RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ ABDELHAMID IBN BADIS-MOSTAGANEM
FACULTÉ DES SCIENCES EXACTES ET DE L'INFORMATIQUE

DÉPARTEMENT DE MATHÉMATIQUES ET INFORMATIQUE

Mémoire de fin d'étude

Pour l'obtention du diplôme de Master en Mathématiques
Cycle LMD

OPTION : Modélisation, Contrôle et Optimisation

Présenté par

Melle Abbassa Nadira

Soutenu le 26 Mai 2015

Intitulé

L'heuristique de Clarke & Wright pour résoudre le problème de livraison de fioul

Devant le Jury

Hocine ABLAOUI	**Président**	MAA	**U. MOSTAGANEM**
Rachid BELGACEM	**Examinateur**	MAA	**U. Chlef**
Abdessamad AMIR	**Encadreur**	MCA	**U. MOSTAGANEM**

Année universitaire 2014/2015

Dédicaces

Je dédie ce modeste travail

À ma très chère mère
À mon très cher père

À mon frère Mohamed Chérif
À ma petite sœur Senia

À tous mes proches de la famille
ABBASSA et KAID OMAR

À toutes mes chères amies et mes collègues de
l'Université de Mostaganem

À toutes les personnes que j'aime.

Nadira

Remerciements

Au terme de ce travail, je tiens à remercier vivement et avant tout notre grand **Dieu** *le tout puissant et miséricordieux, qui m'a donné la volonté, la force et surtout la patience pour mener ce mémoire.*

Il m'est agréable d'exprimer mes vifs remerciements et ma profonde gratitude à Mr AMIR Abdessamad, pour son encadrement, ses conseils et ses remarques constructives qui ont directement contribué pour effectuer ce travail.

Mes remerciements les plus respectueux s'adressent aux membres de jury, Mr ABLAOUI Hocine pour l'honneur qu'il m'a fait en acceptant d'être président de mon jury de ce mémoire, et Mr BELGACEM Rachid d'avoir accepté d'examiner mon mémoire et de faire partie de mon jury.

Ensuite, je présente mes remerciements à tous ceux qui ont contribué, directement ou indirectement à l'élaboration de ce travail.

Je remercie du fond du cœur mes parents, mon frère et ma sœur pour leur présence effective, leur soutien inconditionnel et leurs encouragements durant toutes les années d'études.

Finalement, merci à toutes mes amies, sans oublier dans mes remerciements tout mes enseignants qui ont contribué à ma formation.

Veuillez bien trouver ici le témoignage de ma profonde estime.

Résumé

Ce mémoire porte sur la résolution de problème de livraison de fioul avec une heuristique. Il s'agit d'un cas particulier du problème de tournées de véhicules. Il est modélisé comme un programme linéaire à variables mixtes : variables binaires et variables continues réelles. Comme la résolution exacte de ce type de problèmes ne donne pas en général satisfaction, nous proposons une méthode approchée pour résoudre le problème de livraison de fioul, c'est l'heuristique de Clarke & Wright. Cette heuristique est également testée sur quelques instances réelles.

Mots clés : Optimisation combinatoire, Problèmes de tournées de véhicules, Problème de livraison de fioul, Heuristique de Clarke & Wright.

Table des matières

Introduction

Depuis le début de la seconde moitié du $20^{ème}$ siècle, des spécialistes de la recherche opérationnelle se sont penchés sur la résolution des problèmes de transport routier qui se trouvent au cœur des problématiques de la logistique. Avant de parvenir au consommateur final, un produit parcourt, en général, des centaines voire des milliers de kilomètres par voie terrestre. Dans le monde de l'industrie où le transport se révèle incontournable, le problème de tournées de véhicules (VRP) a été le sujet d'une recherche intensive durant plus de cinquante ans, lié à son importance dans le domaine de la logistique. Un exemple du VRP est abordé dans ce mémoire, il s'agit du problème de livraison de fioul qui consiste à déterminer une séquence optimale des itinéraires des camions-citernes afin de satisfaire la demande des clients.

Par sa formulation, le VRP est dans la classe des problèmes NP-difficiles. Ainsi, les méthodes exactes se limitent à la nature combinatoire de ces problèmes, rendant la résolution des problèmes réels difficiles par les temps de résolution prohibitifs, donc on a recours aux méthodes heuristiques. Les heuristiques sont des méthodes approchées, souvent basées sur le bon sens ou sur des observations empiriques, qui permettent d'obtenir des solutions réalisables. La contribution de ce mémoire est de proposer une des heuristiques la plus utilisées, pour résoudre le VRP et ainsi le problème de livraison de fioul, cette heuristique porte le nom de Clarke & Wright.

Le manuscrit est répartit en quatre chapitres. Le premier chapitre décrit le problème de voyageur de commerce nécessaire pour une bonne compréhension de la suite. Le deuxième chapitre se focalise sur le problème de tournées de véhicules, ce problème est formulé à l'aide d'un modèle mathématique à variables binaires, et un autre modèle à variables mixtes. Quelques variantes de VRP seront décrites. Le troisième chapitre est consacré à l'étude de l'heuristique de Clarke & Wright. On a présenté son principe et ses différents modes. Les différentes versions de cette heuristique ont été implémentées sous Matlab. Le dernier chapitre est dédié à décrire le problème de livraison de fioul. Nous avons appliqué l'heuristique de Clarke & Wright abordée au troisième chapitre pour résoudre un exemple réel de livraison de fioul posé dans la région de Donges dans le nord ouest de la France. On clôture ce travail par une conclusion, qui présente une synthèse des travaux réalisés et propose des perspectives de recherches futures, et une bibliographie.

Chapitre 1

Problème de voyageur de commerce

1.1 Introduction

Le problème du voyageur de commerce ("traveling salesman problem" TSP) nait d'une problématique vécue par des vendeurs ou commerciaux qui se déplacent pour livrer ou rencontrer leurs clients. C'est l'un des problèmes le plus étudié dans l'optimisation combinatoire (programme linéaire discret "à variables entières" binaires "0 ou 1") qui consiste à la recherche d'un trajet minimal permettant à un voyageur de visiter n villes séparées par distances données en passant par chaque ville exactement une fois. Il commence par une ville quelconque et termine en retournant à la ville de départ. Quel chemin faut-il choisir afin de minimiser la distance parcourue ?

La notion de distance peut-être remplacée par d'autres notions comme le temps qu'il met ou l'argent qu'il dépense. En général, on cherche à minimiser le coût. Théoriquement, le TSP peut se représenter sous forme d'un graphe $G = (V, E)$ où V est l'ensemble de n sommets (les villes) $V = \{1, 2, ..., n\}$ et E est l'ensemble des arêtes ou arcs $E = \{(i, j), i \neq j, i, j \in V\}$, chaque arête a un poids c_{ij} (le coût) avec $c_{ii} = \infty, i = 1, ..., n$ (Afin d'éviter les sous-tours d'ordre 1 "$x_{ii} = 1$"). Le problème est de trouver un circuit qui passe par tous les sommets une et une seule fois avec un coût minimal. On distingue deux types de TSP ; symétrique ($c_{ij} = c_{ji}$) et asymétrique ($c_{ij} \neq c_{ji}$), ici, uniquement le cas symétrique est considéré.

1.2 Formulation du problème

Un programme linéaire (linear programming "LP") en variables continues est formulé

$$(LP) \begin{cases} \min c^t x \\ Ax \leq b \\ x \geq 0 \end{cases},$$

avec $A \in \mathbb{R}^{m \times n}$, le second membre $b \in \mathbb{R}^{m \times 1}$ et $c \in \mathbb{R}^{n \times 1}$.

Lorsque toutes les variables doivent être entières, le problème résultant noté (ILP) (integer linear programming) est le problème général de la programmation linéaire en variables entières

$$(ILP) \begin{cases} \min c^t x \\ Ax \leq b \\ x \in \mathbb{N}^n \end{cases}.$$

Si les variables $x \in \{0,1\}^n$, on dit qu'on a un programme linéaire en variables binaires (BIP) (binary integer programming)

$$(BIP)\begin{cases} \min c^t x \\ Ax \leq b \\ x \in \{0,1\}^n \end{cases}.$$

Si une partie seulement des variables doit être entière, le problème résultant est un problème de programmation mixte en variables entières et variables réelles, noté $(MILP)$ ("mixed integer linear programming"). En séparant les deux types de variables dans la fonction objectif et les contraintes, un $MILP$ se formule :

$$(MILP)\begin{cases} \min(c_1^t x + c_2^t y) \\ A_1 x + A_2 y \leq b \\ x \in \mathbb{N}^p, y \in \mathbb{R}_+^{n-p} \end{cases},$$

où $A_1 \in \mathbb{R}^{m\times p}$, $A_2 \in \mathbb{R}^{m\times(n-p)}$, $c_1 \in \mathbb{R}^{p\times 1}$, $c_2 \in \mathbb{R}^{(n-p)\times 1}$ et le second membre $b \in \mathbb{R}^{m\times 1}$.

Le TSP peut se modéliser comme un (BIP). En effet, étant donnée n villes, notons $C = (c_{ij})$ la matrice des coûts et x_{ij} les variables de décision définies par

$$x_{ij} = \begin{cases} 1 \text{ si le voyageur va immédiatement de la ville } i \text{ vers la ville } j \\ 0 \text{ sinon} \end{cases}.$$

La formulation mathématique est

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1.1}$$

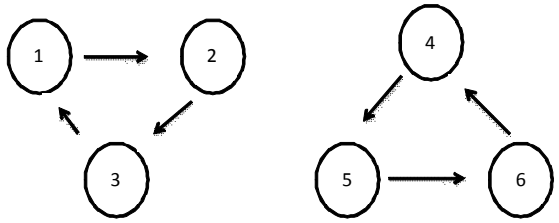
$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 1,...,n \tag{1.2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad j = 1,...,n \tag{1.3}$$

$$\sum_{i\in S}\sum_{j\in \bar{S}} x_{ij} \geq 1, \quad S \subset V, \quad 2 \leq |S| \leq n-2 \tag{1.4}$$

$$x_{ij} \in \{0,1\} \quad i,j = 1...n, i \neq j. \tag{1.5}$$

Dans cette formulation, la relation (1.1) décrit la fonction objectif. Les contraintes (1.2) et (1.3) s'appellent les contraintes de degré qui assurent qu'une ville n'est visitée qu'une seule fois : on y arrive une et une seule fois (1.2), on en part une et une seule fois (1.3), ces contraintes ne sont pas suffisantes pour décrire les tours (voir figure (1.6)), d'où la nécessité d'introduire Les contraintes (1.4) appelées contraintes d'élimination des sous-tours, avec S un sous ensemble de V et $\bar{S}$ son complémentaire dans V, $|S|$ est le cardinal de S. Enfin, les contraintes (1.5) sont les contraintes d'intégrité de variables.



(1.6)

Une autre formulation des contraintes (1.4) peut être donnée par la relation suivante

$$\sum_{i,j\in S} x_{ij} \leq |S| - 1, \quad S \subset V, \quad 2 \leq |S| \leq n - 2. \tag{1.7}$$

Cette formulation de TSP contient $n(n-1)$ variables binaires, $2n$ contraintes de degré et $2^n - 2n - 2$ contraintes d'élimination de sous-tours.

De nombreuses variantes de ce problème existent, entre-autres le problème de voyageurs de commerce multiples ([1]), également connu sous le nom de multiple Traveling Salesman Problem (mTSP). Le mTSP consiste en une généralisation du TSP. Dans cette version, un nombre m de voyageurs sont à l'origine localisés à la ville (dépôt) de départ. L'objectif est de calculer le nombre m de tournées au départ du dépôt qui visitent une seule fois les villes (nœuds) et qui se terminent à la ville de départ. La somme des coûts des tournées obtenues doit être minimale. Le mTSP s'est révélé assez proche d'autres variantes de problèmes de routage. En particulier, l'addition de quelques contraintes supplémentaires permet de le transformer en un pur problème de tournées de véhicules ou Vehicle Routing Problem (VRP) (chapitre 2). Pour cela, on peut, par exemple, considérer que les voyageurs sont des véhicules de capacité limitée, ce qui donne un problème de VRP avec contraintes de capacité ou Capacitated VRP (CVRP). Etant donnée sa proximité au VRP, la résolution du mTSP a été utilisée par certains auteurs (Laporte. [10]) dans le cadre d'une approche hiérarchique pour la résolution de problèmes de VRP beaucoup plus complexes. La formulation de mTSP

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1.8}$$

$$\sum_{i=1}^{n} x_{i1} = m \tag{1.9}$$

$$\sum_{j=1}^{n} x_{1j} = m \tag{1.10}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 2, ..., n \tag{1.11}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad j = 2, ..., n \tag{1.12}$$

$$\sum_{i,j\in S} x_{ij} \leq |S| - 1, \quad S \subseteq V \setminus \{1\}, \quad 2 \leq |S| \leq n - 2 \tag{1.13}$$

$$x_{ij} \in \{0, 1\} \quad i, j = 1...n, i \neq j. \tag{1.14}$$

Les contraintes (1.9) et (1.10) imposent que les m voyageurs partent de et retournent au nœud 1 (dépôt). Les contraintes (1.11) et (1.12) sont les contraintes de degré, les contraintes (1.14) d'intégrité. Enfin, les contraintes (1.13) sont les contraintes d'éliminations des sous-tours.

1.3 Méthodes de résolution

Il existe trois grandes catégories de méthodes de résolution des problèmes (BIP) : les méthodes exactes, les méthodes heuristiques et les méthodes métaheuristiques.

Les méthodes exactes permettent d'obtenir une solution optimale à chaque fois, mais leurs durées de calcul tendent à augmenter exponentiellement avec la taille du problème. A l'inverse, les méthodes heuristiques sont des méthodes spécifiques, permettant d'obtenir rapidement une solution approchée de bonne qualité, mais qui n'est donc pas nécessairement optimale. Les plus utilisées sont celles du "plus proche voisin", et les "méthodes d'insertion". Les méthodes métaheuristiques ([16]) sont des algorithmes d'optimisation généralement de type stochastique combinant plusieurs approches heuristiques. Les métaheuristiques sont souvent inspirées par des systèmes naturels, qu'ils soient pris en physique (cas du recuit simulé), en biologie de l'évolution (cas des algorithmes génétiques) ou encore en éthologie (cas des algorithmes de colonies de fourmis), ce type de méthodes ne sera pas abordé dans ce mémoire.

Nous donnerons un exemple des méthodes heuristiques ultérieurement au chapitre 3. A présent, nous donnons les grandes lignes de quelques méthodes exactes pour le TSP.

1.3.1 Méthodes exactes

Les méthodes exactes, procèdent par exploration intelligente des solutions afin d'en trouver une de coût minimal.

Méthode Branch and Bound

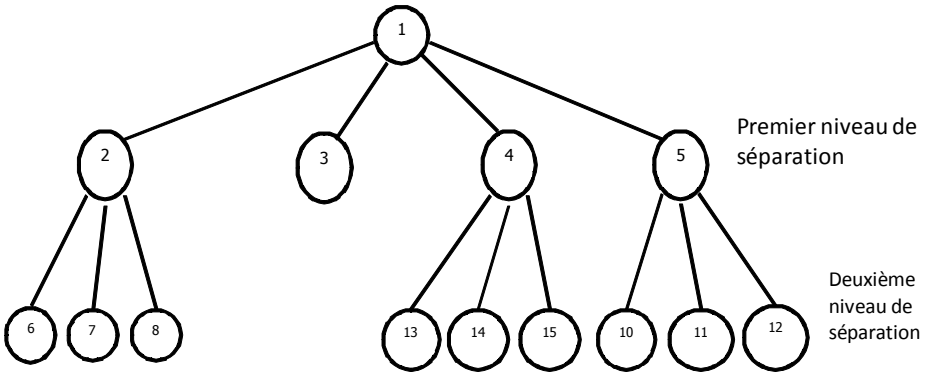
L'une des méthodes exactes la plus utilisée pour le TSP est la procédure par Séparation et Evaluation (Branch and Bound). Ces méthodes sont basées sur une énumération "intelligente" des solutions admissibles d'un problème d'optimisation combinatoire. L'idée, est de prouver l'optimalité d'une solution en partitionnant l'espace des solutions. L'algorithme consiste à :

-Séparation : séparer (brancher) de manière récursive le problème en sous-problèmes de cardinalité inférieure tels que l'union de leurs espaces de solutions forme l'espace des solutions du problème-père, le nœud séparé en priorité est celui qui produit la borne inférieure la plus faible. Un sous-ensemble qui ne peut être séparé est appelé ensemble sondé.

-Evaluation : déterminer une borne inférieure de chaque sous-problème.

Cette recherche par décomposition de l'ensemble des solutions peut être représentée gra-

phiquement par un arbre, où chaque nœud correspond à un sous-problème (1.15).



Arbre de Branch and Bound

(1.15)

La stratégie de cette méthode favorise l'exploration des sous-problèmes possédant la plus petite borne inférieur. Elle dirige la recherche là où la probabilité de trouver une meilleure solution est la plus grande. Elle permet aussi d'éviter l'exploration de tous les sous-problèmes qui possèdent une évaluation supérieure à la valeur optimale ([2]).

Méthode de branchement et coupes

La méthode de coupes polyédrales (En anglais, Cutting plane) ([3], [5]) est basée sur l'idée de simplifier la résolution d'un problème en relaxant certain de ses contraintes. Si la solution optimale du problème relaxé est une solution réalisable pour le problème original, alors elle est l'optimum recherché, sinon, il existe certainement des contraintes du problème initial qui sont violées par la solution courante. Il faut donc trouver un sous-ensemble de ces contraintes alors appelées coupes, puis les ajouter à la relaxation avant de la résoudre à nouveau. La procédure se poursuit jusqu'a ce que la solution obtenue soit réalisable pour le problème original. Soit le problème (LP), en plus, certaines des variables sont spécifiées comme entiers ou bien A contient un nombre important, par exemple exponentiel par rapport à n de contraintes. Choisissons tout d'abord un sous-ensemble $A'x \leq b'$ de contraintes de $Ax \leq b$ tel qu'il existe une solution finie x' au problème relaxé. La procédure des coupes est :

Etape 1 : résoudre le nouveau problème et soit x' sa solution.
Etape 2 : Trouver une ou plusieurs inégalités du problème original qui sont violées par x'.
Etape 3 : Si aucune n'est trouvée, arrêter. Sinon, ajouter les inégalités violant au nouveau problème et passer à l'étape 1.

Pour le TSP, on relaxe les contraintes d'intégrités et les contraintes d'élimination des sous-tours. Le problème relaxé résultant est :

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, ..., n$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad j = 1, ..., n$$

$$0 \leq x_{ij} \leq 1 \quad i, j = 1...n, i \neq j.$$

Une méthode de coupes seule peut ne fournir que des solutions fractionnaires pour un ILP (non réalisables), alors il est nécessaire de subdiviser l'espace de solution. L'algorithme de branchement et coupes (En anglais, "Branch and Cut") est une méthode qui conjugue les efforts de l'algorithme du "Branch and Bound" et de la méthode des coupes polyédrales. Ainsi, pour résoudre un programme linéaire en nombres binaires, le "Branch and Cut" commence par résoudre une relaxation du problème, puis, on applique la méthode des coupes sur la solution trouvée. Si celle-ci n'arrive pas à obtenir une solution entière par exemple $0 < x_{ij} < 1$, alors le problème est divisé en deux sous problèmes ($x_{ij} = 0$ et $x_{ij} = 1$) qui seront résolus de la même façon.

Chapitre 2

Le problème de tournées de véhicules

2.1 Introduction

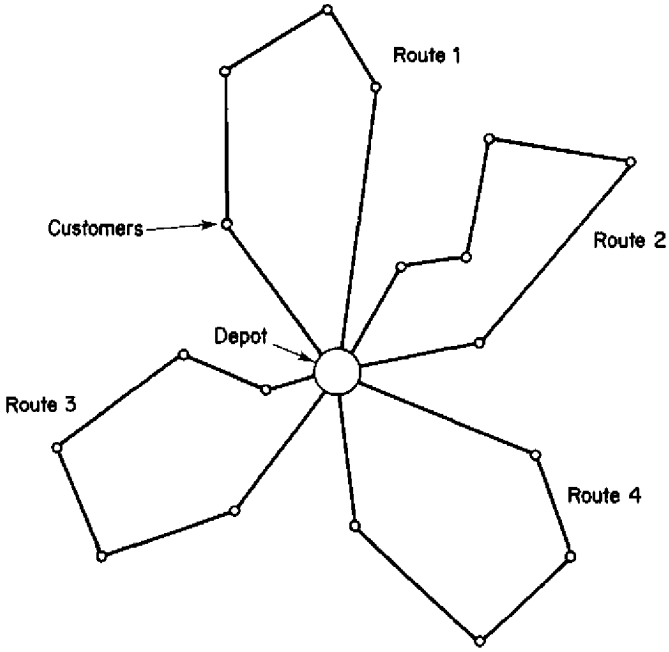
Le problème de tournées de véhicules (En anglais, Vehicle Routing Problem (VRP)) est une généralisation du TSP. On dispose de m véhicules (donc $mTSP$) au dépôt afin de satisfaire les demandes de n clients. Ce problème n'est pas nouveau, la première formulation du problème est attribuée à Dantzig et Ramser (1954) sous le nom de "Truck Dispatching Problem" (problème d'expédition de camions). Plus généralement, on parlera de problème de tournées de véhicules avec capacité (En anglais, Capacitated Vehicle Routing Problem (CVRP)). Concrètement, une flotte de m véhicules de capacité Q, basée au dépôt, doit effectuer des tournées réalisables, c'est-à-dire quitter le dépôt, visiter une seule fois des clients dont la somme des demandes ne dépasse pas la capacité de véhicule, avant de retourner au dépôt. Chaque client doit être servi par un seul véhicule qui satisfait totalement sa demande. Le VRP ou le CVRP apparaît fréquemment dans des situations non liées à la livraison des marchandises, par exemple pour réaliser des tournées d'intervention (maintenance, réparation, contrôles, la collecte du courrier...) ou de visites (visites médicales, commerciales, etc.)

2.2 Formulation mathématique du VRP

Le but de cette section est de présenter certaines formulations de VRP qui sont nombreuses et variées. Le problème de VRP ou CVRP peut être modélisé mathématiquement sous la forme d'un (BIP) ou ($MILP$). Le CVRP consiste à déterminer m routes (tournées) telles que :

- Chaque véhicule commence sa tournée et se termine au dépôt ;
- Exactement chaque client doit être visité une et une seule fois ;
- La demande totale de chaque route ne dépasse pas la capacité du véhicule ;

– Le coût total de voyage est minimisé.



forme de la solution de VRP (2.1)

Dans ce mémoire, on se concentre sur le cas où les véhicules ont une capacité identique (véhicules homogènes), c'est à dire

$$Q_k = Q, \forall k = 1, \ldots, m. \tag{2.2}$$

On peut déterminer une borne inférieure sur le nombre de véhicules nécessaires pour satisfaire les demandes des clients par la formule

$$m \geq \left\lceil \frac{\sum_{i=2}^{n} q_i}{Q} \right\rceil,$$

où $\lceil \rceil$ est la partie entière supérieure.

Une des formulations de VRP les plus utilisées est une extension de la formulation du TSP. Les clients sont indexés par $i = 2, \ldots, n$ et expriment chacun une demande q_i. Le dépôt est indexé par $i = 1$. Chacun des m véhicule a une capacité Q tels que

$$\sum_{i=2}^{n} q_i \leq m * Q. \tag{2.3}$$

c_{ij} désigne les frais (le coût) de parcours de l'arc (i, j). Les variables de décision utilisées sont ainsi définies :

$$x_{ij} = \begin{cases} 1 & \text{si le client } j \text{ est juste après le client } i \text{ dans une tournée} \\ 0 & \text{sinon,} \end{cases}$$

Nous formulons le problème (VRP), où il sagit de minimiser la fonction objectif

$$\min \sum_{i,j=1}^{n} c_{ij} x_{ij}, \tag{2.4}$$

sous les contraintes

$$\sum_{i=2}^{n} x_{ij} = 1 \quad \forall j = 2, ..., n \tag{2.5}$$

$$\sum_{j=2}^{n} x_{ij} = 1 \quad \forall i = 2, ..., n \tag{2.6}$$

$$\sum_{i=2}^{n} x_{i1} = m \tag{2.7}$$

$$\sum_{i=2}^{n} x_{1i} = m \tag{2.8}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - r(S), \quad S \subseteq V \setminus \{1\}, \quad 2 \leq |S| \leq n-2 \tag{2.9}$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, ..., n \tag{2.10}$$

La relation (2.4) exprime la minimisation de la somme des coûts de toutes les tournées. Les contraintes (2.5) et (2.6) sont les contraintes de degré ; les contraintes (2.7) et (2.8) assurent que les m véhicules quittent le dépôt, puis y retournent. Les contraintes (2.9) sont les contraintes d'élimination des sous-tours habituelles où $r(S)$ est une borne inférieure appropriée sur le nombre de véhicules nécessaires pour visiter tous les clients de S dans la solution optimale définie par

$$r(S) = \left\lceil \frac{\sum_{i \in S} q_i}{Q} \right\rceil .$$

Enfin, les contraintes (2.10) sont des contraintes d'intégrité. Ce modèle est un (BIP).

Pour éliminer les sous-tours, on peut utiliser une autre formulation ([8], [9], [17]) qui comprend aussi des variables continues supplémentaires u_i pour chaque $i = 2, ..., n$. Ces variables représentent le cumul du véhicule après qu'il visite le client i.

$$q_i \leq u_i \leq Q \tag{2.11}$$

Cette relation (2.11) exige que le cumul u_i doit être supérieur (ou égal) à la demande q_i et ne dépasse pas la capacité Q.

$$u_i - u_j + Q x_{ij} + (Q - q_i - q_j) x_{ji} \leq Q - q_j, \quad \forall i, j = 2, ..., n, i \neq j, q_i + q_j \leq Q \tag{2.12}$$

La connectivité entre les clients sur une route est satisfaite par les contraintes (2.12), c'est à dire si le véhicule visite le client i avant le client j dans une tournée ($x_{ij} = 1$), alors les contraintes (2.12) deviennent

$$u_j \geq u_i + q_j,$$

si le client i est visité après le client j dans la même tournée ($x_{ji} = 1$), le cumul $u_i = u_j + q_i$ ($u_i \leq u_j + q_i$). Sinon, ces contraintes deviennent redondantes. Les contraintes (2.13), (2.14) et (2.15) sont les contraintes d'élimination des sous-tours et de capacité,

$$u_i \leq Q - (Q - q_i)x_{1i} \ \forall i = 2, ..., n \tag{2.13}$$

$$u_i \geq q_i + (Q - \bar{q}_i - q_i)x_{i1} \ \forall i = 2, ..., n \tag{2.14}$$

$$x_{1i} + x_{i1} \leq 1 \ \ \forall i = 2, ..., n, q_i + \bar{q}_i \leq Q, \tag{2.15}$$

notez que les sous-tours isolés violent les contraintes d'élimination des sous-tours, si le client i est le premier dans une tournée ($x_{1i} = 1$), le cumul $u_i = q_i$ ($u_i \leq q_i$ et $u_i \geq q_i$) par (2.13), les contraintes (2.15) impliquent que $x_{i1} = 0$. Si le client i est le dernier dans une tournée ($x_{i1} = 1$), les contraintes (2.14) sont équivalentes à

$$u_i \geq Q - \bar{q}_i,$$

où

$$\bar{q}_i = \min_{j, j \neq i} \{q_j\},$$

où le client j n'est pas encore affecté dans une tournée. Cette contrainte signifie qu'on ne peut pas ajouter un autre client dans cette tournée car les contraintes (2.11) vont être violées (le cumul va dépasser la capacité Q), dans les autres cas, ces contraintes sont redondantes.

$$u_i \geq 0 \quad i = 2, ..., n. \tag{2.16}$$

Enfin, les contraintes (2.16) sont les contraintes de positivité. Le (VRP) devient

$$\min \sum_{i,j=1}^{n} c_{ij} x_{ij}$$

sous les contraintes

(2.5), (2.6), (2.7), (2.8), (2.10), (2.11), (2.12), (2.13), (2.14), (2.15) et (2.16).

Cette formulation appartient à la classe des problèmes ($MILP$) (x_{ij} binaires et u_i réelles continues et positives).

2.3 Variantes du VRP

Le VRP de base est, avant tout, une version simplifiée au regard de l'ensemble des problèmes de tournées de véhicules que l'on peut rencontrer dans la réalité. En effet, en fonction des contraintes à prendre en compte, plusieurs variantes du VRP ont été définies en littérature. Un certain nombre de paramètres permettant de classifier les problèmes de tournées sont principalement :

Distances entre les villes (distances symétriques, non symétriques, euclidiennes ou non euclidiennes) ; Nombre de dépôts (un seul dépôt, deux ou plusieurs dépôts) comme le problème "**MDVRP**" (Multi-Depot Vehicle Routing Problem) ; Composition de la flotte (véhicules homogènes, véhicules hétérogènes) ; Taille de la flotte (un seul véhicule, plus d'un

véhicule) ; Nature des demandes (déterministe, stochastique) ; Longueur des tournées (restreinte comme le problème "**DCVRP**" (Distance-Constrained Vehicle Routing Problem) ou non restreinte) ; Durée des tournées (restreinte, non restreinte) ;Nombre de livraisons/visites à un client (unique, multiple) ; Les horaires de livraisons sont non définis, définis de manière précise (rendez-vous), définis dans des intervalles horaires souples (soft time windows), définis dans des intervalles horaires rigides (hard time windows) : parmi les variantes de ces paramètres, le VRP avec fenêtres de temps "**VRPTW**" (Vehicle Routing Problem with Time Windows) ; Nombre de tournées admises par véhicule (unique, multiple) ; Calcul des coûts de transport (fonction de la distance, fonction du temps, fonction du nombre de véhicules) ; Répartition des charges dans les véhicules (compartiment unique, multi-compartiments où les charges doivent être séparées dans un même véhicule comme la variante "**MC-VRP**" (Multi-Compartment Vehicle Routing Problem)) ; Opérations de collectes (collectes et livraisons simultanées chez les clients, soit collecte soit livraison chez un client), parmi les variantes le problème "**VRPB**" qui inclut, outre des livraisons, des collectes dans les tournées de livraisons (Vehicle Routing Problem with Backhauls).

2.4 Les méthodes de résolution

Dans le cadre d'une démarche de résolution des problèmes de VRP, plusieurs méthodes ont été développées dans la littérature. Comme le VRP est un BIP (ou $MILP$), ces méthodes ont été classées en trois grandes familles : les méthodes exactes, les méthodes heuristiques et les métaheuristiques.

Les méthodes approchées (heuristiques), quoique fournissent généralement des solutions non optimales, ne cessent d'être améliorées tant en qualité de solution qu'en temps de calculs. Dans le chapitre suivant, nous verrons une des plus célèbre heuristique appliquée au VRP.

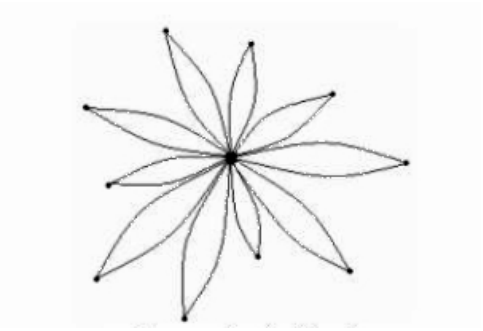
Chapitre 3

Heuristique de Clarke et Wright

3.1 Présentation de la méthode

Les méthodes heuristiques ont pour but de produire des solutions réalisables de bonne qualité pour des problèmes NP-difficiles (Il n'existe pas jusqu'à ce jour un algorithme polynomiale qui résout ce problème). Leurs durées de calcul sont souvent très inférieures à celles des méthodes exactes, ce qui les rendent attractives pour la résolution des problèmes réels. L'efficacité de tels algorithmes est jugée en comparant le temps de calcul ainsi que la valeur de la solution obtenue avec la meilleure solution connue ou une borne inférieure. L'algorithme des "savings" de Clarke & Wright est sans aucun doute, une des heuristiques les plus connues pour le VRP.

En 1964, Clarke et Wright ([4]) ont développé une heuristique pour résoudre le CVRP dans laquelle le nombre de véhicules est libre. Cette heuristique est connue sous le nom "des distances sauvées". Le principe de cette méthode est de commencer par dédier une tournée à chaque client de façon à le relier à l'unique dépôt ([6], [11]). Le résultat donne une solution initiale triviale ayant une allure de marguerite (voir la figure 3.1)

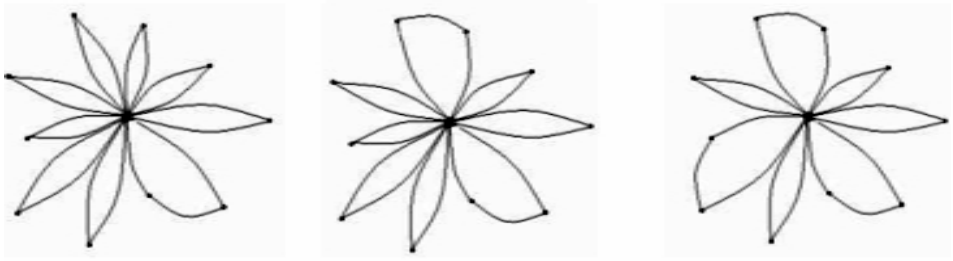


Marguerite de depart

(3.1)

Ensuite, le but est de fusionner les tournées obtenues afin de réduire le coût de la solution courante, on effectue successivement les fusions réalisables (compatibles avec les capacités et la disponibilité des véhicules) et de gain maximal, jusqu'à ce qu'on n'en trouve plus ou que le nombre de tournées désiré soit atteint. Les fusions sont appliquées en commençant par

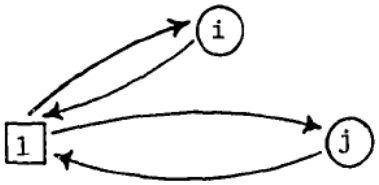
celles qui produisent les réductions de coût les plus importantes.



Fusions successives des tournees

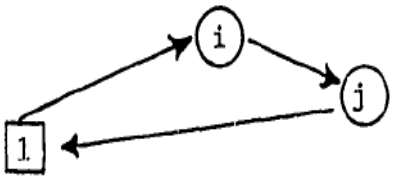
(3.2)

L'algorithme de Clarke & Wright est un algorithme «d'échange» dans le sens où à chaque étape une série de visites est échangée pour un meilleur ensemble de visites. Initialement, nous supposons que tous les deux clients i et j sont visités individuellement par deux vehicules.



(3.3)

Le concept des distances sauvées de base est exprimé par les économies des coûts obtenues quand la fusion de deux tournées dans une seule tournée est effectuée, comme illustré dans la figure ci-dessous



Les clients i et j sont lies

(3.4)

Dans la figure (3.3), les clients i et j sont desservis par deux tournées séparées $(1, i, 1)$ et $(1, j, 1)$. Dans un tel cas, le coût total du transport sera

$$\begin{aligned} D_1 &= c_{1i} + c_{i1} + c_{1j} + c_{j1} \\ &= 2(c_{1i} + c_{1j}), \end{aligned}$$

où 1 représente le dépôt, c_{ij} est le coût de transport entre i et j et $c_{1j} = c_{j1}$ (Nous considérons ici que le cas symétrique).

D'une manière équivalente, le coût de transport D_2 de la route $(1, i, j, 1)$ de la figure (3.4) est :

$$D_2 = c_{1i} + c_{ij} + c_{j1}.$$

En combinant les deux trajets, on obtient le "saving" S_{ij} :

$$S_{ij} = D_1 - D_2 = c_{1i} + c_{1j} - c_{ij}. \tag{3.5}$$

Pour chaque paire possible de clients i et j, il y a un correspondant "saving" (distance sauvée) S_{ij}. Encore une fois, cette distance peut être positive, négative ou nulle ([12], [21]). Si les distances satisfont l'inégalité triangulaire qui signifie

$$c_{ik} + c_{kj} \geq c_{ij},$$

alors $S_{ij} \geq 0$. Pour n villes (1 se réfère au dépôt), on a C_{n-1}^2 valeurs de S_{ij} avec $i = 2, ..., n-1$, $j = 3, ..., n,\ \ i < j$, dans le cas symétrique.

Dans la première étape de l'algorithme du "saving", les distances (S_{ij}) pour toutes les paires de clients sont calculées, une liste de savings est obtenue, ensuite, trier les distances en ordre décroissant. En partant du haut de la liste, nous relions les clients i et j sur une seule tournée où S_{ij} représente le "saving" maximal courant et les conditions suivantes sont satisfaites :

1. Les clients i et j ne sont pas déjà sur le même parcours de véhicule ;
2. ni i ni j sont dans un tour existant ;
3. la capacité des véhicules n'est pas dépassée.

Lorsque toutes les contraintes sont vérifiées (capacités, sous-tours, ...), la fusion est effectuée. L'algorithme considère alors le "saving" qui suit. Nous continuons de la même manière jusqu'à ce qu'aucune autre liaison entre les clients ne puisse être faite qui réduit le coût total.

Le coût total z_{hcw} vérifie la relation suivante en notant par dts, la distance totale sauvée

$$z_{hcw} = 2 * \sum_{i=2}^{n} c_{1i} - dts. \tag{3.6}$$

Il existe deux versions d'implémentation de l'algorithme de Clarke & Wright : la version parallèle et la version séquentielle.

3.1.1 Version séquentielle

Dans la version séquentielle, exactement une route est construite à un moment. Lorsque cette dernière (route) ne peut plus être agrandie à cause des contraintes, une nouvelle route est crée, il faut recommencer à partir du haut de la liste puisque les combinaisons qui n'étaient pas admissibles jusqu'à maintenant peuvent devenir admissibles.

Algorithm 3.1 ***Etape 1*** *: Calculer les savings $S_{ij} = c_{1i} + c_{1j} - c_{ij}$ pour toutes les paires de clients i et j*

Etape 2 *: Classer les savings dans l'ordre décroissant*

Etape 3 *: En partant du haut de la liste, procéder comme suit :*

Etape 4 *: Trouver la première liaison réalisable dans la liste, qui peut être utilisée pour étendre l'une des deux extrémités de la voie actuellement construite.*

Etape 5 *: Si la route ne peut pas être étendue plus loin, mettre fin à la route. Choisissez la première liaison possible dans la liste pour commencer un nouvel itinéraire.*

Etape 6 *: Répéter les étapes 4 et 5 jusqu'à ce que plus aucune liaison ne puisse être choisie.*

3.1.2 Version parallèle

Dans la version parallèle, plusieurs routes peuvent être construites en parallèle, elle ne nécessite qu'un seul passage dans la liste. L'algorithme considère alors le plus grand "saving" disponible et regarde si les deux clients i et j ne sont pas encore affectés, alors une nouvelle tournée est considérée en fusionnant les deux routes $(1, i, 1)$ et $(1, j, 1)$ tout en respectant les contraintes.

Algorithm 3.2 ***Etape 1*** *: Calculer les savings $S_{ij} = c_{1i} + c_{1j} - c_{ij}$ pour toutes les paires de clients i et j*

Etape 2 *: Classer les savings dans l'ordre décroissant*

Etape 3 *: En partant du haut de la liste, procéder comme suit :*

Etape 4 *: Si la réalisation d'une liaison donnée en résulte une voie possible selon les contraintes de VRP, ajoutez cette liaison à la solution ; sinon, rejeter la liaison.*

Etape 5 *: Essayez la liaison suivante dans la liste et répéter l'étape 4 jusqu'à ce que plus aucune liaison ne puisse être choisie.*

Les solutions avec un nombre fixe de véhicules ([10]) (c'est à dire le nombre de tournée est imposé) peuvent être obtenues en répétant l'étape 4 pour les deux versions jusqu'à ce que le nombre requis de routes (véhicules) soit atteint, même si les "savings" deviennent négatifs. La façon dont fonctionne l'algorithme est illustrée dans ce qui suit au moyen d'un exemple numérique.

3.1.3 Exemple illustratif

Nous considérons un problème avec cinq clients ([13]). Les coûts de transport entre toutes les paires de points sont présentés dans le tableau suivant, où 1 représente le dépôt.

	1	2	3	4	5	6
1	–	28	31	20	25	34
2		–	21	29	26	20
3			–	38	20	32
4				–	30	27
5					–	25
6						–

(3.7)

Les demandes des clients qui doivent être livrés à partir du dépôt sont données dans le tableau (3.8). La capacité du véhicule est de 100 unités.

Client	Quantité
2	37
3	35
4	30
5	25
6	32

(3.8)

Les "savings" S_{ij} sont calculés, seule la moitié supérieure du tableau est indiquée, car les "savings" sont symétriques en raison des coûts symétriques :

$$
\begin{array}{cc|ccccc}
 & & \multicolumn{5}{c}{j} \\
 & & 2 & 3 & 4 & 5 & 6 \\
\hline
 & 2 & - & 38 & 19 & 27 & 42 \\
 & 3 & & - & 13 & 36 & 33 \\
i & 4 & & & - & 15 & 27 \\
 & 5 & & & & - & 34 \\
 & 6 & & & & & -
\end{array}
\tag{3.9}
$$

Maintenant, les "savings" sont triés dans l'ordre décroissant. Cela donne la liste triée suivante des paires de points :

$2-6$
$2-3$
$3-5$
$5-6$
$3-6$
$2-5$
$4-6$
$2-4$
$4-5$
$3-4$

Version séquentielle

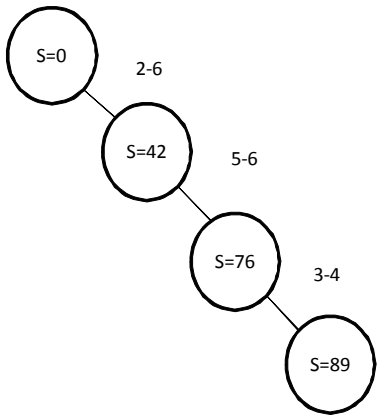
Dans l'exemple, les clients 2 et 6 sont considérés en premier. Ils peuvent être affectés à la même route puisque leur demande conjointe de 69 unités ne dépasse pas la capacité du véhicule. Maintenant, nous établissons la connexion $2-6$, et les points 2 et 6 ainsi seront voisins sur une route dans la solution finale.

Ensuite, nous considérons les clients 2 et 3. Si les clients 2 et 3 devraient être voisins sur une route, cela nécessiterait la séquence à la clientèle $3-2-6$ (ou $6-2-3$) sur une route, parce que nous avons déjà établi que 2 et 6 doit être visité en succession immédiate sur la même route. La demande totale (104) sur cette route serait dépasser la capacité du véhicule (100). Par conséquent, les clients 2 et 3 ne sont pas reliés.

Si les points 3 et 5, ce qui est la paire suivante dans la liste, ont été reliés à ce stade, nous serions construisons plus d'une route ($2-6$ et $3-5$). Puisque la version séquentielle de l'algorithme se limite à une seule voie à la fois, on fait abstraction de la paire de points 3 et 5.

La combinaison de la paire de points 5 et 6, les résultats dans le trajet $2-6-5$ avec une demande totale de 94. Cette combinaison est possible, et on établit la liaison entre 5 et 6 en tant que partie de la solution. En parcourant la liste, nous trouvons que, en raison de la restriction des capacités pas plus de points peuvent être ajoutés à l'itinéraire. Ainsi nous avons formé la route $1-2-6-5-1$. Dans le passage suivant de la liste des "savings" nous ne trouvons que la paire de points 3 et 4. Ces deux points peuvent être visités sur la même

route, et nous faisons le trajet $1-3-4-1$.



(3.10)

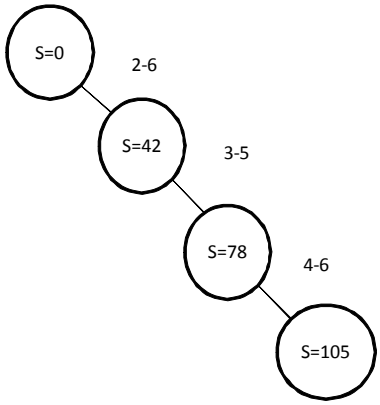
L'algorithme séquentiel a construit une solution avec deux routes. Les coûts de transport pour la route $1-2-6-5-1$ sont 98, et pour la route $1-3-4-1$ les coûts de transport sont 89. Les coûts totaux de transport sont donc 187.

La distance totale sauvée dts par cette version est 89. Alors, on peut obtenir les coûts totaux de transport par la relation (3.6)

$$\begin{aligned} 2 * \sum_{i=2}^{6} c_{1i} - dts &= 276 - 89 \\ &= 187. \end{aligned}$$

Version parallèle

Dans la version parallèle 2 et 6 sont également combinés en premier. Comme la version parallèle peut construire plus d'une voie à la fois, les points 3 et 5 sont également combinés. Enfin, les points 4 et 6 sont combinés. De cette façon, l'algorithme construit les routes $1-2-6-4-1$ avec coût 95 et $1-3-5-1$ et coût de 76 avec les frais de transport totaux s'élevant à 171.



(3.11)

La distance totale sauvée dts par cette version est 105. Alors on peut obtenir les coûts totaux de transport par la relation (3.6)

$$\begin{aligned} 2 * \sum_{i=2}^{6} c_{1i} - dts &= 276 - 105 \\ &= 171. \end{aligned}$$

Il est à noter que l'algorithme parallèle donne souvent de meilleurs résultats que l'algorithme séquentiel. Cependant, l'algorithme parallèle peut également impliquer plus de calcul dans le cadre de la gestion de plusieurs voies en même temps.

3.2 Résultats numériques

3.2.1 Instances et implémentation

L'algorithme de Clarke & Wright est codé en Matlab et exécuté sur un PC disposant d'un processeur Intel Core i3 avec une fréquence de 2.4 GHz et une mémoire vive de 4 Go tournant sous Microsoft Windows 7, et il est testé sur quelques instances de TSPLIB ([18], [19], [20]). Les instances sont des donnés de problèmes réels dont on connait la solution optimale ou une borne supérieure. En vue de réaliser des tests numériques, nous avons utilisé quelques instances pour le CVRP [22], dans le tableau (3.12), on a cité quelques instances de taille différentes entre 7 et 256. La colonne "inst" indique le nom de l'instance, la deuxième présente les noms des auteurs de l'instance. La colonne "best val" indique la meilleure valeur connue de l'objectif. Les colonnes "dim", "nbr veh" et "Q" présentent la dimension du problème (le nombre de clients plus le dépôt), le nombre minimal des véhicules et leur capacité. La dernière colonne "type dis" précise le type de poids de chaque arête (la distance) : s'il est explicite (matrice des coûts), ou chaque sommet (client) est définit par ces coordonnés euclidiens "Euc_2D", et les distances (c_{ij}) sont calculées en utilisant une norme euclidienne. Toutes les distances des instances de type "Euc_2D" sont arrondies, sauf celles de l'instance $E-n256-k14$.

inst	Auteurs	best val	dim	nbr veh	Q	type dis
$A-n33-k5$	Augerat et Al	661	33	5	100	Euc_2D
$A-n45-k7$	Augerat et Al	1146	45	7	100	Euc_2D
$A-n60-k9$	Augerat et Al	1354	60	9	100	Euc_2D
$A-n80-k10$	Augerat et Al	1763	80	10	100	Euc_2D
$E-n7-k2$	Eilon et Al	114	7	2	3	explicite
$E-n13-k4$	Eilon et Al	290	13	4	6000	explicite
$E-n22-k4$	Christophides et Eilon	375	22	4	6000	Euc_2D
$E-n33-k4$	Christophides et Eilon	835	33	4	8000	Euc_2D
$E-n51-k5$	Christophides et Eilon	521	51	5	160	Euc_2D
$E-n76-k7$	Christophides et Eilon	683	76	7	220	Euc_2D
$E-n76-k8$	Christophides et Eilon	735	76	8	180	Euc_2D
$E-n76-k10$	Christophides et Eilon	847	76	10	140	Euc_2D
$E-n76-k14$	Christophides et Eilon	1032	76	14	100	Euc_2D
$E-n101-k8$	Christophides et Eilon	825	101	8	200	Euc_2D
$E-n101-k14$	Christophides et Eilon	1077	101	14	112	Euc_2D
$E-n256-k14$	Golden et Al	587.092	256	14	1000	Euc_2D
$F-n45-k4$	Fisher	724	45	4	2010	Euc_2D
$F-n72-k4$	Fisher	238	72	4	30000	Euc_2D
$F-n135-k7$	Fisher	1165	135	7	2210	Euc_2D
$gr17$	Grotschel	2685	17	3	6	explicite

(3.12)

On a testé les deux versions "parallèle et séquentielle" pour les deux cas de l'heuristique de Clarke & Wright : le premier, le nombre minimal de véhicules nécessaires (qui est aussi le "nbr veh" dans la tableau (3.12)) est imposé et non imposé dans le second cas.

3.2.2 Résultats

Les résultats fournis par notre code pour le premier cas sont illustrés dans le tableau suivant :

inst	nombre de véhicules		imposé		best
	sol par	temps	sol séq	temps	val
$A-n33-k5$	696	0.047	864	0.0512	661
$A-n45-k7$	1236	0.0723	1352	0.0813	1146
$A-n60-k9$	1363	0.1136	1607	0.1545	1354
$A-n80-k10$	1851	0.2554	2059	0.2898	1764
$E-n7-k2$	119	0.0292	119	0.0272	114
$E-n13-k4$	290	0.0309	290	0.0291	290
$E-n22-k4$	387	0.0349	458	0.0291	375
$E-n33-k4$	942	0.0479	954	0.0465	835
$E-n51-k5$	595	0.0807	733	0.0905	521
$E-n76-k7$	766	0.1993	933	0.2279	683
$E-n76-k8$	824	0.1962	986	0.2440	735
$E-n76-k10$	939*(68)	0.2116	1013*(68)	0.2636	847
$E-n76-k14$	1067*(76)	0.2218	1190*(7, 76)	0.3122	1032
$E-n101-k8$	948	0.442	1086	0.5362	825
$E-n101-k14$	1112	0.4376	1364	0.6545	1077
$E-n256-k14$	620.73*(2)	14.7427	854.1473	14.6755	587.092
$F-n45-k4$	842	0.0649	1328	0.0666	724
$F-n72-k4$	369*(12)	0.1638	738	0.1659	238
$F-n135-k7$	1713*(104)	1.2481	2059	1.6508	1165
$gr17$	2685	0.0311	2769	0.0307	2685

(3.13)

Dans le tableau 3.13, la colonne intitulée "sol par" donnent les solutions de la version parallèle, et "sol seq" donnent les solutions de la version séquentielle. Les solutions marquées avec (*) sont non réalisables, par exemple, la solution "939*(68)" de l'instance $E-n76-k10$ signifie que le client (nœud) 68 n'est pas affectés à une tournée parmi les 10 tournées (il est affecté dans une tournée tout seule) ; on remarque que les solutions obtenues par notre code sont approchées de celles de [22], sauf quelques unes, qui ne sont pas réalisables, on remarque aussi que les solutions de la version séquentielle pour les instances de Fisher sont éloignées un petit peu. Il est aussi à noter que le temps CPU est très petit. Pour le deuxième cas, voici

le tableau 3.14 où, la colonne "nbr tour" représente le nombre de tournées réalisées.

inst	nombre de véhicules			non imposé			best
	sol par	temps	nbr tour	sol seq	temps	nbr tour	val
$A-n33-k5$	696	0.0384	5	864	0.0585	5	661
$A-n45-k7$	1251	0.0543	8	1352	0.1091	7	1146
$A-n60-k9$	1375	0.0918	10	1607	0.2032	9	1354
$A-n80-k10$	1851	0.19	10	2059	0.4371	10	1764
$E-n7-k2$	119	0.0254	2	119	0.0265	2	114
$E-n13-k4$	290	0.0271	4	290	0.0311	4	290
$E-n22-k4$	387	0.0316	4	458	0.0365	4	375
$E-n33-k4$	985	0.0381	5	954	0.0561	4	835
$E-n51-k5$	591	0.0666	7	733	0.1264	5	521
$E-n76-k7$	795	0.1640	9	933	0.3510	7	683
$E-n76-k8$	847	0.1642	10	986	0.3654	8	735
$E-n76-k10$	908	0.1626	11	1013	0.3818	11	847
$E-n76-k14$	1067	0.1648	15	1186	0.4219	15	1032
$E-n101-k8$	998	0.3858	12	1086	0.8277	8	825
$E-n101-k14$	1112	0.3842	14	1364	0.9328	14	1077
$E-n256-k14$	620.73	12.5516	15	854.1473	20.3904	14	587.092
$F-n45-k4$	961	0.0543	6	1328	0.0836	4	724
$F-n72-k4$	326	0.1436	7	738	0.2994	4	238
$F-n135-k7$	1457	1.0458	13	1947	2.338	7	1165
$gr17$	2685	0.0279	3	2769	0.0316	3	2685

(3.14)

Chapitre 4

Problème de livraison de fioul

4.1 Introduction

Le problème de livraison de fioul est un cas particulier de CVRP, qui a pour but d'optimiser le parcours des camions-citernes identiques basés au dépôt ($i = 1$) devant effectuer des tournées de livraisons de fioul à $n - 1$ clients pour satisfaire leur demandes.

4.2 Modélisation

Soient des variables binaires x_{ij} valant 1 si la ville i est visitée avant la ville j dans une tournée. 0 sinon. Notant n le nombre de villes, D_{ij} la distance séparant les deux villes i et j, Q_i la quantité de fioul demandée par le client i, $Q_{\max}$ la capacité identique des camions-citernes ([7]). Le modèle mathématique de ce problème est inspiré de la deuxième formulation de CVRP (au chapitre 2) :

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} D_{ij} x_{ij} \tag{4.1}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1, \forall j = 2...n \tag{4.2}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = 1, \forall i = 2...n \tag{4.3}$$

$$Q_i \leq c_i \leq Q_{\max}, \ \forall i = 2...n \tag{4.4}$$

$$c_i \leq Q_{\max} - (Q_{\max} - Q_i)\, x_{1i}, \ \forall i = 2...n \tag{4.5}$$

$$c_j \geq c_i + Q_j - Q_{\max} + Q_{\max}.x_{ij} + (Q_{\max} - Q_j - Q_i)\, x_{ji}, \forall i = 2...n, \forall j = 2...n, i \neq j \tag{4.6}$$

$$c_i \geq 0, \ \forall i = 2...n \tag{4.7}$$

$$x_{ij} = \{0, 1\}, \ \forall i = 1...n, \ \forall j = 1...n. \tag{4.8}$$

L'objectif est de minimiser le nombre de kilomètres parcourus (4.1). Chaque ville doit être livrée en une seule fois. Ceci se traduit par les deux contraintes (4.2) et (4.3) qui imposent

respectivement qu'on doit entrer une et une seule fois dans chaque ville (sauf le dépôt) et qu'on doit quitter chaque ville (sauf le dépôt) une et une seule fois.

Afin de respecter la capacité des camions-citernes, nous allons utiliser des variables continues et positives c_i (c comme cumul) correspondant à la quantité de fioul livrée aux différents clients le long du trajet allant du dépôt au client i inclus. Cette quantité c_i doit être supérieure à la quantité Q_i demandée par le client i et inférieure à la capacité $Q_{\max}$ des camions-citernes (4.4). De plus, si le client i est le premier de la tournée, alors c_i doit être égal à la quantité demandée par le client i ($c_i = Q_i$). Cette contrainte se traduit par les deux contraintes (4.4) et (4.5). En effet, si i est le premier client d'une tournée, alors x_{1i} vaut 1 et, aprés simplification, la contrainte (4.5) est équivalente à

$$c_i \leq Q_i. \tag{4.9}$$

Alors (4.9) et (4.4) impliquent que c_i est égal à la demande du client i. Dans le cas où i n'est pas le premier de la tournée, x_{1i} vaut 0 et la contrainte (4.5) est équivalente à la contrainte (4.4). Considérons maintenant le cas où i n'est pas le premier client de la tournée. Alors c_i doit être égal à la somme des quantités livrées entre le dépôt et i inclus. Ainsi, si le client j est après le client i sur une tournée, on peut écrire que c_j doit être égal à la quantité livrée sur le trajet du dépôt à i inclus, quantité à laquelle s'ajoute la quantité demandée par j ($c_j = c_i + Q_j$). Cette relation est traduite par la contrainte (4.6). En effet, si j est juste après i sur une tournée, x_{ij} vaut 1 et x_{ji} vaut 0. La contrainte (4.6) est équivalente à la contrainte (4.10)

$$c_j \geq c_i + Q_j. \tag{4.10}$$

Dans le cas où j n'est pas juste après i, la contrainte (4.6) reste vérifiée. En effet, dans le cas où j est juste avant i, la contrainte (4.6) devient (4.11)

$$c_j \geq c_i - Q_i. \tag{4.11}$$

Cette contrainte signifie que la quantité livrée sur le trajet du dépôt à j est supérieure à la quantité livrée entre le dépôt et le successeur i de j sur la tournée, quantité à laquelle on doit retirer la quantité livrée en i ($c_j = c_i - Q_i$). Enfin, si i et j ne sont pas côte à côte sur une tournée, on obtient la contrainte (4.12). Comme elle a un second membre inférieur ou égal à Q_j, cette contrainte est redondante puisque elle est déjà exprimée par la contrainte (4.4)

$$c_j \geq c_i + Q_j - Q_{\max}, \tag{4.12}$$

car

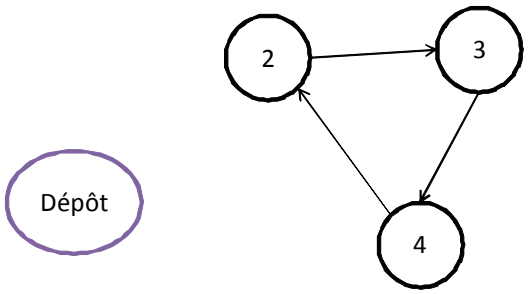
$$c_i - Q_{\max} \leq 0,$$

donc

$$c_i - Q_{\max} + Q_j \leq Q_j \leq c_j.$$

Enfin, les contraintes (4.7) et (4.8) indiquent que les variables c_i sont positives et que les x_{ij} sont des variables binaires. Notons pour terminer que l'affectation de variables c_i à chaque sommet i garantit le respect de la capacité des camions, tout en interdisant la création de tournées ne passant pas par le dépôt. En effet, sans ces variables, il serait possible d'obtenir

une solution comme celle de la figure (4.13)



Exemple de solution non realisable

(4.13)

Cette solution vérifie les deux contraintes (4.2) et (4.3) puisque ; on entre et on quitte chaque sommet une et une seule fois, mais elle n'est pas réalisable puisque la tournée proposée ne passe pas par le dépôt. Le fait d'affecter des valeurs c_i strictement croissantes le long d'une tournée permet d'interdire ce genre de solution. Ce modèle est obtenu en retirant certaines contraintes de la formulation ($MILP$) du (VRP), en notant que le nombre de véhicules (des camions-citernes) n'est pas déterminer et en prenant les variables $u_i = c_i$.

4.3 Exemple

Un transporteur doit livrer du fioul à un certain nombre de clients de la région Ouest à partir de la raffinerie de Donges (D). Ses clients se situent à Brain-sur-l'Authion (B-A), Carquefou (C), Guérande (G), la Haie Fouassière (H-F), Mésanger (M) et les Ponts-de-Cé (P-C). Le tableau suivant contient les demandes en litres pour les différents sites de ce problème

B-A	C	G	H-F	M	P-C
14000	3000	6000	16000	5000	15000

(4.14)

La matrice des distances en kilomètres séparant les clients est donnée au tableau 4.15

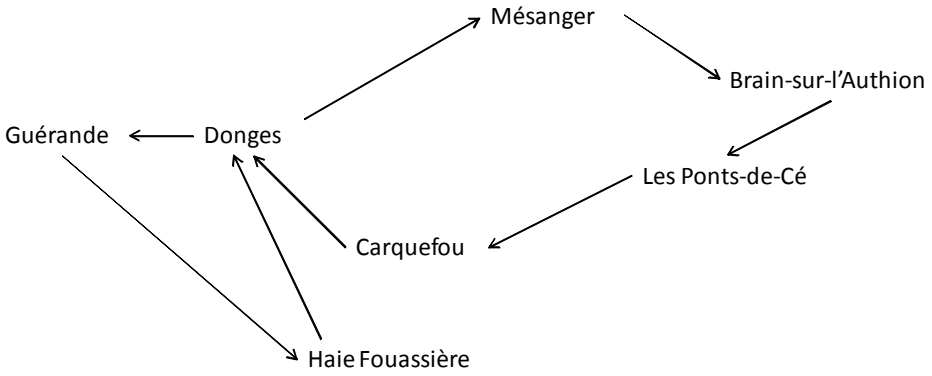
	D	B-A	C	G	H-F	M	P-C
D	–	148	55	32	70	73	140
B-A	148	–	93	180	99	72	12
C	55	93	–	85	20	28	83
G	32	180	85	–	100	99	174
H-F	70	99	20	100	–	49	85
M	73	72	28	99	49	–	73
P-C	140	12	83	174	85	73	–

(4.15)

Pour faire ses livraisons, le transporteur dispose de camions-citernes pouvant contenir jusqu'à 39000 litres. Déterminer les tournées à réaliser pour livrer tous les clients de façon à minimiser le nombre de kilomètres parcourus.

La solution optimale de cet exemple ([7]) consiste à faire deux tournées. La première dessert Guérande, puis la Haie- Fouassière. La seconde livre en premier le client de Mésanger, puis ceux de Brain-sur-l'Authion et des Ponts-de-Cé pour terminer par celui situé à

Carquefou. Dans la première tournée, 22000 litres de fioul sont livrés, 37000 dans la seconde. Les tournées sont représentées sur la figure 4.16. Le nombre total de kilomètres parcourus est 497.



Tournees optimales

(4.16)

4.4 Application de l'heuristique de Clarke et Wright au problème de livraison de fioul

Le tableau (4.17) illustre les résultats fournis par le code Matlab de l'heuristique de Clarke & Wright sur l'exemple de livraison de fioul cité dans la section 3 de ce chapitre, où on a comparé entre la solution exacte de cet exemple et les solutions obtenues par les deux versions de cet algorithme dans les deux cas : le premier le nombre de véhicules nécéssaires est imposer et dans le deuxième non. Les villes sont numérotées de 1 à 7 (1 est la raffinerie de Donges)

heuristique	nbr véhicules	imposé	nbr véhicules	non imposé
	version par	version séq	version par	version séq
tournées	$[1,6,2,7,3,1]$ $[1,4,5,1]$	$[1,6,2,7,3,1]$ $[1,4,5,1]$	$[1,6,2,7,3,1]$ $[1,4,5,1]$	$[1,6,2,7,3,1]$ $[1,4,5,1]$
nvu	2	2	2	2
nvn	2	2	2	2
q.f.l.t	$[37000, 22000]$	$[37000, 22000]$	$[37000, 22000]$	$[37000, 22000]$
val parcours	497	497	497	497
temps CPU en (s)	0.0319	0.0281	0.0254	0.0295

(4.17)

Tous les cas testés de l'heuristique de Clarke & Wright donnent la même solution qui est la solution optimale. D'après ce tableau, deux camions-citernes sont utilisés pour faire les deux tournées (nvu), la première commence au dépôt, ensuite les villes $6-2-7-3$, et le véhicule se termine au dépôt, c'est à dire les villes Donges- Mésanger- Brain-sur-l'Authion- Ponts-de-Cé- Carquefou- Donges avec 37000 litres de fioul livrés (q.f.l.t) dans cette tournée, et la deuxième, $1-4-5-1$ (Donges- Guérande- Haie Fouassière- Donges) avec 22000 litres livrés (q.f.l.t) et 497 de kilomètres parcourus dans les deux tournées (val parcours). La seule différence entre les deux cas par leurs deux versions est en temps CPU (en secondes), mais

elle n'est pas très importante, on remarque que dans les cas où le nombre de véhicules est imposé, la version parallèle met plus de temps que la version séquentielle, mais elle met moins de temps que la deuxième version quand le nombre de véhicules n'est pas imposé.

Conclusion

Dans ce mémoire, on a appliqué les deux versions de l'heuristique de Clarke & Wright : version séquentielle et version parallèle pour résoudre le problème de livraison de fioul où le nombre de tournées est imposé dans le premier cas, et dans le deuxième cas non imposé. D'après les résultats fournis par notre code Matlab pour l'exemple réel de chapitre 4, on conclut que l'heuristique de Clarke & Wright est une bonne méthode pour résoudre ce type de problème, surtout qu'il est de taille petite (6 villes et un dépôt), car elle a fourni la solution optimale de cet exemple en un temps très petit. Les résultats des instances de TSPLIB obtenus par le code sont en général très bonnes et très proches de la meilleure solution connue. Cependant, les résultats de quelques instances sont non réalisables, et d'autres sont éloignés.

L'heuristique de Clarke & Wright reste une méthode approchée, comme perspectives nous proposons d'étudier la méthode de branchement et coupes "Branch & Cut" qui demeure la plus efficace et la plus utilisée dans ce contexte actuellement ([3], [14]).

Bibliographie

[1] T, Bektas, **The multiple traveling salesman problem : an overview of formulations and Solution procedure**, Omega 34 (2006), 209-219.

[2] R. Belgacem, **Sur Quelques Méthodes de Résolution pour le TSP " Travelling Salesman Problem ",** Mémoire de Magister en mathématiques, Option : Analyse des Systèmes, Contrôle et Optimisation Numérique, Université de Mostaganem, 2012.

[3] L. Cacceta, S. P. Hill, **Branch and Cut Methods for Network Optimization**, Mathematical and Computer Modelling 33, 517-532, 2001.

[4] G. Clarke, J. W. Wright, **SCHEDULING OF VEHICLES FROM A CENTRAL DEPOT TO A NUMBER OF DELIVERY POINTS**, Operations Research, 12(4), 568-581, 1964.

[5] P. Fouilhoux, **Programmation mathématique Discrète et Modèles Linéaires,** Université Pierre et Marie Curie, Master IAD, Module PDML, 29 septembre 2013.

[6] B. L. Golden, **VEHICLE ROUTING PROBLEM : FORMULATIONS AND HEURISTIC SOLUTION TECHNIQUES**, Technical Report NO. 113, OPERATIONS RESEARCH CENTER, Massachusetts Institue of Technology, August 1975.

[7] C. Guéret, C. Prins, M. Sevaux, **Programmation linéaire**, 65 problème d'optimisation modélisés et résolus avec Visual Xpress, édition Eyrolles 2 éme tirage, 2003.

[8] I. Kara, T. Bektas, **Integer Linear Programming Formulation of the Generalized Vehicle Routing Problem**, Presented in 5^{th} EURO/ INFORMS joint International Meeling, july 6-10 2003 Istanbul, Turkey.

[9] I. Kara, G. Laporte, T. Bektas, **A note of the lifted Miller- Tucker- Zemlin subtour elimination constraints for the capacitated vehicle routing problem**, European Journal of Operational Research, 158 (2004), 793-795.

[10] G. Laporte, **The Vehicle Routing Problem : An overview of exact and approximate algorithms**, European Journal of Operational Research 59, 345-358, 1992.

[11] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys, **THE TRAVELING SALESMAN PROBLEM,** A Guided Tour of Combinatorial Optimization, 1985.

[12] Lec_29_ Vehicle Routeing Problem - - - you tube mp4.

[13] J. Lysgaard, **Clarke & Wright's Savings Algorithm**, Department of Management Science and Logistics, The Aarhus School of Business, September 1997.

[14] J. Lysgaard, A. N. Letchford, R. W. Eglese, **A new Branch-and-Cut Algorithm for the Capacitated Vehicle Routing Problem**, Octobre 28, 2003.

[15] S. Maraj, **Distribution de fioul**, Mémoire de Master en mathématiques, Option : Modélisation, Contrôle et Optimisation, Université de Mostaganem, 2014.

[16] **Métaheuristique**, URL : http ://fr.wikipedia.org/wiki/métaheuristique.

[17] F. Odonez, I. Sungur, M. Dessouky, **A priori performance measures for arc-based formulations of the Vehicle Routing Problem**, Industrial and Systems Engineering, University of Southern California, 2006.

[18] G. Reinelt, **TSPLIB : The Travelling Salseman problem library**, URL : http ://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html.

[19] G. Reinelt, **TSPLIB, The Travelling Salseman problem library**, URL : http ://elib.zib.de/pub/mp-testdata/tsp/tsplib/vrp/index.html.

[20] M. Sevaux, M. J Geiger, **Inventory Routing and On- line Inventory Routing File Format**, janvier 2011.

[21] G. Srinivasan, **Lecture - 29 Vehicle Routing Problem**, Advanced Operations Research, Department of Management Studies, Indian Institute of Technology, Madras, http :// text of video.nptel.ac.in/112106131/lec 29.pdf.

[22] **The VRP Web**, http ://www.bernabe.dorronsoro.es/vrp/.