N° d'ordre :26/2010-M/MT

REPUBLIQUE ALGERIENNE DEMOCRATIQUE POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

Université des Sciences et de la Technologie Houari Boumediene

Faculté des Mathématiques

Mémoire

Présenté pour l'obtention du diplôme de MAGISTER

En : Mathématiques

Spécialité : Recherche Opérationnelle : Mathématiques Discrètes et Optimisation

Par : **Zitout Zohra**

Thème

Problème de distribution avec les rechargements

Soutenu publiquement le 21/06/2010 à 11h, devant le jury composé de :

A. Berrachedi	Professeur	à l'USTHB	Président
M. Aïder	Professeur	à l'USTHB	Directeur de thése
R. Ouafi	Maitre de Conférences /A	à l'USTHB	Examinateur
C. Adiche	Maitre de Conférences / B	à l'USTHB	Invitée

Résumé

Dans ce mémoire, nous nous intéressons à une variante du problème de tournées de véhicules, dans lequel on va rajouter la récupération des consignes auprès des clients et de les remettre au point de départ de chaque demande, avec la condition que n'importe quelle demande ou consigne peut être déchargée ailleurs qu'en sa destination, rechargée plus tard par le même véhicule ou un autre véhicule et ainsi de suite jusqu'à ce qu'elle arrive à sa destination finale.

Il s'agira de discuter de la complexité de ce problème et de ses différentes variantes et d'étudier les nombreuses formulations qu'il peut présenter. Nous évoquerons ainsi l'approche algorithmique basée notamment sur les méthodes exactes de type branch and bound.

Mots Clés : Problème de transport, Programmation linéaire en nombres entiers, Problèmes de flots.

Remerciements

D'abord, je remercie le Bon Dieu qui m'a donné la force et le courage pour terminer ce travail.

Je tiens à remercier particulièrement Mr Méziane Aïder, mon directeur de thèse, pour avoir été présent à tout instant, pour ses connaissances et conseils avisés. Qu'il trouve ici l'expression de ma profonde reconnaissance pour ses qualités humaines et scientifiques.

Je le remercie pour toute sa confiance et la liberté qu'il m'a accordée durant la réalisation de ce travail.

Mes sincères remerciements vont également à Mr Abdelhafidh BERRACHEDI qui m'a honoré par son accord d'être président du jury de ma thèse.

Je remercie vivement Mr Rachid OUAFI et Mme Chahrazed ADICHE d'avoir accepté d'examiner ce travail.

Je ne terminerai pas sans adresser un immense merci à mes chers parents et à mon cher mari pour le soutien et l'encouragement qu'ils m'ont apportés durant toutes mes années d'études, ainsi que mes sœurs, mes frères et tous ceux qui m'ont de près ou de loin aidée, merci à tous.

Dédicaces

Je dédie ce travail à mes chers parents,

à mon cher mari,

à ma grand-mère,

à mes sœurs et mes frère,

à toute ma famille,

à mes collègues, mes amies ,

à tous ceux qui m'ont témoigné leurs soutiens.

A Nabil

Table des matières

Introduction Générale

Les opérations de transport sont essentielles au bon fonctionnement de notre économie. Le transport routier des marchandises permet l'acheminement des matières et des produits entre entreprises et vers les consommateurs. Il permet entre autres la distribution de tous les produits tels que la nourriture, les vêtements, les meubles, etc. Le transport joue un rôle clé dans la chaîne logistique puisqu'un produit est rarement fabriqué et consommé au même endroit. Pratiquement chaque produit acheté par un consommateur est passé par un camion ou autre véhicule de transport. Certains produits passent par quatre ou cinq camions ou même plus avant d'aboutir entre les mains du consommateur final. Étant donné l'étendue géographique de notre pays, le transport routier des marchandises joue un rôle économique primordial.

Le nombre de camions sur les routes ne cesse d'augmenter d'année en année. Par ailleurs, les coûts des carburants augmentent également constituant une part importante des frais d'opération d'un transporteur. Par conséquent, la rentabilité d'une entreprise est grandement affectée par les décisions relatives à l'assignation et à la séquence des livraisons. La planification des itinéraires est donc un problème majeur pour les entreprises de transport. Étant donné les conséquences économiques et stratégiques des problèmes de transport, ces derniers ont attiré l'attention de nombreux chercheurs et ce, depuis plusieurs années.

Le problème de base en transport, probablement le plus étudié, est le problème du voyageur de commerce (Traveling Salesman Problem) qui permet de visiter à moindres coûts un ensemble de clients chacun une et une seul fois. Le problème consiste à trouver l'ordre dans lequel chacun des clients sera visité.

À ce problème, de nombreuses contraintes peuvent s'ajouter permettant ainsi de

l'adapter à des problèmes pratiques rencontrés dans l'industrie du camionnage. Par exemple, le problème de tournées de véhicules (Vehicle Routing Problem) qui est un autre problème également étudié intensivement, traite le cas où chacun des clients a une demande déterminée et où la flotte est homogène (présence de plusieurs véhicules de transport de même type).

Le problème proposé dans ce mémoire est un problème de tournées de véhicules avec une flotte homogène et plusieurs produits (produits livrés et consignes récupérés). Il s'agit d'un problème de tournées de véhicules puisque les clients à visiter requièrent une demande connue à l'avance et les véhicules sont limités par leur capacité en terme de poids pour les produits et de volume pour la récupération. On dit que la flotte est homogène puisque la flotte de camions de l'entreprise est constituée de camions ayant des capacités égales. De plus, l'entreprise distribue plusieurs produits ayant des poids différents. Finalement, l'entreprise doit aussi récupérer auprès des clients les bouteilles et canettes vides.

Le problème consiste donc à visiter tous les clients autant de fois que nécessaire afin de leur livrer la marchandise commandée et rapporter la récupération tout en respectant les contraintes de volume et de capacité. L'objectif est de minimiser le coût des déplacements pour visiter tous les clients.

Pour résoudre ce problème, nous allons utiliser la méthode "Branch and Bound".

Notre mémoire est organisé comme suit :

Le chapitre 1 est adressé au lecteur non averti pour le familiariser avec les différentes notions de base de l'optimisation combinatoire.

Nous y présentons, la définition de la problématique de l'optimisation combinatoire, les difficultés inhérentes, ainsi que les différentes facettes développées pour son approche. Pour ce faire, le lecteur trouvera aussi bien les définitions formelles qu'informelles. Ces dernières lui serviront de guide dans le suivi du raisonnement établi et pour la compréhension de la problématique.

Le chapitre 2 présente les principaux variantes de problèmes de tournées de véhicules, les algorithmes utilisés pour les résoudre ainsi que les résultats qui en découlent.

A cet effet, une revue des articles récents sera effectuée de manière à mettre à jour les connaissances dans le domaine des tournées de véhicules. Ensuite, nous discutons de sa NP-complétude à travers les résultats existants et des exemples que nous construisons nous-mêmes. Nous exposons, également, les algorithmes proposés pour résoudre les problèmes de tournées de véhicules les plus en vogue ces dernières années.

Après avoir décrit le problème de tournées de véhicules pour permettre au lecteur de comprendre notre travail, nous développons à partir du troisième chapitre l'objet de notre mémoire.

Le chapitre 3 présente le problème de distribution avec les rechargements. Dans un premier temps, la problématique sera présentée de façon détaillée. Puis, nous présentons la formulation par la programmation linéaire mixte. Ensuite, nous proposons les coupes que nous introduisons dans notre modèle et enfin nous présentons la méthode "branch and bound" que nous utilisons pour résoudre notre problème .

Le chapitre 4 décrit l'algorithme "branch and bound" utilisée pour résoudre la problématique, suivi des résultats numériques.

La conclusion présente une synthèse des travaux que nous avons réalisés et propose quelques perspectives de recherche future.

1

Optimisation Combinatoire

Contents

1.1 Introduction

Le but de ce chapitre est de situer le cadre théorique dans lequel s'inscrit notre travail de recherche. A ce titre, nous nous proposons d'abord de familiariser le lecteur non averti aux différentes notions de base de l'optimisation combinatoire.

Nous présentons la définition de la problématique de l'optimisation combinatoire, les difficultés inhérentes, ainsi que les différentes facettes développées pour son approche telles que : la complexité des algorithmes, la théorie de la complexité des problèmes, les problèmes complets et les problèmes NP-complets. Nous terminons en présentant des méthodes générales exactes et approchées pour la résolution des problèmes NP-Complets.

Pour ce faire, le lecteur trouvera aussi bien des définitions qui lui serviront de guide pour la compréhension de notre travail.

1.2 Problématique de l'optimisation combinatoire

"Combinatoire" désigne la discipline des mathématiques concernée par les structures "discrètes" ou "finies". Citons quelques branches de cette discipline : la théorie des graphes, la combinatoire énumérative, les problèmes de dénombrement, la combinatoire polyédrale, l'optimisation combinatoire, etc.
Les frontières entre les branches ne sont pas hermétiques, les différentes branches expriment plutôt des orientations méthodologique différentes.

L' "optimisation", ou "programmation mathématique" sont des termes utilisés pour recouvrir toutes les méthodes qui servent à déterminer l'optimum d'une fonction avec (ou sans) contraintes. Cette fonction modélise le choix optimal. Les ingénieurs, les économistes..., font souvent des choix optimaux (ou quasi-optimaux), d'où l'importance de l'optimisation dans les sciences, qu'elles soient techniques, mathématiques, physiques, informatiques, économiques, naturelles, etc.

L' "Optimisation combinatoire" consiste à trouver le meilleur entre un nombre fini de choix. Autrement dit, minimiser une fonction, avec contraintes, sur un ensemble fini.

Definition 1.2.1. *Plus formellement, l'optimisation est l'étude des problèmes qui sont de*

la forme :

Étant donné *: une fonction $f : S \to R$, d'un ensemble S dans l'ensemble des nombres réels,*

Rechercher *: un élément x_0 de S tel que $f(x_0) \geq f(x)$ pour tous les $x \in S$ (« maximisation ») ou tel que $f(x_0) \leq f(x)$ pour tous les $x \in S$ (« minimisation »).*

Une telle formulation est parfois appelée programme mathématique. Plusieurs problèmes théoriques et pratiques peuvent être étudiés dans cet encadrement général.

Remarque 1.2.1. *Étant donné que la maximisation de f est équivalente à la minimisation de $-f$, une méthode pour trouver le minimum (ou le maximum) suffit à résoudre le problème d'optimisation.*

Les formulations des problèmes mathématiques dépendent de la nature de la fonction objectif de l'ensemble des contraintes. Les sous-domaines majeurs suivants existent :

- La programmation linéaire étudie les cas où la frontière de l'ensemble S et la fonction objectif sont linéaires.
- La programmation linéaire en nombres entiers étudie les programmes linéaires dans lesquels certaines ou toutes les variables sont contraintes à prendre des valeurs entières. Ces problèmes peuvent être résolus par différentes méthodes : séparation et évaluation, plans sécants.
- La programmation quadratique permet à la fonction objectif d'avoir des termes quadratiques, tout en conservant une description de l'ensemble S à partir d'égalités/inégalités linéaires.
- La programmation non-linéaire étudie le cas général dans lequel l'objectif ou les contraintes (ou les deux) contiennent des parties non-linéaires.
- La programmation stochastique étudie le cas dans lequel certaines des contraintes dépendent de variables aléatoires.
- La programmation dynamique utilise la propriété qu'une solution optimale se compose nécessairement de sous-solutions optimales (le contraire n'est pas vrai en général) pour décomposer le problème en évitant l'explosion combinatoire. Elle n'est utilisable que lorsque la fonction objectif est monotone croissante.

Exemples de problèmes d'optimisation combinatoire

Exemple 1.2.1. *[?]* Le problème de cheminement : Il s'agit de choisir un itinéraire de longueur minimum pour aller d'une ville A à une ville B. On suppose donnée une carte routière à laquelle on associe un "graphe" $G = (X, U)$ où X, l'ensemble des "sommets", est en bijection avec l'ensemble des carrefours de la carte. Deux sommets x et y étant joints par un "arc", segment orienté élément de U, si on peut se rendre du carrefour correspondant à x à celui correspondant à y par un tronçon de route direct. En général le graphe d'une carte routière est symétrique (c'est à dire $(xy) \in U \Rightarrow (yx) \in U$). Il n'en est pas nécessairement de même pour le graphe d'un plan de ville (penser aux sens uniques). A chaque arc $u \in U$, on associe sa "longueur" $d(u)$ et on appelle "réseau" $R = (X, U, d)$ le graphe muni de l'application d.

Le problème du plus court chemin de A à B peut maintenant se formuler de la manière suivante. Étant donnés deux sommets A et B d'un réseau $R = (X, U, d)$, trouver une séquences d'arcs dont l'extrémité initiale du premier arc est A et l'extrémité terminale du dernier arc est B, de longueur minimum, la longueur d'un chemin étant égale à la somme des longueurs des arcs composant la séquence. Le graphe (X, U) étant fini, le nombre de chemins de A à B ne repassant pas par le même sommet (on dit "élémentaires") est également fini.

Exemple 1.2.2. Le problème du voyageur de commerce : Un voyageur de commerce ayant n villes à visiter souhaite établir une tournée qui lui permette de passer une fois et une seule dans chaque ville pour finalement revenir à son point de départ, ceci en minimisant la longueur du chemin parcouru. Il s'agit donc, étant donné un réseau $R = (X, U, d)$ de trouver un circuit passant une fois et une seule par tous les sommets (qui représentent les villes), c'est à dire un chemin élémentaire dont les extrémités sont confondues et ayant un nombre d'arcs égal au nombre n de villes, de longueur totale minimum.
Une manière équivalente de formuler ce problème est de se donner une $n \times n - matrice$ dont l'élément D_i^j de la i^{me} ligne et de la j^{me} colonne est égal à :

* la distance de la ville i à la ville j s'il existe un moyen d'aller directement de i à j (c'est à dire si $(ij) \in U$).
* ∞, sinon.

Le problème consiste alors à trouver une permutation p sans cycle (i.e. non décomposable) de n objets telle que

$$\sum_{i=1}^{n} D_i^{p(i)} \text{ soit minimum}$$

Telle que nous l'entendons, l'optimisation combinatoire se situe au carrefour de la théorie des graphes, de la programmation mathématique, de l'informatique théorique (algorithmique et théorie de la complexité).

1.3 Graphe et programmation mathématiques

1.3.1 Graphe

La théorie des graphes est née en 1736 quand Euler démontra qu'il était impossible de traverser chacun des sept ponts de la ville russe de Königsberg (aujourd'hui Kaliningrad) une fois exactement et de revenir au point de départ. Les ponts enjambent les bras de la Pregel qui coule de part et d'autre de l'île de Kneiphof. Dans la figure suivante, les nœuds représentent les rives.

FIG. 1.1 – Les sept ponts de Königsberg

La théorie des graphes constitue un domaine des mathématiques qui, historiquement, s'est aussi développé au sein de disciplines diverses telles que la chimie (modélisation de structures), la biologie (génome), les sciences sociales (modélisation des relations) ou en vue d'applications industrielles (problème du voyageur de commerce). Elle constitue l'un des instruments les plus courants et les plus efficaces pour résoudre des problèmes discrets posés en Recherche Opérationnelle (RO).

De manière générale, un graphe permet de représenter simplement la structure, les connexions, les cheminements possibles d'un ensemble complexe comprenant un grand nombre de situations, en exprimant les relations, les dépendances entre ses éléments (e.g., réseau de communication, réseaux ferroviaires ou routiers, arbres généalogiques, diagrammes de succession de tâches en gestion de projets, ...).
[**?**, **?**]

1.3.1.1 Définitions fondamentales de la théorie des graphes

Concepts non orientés

Definition 1.3.1. *Un graphe G consiste en un ensemble fini non vide $V = V(G)$ de n sommets, dit ensemble de sommets et un ensemble $E = E(G)$ de m paires de sommets distincts de V, appelées arêtes de G. Le graphe G est dit d'ordre n et de taille m.*

Si $e = (x, y)$ est une arête (notée également xy) de G, alors nous dirons que les sommets x et y sont reliés par l'arête e. Ce sont alors des voisins et ils sont adjacents. De plus, ils constituent les extrémités de l'arête e, qui est alors dite incidente à chacune de ses extrémités. Lorsque deux arêtes ont une extrémité commune, elles sont dites adjacentes. Autrement, elles sont dites disjointes.

On dit qu'un graphe est sans boucle si E ne contient pas d'arête de la forme (x, x), c'est-à-dire joignant un sommet à lui-même.

Le degré d'un sommet x de G, noté $d(x)$ ou $d_G(x)$, est le nombre d'arêtes qui lui sont incidentes. C'est aussi, pour les graphes simples, le nombre de voisins de x, c'est à dire :

$$d_G(x) = |N(x)|$$

où $N(x)$ désigne l'ensemble des voisins de x dans G.

Definition 1.3.2. *Soit $G = (V, E)$ un graphe et soit $V' \subset V$ et $E' \subset E$.*

Le sous-graphe engendré par V', noté $G[V']$, est le graphe dont l'ensemble de sommets est V' et dont les arêtes sont celles dont les extrémités sont des sommets de V'. Un sous-graphe engendré par un sous-ensemble de sommets est dit sous-graphe induit.

Le sous-graphe engendré par E', noté $G[E']$, est le sous-graphe de G dont l'ensemble de sommets est égal à l'ensemble des extrémités des arêtes de E' et l'ensemble des arêtes est E'.

Nous appelons graphe partiel de G, un sous-graphe de G de la forme $G' = (V, E')$ avec $E' \subset E$.

Definition 1.3.3. *Une chaîne est une séquence finie et alternée de sommets et d'arêtes, débutant et finissant par des sommets. Une arête ne doit pas intervenir plusieurs fois dans la séquence contrairement à un sommet.*

Le premier et le dernier sommet sont appelés (sommets) extrémités de la chaîne.

La longueur de la chaîne est égale au nombre d'arêtes qui la composent.

Si aucun des sommets composant la séquence n'apparaît plus d'une fois, la chaîne est dite chaîne élémentaire.

Si aucune des arêtes composant la séquence n'apparaît plus d'une fois, la chaîne est dite chaîne simple.

Un cycle est une chaîne dont les extrémités coïncident.

Un cycle élémentaire (tel que l'on ne rencontre pas deux fois le même sommet en le parcourant) est un cycle minimal pour l'inclusion, c'est-à-dire ne contenant strictement aucun autre cycle.

Graphe complet

Definition 1.3.4. *Un graphe $G = (V, A)$ est dit complet si, pour toute paire de sommets (x, y), il existe au moins un arc de la forme (x, y) ou (y, x).*

Un graphe simple complet d'ordre n est noté K_n.

Concepts orientés

Definition 1.3.5. *On appelle graphe orienté $G = (V, A)$ la donnée d'un ensemble V dont les éléments sont appelés sommets et d'une partie A de $V \times V$ dont les éléments sont appelés arcs.*

En présence d'un arc $a = (x, y)$ qui peut être noté simplement xy, on dit que x est l'origine (ou extrémité initiale) et y l'extrémité (terminale) de a, que a est sortant en x et incident en y, et que y est un successeur de x tandis que x est un prédécesseur de y. On dit aussi que x et y sont adjacents.

Definition 1.3.6. *Un chemin est une séquence finie et alternée de sommets et d'arcs, débutant et finissant par des sommets, telle que chaque arc est sortant d'un sommet et incident au sommet suivant dans la séquence (cela correspond à la notion de chaîne "orientée").*

Si aucun des sommets composant la séquence n'apparaît plus d'une fois, le chemin est dit chemin élémentaire.

Si aucune des arêtes composant la séquence n'apparaît plus d'une fois, le chemin est dit chemin simple.

Un circuit est un chemin dont les extrémités coïncident.

En parcourant un circuit élémentaire, on ne rencontre pas deux fois le même sommet.

Réseaux, flots, capacité résiduelle et coupes

Definition 1.3.7. *Soit $G = (V, A)$ un graphe orienté, s et $t \in V$ deux sommets, et $c : \overrightarrow{A} \to N$ une application ; on appelle c la fonction de capacié sur G, alors le quadriplet $N = (G, s, t, c)$ est un réseau.*

Definition 1.3.8. *Un flot est une fonction f qui associe à chaque arc (x, y) un entier $f(x, y)$, son flux, avec*

$$0 \leq f(x, y) \leq c(x, y)$$

, et qui satisfait la loi de Kirchhoff, c'est à dire que pour chaque sommet sauf pour la source et la destination, la somme des flux des arcs entrants est égale à celle des flux des arcs sortants.

Definition 1.3.9. *$R = (V, A, c)$ réseau avec sources s et destination t, f flot sur R. Pour tout arc $a \in A$, la capacité résiduelle de a est*

$$c_{(f)}(a) = c(a) - f(a)$$

.

Definition 1.3.10. *Une coupe de $R = (V, A, c)$ est une partition de V en Y et $\overline{Y}$ avec $s \in Y et \ \ p \in \overline{Y}$.*

1.3.2 Programmation mixte

Il s'agit de problèmes d'optimisation combinatoire dont le nom est une traduction approximative de l'anglais "mixed-integer programming", et dont la forme la plus générale s'écrit :

$$(P') \begin{cases} \text{Minimiser} \ \ c^T x \\ \text{sous les contraintes} \\ Ax \leq b \\ (x_1, ..., x_n) \in \mathbb{Z} \\ (x_{n+1}, ..., x_{n+p}) \in \mathbb{R}^p \end{cases}$$

avec $m, n, p \in \mathbb{N}, A \in M_{m,n+p}(\mathbb{R})$, et $b \in \mathbb{R}^m$ donnés. Lorsque $p = 0$ on parle de programmation entière, sinon de programmation mixte.
Géométriquement, il s'agit de trouver un point appartenant à un polyèdre de $\mathbb{R}^n$, où certaines coordonnées entières, qui minimise une fonction objectif linéaire .
Les problèmes de ce type sont extrêment importants en pratique. Ils permettent de formuler des problèmes concrets de décision optimale sous contraintes de ressources limitées, production minimale à atteindre,.... La programmation entière est un outil indispensable dans l'industrie (pour gérer la production, la logistique, le personnel) et les services (distribution, penser au célèbre problème du voyageur de commerce !), dans la gestion des réseaux (transports, eau, électricité, communication), etc. Résoudre les problèmes de programmation entière est donc un enjeu très important. On trouve sur Internet [**?**] un ensemble de problèmes-tests qui servent à comparer les solveurs existants et dont de nombreux sont issus du "monde réel".

1.4 Structures polyédrales

1.4.1 Les polyèdres

Definition 1.4.1. *On appelle un polyèdre dans R^n l'ensemble de points satisfaisant un nombre fini d'inégalités.*

$$P = \{x \in R^n : Ax \leq b\}$$

où A est une $m \times n$ matrice et bun m vecteur. Autrement dit c'est l'intersection de m demi-espaces.
On notera $P = (A, b)$ un tel polyèdre.
On appelle polytope un polyèdre borné.

Definition 1.4.2. *Une inégalité linéaire $ax \leq \alpha$ est dite valide pour un polyèdre $P \in R^n$ si elle est vérifiée par tout point de P.*

1.5 Théorie de la complexité

La théorie de la complexité, née à la suite des travaux de Edmonds [**?**] puis de Cook [**?**] et Karp [**?**], a pour objet de lier le nombre de calculs effectués lors de la résolution d'un problème au moyen d'un algorithme donné à la taille des données de ce problème. D'une autre manière, elle offre un cadre d'étude mathématique dans lequel les problèmes peuvent être classés en problèmes faciles ou difficiles. Ici on ne donnera que des généralités. Pour

de plus amples informations sur la théorie de la complexité, consulter le livre de Garey et al[**?**].

1.5.1 Notions de base

On formalise la notion de *problème*, en le décomposant en un couple constitué de paramètres et d'une question (l'objectif).
Une instance **I** est un ensemble de données attribuées aux paramètres. On peut attacher à l'instance **I** un entier $\mu(\mathbf{I})$ qui mesure la longueur de ces données, (nombre de bits nécessaires pour la stocker).

Definition 1.5.1. *Un algorithme est une suite d'opérations élémentaires qui, lorsqu'on lui fournit une instance d'un problème en entrée, s'arrête après exécution de la dernière opération en nous renvoyant un résultat.*

Les deux paramètres les plus importants pour mesurer la qualité d'un algorithme sont le temps d'exécution et l'espace mémoire qu'il utilise. D'où les notions de *complexité en temps* et *complexité en espace*.
Ce qu'on appelle la *complexité en temps* ou simplement *complexité* d'un algorithme correspond à une indication du temps qu'il prendra pour résoudre un problème d'une taille donnée.
La complexité en espace est une fonction qui associe, à la taille d'une instance d'un problème donné, un ordre de grandeur du nombre de cases mémoire utilisées pour les opérations nécessaires à la résolution du problème.

Definition 1.5.2. *Un algorithme est dit de l'ordre de $f(n)$ (noté $\bigcirc(f(n))$) s'il existe un scalaire c et un entier n_0 tels que son temps d'exécution est au plus $c * f(n)$ pour tout $n \geq n_0$.*

Si le temps d'exécution d'un algorithme est borné par une fonction polynomiale en la taille du problème, alors l'algorithme sera dit *polynomial*.

Definition 1.5.3. *Un problème de reconnaissance (ou de décision) est un problème dont la réponse est soit "oui" ou "non".*

Exemple 1.5.1. *Soit G un graphe non orienté, existe-t-il un cycle hamiltonian dans G ?*

Théorème 1.5.1. *[?] Si le problème de décision associé à un problème d'optimisation combinatoire donné est difficile, le problème d'optimisation combinatoire est lui même difficile.*

1.5.2 Classes des problèmes *P* et *NP*

Definition 1.5.4. *Un problème (*P*) est dit appartenir à la classe* P*, si il existe un algorithme polynomial pour le résoudre.*

Les problèmes de la classe P sont dites *faciles*. On peut citer comme exemples : le problème du plus court chemin, le problème du couplage maximum,...

Definition 1.5.5. *Un algorithme non déterministe est un algorithme contenant une instance "choix" qui, opérant sur un ensemble fini, sans spécifier comment ce choix est effectué. Il est caractérisé par le fait que s'il existe (au moins) une manière d'effectuer le choix qui conduit à la réponse oui, c'est suivant cette manière que le choix est fait.*

Les algorithmes non déterministes permettent de définir la classe *NP*.

Definition 1.5.6. *Soit (*P*) un problème de décision et* ***I*** *les instances de ce problème pour lesquelles la réponse est "oui". (*P*) est dit* NP *(Nondeterministic Polynomial) s'il existe un algorithme non déterministe polynomial qui permet de vérifier que la réponse est "oui" pour toute instance de* ***I****.*

Parmi les problèmes *NP* nous distinguons les problèmes de la classe P, et les problèmes *NP complets*.

Definition 1.5.7. *Soit (*P*) un problème de décision et* ***I*** *les instances de ce problème pour lesquelles la réponse est "non". (*P*) est dit* Co-NP *s'il existe un algorithme polynomial qui permet de vérifier que la réponse est "oui" pour toute instance de* ***I****.*

1.5.3 Problèmes *NP* complets

La notion principale de la *NP-complétude* est celle de la réduction polynomiale.

Definition 1.5.8. *On dira que le problème (*P*) se réduit à un problème (*P'*) si :*

1. *Il existe un algorithme* ***A*** *qui transforme les instances (données)* I *de (*P*) en instances* I' *de (*P'*).*
2. *Il existe un algorithme* ***A'*** *qui transforme les solutions* S' *de (*P'*) (pour les données* I'*) en solution* S *de (*P*).*

Definition 1.5.9. *On dira que le problème (*P*) se réduit polynomialement au problème (*P'*)si :*

1. *(*P*) se réduit à (*P'*) ;*
2. *Les algorithmes **A** et **A'** sont polynomiaux.*

Definition 1.5.10. *Un problème de* NP *est dit* NP-complet*, si tout problème de* NP *se réduit polynomialement à lui.*

Definition 1.5.11. *Un problème de satisfaisabilité noté* SAT *est donné par :*

- *un ensemble de variables booléennes $X = (x_1, x_2, ..., x_n)$.*
- *une expression booléenne en terme de ces variables : $E = C_1 \wedge C_2 \wedge ... \wedge C_m$ où chaque clause C_i $(i = 1, ..., m)$ est une expression de $C_i = u_{j_1} \vee u_{j_2} \vee ... \vee u_{j_k}$ et où chaque u_{j_q} est une variable de X.*

Le problème consiste à chercher s'il existe une affectation des variables x_k pour $k = 1, ..., n$ à 0 ou 1 telle que $E = 1$.

Théorème 1.5.2. *[?]Tout problème NP se réduit polynomialement à SAT.*

Cook a été le premier à montrer la *NP-complétude* d'un problème [**?**], celui de la satisfaisabilité, noté *SAT*. La réduction utilisée (souvent appelée réduction générique) repose sur la théorie des langages récursifs et les machines de Turing. Pour plus d'approfondissement sur le concept des machines de Turing voir les travaux de Hopcroft et al [**?**].

Remarque 1.5.1. *La question dans le cas des problèmes d'optimisation combinatoire n'appelle pas une réponse oui ou non, mais la valeur optimale d'une fonction.*

A tout problème d'optimisation correspond un problème de décision : si la question du problème d'optimisation est de décider de l'optimum d'une fonction f, on peut poser la question pour un entier positif quelconque k de l'existence d'une solution pour laquelle la valeur de f est inférieure ou égale à k.
Un problème d'optimisation est dit *NP-difficile* si le problème de décision qui lui est associé est *NP-complet*

1.6 Résolution des problèmes difficiles

Étant donné l'importance des problèmes d'optimisation combinatoire, de nombreuses méthodes de résolution ont été développées en recherche opérationnelle et en intelligence

artificielle. Ces méthodes peuvent être classées sommairement en deux grandes catégories : les méthodes exactes (complètes) qui garantissent la complétude de la résolution ; et les méthodes approchées (incomplètes) qui perdent la complétude pour gagner en efficacité.

1.6.1 Méthodes exactes

Méthode d'énumération implicite "Branch and Bound"

Soit IP un problème d'optimisation combinatoire dont l'ensemble des solutions réalisables est S. Les méthodes "Branch and Bound" (séparation et évaluation) reposent sur le principe "diviser pour régner". Plus précisément, on divise l'ensemble S des solutions réalisables d'un problème d'optimisation combinatoire en sous-ensembles de plus en plus petits afin d'isoler dans l'un de ces sous-ensembles une solution optimale. Le processus peut être représenté par une arborescence d'énumération et d'évaluation :

- la racine correspond au problème de départ IP sur l'ensemble S,
- les fils représentent les sous-problèmes IP^i définis sur des sous-ensembles S_i de S.

L'algorithme "Branch and Bound" est donnée comme suit :

Algorithm 1 Branch and Bound

1: *Initialisation* : $L = \{IP\}, S^0 = S, \overline{z}^0 = \infty$, et $z_R^i = -\infty$.

2: *Le test d'arret* : Si $L = \emptyset$, alors la solution x^0 qui donne $\underline{z}_{IP} = cx^0$ est optimale.

3: *Sélection du problème et relaxation* : Sélectionner un problème IP^i puis le supprimer de L.

Résoudre sa relaxation RP^i. Soit z_R^i la valeur optimale de la relaxation et soit x_R^i une solution optimale, s'il en existe une.

4: *Elagage* :
- Si $z_R^i \leq \underline{z}_{IP}$, aller a l'étape 2.
- Si $x_R^i \notin S^i$, aller à l'étape 5.
- $x_R^i \in S^i$ et $cx_R^i > \underline{z}_{IP}$, soit $\underline{z}_{IP} = cx_R^i$. Supprimer tout les problèmes de L vérifiant $\overline{z}^0 \leq \underline{z}_{IP}$. Si $cx_R^i = z_R^i$, aller à l'étape 2 ; sinon aller à l'étape 5.

5: *Séparation* : Soit $\{S^{ij}\}_{j=1}^k$ une partition de S^i. Ajouter les problèmes $\{IP^{ij}\}_{j=1}^k$ à la liste L, où $\overline{z}^{ij} = z_R^i$ pour tout $j = 1, ..., k$. Aller à l'étape 2.

La méthode "Branch and Bound" a été initialement proposée par Little et al.[**?**] pour résoudre le problème du voyageur de commerce, puis, a été reprise par d'autres auteurs pour proposer différentes variantes. Le principe est de partitionner l'ensemble des solutions

du problème en deux ou plus sous-ensembles, en éliminant les parties ne contenant pas de solution entière réalisable (séparation). A chaque étape, on sélectionne, en utilisant une "stratégie de sélection", un sous ensemble prometteur qui conduit à une meilleure solution réalisable.

Choix de stratégies d'exploration

Soit L une liste de sous problèmes, autrement dit un arbre partiel de sommets actifs, une question s'impose : quelle est le sommet (noeud) qui doit être exploré en premier ? L'exploration des sommets de l'arborescence obéit à une stratégie donnée. Dans la littérature, on distingue trois types de stratégies :

- **Profondeur d'abord :** La stratégie "profondeur d'abord" sépare un sommet tant que c'est possible, i.e, jusqu'au moment où on coupe sa branche, pour revenir après à une autre branche. A chaque étape, on change de niveau.
- **Largeur d'abord :** Cette stratégie résout d'abord les problèmes d'un même niveau, puis sépare tous les sommets (séparables) pour résoudre après les problèmes du niveau suivant.
- **Meilleur d'abord :** Contrairement aux stratégies citées ci-dessus, celle ci n'a pas un mode d'exploration bien établi. On sépare le sommet ayant la meilleure valeur de la fonction objectif. A partir d'une liste de sommets (nœuds) actifs L, on choisit un sommet $i \in L$ qui maximise $\overline{z}^i$.

Méthodes de coupes

On considère un problème d'optimisation combinatoire de la forme :

$$\max\{cx : x \in S\}, \text{tel que } \; S = \{Ax \leq b, x \;\; entier\} \tag{1.1}$$

Le problème relaxé est :

$$\max\{cx : x \in S'\}, \text{tel que } \; S' = \{Ax \leq b, x \geq 0\} \tag{1.2}$$

- Si la solution optimale x^0 du problème (1.2) est entière alors x^0 est aussi solution optimale du problème (1.1).
- Si la solution optimale du problème (1.2) n'est pas entière, il doit exister une inégalité valide (a', b') (qui soit vérifiée par toute solution de (1.1)) qui n'est pas vérifiée

par x^0. Cette contrainte peut être ajoutée au système $Ax \leq b$ et cela nous permet d'obtenir une plus "forte" relaxation. Une telle contrainte est appelée une "coupe".

L'étape cruciale de la méthode est la génération de ces inégalités valides (contraintes vérifiées par toute solution de (1.1)). L'une des premières techniques pour identifier (générer) ces contraintes a été introduite par *Gomory* [**?**] et [**?**]. En se basant sur cette technique *Chvàtal* [**?**] a développé une procédure générale pour générer ce genre de contraintes. La procédure est basée sur le fait que si p est entier et $p \leq \lfloor p \rfloor$.

La procédure de "Chvàtal-Gomory" est souvent utilisée pour identifier des inégalités valides pour le polytope associé à un problème d'optimisation combinatoire. Ces contraintes seront alors utilisées dans un *algorithme de coupes*. L'algorithme de coupes "élémentaire" peut être illustré comme suit.

Algorithm 2 Algorithme de coupe

1: *Initialisation* : Résoudre le problème (1.2), soit x^* la solution de (1.2). Aller à l'étape 2.

2: *Test d'optimalité* : Si la solution x^* est entière, stop. Sinon, aller à l'étape 3.

3: *Coupe et pivot* : Générer une inégalité valide pour ILP par une des méthodes de génération d'inégalité, ajouter l'inégalité valide au problème (1.2). Et aller à l'étape 1.

Méthodes "Branch and Cut"

Les algorithmes basés sur les techniques polyédrales sont généralement utilisés dans le cadre de méthodes arborescentes connues sous le nom d'algorithmes de *coupe et branchement* (*Branch and Cut algorithms*). L'idée générale de la méthode de coupes est de résoudre un programme en nombres entiers comme une séquence de programmes linéaires. On considère la relaxation linéaire du problème (le programme linéaire obtenu en relâchant les contraintes d'intégrité du problème). Si la solution optimale, disons x^*, de ce programme est entière, elle est alors optimale. Si ce n'est pas le cas, alors il doit exister une contrainte valide pour le problème qui soit violée par x^*. Une telle contrainte peut être rajoutée au programme pour couper une partie inutile du polyèdre de la relaxation contenant x^*. On détermine alors une (ou plusieurs) contraintes violées par x^* (coupe) que l'on ajoute au programme. Si la solution optimale du nouveau programme est entière, alors elle est optimale. Sinon, on détermine de nouvelle contraintes violées et ainsi de suite. Cette procédure continue jusqu'à ce qu'on trouve une solution entière et donc optimale,

ou alors il n'est plus possible de générer de contraintes violées. Dans ce cas, on utilise une technique par séparation et évaluation (branch and bound) pour déterminer une solution optimale. On choisit une variable fractionnaire x_i, et on divise le problème en deux sous-problèmes en fixant $x_i = 0$ pour l'un et $x_i = 1$ pour l'autre. On détermine une borne supérieure (inférieure) pour chaque sous-problème en résolvant la relaxation linéaire du problème. Si pour un des sous-problèmes, la solution optimale est entière, on arrête son exploration. Sinon, on choisit un des sous-problèmes encore explorables (une des feuilles de l'arbre de résolution) et on sépare en deux sous-problèmes, et ainsi de suite. La procédure s'arrête lorsque toutes les feuilles de l'arbre ne sont plus exploitables. La meilleure solution trouvée sera optimale. Pour calculer une borne pour chaque sous-problème de l'arbre, on peut ajouter des contraintes violées. On peut ainsi améliorer la borne et accélérer davantage la résolution du problème. Un algorithme de coupes et branchements (Branch and Cut algorithm) est une technique de séparation et évaluation dans laquelle on applique l'algorithme de coupes pour calculer la borne de chaque sous-problème. Cette méthode introduite par Padberg et Rinaldi [**?**] pour le problème du voyageur de commerce s'est avérée très efficace, et est maintenant largement utilisée pour résoudre d'une manière exacte des problèmes d'optimisation combinatoire.

La programmation dynamique

La programmation dynamique, due à Richard Bellman [**?**], a été utilisée pour résoudre des problèmes d'optimisation combinatoire tels que le problème du sac à dos.

La programmation dynamique utilise la propriété qu'une solution optimale se compose nécessairement de sous-solutions optimales (le contraire n'est pas vrai en général) pour décomposer le problème en évitant l'explosion combinatoire.
Cette approche a permis de mettre en évidence des algorithmes de résolution pour des problèmes combinatoires définis sur des graphes obtenus par composition. En général, la complexité de ces algorithmes est polynomiale en taille du graphe.

1.6.2 Méthodes approchées

Les méthodes approchées consiste à trouver une solution réalisable d'un problème d'optimisation. En effet, ces méthodes sont utilisées depuis longtemps par de nombreux praticiens. On distingue les méthodes *heuristiques*, et les méthodes α-*approximation*.

Les heuristiques

Une heuristique est un algorithme de résolution fournissant nécessairement une solution réalisable, pour un problème d'optimisation donné.
Outre les heuristiques spécifiques à un problème d'optimisation donné, il existe plusieurs *métaheuristiques* pouvant être adaptées à de nombreux problèmes de concepts fondamentaux (par exemple, la liste tabou et les mécanismes d'intensification et de diversification pour la métaheuristique tabou), qui permettent d'aider à la conception de méthodes. Ainsi, les métaheuristiques sont adaptables et applicables à une large classe de problèmes. Elles sont représentées essentiellement par les méthodes de voisinage comme le recuit simulé et la recherche tabou, et les algorithmes évolutifs comme les algorithmes génétiques et les stratégies d'évolution. Grâce à ces méthodes, on peut proposer aujourd'hui des solutions approchées pour des problèmes d'optimisation classique de plus grande taille et pour de très nombreuses applications qu'il était impossible de traiter auparavant.

Les méthodes α-approximation

Pour certaines heuristiques, il est possible de garantir, pour n'importe quelle instance I du problème, une qualité minimale de la solution trouvée. On parle alors d'*approximations*. La qualité de la solution est mesurée par une performance qui peut être relative ou absolue. Notons, $\Phi_H(I)$ et $\Phi^*(I)$ la valeur fournie par l'heuristique H et la valeur optimale du problème pour l'instance I respectivement. H a une performance absolue α si $|\frac{\Phi_H(I)}{\Phi^*(I)}| = \alpha$ pour toute instance I.
Une heuristique de performance absolue α est une $\alpha - approximation$

1.7 Conclusion

Plusieurs problèmes issus de domaines divers tels que l'industrie, le transport, l'économie et autres se ramènent à des problèmes d'optimisation combinatoire. Un problème d'optimisation combinatoire peut être défini comme étant celui de déterminer un plus petit (ou un plus grand) élément d'un ensemble fini. A première vue, un tel problème paraît très facile à résoudre vu le caractère fini du nombre de ses solutions. Mais en pratique, le nombre de ces solutions peut être exponentiel. Et dans ce cas, une méthode qui consisterait à énumérer toutes les solutions ne peut être envisagée. Plusieurs approches ont, par conséquent, été développées pour ces problèmes comme la programmation linéaire, la programmation en nombres entiers et les approches polyédrales.

Une des méthodes récentes et puissantes pour résoudre ces problèmes est l'approche polyédrale. Une méthode introduite par Edmonds [**?**] pour le problème de couplage, cherche à décrire l'enveloppe convexe des solutions du problème par un système d'inégalités linéaires et donc se ramener à la résolution d'un programme linéaire. Dantzig a été le premier à avoir proposé un algorithme de résolution (méthode du simplexe) pour résoudre ce type de programmes. Ce passage de l'optimisation sur un ensemble discret à l'optimisation sur un domaine convexe a permis un nouvel essor de l'optimisation combinatoire et a propulsé en puissance l'approche polyédrale.

2

Revue de la littérature sur le VRP

Contents

2.1 Introduction

La recherche en transport a beaucoup évolué au cours des dernières années. Les entreprises ont découvert qu'une meilleure planification des tournées permettrait d'épargner des coûts importants. Effectivement, les coûts de transport effectués à travers la chaîne logistique et durant le processus de vente représentent généralement 10 à 25% du coût total de production. Cela peut représenter jusqu'à 57% des coûts logistiques, surpassant ainsi les coûts de stockage (31%) et d'entreposage (8%). Les problèmes étudiés en transports sont variés. Ce chapitre effectuera un survol des plus récentes publications concernant les problèmes de tournées de véhicules ainsi que leurs principales généralisations. Chaque article est traité à la fois par rapport à sa problématique et à l'algorithme utilisé pour trouver une solution. Ce chapitre permettra d'explorer les grandes tendances des dernières années tant au niveau des problèmes que des algorithmes utilisés. L'étude des problématiques étudiées permettra de cibler l'avancement des recherches sur les problèmes de transport tandis que l'étude des algorithmes permettra de déterminer les méthodes de résolution les plus efficaces.

2.2 Définition du problème

La problématique du transport n'est pas une préoccupation récente et de nombreuses entreprises doivent le confronter tous les jours. Le problème de base de transport consiste à affecter en ordre les clients à une seule route. Or, cela ne reflète qu'une minorité d'entreprises puisque dans la majorité des cas, elles doivent prendre une panoplie de décisions à savoir dans quel ordre desservir un ensemble de clients en utilisant une flotte de véhicule (ou un seul véhicule) à partir d'un ou plusieurs entrepôts. À cela s'ajoute les décisions d'affectation des clients et des camions aux différentes routes.

Plusieurs variantes de ce modèle sont traitées dans la littérature. Effectivement, de nombreuses contraintes peuvent y être ajoutées ou ôtées. Ainsi, l'entreprise peut posséder un seul ou plusieurs véhicules. Ces derniers peuvent être identiques ou hétérogènes à des niveaux distincts, soient les coûts d'utilisation, la capacité de chargement, etc. La demande peut se trouver sur les arcs ou sur les sommets selon le type de problème auquel nous faisons face. Chaque fois que l'on ajoute ou que l'on ôte une contrainte, un nouveau problème apparaît. Tous ces ajouts permettront de se rapprocher de la réalité à laquelle une entreprise peut être confrontée. Les sections suivantes tenteront de faire la lumière

sur tous ces types de problèmes apparaissant dans la littérature des dernières années.

Notons que peu importe les contraintes du problème, il demeure toujours NP-Difficile ce qui signifie qu'aucun algorithme connu ne peut garantir de trouver en un temps polynomial une solution exacte à ce problème.

2.3 Problèmes de tournées de véhicules et généralisations

Le premier papier traitant du *problème de tournée de véhicules* a été publié vers la fin des années 1950 par Dantzig et Ramser (1959). Ce problème, plus souvent appelé *Vehicle Routing Problem* (VRP) a ensuite attiré un grand nombre de chercheurs car il est théoriquement très intéressant. De plus, les applications du VRP sont nombreuses. Ainsi, la plupart des entreprises qui doivent livrer un produit à plusieurs clients font face à ce problème. La littérature du VRP est par conséquent très volumineuse.

Le problème de tournées de véhicules peut être défini comme un problème où de nombreux clients doivent être desservis à partir d'un unique dépôt avec des demandes connues. Mathématiquement, le VRP se définit sur un graphe $G = (V, A)$ où $V = v_0, ..., v_n$ représente l'ensemble des points, c'est-à-dire des clients à visiter et $A = (v_i, v_j) : v_i, v_j \in V, i \neq j$ représentant l'ensemble des arcs possibles (routes entre les clients). Le point v_0 représente le dépôt qui est le point de départ et d'arrivée de toutes les routes. Une distance d_{ij} est associée à chaque arc$(i, j) \in A$, ces distances sont symétriques c'est-à-dire que $d_{ij} = d_{ji} \ \ \forall i, j \in A$. On pose comme hypothèses que les véhicules sont identiques avec une contrainte de capacité Q et que les clients ont une demande déterminée q_i. Une limite L peut également être imposée sur la durée maximale des routes. Dans quelques versions du problème, le nombre de véhicules est déterminé a priori. Dans les autres, le nombre de véhicules est une variable de décisions. Les routes doivent permettre de visiter tous les clients une et une seule fois. De plus, la demande totale de tous les clients d'une route ne peut excéder la capacité Q d'un véhicule. Les véhicules sont affectés aux routes de manière à minimiser l'objectif qui peut être, par exemple, la distance parcourue pour visiter tous les clients.

Il existe de nombreuses formulations du problème de tournées de véhicules. La formulation suivante est tirée de Fisher et Jaikumar [**?**]. Définissons tout d'abord l'ensemble des

variables nécessaires pour effectuer la formulation mathématique.

Paramètres :

K : Nombre de camions disponibles.

n : Nombre de clients à visiter. Les clients sont numérotés de 1 à n et l'entrepôt a le numéro 0.

Q_k : Capacité du camion k.

q_i : Demande du client i.

d_{ij} : Distance entre le client i et j.

Variables de décision :

y_{ik} : Variable de décision binaire qui est égale à 1 si la commande du client i est livrée par le camion k et à 0 autrement.

x_{ijk} : Variable de décision binaire qui est égale à 1 si le camion k voyage de cliente i vers le client j et à 0 autrement.

Formulation :

$$\text{Minimiser } Z = \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{K} d_{ij}x_{ijk} \tag{2.1}$$

Sujet à des contraintes de problèmes d'affectation généralisé :

$$\sum_{i=1}^{n} q_i y_{ik} \leq Q_k \qquad k = 1, ..., K \tag{2.2}$$

$$\sum_{k=1}^{K} y_{ik} = \begin{cases} K & i = 0 \\ 1 & i = 1, ..., n \end{cases} \tag{2.3}$$

$$y_{ik} = 0 \text{ ou } 1 \qquad i = 0, ..., n; k = 1, ..., K \tag{2.4}$$

et à des contraintes du problème du voyageur de commerce (TSP)

$$\sum_{i=0}^{n} x_{ijk} = y_{ik} \qquad j = 0, ..., n; k = 1, ..., K \tag{2.5}$$

$$\sum_{j=0}^{n} x_{ijk} = y_{jk} \qquad i = 0, ..., n; k = 1, ..., K \tag{2.6}$$

$$\sum_{I \in S} \sum_{J \in D} x_{ijk} \leq |D| - 1 \qquad \begin{array}{l} \text{pour tout } D \text{ de } i = 0, ..., n \\ 2 \leq |D| \leq n - 1 \\ k = 1, ..., K \end{array} \tag{2.7}$$

$$x_{ijk} = 0 \text{ ou } 1 \qquad \begin{array}{l} i = 0, ..., n \\ j = 0, ..., n \\ k = 1, ..., K \end{array} \tag{2.8}$$

Cette formulation permet de minimiser la distance parcourue par l'ensemble des camions. La contrainte (2.2) permet de s'assurer que le chargement des véhicules respecte leur capacité. La contrainte (2.3) garantit que chacune des routes débute et se termine au dépôt et que chacun des clients est affecté à un et un seul camion. Les contraintes (2.7) permettent d'éviter les sous-tours et assurent que chacun des clients est visité une et une seule fois. Par conséquent, il s'agit des contraintes utilisées pour un TSP.

Le problème de tournées de véhicules et ses généralisations ont été largement étudié au cours des dernières années. La figure 2.1. illustre les articles les plus récents concernant ce problème.

2.3.1 VRP standard

Des articles traitent du problème de tournées de véhicules à un seul dépôt avec contrainte de capacité. Dans ce cas, un ensemble de clients avec une demande et une adresse connues requièrent une livraison à partir d'un dépôt unique et avec une flotte de véhicules identiques. La demande cumulée des clients se trouvant sur la même route ne doit pas dépasser la capacité des véhicules utilisés. L'objectif consiste à minimiser la distance parcourue par tous les véhicules pour visiter une et une seule fois tous les clients puisque ce problème ne tient pas compte de la limite sur la durée des tournées, il est nommé le "capacitated vehicle routing problem (CVRP)".

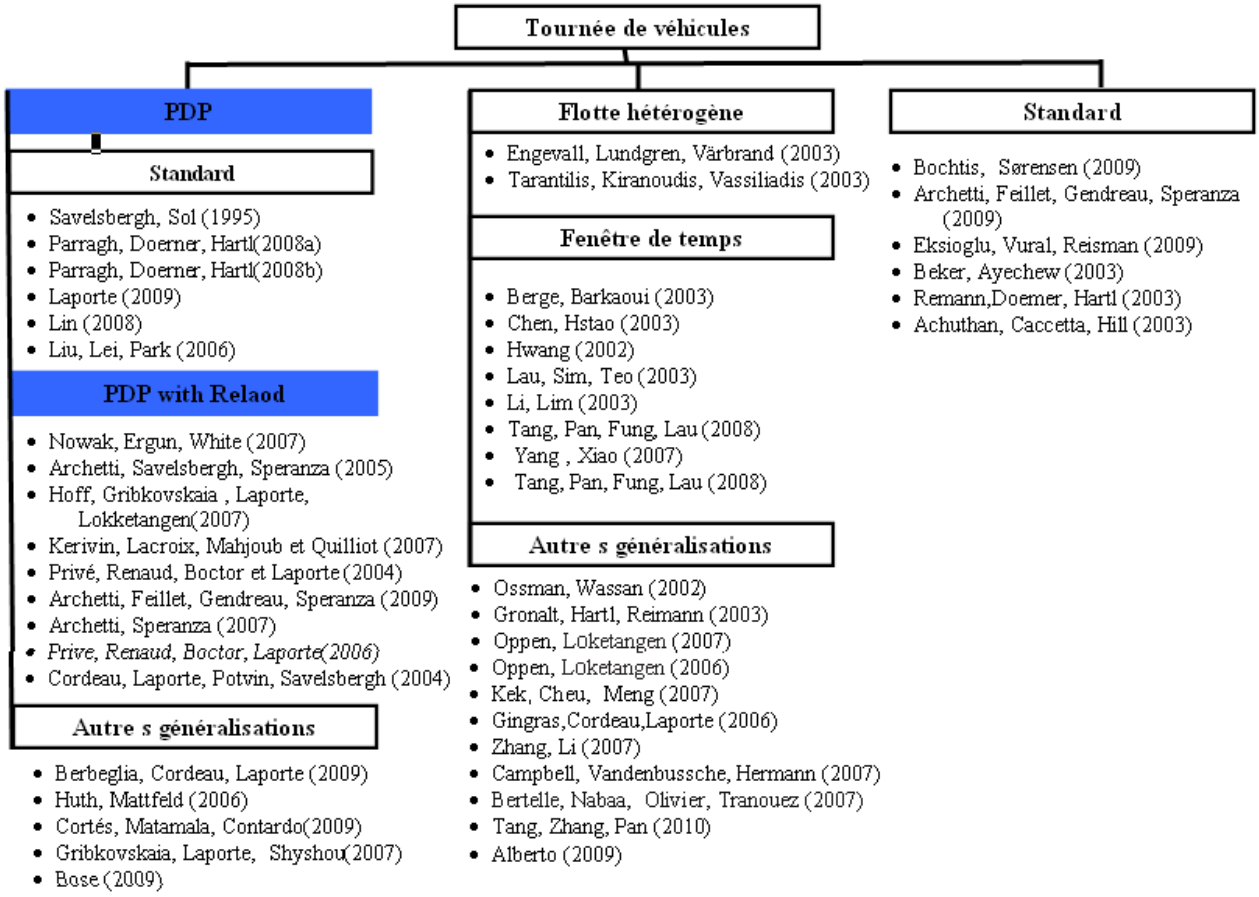

FIG. 2.1 – Classification des problèmes de tournées de véhicules

Toth et al [?] présentent une revue des algorithmes exacts utilisés pour résoudre le CVRP. Dans leur papier, seules les restrictions sur la capacité des véhicules sont imposées et l'objectif consiste à minimiser le coût total (i.e. le nombre de routes et/ou leur longueur ou le temps de route nécessaire pour servir tous les clients). Par conséquent, il n'y a aucune contrainte concernant la durée maximale des routes. On considère deux cas, soit celui où les distances sont symétriques et le second où elles sont asymétriques. Les auteurs présentent deux algorithmes de " branch and bound" pour résoudre ce problème. Les algorithmes exacts permettent de résoudre des problèmes CVRP asymétriques allant jusqu'à 300 points et quatre véhicules. Ces résultats sont obtenus en 1000 CPU secondes. Dans leur revue, Toth et al [?] discutent des algorithmes exacts permettant de résoudre le CVRP symétrique. Ils présentent les algorithmes de Fisher et Miller en soulignant qu'il est difficile de les comparer puisque chacun des auteurs posent des hypothèses différentes sur les mêmes problèmes. Par exemple, Fisher [?] interdit les routes qui visitent un seul

client tandis que Miller [**?**] les permet.

L'article de Tavakkoli-Moghaddam et al[**?**] présente un programme linéaire en nombres entiers de CVRP avec des longueur des routes indépendantes pour minimiser le coût de la flotte hétérogène et maximiser la capacité d'utilisation. Dans leur modèle le coût de la flotte et les longueur des routes sont indépendantes . Le modèle proposé est résolu par l'hybridation de recuit simulé basé sur la recherche voisinage. Ils ont montré que leur modèle est incapable de planifier des routes pour servit tout les client par un nombre minimum de véhicule. L'heuristique proposée peut trouver de meilleurs solutions en temps raisonnable. Ils ont testé l'heuristique sur des problèmes de petite taille et de grande taille.

Étant donné que les algorithmes exacts ne réussissent toujours pas à trouver des solutions pour des problèmes de grandes tailles, plusieurs chercheurs se sont orienté vers des heuristiques. D'énormes progrès ont été effectués ces dernières années concernant les problèmes de tournées de véhicules. Taillard [**?**] a apporté une grande contribution en trouvant la meilleure solution pour la plupart des problèmes de la littérature. Son heuristique consiste essentiellement à décomposer le problème en sous-problèmes et il propose deux méthodes distinctes de partitionnement. Chacun des sous-problèmes est résolu en utilisant une approche tabou. Après un certain nombre d'itérations, les sous-problèmes sont réunis pour reformer le problème entier. Il a remarqué que le type d'algorithme le plus utilisé pour résoudre le problème de tournée de véhicule est celui de la recherche tabou.

2.3.2 Autres variantes

VRP avec flotte hétérogène

Le problème général de tournées suppose que les camions d'une entreprise sont homogènes. Pourtant, il est rare que tous les camions soient identiques. Il existe souvent des distinctions entre les camions, ils n'ont pas tous le même âge, donc ont des coûts variables différents, la capacité de chargement n'est pas la même ou encore il peut s'agir de différences concernant le type de véhicule (i.e. camions réfrigérés, camions citernes). Cet aspect du problème a été traité récemment dans deux articles.

Le papier de Renaud et Boctor [**?**] présente le problème de tournée avec une flotte de

véhicules hétérogène, composée de véhicules avec des capacités différentes. Aussi, les coûts fixes et variables des véhicules peuvent différer d'un véhicule à l'autre. La location d'une partie ou de la totalité de la flotte est possible, procurant ainsi l'avantage d'une grande flexibilité puisque la composition de la flotte peut varier fréquemment. L'objectif consiste à minimiser le coût total composé des coûts fixes et variables de l'utilisation des véhicules. De plus, il faut trouver la meilleure affectation des véhicules aux différentes routes. Dans la version étudiée, les arcs sont non orientés puisque la matrice est symétrique. En plus de respecter les capacités des véhicules tout en répondant aux demandes, une autre contrainte s'ajoute. Effectivement, la durée totale de chacune des routes incluant le temps de parcours et le temps de service ne doit pas excéder une durée maximale. Les auteurs utilisent une heuristique de balayage pour générer différentes routes. Leur heuristique peut résoudre les problèmes euclidiens de même que les problèmes non-euclidiens ce qui est une première pour les algorithmes de balayage. Les résultats, tirés de tests effectués sur 20 problèmes de 50 à 100 points, démontrent que cette heuristique fonctionne mieux que les heuristiques connues dans la littérature. L'heuristique proposée permet d'obtenir un écart de 0,49% en moyenne par rapport à la meilleure solution obtenue par la recherche tabou et ce en un temps de calcul égal à 179 secondes. Aussi, les auteurs proposent une version tronquée de cette heuristique qui permet de réduire le temps de calcul à 154 secondes mais en acceptant une légère augmentation de l'écart se situant maintenant à 0,63%. Cette dernière version peut représenter une alternative intéressante lorsque l'on désire obtenir rapidement une solution.

Tarantilis et al [**?**] approchent ce problème d'une autre façon. Dans leur cas, la flotte comporte plusieurs types de véhicules ayant chacun une capacité variée. Par contre, le nombre de véhicules de chaque type est limité. La flotte est donc fixe, c'est-à-dire qu'on ne considère pas la possibilité pour les entreprises de varier la composition de la flotte de véhicules. L'article décrit une méthode basée sur un seuil d'acceptation appelé " back-tracking adaptive treshold accepting " (BATA). Les problèmes étudiés comportent entre 50 et 100 points, localisés aléatoirement dans un graphe. Ce problème a été peu étudié dans la littérature. Seulement, Taillard [**?**] l'avait déjà présenté. La méthode BATA donne des solutions en moyenne 0,31% meilleures que Taillard [**?**]. Cet algorithme procure six nouvelles meilleures solutions.

VRP avec Fenêtres de temps

Les problèmes de tournées de véhicules peuvent être complexifiés si on ajoute des fenêtres de temps. En anglais ce problème est nommé Vehicle Routing Problem with Time Windows ($VRPTW$) et il se définit de la façon suivante. Soit $G = (V, A)$ ou $V = \{v_0, ..., v_n\}$ représente l'ensemble des points, i.e. des clients à visiter et $A = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ représentant l'ensemble des arcs possibles. Le point v_0 représente le dépôt qui est le point de départ et d'arrivée de toutes les routes. On pose comme hypothèse que les véhicules sont identiques avec une contrainte de capacité Q et que les clients ont une demande déterminée q_i. Chaque client a un temps de service δ_i. Le début de la visite du client v_i doit être à l'intérieur de la fenêtre de temps $[e_i, f_i]$. Cela signifie que le client désire être visite à une période déterminée de la journée. Ainsi, un client peut préférer recevoir ses colis durant les heures d'ouverture de son entreprise. Cet ajout au problème limite la flexibilité car deux clients voisins peuvent être disponibles à deux périodes très différentes de la journée. Par contre, l'ajout de la contrainte de fenêtres de temps est très utile pour un client qui sait ainsi à quelle heure son coli lui sera livré et qui pourra donc prévoir cette réception dans la planification des tâches prévues à l'horaire de la journée. Le coût considéré lors du $VRPTW$ n'est pas seulement le coût de parcours de la distance entre tous les clients mais également le coût associé au temps d'attente car lorsqu'un véhicule arrive en avance chez un client il doit attendre que celui-ci soit prêt à le recevoir.

Les auteurs Tang et al [**?**] traitent d'une variante du problème VRP avec fenêtres de temps dans lequel un nombre limité de véhicules est disponible (m-VRPTW). Les auteurs proposent une approche tabou avec une liste permettant de conserver les clients non visités ainsi qu'un mécanisme introduisant un nouveau véhicule afin de maximiser le nombre de clients visités dans cette route tout en respectant la capacité du véhicule. De plus, ils relaxent les fenêtres de temps en permettant des retards mais avec un coût de pénalité associé à ce retard.

Yang et Xiao [**?**] propose un article qui traite à la fois le VRP avec un seul produit en multi-période et le VRP avec plusieurs produits en une seule période. Ils ont proposé une modélisation mathématique et des algorithmes. Pour le premier problème, ils ont appliqué la programmation dynamique pour séparer le problème en sous-modèles et un algorithme qui combine la méthode en deux étapes et la technique de branch-and-bound est développée pour résoudre les sous modèles. Le deuxième problème est l'étendu du première, ils ont aussi appliqué la même technique pour fournir des solutions. Selon la

comparaison des résultats de tests avec les résultats déclarés de la littérature, ils ont trouvé que l'algorithme amélioré prévoit en effet des solutions avec un degré satisfaisant plus élevés que ceux d'origine et réduit la complexité de calcul et qu'il est possible d'appliquer l'algorithme dans la réalité pour aider les entreprises de logistique.

2.3.3 VRP avec cueillette et livraison

Tout comme le problème de voyageur de commerce, le VRP peut lui aussi permettre d'effectuer différents types d'opérations. Ces opérations peuvent être des cueillettes et/ou livraisons. Lorsque ces deux types d'opérations sont combinés, on doit effectuer les cueillettes avant les livraisons associées. Par ailleurs, il est possible que les chargements soient complets ou partiels selon le poids ou l'espace utilisé dans le camion. Aussi, des contraintes de livraison avec cueillettes au retour peuvent être imposées.

Plusieurs articles récents traitent le problème de tournées de véhicules avec cueillettes et livraisons.
Le document de Parragh et al[**?**, **?**] est le premier volet d'une enquête exhaustive sur les problèmes de ramassage et de livraison. Deux classes de problèmes peuvent être distingués. La première classe, portant sur le transport de marchandises à partir du dépôt et des clients. Cette classe est notée problèmes de tournées de véhicules retours (VRPB). Quatre sous-types peuvent être considérés, à savoir les Problèmes de tournées de véhicules avec retours en cluster (VRPCB), le problème de tournées de véhicule avec retours mixte (VRPMB), les problèmes de tournées de véhicules avec Livraison divisible et ramassage (VRPDDP - les clients les plus exigeants de livraison et de ramassage peuvent être visités à deux reprises), et les problèmes de tournées de véhicules avec cueillettes livraison simultanée(VRPSDP - les clients exigent que les services doivent être visités exactement une fois). La seconde classe, traitée dans la deuxième partie de cette enquête, se réfère à tous ces problèmes lorsque des marchandises sont transportées entre ramassage et livraison. Ce sont les problèmes de tournées de véhicules avec ramassage et livraison (PDVRP - un point de ramassage et des points de livraison), le problème classique de ramassage et livraison (PDP), et le problème de transport des passagers (Dial-A-Ride) (DARP). Des formulations mathématiques sont données pour tous les sept types de VRP avec cueillette et livraison, des méthodes de résolution exactes, heuristiques et métaheuristiques sont discutées.

L'article de Kerivin et al [**?**] traite un problème NP-difficile appelé *The splittable pi-*

ckup and delivery problem with reloads. Ils ont proposé deux formulations de programmes linéaires en nombres entiers mixtes pour le problème et étudié l'approche des plans sécant pour le résoudre. Ils ont identifié quelques inégalités valides. En particulier, ils ont introduit une nouvelle famille de contraintes qui généralise ce qu'on appelle *les inégalités de la capacité résiduelle d'arc*. Leurs inégalités sont utilisables pour le résolution de la formulation de flot multi-produit.

Archetti et Speranza [**?**] traitent le problème classique de tournées de véhicules avec flotte homogène. Chaque client est tenu d'être visité par exactement un véhicule et l'objectif est de minimiser la distance totale parcourue. Dans le problème de tournées de véhicules avec split-livraison (SDVRP) la restriction que chaque client doit être visité une seule fois est éliminée, c'est à dire, les fractions de livraisons sont autorisées. Ils ont présenté une étude de l'état de l'art sur la SDVRP. La complexité du problème et les propriétés structurelles de la solution optimale sont connus. Cependant, alors que certaines heuristiques efficace ont été conçues et testées, les algorithmes exacts proposés jusqu'à maintenant ne peuvent résoudre seulement les cas de très petites tailles.

Archetti et al[**?**] ont prouvé que la complexité du calcul du PRP et du SDVRP sur les structures particulières du graphe sous-jacent, à savoir une ligne, une étoile, un arbre et un cercle, à la fois dans le cas de la flotte limitée et illimitée est toujours NP-Hard, sauf dans une seule exception de l'étoile et flotte illimitée. La NP-Difficulte est montrée par une transformation à partir du problème de partition. Le SDVRP n'est pas seulement résoluble polynomialement sur une étoile avec la flotte illimitée, mais également sur les lignes et les cercles, dans le cas de flottes limitées et illimitées. Leurs résultats suggèrent que la SDVRP ne peut être plus difficile à résoudre que les VRP.

Le problème de livraison multi-produits est un problème opérationnel commun à de nombreuses industries, comme le transport. Liu et al[**?**] présente une étude sur le problème de livraison de plusieurs produits qui contribue à la littérature dans les deux aspects suivants. Premièrement, ils introduisent une nouvelle variante du problème split-livraison, ce qui implique de plusieurs produits de différentes tailles dans chaque commande, cette variante ne peuvt pas être résolue directement par l'algorithme de la répartition actuelle à un seul produit à livrer. Deuxièmement, ils présentent plusieurs nouveaux résultats théoriques qui révèlent les propriétés de ce type de problèmes de split-livraison.

Ils ont également évalué de façon empirique la performance d'un algorithme de recherche qui résout ces problèmes de split-livraison multi-produits via 14.000 cas de test générés de manière aléatoire. Les résultats de cette étude peuvent également bénéficier de la conception de la planification des heuristiques pour plusieurs problème, mais plus générale, les problèmes de split-livraison et multi-produits , tels que, le problème de tournées de véhicule avec capacité et flotte hétérogènes et celles soumises à des fenêtres de temps de réception.

2.4 Cas réel d'application pratique de problème de tournées de véhicules

L'article de Privé et al[**?**] décrit et résout un problème réel complexe apparaissant dans la distribution des boisson non alcoolisées. Un dispositif distinctif de ce problème est que les véhicules doivent livrer des produits et ramasser les matériaux recyclables à l'emplacement des clients. Ils ont modélisé ce problème comme un programme en nombres mixtes et ils ont proposé trois heuristiques pour sa résolution. Une heuristique constructive et deux heuristiques Petal sont développées. Elles sont testées sur des cas réels et sur 10 instances générées aléatoirement. Les résultats obtenus sur les cas réels montrent que une réduction de 23% de distance peut être atteinte.
Bien que cet article ne soit pas récent, Privé et al[**?**] décrivent une application dans l'industrie des boissons. Cette application nous intéresse puisqu'elle est celle qui est la plus près du sujet que nous traitons dans ce mémoire.

2.5 Algorithmes

Dans le but de résoudre efficacement les problèmes présentés dans les sections précédentes, divers algorithmes ont été proposés. Le choix d'un algorithme repose évidemment sur le type de problème mais il est aussi primordial de considérer le temps de calcul et l'effort nécessaire pour trouver une solution. Dépendamment du choix de l'algorithme utilisé, nous obtiendrons une solution de qualité différente demandant un effort différent. Habituellement, la qualité de la solution est corrélée avec l'effort nécessaire pour l'obtenir.

Premièrement, nous pouvons séparer les algorithmes en deux catégories soit : les algo-

rithmes exacts et les heuristiques. La principale distinction à faire entre ces deux approches est la suivante, l'algorithme exact permet de trouver l'optimum tandis que l'heuristique s'en approche sans garantir de le trouver mais avec un moindre effort.

Les prochaines sections présentent brièvement les tendances dans le développement des algorithmes pour les problèmes de tournées de véhicules. Nous nous sommes restreints aux publications des cinq dernières années.

2.5.1 Méthodes exactes

Comme il a été mentionné précédemment, un algorithme exact permet de trouver une solution optimale. Or, cela exige souvent un temps de calcul important puisque implicitement elle consiste à énumérer l'ensemble des solutions possibles. Ainsi, comme le temps de calcul risque d'augmenter exponentiellement avec la taille du problème, il n'est pas rare que ces méthodes rencontrent des difficultés lorsque la taille du problème augmente. On peut diviser les méthodes exactes selon trois différentes approches : la méthode des plans de coupe, la programmation dynamique et la séparation et l'évaluation progressive.

Génération de coupes

La génération de coupes, en anglais "branch and cut", est une généralisation de la séparation et l'évaluation progressive. Plusieurs coupes y sont en fait générées afin de restreindre l'espace des solutions et ainsi augmenter la valeur de la relaxation linéaire. Cet algorithme a été utilisé dans deux articles traitant de localisation de la demande sur les arcs.

Lysgaard[**?**] a introduit une nouvelle classe de coupes appelé coupes d'accessibilité (R-coupes). En particulier, une relation dominante existe entre la R-coupes et les inégalités k-chemin, tel que tout R-coupe domine une ou plusieurs inégalités k-chemin. Ses résultats numériques ont été réalisées avec un algorithme de plan coupant qui implique les inégalités du capacité, les inégalités du tournoi, et R-coupes. A travers ses résultats, il a montré que les inégalités du capacité et de R-coupes donnent des bornes meilleurs que celles obtenues par les inégalités du capacité et les inégalités du tournoi. Donc, il a montré que la nouvelle classe de coupes est compétitive dans le traitement des aspects temporels des instances de VRPTW. Enfin, une procédure de séparation avec un temps raisonnables d'exécution

a été proposé pour la nouvelle classe de coupes.

Dans l'article de Arbex Valle et al[**?**], ils ont étudié un problème de tournées de véhicule non capacité (VRP), où pas nécessairement de visiter tous les clients et le but est de minimiser la durée de la plus longue route du véhicule. Ils ont proposé une formulation par la programmation en nombres entiers, ils ont utilisé la méthode branch-and-cut (BC) pour résoudre le problème, une méthode de branchement locale (LB) est utilisé à l'intérieur de (BC) pour calculer les bornes supérieures.

Gutiérrez-Jarpa et al[**?**] présentent un algorithme exact branch-and-price pour les problèmes de tournées de véhicules avec livraisons, ramassages sélective et fenêtre de temps. Ils ont conclu que l'algorithme est capable de résoudre cinq variantes du problème : une demande avec des routes mixtes, une demande avec retours, des demandes combiné avec une seul visite, des demandes combinés avec plusieurs visites et des routes mixtes, et des demandes combinées avec plusieurs visites et avec retours. Ils ont résolu d'une façon optimale des instances de 50 clients.

2.5.2 Heuristiques

Les heuristiques ne garantissent pas l'obtention d'une solution optimale mais fournissent en général, une solution dont les performances sont assez bonnes. Les heuristiques de construction permettent de former pas à pas une solution initiale qui pourra par la suite être améliorée grâce aux heuristiques d'amélioration. La solution initiale influe grandement sur la qualité de la solution finale qui sera trouvée, c'est pourquoi la recherche d'heuristiques de construction est très importante. Par ailleurs, lorsqu'un algorithme est composé de plusieurs heuristiques, on dit alors qu'il s'agit d'une heuristique composite.

Heuristiques de construction et composites

Les heuristiques de construction et les heuristiques composites occupent une place importante en optimisation. Un nombre important d'auteurs présente des articles utilisant ce type d'heuristiques pour tous les problèmes de tournées. Effectivement, beaucoup d'articles présentent une heuristique de construction et couvrent presque l'ensemble des problèmes de tournées. Souvent, ces articles présentent une heuristique de construction pour ensuite proposer une heuristique d'amélioration.

L'article de Lu et Dessouky [**?**] présente une nouvelle heuristique de construction pour résoudre le problème MVPDPTW (multi-vehicle pickup and delivery problem with time windows). La nouvelle heuristique d'un TSP asymétrique et une troisième basée sur la combinaison des deux premières. Les auteurs développent des heuristiques de construction pour un graphe orienté dans lequel chacun des arcs a un poids différent. Sept familles de problèmes ayant des caractéristiques différentes ont été étudiées afin de déterminer la performance de ces algorithmes sur différents types de problèmes. Les résultats obtenus sont par conséquent différents d'une famille à une autre. Ainsi, pour tous les problèmes asymétriques du TSPLIB, la première heuristique basée sur le Karp-Steele patching est meilleure se situant en moyenne à 3,36% de l'optimum. Par contre, si l'on considère la troisième famille dans laquelle des poids sont attribués aux arcs alors la troisième heuristique est préférée aux deux autres, et elle se situe à 1,88% de l'optimum. Étant donné que les résultats sont très distincts d'un problème à un autre, il sera approprié de choisir l'heuristique selon la famille auquel ce problème appartient et ce même si généralement, la troisième heuristique donne de meilleurs résultats.

Heuristiques d'amélioration

Les heuristiques d'amélioration débutent avec une solution initiale, générée par une autre méthode, et à l'aide de divers échanges tentent de trouver une meilleure solution. Ce type d'heuristiques peut être divisé en deux selon que l'heuristique soit ou non déterministe. Les heuristiques non déterministes peuvent aussi être subdivisées en plusieurs catégories. Toutes ces méthodes seront traitées dans la prochaine section.

L'article de Girard et al [**?**] propos de nouvelles heuristiques basées sur la méthode des économies de Clarke et Wright. Ces heuristiques ont été développées avec un objectif de simplicité et de rapidité. Ainsi, elles n'utilisent que le calcul classique des économies de Clarke et Wright ainsi que les algorithmes de base d'amélioration 2-opt et 3-opt qui sont facilement accessibles à tous. La première heuristique, CW+, est une procédure de construction qui utilise le concept de perturbation afin de générer un large éventail de solutions. L'heuristique CW+ est très rapide et permet d'obtenir des solutions initiales intéressantes. Ils ont également proposé une approche d'amélioration, DCW+, qui décompose la solution à améliorer en plusieurs sous problèmes de tournées qui sont eux-mêmes améliorés à l'aide de l'algorithme CW+. L'heuristique DCW+ est rapide et offre une per-

formance globale similaire à celle de plusieurs algorithmes tabous connus. Pour l'ensemble des 34 problèmes, DCW+ offre une déviation moyenne de 1,92% en 88 secondes.

Méthodes déterministes

Une heuristique est dite déterministe lorsque à toutes les fois que l'on exécute cette heuristique sur le même problème à partir d'une même solution initiale, on trouve une solution identique. Le hasard n'intervient en aucun cas dans ce type d'heuristique. La recherche dans le voisinage est systématique et déterministe.

Une nouvelle heuristique générale, notée ALNS a été présenté dans l'article de Pisinger et Ropke [**?**]. L'heuristique a été utilisée pour résoudre plusieurs variantes de problèmes de tournées de véhicules telles que le problème de tournées de véhicules avec fenêtres de temps (VRPTW), le problème de tournées de véhicules avec capacité (CVRP), le problème de tournées de véhicules multi-dépôt(MDVRP), le problème de tournées de véhicules avec des sites dépendante(SDVRP), le problème de tournées de véhicules ouvert(OVRP), le problème de livraison et ramassage avec fenêtres de temps (PDPTW), le problème de tournées de véhicules problème avec retours (VRPB), le problème de tournées de véhicules avec retours mixtes (MVRPB), le problème de tournées de véhicules multi-dépôt avec retours mixtes (MDMVRPB), le problème de tournées de véhicules avec retours et fenêtres de temps (VRPBTW), le problème de tournées de véhicules mixtes avec retours et fenêtre de temps (MVRPBTW) et le problème de tournées de véhicules avec les livraisons et ramassages simultanées (VRPSDP).
Grace au généralité de ALNS et les résultats encourageants démontrée pour un large éventail de problèmes VRP, ils pensent que ALNS doit être considéré comme l'un des heuristiques standard pour résoudre les problèmes d'optimisation des grandes entreprises. La gestion de la chaîne d'approvisionnement est un domaine de recherche attirer toujours l'attention. En coordonnant les activités dans la chaîne d'approvisionnement, les entreprises peuvent rationaliser le processus entraînant des gains mutuels. Si les entreprises concernées coordonnent leurs activités de transport, ils auront besoin de résoudre les problèmes de transport mixte. Afin de gérer les changements futurs dans la structure de distribution, ces algorithmes doivent être stables pour les différents types d'entrées. Il doit être clair que l' ALNS est très prometteur pour ces types. En conclusion, ils ont vu qu'un mélange de bonnes heuristiques et de moins bon conduire à des solutions meilleures que d'utiliser de bonnes heuristiques seulement. alors ils ont fourni une approche de résolution robuste et performante.

Méthodes non déterministes

Les méthodes non déterministes peuvent être définies comme des méthodes qui utilisent l'aléatoire dans la sélection des voisinages à explorer ce qui aide à diversifier la recherche. Certaines se basent sur des phénomènes existant dans la nature pour explorer l'ensemble des solutions possibles. Aussi, elles utilisent un processus itératif qui se termine lorsqu'il atteint un critère d'arrêt prédéfini. Il existe une panoplie de critères d'arrêt. Par exemple, ce critère peut être l'obtention de la solution optimale. Si cette dernière est connue, un nombre maximum d'itérations, un temps de calcul écoulé, etc. Le succès de ces méthodes dépend de plusieurs facteurs, comme la facilité d'implantation, l'habilité à adapter les contraintes d'applications réelles et la qualité des solutions produites. Les méthodes non déterministes ont été regroupés selon quatre catégories : la recherche tabou, l'algorithme génétique, le recuit simulé et les autres métaheuristiques.

La recherche tabou

Au cours des dernières années, la recherche tabou a été appliquée au problème de tournées par plusieurs auteurs.

Plus récemment, Bolduc et al[**?**] ont présenté le problème de tournées de véhicule avec un calendriers de demandes et demande partagé, qui est une nouvelle variante du VRP, et d'après eux cette varainte n'a jamais été étudié. Le problème a été modélisé comme un programme mixte en nombres entiers et résolus par la recherche tabou. Leurs résultats démontrer qu'il est possible d'obtenir une excellente solution avec un temps raisonnable. Ils ont également introduit plusieurs stratégies de réduction de voisin qui s'est avérée efficace non seulement à réduire le temps de calcul mais aussi à l'amélioration de la qualité de la solution globale.

L'article de Gribkovskaia et al[**?**] a étudie le problème de tournées de véhicule avec la selection des livraisons et des ramassage (SVRPDSP), un problème rencontrés dans la logistique. Ils ont présenté une procédure de construction et d'amélioration, ainsi que la recherche tabou. Ils ont testé leurs procédures sur un ensemble d'instances de VRPLIB. Les deux versions de leur recherche tabou ont été obtenus avec les heuristiques constructives (CI) et(NN). Enfin, leurs résultats indiquent qu'il est souvent profitable de faire une deuxième visite au même client.

Colonie de fourmis

Les algorithmes de colonies de fourmis sont des algorithmes inspirés du comportement des fourmis et qui constituent une famille de métaheuristiques d'optimisation.

Initialement proposé par Dorigo, Colorni et Maniezzo dans les années 1990, pour la recherche de chemins optimaux dans un graphe, le premier algorithme s'inspire du comportement des fourmis recherchant un chemin entre leur colonie et une source de nourriture. L'idée originale s'est depuis diversifiée pour résoudre une classe plus large de problèmes et plusieurs algorithmes ont vu le jour, s'inspirant de divers aspects du comportement des fourmis.

Le VRP a été un problème important dans le domaine de la distribution et de la logistique. Depuis la déroute de livraison de n'importe quelle combinaison des clients, ce problème appartient à la classe des problèmes NP-difficile. L'article de Bin et al [**?**] présente une IACO avec la stratégie des poids des fourmis et une opération de mutation. Les résultats de calcul de référence du 14 problèmes révèlent que l'IACO proposé est efficace.

Algorithme génétique

Les algorithmes génétiques appartiennent à la famille des algorithmes évolutionnistes. Leur but est d'obtenir une solution approchée à un problème d'optimisation, lorsqu'il n'existe pas de méthode exacte (ou que la solution est inconnue) pour le résoudre en un temps raisonnable. Les algorithmes génétiques utilisent la notion de sélection naturelle et l'appliquent à une population de solutions potentielles au problème donné.

L'utilisation d'algorithmes génétiques, dans la résolution de problèmes, est à l'origine le fruit des recherches de John Holland et de ses collègues et élèves de l'Université du Michigan qui ont, dès 1960, travaillé sur ce sujet. La nouveauté introduite par ce groupe de chercheurs a été la prise en compte de l'opérateur d'enjambement en complément des mutations. Et c'est cet opérateur qui permet le plus souvent de se rapprocher de l'optimum d'une fonction en combinant les gènes contenus dans les différents individus de la population. Le premier aboutissement de ces recherches a été la publication en 1975 de Adaptation in Natural and Artificial System.

L'article de Bae et al [**?**] concerne le développement d'une approche de VRP intégrée pour les problèmes d'approvisionnement multi-dépôt basé sur un algorithme génétique amélioré et la programmation graphique. Dans cet article, ils ont proposé une approche en trois étapes pour convertir un problème d'approvisionnement de multi-dépôt en un problème d'approvisionnement d'un seul dépôt de manière à rendre facile la résolution. Ils ont développé un modèle intégré de VRP utilisant la méthode heuristique et l'algorithme génétique amélioré (GA), dont les opérateurs et la population initiale sont améliorées. Aussi ils ont développé une interface utilisateur graphique (GUI-type), basés sur une approche en trois étapes telles que : (1) module de regroupement pour transformer les problèmes d'approvisionnement multi-dépôt en des problèmes d'approvisionnement d'un seul dépôt qui sont plus faciles à résoudre, (2) l'amélioration des tournées des véhicules, et (3) module de GA-TSP. L'objectif de ce problème est de minimiser le coût logistique pour un ensemble de clients sans être en retard ou dépassement du délai de Voyage et de la capacité des véhicules. Pour le confort de l'utilisateur, ils ont comparé les résultats des testes par rapport à ceux des méthodes existantes.

3

Problème de distribution avec les rechargements

Contents

3.1 Introduction

Le problème de base en transport, probablement le plus étudié, est le problème du voyageur de commerce (traveling salesman problem) qui permet de visiter un ensemble de clients avec un seul véhicule. Le problème consiste donc à trouver l'ordre dans lequel chacun des clients sera visité. De nombreuses contraintes peuvent s'ajouter permettant ainsi de l'adapter à des problèmes pratiques rencontrés dans l'industrie du camionnage. Par exemple, le problème de tournées de véhicules (vehicle routing problem) qui est un autre problème également étudié intensivement, il traite le cas où chacun des clients a une demande déterminée et où la flotte de véhicules est homogène.

Le problème de distribution avec les rechargements (PDR) proposé dans ce mémoire est un problème de tournées de véhicules avec une flotte homogène et plusieurs produits. Il s'agit d'un problème de tournées de véhicules puisque les clients à visiter requièrent une demande connue à l'avance et les véhicules sont limités par leur capacité en terme de poids pour les produits et de volume pour la récupération. On dit que la flotte est homogène puisque la flotte de camions de l'entreprise est constituée de camions ayant des capacités égales. De plus, l'entreprise distribue plusieurs produits ayant des poids différents. Finalement, l'entreprise doit aussi récupérer auprès des clients les bouteilles et canettes vides.

Le problème consiste donc à visiter tous les clients autant de fois que nécessaire afin de leur livrer la marchandise commandée et rapporter la récupération tout en respectant les contraintes de volume et de capacité. L'objectif est de minimiser le coût des déplacements pour visiter tous les clients.

Pour aider le lecteur à comprendre notre problème, nous allons donner quelques exemples illustratifs, après avoir défini notre problématique

3.2 Problématique

Considérons un certain nombre de villes et supposons qu'un ensemble de demandes $p^1, ..., p^k$ et un ensemble de consignes $q^1, ..., q^k$ doivent être acheminées entre ces villes. Chaque demande (ou consigne) est spécifiée par une ville d'origine, une ville de destination, un volume et une capacité. Les demandes (ou les consignes) doivent être acheminées à travers le réseau, de leurs origines à leurs destinations. Pour satisfaire les demandes de

livraison et de collecte nous disposons d'une flotte de véhicules homogène qui partent et reviennent tous au dépôt. Chaque demande $p \in P$ (resp. consigne $q \in Q$) peut être déchargée (complètement ou partiellement) à n'importe quel point intermédiaire (i.e : point différent de sa destination) et peut être alors ramassée plus tard par le même véhicule ou un autre.

Ce processus de déchargement/ramassage appelé rechargement peut être répété plusieurs fois pour une demande (ou consigne) jusqu'à ce que son point de destination soit trouvé.
De plus, chaque demande (ou consigne) peut être divisée sur des routes différentes et peut être acheminée par plusieurs véhicules, qui peuvent passer par n'importe quel sommet ou arc autant de fois que nécessaire.

Lorsqu'un véhicule décharge la demande, il recharge soit immédiatement soit plus tard la consigne (bouteilles ou cannetes vides), dans le but de voir les bouteilles livrées antérieurement dans leurs stops origines. Donc, on conclut que notre problème contient deux types de demandes (la demande $p^1, ..., p^k$ et la consigne $q^1, ..., q^k$).

Un véhicule reliant deux villes entraîne un coût positif. On définit le coût d'une tournée de véhicules comme étant la somme des coûts de toutes les tournées parcourues par les véhicules de la flotte F . De plus, on suppose que ces coûts satisfont les inégalités triangulaires et que le coût et le temps de rechargement sont négligés.

Le Problème de Distribution avec les Rechargement/Déchargement (PDR), consiste alors à trouver les tournées des véhicules de telle sorte que toutes les demandes et consignes soient transportées à leurs destinations, qu'aucun véhicule ne soit surchargé et que le coût total de ces tournées soit minimum.

3.3 Exemples

Considérons d'abord le réseau donné dans FIG 3.1. Supposons qu'on a trois demandes $(p^1, p^2$ et $p^3)$ et trois consignes $(q^1, q^2$ et $q^3)$ à acheminer. Les demandes et les consignes ont des volumes égaux à la capacité de transport du véhicule.
La résolution de l'exemple est faite sous deux conditions ; la première est d'autoriser le rechargement tandis que la deuxième est d'interdire le rechargement. Avec la condition supplémentaire de rapporter les consignes soit au dépôt soit à leur stop origine.

1. **Sans rechargement**
   - **Retour au stop origine**

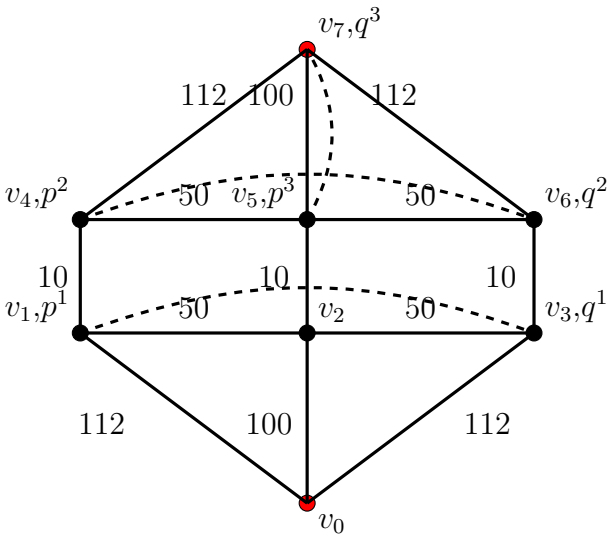

FIG. 3.1 – Exemple 1

Une solution optimale de PDR consiste en premier à charger la demande p^3 qui est dans le somment v_5 et la transporter à sa destination v_7 puis recharger la consigne q^3 pour la décharger en v_5. Ensuite le véhicule va au sommet v_4 où il recharge p^2, la transporte en v_6 à travers (v_4, v_6) puis il retourne en v_4 à travers ce dernier pour décharger la consigne q^2 et passer par (v_4, v_3) et arriver en v_3 pour commencer le transport de q^1. Une fois q^1 décharge à sa destination v_1, il recharge p^1 pour la décharger en v_3, le véhicule retourne directement au dépôt. Le coût associé à cette planification est 684.

- **Retour au dépôt**

  Une solution optimale de PDR consiste en premier à charger la demande p^3 qui est dans le somment v_5 et la transporter à sa destination v_7 puis recharger la consigne q^3. Ensuite le véhicule va en v_6 où il recharge q^2, puis va directement au dépôt. Ensuite le véhicule retourne en v_4 pour transporter p^2 vers v_6 puis à travers (v_6, v_1) le véhicule passe au sommet v_1 pour acheminer la dernière demande p^1 vers v_3 où il décharge p^1 et recharge q^1 pour retourner directement au dépôt. Le coût associé à cette planification est 701.

2. **Avec rechargement**

- **Retour au stop origine**

  Une solution optimale peut être comme suit : le véhicule commence son chemins par v_1 où il charge p^1. Il l'achemine de v_1 à v_3 à travers (v_1, v_3) et ensuite va en

v_4 pour acheminer p^2. Il arrive ensuite au sommet v_5 où il décharge p^2 et recharge p^3. Puis il achemine p^3 de v_5 en v_7 et retourne en v_5 utilisant l'arc (v_7, v_5), chargé de la consigne q^3. Le véhicule continue son chemin pour recharger q^2 et de la décharger en v_4 ; le véhicule retourne en v_4 à travers l'arc (v_4, v_6) charger de q^2 . Puis le véhicule passe en v_3 pour recharger q^1 et la déposer en v_1. A la fin de la distribution, le véhicule retourne au dépôt. Le coût associé est 656.

- **Retour au dépôt**

  Une solution optimale peut être comme suit : le véhicule commence son chemin par v_4 où il charge p^2, passe par le sommet v_5 où il décharge p^2 et charge p^3, puis il achemine p^3 de v_5 en v_7. Le véhicule charge q^3 puis passe par v_6 pour charger aussi q^2 et les transporter au dépôt. Le véhicule repart du dépôt vers v_5 pour terminer la livraison de p^3, ensuite le véhicule passe au sommet v_1 pour transporter p^1 de v_1 à v_3 à travers (v_1, v_3) puis il recharge q^1 et retourne directement au dépôt. Le coût associé à cette planification est 701.

Dans le deuxième exemple présenté dans la figure 3.2, on pend en compte un véhicule de volume égal à 3 et trois demandes, chacune ayant un volume égal à 2.

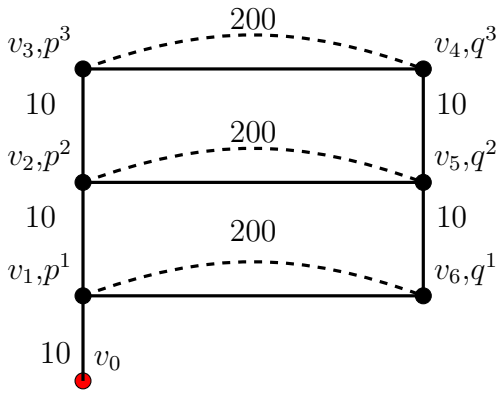

FIG. 3.2 – Exemple 2

1. **Sans rechargement**
   - **Retour au stop origine**

     Une solution optimale de PDR consiste en premier à charger la demande p^1 du somment v_1 et la transporter à sa destination v_6 puis charger la consigne q^1 pour la décharger en v_1. Ensuite le véhicule va en v_2 où il charge p^2, la transporte en v_5 à travers (v_2, v_5) puis il retourne en v_2 à travers ce dernier pour décharger la consigne q^2 et passer par (v_2, v_3) et arriver en v_3 pour commencer le transport de p^3. Une fois que p^3 est décharge à sa destination, il charge q^3 pour la décharger en

v_3, le véhicule retourne directement au dépôt. Le coût associé à cette planification est 684.

- **Retour au dépôt**

  Une solution optimale de PDR consiste en premier à charger la demande p^1 qui est dans le somment v_1 et la transporter à sa destination v_6 puis recharger la consigne q^1 pour la décharger au dépôt. Ensuite le véhicule va en v_2 où il recharge p^2, la transporte en v_5 à travers (v_2, v_5) puis il retourne au dépôt pour décharger la consigne q^2 et passer par (v_0, v_3) et arrive en v_3 pour commencer le transport de p^3. Une fois Une fois que p^3 est décharger à sa destination, il recharge q^3 pour la décharger au dépôt. Le coût associé à cette planification est 1320.

2. **Avec rechargement**

- **Retour au stop origine**

  Une solution optimale de PDR consiste en premier à charger la demande p^1 du somment v_1 et une partie de p^2 qui est au sommet v_2, les transporter à leur destination v_5 et v_6 puis recharger les consignes q^1 et une partie de q^2 pour les décharger en v_1 et v_2 respectivement. Ensuite le véhicule arrive en v_2, il recharge le reste de p^2 et va en v_3 pour charger p^3, les transporter en v_4 et en v_5 puis il retourne en v_3 pour décharger la consigne q^3 et passer par (v_3, v_2) et arriver en v_2 pour décharger le reste de q^2, le véhicule retourne directement au dépôt. Le coût associé à cette planification est 900.

- **Retour au dépôt**

  Une solution optimale de PDR consiste en premier à charger la demande p^1 du somment v_1 et une partie de p^2 qui est au sommet v_2, les transporter à leur destination v_5 et v_6 puis recharger les consignes q^1 et une partie de q^2 pour les décharger au dépôt. Ensuite le véhicule revient en v_2 où il recharge le reste de p^2 et va en v_3 pour charger p^3, les transporter en v_4 et en v_5 puis il retourne au dépôt charger q^3 et reste de q^2. Le coût associé à cette planification est 900.

3.4 Complexité du PDR

Le PDR est un problème **NP-difficile**. En effet, lorsque l'ensemble de consignes est vide, ce qui constitue un cas particulier de PDR, le problème est connu pour être **NP-difficile**[Kerivin et al[**?**]].

3.5 Formulation mathématique

Dans la suite, nous donnons la formulation de PDR en terme de graphes. Cette formulation se fait en deux étapes. On construit d'abord le graphe auxiliaire qui représente le réseau à chaque unité de temps, ensuite la modélisation en programme linéaire en nombres mixtes.

3.5.1 Graphe auxiliaire

Cette partie est consacrée à la construction d'un graphe auxiliaire, noté $G' = (V', A')$ à partir du graphe initial G. Ainsi, il n'y a pas de demande incidente au dépôt, ni de rechargement et la fonction de coût satisfait l'inégalité triangulaire. Nous remarquons ici que les véhicules ne passent par le dépôt que lorsqu'ils commencent ou terminent leur circuit. Donc chaque véhicule passe par v_0 au plus deux fois. Cela nous mène à séparer le dépôt des autres sommets dans la construction du graphe auxiliaire.

Pour chaque sommet $u \in V\backslash\{v_0\}$, on associe $T+1$ sommets $u_0, u_1, ..., u_T$ dans G'. Le sommet $u \in V\backslash\{v_0\}$ est visité à l'instant $t \in \{0, ..., T\}$ (T est la durée maximale pour visiter tout les clients). On note V_{st} cet ensemble de sommets. Ensuite on prend en considération trois ensembles d'arcs dans G' ; le premier ensemble est défini par $A_T = \{(u_t, u_{t+1})/u \in V\backslash\{v_0\}, t \in \{0, ..., T-1\}\}$. Un arc de A_T correspond au véhicule ou à la demande (consigne) qui reste dans un sommet pour une unité de temps. Le deuxième est $\widetilde{A} = \{(u_t, w_{t+l_{(u,w)}})/(u,w) \in A\backslash\delta(v_0), t \in \{0, ..., T - l_{u,w}\}\}$. Un arc $a = (u_t, w_{t'}) \in \widetilde{A}$ correspond au véhicule qui va du sommet u à la date t au sommet w à la date t' avec $t' = t + l_{(u,w)}$. Les arcs de A_T ont des coûts nuls (ainsi on suppose que le coût correspondant au fait de rester dans un sommet du réseau est nul). Cependant un arc $a = (u_t, w_{t'}) \in \widetilde{A}$ a un coût égal à $C_{(u,w)}$.

On ajoute deux sommets O et D qui représentent l'origine et la destination des itinéraires des véhicules. Ces deux sommets correspondent au dépôt dans le graphe initial. Sans perte de généralité, on oblige tous les véhicules du parc à commencer leurs chemins à la date 0 et de le terminer à la date T. Pour tous les sommets $u \in V\backslash\{v_0\}$, on ajoute les arcs (O, u_0) et (u_T, D) avec le même coût et la même durée, égaux à ceux des arcs (v_0, u) et (u, v_0) dans G. Le troisième est défini par les ensembles $A^O = \{(O, u_0)/u \in V\backslash\{v_0\}\}$ et $A^D = \{(u_T, D)/u \in V\backslash\{v_0\}\}$. Pour conclure, on construit un graphe auxiliaire $G' = (V', A')$ avec $V' = V_{st} \cup \{O, D\}$ et $A' = A_T \cup \widetilde{A} \cup A^O \cup A^D$.

La figure suivante (FIG. 3.3) illustre la construction du graphe auxiliaire G' associé au

graphe initial $G = (V, A)$. Dans cet exemple, on suppose que tous les arcs $a \in A$ ont des durées l_a égales à 1 et T égal à 2. Commençons avec un graphe initial ayant 4 sommets et 9 arcs, on arrive à un graphe auxiliaire ayant 11 sommets et 18 arcs.

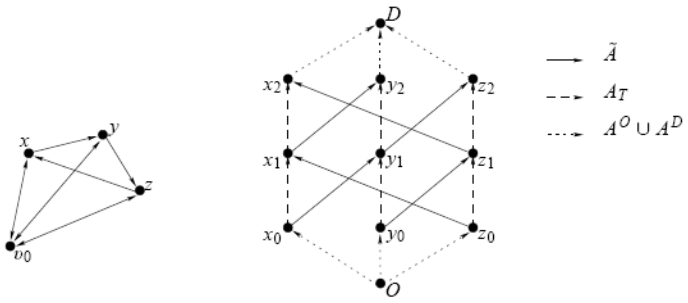

FIG. 3.3 – graphe initial et son graphe auxiliaire

3.5.2 Formulation en nombres mixtes

Dans cette formulation, nous avons trois types de variables :

- $y \in \mathbb{Z}_+^{|A|}$ où y_a représente le nombre de véhicules passant sur l'arc $a \in A'$.
- $x \in \mathbb{R}_+^{|\widetilde{A} \cup A_T| \times |P|}$ où x_a^p représente la quantité de demande à acheminer sur l'arc $a \in \widetilde{A} \cup A_T$.
- $z \in \mathbb{R}_+^{|\widetilde{A} \cup A_T| \times |Q|}$ où z_a^q représente la quantité de consigne à acheminer sur l'arc $a \in \widetilde{A} \cup A_T$.

Pour un sommet u_t avec $u \in V \backslash \{v_0\}$ et $t \in \{0, ..., T\}$, et une demande $p \in P$, on considère le nombre $b_{u_t}^p$ défini par

$$b_{u_t}^p = \begin{cases} k^p & \text{si } u = o^p \text{ et } t = o \\ -k^p & \text{si } u = d^p \text{ et } t = T \\ 0 & \text{sinon} \end{cases}$$

Et pour une consigne $q \in Q$, on considère le nombre $b_{u_t}^q$ défini par

$$b_{u_t}^q = \begin{cases} k^q & \text{si } u = o^q \text{ et } t = o \\ -k^q & \text{si } u = d^q \text{ et } t = T \\ 0 & \text{sinon} \end{cases}$$

Ces deux nombres représentent respectivement l'offre et la demande associées aux sommets du graphe. On remarque que pour un $p \in P$ (resp. $q \in Q$) donné, il y a exactement deux sommets de V_{st} pour lesquels b_v^p (resp. b_v^q) est non nul.

Le PDR peut se formuler comme un programme linéaire en nombres mixtes comme suit :

$$
PDR \begin{cases}
\min \sum_{a \in A'} c_a y_a & & \\
\text{s.t} & & \\
\sum_{a \in A'} y_a \leq |F| & & (1) \\
\sum_{a \in \delta^+(v)} y_a - \sum_{a \in \delta^-(v)} y_a = 0 & \forall v \in V_{st} & (2) \\
\sum_{a \in \delta^+(v) \setminus A^D} x_a^p - \sum_{a \in \delta^-(v) \setminus A^O} x_a^p = b_v^p & \forall v \in V_{st}, \forall p \in P & (3) \\
\sum_{a \in \delta^+(v) \setminus A^D} z_a^q - \sum_{a \in \delta^-(v) \setminus A^O} z_a^q = b_v^q & \forall v \in V_{st}, \forall q \in Q & (4) \\
\sum_{p \in P} x_a^p + \sum_{q \in Q} z_a^q - B y_a \leq 0 & \forall a \in \widetilde{A} & (5) \\
\sum_{q \in Q} z_a^q - E y_a \leq 0 & \forall a \in \widetilde{A} & (6) \\
x_a^p \geq 0 & \forall a \in \widetilde{A} \cup A_T, \forall p \in P & (7) \\
z_a^q \geq 0 & \forall a \in \widetilde{A} \cup A_T, \forall q \in Q & (8) \\
y_a \geq 0 & \forall a \in A' & (9) \\
y_a \text{entier} & \forall a \in A' & (10)
\end{cases}
$$

La fonction objectif affirme que le coût total relatif au véhicule doit être minimisé. (On remarque que cette fonction objectif peut facilement s'étendre si on considère que les coûts sont proportionnels à la quantité de demande (consigne) acheminée sur les arcs).

La contrainte (1) limite à $|F|$ le nombre de véhicules utilisés. La contrainte (2) concerne la conservation de flot des véhicules. Les contraintes (3) et (4) assurent la conservation des demandes et consignes respectivement. La contrainte (5) impose que la quantité des demandes et consignes acheminées sur un arc de $\widetilde{A}$ ne doit pas dépasser la capacité totale des véhicules passant sur ces arcs. La contrainte (6) impose que le volume des consignes acheminées sur un arc de $\widetilde{A}$ ne doit pas dépasser le volume total des véhicules passant sur ces arcs.

Ce modèle contient $|A'|+|\widetilde{A} \cup A_T| \times (|P|+|Q|)$ variables et $|V_{st}| \times (|P|+|Q|+1)+2|\widetilde{A}|+1$ contraintes.

Remarque 3.5.1. *Il existe d'autres formulations de ce problème, telle que la formulation par les contraintes métriques. Cela peut se faire par la considération d'une seule variable y_a pour tout $a \in A'$ et le remplacement des contraintes (3), (4), (5) et (6) par ce qu'on*

*appelle les contraintes métriques[**?**].*

3.6 Résolution

Cette section est consacrée à la présentation des méthodes de résolution. Dans un premier temps, une présentation des coupes proposées pour la méthode "Branch and Cut", suivie d'une présentation de l'algorithme "Branch and Bound" utilisé pour notre problème.

3.6.1 Coupes proposée

On présente d'abord trois familles de contraintes qu'on va utiliser dans notre algorithme pour renforcer la relaxation linéaire.

- La première peut être utilisée par les deux formulations.
- La deuxième et la troisième sont basées sur la valeur du flot des variables associées aux demandes et consignes (i.e. variables x et z) et ne peuvent pas être utilisées dans la la formulation des contraintes métriques.

Contraintes de coupes

Dans cette partie, on introduit des contraintes appelées *contraintes de coupes* qu'on va utiliser dans notre algorithme "branch-and-cut" pour renforcer les deux relaxations linéaires de PDR. Ces contraintes sont déjà utilisées dans plusieurs problèmes d'optimisation. (voir (Bienstock et al.[**?**]),(Barahona [**?**])).

Elles sont basées sur la notion de coupe dans G'. Pour un sous ensemble de sommets $W \subseteq V_{st}$, soit $l\ [W,\ V_{st}\backslash W]$ (resp. $h\ [W,\ V_{st}\backslash W]$) le volume total des demandes (resp. consignes) de P (resp. Q) ayant leurs origines dans W et leurs destinations dans $V_{st}\backslash W$.

Proposition 3.6.1. *Soit $W \subseteq V_{st}$ un sous-ensemble de sommets et $\delta^+(W)$ la coupe correspondante telle que $\delta^+(W) \cap A_T = \emptyset$. Alors les contraintes de coupes*

$$y(\delta^+(W)) \geq \left\lceil \frac{l\ [W,\ V_{st}\backslash W]}{B} \right\rceil \tag{3.1}$$

$$y(\delta^+(W)) \geq \left\lceil \frac{h\ [W,\ V_{st}\backslash W]}{E} \right\rceil \tag{3.2}$$

$$y(\delta^+(W)) \geq \left\lceil \frac{h\ [W,\ V_{st}\backslash W]}{B} \right\rceil \tag{3.3}$$

sont valides pour PDR.

preuve

Il est clair que les contraintes $By(\delta^+(W)) \geq l\ [W,\ V_{st}\backslash W]$ (resp. $By(\delta^+(W)) \geq h\ [W,\ V_{st}\backslash W]$ et $Ey(\delta^+(W)) \geq h\ [W,\ V_{st}\backslash W]$) sont valides pour PDR. En effet, elles expriment le fait que la somme des capacités (resp. volumes) des véhicules sur les arcs de $\delta^+(W)$ ne doit pas dépasser le volume des demandes (resp. consignes) de W vers $V_{st}\backslash W$. Puisque y est entier, il suffit de diviser les deux membres de cette inégalité par B et prendre la partie entière supérieure.

On remarque que seules les coupes qui ne s'intersectent pas avec A_T sont considérées, parce que les contraintes de capacité ne sont pas appliquées pour ces arcs. (Dans la première formulation il n'y a pas de contrainte (5) et (6) pour ces arcs et dans la deuxième, on considère une capacité infinie) Cela vient du fait que n'importe quelle demande (resp. consigne) peut rester sans aucune condition sur les sommets du réseau.

Contraintes étendue de capacité résiduelle des arcs

Les contraintes étendues de capacité résiduelle sont utilisées pour renforcer la relaxation linéaire. Elles ne peuvent pas être utilisées dans la formulation métrique puisqu'elles dépendent des variables x et z.

Pour un sous ensemble donné $K \subseteq (P \cup Q)$, le volume total des demandes $p \in K$ et des consignes $q \in K$ est égal à $q(K) = \sum_{p\in K} q^p + \sum_{q\in K} q^q$.
Posons $\gamma_K = \lceil \frac{q(K)}{B} \rceil$ et $r_K = (q(k)\ \ mod\ \ B)$. Par convention, on pose r_K égal à B si $q(k)$ est un multiple de B (i.e., $r_K \in [0, B]$).

Proposition 3.6.2. *Soit $K \subseteq (P \cup Q)$ et $a \in \widetilde{A}$. La contrainte de capacité résiduelle d'un arc*

$$(\sum_{p\in K} x_a^p + \sum_{q\in K} z_a^q) - r_K y_a \leq (\gamma_K - 1)(B - r_k) \tag{3.4}$$

est valide pour PDR.

preuve

Supposons que $y_a \geq \gamma_K$. Alors, la contrainte (3.4) est équivalente à $\sum_{p,q\in K}(x_a^p + z_a^q) \leq q(K) + (y_a + \gamma_K)r_K$, qui est dominé par la somme des bornes des contraintes

$x_a^p + z_a^q \leq q^p + q^q$ pour tout $p, q \in K$. D'un autre coté, supposons que $y_a \leq \gamma_K - 1$. En remplaçant r_K par $B - m$ où $0 \leq m < B$ dans (3.4), on obtient la contrainte $\sum_{p,q \in K}(x_a^p + z_a^q) \leq By_a + m((\gamma_K - 1) - y_a)$ qui est dominée par les contraintes (5). Donc, les contraintes du capacité résiduelle des arcs (3.4) sont valides pour le PDR.

Dans ce qui suit, on donne un exemple de contrainte de capacité résiduelle d'un arc violé.

Exemple 3.6.1. *Supposons qu'on a deux demandes et deux consignes qui ont respectivement les volumes* $q^1 = 6$, $q^2 = 12$, $q_1 = 3$ *et* $q_2 = 6$, *et la capacité du véhicule* B *est égale à* 10.

Soit $(\overline{y}, \overline{x}, \overline{z})$ *la solution fractionnaire qui vérifie (1)-(8). Alors il existe* $a' \in \widetilde{A}$ *avec* $x_{a'}^p = q^p$ *pour* $p = 1, 2$, $z_{a'}^p = q_q$ *pour* $q = 1, 2$ *et* $y_{a'} = 2.7$. *Il existe une contrainte de capacité résiduelle d'un arc défini sur l'arc* a' *qui est violé par* $(\overline{y}, \overline{x}, \overline{z})$. *En effet, si on considère l'ensemble des demandes et consignes* $K = P \cup Q$, *on aura* $\gamma_K = 3$ *et* $r_K = 7$, *et alors la contrainte implique qu'il faut avoir* $7y_{a'} \geq (6+12+3+6) - (3-1)(10-7) = 21$, *ce qui implique* $y_{a'} \geq 3$.

On introduit maintenant l'étendue de la contrainte de capacité résiduelle d'un arc. Pour cela, on présente d'abord quelques notations.
Pour $\phi \in \{0, ..., T\}$, on défini $V^\phi = \{u_t \in V_{st} | t \in \phi\}$ et $A^\phi = \delta^+(V^\phi) \cap \widetilde{A}$. V^ϕ correspond aux sommets de $V \backslash \{v_0\}$ à la date ϕ et A^ϕ correspond aux arcs de $A \backslash \delta(v_0)$ qui commencent à la date ϕ.

Proposition 3.6.3. *Soient* $K \subseteq (P \cup Q)$, $\phi \in \{0, ..., T-1\}$ *et* $X \subseteq A^\phi$. *Alors l'étendue de la contrainte de capacité résiduelle d'un arc*

$$\sum_{p,q \in K} \sum_{a \in X} (x_a^p + z_a^q) - r_K \sum_{a \in X} y_a \leq (\gamma_K - 1)(B - r_K) \tag{3.5}$$

est valide pour PDR.

preuve

On remarque d'abord que le graphe auxiliaire n'a pas d'arcs parallèle. A alors, il ne contient aucun arc (u, w) avec $u \in V_1^\phi, w \in V_2^\phi$ et $\phi_1 \leq \phi_2$. Puisque $x_a^p + z_a^q \leq q^p + q_q$ pour tout $a \in \widetilde{A} \cup A_T$, la contrainte $\sum_{a \in \delta^+(V^\phi)}(x_a^p + z_a^q) \leq q^p + q_q$ est retenue pour tout $\phi \in \{0, ..., T-1\}$ et pour tous $p \in P$ et $q \in Q$.

Soit $\phi \in \{0, ..., T-1\}, K \subseteq (P \cup Q)$ et $X \subseteq A^{\phi}$. De maniere similaire à celle de la preuve de la proposition (3.6.2), on peut montrer que la contrainte (3.4) est dominée par la somme des contraintes $\sum_{a \in X}(x_a^p + z_a^q) \leq q^p + q_q$ pour tous $p, q \in K$ (resp. les contraintes (4) pour tout arc dans X) si $\sum_{a \in X} y_a \geq \gamma_K$ (resp.$\sum_{a \in X} y_a \geq \gamma_K$). Donc, les contraintes étendues de la capacité résiduelle des arcs (3.5) sont valides pour le PDR.

3.6.2 Algorithme "Branch and Bound"

Algorithm 3 Branch and Bound

1: *Initialisation* : $L = \{IP\}, S^0 = S, \overline{z}^0 = \infty$, et $z_R^i = -\infty$.

2: *le test d'arret* : Si $L = \emptyset$, alors la solution x^0 qui donne $\underline{z}_{IP} = cx^0$ est optimale.

3: *Sélection du problème et relaxation* : Sélectionner un problème IP^i puis le supprimer de L. Résoudre sa relaxation RP^i. Soit z_R^i la valeur optimale de la relaxation et soit x_R^i la solution optimale, s'il en existe une.

4: *Elagage* :

- Si $z_R^i \leq \underline{z}_{IP}$, aller a l'étape 2.
- Si $x_R^i \notin S^i$, aller à l'étape 5.
- $x_R^i \in S^i$ et $cx_R^i > \underline{z}_{IP}$, soit $\underline{z}_{IP} = cx_R^i$. Supprimer tous les problèmes de L vérifiant $\overline{z}^0 \leq \underline{z}_{IP}$. Si $cx_R^i = z_R^i$, aller à l'étape 2 ; sinon aller à l'étape 5.

5: *Séparation* : Soit $\{S^{ij}\}_{j=1}^k$ une séparation de S^i. Ajouter les problèmes $\{IP^{ij}\}_{j=1}^k$ à la liste L, où $\overline{z}^{ij} = z_R^i$ pour tout $j = 1, ..., k$. Aller à l'étape 2.

3.7 conclusion

Dans ce chapitre nous nous sommes intéressés en premier lieu à la présentation et à la modélisation de problème de distribution avec les rechargements, puis à la méthode de résolution de cette nouvelle variante du problème de tournées de véhicules.

4

Branch and Bound pour PDR

Contents

4.1 Introduction

Dans ce chapitre, nous exposerons la deuxième partie de notre travail comprenant la méthode de résolution du problème d'optimisation combinatoire en nombres mixtes "Branch and Bound".

Pour plusieurs problèmes, en particulier les problèmes d'optimisation, l'ensemble des solutions est fini (en tous les cas, il est dénombrable). Il est donc possible, en principe, d'énumérer toutes ces solutions, et ensuite de prendre celle qui nous arrange. L'inconvénient majeur de cette approche est le nombre prohibitif du nombre de solutions : il n'est guère évident d'effectuer cette énumération.

La méthode "branch and bound" (procédure par séparation et évaluation progressive) consiste à énumérer les solutions d'une manière intelligente en ce sens que, en utilisant certaines propriétés du problème en question, cette technique arrive à éliminer des solutions partielles qui ne mènent pas à la solution que l'on recherche. De ce fait, on arrive souvent à obtenir la solution recherchée en des temps plus ou moins raisonnables. Bien entendu, dans le pire cas, on retombe toujours sur l'élimination explicite de toutes les solutions du problème.

Pour ce faire, cette méthode se dote d'une fonction qui permet de mettre une borne sur certaines solutions pour soit les exclure soit les maintenir comme des solutions potentielles. Bien entendu, la performance d'une méthode "branch and bound" depend, entre autres, de la qualité de cette fonction (de sa capacité d'exclure des solutions partielles tôt).

4.2 Algorithme Général

Par convenance, on représente l'exécution de la méthode "branch-and-bound" à travers une arborescence. La racine de cette arborescence représente l'ensemble de toutes les solutions du problème considéré. Dans ce qui suit, nous résumons la méthode "branch-and-bound" sur des problèmes de minimisation.

Pour appliquer la méthode "branch-and-bound", nous devons être en possession :

1. d'un moyen de calcul d'une borne inférieure d'une une solution partielle
2. d'une stratégie pour subdiviser l'espace de recherche et créer des espaces de recherche de plus en plus petits.
3. d'un moyen de calcul d'une borne supérieure pour au moins une solution.

La méthode commence par considérer le problème de départ avec son ensemble de solutions, appelé la racine. Des procédures de bornes inférieures et supérieures sont appliquées à la racine. Si ces deux bornes sont égales, alors un solution optimale est trouvée, et on arrête là. Sinon, l'ensemble des solutions est divisé en deux ou plusieurs sous-problèmes, devenant ainsi des enfants de la racine. La méthode est ensuite appliquée récursivement à ces sous-problèmes, engendrant ainsi une arborescence. Si une solution optimale est trouvée pour un sous-problème, elle est réalisable, mais pas nécessairement optimale, pour le problème de départ. Comme elle est réalisable, elle peut être utilisée pour éliminer toute sa descendance : si la borne inférieure d'un sommet dépasse la valeur d'une solution déjà connue, alors on peut affirmer que la solution optimale globale ne peut être contenue dans le sous-ensemble de solutions représenté par ce sommet. La recherche continue jusqu'à ce que tous les sommets sont soit explorés ou éliminés.

4.3 Implémentation

On commence cette section par la présentation de l'algorithme "Branch-and-Bound" utilisé pour la résolution de modèle introduit dans le chapitre 3.

Dans la section précédente, nous avons présenté notre algorithme construit par la méthode Branch and Bound, que nous avons implémenté en utilisant le langage de programmation MATLAB Version 7.0. Notre choix s'est porté sur ce langage car nous estimons qu'il est l'un des langages de programmation mathématique les plus puissants qui existent actuellement.

Pour résoudre les programme linéaire relaxé on a utilisé comme solveur le "Linprog" et le "Qsopt". Notre programme est testé sous Packard Bell System, 1,60GHz avec 1Go de Ram, exécuté sous Windows XP.
Dans cette section du chapitre, nous avons choisi de présenter quelques résultats obtenus en exécutant l'algorithme proposé. Pour ce faire, nous allons détailler les résultats trouvés pour quatre exemples.

Les exemples traités dans cette section sont des instances tirés du graphe cité dans l'article de Kerevin et al[**?**].

Les instances sont créées de telle sorte quelles soient réalisables sous la borne de la durée maximale T. Pour cela, on suppose que la durée de parcourt de chaque arc est $1h$.

En effet, chaque véhicule démarre du dépôt et va directement à l'origine de la première demande à acheminer. Une fois que le véhicule arrive à la destination de p, il va à l'origine de la demande suivante p' à travers l'arc $(d^p,o^{p'})$ et ainsi de suite. Le véhicule termine son chemin au dépôt.

Remarque 4.3.1. *Cette solution est réalisable tant que le volume de chaque demande est inférieur à la capacité du véhicule, et la durée de parcourt de chaque arc est* $1h$.

Avant de commencer, on donne quelques notations qu'on va utiliser par la suite :

$\mid T \mid$: Limite de temps, $T = 5$.
$\mid V \mid$: Nombre de sommets dans le graphe initial.
$\mid V' \mid$: Nombre de sommets dans le graphe auxiliaire.
$\mid P \mid$: Nombre de demandes.
$\mid Q \mid$: Nombre de consignes.
$\mid F \mid$: Nombre de véhicules.
$\mid Iter \mid$: Nombre d'itérations.
CPU : Durée d'exécution en secondes.

Chaque instance est d'abord transformée en un graphe auxiliaire tout en suivant les étapes citée dans le troisième chapitre.

Remarque 4.3.2. *Un graphe initial ayant n sommets engendre un graphe auxiliaire avec* $6n + 2$ *sommets et* $2n + 5n + \sum_{\delta^+(v)\setminus(A^O \cup A^D)} n_{d(v)} \times d(v)$ *arcs. où* $d(v)$ *est le degré du sommet* v *dans le graphe initial.*

Cette transformation nous mène à un modèle contenant
$(|P| + |Q| + 1)(5n + \sum_{\delta^+(v)\setminus(A^O \cup A^D)} n_{d(v)} \times d(v)) + 2n$ variables et $(6n + 2) \times (|P| + |Q| + 1) + 2(5n + \sum_{\delta^+(v)\setminus(A^O \cup A^D)} n_{d(v)} \times d(v)) + 1$ contraintes.

4.4 Exemples Numériques

Dans la section précédente nous avons présenté notre algorithme “Branch and Bound”, que nous avons implementé en utilisant le langage de programmation MATLAB Version 7.0.

Dans cette section, nous allons présenter les exemples choisis.

4.4.1 Exemple 1

Pour cet exemple on a :
$| V |= 2$
$| V' |= 14$
$| P |= 1$ avec $p^1 = 2$
$| Q |= 1$ avec $q^1 = 2$
$| F |= 1$
$| B |= 3$

La formulation mathématique de cet exemple sous forme graphique est donnée par FIG.4.1 :

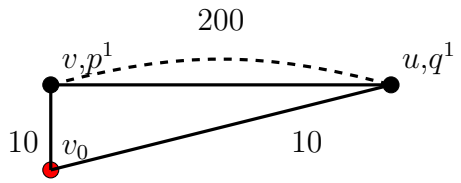

FIG. 4.1 – Exemple 1

Son graphe auxiliaire est représenté à la FIG.4.2 :
avec :
201 contraites
308 variable

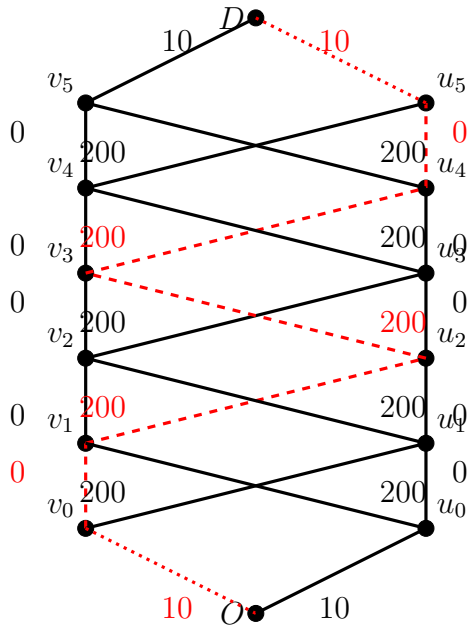

FIG. 4.2 – Le graphe auxiliaire de l'Exemple 1

Nous avons résolu ce problème par l'algorithme proposé et nous avons obtenu les résultats suivantes :
La solution de cet exemple est présenté sur le graphe auxiliaire, cette solution nous donne le chemin à suivre par le véhicule pour livrer la demande p qui est au sommet v et pour récupérer la consigne q qui est au sommet u avec un coût égal à 1240, en 166.562 secondes et en 65 itérations.

4.4.2 Exemple 2

Pour cet exemple on a :

$| V |= 3$
$| V' |= 20$
$| P |= 2$ avec $p^1 = 2$ et $p^2 = 2$
$| Q |= 1$ avec $q^1 = 2$
$| F |= 1$
$| B |= 3$

La formulation mathématique de cet exemple sous forme graphique est donnée par la FIG.4.3 :

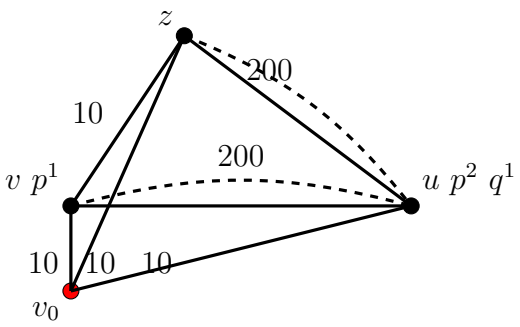

FIG. 4.3 – Exemple 2

Son graphe auxiliaire est représenté à la FIG.4.4 :

avec :

133 contraites

186 variable

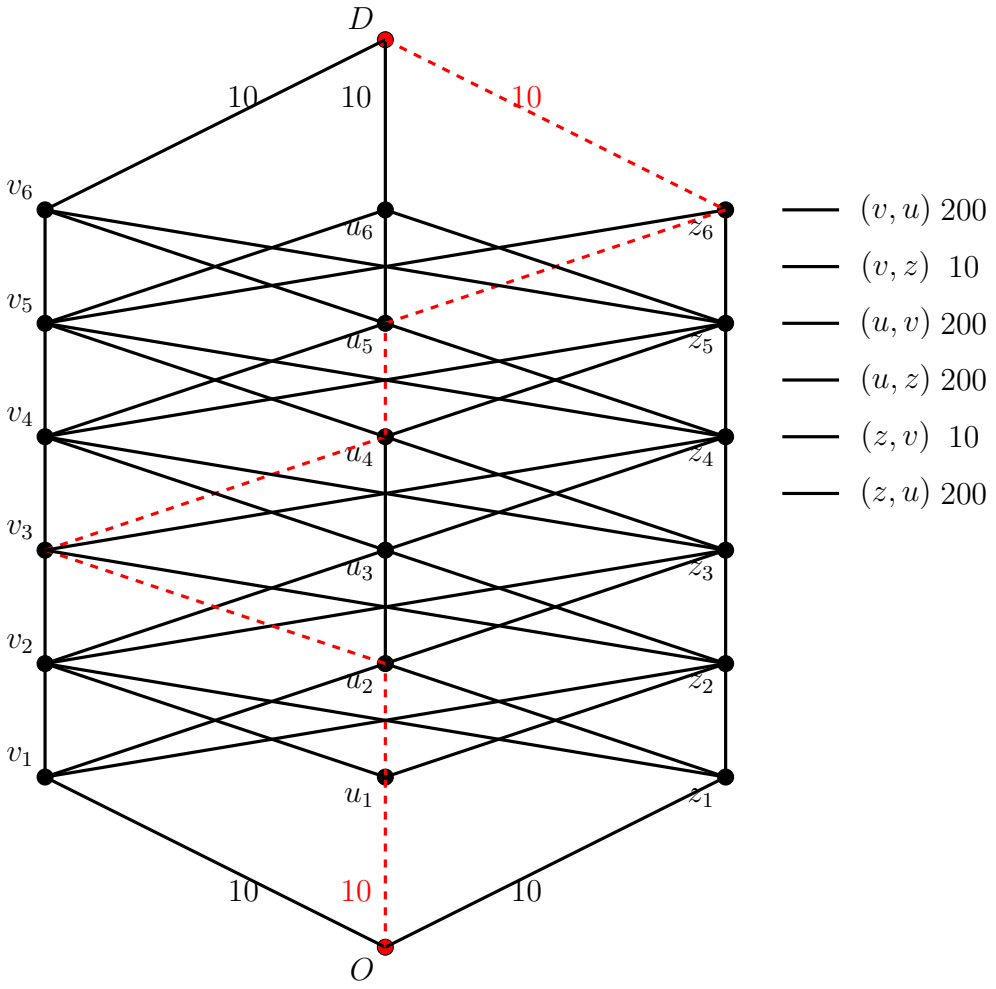

FIG. 4.4 – Le graphe auxiliaire de l'Exemple 2

Nous avons résolu ce problème par l'algorithme proposé et nous avons obtenu les résultats suivantes :
La solution de cet exemple est présenté sur le graphe auxiliaire, cette solution nous donne le chemin à suivre par le véhicule pour livrer les demande p^1, p^2 et pour récupérer la consigne q^1 avec un coût égal à 620, en 226.75 secondes et en 1087 itérations.

4.4.3 Exemple 3

Pour cet exemple on a :

$| V |= 4$
$| V' |= 26$
$| P |= 2$ avec $p^1 = 2$ et $p^2 = 2$

$\mid Q \mid = 2$ avec $q^1 = 2$ $q^2 = 2$
$\mid F \mid = 1$
$\mid B \mid = 3$

La formulation mathématique de cet exemple sous forme graphique est donnée par la FIG.4.5 :

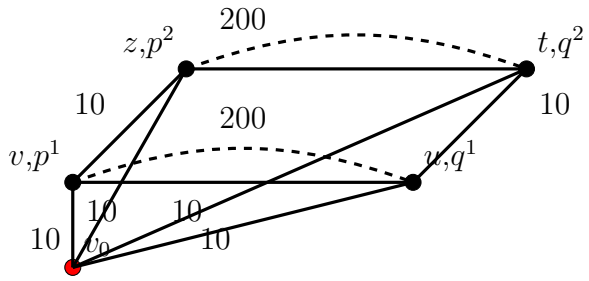

FIG. 4.5 – Exemple 3

Son graphe auxiliaire est représenté à la FIG.4.6 :
avec :
201 contraites
308 variable

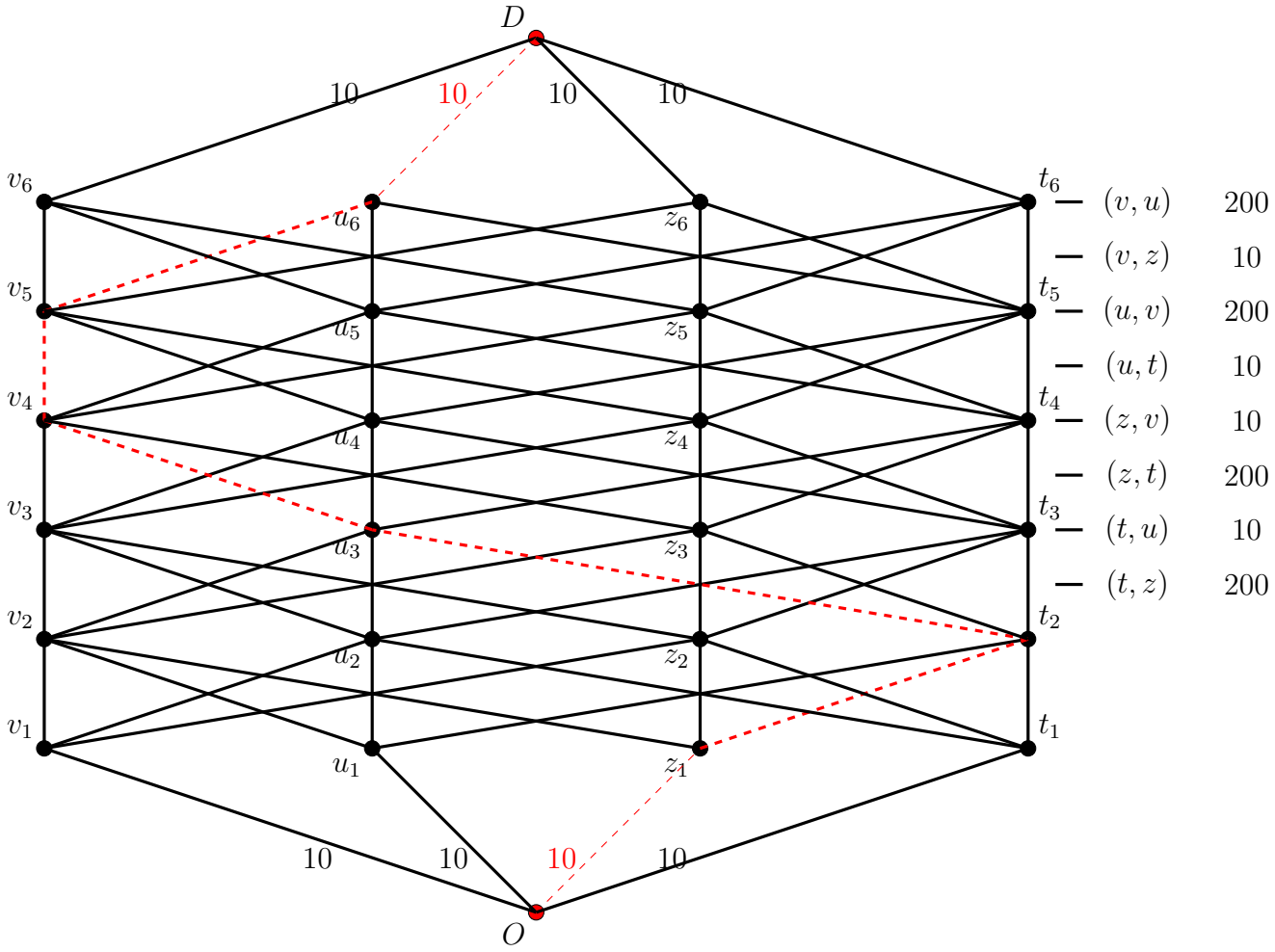

FIG. 4.6 – Le graphe auxiliaire de l'Exemple 3

Nous avons résolu ce problème par l'algorithme proposé et nous avons obtenu les résultats suivantes :
La solution de cet exemple est présenté sur le graphe auxiliaire, cette solution nous donne le chemin à suivre par le véhicule pour livrer les demande p^1, p^2 et pour récupérer la consigne q^1, q^2 avec un coût égal à 630, en 1261.641 secondes et en 3977 itérations.

4.5 Discussion

Sur les trois exemples que nous avons exposés dans cette partie, notre algorithme trouve toutes les solutions optimales. Dans cet algorithme, on commence par la résolution du programme linéaire relaxé soit par Linprog, soit par Qsopt pour trouver une solution de départ. Puis on applique la technique "Branch and Bound" en nombres mixtes, jusqu'à

soit :

* Trouvé la solution optimale ;

* Prouver que le programme mathématique est irréalisable(Sortie sans solution).

4.6 Conclusion

Les méthodes d'optimisation sont divisées en deux grandes classes, selon qu'elles résolvent les problèmes de manière exacte ou approchée. Au fur et à mesure des années et de l'augmentation exponentielle de la puissance de calcul des ordinateurs, ces méthodes d'optimisation sont devenues de plus en plus évoluées.

Cette puissance de calcul disponible a abouti depuis quelques années à de nombreux travaux tentant de résoudre des programmes linéaires en nombres mixtes de tailles élevées d'une manière optimale.

Conclusion Générale

La Recherche Opérationnelle. Ce mot si beau et si effrayant. Beau pour ce qu'il permet de résoudre comme problèmes et effrayant pour ce qu'il présente comme difficultés. Hélas, nulle chose n'est belle si elle n'est pas difficile à atteindre.

Le thème de recherche opérationnelle que nous nous sommes proposés de traiter se trouve justement difficile mais fascinant et passionnant. Il est difficile parce qu'il est NP-complet mais fascinant parce qu'il nous a permis d'apprendre et de prendre plus que ce que nous avons apporté.

Ce problème est connu dans la littérature sous l'appellation de problème de tournée. Son énoncé simple et facile à comprendre peut être trompeur pour ce qu'il présente comme dessous, tenants et aboutissants. Il s'agit, en effet, de livrer et de récupérer un ensemble d'articles avec un coût minimum.

Par son simple énoncé, le lecteur non averti peut passer sur ce problème sans apercevoir sa véritable importance qui a fait de lui un centre d'attraction pour des centaines de chercheurs durant les deux dernières décennies.

Le problème de tournées doit sa valeur, non seulement aux nombreuses applications dont il peut faire l'objet, mais également à sa nature intrinsèque qui a fait de lui un problème NP-complet.

Le long de ce mémoire, nous avons tenté d'approcher différents aspects relatifs à ce problème. Ainsi, nous avons commencé par situer son cadre théorique sans oublier l'aspect pratique qui fait de lui un véritable problème.

L'essentiel de ce mémoire s'articule autour du troisième chapitre dans lequel nous avons présenté notre contribution personnelle .

Nous avons rappelé dans le deuxième chapitre, certains articles qui traitent des problèmes similaires au nôtre. Le troisième chapitre est consacré, dans un premier temps à la modélisation mathématique qui généralise les formulations existantes du problème de ramassage et livraison. Par la suite nous avons proposé deux méthodes de résolution de type "branch and cut" et "branch and bound" pour le problème posé.

La première, basée sur l'introduction du nouvelles inégalités valides, n'a pas été complètement exploité. La deuxième est basée sur l'algorithme "branch and bound" pour les problèmes en nombres entiers mixtes. En dernier lieu, nous avons implémenté l'algorithme "branch and bound" sous MATLAB 7.0, et l'avons appliqué à des exemples tests générés aléatoirement. Nous avons constaté que la méthode "branch and bound" est efficace pour les problèmes de petite taille.

Bibliographie

[1] Alvina G. H. K, R. L. Cheu, Q. Meng, *Distance-constrained capacitated vehicle routing problems with flexible assignment of start and end depots*, Mathematical and Computer Modelling, 47, 140-152, 2008.

[2] Arbex Valle C, A. S. da Cunha, G. R. Mateus, L. C. Martinez, *Exact algorithms for a selective Vehicle Routing Problem where the longest route is minimized*, Electronic Notes in Discrete Mathematics, 35, 133-138, 2009.

[3] Archetti C, M.G. Speranza, *The Split Delivery Vehicle Routing Problem : A Survey*, April 12, 2007.

[4] Archetti C, D. Feillet, M. Gendreau, M. G. Speranza, *Complexity of the VRP and SDVRP*, Transportation Research Part C, 2010.

[6] Bae S. K, H. S. Hwang, G. S. Cho, M. J. Goan, *Integrated GA-VRP solver for multi-depot system*, Computers and Industrial Engineering, 53, 233-240, 2007.

[7] Barahona F, *Network design using cut inequalities*, SIAM Journal on Optimization, 6, 823-837, 1996.

[8] Bellman R,*Some applications of theory of dynamic programming : A review*, Journal of the Oparational Research Society of america, 2(3), 275-288,1954.

[9] Berbeglia G, J. F. Cordeau, G. Laporte, *Dynamic pickup and delivery problems*, European Journal of Operational Research, February 25, 2009.

[10] Bienstock D, S. Chopra, O. Günlünk, et C. Tsai, *Minimum cost capacity installation for multicommodity network flows*, Mathematical Programming, 81, 177-199, 1995.

[11] Bin Y, Y. Zhong-Zhen, Y. Baozhen, *An improved ant colony optimization for vehicle routing problem*, European Journal of Operational Research, 196, 171-176, 2009.

[12] Bixby R. E, S. Ceria, C. M. McZeal, et M. W. P. Savelsbergh, *An updated mixed integer programming library* : MIPLIB 3.0.Optima, 58, 12-15, 1998.

[13] Bochtis D.D, C.G. Sørensen, *The vehicle routing problem in field logistics part I*, bio systems engineering, 104,447-457, 2009.

[14] Bolduc M. C, G. Laporte, J. Renaud, F. F. Boctor, *A tabu search heuristic for the split delivery vehicle routing problem with production and demand calendars*, European Journal of Operational Research, 202, 122130, 2010.

[15] Chvàtal V, *Edmonds polytopes and hierarchy of combinatorial problems*, Discret Mathematics, 4, 305-337, 1973.

[16] Cook S. A, *The complexity of theorem proving procedure*,Proc.3rd ACM sump. on Theory of Computing, 151-158 , 1971.

[17] Cortés C, M. Matamala, C. Contardo, *The pickup and delivery problem with transfers : Formulation and a branch-and-cut solution method*, European Journal of Operational Research, 200, 711-724, 2010.

[18] Edmonds J,*Maximum matching and polyhedron with 0-1 vertices*, Journal of research of the national bureau of Standards, 69(B), 125-130, 1965.

[19] Fisher M.L, R. Jaikumar, *A decomposition algorithm for large-scale vehicle routing*, Working Paper, 78-11-05, Department of Decision Sciences, University of Pennsylvania, 1978.

[20] Fisher M.L, *Optimal solution of vehicle routing problems using minimum k-trees*, Operations Research 4, 626-642, 1994.

[21] Garey M, D.Jonhson, *Computers and intractability : a Guide to the Theory of NP-Completness*, W.H.Freeman, 1979.

[22] Girard S, J. Renaud, F. F. Boctor, *HEURISTIQUES RAPIDES POUR LE PROBLÈME DE TOURNÉES DE VÉHICULES*,ASAC 2006, Banff, Alberta,Faculté des sciences de l'administration, Université Laval, Centre de recherche sur les technologies de l'organisation réseau.

[23] Gomory R. E, *Solving Linear Programming Problems in Integers*, American Mathematical Society, 10, 211-216, 1960.

[24] Gomory R. E, *Outline of an algorithm for integer solution to linear programs*, Buletin of American Mathematical Society, 64, 275-278, 1958.

[25] Gribkovskaia I, G. Laporte, A. Shyshoua, *The single vehicle routing problem with deliveries and selective pickups*, Computers and Operations Research, 35, 2908-2924 , 2008.

[26] Gutiérrez-Jarpa G , G. Desaulniers, G. Laporte, V. Marianov, *A branch-and-price algorithm for the Vehicle Routing Problem with Deliveries, Selective Pickups and Time Windows*, European Journal of Operational Research, 206, 341-349, 2010.

[27] Hoff A, I. Gribkovskaia, G. Laporte, A. Løkketangen, *Lasso solution strategies for the vehicle routing problem with pickups and deliveries*, European Journal of Operational Research, 192, 755-766, 2009.

[28] Hopcroft J. E, J. D. Ullman, C. H. Papadimitriou, *Introduction to automata theory, Languages and computation*, Addison-Wesley, 1979.

[29] Huth T, D.C. Mattfeld, *Dynamics of the Multi-Depot Pickup and Delivery Problem*, October 18, 2006.

[30] Iri M, *On an extension of themaximum-flow minimum-cut theorem tomulticommodity flows*, J. Operations Research Soc. of Japan, Vol. 13, No. 3, 1971.

[31] Karp R. M, *Reducibility among combinatorial problems* in R. E. Miller, J. W. Thatcher *Complexity of computer computation*, Plenum Press, 85-103 , 1972.

[32] Kerivin H. L. M, M. Lacroix, A. R. Mahjoub, A. Quilliot, *The splittable pickup and delivery problem with reloads*, European Journal of Industrial Engineering (EJIE), 2007.

[33] Lin C.K.Y , *A cooperative strategy for a vehicle routing problem with pickup and delivery time windows*, Computers and Industrial Engineering, 55, 766-782, 2008.

[34] Lin S, *Computer solutions of the traveling salesman problem*, The Bell System Technical J, 44, 2245-2269, 1965.

[35] Little J. D, K. G. Murty, D. W. Sweeney, C. Karel, *An algorithm for the traveling salesman problem*, Operation Research, 11, 199-208, 1994.

[36] Liu S, L. Lei, S. Park, *On the multi-product packing-delivery problem with a fixed route*, Transportation Research Part E, 44, 350-360, 2008.

[37] Lopez P, *Cours de GRAPHES*, http ://www.laas.fr/∼lopez/cours/GRAPHES/graphes.html, 30 novembre 2005.

[38] Lougee-Heimer R, *The Common Optimization INterface for Operations Research*, IBM Journal of Research and Development, Vol. 47, No. 1, 57-66, 2003.

[39] Lu Q, M. M. Dessouky, *A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows*, European Journal of Operational Research, 175, 672-687, 2006.

[40] Lysgaard Jens, *Reachability cuts for the vehicle routing problem with time windows*, European Journal of Operational Research, 175, 210-223, 2006.

[41] Miller D.L, *A matching based exact algorithm for capacitated vehicle routing problems*, ORSA Journal on Computing, 7, 19, 1995.

[42] Nowak M, O. Ergun, C. C. White III , *Pickup and Delivery with Split Loads*, Transportation Science.

[43] Onaga K , O. Kakhuso, *On feasibility conditions of multicommodity flows networks*, Transactions on circuit theory, 4, 425-429, 1971.

[44] Oppen J, A. Løkketangen, J. Desrosiers, *Solving a rich vehicle routing and inventory problem using column generation*, Computers and Operations Research, 37, 1308-1317, 2010.

[45] Padberg M, G. Rinaldi, *A branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problems*, SIAM Revew, 33, 60-100,1991.

[46] Parragh S. N, K. F. Doerner, R. F. Hartl, *A survey on pickup and delivery problems Part I : Transportation between customers and depot*, February 19, 2008.

[47] Parragh S. N, K. F. Doerner, R. F. Hartl, *A survey on pickup and delivery problemsPart II : Transportation between pickup and delivery locations*, April 16, 2008.

[48] Pisinger D, S. Ropke, *A general heuristic for vehicle routing problems*,Computers and Operations Research, 34, 2403-2435, 2007.

[49] Privé J, J. Renaud, F. Boctor, G. Laporte, *Solving a vehicle-routing problem arising in soft-drink distribution*, Journal of the Operational Research Society, 57, 1045-1052, 2006.

[50] Renaud J., Boctor F. F., *A sweep-based algorithm for the fleet size and mix vehicle routing problem*, European Journal of Operational Research, 140, 3, 618-628, 2002.

[50] Sakarovitch M, *Optimisation Combinatoire*, HERMANN Enseignement des Science, Paris, 1984.

[51] Savelsberg M.W. P and Sol M , *The general pickup and delivery problem*, Transportation Science, Vol. 29, pp.17-29, 1995.

[52] Schrijver A, *Theory of linear Integer programming*, John Wiley and sons, New York, Wiley interscience in discrete matheematics and optimisation edition, 1986.

[53] Sigward E, *Introduction à la théorie des graphes*, e.sigward@ac-nancy-metz.fr, 2002.

[54] Sirdey R, *Optimiser. . .En découpant des polyèdres*, journal L'Ouvert, 28 mars 2006.

[55] Taillard E. D., *Parallel iterative search methods for vehicle routing problems*, Networks, 23, 661-673, 1993.

[56] Tang J, Z. Pan, R. Y. K. Fung, H. Lau , *Vehicle routing problem with fuzzy time windows*, Fuzzy Sets and Systems, 160, 683-695, 2009.

[57] Tang J, J. Zhang, Z. Pan, *A scatter search algorithm for solving vehicle routing problem with loading cost*, Expert Systems with Applications, 37, 4073-4083, 2010.

[58] Tarantilis C.D. , V.S. Kiranoudis, V.S. Vassiliadis, *A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem*, European Journal of Operational Research, 152, 148-158, 2004.

[59] Tavakkoli-Moghaddam. R, N. Safaei, Y. Gholipour, *A hybrid simulated annealing for capacitated vehicle routing problems with the independent route length*, Applied Mathematics and Computation, 176, 445-454, 2006.

[60] Toth P, Vigo D, *Models, relaxations and exact approaches for the capacitated vehicle routing problem*, Discrete Applied Mathematics, 123, 487-512, 2002.

[61] YANG F. M, XIAO H. J, *Models and Algorithms for Vehicle Routing Problem with Transshipment Centers*, Systems Engineering - Theory and Practice Volume 27, Issue 3, March 2007