N^o d'ordre : 04/2011 - M / MT

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET
DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE
HOUARI BOUMEDIENE
FACULTE DE MATHEMATIQUES

Présenté pour l'obtention du diplôme de MAGISTER
En : MATHEMATIQUES
Spécialité : Recherche Opérationnelle (Génie Mathématique)
Par : BOUZID Mouaouia Cherif

Sujet

Sur une relation entre la coloration de graphes et le problème du voyageur de commerce multiple

Soutenu publiquement le 10/09/2011, devant le jury composé de :

MOULAÏ Mustapha	Professeur à l'USTHB	Président
AIT HADDADENE Hacène	Professeur à l'USTHB	Directeur de Thèse
BENMEZIANE Zineb	Maître de Conférences A à l'USTHB	Examinatrice
MECHEBBEK Meriem	Maître Assistante A à l'USTHB	Invitée

À mes parents
À mes grands parents
À mes soeurs

Sans Toi, rien ne pourrait être
Louanges à Toi, Seigneur des mondes

Remerciements

Beaucoup de personnes ont contribué à l'aboutissement de ce travail, je tente ici de leur montrer ma gratitude :

Je commence par vous, Mr AIT HADDADENE, qui avait accepté d'être mon directeur de thèse. Je ne serai vous remercier de m'avoir fait bénéficier de votre expérience et de m'avoir guidé tout au long de cette année.

Je remercie Mr MOULAÏ de m'avoir fait l'honneur d'être le président de mon jury.

Que Mme BENMEZIANE et Mlle MECHEBBEK trouvent ici ma profonde reconnaissance d'avoir accepté d'examiner ce travail et d'être les membres de mon jury.

Je ne pourrais remercier assez Mr BELBACHIR et Mr YAGOUNI pour l'aide qu'ils m'ont apportée.

Je tiens à remercier Mr SALHI ainsi que Mr BEKTAS d'avoir correspondu avec moi et de m'avoir éclairé par leur savoir.

Je n'oublie pas également de remercier l'équipe de LEMON library qui a toujours répondu à mes interrogations concernant cette bibliothèque de théorie des graphes.

Je remercie tata Aïcha, Djalel, Samy et Mahdia pour leur aide précieuse et les longues heures de relecture. Sans eux, le présent document n'aurait pas vu le jour.

Un grand merci à maman, papa et à tous mes proches parents. Sans vous, ce n'est pas le document mais l'auteur qui n'aurait pas vu le jour.

Enfin, merci à tous ceux qui, de loin ou de près, ont participé à l'aboutissement de ce travail mais qui n'ont pas été cité. Je ne saurai vous remercier assez.

Table des matières

Liste des abréviations

ACVRP Asymetric Capacitated Vehicle Routing Problem ;

ATSP Asymetric Traveling Salesman Problem ;

BPP Bin Packing Problem ;

CVRP Capacitated Vehicle Routing Problem ;

MCE Méthode de Clustering d'Elghazel et al. ;

m**-TSP** Multiple Traveling Salesman Problem ;

SCVRP Symetric Capacitated Vehicle Routing Problem ;

SEP Séparation Evaluation Progressive ;

STSP Symetric Traveling Salesman Problem ;

TSP Traveling Salesman Problem ;

VRP Vehicle Routing Problem ;

VRPTW Vehicle Routing Problem with Time Windows ;

Table des figures

Liste des tableaux

Introduction

Qu'il est stressant pour un chef d'entreprise, peu informé des techniques d'optimisation, d'organiser la livraison de ses produits. Bien que disposant d'un parc de véhicules capables de livrer tous les clients en une seule fois, déterminer un routage des véhicules qui soit économique et qui passe par chacun des clients tout en ramenant les véhicules en fin de journée au parc est une tâche complexe. Le chef d'entreprise pourrait énumérer toutes les solutions au problème mais cela devient impraticable lorsque le nombre de villes et de véhicules augmentent légèrement. En effet, en supposant qu'il dispose de 3 véhicules et de 15 villes à visiter, le chef d'entreprise doit évaluer pas moins de 19833061248000 solutions possibles et retenir la meilleure. Cela prendrait bien entendu trop de temps, d'autant plus que les clients changent quotidiennement.

Le problème auquel est confronté le chef d'entreprise est un problème d'optimisation combinatoire. On peut, comme lui, résoudre les problèmes d'optimisation combinatoire en énumérant toutes les solutions mais cela est rarement envisageable car le nombre des solutions est généralement très élevé.

Ce problème de distribution est connu sous le nom de *problème du voyageur de commerce multiple* ou «multiple traveling salesman problem» en Anglais, noté m-TSP. Il est à l'intersection de deux problèmes d'optimisation combinatoire plus connus qui sont le *problème du voyageur de commerce* et le *problème de tournés de véhicules*. Depuis l'heuristique de Russel en 1977 (cf. [53]), différentes approches de résolution du m-TSP ont été proposées. Tantôt approchées et tantôt exactes, elles ont résolu des instances allant jusqu'à 500 villes et 10 véhicules (cf. [6]).

Le m-TSP peut être abordé de plusieurs manières. L'une d'elles est d'abord, de partager les villes à visiter sur les différents véhicules ; puis, de chercher une tournée optimale pour chacun des véhicules. Dans cette approche, on comprend que l'étape du partage des villes est cruciale pour espérer obtenir une bonne solution.

Le partage d'objets en groupe est un domaine abondamment étudié en mathématiques. Si les classes auxquelles les objets doivent être affectés sont connues a priori alors on parle de *classification* sinon on parle de *clustering*.

En 2006, Elghazel et al. ont développé une nouvelle méthode de clustering basée sur la théorie des graphes que nous notons MCE pour Méthode de Clustering d'Elghazel et al. (cf. [18]). Cette méthode s'est montrée compétitive sur une batterie de tests (cf. [20]) et fut appliquée à divers domaines (cf. [22] et [25]).

L'objectif de notre étude est d'utiliser MCE pour générer une partition intéressante des villes à visiter dans un m-TSP. Cette partition servirait alors a construire un routage pour le m-TSP.

Le document est organisé comme suit :

Dans le premier chapitre, nous rappelons des notions de théorie des graphes et de complexité algorithmique.

Dans le deuxième chapitre, nous présentons littéralement et formellement le problème du voyageur de commerce et ses extensions, dont le m-TSP. Nous donnons également un aperçu de l'évolution des méthodes, exactes et approchées, développées pour ce dernier.

Dans le troisième chapitre, nous présentons les outils que nous utilisons pour aborder le m-TSP. Des concepts de théorie des graphes, de classification et de clustering sont rappelés. Un descriptif détaillé de MCE est donné, accompagné d'une justification des algorithmes et du rappel de leurs complexités.

Dans le quatrième chapitre, nous expliquons nos motivations et la manière dont nous souhaitons appliquer MCE au m-TSP.

Dans le cinquième chapitre, nous analysons les résultats obtenus en appliquant notre approche sur des exemples et des instances connues dans la littérature. Enfin, nous tirons des conclusions et nous dévoilons une perspective de recherche.

Chapitre 1

Préliminaires

Nous rappelons dans ce chapitre des notions de théorie des graphes et de complexité algorithmique. Le lecteur averti est invité à aller directement au chapitre 2.

1.1 Notions de théorie des graphes

Nous nous basons pour définir ces notions sur les cours de Abbas [1] et Maquin [42].

Graphes

Un graphe $G = (V, U)$ est une structure constituée par un ensemble fini et non vide V d'éléments appelés *sommets* et d'une famille U de couples de sommets appelés *arcs*.

Le cardinale de V est appelé *ordre* du graphe G.

En présence d'un arc $u = (x, y)$ de E, on dit que x (resp. y) est l'extrémité initiale (resp. terminale) de u. Si l'on considère toujours le même arc, le sommet y est appelé *successeur* du sommet x et ce dernier est appelé *prédécesseur* du sommet y ; on dit également que les sommets x et y sont *voisins* ou *adjacents*.

Un arc dont les extrémités se confondent est appelé boucle. Si il existe plusieurs arcs de la forme (x, y), ils sont appelés arcs parallèles.

L'*ensemble des voisins* d'un sommet x, noté $N(x)$, est l'ensemble des sommets de V auxquels il est adjacent. Le cardinal de $N(x)$ compte le nombre de voisins de x, on l'appelle *degré* du sommet x et on le note $d(x)$. Le degré du sommet ayant le plus de voisins dans G est appelé le *degré maximum* ; on le note Δ_G.

On peut associer aux arcs du graphe G une pondération $l : U \to \mathbb{R}$ et on note le poids de l'arc $u = (x, y)$ par $l(u)$ ou l_{xy} lorsqu'il n'y a pas d'ambiguïté.

Concept non orienté

Certaines propriétés des graphes ne font pas intervenir l'orientation des arcs, autrement dit, l'ordre des extrémités dans les arcs n'est pas important. Dans ce cas, en présence d'une paire de sommets x et y, on cherche à savoir uniquement si x et y sont en relation ou non sans considérer un ordre entre x et y. Une relation entre deux sommets x et y définit une *arête* notée (x, y) ou xy. Le graphe G est dit non orienté et on note $G = (V, E)$ où E est l'ensemble des arêtes.

Sous graphes, graphes partiels et sous graphes partiels

Soit $G = (V, U)$ un graphe,

- Soit $S \subseteq V$, le graphe $G[S] = (S, U_S)$ est dit *sous graphe de G engendré* (induit) par S, où $U_S = \{u \in U \,/\, \text{les extrémités de } u \text{ appartiennent à } S\}$;
- Soit $U' \subseteq U$, $G' = (V, U')$ est appelé *graphe partiel de G engendré* (induit) par U' ;
- Soient $S \subseteq V$ et $U'_S \subseteq U_S$ alors $G'[S] = (S, U'_S)$ est appelé *sous graphe partiel de G*.

Chaînes, cycles, élémentaires et simples

Etant donné un graphe $G = (V, U)$,

Une *chaîne* dans G est une séquence finie et alternée de sommets et d'arcs, débutant et finissant par des sommets, telle que tout arc est incident aux sommets qui l'encadrent dans la séquence.

Un *cycle* est une chaîne dont les extrémités coïncident.

Si tout sommet n'apparaît qu'une fois dans la séquence qui forme la chaîne (respectivement cycle) alors la chaîne (resp. cycle) est dite *élémentaire*.

Si tout arc n'apparaît qu'une fois dans la séquence qui forme la chaîne (resp. cycle) alors la chaîne (resp. cycle) est dite *simple*.

La *longueur* d'une chaîne (resp. cycle) est donnée par le nombre d'arcs qui la forment ou par la somme de leurs poids si les arcs du graphe sont pondérées.

Un graphe $G = (V, U)$ est dit *connexe* ssi pour tout couple de sommets x et y de V, il existe une chaine de x à y.

Chemins, circuits, élémentaires et simples

Etant donné un graphe $G = (V, U)$,

Un *chemin* dans G est une séquence finie et alternée de sommets et d'arcs, débutant et finissant par des sommets, telle que tout arc dans la séquence est sortant du sommet qui le précède et incident au sommet qui le suit dans la séquence.

Un *circuit* est un chemin dont les extrémités coïncident.

Si tout sommet n'apparaît qu'une fois dans la séquence qui forme le chemin (resp. circuit) alors le chemin (resp. circuit) est dit *élémentaire*.

Si tout arc n'apparaît qu'une fois dans la séquence qui forme le chemin (resp. circuit) alors le chemin (resp. circuit) est dit *simple*.

La *longueur* d'un chemin (resp. circuit) est donnée par le nombre d'arcs qui le forment ou par la somme de leurs poids si les arcs du graphes sont pondérés.

Cocycle

Etant donné un graphe $G = (V, E)$ et S un sous ensemble de sommets de V, le *cocycle* de S est défini par l'ensemble des arcs (arêtes) qui relient S et $V \setminus S$.

Certains types de graphes

Un graphe est dit *simple* si il ne contient ni boucle, ni arcs (arêtes) parallèles.

Un graphe est dit *complet* si tout couple de sommets est relié par au moins un arc (arête).

Un graphe simple et complet est appelé *clique*.

Un *arbre* est un graphe connexe et sans cycle.

Etant donné un graphe $G = (V, U)$, un *arbre maximal* de G, appelé également *arbre couvrant* de G, est un graphe partiel de G qui est un arbre.

1.2 Notions de complexité algorithmique

Nous nous basons principalement dans cette partie sur la contribution de Bachelet [5] et le livre de Allab [3].

Intérêt de la théorie de la complexité algorithmique

Un algorithme est une suite finie d'instructions qui reçoit en entrée une chaîne de caractères (données d'un problème) et qui retourne en sortie une autre chaîne de caractère (solution si trouvée).

Pour un problème donné, il peut exister plusieurs algorithmes de résolution. La comparaison des performances des algorithmes doit se faire indépendamment de la performance des machines sur lesquelles ils sont programmés. La théorie de la complexité traite de la difficulté (ou complexité) qu'ont les algorithmes à résoudre des problèmes. Cette complexité peut être mesurée par rapport aux temps nécessaire pour exécuter l'algorithme sur une instance de taille donnée, on parle alors de complexité temporelle ou par rapport à la mémoire utilisée, on parle alors de complexité spatiale.

La complexité peut être mesurée de plusieurs manières :

- Dans le meilleur des cas : L'analyse de l'algorithme se fait selon le cas le plus favorable ;
- En moyenne : On analyse le comportement moyen de l'algorithmes sur plusieurs instances du problème tirées aléatoirement ;
- Dans le pire des cas : L'analyse est faite selon le cas le plus défavorable.

Dans la suite de ce document, le terme «complexité» désignera la complexité temporelle dans le pire des cas.

Types de problèmes

Différents types de problèmes existent dans littérature, nous citons les suivants :

- **Problèmes de reconnaissance** Appelés également problèmes de décision, ce sont des problèmes dont la solution est oui ou non ;
  Exemple. Etant donné un entier k, un graphe $G = (V, U)$ admettant une pondération des arcs et deux sommets de G, x et y. Le problème "Existe t-il un chemin dans G allant de x à y de longueur inférieure ou égale à k ?" est un problème de reconnaissance.
- **Problèmes d'optimisation** Ce sont des problèmes où l'on cherche une solution qui soit optimale pour un critère et qui respecte un certain nombre de contraintes ;

Exemple. Etant donné un graphe $G = (V, U)$ admettant une pondération des arcs et deux sommets de G, x et y. Le problème "Trouver dans G un chemin allant de x à y de longueur minimum" est un problème d'optimisation.

Problème de reconnaissance associé à un problème d'optimisation

Etant donné le problème d'optimisation O :

$$(O) \begin{cases} \text{Minimiser} & f(x) \\ \text{sous contrainte} & x \in S \end{cases}$$

On peut associer à O le problème de reconnaissance suivant :

"Etant donné un réel k, existe t-il dans S un x tel que $f(x) \leq k$?"

Fonction dominée par une autre

Définition. *Soient f, g deux fonctions définies dans un voisinage du point x_0, sauf peut être en x_0.*

On dit que f est dominée *par g lorsque x tend vers x_0, et l'on écrit $f = O(g)$, si*

$$\exists k > 0, \exists \delta > 0, \forall x[0 < |x - x_0| < \delta \Rightarrow |f(x)| \leq k|g(x)|]$$

Définition. *Soient f, g deux fonctions définies dans $]a, +\infty[$ alors*

$$f = O(g) \text{ lorsque } x \to +\infty \Leftrightarrow \exists k > 0,\ \exists \delta > 0, \forall x[x > \delta \Rightarrow |f(x)| \leq k|g(x)|]$$

Le symbole O s'appelle notation de Landau.

Mesure de la complexité d'un algorithme

Etant donné une instance I d'un problème P de taille n et un algorithme de résolution A pour le problème P, on cherche à borner le nombre d'opérations effectuées par l'algorithme A pour résoudre l'instance I, noté $T_A(n)$, dans le pire des cas.

Par hypothèse, les opérations d'écriture, de lecture, d'affectations et les opérations élémentaires (addition, soustraction, multiplication, division) sont considérées de complexité constante ($O(1)$).

Pour mesurer la complexité de l'algorithme, on analyse la complexité des différentes instructions selon les propriétés de la dominance.

La complexité de l'algorithme A varie selon la fonction qui domine $T_A(n)$ à l'infini :

- $T_A(n) = O(1)$: complexité constante ;
- $T_A(n) = O(n)$: complexité linéaire ;
- $T_A(n) = O(\log n)$: complexité logarithmique ;
- $T_A(n) = O(2^n)$: complexité exponentielle ;
- etc.

Si $T_A(n)$ est dominé à l'infini par un polynôme alors l'algorithme A est dit *polynômial*.

Classes P, NP et NP-complets

Les problèmes de reconnaissance pour lesquels on connaît un algorithme polynômial de résolution constituent la classe **P**. On dit également que les problèmes de cette classe sont *faciles*.

La classe des problèmes **NP** est la classe des problèmes pour lesquels on peut vérifier une solution par un algorithme polynomial. La classe **NP** contient la classe **P**.

Etant donné deux problèmes $P1$ et $P2$, on dit que $P1$ se *réduit* à $P2$ si on peut résoudre $P1$ en faisant appel à un algorithme de résolution de $P2$ comme procédure. Cette réduction est dite *polynomiale* si l'algorithme pour $P1$ est polynomial en comptant l'appel à l'algorithme de $P2$ comme une opération élémentaire (de complexité $O(1)$).

Un problème est dit **NP**-complet si il est dans **NP** et si tous les problèmes dans **NP** se réduisent polynomialement à lui.

Chapitre 2

Le problème du voyageur de commerce et ses extensions

2.1 Introduction

Parmi tous les problèmes de l'optimisation combinatoire, il en est un qui a fait coulé beaucoup d'encre. Ce problème, à l'énoncé très simple mais à la difficulté reconnue, s'appelle le problème du voyageur de commerce. On remonte la première description du problème à un manuel allemand de mathématiques daté de 1832 (cf. [55]). Certains auteurs pensent que ce problème a été étudié formellement pour la première fois par K. Menger en 1932 (cf. [29], [55]). Menger remarque qu'on peut résoudre le problème en énumérant toutes les solutions mais que cela devient difficile lorsque l'instance est trop grande. L'auteur déclare même que la règle proposant de construire une tournée en démarrant d'une ville et en choisissant la ville la plus proche, à chaque itération, ne mène généralement pas à la tournée la plus courte (cf. [55]).

Beaucoup d'approches et d'outils informatiques furent développés depuis cette époque jusqu'à arriver, par exemple, au solveur développé par David Applegate, Robert E. Bixby, Vašek Chvátal et William J. Cook. Ce solveur, nommé *Concorde*[1], a permis de résoudre des instances du problème du voyageur de commerce allant jusqu'à 85.900 villes (cf. [4]).

Nous présentons dans ce chapitre ce problème ainsi que quelques unes de ses extensions qui nous intéressent.

2.2 Problème du voyageur de commerce

Le problème du voyageur de commerce, en Anglais «Traveling Salesman Problem» (TSP), consiste à optimiser la tournée d'*un* véhicule sur un ensemble de *villes* de manière à ce que :

1. Le véhicule démarre et finisse sa tournée à une même ville ;
2. Chacune des villes soit visitée exactement une fois.

1. http ://www.tsp.gatech.edu/concorde.html

2.2.1 Formulation

Soit $G = (V, E)$ un graphe complet. $V = [n] = \{1, \ldots, n\}$ est l'ensemble des *villes*.

Soit la matrice des distances $D = (d_{ij})_{i,j \in [n]} \in R_+$ où d_{ij} est la longueur d'un plus court chemin de i à j. On suppose que $d_{ii} = 0$, $\forall i \in [n]$.

Cas symétrique et cas asymétrique Si $d_{ij} = d_{ji}$ pour tout couple de sommets (i, j) dans $[n]$ alors G est un graphe *non orienté* et le problème est dit *symétrique* (noté STSP) sinon G est un graphe *orienté* et le problème est dit *asymétrique* (noté ATSP).

Inégalité triangulaire Dans le cas où $d_{ij} \leq d_{ik} + d_{kj}$, $\forall i, j, k \in [n]$ alors on dit que le problème satisfait l'*inégalité triangulaire*.

Cas euclidien Dans ce cas les villes sont définies par leurs coordonnées (x_i, y_i) $(i \in [n])$ dans un plan et la distance séparant deux villes est euclidienne, i.e.

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Ce type de problème est symétrique et il satisfait l'inégalité triangulaire.

Le STSP (resp. ATSP) consiste à trouver dans G un *cycle* (resp. *circuit*) de distance *minimum* tel que chaque sommet de V soit visité exactement une fois. Un tel cycle (resp. circuit) est appelé *cycle* (resp. *circuit*) *hamiltonien*.

Complexité Le problème de décision associé au STSP (resp. ATSP) «Pour un entier k donné, existe-t-il une tournée du véhicule, visitant toutes les villes exactement une fois, qui soit inférieure ou égale à k ?» est NP-complet (cf. [34]). Cela vient du fait que le problème de décision "Pour un entier k donné, existe-t-il un cycle (resp. circuit) hamiltonien de longueur inférieur ou égale à k dans G ?" soit NP-complet (cf. [33]).

Nombre de tournées possibles Le nombre de tournées possibles dans le ATSP est équivalent au nombre de façon d'asseoir n convives autour d'une table ronde contenant n places, soit $(n-1)!$. Le nombre de tournées dans le STSP est lui de $(n-1)!/2$ car une tournée dans le STSP est équivalent à deux tournées dans le ATSP. (cf. [29] et [12])

2.2.2 Programme mathématique

D'après Gutin [29] et Laporte [34], le plus ancien (mais toujours utile) modèle mathématique en nombre entier du TSP remonte à celui de Dantzig, Fulkerson et Johnson [12]. Nous rapportons ici leurs formulations.

Cas asymétrique

Soit la variable de décision $x_{ij} = 1$ si l'arc (i, j) figure dans la tournée $(i, j \in [n], i \neq j)$.

Fonction objectif : La distance totale parcourue est à minimiser ;

$$\text{minimiser } \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij}$$

Contrainte des degrés : Chaque ville doit être visitée exactement une fois ;

$$\sum_{i=1}^{n} x_{ij} = 1, \forall j \in [n]$$

$$\sum_{j=1}^{n} x_{ij} = 1, \forall i \in [n]$$

Contrainte d'élimination des sous tours : Quelque soit le sous ensemble S de villes de V (sauf l'ensemble vide et V lui même), aucun cycle ne doit exister dans le sous graphe induit $G[S]$. Sachant que le nombre d'arcs contenus dans un arbre maximal de $G[S]$ est $|S| - 1$, alors on peut écrire la contrainte suivante

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1, \forall S \subsetneq [n], S \neq \emptyset$$

Le problème s'écrit alors :

$$\begin{aligned}
&\text{minimiser} \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \\
&\text{s.c.} \\
&\quad \sum_{i=1}^{n} x_{ij} = 1, && \forall j \in [n] && (2.1)\\
&\quad \sum_{j=1}^{n} x_{ij} = 1, && \forall i \in [n] && (2.2)\\
&\quad \sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1, && \forall S \subsetneq [n], S \neq \emptyset && (2.3)\\
&\quad x_{ij} \in \{0, 1\}, && \forall i, j \in [n], i \neq j && (2.4)
\end{aligned}$$

Les familles de contraintes 2.1 et 2.2 sont les contraintes du problème d'affectation ;
La famille de contraintes 2.3 peut aussi s'écrire :

$$\sum_{i\in S}\sum_{j\in \bar{S}} x_{ij} \geq 1, \forall S \subsetneq [n], S \neq \emptyset$$

car on doit pouvoir ressortir de tout sous ensemble de villes S de $[n]$. Ces contraintes sont appelées aussi *contraintes de connexité*.

Cas symétrique

Soit la variable de décision $x_{ij} = 1$ si l'arête (i, j) figure dans la tournée, $i, j \in [n], i \neq j$. On remarque que $x_{ij} = x_{ji}$ pour toute valeur définie de i et j.

Fonction objectif : La distance totale parcourue est à minimiser. On remarque que la partie inférieure de la matrice des variables de décision suffit à décrire la tournée car on est dans le cas symétrique ;

$$\text{minimiser} \sum_{j=2}^{n} \sum_{i=1}^{j-1} d_{ij} x_{ij}$$

Chaque ville est visitée exactement une fois : Chaque ville i doit avoir exactement deux arêtes qui lui soient incidentes ;

$$\sum_{j=1;\, j\neq i}^{n} x_{ij} = 2, \forall i \in [n]$$

Contrainte d'élimination des sous tours : On impose que le cocycle de tout sous ensemble S de V contienne au moins 2 arêtes. Il suffit de considérer que les ensembles de tailles inférieure à $n/2$ car on est dans le cas symétrique ;

$$\sum_{i\in S} \sum_{j\in \bar{S}} x_{ij} \geq 2, \forall S \subsetneq [n] \text{ t.q. } 2 \leq |S| \leq n/2$$

Le problème s'écrit alors :

$$\begin{array}{lll}
\text{minimiser} & \sum_{j=2}^{n} \sum_{i=1}^{j-1} d_{ij} x_{ij} & \\
\text{s.c.} & & \\
& \sum_{j=1;\, j\neq i}^{n} x_{ij} = 2, & \forall i \in [n] \\
& \sum_{i\in S} \sum_{j\in \bar{S}} x_{ij} \geq 2, & \forall S \subsetneq [n] \text{ t.q. } 2 \leq |S| \leq n/2 \\
& x_{ij} \in \{0,1\}, & \text{pour tout } i \neq j \in [n]
\end{array}$$

Le problème peut s'écrire brièvement de cette manière (cf. [8])

$$\begin{array}{lll}
\text{minimiser} & \sum_{j=2}^{n} \sum_{i=1}^{j-1} d_{ij} x_{ij} & \\
\text{s.c.} & & \\
& \sum_{i\in S} \sum_{j\in \bar{S}} x_{ij} \geq 2, & \forall S \subsetneq [n] \text{ t.q. } 1 \leq |S| \leq n/2 \\
& x_{ij} \in \{0,1\}, & \forall i, j \in [n], j \neq i
\end{array}$$

Ainsi, un sommet peut être considéré comme un sous ensemble S de $[n]$ et on impose que son cocycle contienne deux arêtes.

Il est à noter que la contrainte d'élimination des sous tours telle qu'elle est écrite induit un nombre exponentiel de contraintes résultant du nombre de choix possibles de S (cf. [8]) :

$$\text{nombre de choix de } S = \sum_{i=1}^{n-1} \left(\begin{array}{c} n-1 \\ k \end{array} \right) = 2^{n-1} - 1$$

Miller, Tucker et Zemlin ont proposé cependant une manière de réécrire cette dernière famille de contraintes qui permet d'avoir une écriture succincte du programme linéaire en nombres entiers (cf. [45]).

2.3 Problème de tournées de véhicules

Le problème de tournées de véhicules, en Anglais «Vehicle Routing Problem» (VRP), a été initialement introduit par Dantzig et Ramser dans [13] en 1959 sous le nom de «Truck Dispatching Problem». On peut lire dans le résumé de l'article

> *The paper is concerned with the optimum routing of a fleet of gasoline delivery trucks between a bulk terminal and a large number of service stations supplied by the terminal. The shortest routes between any two points in the system are given and a demand for one or several products is specified for a number of stations within the distribution system. It is desired to find a way to assign stations to trucks in such a manner that station demands are satisfied and total mileage covered by the fleet is a minimum. ...* [1]

Ce résumé décrit une bonne partie de ce qu'est le problème de tournées de véhicules.

On peut lire dans le livre de Toth et Vigo [63] que le VRP consiste à optimiser la visite d'un ensemble de clients, par une *flotte* de véhicules, basés à des dépôts, de manière à ce que :

1. Les véhicules démarrent et finissent leurs tournées à leurs dépôts ;
2. Chacun des clients soit visité exactement une fois ;
3. La demande des clients se trouvant sur une tournée ne dépasse pas la capacité du véhicule qui les livre.

Il existe des variantes du VRP qui tiennent compte de contraintes supplémentaires. On cite parmi elles les cas où :

- Les véhicules sont basés à un même dépôt. Ce problème, auquel il est généralement fait allusion lorsqu'on parle du VRP, est connu sous l'appellation de «Capacited Vehicle Routing Problem» (CVRP) ;
- Le nombre de clients visités lors d'une tournée est limité ;
- La longueur des tournées est limitée ;
- Les clients ne peuvent être visités que pendant une certaine période. On parle, dans ce cas, de problème de tournées de véhicules avec fenêtres de temps, en Anglais «VRP with Time Windows» (VRPTW) ;
- Les véhicules n'ont pas les même capacités ou les mêmes coûts d'utilisation. On parle de VRP avec flotte hétérogène ou mixte, en Anglais «Heterogeneous Fleet VRP» ou «Mixed Fleet VRP» ;
- Les véhicules doivent livrer et éventuellement récupérer des produits chez les clients ;
- Des contraintes de précédence peuvent être considérées :
  - Un client doit appartenir à une tournée où se trouve un sous ensemble d'autres clients et ce client doit être visité avant ou après ces clients ;

1. Ce papier traite du routage optimum d'une flotte de camions de livraison d'essence entre un terminal vrac et un grand nombre de stations services alimentées par le terminal. Le plus court chemin entre deux points du système est supposé connu et la demande d'un ou plusieurs produits est spécifiée pour un ensemble de stations du système. Il s'agit de trouver une affectation des stations aux camions de telle sorte que la demande des stations soit satisfaite et que le mileage [distance en miles] total parcouru par la flotte soit minimum. ...

- Si des clients de types différents sont servis dans une tournée, l'ordre de visite est fixé a priori.
- Le cout de transport, la demande des clients ou un des paramètres du problème est aléatoire. Ceci mène à considérer un VRP stochastique ;
- Un véhicule peut recevoir de nouvelles commandes durant sa tournée. Dans ce cas, l'instance est décrite progressivement durant la résolution de celle-ci. Ce problème est connu sous l'appellation de *Online VRP*.

Il peut y avoir aussi des variantes par rapport à la fonction objectif :

- minimiser le coût ou le temps de transport global ;
- minimiser le nombre de véhicules utilisés ;
- équilibrer la charge de travail entre les véhicules (*balancing workload*) ;
- minimiser les pénalités associées à un service partiel des clients.

ou optimiser une combinaison pondérée de ces critères.

2.3.1 Formulation du Capacited Vehicle Routing Problem

Soit $G = (V, E)$ un graphe complet. $V = \{1, \ldots, n\}$ est l'ensemble des sommets, le sommet 1 représentant le dépôt et les sommets $\{2, \ldots, n\}$ représentant les clients.

Soit la matrice des distances $D = (d_{ij})_{i,j\in[n]} \in R_+$ où d_{ij} est la longueur d'un plus court chemin de i à j. On suppose que $d_{ii} = 0, \forall i \in V$.

Si D est symétrique le problème est dit symétrique et on note SCVRP (généralement on parle de ce cas lorsqu'on traite du CVRP) sinon il est asymétrique et on écrit ACVRP. Les remarques sur l'inégalité triangulaire et la distance euclidienne du paragraphe 2.2.1 sont maintenues pour ce problème.

Notation. *On introduit les notations suivantes :*

d_i : demande positive ou nulle du client i avec $d_1 = 0$.

Soit $S \subseteq V - \{1\}$,

$d(S)$: somme des demandes des clients se trouvant dans S. Elle vaut $\sum_{i\in S} d_i$.

m : nombre de véhicules de capacité identique C se trouvant au dépôt. m peut être fixe ou variable. On suppose que $d_i \leq C$, pour tout $i \in V$.

m_{min} : nombre minimum de véhicules requis pour servir tous les clients.

$r(S)$: nombre minimum de véhicules nécessaires à la satisfaction de la demande en S.

On remarque que $r(V - \{1\}) = m_{min}$. Ce paramètre peut être trouvé en résolvant un problème de remplissage de boites (cf. [63]), en Anglais «Bin Packing Problem» (BPP), où :

- les boites de capacités C sont les véhicules ;
- les n objets de volume d_i sont les clients ;
- on cherche à mettre en boite les objets en utilisant un minimum de boites.

Ainsi, m_{min} est une borne inférieure pour le nombre de véhicules utilisés, d'où $m \geq m_{min}$.

Il est à noter que même si le BPP est NP-difficile, des instances de plusieurs centaines d'objets peuvent être résolues en un temps raisonnable (cf. [43]).

Aussi, on peut simplement donner à $r(S)$ la valeur $\lceil \frac{d(S)}{C} \rceil$ (cf. [63]).

Le CVRP (resp. ACVRP) consiste à trouver dans G, m *cycles* (resp. *circuits*) *élémentaires* de *coût total minimum* tels que :

- Le sommet 1 soit dans tout cycle (resp. circuit) ;
- Tout sommet dans $V - \{1\}$ soit dans exactement un cycle (resp. circuit) ;
- Pour tout cycle (resp. circuit) T_k, $\sum_{v \in T_k} d_v \leq C$ (la demande des clients se trouvant sur une même tournée ne doit pas excéder la capacité du véhicule).

Le CVRP peut se ramener à d'autres problèmes de routages. En effet, si l'on pose C infini, alors on obtient un m-TSP. Si de plus, on pose $m = 1$ alors on obtient un TSP. Comme le TSP est un problème NP-difficile et qu'il est un cas particulier du CVRP alors ce dernier est aussi NP-difficile (cf. [57]).

2.3.2 Programme mathématique

Toth et Vigo présentent dans leur livre (cf. [63]) plusieurs modèles mathématiques en nombres entiers pour le CVRP. Nous rapportons ici le modèle basé sur le problème d'affectation (ou *flow model*).

Cas asymétrique

Soit $x_{ij} = 1$ si l'arc (i, j) figure dans le routage $(i, j \in [n], i \neq j)$.

Fonction objectif : La distance totale parcourue est à minimiser

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

Contrainte des degrés sur le dépôt : m véhicules doivent sortir et entrer au dépôt

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

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

Contrainte des degrés sur les clients : Tout client a exactement un successeur et un prédecesseur dans le routage

$$\sum_{j=1}^{n} x_{ij} = 1, \forall i \in V - \{1\}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \forall j \in V - \{1\}$$

Contrainte d'élimination des sous tours et de respect de la capacité des véhicules : On impose que pour chaque sous ensemble S de clients, il y ait au moins $r(S)$ arcs sortant de S et au moins $r(S)$ entrant dans S. Sachant qu'un arc sortant de S est aussi un arc entrant dans $\bar{S}$, il suffit d'écrire la contrainte comme suit

$$\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geq r(S), \forall S \subseteq V - 1, S \neq \emptyset$$

Le problème s'écrit alors :

$$\begin{cases} \text{minimiser} & \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} & \\ \text{s.c.} & & \\ & \sum_{j=2}^{n} x_{1j} = m & \\ & \sum_{i=2}^{n} x_{i1} = m & \\ & \sum_{j=1}^{n} x_{ij} = 1, & \forall i \in V - \{1\} \\ & \sum_{i=1}^{n} x_{ij} = 1, & \forall j \in V - \{1\} \\ & \sum_{i\in S}\sum_{j\in \bar{S}} x_{ij} \geq r(S), & \forall S \subseteq V-1, S \neq \emptyset \\ & x_{ij} \in \{0,1\}, & \forall i,j \in V, i \neq j \end{cases}$$

Cas symétrique

Soient les variables de décision

$x_{ij} = 1$ si l'arête (i,j) figure dans le routage $(1 < i < j \leq n)$

$$x_{1j} = \begin{cases} 1 & \text{si le dépôt est relié une fois à la ville } j \\ 2 & \text{si un véhicule ne visite que la ville } j \text{ et revient au dépôt} \\ 0 & \text{sinon} \end{cases}$$

avec $j \in \{2, \ldots, n\}$

Fonction objectif : La distance totale parcourue est à minimiser

$$\sum_{j=2}^{n}\sum_{i=1}^{j-1} d_{ij}x_{ij}$$

Contraintes des degrés sur le dépôt : Exactement $2m$ arêtes doivent être incidentes au dépôt dans le routage

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

Contrainte des degrés sur les villes : Exactement deux arêtes doivent être incidentes à chaque ville dans le routage

$$\sum_{k=1}^{i-1} x_{ki} + \sum_{k=i+1}^{n} x_{ik} = 2, \forall i \in V - \{1\}$$

Contrainte d'élimination des sous tours : Pour tout sous ensemble de clients, on impose qu'il y est au moins $2r(S)$ arêtes dans le cocycle de S

$$\sum_{k<i;k\notin S}\sum_{i\in S} x_{ki} + \sum_{i<k;i\in S}\sum_{k\notin S} x_{ik} \geq 2r(S), \forall S \subseteq V-1, 2 \leq |S| \leq n-2$$

Le problème s'écrit alors :

$$\begin{cases} \text{minimiser} & \sum_{j=2}^{n}\sum_{i=1}^{j-1} d_{ij}x_{ij} & \\ \text{s.c.} & & \\ & \sum_{j=2}^{n} x_{1j} = 2m & \\ & \sum_{k=1}^{i-1} x_{ki} + \sum_{k=i+1}^{n} x_{ik} = 2, & \forall i \in V - \{1\} \\ & \sum_{k<i;k\notin S}\sum_{i\in S} x_{ki} + \sum_{i<k;i\in S}\sum_{k\notin S} x_{ik} \geq 2r(S), & \forall S \subseteq V-1, 2 \leq |S| \leq n-2 \\ & x_{ij} \in \{0,1\}, & \forall i,j \in V - \{1\}, i < j \\ & x_{1j} \in \{0,1,2\}, & \forall j \in V - \{1\} \end{cases}$$

2.4 Problème du voyageur de commerce multiple

Le problème du voyageur de commerce multiple, en Anglais «Multiple Traveling Salesman Problem» (m-TSP), consiste à optimiser la tournée d'une flotte de véhicules basés à un dépôt sur un ensemble de villes de manière à ce que :

1. Les véhicules démarrent et finissent leurs tournées au dépôt ;
2. Chacune des villes soit visitée exactement une fois.

Bektas recense dans son article [6] différentes variantes du problème que nous rapportons :

- Les véhicules se trouvent initialement à plusieurs dépôts. Dans ce cas, on distingue deux situations :
  - Les véhicules doivent démarrer et finir leurs tournées à un même dépôt. On parle de *fixed destination case* (cas où la destination est fixe) ;
  - En début de routage, chaque dépôt accueille un nombre connu de véhicules. Les véhicules peuvent finir leurs tournées dans n'importe quel dépôt *mais* le nombre de véhicules retournant à un dépôt en fin de routage doit être le même qu'initialement. On parle de *nonfixed destination case* (cas où la destination n'est pas fixe) ;
- Le nombre des véhicules m peut être fixe ou variable et borné ;
- Lorsque le nombre de véhicules m n'est pas fixe, un coût peut être associé à l'utilisation d'un véhicule. Dans ce cas, le critère qui minimise le nombre de véhicules utilisés peut être considéré ;
- Les villes ne peuvent être visitées que pendant des périodes spécifiques. On parle de m-TSP *with Time Windows* (m-TSP avec fenêtres de temps).

2.4.1 Formulation

Nous reprenons dans cette partie la formulation donnée dans le paragraphe 2.3.1 en considérant une capacité infinie des véhicules, ce qui amène à négliger les demandes d_i.

Soit $G = (V, E)$ un graphe complet. $V = \{1, \ldots, n\}$ est l'ensemble des sommets, le sommet 1 représentant le dépôt et les sommets $\{2, \ldots, n\}$ représentant les villes à visiter.

Soit la matrice des distances $D = (d_{ij})_{i,j\in[n]} \in R_+$ où d_{ij} est la longueur d'un plus court chemin de i à j. On suppose que $d_{ii} = 0, \forall i \in V$.

Si D est symétrique le problème est dit *symétrique* sinon il est *asymétrique*. Les remarques sur l'inégalité triangulaire et la distance euclidienne du paragraphe 2.2.1 sont maintenues pour ce problème.

Soit m le nombre de véhicules se trouvant au dépôt. Il peut être fixe ou variable.

Le m-TSP symétrique (resp. asymétrique) consiste à trouver dans G, m cycles (resp. circuits) élémentaires de coût total minimum tels que :

- Le sommet 1 soit dans tout cycle (resp. circuit) ;
- Tout sommet dans $V - \{1\}$ soit dans un exactement un cycle (resp. circuit) ;

Complexité Ce problème est une généralisation du TSP où l'on considère plus d'un véhicule. Comme le TSP est NP-difficile, il en est de même pour le m-TSP. (cf. [69]).

Nombre de routages possibles Nous pouvons voir le m-TSP asymétrique sous l'optique de la combinatoire énumérative. Pour cela, nous introduisons d'abord un outil de calcul combinatoire appelé les nombres de Lah non signés.

Définition. *(k-partition totalement ordonnée) Etant donné l'ensemble $[n] = \{1, \ldots, n\}$. Une k-partition totalement ordonnée de $[n]$ est une partition de $[n]$ en k sous ensembles E_i, $i = \overline{1,k}$, telle que :*

- *L'ordre entre les sous ensembles E_i, $i = \overline{1,k}$, n'est pas important ;*
- *L'ordre à l'intérieur d'un sous ensemble E_i, $i = \overline{1,k}$, est important.*

Définition. *(Nombres de Lah non signés) Le nombre de Lah non signé $L(n,k)$, introduit par Ivo Lah en 1955 (cf. [52]), compte le nombre de k-partitions totalement ordonnées que l'on peut construire à partir de l'ensemble $[n]$. Autrement dit, $L(n,k)$ compte le nombre de manières de partitionner $[n]$ en k* files *non vides telles que l'ordre entre les files ne soit pas important.*

Le calcul du nombre des solutions réalisables dans un m-TSP asymétrique peut se faire par les nombres de Lah non signés. En effet, si l'on note $N(n,m)$ le nombre de solutions réalisables pour un m-TSP asymétrique, avec $n-1$ villes à visiter et m véhicules et que l'on considère que le dépôt est le sommet n, alors

$$N(n,m) = L(n-1,m)$$

En effet, une solution du m-TSP asymétrique est une partition des $n-1$ villes à visiter en m sous ensembles telle que :

- l'ordre entre les m sous ensembles n'est pas important ;
- l'ordre à l'intérieur d'un sous ensemble est important.

Autrement dit, une solution est une m-partition totalement ordonnée de $[n-1]$. Par conséquent, le nombre des solutions réalisables pour le m-TSP asymétrique est $L(n-1,m)$.

2.4.2 Programme mathématique

Bektas présente dans son article [6] plusieurs modèles mathématiques pour le m-TSP. Nous rapportons ici un modèle pour le cas asymétrique et un autre pour le cas symétrique.

Cas asymétrique

Soit la variable de décision $x_{ij} = 1$ si l'arc (i,j) figure dans le routage, $i \neq j \in V$.

Fonction objectif : La distance totale parcourue est à minimiser

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

Contraintes des degrés sur le dépôt : m véhicules doivent sortir et entrer au dépôt

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

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

Contrainte des degrés sur les villes : Toute ville a exactement un successeur et un prédécesseur dans le routage

$$\sum_{j=1}^{n} x_{ij} = 1, \forall i \in V - \{1\}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \forall j \in V - \{1\}$$

Contrainte d'élimination des sous tours : Pour tout sous ensemble de villes $S \subseteq V - \{1\}$, on interdit qu'un cycle se forme à l'intérieur de $G[S]$ ($|S| - 1$ est le nombre d'arcs dans un arbre maximal de $G[S]$)

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \forall S \subseteq V - 1, S \neq \emptyset$$

Le problème s'écrit alors :

$$\begin{cases} \text{minimiser} & \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} & \\ \text{s.c.} & & \\ & \sum_{j=1}^{n} x_{1j} = m & \\ & \sum_{i=1}^{n} x_{i1} = m & \\ & \sum_{j=1}^{n} x_{ij} = 1, & \forall i \in V - \{1\} \\ & \sum_{i=1}^{n} x_{ij} = 1, & \forall j \in V - \{1\} \\ & \sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, & \forall S \subseteq V - \{1\}, S \neq \emptyset \\ & x_{ij} \in \{0, 1\}, & \forall i \neq j \in V \end{cases}$$

Cas symétrique

Soit la variable de décision $x_{ij} = 1$ si l'arête (i, j) figure dans le routage $(1 < i < j \leq n)$,

et soit la variable de décision $x_{1j} = \begin{cases} 1 & \text{si l'arête } (1, j) \text{ figure dans le routage} \\ 2 & \text{si deux arêtes relie le dépôt à la ville } j \quad j = 2, \ldots, n \\ 0 & \text{sinon} \end{cases}$

Fonction objectif : La distance totale parcourue est à minimiser

$$\sum_{j=2}^{n} \sum_{i=1}^{j-1} d_{ij} x_{ij}$$

Contraintes des degrés sur le dépôt : Exactement $2m$ arêtes doivent être incidentes au dépôt dans le routage

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

Contrainte des degrés sur les villes : Exactement deux arêtes doivent être incidentes à chaque ville dans le routage

$$\sum_{k=1}^{i-1} x_{ki} + \sum_{k=i+1}^{n} x_{ik} = 2, \forall i \in V - \{1\}$$

Contrainte d'élimination des sous tours : Pour tout sous ensemble de villes $S \subseteq V - \{1\}$, on interdit qu'un cycle se forme à l'intérieur de $G[S]$ ($|S| - 1$ est le nombre d'arêtes dans un arbre maximal de $G[S]$). Notons que par définition ni une boucle ni un cycle de longueur 2 ne peuvent se former car ni x_{ii} ni x_{ji} ($j > i$) ne sont définis

$$\sum_{i \in S} \sum_{j \in S; i<j} x_{ij} \leq |S| - 1, \forall S \subseteq V - 1, |S| \geq 3$$

Le problème s'écrit alors :

$$\begin{cases} \text{minimiser} & \sum_{j=2}^{n} \sum_{i=1}^{j-1} d_{ij} x_{ij} & \\ \text{s.c.} & & \\ & \sum_{j=2}^{n} x_{1j} = 2m & \\ & \sum_{k=1}^{i-1} x_{ki} + \sum_{k=i+1}^{n} x_{ik} = 2, & \forall i \in V - \{1\} \\ & \sum_{i \in S} \sum_{j \in S; i<j} x_{ij} \leq |S| - 1, & \forall S \subseteq V - 1, |S| \geq 3 \\ & x_{ij} \in \{0, 1\}, & \forall 1 < i < j \leq n \\ & x_{1j} \in \{0, 1, 2\}, & \forall j \in [n] - \{1\} \end{cases}$$

2.4.3 Méthodes de résolution

Le m-TSP a été moins étudié dans la littérature que le TSP ou le VRP. Bektas donne une vue d'ensemble sur le m-TSP et reprend l'évolution des méthodes exactes et approchées proposées pour ce problème (cf. [6]). Nous donnons ici quelques résultats qu'il cite dans son article ainsi que d'autres contributions parues depuis.

Évolution des méthodes exactes

1980 Laporte et Nobert considèrent dans [37] un m-TSP où un coût est associé à chaque véhicule utilisé. Ils s'inspirent de deux méthodes de coupes développées pour le TSP par Miliotis, basées sur la relaxation des contraintes d'intégrité et d'élimination des sous tours [44]. Les auteurs comparent la résolution directe d'un m-TSP en adaptant l'idée de Miliotis avec la résolution, par l'approche de cet auteur, d'un 1-TSP obtenu en transformant le m-TSP initial. Pour $m = 2, 3, \ldots, 10$, Laporte et Nobert remarquent que l'approche directe se revéle meilleure, en temps d'execution, que la transformation vers le 1-TSP lorsque m augmente. Les auteurs remarquent également que le temps d'exécution est proportionnel à m lorsqu'on considère la transformation vers le 1-TSP tandis qu'elle devient inversemment proportionnel à m lorsque le problème est abordé directement. Des instances du m-TSP symétrique non euclidiennes de 100 villes sont résolues avec un temps moyen de 3 minutes et 28 secondes sur un CYBER 173.

1986 Ali et Kennington proposent dans [2] une méthode de Séparation Évaluation Progressive (SEP) qui a été testée sur des instances asymétriques de taille $n = 100$ et symétriques et euclidiennes de taille $n = 59$;

1986 Gavish et Srikanth proposent dans [27] une méthode de SEP pour les m-TSP de plus grande échelle. Des problèmes symétriques non euclidiens de taille $n = 500$ avec $m = 2, 4, 6, 8, 10$ ont été résolus avec un temps moyen de 24 minutes et 33

secondes sur un IBM 3081D tandis que des problèmes symétriques euclidiens de taille $n = 100$ avec les mêmes nombres de véhicules ont été résolus en un temps moyen de 3 heures et 52 minutes sur la même machine. Les auteurs constatent que, dans le cas euclidien, leur méthode concurrence celle de Laporte et Nobert et qu'elle est plus rapide que celle d'Ali et Kennington. Dans le cas non euclidien, la méthode de Gavish et Srikanth se révèle supérieure aux deux autres.

1992 Gromicho et al. proposent une méthode de SEP. La plus grande instance résolue par cette approche est de taille $n = 120$ avec m variant de 2 à 12 (résultat communiqué en 2003) (cf. [6]).

En 2006, les plus grandes instances du m-TSP résolues sont d'environ 500 villes pour le problème asymétrique et 100 villes pour le problème symétrique (cf. [6]).

Évolution des méthodes d'optimisation approchée

Heuristiques

1977 Russel propose dans [53] une heuristique basée sur la transformation du m-TSP en TSP en tenant compte de contraintes additives (précédence, distance et temps, etc) . L'heuristique qu'il propose est une extension de celle de Lin et Kernighan (cf. [38]) qui est initialement confectionnée pour le TSP. Il déclare obtenir toujours de meilleurs résultats que les heuristiques présentes à ce moment dans la littérature ;

1989 Potvin et al. proposent une heuristique basée sur une *procédure d'échange* (cf. [6]) ;

2008 Yadlapalli et al. considèrent dans [66] une variante du m-TSP avec plusieurs dépôts où l'on doit choisir parmi plusieurs véhicules ceux qui effectueront une tournée et donner un routage des véhicules choisis. Leur expérimentation indiquerait que l'algorithme donne des solutions à 5 % de la solution optimale pour des instances d'environ 45 villes et 6 véhicules ;

2010 Tian-Long et Yong-Zai ont proposé un algorithme pour une variante du m-TSP où chaque véhicule ne peut visiter qu'un nombre limité de villes. Ils déclarent obtenir de bonnes performances sur une série d'instances (cf. [62]) ;

2010 Xu et Rodrigues proposent dans [65] un algorithme polynomiale d'approximation à 3/2 pour le m-TSP à plusieurs dépôts (nombre de dépôts constant).

Méta-heuristiques

1990 Fogel propose une méthode de programmation parallèle basée sur la programmation évolutionnaire. La fonction objectif considérée est la minimisation de la différence entre les longueurs des tournées (équilibrage de la charge de travail). Cette méthode a permis la résolution de deux instances de $n = 25$ et $n = 50$ pour $m = 2$ (cf. [6]) ;

1999 Zhang et al. proposent une modélisation, par le m-TSP, d'un problème de planification de la tournées de photographes sur plusieurs écoles primaires et

secondaires (pour des photos de classes). Un algorithme génétique est appliqué pour la recherche d'une solution (cf. [68]) ;

2000 Tang et al. modélisent un problème d'ordonnancement dans l'industrie métallurgique chinoise par le m-TSP. Un algorithme génétique est utilisé (cf. [61]) ;

2002 Yu et al. utilisent un algorithme génétique dans le cas de la planification du travail d'un groupe de robots devant accomplir plusieurs tâches. Les objectifs considérés sont de minimiser la longueur du routage globale et minimiser la longueur de la plus grande tournée (cf. [67]) ;

1998 Ryan et al. adaptent le paradigme de la recherche Tabou au m-TSP avec fenêtres de temps (cf. [54]) ;

2003 Song et al adaptent le paradigme du recuit simulé au m-TSP et testent leur méthode sur une instance de $n = 400$ et $m = 3$. La résolution de ce problème sur un IBM PC 586 (400 Mhz) prend 51 minutes (cf. [59]) ;

2009 Liu et al. ont proposé une approche au problème par les colonies de fourmis en considérant deux objectifs (similaires à ceux considérés par Yu et al.) : minimiser la distance parcourue par tous les véhicules et minimiser la distance parcourue par chacun des véhicules. Ils comparent un algorithme génétique avec leur méthode sur un ensemble d'instances et déclarent que cette méthode est compétitive pour les deux objectifs (cf. [39]) ;

2011 Ghafurian et Javadian ont proposé une approche par les colonies de fourmis pour le cas impliquant plusieurs dépôts. Les auteurs ont comparé leurs résultats avec ceux du solveur Lingo 8.0 et ont observé qu'ils étaient proches des optimums en un temps relativement faible (cf. [28]).

Apprentissage non supervisé et supervisé

1989 Wacholder et al. proposent un algorithme basé sur les réseaux de neurones pour le problème en le transformant en un TSP. Ils obtiennent des solutions valides (selon les termes des auteurs) pour des instances allant jusqu'à 30 villes et 5 véhicules (cf. [64]) ;

1991 Hsu et al. proposent également une approche au m-TSP par les réseaux de neurones en tentant la résolutions de m TSP. Les auteurs reviennent sur un exemple de Wacholder et al. (cf. [31]) ;

1999 Modares et al. développent une approche par les réseaux de neurones «auto-organisant» (self-organizing neural network) et l'appliquent au m-TSP et au VRP. Le test de l'algorithme indique des avancées significatives dans la plupart des cas (cf. [6],[46]) ;

1999 Somhom et al. proposent une approche par les réseaux de neurones pour le *minmax* m-TSP qui consiste à minimiser la distance parcourue dans la plus longue des tournées. Les auteurs déclarent qu'en général, leur solutions sont proches de celles qui auraient été obtenues par une recherche tabou et cela avec un temps d'exécution inférieur (cf. [58]).

2002 Gomes et Zuben appliquent au problème une méthode d'optimisation multi-critère basée sur l'apprentissage non supervisé (cf. [14]) ;

2009 Nallusamy et al. proposent une heuristique en trois phases qui partitionne d'abord les villes par la méthode des k-means, génère des tournées pour les véhicules puis améliore la solution générale par une méta-heuristique (cf. [48]).

2.5 Conclusion

Nous avons pu voir dans ce chapitre plusieurs problèmes de routage. Nous avons particulièrement insisté sur le problème du voyageur de commerce multiple auquel nous nous intéressons dans notre étude. Nous aimerions développer une heuristique rapide et efficace pour ce problème ou une variante de celui-ci.

Bektas déclare dans son article [6] :

> *... we believe that other modern heuristic methodologies for the mTSP deserve much more attention. To the best of our knowledge, no efficient heuristic algorithms exist for the solution of large-scale mTSPs.* [1]

1. ...Nous pensons que d'autres méthodes heuristiques modernes [conçues] pour le m-TSP méritent bien plus d'attention. A notre connaissance, il n'existe pas d'heuristiques efficaces pour les m-TSP de grande échelle.

Chapitre 3

Outils de partitionnement

3.1 Introduction

Nous avons vu dans le chapitre précédent le problème du voyageur de commerce multiple. Nous pensons que pour obtenir une bonne solution à ce problème, il est crucial dans notre approche, d'avoir une partition intéressante des villes à visiter. Dans ce chapitre, nous présentons différents outils de clustering et nous insistons sur l'un d'eux qui est basé sur la b-coloration (cf. paragraphe 3.4 page 28). Un tel outil nous servira pour une application au problème du voyageur de commerce multiple.

3.2 Coloration de graphes

3.2.1 Coloration de graphes et nombre chromatique

On appelle *coloration propre* du graphe $G = (V, E)$ l'application allant de V dans $\{1...k\}$ qui associe à chaque sommet de V une couleur entre 1 et k telle que deux sommets adjacents n'aient pas la même couleur.

Le *nombre chromatique* de G, noté $\chi(G)$, est le plus petit entier pour lequel G admet une coloration propre.

Le problème de décision associé au nombre chromatique peut être formulé comme suit «Etant donné un entier k, existe t-il une coloration propre de G utilisant un nombre de couleurs inférieur ou égal à k ?». Il a été montré que ce problème est NP-complet (cf. [33]).

On trouve des applications de la coloration de graphe dans divers domaines tels que l'ordonnancement, la confection d'horaires et l'allocation des fréquences (cf. [41]).

3.2.2 b-coloration de graphes et nombre b-chromatique

Irving et Manlove ont introduit en 1999 dans [32] un nouveau type de coloration appelé la *b-coloration de graphes*.

On appelle b-coloration du graphe $G = (V, E)$ toute coloration propre de G telle que pour chaque couleur, il existe au moins un sommet adjacent à toutes les autres couleurs. Ces sommets sont appelés *sommets dominants* ou *b-sommets*. Une couleur est

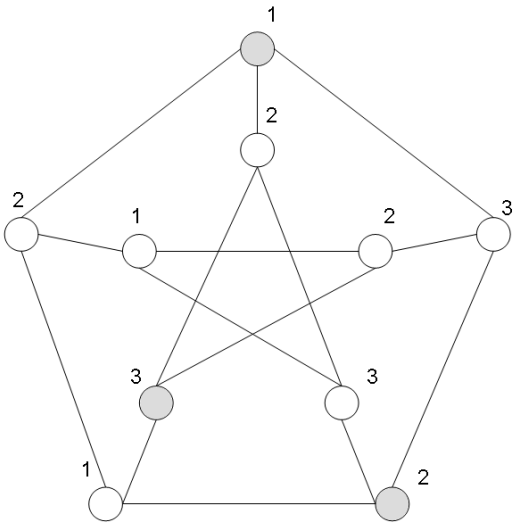

FIGURE 3.1 – Graphe de Petersen (cf. [30]) : Coloration propre utilisant un minimum de couleurs

dite *dominante* si elle admet un b-sommet. La coloration dans la figure 3.1 est aussi une b-coloration où les sommets dominants sont en gris.

Le *nombre b-chromatique* de G, noté $\varphi(G)$, est le plus grand entier pour lequel G admet une b-coloration.

Irving et Manlove ont montré que déterminer le nombre b-chromatique pour un graphe quelconque est NP-complet (cf. [32]). Néanmoins, les mêmes auteurs ont montré que ce problème devient polynomial pour les arbres et ont prouvé que $\varphi(G)$ admet les bornes suivantes : $\chi(G) \leq \varphi(G) \leq \Delta_G + 1$.

Si une b-coloration avec k couleurs existe pour un graphe, cela n'implique pas qu'il existe une b-coloration avec $k+1$ couleurs (cf. [32]).

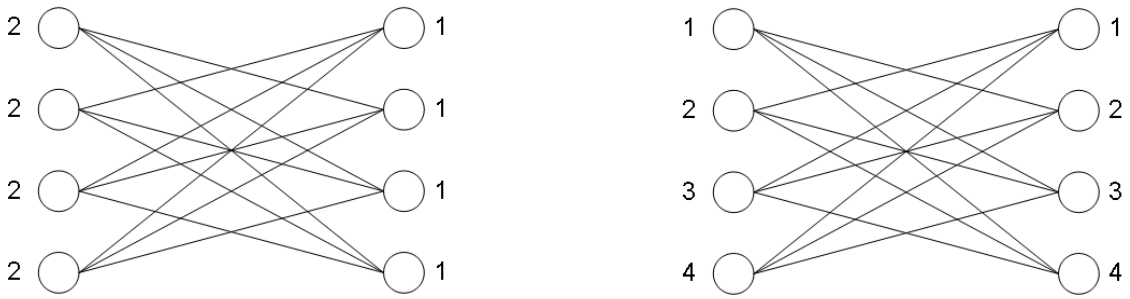

FIGURE 3.2 – b-coloration d'un $K'_{4,4}$ (biparti complet moins couplage parfait) par 2 et 4 couleurs

b-spectre et b-continuité

Le *b-spectre* d'un graphe G est l'ensemble des entiers pour lequel G admet une b-coloration.

Un graphe est dit *b-continu* si et seulement si son b-spectre est composé uniquement d'entiers consécutifs.

3.3 Comment regrouper des objets ?

Nous sommes souvent amenés à réunir des objets en groupes pour mettre de l'ordre et voir plus clair. Ce problème est richement étudié dans la littérature. On distingue deux manière de regrouper des objets : la *classification* et le *clustering*. La différence entre ces deux domaines réside dans le fait de savoir si l'on connait *a priori* les groupes auxquels seront affectés les objets ou non. Nous donnons ici un bref descriptif de ces deux domaines en nous basant sur la thèse de Belacel [7].

3.3.1 Classification

La classification consiste à affecter des objets à des groupes appelés *classes* qui sont définies *a priori*. Ces classes sont généralement identifiées par des *règles d'affectation* ou bien par des *objets de référence*. Ces derniers servent à connaître la classe à laquelle s'apparente le plus un objet à classer, en opérant une comparaison de celui-ci avec les objets de référence.

Le problème se résume alors en la détermination des objets de référence ou des règles d'affectation puis en l'exploitation de ceux-là. L'inférence de ces informations se fait par la *fouille* d'une *base d'apprentissage* qui contient un ensemble d'objets déjà classés. Il s'agit alors de *déduire* ou d'*induire* les règles à partir de cette base. On appelle cette étape *phase d'apprentissage*. Enfin, l'affectation d'un objet à une classe se fait au moyen des règles d'affectation ou des objets de références qui ont été déterminés durant la phase d'apprentissage.

Plusieurs classificateurs (méthodes de classification) ont été développés, on peut citer :

La méthode des k - plus proches voisins Elle fut introduite en 1951 par Fix et Hodges (cf. [24]). Etant donné un entier k fixé par l'utilisateur, une base d'apprentissage T contenant des objets déjà classés et x un individu à classer, la méthode cherche dans T, k individus qui soient les plus *proches* de x. On affecte alors x à la classe qui contient le plus d'individus parmi les k choisis dans la base. On peut faire varier k pour obtenir de meilleurs résultats ;

Méthodes d'analyse discriminante Ce type de méthode consiste à produire des règles d'affectation sous forme de fonctions discriminantes appelées également *fonctions de décision*. La nature de ces fonctions dépend de la forme des classes, elles peuvent être linéaires, quadratiques ou autre. Dans le cas linéaire, si l'on considère que les objets sont caractérisés par p attributs, on cherche à déterminer pour chaque classe C_i une fonction de la forme

$$d_i(x) = w_1 x^1 + w_2 x^2 + \ldots + w_p x^p$$

où les x^j, $j = \overline{1,p}$, sont les valeurs de l'objet x sur les p attributs et w_j, $j = \overline{1,p}$ les paramètres caractérisant la classe C_i pour chaque attribut. Après la détermination de ces paramètres par une phase d'apprentissage, l'affectation d'un individu a à une classe C_i se fait selon la règle suivante : Si $d_i(a) > 0$ alors l'individu a est affecté à la classe C^i.

Le plus ancien article que nous trouvons et qui traite de ce type de méthode est un travail de Fisher paru en 1936 où l'auteur s'intéresse à l'application de l'analyse discriminante à un problème de taxinomie (cf. [23]).

3.3.2 Clustering

Le clustering consiste à regrouper des objets en groupes, appelés *clusters*, qui ne sont pas connus *a priori*. Ceci doit se faire de manière à ce que :

- La densité *intra* cluster soit la plus petite possible. Autrement dit, les objets au sein d'un même cluster doivent être le plus similaires possibles ;
- L'écart *inter* clusters soit le plus grand possible. Autrement dit, les objets au sein de clusters différents doivent être le plus différents possibles.

Les méthodes de clustering doivent passer par une étape de validation afin d'évaluer la qualité de la partition qu'elles génèrent. Pour cela, on fait appel à des indices de validité qui reflètent chacun l'aspect de la partition par rapport à un ensemble de critères.

Plusieurs méthodes de clustering ont été développées, on peut citer :

La méthode des leaders Cette méthode considère chaque objet une seule fois. Lorsqu'un premier objet arrive il devient automatiquement leader et représentant d'un cluster, la méthode consiste ensuite à affecter chaque nouvel objet de la manière suivante : On compare l'objet à affecter aux leaders existants. Si la distance à l'un des leaders est en dessous d'un seuil fixé, l'objet est affecté au cluster de ce leader. Dans le cas contraire, l'objet à affecter devient leader d'un nouveau cluster.

La méthode des k-means Le terme «k-means» fut utilisé pour la première par MacQueen dans [40] bien qu'on remonte l'idée de la méthode à Steinhaus [60]. Etant donné n objets $x_1, x_2, \ldots, x_n$ décrits par p attributs, on cherche à partitionner les objets en k clusters $P = \{C_1, C_2, \ldots, C_k\}$ de manière à ce que la somme des carrés à l'intérieur de chaque cluster soit minimum [1] i.e.

$$arg\min_{P} \sum_{j=1}^{k} \sum_{x_i \in C_j} ||x_i - \mu_j||^2$$

où μ_j est le centre de gravité du cluster C_j et $||x_i - \mu_j||^2 = \sum_{l=1}^{p} (x_i^l - \mu_j^l)^2$.

L'algorithme se déroule comme suit :

1. On prend k centres arbitraires initialement $m_1^{(1)}, m_2^{(1)}, \ldots, m_k^{(1)}$
2. Répéter jusqu'à convergence :
   (a) Les n objets sont affectés aux centres auxquels ils sont les plus proches i.e.

$$C_j^{(t)} = \{x_i / ||x_i - m_j^{(t)}||^2 \leq ||x_i - m_{j^*}^{(t)}||^2, j^* = 1, \bar{} k\}$$

   (b) Les centres de gravité des clusters ainsi formés sont mis à jour i.e.

$$m_j^{(t+1)} = \frac{1}{|S_j^{(t)}|} \sum_{x_i \in S_j^{(t)}} x_i$$

1. Déscription provenant de `http://fr.wikipedia.org/wiki/K-means`

Ces méthodes présentent l'avantage d'être rapides mais elles présentent aussi quelques inconvénients. La performance de la méthode des leaders reposent sur l'ordre de présentation des objets, ceci peut éloigner la procédure de la classification adéquate. La méthode des k-means doit tester différentes valeurs de k pour atteindre une partition optimale ce qui augmente le temps de calcul.

Toutes les méthodes citées supposent qu'on connait une métrique pour décrire ou comparer les individus. Ce genre d'approche n'est pas toujours autorisée car un individu peut être décrit par plusieurs *attributs* qui ne peuvent être agrégés en une seule fonction. On peut recourir alors à des classificateurs multicritères basés sur le paradigme de l'analyse multicritère tels que PROMETHEE TRI, ELECTRE TRI ou PROAFTN, ou à des méthodes de clustering multicritère. On pourra consulter les thèses de Belacel [7] et Nemery de Belleveaux [49] pour une vue d'ensemble sur la classification et le clustering multicritère.

Evaluation d'un clustering

L'évaluation d'un clustering consiste à évaluer la qualité d'une partition générée par une méthode de clustering par rapport à un indice. Différents indices existent dans la littérature, nous citons les suivants :

Indice de précision Cet indice est utilisé pour évaluer la qualité des systèmes de catégorisation de documents. Si l'on suppose que nous connaissons a priori les catégories auxquelles appartiennent un ensemble de documents, et que l'on opère un clustering par une méthode sur cet ensemble alors l'indice de précision est calculé comme suit :

$$\text{Précision sur la classe } C_i = \frac{\text{Nombre de documents correctement attribués à la classe } C_i}{\text{Nombre de documents attribués à la classe } C_i}$$

Indice de Dunn généralisé Etant donnée une distance d qui mesure la dissemblance entre deux objets alors on définit les indices suivants :

- $s_a(C_i)$: L'indice de mesure de la distance moyenne dans un cluster C_i ;

$$s_a(C_i) = \frac{1}{n_i(n_i-1)} \sum_{p=1}^{n_i} \sum_{q=1}^{n_i} d(v_p, v_q)$$

- $d_a(C_i, C_j)$: L'indice de mesure de la séparation entre deux clusters C_i et C_j ;

$$d_a(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p=1}^{n_i} \sum_{q=1}^{n_j} d(v_p, v_q)$$

L'indice de Dunn généralisé se définit comme suit :

$$Dunn_G = \frac{\min_{i,j,i\neq j} d_a(C_i, C_j)}{\max_h s_a(C_h)}$$

Cet indice peut se lire comme le rapport, pour une partition donnée, de la plus petite distance moyenne séparant deux clusters sur la plus grande distance moyenne au sein d'un cluster. L'objectif est alors de maximiser ce rapport pour obtenir une partition de clusters denses et bien séparés.

3.4 Méthode de clustering basée sur la b-coloration

En 2006, Elghazel et al. proposent une méthode de clustering basée sur la b-coloration (cf. [18]). La complexité de l'algorithme proposé est traitée et une application au domaine médicale est donnée. La méthode présente de meilleurs résultats en comparaison avec d'autres méthodes.

En 2007, la méthode est testée et comparée avec d'autres méthodes de clustering sur des instances connues du domaine (cf. [20]). Un algorithme de recoloration glouton est proposé pour améliorer la qualité du clustering (cf. [21]). La méthode connait également une extension au cas du «*dynamic clustering*» où l'on doit garder les groupes homogènes et bien séparés lorsque de nouveaux objets doivent être affectés (cf. [19]). Par ailleurs et durant la même année, une nouvelle extension tenant compte des contraintes «*must-link*» et «*cannot-link*» est proposée. Ces contraintes forcent deux objets à se trouver dans un même cluster ou au contraire à être dans des clusters différents (cf. [17]).

En 2008, une méthode hybride, basée sur le clustering par la b-coloration et les probabilités, est proposée avec une nouvelle application au domaine médical (cf. [22]).

En 2009, Djamel Gaceb et al. proposent une application des algorithmes de MCE au tri automatique des documents (cf. [25]). Une application est proposée aussi dans le domaine des Self Organizing Maps par Elghazel et al. (cf. [16]).

En 2010, Ogino et Yoshida argumentent que l'algorithme de recoloration d'Elghazel et al. (cf. [21]) est restrictif en terme d'espace de recherche car glouton et séquentiel (cf. [50]). Ils proposent en conséquence un algorithme de recoloration non glouton. Les expérimentations montrent que l'algorithme est supérieur lorsqu'on considère le critère de précision qui teste l'affectation des objets à leurs classes réelles. Cependant, l'algorithme de recoloration d'Elghazel et al. reste meilleure dans la majorité des cas lorsqu'on se base sur l'indice $Dunn_G$. Il est à noter que le critère de précision ne peut être utilisé directement en clustering car cela suppose que l'on connaisse les classes des objets ce qui n'est pas vrai (cf. paragraphe 3.3.2 page 26).

3.4.1 Méthode de Clustering d'Elghazel et al. (MCE)

Considérons $G = (V, E, d)$, $|V| = n$, un graphe simple complet non orienté où :
- V : ensemble des objets
- $d(x, y)$: distance entre l'objet x et l'objet y (distance de dissemblance)

L'objectif de la procédure est d'obtenir une partition des objets en classes homogènes et bien séparées sans connaître le nombre de classes au préalable.

Esquisse de la méthode

Pour toute distance S_d présente dans la matrice des distances **faire**

1. Construction du graphe partiel, appelé *graphe seuil-supérieur*, $G_{>S_d} = (V, E\prime, d)$ avec $e \in E\prime$ ssi $d(e) > S_d$;
2. Initialiser la coloration de $G_{>S_d}$ sans tenir compte des distances en utilisant $\Delta_{G_{>S_d}}+1$ couleurs ;

3. Chercher une b-coloration de $G_{>S_d}$ de manière à ce que les sommets ayant la même couleur soient le moins distant possible ;
4. Évaluation de la partition trouvée.

fin pour.
La meilleure partition trouvée est retenue en sortie.

Interprétation

La b-coloration garantie que pour chaque couleur, il existe un sommet b-sommet qui contient dans son voisinage toutes les autres couleurs existantes, ainsi :

- Les objets ayant une même couleur sont similaires (homogénéité) ;
- Pour chaque couleur, il existe un objet (b-sommet) différent des objets se trouvant dans les autres groupes. (hétérogénéité).

Algorithme

Notation.

- Δ *: Degré maximum du graphe* $G_{>S_d}$ *;*
- $c(v_i)$ *: Couleur (valeur entière) du sommet* v_i *;*
- $N(v_i)$ *: Voisinage de* v_i *dans* $G_{>S_d}$ *;*
- $N_c(v_i)$ *: Ensemble des couleurs se trouvant dans le voisinage de* v_i *;*
- L *: Ensemble des couleurs se trouvant dans* $G_{>S_d}$ *;*
- D_m *: Ensemble des couleurs ayant un sommet dominant ;*
- ND_m *: Ensemble des couleurs n'ayant pas de sommet dominant ;*
- $dist(v_i, c)$ *: Distance entre le sommet* v_i *et la couleur* c *qui est définie comme étant la distance entre* v_i *et le plus proche sommet ayant la couleur* c

$$dist(v_i, c) = min\{d_{v_i,v_j} | v_i, v_j \in V, v_i \neq v_j \text{ et } c(v_j) = c\}$$

- n_i *: Taille du cluster* C_i *;*
- T *: Ensemble des sommets colorés triés dans le sens décroissant de leurs degrés.*

Début de la méthode

Pour toute distance S_d présente dans la matrice des distances D **faire**

Étape 1 Construire le graphe seuil-supérieur $G_{>S_d}$ et initialiser sa coloration par l'algorithme 1 (page 30).

L'objectif de l'algorithme est d'obtenir une coloration propre de $G_{>S_d}$ utilisant exactement $\Delta+1$ couleurs. La coloration utilise ainsi le maximum de couleurs que puisse utiliser une b-coloration (cf. paragraphe 3.2.2 page 23).

L'algorithme commence par attribuer au sommet v, de degré maximum, la couleur 1 et ajoute v à T. L'algorithme colore le reste des sommets en respectant le principe suivant :

Pour tout sommet non coloré v_j, voisin du sommet v_i de degré maximum choisi en T (v initialement) :

- on affecte une couleur qui n'est ni dans le voisinage de v_j ni dans celui de v_i ; ceci permet d'utiliser un maximum de couleurs ;
- si une telle affectation n'est pas possible (les Δ+1 couleurs sont dans $N_c(v_i) \cup N_c(v_j)$), on affecte à v_j une couleur qui n'est pas dans son voisinage.

Ce principe permet d'obtenir une coloration propre utilisant $\Delta + 1$ couleurs. En effet, dans les deux cas, la couleur choisie pour v_j n'est pas présente dans son voisinage alors la coloration est propre. Aussi, dans le cas où les $\Delta + 1$ couleurs sont dans $N_c(v_i) \cup N_c(v_j)$, on peut toujours trouver une couleur en dehors de $N_c(v_j)$ qui soit dans $\{1, \ldots, \Delta + 1\}$. Le contraire voudrait dire que v_j est un sommet de degré supérieur à Δ, ce qui est absurde.

Algorithm 1 Init-b-coloring()

Requiert: $G_{>S_d} = (V, E\prime, d)$; {L'algorithme reçoit en entrée un graphe simple non orienté}

1: Choisir un sommet v avec un degré maximum ;
2: $c(v) := 1$;
3: Ajouter v à T ;
4: $L := \{1, 2, \ldots, \Delta + 1\}$;
5: **répéter**
6: Choisir v_i dans T ;
7: $M := N_c(v_i) \cup c(v_i)$;
8: **pour tout** $v_j \in N(v_i)$ tel que $c(v_j) = 0$ **faire**
9: $q := min\{h | h > q, h \notin M$ et $h \notin N_c(v_j)\}$;
10: **si** $q <= \Delta + 1$ **alors**
11: $c(v_j) := q$;
12: **sinon**
13: $c(v_j) := min\{k | k \notin N_c(v_j)\}$;
14: **fin si**
15: Ajouter v_j à T ; {Tout en respectant l'ordre décroissant des degrés}
16: **pour tout** $v_h \in N(v_j)$ **faire**
17: Ajouter $c(v_j)$ à $N_c(v_h)$;
18: **fin pour**
19: **fin pour**
20: **si** $N_c(v_i) = L - \{c(v_i)\}$ **alors**
21: Ajouter $c(v_i)$ à D_m ;
22: **fin si**
23: Retirer v_i de T ;
24: **jusqu'à** $T = \emptyset$

Proposition 1. *(Elghazel et al. [18])*
L'algorithme 1 est en $O(n^2\Delta)$

Étape 2 Une fois une coloration initiale de $G_{>S_d}$ obtenue, on cherche une b-coloration du graphe par l'algorithme 2. Le principe de l'algorithme est de retirer les couleurs non dominantes une à une du graphe tout en recolorant *judicieusement* les sommets jusqu'à obtenir une b-coloration. Le choix de la couleur utilisée pour la recoloration d'un sommet se fait sur la base de la proximité (au sens de la fonction *dist*). Ceci permettrait d'obtenir des clusters plus homogènes.

Si la couleur d'un sommet v_i est modifiée, elle est choisie de manière à ce qu'elle ne soit pas présente dans son voisinage. Ainsi, la coloration générée par l'algorithme 2 est *propre*.

Par ailleurs, tout sommet ayant une couleur non dominante peut être recoloré. En effet, si un sommet a une couleur non dominante cela implique qu'il existe une couleur c à laquelle il n'est pas adjacent et par laquelle il peut être recoloré.

Enfin, l'algorithme retire les couleurs non dominantes jusqu'à leur élimination. La coloration générée par l'algorithme est donc bien une *b-coloration*. (cf. [18])

Algorithm 2 find-b-coloring()

Requiert: $G_{>S_d} = (V, E\prime, d)$ un graphe ayant une coloration utilisant $\Delta + 1$ couleurs

1: **répéter**
2: $q := max\{h/h \in ND_m\}$;
3: $L := L - \{q\}$;
4: $ND_m := L - D_m$;
5: **pour tout** v_i tel que $c(v_i) = q$ **faire**
6: $H := \{h/h \in L$ et $h \notin N_c(v_i)\}$;
7: $c(v_i) := argmin_h \in H(dist(v_i, h))$; {On choisie la couleur la plus proche possible}
8: **fin pour**
9: **pour tout** v_j tel que $c(v_j) \in ND_m$ **faire**
10: Actualiser($N_c(v_j)$) ; {$N_c(v_j)$ est mis à jour après élimination de q}
11: **si** $N_c(v_j) = L - \{c(v_j)\}$ **alors**
12: Ajouter $c(u)$ à D_m ;
13: **fin si**
14: **fin pour**
15: **jusqu'à** $ND_m = \emptyset$

Proposition 2. *(Elghazel et al. [18])*
L'algorithme 2 est en $O(n\Delta^2)$.

Étape 3 La partition obtenue par la b-coloration doit être maintenant évaluée. Pour cela, Elghazel et al. optent pour l'indice de Dunn généralisé (cf. paragraphe 3.3.2).

Fin pour

On retourne la partition ayant un indice $Dunn_G$ maximum. Cette partition est une solution de compromis entre les critères de densité et de bonne séparation des clusters.

Fin de la méthode.

Exemple. On considère l'exemple donné par la matrice des distances suivante

$$\begin{array}{c} \begin{array}{ccccccc} A & B & C & D & E & F & G \end{array} \\ \left[\begin{array}{ccccccc} 0 & & & & & & \\ 12 & 0 & & & & & \\ 3 & 11 & 0 & & & & \\ 11 & 5 & 11 & 0 & & & \\ 10 & 4 & 4 & 7 & 0 & & \\ 9 & 4 & 12 & 10 & 11 & 0 & \\ 7 & 7 & 3 & 10 & 9 & 11 & 0 \end{array}\right] \end{array}$$

Tab. 3.1 – Matrice des distances

MCE consiste à construire plusieurs graphes seuil-supérieur (en Anglais *superior threshold graphs*) et chercher une b-coloration pour chacun d'eux. La meilleure b-coloration obtenue par rapport à l'indice $Dunn_G$ est retenue. Nous construisons le graphe seuil-supérieur $G_{>7}$ (cf. Fig 3.3).

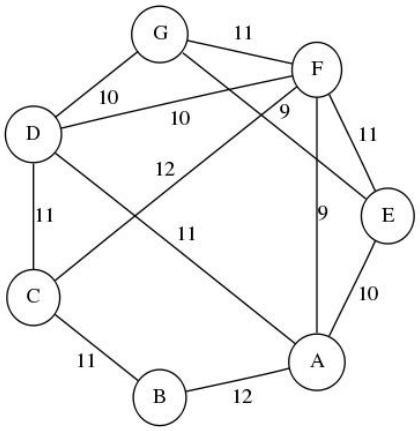

FIGURE 3.3 – $G_{>7}$

Dans un premier temps, on cherche à l'aide de l'algorithme Init-b-coloring() une coloration initiale de $G_{>7}$ utilisant $\Delta_{G_{>7}} + 1$ couleurs.

Initialement, le sommet F, de degré maximum, prend la couleur 1 et est ajouté à la liste T. Init-b-coloring() sélectionne l'unique sommet présent dans T, F. Les sommets sans couleur présents dans le voisinage de F sont alors colorés et ajoutés à T en respectant un ordre décroissant des degrés.

Après une première itération, les sommets voisins de F prennent la coloration de la figure 3.4 et T vaut (F D A G E C). Le sommet B n'est pas encore coloré car il n'est pas voisin de F. On supprime F de T et on prend le sommet D à la prochaine itération.

A la sortie, on obtient la coloration de la figure 3.5. On remarque que la couleur 1 est dominante tandis que les couleurs $\{2, 3, 4, 5, 6\}$ ne le sont pas ($D_m = \{1\}$ et $ND_m = \{2, 3, 4, 5, 6\}$).

Dans un second temps, on cherche à l'aide de la procédure find-b-coloring() une b-coloration de $G_{>7}$ à partir de la coloration initiale obtenue par Init-b-coloring().

On choisit de supprimer la couleur non dominante 6. Les sommets de cette couleur prennent la couleur qui leur est la plus proche.

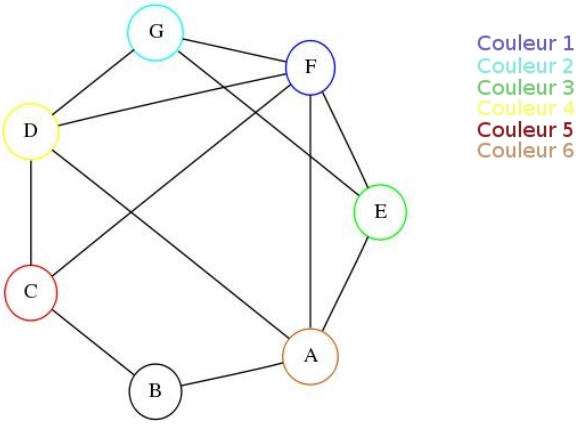

FIGURE 3.4 – Init-b-coloring() : Première itération

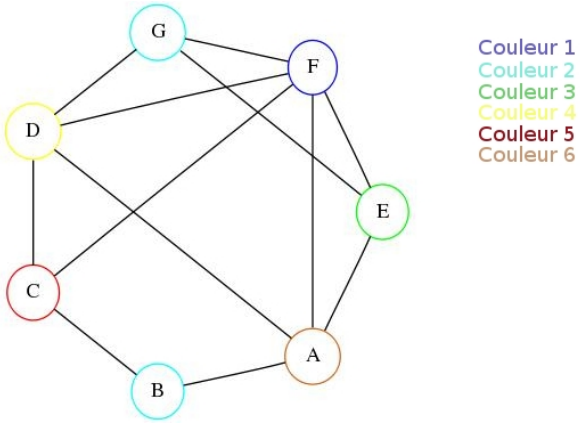

FIGURE 3.5 – Init-b-coloring() : Coloration propre obtenue

Le sommet A de couleur 6 ne peut prendre que la couleur 5 car elle ne figure pas dans son voisinage. Cette affectation permet à la couleur 5 de devenir dominante car A est un b-sommet pour cette dernière (cf. figure 3.6). $D_m = \{1, 5\}$ et $ND_m = \{2, 3, 4\}$.

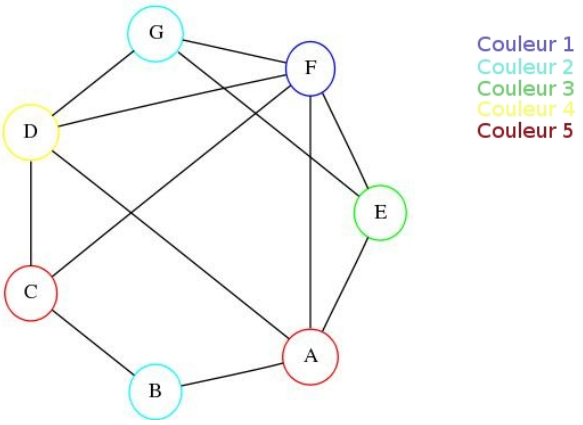

FIGURE 3.6 – find-b-coloring() : Première itération

On obtient après application de find-b-coloring() la coloration de la figure 3.7. Toutes les couleurs sont dominantes.

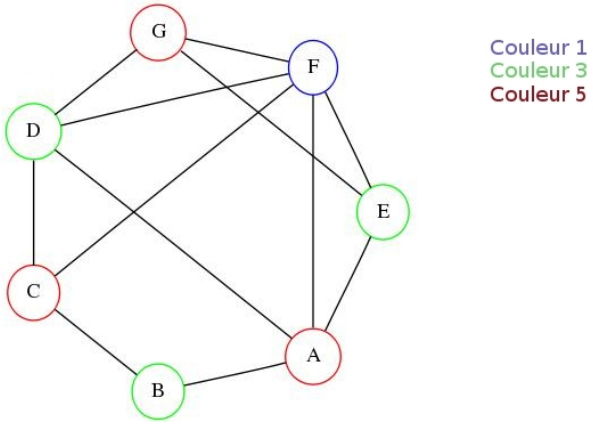

FIGURE 3.7 – find-b-coloring() : b-coloration obtenue

Enfin, on évalue la partition trouvée par rapport à l'indice $Dunn_G$. La partition obtenue pour $G_{>7}$ a un indice de 1.56 et c'est la meilleure partition trouvée pour cet exemple.

Le tableau 3.2 résume toutes les partitions trouvées pour cet exemple.

S_d	Partition trouvée	#Clusters	$Dunn_G$
3	$\{F\}\{G\}\{E\}\{D\}\{C,A\}\{B\}$	6	1.33
4	$\{D\}\{G\}\{F\}\{E,B\}\{C,A\}$	5	1.25
5	$\{G,C\}\{F\}\{E,B\}\{D\}\{A\}$	5	1.25
7	$\{F\}\{E,D,B\}\{G,C,A\}$	3	1.56
9	$\{F,A\}\{G,E,C\}\{D,B\}$	3	0.93
10	$\{F,A\}\{G,D,B\}\{E,C\}$	3	0.83
11	$\{G,F,E,D,B\}\{C,A\}$	2	1.15

Tab. 3.2 – Partitions obtenues

3.5 Conclusion

Nous avons vu dans ce chapitre divers outils de partitionnement. Nous avons donné un descriptif détaillé de l'un de ces outils qui est MCE, la méthode de clustering développée par Elghazel et al. en 2006 (cf. [18]). Depuis son introduction, MCE a fait ses preuves (cf. [20], [25]) et a connu de multiples extensions (cf. [19], [17]). Sur le base de [18], la recherche d'une partition par la méthode pour un seuil donné est en $O(n^2\Delta)$.

Nous proposons dans le chapitre suivant une manière d'appliquer MCE au problème du voyageur de commerce multiple afin d'obtenir dans ce dernier une partition intéressante des villes à visiter. Une telle partition nous permettrait de développer une heuristique efficace pour ce problème ou l'une de ses variantes.

Chapitre 4

Application au m-TSP

4.1 Introduction

MCE est une méthode de clustering, basée sur la b-coloration, qui a connu de multiples évolutions et applications (cf. paragraphe 3.4 page 28). Une étude comparative de la méthode avec d'autres solutions de clustering menée en [20] a montré des résultats probants qui attestent de la robustesse de la méthode.

Son application au domaine médical (cf. [18]) et au domaine du tri automatique de documents (cf. [25]) a été concluante.

Nous proposons ici une application au problème du voyageur de commerce multiple dans le but de développer une heuristique efficace pour ce problème ou une variante de celui-ci.

4.2 Motivation

La solution d'un m-TSP peut être décrite sous la forme d'une *partition* en m sous ensembles de villes *ordonnées*, chaque sous ensemble étant visité par un véhicule. Plusieurs approches peuvent être envisagées pour trouver une solution approchée au problème. Elles s'inscrivent généralement dans l'une des deux techniques suivantes (cf. [10]) :

- Chercher d'abord une tournée globale qui fasse passer un véhicule par toutes les villes puis éclater celle-ci en m tournées, ou bien ;
- Partitionner les villes à visiter en m sous ensembles puis chercher une tournée pour chaque véhicule.

Si l'on considère la deuxième approche, on comprend que l'étape de partition des villes est cruciale pour obtenir une bonne solution au m-TSP.

Notre but est d'appliquer MCE au m-TSP pour obtenir une partition des villes de manière à ce que :

- Les villes se trouvant au sein d'un même groupe soient à proximité l'une de l'autre ;
- Les villes se trouvant au sein de groupes différents soient trop éloignées pour être visitées par un même véhicule.

Pourquoi la méthode d'Elghazel et al. ?

Nous présentons dans ce paragraphe des arguments qui justifient le choix de MCE.

Performance Comme cité dans l'introduction, la méthode de clustering développée par Elghazel et al. est une méthode qui, comparée à d'autres, se révèle compétitive par rapport à l'indice de Dunn généralisé (cf. [20]).

Des extensions et des applications Bien qu'elle soit récente, la méthode a connue de multiples extensions et de multiples applications.

Nous pensons que les évolutions connues par la méthode peuvent servir dans des variantes du m-TSP. En effet, dans [17], la méthode a été adaptée au *dynamic clustering* où l'on cherche à garder les groupes bien denses et bien séparés à la venue de nouveaux objets ou au départ de ces derniers. Cette extension est intéressante dans le cas du *Online* m-TSP où le problème évolue pendant que les véhicules sont en service. Autrement dit, de nouvelles villes, qui ne se trouvait pas dans le routage initiale, doivent être affectées *judicieusement* aux véhicules pendant la tournée de ces derniers. Une autre extension connue par la méthode est la prise en compte des contraintes *must link* et *cannot link*. Ces contraintes forcent deux objets à être dans un même groupe ou au contraire à se trouver dans des groupes différents. Dans le m-TSP, cette extension serait utile si on impose que deux villes soient visitées par un même véhicule ou au contraire qu'elle ne le soient pas.

Par ailleurs, la méthode a été appliqué avec réussite au domaine médicale et au tri automatique des documents. Cependant, nous ne trouvons pas, dans la littérature, une application de la méthode à un problème d'optimisation combinatoire. Cela nous encourage à en proposer une.

Exemple. *On considère l'instance du VRP de Christofides et Eilon à sept villes (disponible dans la bibliothèque TSPLIB*[1]*). Le sommet 1 représente le dépôt. Les véhicules sont supposés avoir une capacité suffisamment grande pour visiter tous les clients (pour se ramener au cas du m-TSP).*

$$\begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \left[\begin{array}{ccccccc} 0 & & & & & & \\ 10 & 0 & & & & & \\ 20 & 12 & 0 & & & & \\ 25 & 20 & 10 & 0 & & & \\ 25 & 25 & 11 & 2 & 0 & & \\ 20 & 30 & 22 & 11 & 10 & 0 & \\ 10 & 20 & 30 & 25 & 20 & 12 & 0 \end{array}\right] \end{array}$$

Tab. 4.1 – Matrice des distances

On cherche une partition intéressante des villes $\{2, \ldots, 7\}$*. Pour cela, on construit autant de graphes seuil-supérieur qu'il y a de valeurs dans la matrice des distances. On*

1. http ://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html

cherche une b-coloration à chacun de ces graphes et on évalue la b-coloration trouvée selon l'indice $Dunn_G$.

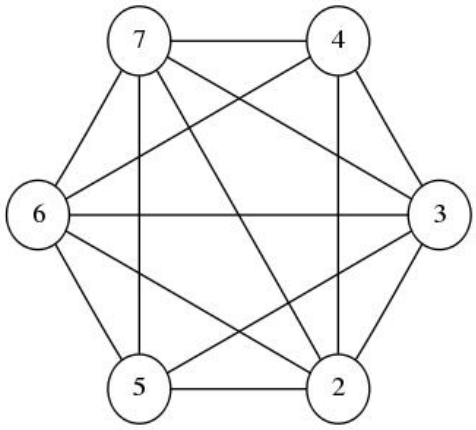

FIGURE 4.1 – Graphe seuil-supérieur $G_>2$

Le graphe $G_>2$ (cf. Fig 4.1) est celui pour lequel on trouve une meilleure b-coloration (cf. Fig 4.2).

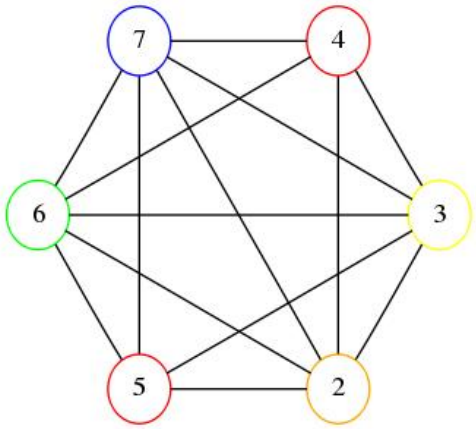

FIGURE 4.2 – b-coloration de $G_>2$

Cette b-coloration utilise 5 couleurs et a un indice $Dunn_G$ de 5.25. La partition de l'ensemble des villes $\{2,\ldots,7\}$ qu'elle décrit est :

$$\{2\}\{3\}\{4,5\}\{6\}\{7\}$$

.

La recherche de 5 tournées optimales démarrant du dépôt et passant par les villes de chacun de ces sous ensembles mène à trouver le routage suivant :

$$(121),(131),(1451),(161),(171)$$

La recherche d'un routage à notre problème par le solveur SYMPHONYVRP[1] *donne ce routage même.*

1. http ://branchandcut.org/

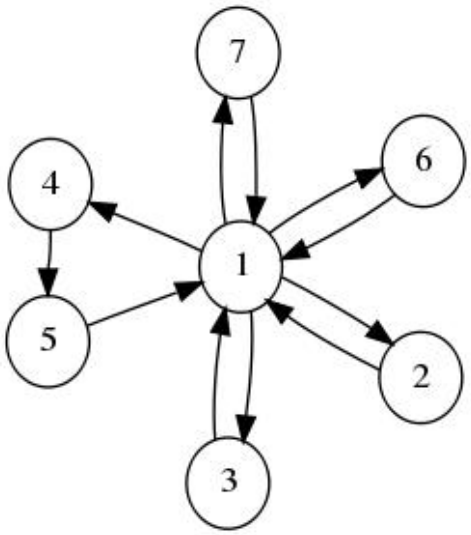

FIGURE 4.3 – Routage utilisant 5 véhicules (distance totale 102)

S_d	$Dunn_G$	Partition trouvée	m	Routage de SYMPHONY VRP utilisant m véhicules
2	5.25	$\{2\}\{3\}\{4,5\}\{6\}\{7\}$	5	$(2),(3),(45),(6),(7)$
10	1.15	$\{2\}\{3,4\}\{5,6\}\{7\}$	4	$(2),(345),(6),(7)$
11	1.56	$\{2\}\{3,4,5\}\{6\}\{7\}$	4	$(2),(345),(6),(7)$
12	1.58	$\{2,3\}\{4,5,6\}\{7\}$	3	$(2),(3456),(7)$
20	0.55	$\{2,7\}\{3,5\}\{4,6\}$	3	$(2),(3456),(7)$
22	1.09	$\{2,7\}\{3,4,5,6\}$	2	$(23456),(7)$
25	1.57	$\{2,3\}\{4,5,6,7\}$	2	$(23456),(7)$

Tab. 4.2 – Partitions obtenues et routages de SYMPHONY VRP

4.3 Approche

4.3.1 Préliminaire

Rappelons que notre but est de générer, à partir de la méthode de clustering basée sur la b-coloration, une partition intéressante des villes à visiter dans le m-TSP. Pour évaluer l'utilité d'une partition P, nous générons une tournée optimale passant par toutes les villes d'un cluster ainsi que par le dépôt. Nous appelons un tel routage, le routage *induit* par la partition P.

Pour ce faire, nous faisons appel à un solveur du TSP symétrique appelé *Concorde*[1]. Ce solveur développé par Applegate et al. a permis de résoudre des instances du STSP allant jusqu'à 85.900 villes (cf. [4]).

4.3.2 Une première approche

Nous pensions initialement appliqué la méthode de clustering d'Elghazel et al. au m-TSP de la manière suivante :

1. http ://www.tsp.gatech.edu/concorde/index.html

– Appliquer la méthode, telle que décrite dans [18], sur l'ensemble des villes à visiter (sans le dépôt) et retenir la meilleure partition trouvée ;
– Pour chaque cluster, trouver une tournée optimale d'un véhicule passant par le dépôt ainsi que par toutes les villes du cluster.

Cette approche s'est révélée inutilisable en pratique. En effet, la méthode d'Elghazel et al. ne permet pas de contrôler le nombre de couleurs utilisées lors de la recherche d'une b-coloration. Ainsi, la meilleure b-coloration (ou partition) trouvée par la méthode peut utiliser deux couleurs comme elle peut en utiliser $n-1$ (n étant le nombre d'objets). Ceci peut impliquer, dans le m-TSP, l'obtention d'un routage utilisant un nombre raisonnable de véhicules (cf. Fig 4.4) ou au contraire un routage utilisant quasiment un véhicule pour chaque ville (cf. Fig 4.5), chose pour laquelle nous ne voyons pas d'utilité pratique.

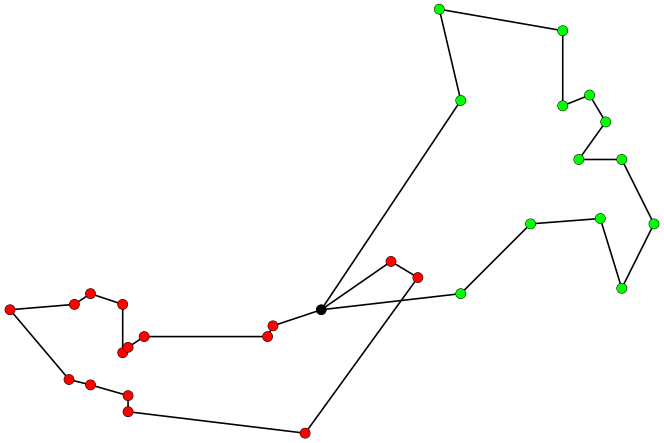

FIGURE 4.4 – Routage induit par la meilleure partition trouvée sur eil30 (instance de TSPLIB) : Sd **79**, distance totale **409**, véhicules utilisés **2**

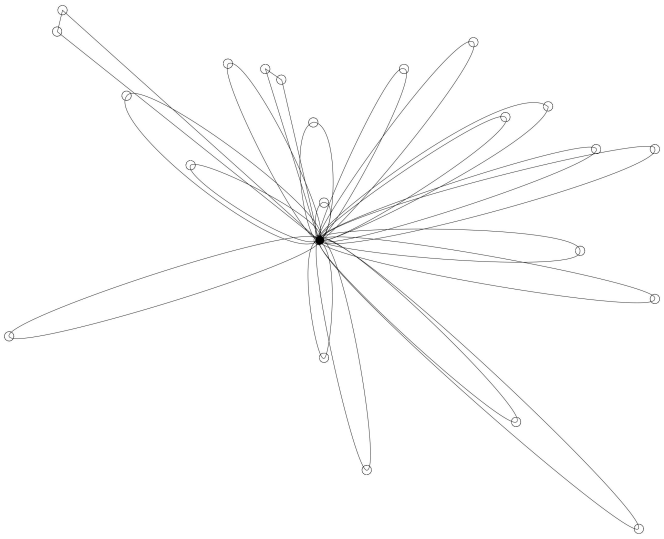

FIGURE 4.5 – Routage induit par la meilleure partition trouvée sur eil23 (instance de TSPLIB) : Sd **4**, distance totale **1788**, véhicules utilisés **20**

Par ailleurs, Elghazel et al. ont proposé en 2007 dans l'article [21] un algorithme de recoloration permettant d'*améliorer* la partition retournée en sortie par MCE.

L'algorithme partitionne les sommets du graphe seuil-supérieur où la meilleure b-coloration a été obtenue en deux sous ensembles de sommets. Le premier est constitué de sommets appelés sommets *critiques* sur lesquels aucune recoloration n'est effectuée. Le deuxième est constitué de sommets appelés sommets *non critiques* sur lesquels l'algorithme opère une recoloration pour améliorer l'indice $Dunn_G$ tout en préservant la b-coloration (coloration propre et préservation d'un b-sommet pour chaque couleur). Les auteurs prouvent que la partition ainsi obtenue est de même qualité ou meilleure que la partition sur laquelle la recoloration est effectuée.

Nous choisissons d'utiliser cet algorithme pour améliorer la qualité de la solution que nous proposons pour le m-TSP.

4.3.3 Approche choisie

Les constatations et remarques soulevées au paragraphe précédent, nous ont mené à adopter l'approche suivante :

Soit $G = (V, E)$ le graphe complet représentatif d'un m-TSP symétrique et D la matrice des distances :

- $V = \{1, \ldots, n\}$: ensemble des villes avec 1 représentant le dépôt ;
- $d(v_i, v_j)$: longueur d'un plus court chemin entre v_i et v_j ;

Pour toute distance S_d présente dans D **faire**

1. Construction du graphe seuil-supérieur $G_{>S_d} = (V\prime, E\prime)$ où :
   - $V\prime = \{2, \ldots, n\}$: ensemble des villes à visiter (le dépôt n'est pas considéré) ;
   - $(v_i, v_j) \in E\prime$ ssi $d(v_i, v_j) > S_d$;
2. Initialiser la coloration de $G_{>S_d}$ par l'algorithme *init-b-coloring* page 30 en utilisant $\Delta_{G_{>S_d}} + 1$ couleurs ;
3. Chercher une b-coloration de $G_{>S_d}$ par l'algorithme *b-coloring* page 31 ;
4. Chercher par l'algorithme de recoloration (cf. [21]) une nouvelle partition à partir de la b-coloration obtenue en 3. . Notons que l'indice $Dunn_G$ est recalculé à chaque itération et que $C\prime$ n'est modifié que si il y a amélioration ;
5. Chercher le routage induit par la b-coloration trouvée en 3. que nous appelons routage *initial*) ;
6. Chercher le routage induit par la recoloration que nous appelons routage *alternatif*) ;

fin pour

De cette manière, nous obtenons un ensemble de partitions utilisant différents nombres de couleurs, ceci permet de remédier au cas où la partition générée par MCE utilise un nombre trop important de couleurs. Par ailleurs, l'introduction de l'algorithme de recoloration permet d'enrichir l'ensemble des routages induits et d'améliorer la qualité de la solution proposée pour le m-TSP.

4.4 Conclusion

Nous avons proposé dans ce chapitre une approche pour l'application de MCE au m-TSP. Cette application a pour but de générer une partition intéressante des villes à visiter dans le m-TSP. Pour évaluer cette partition, nous faisons appel à un solveur de STSP nommé *Concorde*. Dans le chapitre suivant, nous évaluons cette approche sur des instances connues de la littérature et nous tirons des conclusions.

Chapitre 5

Résultats et discussion

5.1 Introduction

Ce chapitre est consacré au test de notre approche. Deux types d'instances sont retenues. Les premières sont de petits exemples traités par Elghazel et al. dans leurs articles [18] et [20]. Les secondes sont des instances connues du VRP. Nous adaptons ces dernières au m-TSP en affectant aux capacités des véhicules, une valeur supérieure à la demande de la totalité des clients. Le second type d'instances provient de la bibliothèque TSPLIB.

Nous commençons par proposer une manière d'obtenir, avec les outils disponibles, un clustering de meilleure qualité qu'en recolorant uniquement la meilleure b-coloration initiale (cf. paragraphe 5.2). Nous testons ensuite cette amélioration au paragraphe 5.3.1. Enfin, nous évaluons l'approche proposé pour le m-TSP au chapitre précédent et nous ouvrons des perspectives de recherche (cf. paragraphes 5.3.2, 5.3.3 et 5.3.4).

5.2 A propos du Clustering

Elghazel et al. présentent leurs contributions dans les articles [18] et [20] en utilisant les exemples de clustering présentés dans les tableaux 5.1 et 5.2 respectivement.

v_i	A	B	C	D	E	F	G
A	0						
B	0.20	0					
C	0.20	0.30	0				
D	0.10	0.20	0.25	0			
E	0.20	0.20	0.10	0.40	0		
F	0.20	0.20	0.20	0.25	0.20	0	
G	0.25	0.20	0.20	0.10	0.20	0.65	0

Tab. 5.1 – Matrice des distances du *premier* exemple (cf. [18])

Les tableaux 5.3 et 5.4 résument les résultats que nous obtenons, avec notre programme, pour le *premier* et le *second* exemple respectivement.

v_i	A	B	C	D	E	F	G	H	I
A	0								
B	0.20	0							
C	0.10	0.30	0						
D	0.10	0.20	0.25	0					
E	0.20	0.20	0.10	0.40	0				
F	0.20	0.20	0.20	0.25	0.65	0			
G	0.15	0.10	0.15	0.10	0.10	0.75	0		
H	0.10	0.20	0.10	0.10	0.05	0.05	0.05	0	
I	0.40	0.075	0.15	0.15	0.15	0.15	0.15	0.15	0

Tab. 5.2 – Matrice des distances du *second* exemple (cf. [20])

S_d	Partition d'Elghazel et al.	$Dunn_G$	Notre partition	$Dunn_G$
0.1	$\{A\}\{B\}\{C,E\}\{G,D\}\{F\}$	1.75	$\{B\}\{C,E\}\{D,A\}\{F\}\{G\}$	1.75
0.2	$\{C,G\}\{F,B,E\}\{D,A\}$	1.00	$\{A,D\}\{B,F\}\{C,E,G\}$	1.06
0.25	$\{F,A,B,D\}\{C,E,G\}$	1.37	$\{A,C,E,G\}\{B,D,F\}$	1.15
0.3	$\{F,A,B,C,D\}\{E,G\}$	1.19	$\{A,B,C,E,G\}\{D,F\}$	1.00
0.4	$\{F,A,B,C,D,E\}\{G\}$	1.25	$\{A,B,C,D,E,G\}\{F\}$	1.37

Tab. 5.3 – Résultats pour le premier exemple

Les auteurs remarquent que pour un même graphe seuil-supérieur, plusieurs b-colorations peuvent exister. Ces b-colorations peuvent avoir le même indice $Dunn_G$ (comparer les partitions trouvées pour $S_d = 0.1$ dans le tableau 5.3) ou bien des indices différents (comparer les partitions trouvées pour $S_d = 0.4$ dans le tableau 5.3). En conséquent, les auteurs proposent un algorithme de recoloration permettant d'obtenir une b-coloration *au moins aussi bonne* que celle obtenue initialement (cf. [21]).

S_d	Notre partition	$Dunn_G$
0.05	$\{A\}\{B\}\{C\}\{D\}\{E,H\}\{F\}\{G\}\{I\}$	1.50
0.075	$\{A\}\{B,I\}\{C\}\{D\}\{E,H\}\{F\}\{G\}$	1.00
0.10	$\{A,D,H\}\{E,C\}\{G,B\}\{F\}\{I\}$	1.125
0.15	$\{A,D\}\{B\}\{C,E,G,H,I\}\{F\}$	1.52
0.20	$\{A,C,E,G\}\{B,F,H,I\}\{D\}$	1.27
0.25	$\{A,B,E,G\}\{C,D,F,I,H\}$	1.40
0.30	$\{B,C,D,F,H,I\}\{A,E,G\}$	1.30
0.40	$\{F,A,B,C,D,H,I\}\{E,G\}$	1.32
0.65	$\{A,B,C,D,E,G,H,I\}\{F\}$	**1.92**

Tab. 5.4 – Nos partitions pour le second exemple

Nous remarquons que pour le second exemple, nous obtenons notre meilleure b-coloration en $G_{>}0.65$ (cf. Tab 5.4). Pour le même exemple, les auteurs obtiennent la

meilleure b-coloration en $G_{>}0.15$ (cf. Tab 5.5). En effet, l'indice $Dunn_G$ de notre partition est de 1.92 contre 1.52 pour la partition des auteurs. Or, la partition qu'ils donnent en alternative dans l'article [21] n'est que de 1.53, soit toujours inférieure à celle que nous obtenons.

S_d	Partition d'Elghazel et al.	$Dunn_G$
0.05	$\{A\}\{B\}\{C\}\{D\}\{E,H\}\{F\}\{G\}\{I\}$	1.50
0.075	$\{A\}\{B,I\}\{C\}\{D\}\{E,H\}\{F\}\{G\}$	1.00
0.10	$\{A\}\{B,I\}\{E,C\}\{G,D\}\{F,H\}$	1.25
0.15	$\{A,D\}\{B\}\{C,E,G,H,I\}\{F\}$	**1.52**
0.2	$\{D,A,B,H\}\{E,C,G,I\}\{F\}$	1.16
0.25	$\{A,B,E,G,H\}\{C,D,F,I\}$	1.15
0.30	$\{A,E,G,B,C,H\}\{D,F,I\}$	1.30
0.40	$\{F,A,B,C,D,H,I\}\{E,G\}$	1.32
0.65	$\{F,A,B,C,D,E,H,I\}\{G\}$	1.01

Tab. 5.5 – Partitions des auteurs pour le second exemple

Cette constatation nous encourage à penser que, pour une instance donnée, la recoloration de tous les graphes seuil-supérieur permettrait d'obtenir une partition *au moins aussi bonne* que celle obtenue en recolorant *uniquement* le graphe seuil-supérieur ayant la meilleure b-coloration initialement. Nous discutons les résultats du test de cette approche au paragraphe 5.3.1.

S_d	Partitions	$Dunn_G$
S_d^1	P_1	α_1
S_d^2	P_2	α_2
$\vdots$	$\vdots$	$\vdots$
S_d^i	P_i	$\alpha_i \overset{recoloration}{\rightarrow} \alpha_i^*$
$\vdots$	$\vdots$	$\vdots$
S_d^p	P_p	α_p

S_d	Partitions	$Dunn_G$
S_d^1	P_1	$\alpha_1 \overset{recoloration}{\rightarrow} \alpha_1^*$
S_d^2	P_2	$\alpha_2 \overset{recoloration}{\rightarrow} \alpha_2^*$
$\vdots$	$\vdots$	$\vdots \quad \vdots \quad \vdots$
S_d^i	P_i	$\alpha_i \overset{recoloration}{\rightarrow} \alpha_i^*$
$\vdots$	$\vdots$	$\vdots \quad \vdots \quad \vdots$
S_d^p	P_p	$\alpha_p \overset{recoloration}{\rightarrow} \alpha_p^*$

Tab. 5.6 – Recoloration partielle contre recoloration totale

Comme l'indique le tableau 5.6 Elghazel et al. (à gauche) proposent de recolorer le graphe seuil-supérieur $G_{>S_d^i}$ de la meilleure b-coloration trouvée (P_i). Pour obtenir une meilleure b-coloration, nous proposons de recolorer $G_{>S_d^i}$ mais aussi tous les autres graphes seuil-supérieur (à droite). On obtiendrait ainsi une partition qui soit au moins aussi bonne que celle obtenue en recolorant uniquement $G_{>S_d^i}$. L'indice $Dunn_G$ de cette partition vaut $\alpha^{**} = max\{\alpha_1^*, \alpha_2^*, \ldots, \alpha_i^*, \ldots, \alpha_p^*\}$ qui est supérieur ou égale à α_i^*

5.3 A propos du problème du voyageur de commerce multiple

Nous avons testé l'algorithme sur les instances eil22, eil23, eil30, eil33, eil51, eilA76, eilA101 et gil262. Ces instances de TSPLIB qui sont des instances du problème de tournées de véhicules (VRP) ont été adaptées au m-TSP en affectant à la capacité des véhicules une valeur supérieure à la somme des demandes de tous les clients.

Ces instances sont de type euclidien i.e les villes sont définies par leurs coordonnées sur un plan et les distances entre elles sont calculées selon la distance euclidienne. Ce type d'instances présente l'avantage de pouvoir visualiser la solution obtenue. Cependant, d'autres types d'instances existent mais notre approche reste applicable tant que les distances sont symétriques.

Le matériel utilisé est un intel Core 2 Duo (2×1.8Ghz) avec 1Go de mémoire vive. Le système d'exploitation est Ubuntu 10.04 (lucid). Le langage et l'environnement de programmation sont le C++ et Code : :Blocks. Les bibliothèques et les exécutables auxquels nous faisons appel sont :

- LEMON [1] : une bibliothèque open source écrite en C++ axée sur la manipulation de graphes pour des tâches d'optimisation combinatoire ;
- Concorde [2] : un solveur reconnu de problèmes du voyageur de commerce symétriques. Nous faisons appel ici à la version *bibliothèque* de cet outil pour l'intégrer à notre application ;
- QSopt [3] : une bibliothèque pour la résolution de programmes linéaires à laquelle fait appel Concorde pour trouver ses solutions ;
- SYMPHONY (application au VRP) [4] : un exécutable pour la recherche d'une solution au VRP. Nous l'utilisons ici à titre comparatif.

Pour chaque instance, un prétraitement des données est d'abord effectué où nous calculons la partie entière des distances séparant les villes. Cet opération est en $O(n^2)$ si l'on considère que le nombre de villes est n.

Ensuite, nous appliquons notre approche telle qu'elle est décrite au paragraphe 4.3.3 page 40.

Durant l'exécution du programme, nous relevons une quantité d'informations qui nous permet d'évaluer notre approche. Les données relevées pour chaque instance et pour chaque graphe seuil-supérieur sont :

- Le nombre d'arêtes contenues dans le graphe ;
- Le degré maximum ($\Delta_{G_{>S_d}}$) ;
- Le nombre de couleurs utilisées par la b-coloration ;
- L'indice $Dunn_G$ de la b-coloration initiale ;
- L'indice $Dunn_G$ de la recoloration (sauf pour l'instance gil262) ;
- La distance totale du routage initial ;
- La distance totale du routage alternatif.

1. http ://lemon.cs.elte.hu/trac/lemon
2. http ://www.tsp.gatech.edu/concorde.html
3. http ://www.isye.gatech.edu/~wcook/qsopt/
4. http ://branchandcut.org/

A ces données s'ajoutent les temps d'exécutions mesurés pour arriver à chaque sous itération de la boucle (obtention d'une b-coloration initiale, d'une recoloration, d'un routage de la b-coloration initiale et de la recoloration) ainsi que le temps totale écoulé sur l'instance (lecture, préparation et traitement des données).

Nous analysons dans les paragraphes suivants les résultats obtenus.

5.3.1 Recoloration partielle *versus* recoloration totale

Nous proposons dans le paragraphe 5.2 une manière d'obtenir un clustering meilleur qu'en recolorant uniquement le graphe seuil-supérieur où la meilleure b-coloration a été obtenue. Nous proposons la recoloration de ce dernier graphe seuil-supérieur *ainsi* que la recoloration de tous les graphes seuil-supérieur restants. Nous constatons, sur toutes les instances étudiées, que cette approche permet d'obtenir un clustering qui, en qualité, est égal ou meilleur.

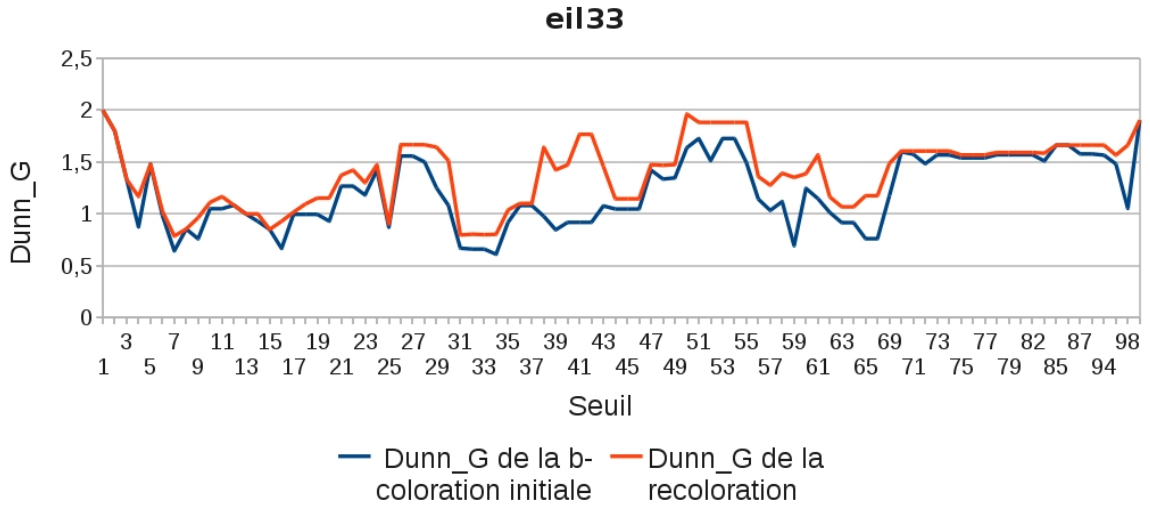

FIGURE 5.1 – eil33 : Nous obtenons le même indice $Dunn_G$

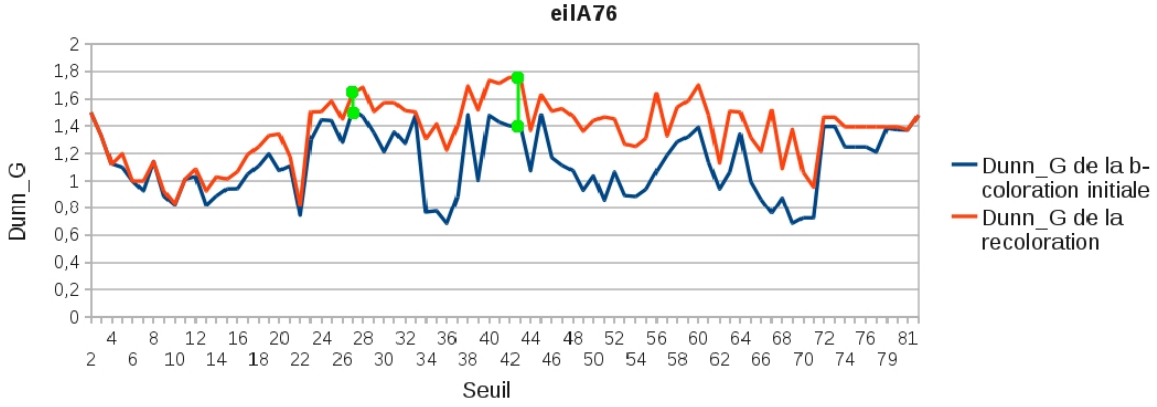

FIGURE 5.2 – eilA76 : Nous obtenons un meilleur indice $Dunn_G$

Comme le montre la figure 5.2, la recoloration de $G_{>43}$ donne une meilleure b-coloration que celle obtenue par la recoloration de $G_{>27}$ (celle ayant la meilleure b-coloration initialement).

5.3.2 Une recoloration est-elle nécessaire ?

Nous remarquons que le temps pris par la recoloration est relativement important pour une faible amélioration du routage. En effet, si nous observons la figure 5.3, nous constatons que, pour un graphe seuil-supérieur, le temps nécessaire à la recoloration est relativement très important. Cependant, le temps nécessaire à l'obtention d'une b-coloration initiale et du routage de celle-ci est relativement faible.

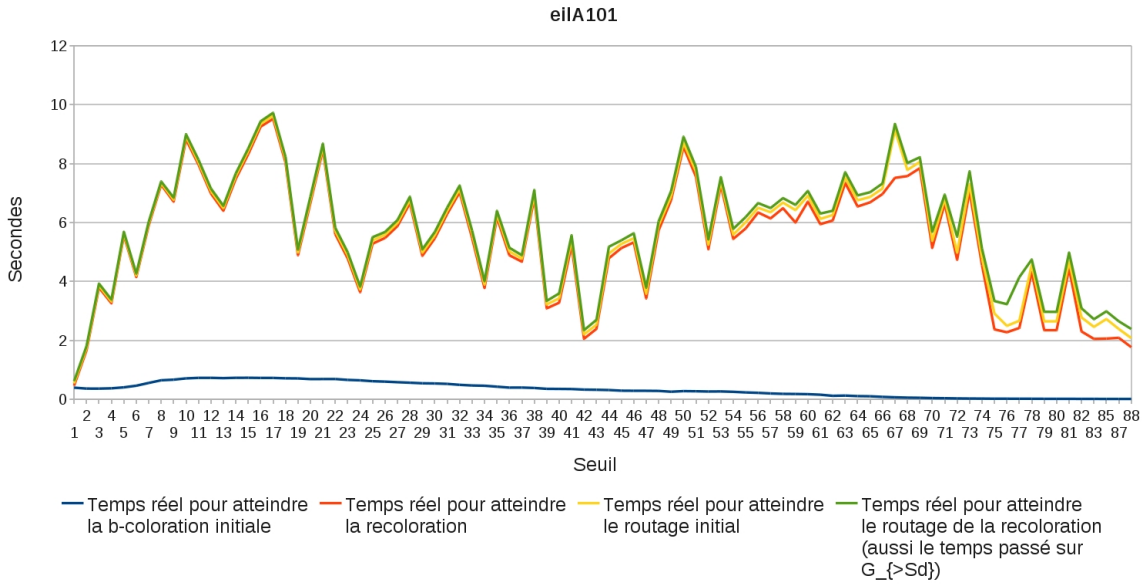

FIGURE 5.3 – eilA101 : Temps de calcul

De plus, nous remarquons que la qualité du routage obtenue sur une recoloration n'est généralement pas très éloignée de celle d'un routage obtenue sur la b-coloration initiale (cf. Fig 5.4).

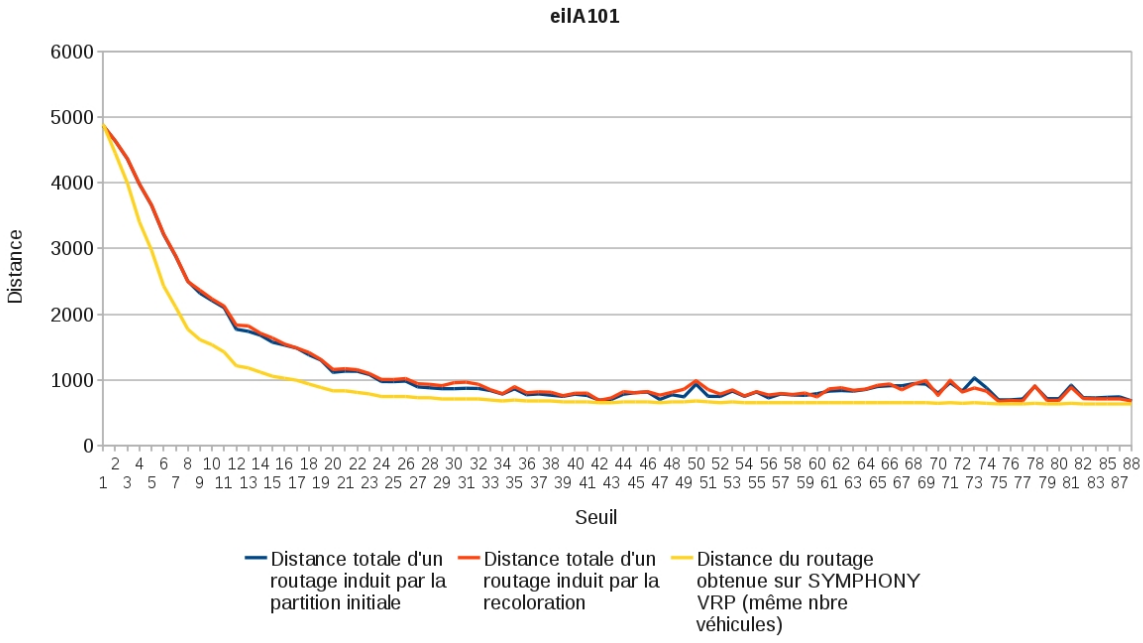

FIGURE 5.4 – eilA101 : Coûts des différents routages

Par conséquent, nous préférons utiliser la recoloration que lorsque nous nous intéressons à un seul graphe seuil-supérieur ou lorsque l'instance est relativement petite.

5.3.3 Doit-on considérer le dépôt lors de la phase de clustering ?

La position du dépôt par rapport aux autres villes se révèle importante. En effet, lorsque le dépôt est trop excentré par rapport aux villes à visiter, la partition qui est générée et le routage qui en découle sont biaisés (cf. Fig 5.5).

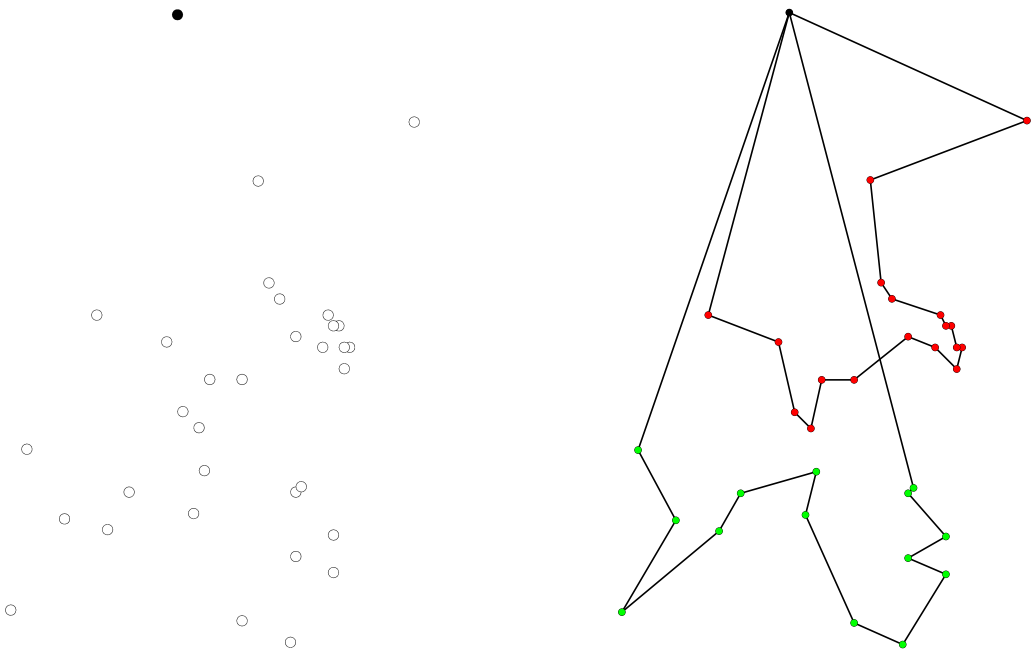

FIGURE 5.5 – eil33 : A gauche, représentation de l'instance (dépôt, en noir, excentré). A droite, routage initial de coût 578

Par contre, si le dépôt est au centre de gravité des villes alors la partition qui est générée et le routage qui en découle sont plus compétitifs (cf. Fig 5.6)

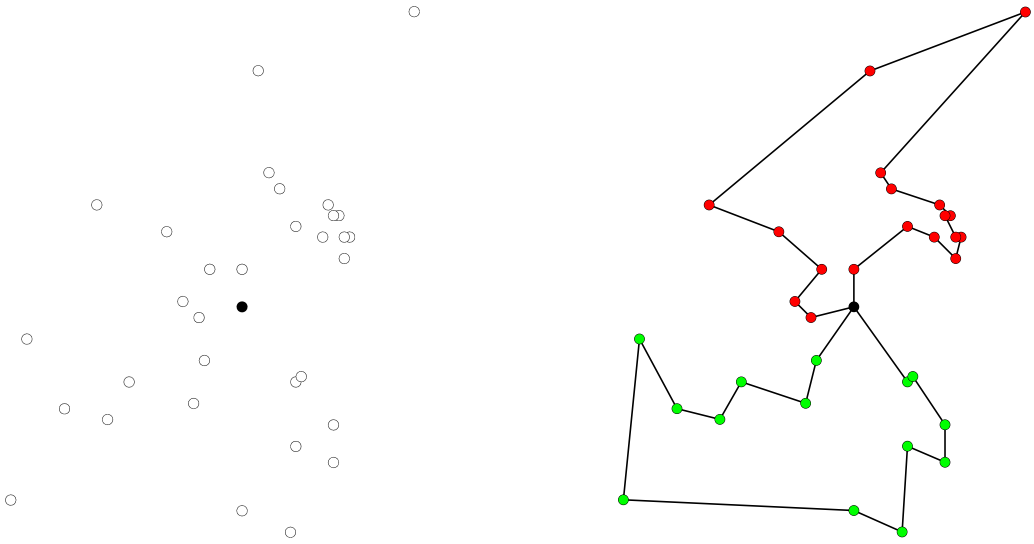

FIGURE 5.6 – eil33 modifié : A gauche, représentation de l'instance modifiée (dépôt, en noir, au centre de gravité). A droite, routage initial de coût 407.

Ceci mène à considérer la position du dépôt lors de la confection d'une solution au

problème. On peut ainsi se poser la question : Comment obtenir une partition des villes à visiter de manière à ce que les groupes formés par les clusters *et* le dépôt soient le plus compacts possibles et que ces mêmes groupes soient le plus séparés possibles.

5.3.4 Une contrainte implicite

Nous constatons que dans la majorité des cas, les routages obtenus par notre approche sont plus longs que ceux obtenus sur SYMPHONY VRP (cf. Fig 5.4 page 47). Nous expliquons cela par une contrainte *implicite* qui est introduite durant la phase de clustering. En effet, les villes se trouvant au sein d'un même cluster sont au plus éloignées d'une distance S_d selon la méthode d'Elghazel.
Notre approche est ainsi plus adaptée à une variante du m-TSP où l'on considère qu'un véhicule ne peut parcourir un tronçon plus long qu'une distance S_d. Cependant, pour parvenir à une solution satisfaisante pour cette variante, nous devons considérer une contrainte supplémentaire. En effet, nous devons garantir que dans le routage, le dépôt soit au plus à une distance S_d des deux villes qui lui sont voisines dans chacune des tournées. Ceci peut mener à une extension de la méthode d'Elghazel et al. au cas où un sommet (le dépôt) peut appartenir à plusieurs clusters, on parle dans ce cas de *fuzzy clustering* [26].

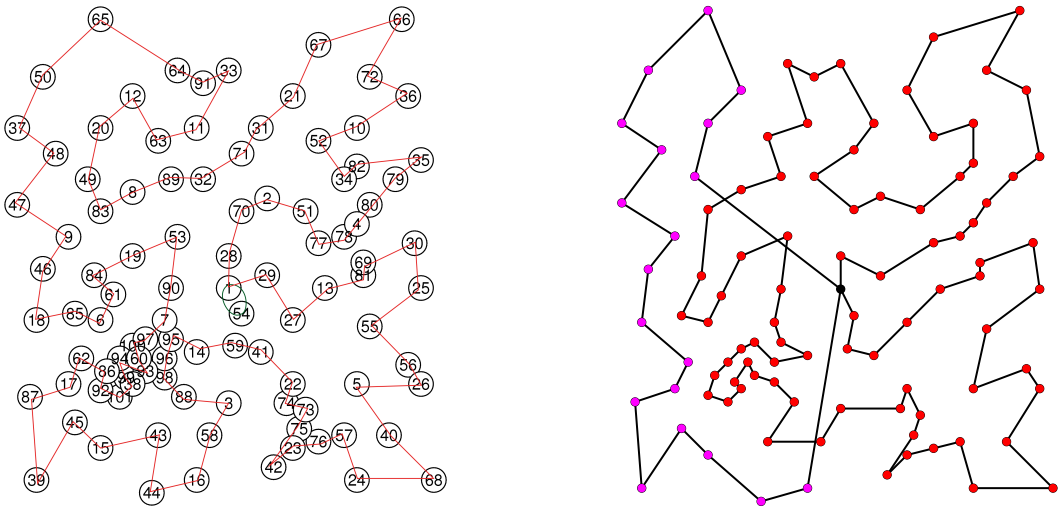

FIGURE 5.7 – eilA101 2 véhicules : A gauche meilleure solution de Symphony de coût 632 (moins *longue*). A droite notre meilleur solution de coût 671 (plus *équilibrée*).

Par ailleurs, il est à souligner que bien que les solutions que nous proposons soient plus *longues* que celles de SYMPHONY VRP, elles sont plus *équilibrées*. En effet, si nous comparons nos meilleurs solutions avec celles de SYMPHONY VRP (cf. Figures 5.7,5.8,5.9,5.10 et 5.11), nous constatons que la charge entre les véhicules est plus équilibrée dans nos solutions que dans celles de SYMPHONY VRP. Les solutions générées par ce dernier affectent généralement un ensemble de villes à un seul véhicule tandis que le restant des véhicules visitent une ville chacun.
Le critère d'équilibrage de la charge de travail est un critère qui est étudié dans la litterature (cf. [11], [47] ou [56] par exemple). Ce critère peut être utilisé lorsque le déséquilibre de la charge de travail sur les différents véhicules (machines, camions, etc) induit un coût (usure de la machine, kilométrage qui induit sur le prix de revente du véhicule, etc).

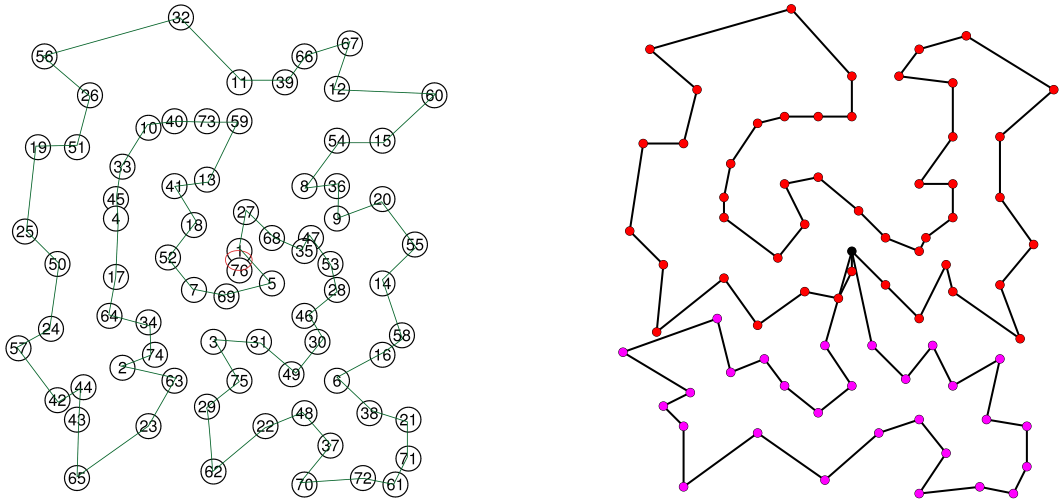

FIGURE 5.8 – eilA76 2 véhicules : A gauche meilleure solution de Symphony de coût 543 (moins *longue*). A droite notre meilleur solution de coût 581 (plus *équilibrée*).

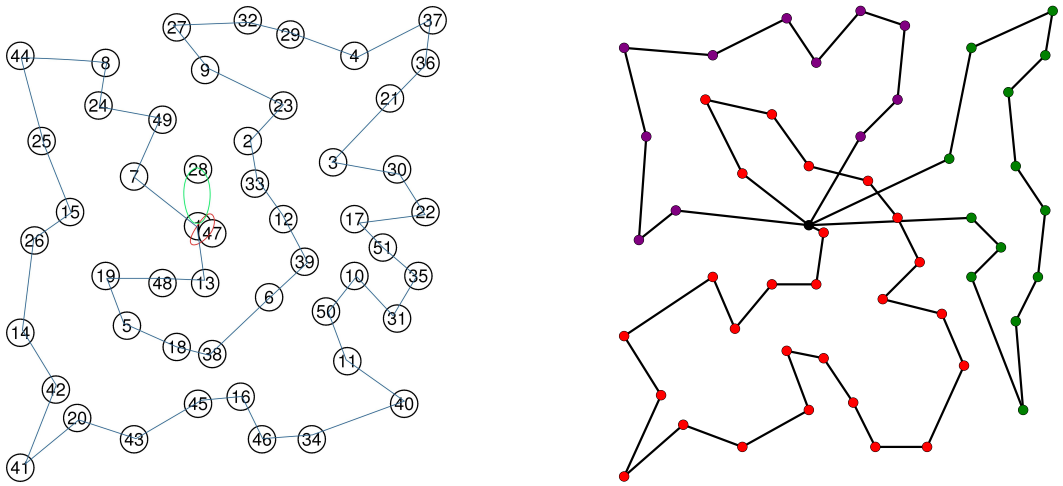

FIGURE 5.9 – eil51 3 véhicules : A gauche meilleure solution de Symphony de coût 439 (moins *longue*). A droite notre meilleur solution de coût 499 (plus *équilibrée*).

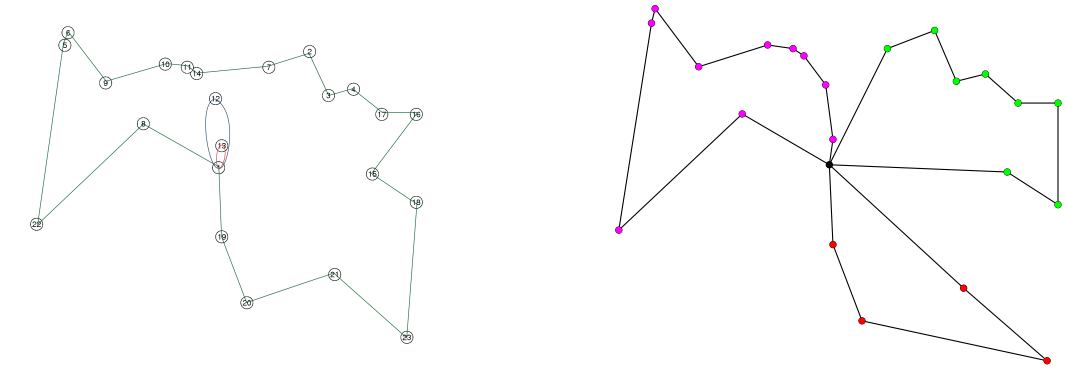

FIGURE 5.10 – eil23 3 véhicules : A gauche meilleure solution de Symphony de coût 516 (moins *longue*). A droite notre meilleur solution de coût 578 (plus *équilibrée*).

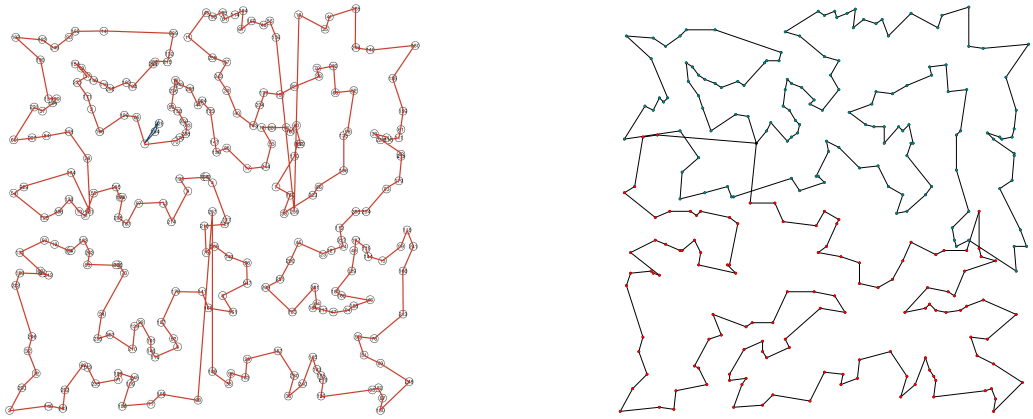

FIGURE 5.11 – gil262 2 véhicules : A gauche meilleure solution trouvée (temps limité) par Symphony de coût 2404 (moins *longue*). A droite notre meilleur solution de coût 2474 (plus *équilibrée*).

5.4 Conclusion

Dans ce chapitre, nous avons d'abord proposé de contribuer à l'amélioration de la qualité du clustering générée par MCE en recolorant l'ensemble des graphes seuil-supérieur au lieu de recolorer uniquement celui pour lequel la meilleur b-coloration est obtenu initialement et nous avons confirmé l'utilité de cette approche. Puis nous avons testé notre approche, proposée au chapitre précédent pour le m-TSP, sur plusieurs instances connues de la littérature. Dans cette dernière partie, nous avons constaté qu'une recoloration totale était coûteuse en temps pour peu d'amélioration du coût de routage. Ceci nous a mené à dire que cette approche était à éviter sauf pour des instances relativement petites (jusqu'à 100 villes). Aussi, nous avons attiré l'attention sur l'importance de considérer le dépôt dans la résolution du problème pour l'obtention de meilleurs solutions. Enfin, la comparaison de nos solutions avec celles de SYMPHONY VRP a permis de noter deux informations. La première est que nos solutions sont généralement plus longues mais plus équilibrées que celles données par SYMPHONY VRP ce qui nous mène à dire que notre approche serait plus adaptée au m-TSP avec équilibrage de la charge de travail. La deuxième est que nos solutions respectent la contrainte qui impose que deux villes se trouvant sur une même tournée (en dehors du dépôt) soient, au plus, éloignées d'une distance S_d. Ceci ouvre la voie à la résolution d'une variante du m-TSP où le véhicule ne peut parcourir un tronçon d'une distance supérieur à S_d (limite en heure de conduite, limite en autonomie, etc).

Conclusion

Le problème du voyageur de commerce multiple est un problème d'optimisation combinatoire. Ce problème consiste à optimiser la visite d'un ensemble de villes par une flotte de véhicules de manière à ce que les villes soient visitées exactement une fois et que les véhicules démarrent et finissent leurs tournées à un même point appelé dépôt. Ce problème, qui est NP-difficile, peut être considéré comme une généralisation, sur le nombre de véhicules, du problème du voyageur de commerce ou comme une relaxation, sur la capacité des véhicules, du problème de tournées de véhicules. Plusieurs méthodes, approchées (heuristiques, apprentissage non supervisé, métaheuristiques) et exactes (SEP, coupes), ont été développées pour sa résolution.

Nous avons proposé dans ce travail une nouvelle approche basée sur MCE, une méthode de clustering récemment introduite. Cette méthode, qui se base sur un concept de la théorie des graphes, la b-coloration, a présenté de meilleurs résultats en comparaison avec d'autres méthodes de clustering. Notre but était de développer une heuristique efficace pour le m-TSP qui soit basée sur MCE. L'idée soutenue était que si l'on parvenait a obtenir une partition intéressante des villes a visiter dans un m-TSP, on obtiendrait également un routage, induit par cette partition, qui soit intéressant.

La programmation et le test de notre approche sur des instances connues dans la littérature a permis de constater et de proposer des choses. Nous avons ainsi proposé et confirmé, sur des instances, l'utilité d'une recoloration globale contre une recoloration partielle dans MCE. Cependant, l'observation des temps d'exécution et des routages induits nous a mené à éviter une recoloration totale sur des instances du m-TSP trop grandes. Ce choix s'est imposé car une recoloration totale est coûteuse en temps d'exécution mais peu efficace du point de vue du routage.
Nous avons également constaté que la position du dépôt, par rapport aux villes à visiter, se répercute sur la qualité du routage induit par notre partition. Ceci nous encourage à prendre en considération ce point durant la phase de clustering ou après, par une heuristique d'amélioration.
Par ailleurs, nous avons pris conscience que notre approche introduisait une contrainte implicite qui empêche deux villes d'un même cluster d'être plus éloignées qu'une certaine distance. Cette contrainte augmente certes la distance totale parcourue par tous les véhicules mais participe à la confection de solutions où la charge de travail est équilibrée entre les véhicules, ce qui est apprécié en pratique.

Nous proposons ainsi les perspectives suivantes pour la suite de ce travail :

– Considérer le critère d'équilibrage de la charge de travail dans le m-TSP lorsqu'on effectue un clustering par la b-coloration sur l'ensemble des villes à visiter ;

– Aborder une variante du m-TSP où un véhicule ne peut parcourir un tronçon de longueur supérieur à S_d. Une telle contrainte serait utile dans le cas où l'autonomie des véhicules est fortement limitée ou que le parcours d'un tronçon trop lent se répercute négativement sur la fonction objectif.

Enfin, nous espérons que notre étude donnera des éléments de réponse aux chefs d'entreprise désirant mener a bien leurs distributions.

Bibliographie

[1] M. Abbas. Cours de théorie des graphes. USTHB, année universitaire 2006-2007.

[2] A.I. Ali and J. L. Kennington. The asymmetric m-travelling salesmen problem : A duality based branch-and-bound algorithm. *Discrete Applied Mathematics*, 13(2-3) :259 – 276, 1986.

[3] K. Allab. *Eléments d'Analyse*, page 126. Office des Publications Universitaires, Alger, 2002.

[4] D.L. Applegate. *The traveling salesman problem : a computational study Princeton series in applied mathematics*, chapter 1, page 56. Princeton University Press, 2006.

[5] B. Bachelet. Efficacité des algorithmes, complexité des problèmes. http ://www.nawouak.net/.

[6] T. Bektas. The multiple traveling salesman problem : an overview of formulations and solution procedures. *Omega*, 34(3) :209 – 219, 2006.

[7] N. Belacel. *Méthodes de classification multicritère, Chapitre 2*. PhD thesis, ULB, 2000.

[8] M. Bellmore and G.L. Nemhauser. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4) :393–410, Nov 1968.

[9] M. C. Bouzid and Y. Gahgouhi. Conception d'un outil d'analyse de la performance du réseau gsm de wta. Technical report, USTHB, 2009.

[10] J. Bramel and D. Simchi-levi. A location based heuristic for general routing problems. *Operations Research*, 43 :649–660, 1993.

[11] N. Chandran, T.T. Narendran, and K. Ganesh. A clustering approach to solve the multiple travelling salesmen problem. *International Journal of Industrial and Systems Engineering*, 1(3) :372 – 387, 2006.

[12] G.B. Dantzig, R. Fulkerson, and S. Johnson. The traveling salesman problem : A survey. *Operations Research*, 16(3) :538 – 558, 1968.

[13] G.B. Dantzig and J.H. Ramser. The truck dispatching problem. *Management Science*, 6(1) :80–91, Oct 1959.

[14] L.C.T. De Gomes and F.J. Von Zuben. Multiple criteria optimization based on unsupervised learning and fuzzy inference applied to the vehicle routing problem. *Journal of Intelligent and Fuzzy Systems*, 13(2-4) :143 – 154, 2002.

[15] L. Dekar and H. Kheddouci. A graph b-coloring based method for composition-oriented web services classification. In Aijun An, Stan Matwin, Zbigniew Ras, and

Dominik Slezak, editors, *Foundations of Intelligent Systems*, volume 4994 of *Lecture Notes in Computer Science*, pages 599–604. Springer Berlin / Heidelberg, 2008.

[16] H. Elghazel and K. Benabdeslem. Towards b-coloring of som. In *MLDM*, pages 322–336, 2009.

[17] H. Elghazel, K. Benabdeslem, and A. Dussauchoy. Constrained graph b-coloring based clustering approach. In *DaWaK*, pages 262–271, 2007.

[18] H. Elghazel, V. Deslandres, M.-S. Hacid, A. Dussauchoy, and H. Kheddouci. A new clustering approach for symbolic data and its validation : Application to the healthcare data. In *ISMIS*, pages 473–482, 2006.

[19] H. Elghazel, H. Kheddouci, V. Deslandres, and A. Dussauchoy. A partially dynamic clustering algorithm for data insertion and removal. In *Discovery Science*, pages 78–90, 2007.

[20] H. Elghazel, H. Kheddouci, V. Deslandres, and A. Dussauchoy. A graph b-coloring framework for data clustering. *Journal of Mathematical Modelling and Algorithms*, 7 :389–423, 2008. 10.1007/s10852-008-9093-x.

[21] H. Elghazel, T. Yoshida, V. Deslandres, M.-S. Hacid, and A. Dussauchoy. A new greedy algorithm for improving b-coloring clustering. In *GbRPR*, pages 228–239, 2007.

[22] H. Elghazel, T. Yoshida, and M.-S. Hacid. An integrated graph and probability based clustering framework for sequential data. In *Discovery Science*, pages 246–258, 2008.

[23] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, (7) :179–188, 1936.

[24] E. Fix and J.L. Hodges. Discriminatory analysis, nonparametric discrimination : Consistency properties. Technical Report 4, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[25] D. Gaceb, V. Eglin, F. Lebourgeois, and H. Emptoz. Implication de la b-coloration de graphes pour la reconnaissance automatique du type de document. In *Actes du dixième Colloque International Francophone sur l'Écrit et le Document*, 2009.

[26] M. Gaertler. Clustering. In *Network Analysis'04*, pages 178–215, 2004.

[27] B. Gavish and K. Srikanth. An optimal solution method for large-scale multiple traveling salesmen problems. *Operations Research*, 34(5) :698 – 717, 1986.

[28] S. Ghafurian and N. Javadian. An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems. *Applied Soft Computing*, 11(1) :1256 – 1262, 2011.

[29] G. Gutin. *Handbook of Graph Theory*, chapter Traveling Salesman and Related Problems. CRC Press, 2003.

[30] D. A. Holton and J. Sheehan. *The Petersen Graph*, chapter The Petersen Graph ; Graph colourings, page 41. Australian Mathematical Society, 1993.

[31] C. Hsu, M. Tsai, and W. Chen. A study of feature-mapped approach to the multiple travelling salesmen problem. In *IEEE International Sympoisum on Circuits and Systems, 1991.*, pages 1589 – 1592. IEEE Computer Society Press, 1991.

[32] R. W. Irving and D.F. Manlove. The b-chromatic number of a graph. *Discrete Applied Mathematics 91(1-3)*, pages 127–141, 1999.

[33] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations : Proc. of a Symp. on the Complexity of Computer Computations*, pages 85–103. The IBM Research Symposia Series, NY : Plenum Press, 1972.

[34] G. Laporte. The traveling salesman problem : An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2) :231 – 247, 1992.

[35] G. Laporte. The vehicle routing problem : An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3) :345 – 358, 1992.

[36] G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet. Classical and modern heuristics for the vehicle routing problem. *Les cahiers du GERAD*, 1999.

[37] G. Laporte and Y. Nobert. A cutting planes algorithm for the m-salesmen problem. *The Journal of the Operational Research Society*, 1980.

[38] S. Lin and B.W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2) :498–516, 1973.

[39] W. Liu, S Li, F. Zhao, and A. Zheng. An ant colony optimization algorithm for the multiple traveling salesmen problem. In *4th IEEE Conference on Industrial Electronics and Applications*, pages 1533 – 1537. IEEE Computer Society Press., 2009.

[40] J. B. MacQueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[41] E. Malaguti. The vertex coloring problem and its generalizations. *4OR*, 2008.

[42] D. Maquin. Eléments de théorie des graphes. INPL, Metz, 2003.

[43] S. Martello and P. Toth. *Knapsack problems : algorithms and computer implementations*, page 240. John Wiley and Sons, 1990.

[44] P. Miliotis. Using cutting planes to solve the symmetric travelling salesman problem. *Mathematical Programming*, 15 :177–188, 1978.

[45] C.E. Miller, A.W. Tucker, and R.A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7 :326–329, Oct 1960.

[46] A. Modares, S. Somhom, and T. Enkawa. A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems. *International Transactions in Operational Research*, 1999.

[47] J. Muralidharan, A. Gunasekaran, S.P. Nachiappan, and A. Nivin Kumar. Efficient mechanism development for multirobot coordination. *International Journal of Industrial and Systems Engineering*, 3(2) :149 – 161, 2008.

[48] R. Nallusamy, K. Duraiswamy, R. Dhanalaksmi, and P. Parthiban. Optimization of non-linear multiple traveling salesman problem using k-means clustering, shrink wrap algorithm and meta-heuristics. *International Journal of Nonlinear Science*, 2009.

[49] P. Nemery de Belleveaux. Multicriteria clustering, 2006. ULB.

[50] H. Ogino and T. Yoshida. Toward improving re-coloring based clustering with graph b-coloring. In Byoung-Tak Zhang and Mehmet Orgun, editors, *PRICAI 2010 : Trends in Artificial Intelligence*, volume 6230 of *Lecture Notes in Computer Science*, pages 206–218. Springer Berlin / Heidelberg, 2010.

[51] M. Petkovsek and T. Pisanski. Combinatorial interpretation of unsigned stirling and lah numbers. In *Preprint Series*.

[52] J. Riordan. *Introduction to Combinatorial Analysis*, page 43. Courier Dover Publications, 1958, reissue 1980, reprinted 2002.

[53] A. Russell. An effective heuristic for the m-tour traveling salesman problem with some side conditions. *Operations Research*, 1977.

[54] J.L. Ryan, T.G. Bailey, J.T. Moore, and W.B. Carlton. Reactive tabu search in unmanned aerial reconnaissance simulations. In *Proceedings of the 30th conference on Winter simulation*.

[55] A. Schrijver. *Handbook of Discrete Optimization*, volume 12, chapter On the history of combinatorial optimization (till 1960), pages 1 – 68. Elsevier, 2005.

[56] A. Singh and A. Baghel. A new grouping genetic algorithm approach to the multiple traveling salesperson problem. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 13 :95–101, 2009.

[57] M.M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 1987.

[58] S. Somhom, A. Modares, and T. Enkawa. Competition-based neural network for the multiple travelling salesmen problem with minmax objective. *Computers and Operations Research*, 1999.

[59] C. Song, K. Lee, and W.D. Lee. Extended simulated annealing for augmented tsp and multi-salesmen tsp. In *Proceedings of the International Joint Conference on Neural Networks, 2003*.

[60] H. Steinhaus. Sur la division des corps matériels en parties. *Bull. Acad. Polon. Sci*, 1 :801–804, 1956.

[61] L. Tang, J. Liu, A. Rong, and Z. Yang. A multiple traveling salesman problem model for hot rolling scheduling in shanghai baoshan iron and steel complex. *European Journal of Operational Research*, 2000.

[62] L. Tian-long and L. Yong-zai. Constrained multiple traveling salesman problem based on self-organizing optimization algorithm. *Journal of Computer Applications*, 2010.

[63] P. Toth and D. Vigo. *The vehicle routing problem*, chapter 1. SIAM, 2002.

[64] E. Wacholder, J. Han, and R.C. Mann. A neural network algorithm for the multiple traveling salesmen problem. *Biological Cybernetics*, 1989.

[65] Z. Xu and B. Rodrigues. A 3/2-approximation algorithm for multiple depot multiple traveling salesman problem. In Haim Kaplan, editor, *Algorithm Theory - SWAT 2010*, volume 6139 of *Lecture Notes in Computer Science*, pages 127–138. Springer Berlin / Heidelberg, 2010.

[66] S. Yadlapalli, W.A. Malik, S. Darbha, and M. Pachter. A lagrangian-based algorithm for a multiple depot, multiple traveling salesmen problem. *Nonlinear Analysis : Real World Applications*, 10(4) :1990 – 1999, 2009.

[67] Z. Yu, L. Jinhai, G. Guochang, Z. Rubo, and Y. Haiyan. An implementation of evolutionary computation for path planning of cooperative mobile robots. In *Intelligent Control and Automation, 2002. Proceedings of the 4th World Congress on ICA*.

[68] T. Zhang, W.A. Gruver, and M.H. Smith. Team scheduling by genetic search. In *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials, 1999*.

[69] B.S.E. Zoraida, M. Arocj, and R. Ponalagusamy. Dna algorithm employing temperature gradient for multiple traveling salesperson problem. *International Journal of Computer Applications*, 2010.

Résumé Le problème du voyageur de commerce multiple (m-TSP) consiste en l'optimisation de la visite d'un ensemble de villes par une flotte de véhicules basés à un dépôt. Ce problème qui est NP-difficile a mené à la conception de méthodes de résolution exactes et approchées. Les heuristiques de routage s'inscrivent généralement dans la classe des méthodes à deux phases : chercher une tournée globale puis la séparer en m tournées *ou* partitionner les villes à visiter puis chercher une tournée pour chacun des véhicules. En nous inscrivant dans cette dernière approche, nous présentons l'application inédite au m-TSP d'une méthode de clustering basée sur la b-coloration. Cette application a pour but de développer une heuristique efficace pour le m-TSP ou l'une de ses variantes en générant un clustering intéressant des villes à visiter. Nous reprenons les principaux travaux effectués dans les domaines du m-TSP et de la méthode et nous révélons une manière de dénombrer le nombre de routages possibles dans un m-TSP asymétrique. Nous présentons ensuite notre approche, nous la testons et nous l'évaluons sur des instances connues dans la littérature. Enfin, nous dévoilons une perspective intéressante pour la résolution d'une variante du m-TSP par notre application.
Mots clé : Problème du voyageur de commerce multiple, clustering par la b-coloration, combinatoire énumérative.

Abstract The multiple traveling salesman problem (m-TSP) consists in optimizing the visit of a set of cities by a fleet of vehicles based at a depot. This problem, which is NP-hard, has led to the design of several exact and heuristic methods. Most of the routing heuristics fall into the class of two phases methods : find one global route then split it into m routes *or* find a partition of the cities then a route for every vehicle. According to the latter approach, we present a new application to the m-TSP of a b-coloring based clustering method. Our goal is to design an efficient heuristic for the m-TSP or a variation by generating an interesting clustering of the cities that are to be visited. We review the main results in the fields of the m-TSP and the b-coloring clustering method and we present a way for counting the number of possible routings in the asymmetric m-TSP. Our approach is then presented, tested and evaluated against benchmarks. Finally, we reveal an interesting perspective for solving a variation of the m-TSP by our approach.
Key words : Multiple traveling salesman problem, b-coloring based clustering, enumerative combinatorics.