considering the driver's satisfaction. *Journal of Industrial Engineering International*, 8, 1–6.

[66] TENG, S. Y., ONG, H. L. et HUANG, H. C. (2004). An integer L-shaped algorithm for time-constrained traveling salesman problem with stochastic travel and service times. *Asia-Pacific Journal of Operational Research*, 21, 241–257.

[67] TOPALOGLU, H. (2007). A parallelizable and approximate dynamic programming-based dynamic fleet management model with random travel times and multiple vehicle

types. V. Zeimpekis, C. Tarantilis, G. Giaglis et I. Minis, éditeurs, *Dynamic Fleet Management*, Springer, chapitre 4. 65–93.

[68] TRICOIRE, F. (2006). *Optimisation des tournées de véhicules et de personnels de maintenance : application à la distribution et au traitement des eaux*. Thèse de doctorat, Ecole des Mines de Nantes.

[69] TRICOIRE, F., BOSTEL, N., DEJAX, P. et GUEZ, P. (2011). Exact and hybrid methods for the multiperiod field service routing problem. *Central European Journal of Operations Research*, Online Fir.

[70] TSILIGIRIDES, T. (1984). Heuristic methods applied to orienteering. *Journal of Operational Research Society*, 35, 797–809.

[71] VANSTEENWEGEN, P., SOUFFRIAU, W., VANDEN BERGHE, G. et VAN OUDHEUSDEN, D. (2009). Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36, 3281–3290.

[72] WANG, X. et REGAN, A. (2001). Assignment models for local truckload trucking problems with stochastic service times and time window constraints. *Transportation Network Modeling*, 1771, 61–68.

[73] XU, H. (1994). *Optimal policies for stochastic and dynamic vehicle routing problems*. Thèse de doctorat, Massachusetts institute of Technology.

[74] ZEIMPEKIS, V., MINIS, I., MAMASSIS, K. et GIAGLIS, G. M. (2007). Dynamic management of a delayed delivery vehicle in a city logistics environment. V. Zeimpekis, C. Tarantilis, G. Giaglis et I. Minis, éditeurs, *Dynamic Fleet Management*, Springer, chapitre 9. 197–217.

[75] ZHANG, T., CHAOVALITWONGSE, W. et ZHANG, Y. (2012). Scatter search for the stochastic travel-time vehicle routing problem with simultaneous pick-ups and deliveries. *Computers & Operations Research*, 39, 2277–2290.