Université de M'sila

FACULTÉ DES MATHÉMATIQUES ET INFORMATIQUES

Département de Mathématiques

Mémoire de fin d'étude

Présenté pour l'obtention du diplôme de **Master**

Domaine :Mathématiques et Informatique

Filière :Mathématiques

Spécialité :Mathématiques Discrètes

Par

Barka Meriem

sujet

Le problème du voyageur de commerce: solution exacte et approchée

Président

Promotion: 2014/2015

Remerciements

Je remercie tout d'abord mon Dieu qui m'a donné la force, la volonté et le courage pour terminer ce modeste travail.

Je remercie mon encadreur **Professeur Belouadah Hocine**, pour la confiance qu'il m'a témoignée en me proposant ce sujet, ses encouragements et sa patience.

mes respects et ma gratitude vont également aux membres du jury qu'ils m'ont fait, en acceptant de juger ce modeste travail.

Je ne peux pas clôturer mes remerciements sans se retourner vers les êtres qui me sont les plus chers ; ma famille qui ont eu un rôle essentiel et continu dans ma réussite.

En fin, on remercie tous ceux qui, de prés ou de loin, ont contribué à la réalisation de ce travail, et surtout **Seghiri Karima** qui l'aide moi dans mon travail.

Merci

Table des matières

Introduction générale **1**

1 Le problème du voyageur de commerce **3**

1.1 Introduction 3

1.2 L'optimisation combinatoire 3

1.3 Complexité des problèmes 4

1.3.1 Le problème de décision 4

1.3.2 Le problème d'optimisation 4

1.4 Quelques problèmes NP-Difficiles 5

1.4.1 Le problème de coloration d'un graphe 5

1.4.2 Le problème du voyageur de commerce 5

1.4.3 Le problème du Postier Chinois 5

1.5 Le problème du voyageur de commerce (PVC) 5

1.5.1 Historique 6

1.5.2 Champs d'applications du PVC 7

1.5.3 Le PVC comme un problème de théorie du graphe 7

1.5.4 Le problème du voyageur de commerce symétrique et asymétrique . . 9

1.5.5 Formulation mathématique du PVC 9

1.5.6 Cas généraux d'un PVC 10

1.5.7 Cas particuliers d'un PVC 12

1.5.8 Le PVC et la notion de voisinage 13

1.6 Conclusion 15

2 Méthodes de solution pour le PVC **16**

2.1 Introduction 16

2.2 Rappel 16

2.3 La méthode par séparation et évaluation (Branch and Bound) 17

2.3.1 Application de la méthode B&B sur le PVC 19

2.4 La méthode descente et descente stochastique 21

2.5 La méthode recuit simulé 22

2.5.1 Application de la méthode recuit simulé sur le PVC 23

2.6 La méthode tabou 24

2.6.1 Application de la méthode tabou sur le PVC 26

2.7 La méthode Kangourou 29

2.7.1 Application de la méthode Kangourou sur le PVC 30

2.8 Conclusion 33

3 Etude de cas **34**

3.1 Introduction 34

3.2 Langage de programmation 34

3.3 Environnement de développement 35

3.4 La méthode descente simple 35

3.5 Simulation numérique 36

3.5.1 Génération des data de PVC 36

3.5.2 Discussion les résultats numériques 41

3.6 Conclusion 42

Conclusion générale **43**

Bibliographie **44**

Annexe **46**

Introduction générale

.

Dans ce mémoire on s'intéresse spécialement à l'un des problèmes célèbre d'optimisation combinatoire connu sous le nom: Le Problème du Voyageur de Commerce (PVC), en anglais "Traveling Salesman Problem". Et ses méthodes de solution: Il y'à n villes toutes doivent être visité par le voyageur chacune une et une seule fois en commençant par une des villes et on y revenant. Les distances entre les villes sont connues d'avance. Le but est de déterminer la tournée dont la distance est la plus courte que possible. L'intérêt de l'étude de ce fameux problème vient du fait que la majorité des chercheurs dans ce domaine utilise ce problème comme un benchmark pour juger la performance des méthodes de solutions proposées.

Dans le premier chapitre on commence par l'étude de la complexité des problèmes d'optimisation combinatoires et les problèmes de décision. Puis on cite quelques problèmes NP-Difficiles. Puis on se concentre à l'un d'entre eux c'est le PVC. On définit ce dernier et on donne des cas généraux et des cas particuliers de ce problème après avoir montré son extrême difficulté lorsqu'il devient de grande taille. Ensuite on définit la construction d'un voisinage pour une Métaheuristique qui est un facteur déterminant pour son efficacité. Pour cela on définit les voisinages λ-Opt et on choisi le voisinage 2-Opt pour le PVC.

Dans le chapitre deux: Après avoir défini et donne les mécanismes de la méthode par séparation et évaluation (Branch and Bound) on montre comment on l'utilise comme méthode de solution exacte pour résoudre le PVC. Puisque celle-ci, comme d'autres méthodes exactes (telle que la programmation dynamique) ont prouvés leurs échecs pour les cas de grandes tailles, une série de Métaheuristiques a été définit et appliqué au PVC pour déterminer une solution locale, citant: Méthode descente, descente stochastique, recuit simulé et méthode tabou.

Pour s'échapper des inconvénients de ces méthodes (stagner dans un même minimum local) une méthode appelle méthode Kangourou a été proposé. Celle-ci évité la recherche par exploration et préfère l'exploitation afin de proposer d'autres solutions admissibles qui peuvent donner des minimums locaux meilleurs.

Finalement dans le dernier chapitre: Une étude numérique est donnée pour appliquer l'algorithme descente simple au PVC. La méthode a été programmée en utilisant le langage Java.

Chapitre 1

Le problème du voyageur de commerce

1.1 Introduction

Dans ce chapitre, nous essayons d'aborder quelques notions sur l'optimisation combinatoire, ainsi on va parler sur un problème d'optimisation combinatoire, c'est le problème du voyageur de commerce, qui est aujourd'hui un problème bien connu et à la fin de ce chapitre on a défini la notion de voisinage. On choisi le voisinage 2-Opt pour le problème du voyageur de commerce.

1.2 L'optimisation combinatoire

L'optimisation combinatoire est le domaine des mathématiques discrètes qui traite la résolution du problème suivant :

Soit S un ensemble de solutions admissibles. Soit f une fonction permettant d'évaluer chaque solution admissible ($f : S \longrightarrow \mathbb{R}$). Il s'agit de déterminer une solution s^* appartenant à S qui minimise ou maximise f. L'ensemble S des solutions admissibles est supposé fini et est en général défini par un ensemble C de contraintes.

$f(s^*) = \max_{s \in S} f(s)$ En cas de problème de maximisation.

$f(s^*) = \min_{s \in S} f(s)$ En cas de problème de minimisation.

1.3 Complexité des problèmes

La théorie de la complexité se concentre sur les problèmes de décision [10]. Elle permet de mesurer la complexité des problèmes en considérant le temps de calcul nécessaire et l'espace mémoire requis pour en résoudre. En effet, on distincte la différence entre le problème de décision et le problème d'optimisation.

1.3.1 Le problème de décision

Le problème de décision est un problème dont les résultats ne peuvent prendre que l'une des deux réponses « **oui** » ou « **non** ».

1.3.2 Le problème d'optimisation

Le problème d'optimisation est un problème pour lequel on doit cherche à déterminer une solution qui optimise un critère. À chaque problème d'optimisation on peut associer un problème de décision.

Définition 1.3.1 *: Étant donné un problème d'optimisation combinatoire:*

Trouver $s^* \in S$ tel que $f(s^*) = \min_{s \in S} f(s)$ $(\max_{s \in S} f(s))$, et un nombre A, on définit « le problème de décision associé» comme suit: Existe-t-il $s^* \in S$ tel que $f(s^*) \leq A$ $(f(s^*) \geq A)$? [17].

Définition 1.3.2 *: Un problème de décision appartient à la classe P, s'il peut être résolu par un algorithme polynomiale.*

Définition 1.3.3 *: Un problème de décision appartient à la classe NP s'il peut être résolu par un algorithme non déterministe polynomiale.*

Définition 1.3.4 *: Un problème de décision est NP-Complet s'il est dans la classe NP et si on peut le ramener, par une transformation polynomiale, à un autre problème de la classe NP.*

Définition 1.3.5 *: Un problème d'optimisation combinatoire est dit NP-Difficile si le problème de décision associé est NP-Complet.*

1.4 Quelques problèmes NP-Difficiles

1.4.1 Le problème de coloration d'un graphe

En théorie des graphes, colorer un graphe G signifie attribuer une couleur à chacun de ses sommets (*arêtes*) de manière que deux sommets adjacents (reliés par une *arête*) ne portent pas la même couleur (ayant une extrémité en commun soient de couleur différente). Le plus petit nombre de couleurs nécessaires pour colorer les sommets d'un graphe G est appelé «*le nombre chromatique*» (Le plus petit nombre de couleurs nécessaires pour colorer les *arêtes* d'un graphe G s'appelle « *l'indice chromatique* ») de G est noté $\chi(G)$ $(q(G))$.

1.4.2 Le problème du voyageur de commerce

Étant donné un ensemble de n villes séparées par des distances données, le problème du voyageur de commerce consiste à trouver la plus courte tournée reliant toutes les villes chacune une seule fois.

1.4.3 Le problème du Postier Chinois

En théorie des graphes et en algorithmique le problème de Postier Chinois consiste à trouver un plus court chemin dans un graphe connexe (entre chaque deux sommets il ya une chaîne) non orienté (c'est à dire un *arc* peut être parcourue indifféremment dans les deux sens), qui passe au mois une fois par chaque *arête* du graphe et revient à son point de départ.

1.5 Le problème du voyageur de commerce (PVC)

Le problème du voyageur de commerce (en anglais, Traveling Salesman Problem **TSP**), doit visiter n villes en passant par chaque ville exactement une seule fois, telle que on commence par n'importe quelle ville et on termine en retournant à la ville de départ. Les distances entre les villes sont connes d'avance. On doit donc minimiser la distance parcourue, on peut remplacer la notion de la distance par le temps ou par le coût. Ce problème, de part sa simplicité, a été toujours considérer comme un benchmark pour l'étude de l'efficacité des

méthodes de résolution de problèmes d'optimisation combinatoires et surtout les méthodes approchées connue sous le nom de Métaheuristiques.

Remarque 1.5.1 *: Le voyageur de commerce peut être remplacer par un véhicule, postier, médecin, appareil ou machine,.....et les villes peuvent être remplacer par des clients, machines, hôpitaux, malades, foyers* [1].

1.5.1 Historique

18^{ieme}siècle :

Les premières approches mathématiques exposées pour le problème du voyageur de commerce ont été traitées au 18^{ieme} siècle par les mathématiciens Sir William Rowan Hamilton et Thomas Penyngton Kirkman. Hamilton en a fait un jeu: Hamilton's Icosian game. Les joueurs devaient réaliser une tournée passant par 20 points en utilisant uniquement les connections prédéfinies.

Années1930 :

Le PVC est traité plus en profondeur par Karl Menger à Harvard. Il est ensuite développé à Princeton par les mathématiciens Hassler Whitney et Merril Flood. Une attention particulière est portée sur les connections par Menger et Whitney ainsi que sur la croissance du PVC [20].

1954 :

Solution du PVC pour 49 villes par Dantzig, Fulkerson et Johnson par la méthode du cutting-plane [8].

1975 :

Solution pour 100 villes par Camerini, Fratta and Maffioli [7].

1987 :

Solution pour 532, puis 2392 villes par Padberg et Rinaldi [15].

1998 :

Solution pour les 13509 villes des Etats-Unis.

2001 :

Solution pour les 15112 villes d'Allemagne par Applegate, Bixby, Chvàtal et Cook des universités de Rice et Princeton [2].

Remarque 1.5.2 *: Le PVC appartient à la classe des problèmes NP-Difficiles* [10]*, où l'existence d'un algorithme polynomiale reste inconnue. Un calcul rapide de la complexité montre qu'elle est* $O(n!)$ *où* n *est le nombre de villes. Donc le nombre de tournées possibles est* $(n-1)!/2$*. Si on suppose que le temps pour évaluer une tournée est* $1\mu s$*, le tableau suivant montre l'explosion combinatoire du PVC.*

Nombre de villes	*Nombre de tournées possibles*	*Temps de calcul*
5	12	12 μs
10	181440	0.18 *second*
15	43 *milliards*	12 *heurs*
20	60×10^{15}	1928 *ans*
25	310×10^{21}	9.8 *milliards d'années*

Tableau 1.1*: Nombre de tournées possibles et temps de calcul estimé* [4]

Remarque 1.5.3 *: Le problème de décision associé au problème du voyageur de commerce peut être comme suit:*

Étant donné un ensemble de n *villes, d'inter distances est* d_{ij} *et un nombre* A*. Le problème est de savoir s'il existe une tournée de coût inférieur ou égal à* A [17]*.*

1.5.2 Champs d'applications du PVC

Le PVC fournit un exemple d'étude d'un problème NP-Difficile dont les méthodes de résolution peuvent s'appliquer à d'autres problèmes de mathématiques discrète. Néanmoins, il a aussi des applications directes, notamment dans les transports et la logistique. Par exemple, trouver le plus cour chemin pour les bus de ramassage scolaire ou, dans l'industrie, pour trouver la plus courte distance que devra parcourir le bras mécanique d'une machine pour percer les trous d'un circuit imprimé (les trous représentent les villes).

1.5.3 Le PVC comme un problème de théorie du graphe

Soit $G = (X, U)$, un graphe dans lequel l'ensemble X des sommets représente les villes à visiter, ainsi que la ville de départ de la tournée, et U l'ensemble des *arêtes* de G,

représente les parcours possibles entre les villes, un cycle à n sommets est alors l'ensemble de n sommets $\{x_0, x_1, ..., x_n\}$ tels que $x_n = x_0$ et $(x_i, x_{i+1}) \in U$ pour $i = 0, 1, 2, ..., n$. À tout $arête(i, j) \in U$, on associe la distance de parcours d_{ij} de la ville i à la ville j. La longueur d'une tournée dans G est la somme des distances associe aux $arcs$ de cette tournée, dans la suite, on notera $n = |X|$ le nombre de sommets du graphe. Le but du problème du voyageur de commerce est déterminé le cycle hamiltonien (un cycle qui visite chaque sommet une et une seule fois) de plus courte longueur c'est -à-dire qui minimise la somme des distances [5].

Exemple 1.5.1 *: Un graphe à 4 sommets*

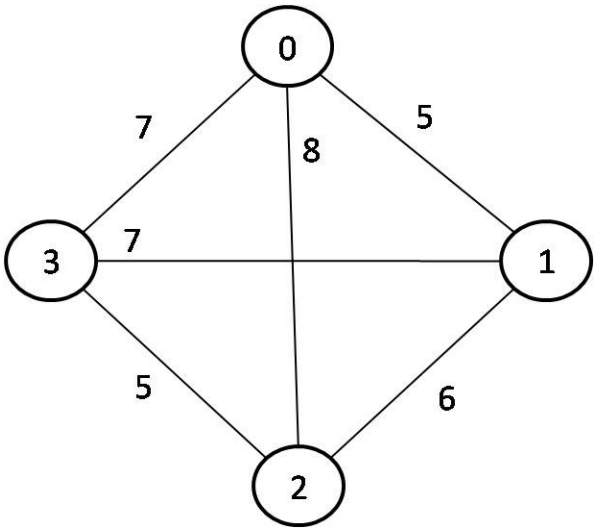


Figure 1.1*: Graphe à 4 sommets*

Résoudre le problème du voyageur de commerce revient à trouver dans ce graphe un cycle hamiltonien et qui soit de longueur minimale. Une solution à ce problème serait le cycle 0, 1, 2, 3 et 0, correspondant à une distance totale de 23 ($d_{01}+d_{12}+d_{23}+d_{30} = 5+6+5+7 = 23$).

Comme il existe un arc entre chaque paire de sommets, on dit que ce graphe est «complet». Tout graphe peut être représenter par une matrice telque: Les lignes représentent les sommets d'origine des $arcs$ et les colonnes représentent les sommets de destination, le coût sur chaque arc apparait à l'intersection de la ligne et de la colonne correspondantes.

Pour notre exemple, cette matrice est la suivante:

$$\begin{pmatrix} 0 & 5 & 8 & 7 \\ 5 & 0 & 6 & 7 \\ 8 & 6 & 0 & 5 \\ 7 & 7 & 5 & 0 \end{pmatrix}$$

Figure 1.2: La matrice associée au graphe

Dans cet exemple, le graphe est «non orienté», cela implique que la matrice soit symétrique. Cette symétrie n'est pas forcement respectée dans le cas d'un graphe orienté. Donc il existe deux catégories de ce problème: Le PVC symétrique et le PVC asymétrique.

1.5.4 Le problème du voyageur de commerce symétrique et asymétrique

Le PVC symétrique

Le problème du voyageur de commerce symétrique est le plus simple, car la distance entre deux villes quelconques i et j est égale à celle entre j et i.

Le PVC asymétrique

Le problème du voyageur de commerce asymétrique, car la distance entre deux villes i et j n'est pas forcement égale à celle entre j et i.

1.5.5 Formulation mathématique du PVC

Soit $G = (X, U)$ un graphe, où X est l'ensemble des sommets et U l'ensemble des *arêtes* (*arcs* si le graphe est orienté). Dantzig et al en 1954 ont proposé une formulation mathématique du PVC où ils définissent:

x_{ij} : Une variable binaire qui prend la valeur de 1 si l'$arc(i, j)$ est utilisé dans la tournée et 0 sinon;

c_{ij} : Le coût de parcours de l'$arc(i, j)$;

n : Nombre de sommets;

Ainsi nous avons la formulation suivante:

$$Minimiser \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \quad (1.1)$$

Les contraintes sont les suivantes:

$$\sum_{i \in X} x_{ij} = 1; \qquad \forall j \in X \quad (1.2)$$

$$\sum_{j \in X} x_{ij} = 1; \qquad \forall i \in X \quad (1.3)$$

$$\sum_{i,j \in s} x_{ij} \leq |s| - 1; \quad \forall s \subset X; 2 \leq |s| - 2 \quad (1.4)$$

$$x_{ij} \in \{0, 1\}; \forall i \in X, \forall j \in X \quad (1.5)$$

La (1.1) formule l'objectif du PVC qui est la minimisation de la longueur totale parcourue par le voyageur. Les contraintes (1.2) et (1.3) assurent que le voyageur passe une et une seule fois par chaque sommet (ville). La (1.4) garante l'élimination de la formation de sous tour [1].

1.5.6 Cas généraux d'un PVC

Le Problème du Voyageur de Commerce Généralisé (PVCG)

Le PVCG (Generalized Traveling Salseman Problem ou GTSP) peut être décrit de la façon suivante: Soit $G = (V, W)$ un graphe complet non orienté, $V = \{v_1, v_2, ..., v_n\}$ un ensemble de villes, $W = \{w_1, w_2, ..., w_m\}$ un ensemble de groupes, avec $0 < m \leq n$. Chaque ville $v_i \in V$ appartient à un et un seul groupe (les groupes sont disjoints deux à deux, i. e. Pour $i \neq j, w_i \cap w_j = \emptyset$. et $w_1 \cup w_2 \cup ... \cup w_m = V$). Les coûts de transport dans notre travail sont notés d_{ij} pour $(v_i, v_j) \in V$. L'objectif est de construire une tournée qui visite exactement une fois chaque groupe tout en minimisant la somme des coûts. Le PVCG est un problème NP-Difficile au sens fort, puisque ce problème est une généralisation du PVC.

En effet, le cas spécial dans le que $m = n$ (une ville par groupe) est le PVC: Le problème est de trouver une tournée qui visite chaque ville une seule fois en minimisant le coût [6].

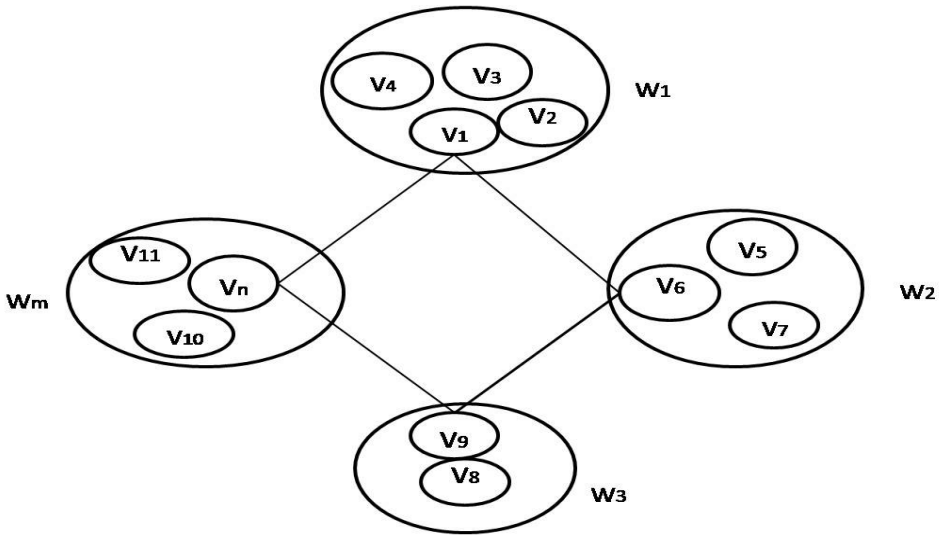


Figure 1.3: Le PVCG

Le problème de tournées de véhicules

Le problème de tournées de véhicules (VRP: Vehicle Routing Problem) est une généralisation du PVC, a été introduit pour la première fois en 1959 par Dantzig et Ramser, est un problème d'optimisation combinatoire, il appartient à la classe des problèmes NP-Difficiles [18].

Le principe du problème de tournées de véhicules consiste à trouver les meilleures tournées à parcourir par une flotte de m véhicules afin visiter n clients (villes) pour des raisons de collecte ou de livraison. Tous les véhicules commencent et terminent leurs tournées à un ville central appelle Dépôt. Chaque client (ville) est affecté à une tournée et doit être visité une seule fois par un seul véhicule. La capacité de transport de ce dernier pour une tournée est limitée et elle ne doit pas être dépassée. Toutes les tournées proposées doivent respecter les différentes capacités des véhicules et satisfaire les quantités demandées par les clients (villes) ou à livrer à ces derniers.

L'objectif principal de problème de tournées de véhicules est de minimiser le coût (distance) totale de parcours des tournées par un nombre minimum de véhicules

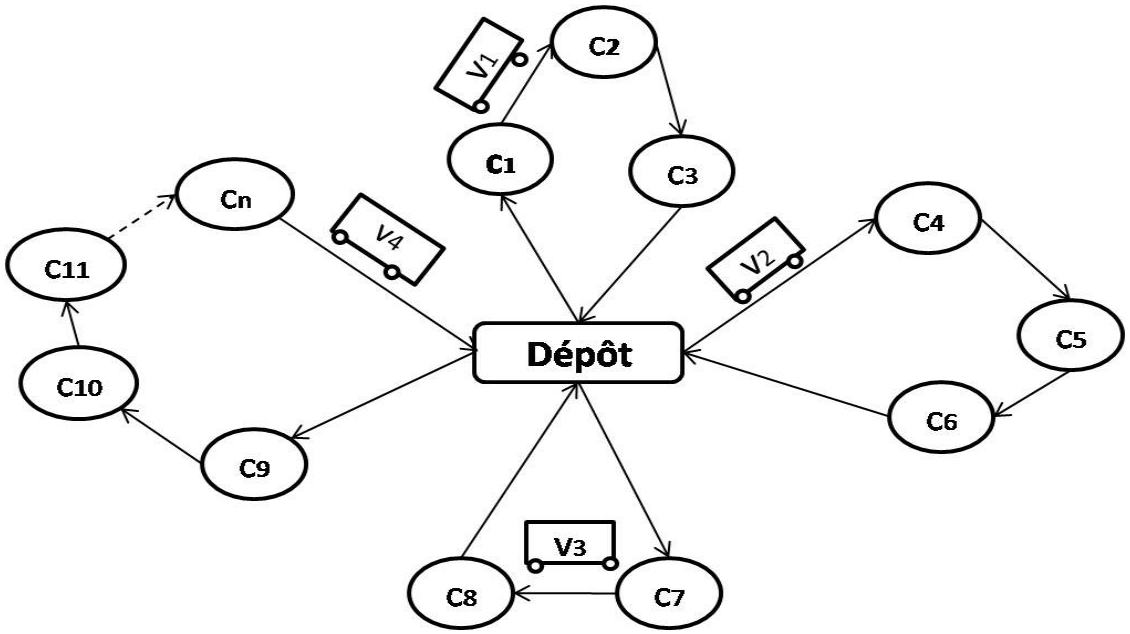


Figure 1.4: Exemple de tournées avec 4 véhicules

Le problème des m voyageurs de commerce

Le problème des m voyageurs de commerce (m-TSP) est une généralisation du PVC où on désire construire m cycles hamiltoniens de coût total minimal ayant un sommet commun (le dépôt). Ce problème peut être considéré comme une version simplifié du VRP où la capacité de chaque véhicule est supérieure à la totalité de la demande (est supposé infini) et où les m véhicules sont utilisés [16].

1.5.7 Cas particuliers d'un PVC

Le PVC métrique

On peut, par réduction au problème de trouver un cycle hamiltonien, que toute ρ-approximation du problème du voyageur de commerce est NP-Difficile ($\rho \geq 1$).

On s'intéresse ici au cas particulier où la fonction de coût est métrique, autrement dit, au cas où elle vérifier l'inégalité triangulaire: $\forall x, y, z \in X,\ f(x, y) + f(y, z) \geq f(x, z)$ [13].

Le PVC euclidien

Le problème du voyageur de commerce euclidien peut être devient comme suit: Les villes sont représentées dans un espace bidimensionnel, la distance entre deux villes quelconques X, Y est calculée à partir des absciesses et des ordonnées des deux villes (distance=$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$) où x_1 et y_1 sont les coordonnées de la ville X et x_2 et y_2 représente les coordonnées de la ville Y.

Les méthodes de résolution d'un problème d'optimisation combinatoire basé sur les méthodes de voisinage qui soit fondées sur la notion de voisinage. Nous allons donc introduire d'abord cette notion fondamentale ainsi quelques notions associées.

1.5.8 Le PVC et la notion de voisinage

Définition 1.5.1 *: Soit S l'ensemble des solutions admissibles d'un problème, on appelle voisinage toute application, à définir, de S dans l'ensemble $P(S)$ des parties de S. On appelle mécanisme d'exploration du voisinage toute procédure qui précise comment la recherche passe d'une solution $s \in S$ à une solution $s' \in V(s)$. Une solution s est un optimum (minimum) local par rapport au voisinage V si $f(s) \leq f(s')$ pour toute solution $s' \in V(s)$* [11].

Une méthode typique de voisinage débute avec une solution initiale, et réalise ensuite un processus itératif qui consiste à remplacer la solution courante par l'un de ses voisins en tenant de la fonction de coût. Ce processus s'arrête et retourne la meilleure solution trouvée quand la condition d'arrêt est réalisée. Les méthodes de voisinage différent essentiellement entre elles par le voisinage utilisé et la stratégie de parcours de ce voisinage.

Les voisinages λ-Opt

Les voisinages λ-Opt ont été proposées par Lin et Kernighan pour le problème du voyageur de commerce, l'algorithme λ-Opt commence avec une tournée admissible donnée, cherche ensuite dans le voisinage de la solution courante défini par l'opération λ-Opt toute tournée améliorant la solution courante. À chaque étape de l'itération, l'algorithme examine, pour des valeurs croissantes de λ (à partir de 2) si l'échange de λ *arêtes* produit une tournée plus courte. L'algorithme continue ainsi jusqu'à ce qu'aucune amélioration ne soit plus possible.

L'opération λ-Opt consiste à supprimer λ *arêtes* et à reconnecter les segments restants par de nouveaux *arêtes*, en renversant si possible le sens de parcours d'un ou de plusieurs de ces segments. Plus la valeur de λ est grande, plus la solution finale est proche de l'optimum et plus le temps d'exécution devient élevé. En général on utilise des valeurs entiers de $\lambda \in \{2, 3, 4, 5\}$[21].

Pour le problème du voyageur de commerce on choisir l'algorithme 2-Opt.

La méthode 2-Opt

En optimisation, 2-Opt est un algorithme de recherche local proposé par Croes en 1958 pour résoudre le problème du voyageur de commerce en améliorant une solution initiale.

Si le graphe comporte n sommets, il ya donc n opérations du type échange de sommets consécutif et $n(n-3)/2$ opération 2-Opt.

L'algorithme 2-Opt

L'algorithme 2-Opt commence avec une tournée admissible donnée, ensuite elle cherche dans le voisinage de la solution courante toute tournée améliorant la solution actuelle. Le principe de ce algorithme est: À chaque étape de la procédure d'amélioration, on choisi deux paires d'*arêtes* non adjacence et on supprime pour les remplacer par nouvelle *arêtes*, donc on peut résumée les étapes de ce algorithme comme suit:

Étape 1: Prendre deux paires d'*arêtes* non adjacence (A,B) et (C,D) et on supprime

Étape 2: Lier les deux extrémités initiales des deux *arêtes* supprimées

Étape 3: Lier les deux extrémités finales des deux *arêtes* supprimées

Étape 4: Changer les directions des *arêtes* de l'extrémité finale du 1^{ere} *arête* à l'extrémité initiale du $2^{éme}$ *arête* supprimés.

Soit le cycle A, B, C, D, E et A

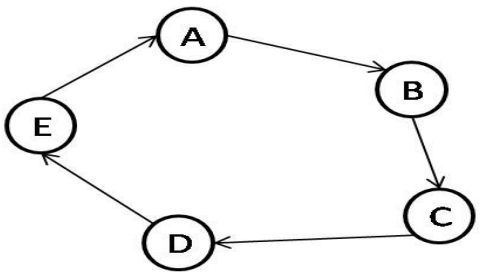


Figure 1.5: Un cycle

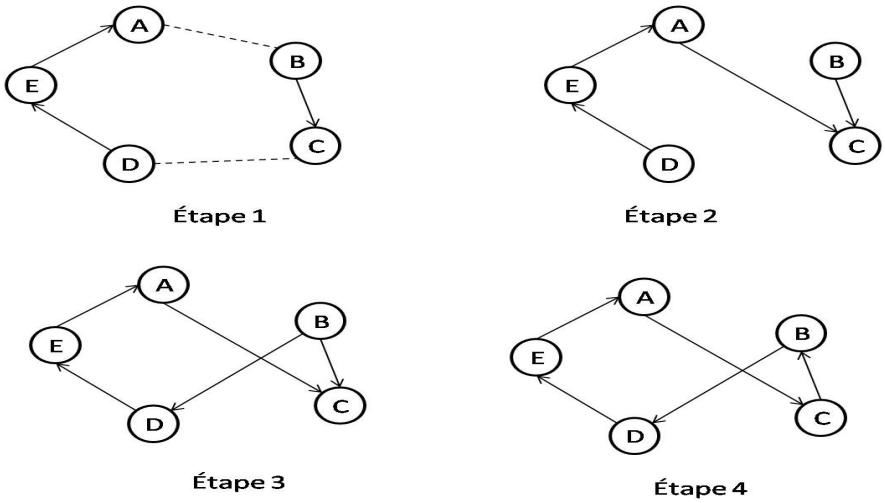


Figure 1.6: Le voisinage 2-Opt

1.6 Conclusion

Nous avons défini dans ce chapitre les problèmes d'optimisation combinatoire ainsi que nous avons étudié le problème du voyageur de commerce théoriquement et formellement. Ce problème appartient à la classe des problèmes NP-Difficiles.

Dans le chapitre qui suit, nous présentons les méthodes de résolution d'un problème d'optimisation combinatoire et dans un cas particulier le PVC.

Chapitre 2

Méthodes de solution pour le PVC

2.1 Introduction

Le but de ce chapitre est présenter deux grand classes de méthodes générales pour résoudre des problèmes d'optimisation combinatoire NP-Difficiles: Les méthodes exactes (séparation et évaluation) et les méthodes approchées (recuit simulé, tabou, Kangourou), nous essayons la solution de PVC par ces méthodes.

2.2 Rappel

La majorité des problèmes d'optimisation combinatoire, sont des problèmes NP-Difficiles et donc ne possédant pas à ce jour un algorithme efficace, i.e. de complexité polynomiale, valable pour toutes les données. Ceci a motivé les chercheurs à développer de nombreuses méthodes de solution en optimisation combinatoire.

La performance des méthodes de solution des problèmes est mesurée en termes de deux notions: Le temps d'exécution et la qualité des solutions fournies, ces méthodes appartiennent à deux grandes catégories: Les méthodes exactes et les méthodes approchées. Les méthodes exactes sont connues par le fait qu'elles garantissent l'optimalité de la solution, mais elles demandent des coûts de recherche (temps de calcul et espace mémoire), on a par exemple la méthode par séparation et évaluation et la programmation dynamique qui n'est pas notre but dans ce mémoire. Tandis que, les méthodes approchées sont connues par le

fait qu'elles ne garantissent pas l'optimalité de la solution (elles fournissent des solutions de bonne qualité et des fois optimales). Les méthodes approchées sont classeés en deux familles: Les heuristiques (sont des méthodes approchées spécifiques à un problème donné. Elle forme un ensemble de règles empiriques). Et les métaheuristiques (sont des méthodes approchées polyvalentes), elles peuvent être classées en deux catégories: Méthodes à base de solution unique: La méthode descente et descente stochastique sont considérés la base de Métaheuristiqes par exemple recuit simulé, tabou et Kangourou. Méthodes à base de population de solutions comme les algorithmes génétiques et colonies de fourmis qui ne sont pas notre but, la figure suivante présente les méthodes de solution d'un problème d'optimisation combinatoire:

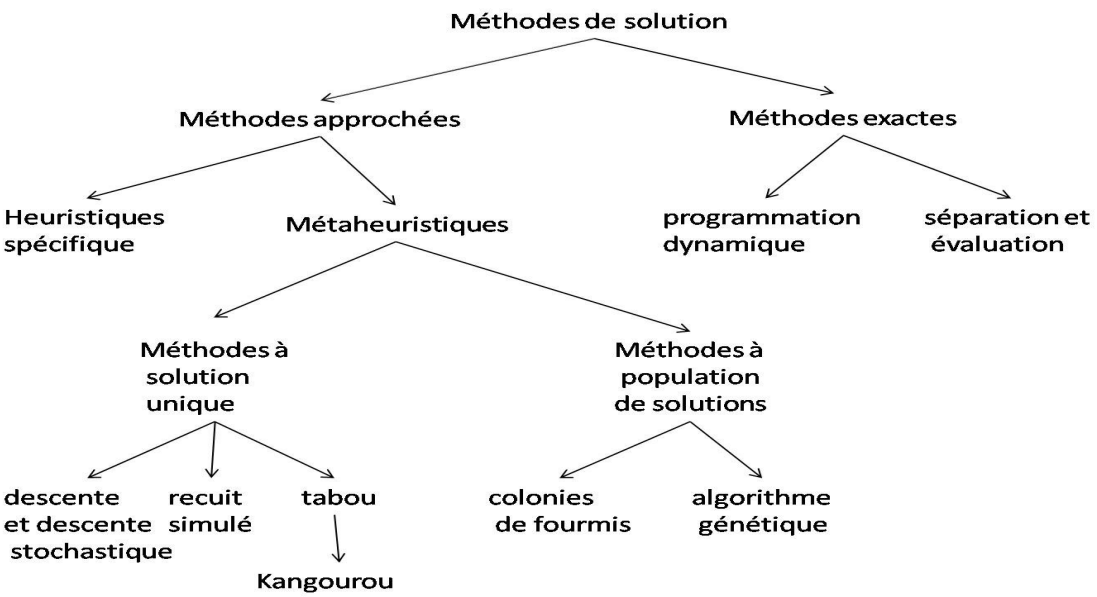


Figure 2.1: Méthodes de solution d'un problème d'optimisation combinatoire [4]

2.3 La méthode par séparation et évaluation (Branch and Bound)

La méthode par séparation et évaluation (nommée Branch and Bound en anglais) est notée B&B. C'est une méthode qui permet la résolution exacte de problèmes d'optimisation combinatoire dont on cherche à minimiser le coût de la recherche.

B&B propose un mécanisme de recherche très intelligent grâce à lequel elle permet une bonne exploitation de l'espace de recherche et l'aboutissement à la solution optimale plus rapidement que d'autres méthodes exactes. Le branch and bound est basé sur trois axes principaux [9]:

- La séparation
- L'évaluation
- La stratégie de recherche

1. **La séparation:**

   La séparation consiste à diviser le problème en certain nombre de sous problèmes. Ainsi, en résolvent tous les sous problèmes et en prenant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial. Ce principe de séparation peut être appliqué de manière récursive à chacun des sous problèmes, cela revient à construire un arbre permettant d'énumérer toutes les solutions.

2. **L'évaluation:**

   L'évaluation permet de réduire l'espace de recherche en éliminant quelques sous ensembles qui ne contiennent pas la solution optimale. L'objectif d'essayer d'évaluer l'intérêt de l'exploration d'un sous ensembles de l'arborescence. Le branch and bound ulilise une élimination de branches dans l'arborescence de recherche de la manière suivante: La recherche d'une solution de coût minimal, consiste à mémoriser la solution de plus bas coût rencontré pendant l'exploration, et à comparer le coût de chaque nœud par parcourus à celui de la meilleure solution. Si le coût du nœud considéré est supérieur au meilleur coût, on arrête l'exploration de la branche et tous les solutions de cette branche seront nécessairement de coût plus élevé que la meilleure solution déjà trouvée.

3. **La stratégie de recherche**

La largueur d'abord:

Cette stratégie favorise les nœuds les plus proches de la racine en faisant moins de séparation du problème initial. Elle est moins efficace que les deux autres stratégies ci-dessous.

La profondeur d'abord:

Cette stratégie cherche les sommets les plus éloignés de la racine.

La meilleure d'abord:

Cette stratégie consiste à explorer des sous problèmes possédant la meilleure valeur de coût. Elle permet aussi d'éviter l'exploration de tous les sous problèmes qui possèdent une mauvaise évaluation par rapport à la valeur optimale.

2.3.1 Application de la méthode B&B sur le PVC

Considérons le problème du voyageur de commerce représenté par le graphe ci-dessous $G(X, U, d)$ où $X = \{A, B, C, D, E\}$ l'ensemble des 5 villes. Les valeurs portées sur les *arêtes* représentent les distances entre les villes. Utiliser une méthode par séparation et évaluation qui détermine la tournée optimale dont la distance est le plut petite parmi les $(5-1)!/2 = 12$ tournées possibles.

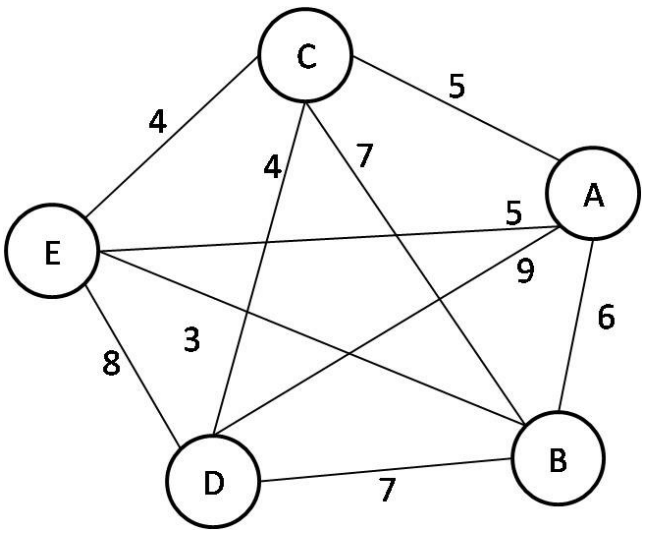


Figure 2.2: Exemple d'un graphe complet [14]

Cette graphe peut être bien entendu représenter par sa matrice D d'adjacence tel que $D[i, j]$ représente le coût sur l'$arc(i, j)$.

$$\begin{pmatrix} 0 & 6 & 5 & 9 & 5 \\ 6 & 0 & 7 & 7 & 3 \\ 5 & 7 & 0 & 4 & 4 \\ 9 & 7 & 4 & 0 & 8 \\ 5 & 3 & 4 & 8 & 0 \end{pmatrix}$$

Figure 2.3 : Matrice d'adjacence de graphe

Résolvons ce problème en commençant par exemple à partir du sommet E, on calcule la borne de la racine E c'est-à-dire la valeur de toutes les solutions incluant le sommet E, le prochain sommet du cycle que nous recherchons est soit C, A, B ou D. Pour chacune de ces solutions partielles, nous calculons une nouvelle borne.

L'idée de cette méthode est la suivante: Toute solution partielle dont le coût est plus grand que celui de la meilleure solution trouvée jusqu'à présent va être exclue de la recherche. Ce processus est continué jusqu'à-avoir une solution complète, si le coût de cette solution complète est inferieure à celui de la meilleure solution que nous avons déjà, alors nous remplaçons ce coût et le considèrent comme étant celui de la meilleure solution courante.

Lemme 2.3.1 *Soit le cycle hamiltonien suivant:* $x_1, x_2, ..., x_n, x_{n+1} = x_1$. *Donc quelque soit la solution, son coût est* $\geq \frac{1}{2}\sum_{i=1}^{n}(arc_{i_1} + arc_{i_2})$ *où* arc_{i_1}, arc_{i_2} *désignent deux arcs adjacents au sommet* i *ayant le plus petit coût* [22].

Appliquons cette borne sur le graphe ci-dessus. En commençant à partir de E, on aura donc: Le coût est $\geq \frac{1}{2}\{(3+4)+(4+4)+(5+5)+(3+6)+(4+7)\} = 22.5$.

Les prochains sommets dans le cycle peuvent être: C, A, B ou D. Pour chacune de ces solutions partielles, une borne est calculée. Par exemple pour C, la nouvelle borne est:

$\frac{1}{2}\{(4+3)+(4+4)+(5+5)+(3+6)+(4+7)\} = 22.5$.

Pour le sommet E, l'$arc(E, C)$ doit être pris, et pour le sommet C, l'$arc(C, E)$ doit être pris. La valeur 22.5 représente donc le plus petit cycle hamiltonien incluant les $arc(E, C)$ et (C, E).

Ceci est fait pour chacune des sommets, on obtient successivement 23 pour le sommet A, 22.5 pour le sommet B et 25 pour le sommet D. Parmi les quatre valeurs, nous allons

explorer le sommet ayant la plus petite valeur. Soit le sommet C, avoir d'explorer le second sommet ayant la plus petite borne, etc.

Il existe d'autres manières de procéder dans le choix de la prochaine solution partielle à explorer. Toutefois, chacune des ces méthodes a ses propres avantages et inconvénients.

Pour notre exemple, nous pouvons citer l'exploration en profondeur d'abord, ce processus est ensuite répéter. Quand une solution complète est trouvée on le mémorise, et si le nœud a une borne plus grande que cette valeur, on annule l'exploration de la branche.

Dans notre exemple, la première solution trouvée a pour coût 26. Donc l'arborescence obtenue est la suivante:

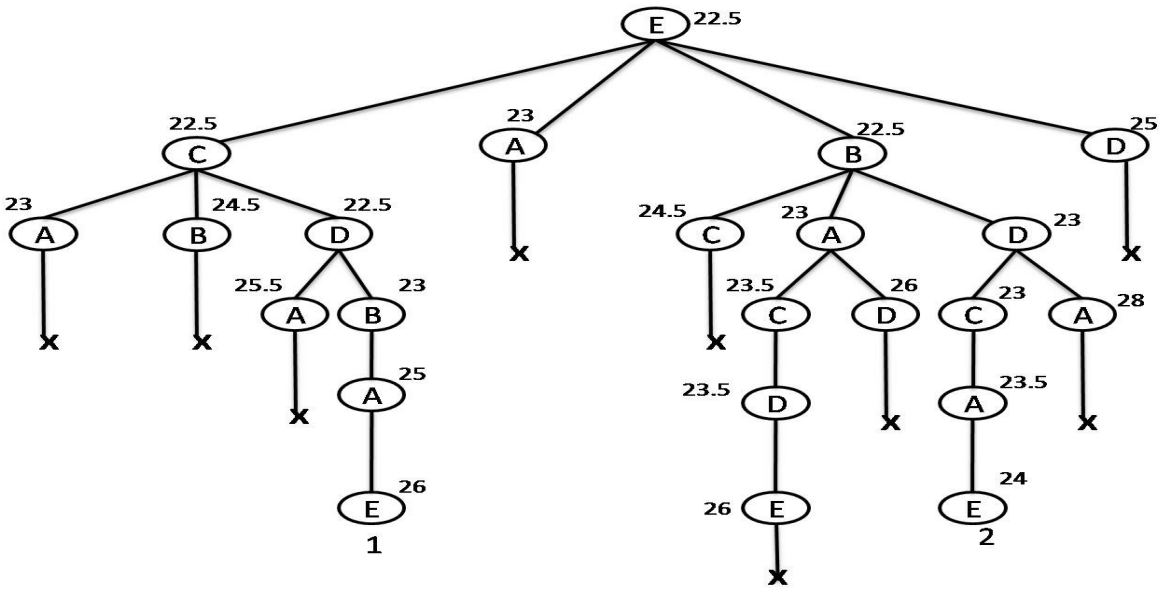


Figure 2.4: Résolution le PVC par la méthode B&B en utilise la stratégie profondeur d'abord

Donc la tournée $(EBDCA)$ est la solution optimale de distance totale 24 qui minimise la fonction f.

2.4 La méthode descente et descente stochastique

La méthode descente (recherche locale), appelée aussi l'amélioration itérative, la recherche locale constitue une arme redoutable pour attaquer des problèmes réputés très difficiles tels que le voyageur de commerce. Le principe de la méthode de descente consiste à partir d'une

solution s et à choisir une solution s' dans un voisinage de s, telle que s' améliore la recherche (généralement telle que $f(s') < f(s)$).

La plupart des métaheuristiques à base de solution unique sont des améliorations de la méthode descente stochastique (aléatoire). Qui consiste à faire une descente aléatoire à partir de plusieurs points choisis de façon aléatoire dans l'espace de recherche. Le principe de la méthode descente stochastique consiste à partir d'une solution s et à choisir une solution aléatoire s' dans un voisinage de s même si elle est mauvaise. Elle s'arrête quand la condition est satisfaire.

2.5 La méthode recuit simulé

La méthode recuit simulé (simulated annealing) a été inventé par les physiciens Kirkpatrick, Gellat et Vecchi en 1983. Elle permit la résolution de manière quasi optimale des PVC à 5000 sommets. Cette méthode tire son origine de la physique statique. Ce processus utilisé en métallurgie consiste à porter un matériau à température élevée, puis à le refroidir d'une manière contrôlée jusqu'à l'obtention d'une structure stable, dans ce processus, l'énergie du matériau est minimisée [3].

Le principe de cette méthode consiste à utiliser en métallurgie pour améliorer la qualité d'un métal solide par recherche d'un état d'énergie minimale qui correspond à une structure stable de ce métal. En partant d'une haute température à laquelle le solide devient liquide, la phase de refroidissement conduit à la matière liquide.

Donc une méthode de recuit simulé devient comme suit:

1) Choisir une température de départ T et une solution initiale s_0, $s \longleftarrow s_0$;

2) Générer une solution aléatoire dans le voisinage de la solution actuelle s', $s' \in V(s)$;

3) Calculer la variation de coût $\Delta f = f(s') - f(s_0)$;

4) Si $\Delta f \leq 0$, le coût diminue et on effectue la transformation améliorante;

5) Si $\Delta f > 0$, le coût remonte, c'est un rebond, qu'on va pénaliser d'autant plus que la température est basse et que Δf est grand. Une fonction exponentielle a les propriétés désirées. On calcule une probabilité d'acceptation $a = e^{-\Delta f/T}$, puis on tire au sort p dans

$[0, 1]$. Si $p \leq a$, la transformation est déclarée acceptée, bien qu'elle dégrade le coût, et on fait $s_0 \longleftarrow s'$. Sinon, la transformation est rejetée : On garde s pour l'itération suivante.

6) Pour assurer la convergence (analogie de la balle qui rebondit de moins en moins), T est diminuée lentement à chaque itération, par exemple $T := k.T$, $k = 0.999$ par exemple. On peut aussi décroître T par paliers. Pour être efficace, un recuit doit diminuer T assez lentement, en plusieurs milliers ou dizaines de milliers d'itérations.

7) On s'arrête quand T atteint un seuil fixe ϵ, proche de 0.

2.5.1 Application de la méthode recuit simulé sur le PVC

La méthode la plus utilisée aujourd'hui est celle du « recuit simulé ». Cette méthode, très populaire ces dernières années a permis de résoudre des problèmes très complexes du type « voyageur de commerce » où les méthodes déterministes sont rapidement piégées dans des minimums locaux. L'algorithme va donc tenter de minimiser la longueur totale du chemin, en modifiant l'ordre des villes à parcourir.

Le but est alors de trouver le cycle hamiltonien de coût minimal dans un graphe. L'énergie représentera la distance totale à parcourir, et un état du système représentera le chemin entre les villes.

L'algorithme va donc tenter de minimiser la longueur totale du chemin, en modifiant l'ordre des villes à parcourir.

Soit le graphe suivant représentant un ensemble de 5 villes:

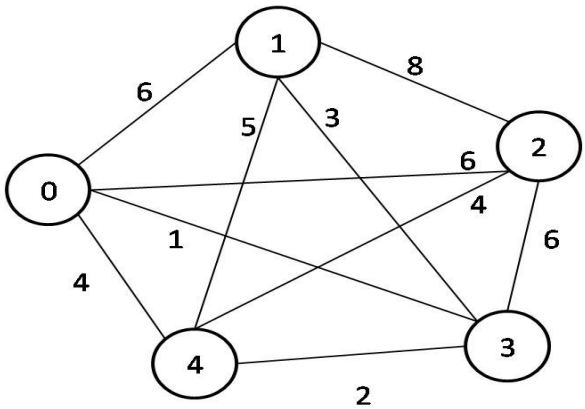


Figure 2.5: Un graphe de 5 villes d'un PVC

La solution la plus simple est de parcourir les villes dans l'ordre : (01234) de coût 26. On choisi une solution $s' \in V(s)$ qui se délacer d'un sommet vers son plus proche voisin cette solution est (03421) de coût 21.

On calcule $\Delta f = f(s') - f(s_0) = 21 - 26 = -5$, $\Delta f \leq 0$, donc le coût diminue et on effectue la transformation améliorante $s \longleftarrow (03421)$; donc on choisi une nouvelle solution $s' \in V(s)$, soit la solution (03412) de coût 22, ce résultat obtenu en échangeant les sommets 1 et 2. La distance totale a augmenté. Cette solution est rejetée car la distance doit être minimisée, mais le recuit simulé pour l'accepter si la température est encore élevée, et cette solution qui est « mauvaise » par rapport à la première va lui permettre de trouver une solution meilleure: Soit la solution $s' = (03142)$ de coût 18, le résultat obtenu en échangeant les sommets 4 et 1.

2.6 La méthode tabou

La méthode tabou est une technique de recherche dont les principes ont été proposés pour la première fois par Fred Glover dans les années 80, et elle est devenue très classique en optimisation combinatoire. Elle se distingue des méthodes de recherche locale simples par le recours à un historique des solutions visitées, de façon à rendre la recherche un peu moins « aveugle ». Il devient donc possible de s'extraire d'un minimum local, mais, pour éviter d'y retomber périodiquement, certaines solutions sont bannies, elles sont rendues «taboues».

A l'inverse du recuit simulé qui génére de manière aléatoire une seule solution voisine à chaque itération, tabou examine un échantillonnage de solutions de $V(s)$ et retient la meilleure s' même si $f(s') > f(s)$. La recherche tabou ne s'arrête donc pas au premier optimum trouvé.

Le danger serait alors de revenir à s immédiatement, puisque s est meilleure que s'. Pour éviter de tourner ainsi en rond, on crée une liste T qui mémorise les dernières solutions visitées et qui interdit tout déplacement vers une solution de cette liste. Cette liste T est appelée liste tabou [3].

La liste tabou

Glover a montré qu'une liste tabou de taille $t = 7$ à 20 suffit en pratique pour empêcher l'algorithme de boucler en revenant sur une solution déjà visitée. T fonctionne donc comme une sorte de mémoire à court terme. À chaque itération, la $t - énième$ solution de T est écrasée par la dernière solution examinée. En pratique, T se gère simplement avec une structure de données de type file.

En théorie, il faudrait stocker complètement les t dernières solutions visitées. Par exemple, pour un PVC avec le voisinage 2-opt, il faudrait conserver les t derniers cycles hamiltoniens trouvés et vérifier que tout nouveau cycle examiné n'est pas déjà dans T. Les voisinages considérés étant souvent grands, il est clair que le test répétitif de présence dans T est très coûteux, sans parler de la mémoire nécessaire pour coder S solutions complètes. Le stockage explicite des solutions n'est donc jamais pratiqué.

Une technique encore plus simple pour générer la liste tabou est d'interdire de repasser par les t dernières valeurs de la fonction objectif: il suffit de stocker uniquement le coût entier des t dernières solutions [14].

Algorithme de tabou

- Initialisation

  Choisir une solution admissible s;
  Calculer $f(s)$; donner la taille de la liste T; la condition d'arrêt;
  Poser $T \longleftarrow \emptyset$; $s^* \longleftarrow s$; $f(s^*) \longleftarrow f(s)$;

- Répéter

  Calculer $V(s)$ et $f(V(s))$;
  Choisir $s' \in V(s)$ tel que $f(s') < f(V(s))$ et $s' \notin T$;
  Si $f(s') \leq f(s^*)$ alors $s^* \longleftarrow s'$;
  Poser $s \longleftarrow s'$ et mise à jour T;

- Jusqu'à ce que le critère de terminaison soit satisfait.

- Fini.

2.6.1 Application de la méthode tabou sur le PVC

Soit l'exemple suivant:

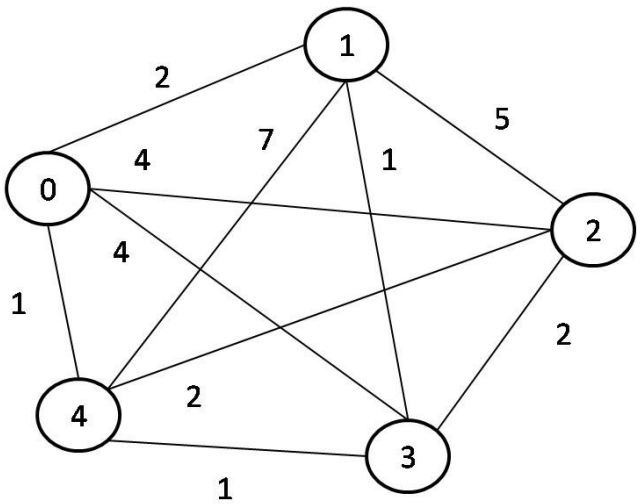


Figure 2.6: Exemple d'un PVC avec 5 villes

Initialisation

Soit $s = (03124)$ une solution admissible

$f(s) = d_{03} + d_{31} + d_{12} + d_{24} + d_{40} = 4 + 1 + 5 + 2 + 1 = 13$

Taille de la liste tabou $(t = 7)$

La condition d'arrête (nombre d'itération= 9)

$T = \{\Phi\}\,; s^* \longleftarrow (03124); f(s^*) \longleftarrow f(s) = 13$

Itération 1 : On a 5 villes, calculer le voisinage par 2-Opt, donc $|V(s)| = 5(5-3)/2 = 5$ donc on a 5 voisinages

$V(03124)$	(01324)	(02134)	(03214)	(03421)	(03142)
$f(V(03124))$	8 non tabou	13	19	15	18

on choisir $s' \in V(s)$ tel que $f(s') < f(Vs))$ et $s' \notin T$

donc $s' = (01324); f(s') = 8$

$8 = f(s') < f(s^*) = 13$

$s^* \longleftarrow s' = (01324); f(s^*) = 8$

$s \longleftarrow s' = (01324)$

$T = \{(01324)\}$

Itération 2 :

$V(01324)$	(03124)	(02314)	(01234)	(01423)	(01342)
$f(V(01324))$	13	15	12	17	11 non tabou

$s' = (01342); f(s') = 11$

$11 = f(s') > f(s^*) = 8$

$s \longleftarrow s' = (01342)$

$T = \{(01324), (01342)\}$

Itération 3 :

$V(01342)$	(03124)	(04312)	(01432)	(01243)	(01324)
$f(V(01342))$	18	13 non tabou	17	15	8 mais tabou

$s' = (04312); f(s') = 13$

$13 = f(s') > f(s^*) = 8$

$s \longleftarrow s' = (04312)$

$T = \{(01324), (01342), (04312)\}$

Itération 4 :

$V(04312)$	(03412)	(01342)	(04132)	(04213)	(04321)
$f(V(04312))$	22	11 mais tabou	15	13	12 non tabou

$s' = (04321); f(s') = 12$

$12 = f(s') > f(s^*) = 8$

$s \longleftarrow s' = (04321)$

$T = \{(01324), (01342), (04312), (04321)\}$

Itération 5 :

$V(04321)$	(03421)	(02341)	(04231)	(04123)	(04312)
$f(V(04321))$	15	17	8 non tabou	19	13

$s' = (04231); f(s') = 8$

$8 = f(s') = f(s^*) = 8; s^* \longleftarrow (04231)$

$s \longleftarrow s' = (04231)$

$T = \{(01324), (01342), (04312), (04321), (04231)\}$

Itération 6 :

$V(04231)$	(02431)	(03241)	(04321)	(04132)	(04213)
$f(V(04231))$	11 non tabou	17	12	15	13

$s' = (02431); f(s') = 11$

$11 = f(s') > f(s^*) = 8$

$s \longleftarrow s' = (02431)$

$T = \{(01324), (01342), (04312), (04321), (04231), (02431)\}$

Itération 7 :

$V(02431)$	(04231)	(03421)	(02341)	(02134)	(02413)
$f(V(02431))$	8 mais tabou	15	17	13 non tabou	18

$s' = (02134); f(s') = 13$

$13 = f(s') > f(s^*) = 8$

$s \longleftarrow s' = (02134)$

$T = \{(01324), (01342), (04312), (04321), (04231), (02431), (02134)\}$

Itération 8 :

$V(02134)$	(01234)	(03124)	(02314)	(02431)	(02143)
$f(V(02134))$	12 non tabou	13	15	11 mais tabou	22

$s' = (01234); f(s') = 12$

$12 = f(s') > f(s^*) = 8$

$s \longleftarrow s' = (01234)$

$T = \{(01342), (04312), (04321), (04231), (02431), (02134), (01234)\}$

Itération 9 :

$V(01234)$	(02134)	(03214)	(01324)	(01432)	(01243)
$f(V(01234))$	13	19	8 non tabou	17	15

$s' = (01324); f(s') = 8$

$8 = f(s') = f(s^*) = 8; s^* \longleftarrow (01324)$

$s \longleftarrow s' = (01324)$

$T = \{(04312), (04321), (04231), (02431), (02134), (01234), (01324)\}$

Itération 10 : Stop

Donc $s^* = (01324); f(s^*) = 8$

2.7 La méthode Kangourou

La méthode Kangourou est une technique d'approximation fondée sur la descente stochastique (aléatoire) à partir l'espace de recherche. Elle a été proposée par Gérard Fleury en 1993.

La méthode Kangourou est un algorithme itératif qui explore l'espace des solutions en choisissant à chaque fois la meilleure solution voisine de la solution courante. La recherche de la meilleure solution voisine est un problème qui peut être aussi difficile que le problème initial. Le processus de descente stochastique s'arrête après un nombre d'itérations fixé au début. La méthode fournit un minimum local. Soit s_0 une solution admissible du problème d'optimisation. Par des déplacements successifs l'algorithme de Kangourou cherche une solution s_i qui minimise la fonction f dans un voisinage de la solution courante. Si la solution s_i est meilleure que la solution précédente, elle est mémorisée et une nouvelle solution est cherchée dans le voisinage de la solution s_i. Si la solution s_i n'est pas meilleure que la solution précédente, l'algorithme trouve un autre solution par un saut, après un nombre d'itérations, et pour évité de tomber dans une solution déjà visitée on crié une liste T' qui on le mémorise toute solution visitée. Un minimum local s^* est trouvé, ce minimum est plus ou moins proche du minimum global [19].

Algorithme de Kangourou

Paramètre:

A : Le nombre d'itération qui laisse échanger la solution optimale

N : Le nombre maximal d'itération

s_0 : La solution initiale

s^* : La solution courante

T' : Une liste

f : La fonction objectif

Initialisation:

$s \longleftarrow s_0$; $s^* \longleftarrow s_0$; $f(s^*) \longleftarrow f(s_0)$;

$N =$ donner le nombre d'itération;

$i \longleftarrow 0$ compteur de N;

$k \longleftarrow 0$ compteur de A;

Tant que: $i < N$ (on va faire une descente stochastique au voisinage de s, et on utilise le voisinage 2-Opt);

$T' = \{\Phi\}$

Descente stochastique:

Si $k < A; u \longleftarrow Rand[V(s)]$;

Si $f(u) < f(s_0); s^* \longleftarrow u$;

Si non $k \longleftarrow k + 1$;

Fini si;

$s \longleftarrow u;\ i \longleftarrow i+; 1$

Si $k \geq A$; **(on va faire une saut)**

Choisir $u \in Rand$ ($S \backslash p$ dernières visites);

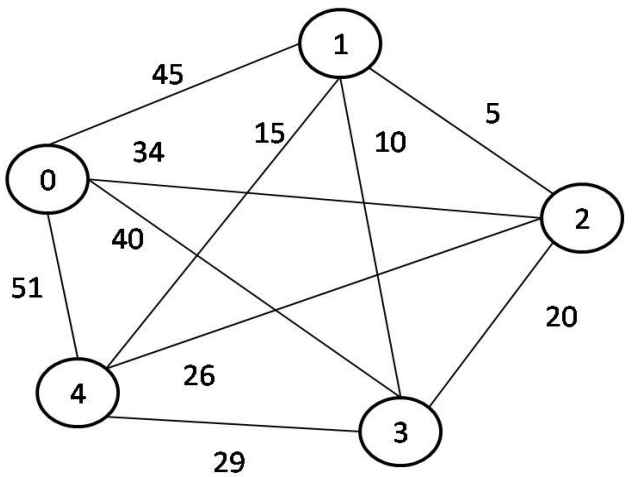
Si $f(u) < f(s^*); s^* \longleftarrow u$;

Fini si;

$s \longleftarrow u;\ i \longleftarrow i + 1;\ k \longleftarrow 0$;

2.7.1 Application de la méthode Kangourou sur le PVC

Considérons le problème du voyageur de commerce représenté par le réseau ci-dessous



Exemple 2.7: Le PVC avec 5 villes

Soit $s_0 = (10324)$ une solution initiale

$f(s_0) = d_{10} + d_{03} + d_{32} + d_{24} + d_{41} = 45 + 40 + 20 + 26 + 51 = 146$

$s \longleftarrow (10324); s^* \longleftarrow (10324); f(s^*) \longleftarrow 146$

$A = 2; N = 6$

$i = 0 \longrightarrow N; k = 0 \longrightarrow A$

$0 < 6$

$T^{'} = \{\Phi\}$ (on va faire une descente stochastique)

$0 < 2$

$V(10324)$	(13024)	(12304)	(10234)	(10423)	(10342)

$u = (10423); f(u) = 152$

$f(u) > f(s^*)$

$k \longleftarrow 0 + 1 = 1$

$s \longleftarrow (10423)$

$i \longleftarrow 0 + 1 = 1$

$i < 6$

$T^{'} = \{(10423)\}$ (on va faire une descente stochastique)

$k = 1 < 2$

$V(10423)$	(14023)	(12403)	(10243)	(10324)	(10432)

$u = (10432); f(u) = 150$

$f(u) > f(s^*)$

$k \longleftarrow 1 + 1 = 2$

$s \longleftarrow (10432)$

$i \longleftarrow 1 + 1 = 2$

$i < 6$

$T^{'} = \{(10423), (10432)\}$

$k = 2 \geq 2$(on va faire un saut)

On choisi $u \in Rand(S$ sauf $s \in T^{'})$

$u = (21340); f(u) = 129$

$f(u) < f(s^*)$

$s^* \longleftarrow (21340)$

$s \longleftarrow (21340)$

$i \longleftarrow 2 + 1 = 3$

$k \longleftarrow 0$

$i < 6$

$T^{'} = \{(10423), (10432), (21340)\}$ (on va faire une descente stochastique)

$k = 0 < 2$

$V(21340)$	(23140)	(24310)	(21430)	(21043)	(21304)

$u = (21043); f(u) = 150$

$f(u) > f(s^*)$

$k \longleftarrow 0 + 1 = 1$

$s \longleftarrow (21043)$

$i \longleftarrow 3 + 1 = 4$

$i < 6$

$T^{'} = \{(10423), (10432), (21340), (21043)\}$

$k = 1 < 2$

$V(21043)$	(20143)	(24013)	(21403)	(21340)	(21034)

$u = (20143); f(u) = 143$

$f(u) > f(s^*)$

$k \longleftarrow 1 + 1 = 2$

$s \longleftarrow (20143)$

$i \longleftarrow 4 + 1 = 5$

$i < 6$

$T^{'} = \{(10423), (10432), (21340), (21043), (20143)\}$

$k = 2 \geq 2$ (on va faire un saut)

On choisi $u \in Rand(S$ sauf $s \in T^{'})$

$u = (40312); f(u) = 132$

$f(u) > f(s^*)$

$s \longleftarrow (40312)$

$i \longleftarrow 5 + 1 = 6$

$k \longleftarrow 0$

stop

$s^* = (21340); f(s^*) = 129$

2.8 Conclusion

La résolution de problème du voyageur de commerce par une méthode exacte est pratiquement difficile et surtout pour le problème de grand taille, ce qui nous repousse l'utilisation des méthodes approchées (Métaheuristiques).

Chapitre 3

Etude de cas

3.1 Introduction

L'objectif principal du travail présenté dans ce mémoire est l'étude du potentiel de différents types d'algorithmes pour résoudre un problème d'optimisation combinatoire et surtout les problèmes qui appartient à la classe NP-Difficile, soit le problème du voyageur de commerce pour atteindre ce but, on choisi l'algorithme de descente simple pour résoudre le PVC et on va programmer cette algorithme par le logiciel Java.

3.2 Langage de programmation

Pour développer notre application, nous avons opté pour le langage Java: Le Java est un langage de programmation très utilisé, notamment par un grand nombre de développeurs professionnels, ce qui en fait un langage incontournable actuellement. Java est un langage de programmation moderne développé par Sun Microsystems, aujourd'hui racheté par Oracle. En fait, on peut distinguer trois grandes phases dans la vie d'un code Java:

La phase d'écriture du code source, en langage Java ;

La phase de compilation de votre code ;

La phase d'exécution [12].

3.3 Environnement de développement

Pour l'environnement de développement nous opté pour NetBeans**.**

NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 En plus de Java, NetBeans permet également de supporter différents autres langages, comme C, C++, JavaScript.

3.4 La méthode descente simple

La méthode descente simple ou la recherche locale simple est un algorithme d'amélioration très ancien. Son principe consiste à explorer le voisinage de la solution courante afin d'améliorer sa qualité progressivement. À chaque itération du processus d'amélioration, l'algorithme modifie un ensemble de composantes de la solution courante pour permettre le déplacement vers une solution voisine de meilleure qualité, le processus est répété itérativement jusqu'à la satisfaction du critère d'arrêt [3].

La méthode descente, on peut la schématiser comme suit:

1. Construire une solution initiale admissible s_0;

   Poser $s^* = s_0$ et calculer $f(s^*)$;

2. Calculer le voisinage de s^*(pour notre travail on appliquer la méthode descente par le voisinage 2-Opt);

3. Trouver s' tel que $f(s') < f(s) \forall s \in V(s^*)$;

4. Si $f(s') < f(s^*)$;

   $s^* \longleftarrow s'$ et aller en 2;

5. Si $f(s') \geq f(s^*)$ stop;

   $s^* \longleftarrow$ la solution; $f(s^*)$.

3.5 Simulation numérique

3.5.1 Génération des data de PVC

Pour appliquer la méthode descente simple sur le PVC on a choisi 10 problèmes. Pour chaque problème, on considère le problème du voyageur de commerce représente par le graphe $G(X, U, d)$ ou $X = \{0, 1, 2, 3, 4\}$ l'ensemble de 5 villes, $U = (i, j)$ représente les *arêtes* et d représente la distance entre les villes, telle que d prend les valeurs dans l'intervalle $[1, 100]$. Cet intervalle est divisé en 10 intervalles équitablement.

Il est important de noter que la solution initiale de chaque problème est générée aléatoirement. Quant au voisinage on a utilisé le 2-Opt. Le programme de l'algorithme a été exécuté en utilisant le langage Java.

Solution initaile $s^* = s_0$	$f(s^*)$	$V(s^*)$	$f(V(s^*))$	$s' \in V(s^*)$	comparaison	s^* et $f(s^*)$
(23401)	31	(24301) (20431) (23041) (23104) (23410)	22 32 21 27 31	(23041)	$f(s') < f(s^*)$	$s^* = (23041)$ $f(s^*) = 21$
(23041)	21	(20341) (24031) (23401) (23140) (23014)	24 20 31 29 19	(23014)	$f(s') < f(s^*)$	$s^* = (23014)$ $f(s^*) = 19$
(23014)	19	(20314) (21034) (23104) (23410) (23041)	20 22 27 31 21	(20314)	$f(s') > f(s^*)$ Stop	$s^* = (23014)$ $f(s^*) = 19$

Tableau 3.1: Résultats de la méthode descente pour le PVC pour des data dans $[1, 10]$

Solution initaile $s^* = s_0$	$f(s^*)$	$V(s^*)$	$f(V(s^*))$	$s' \in V(s^*)$	comparaison	s^* et $f(s^*)$
(23401)	76	(24301) (20431) (23041) (23104) (23410)	81 74 72 73 74	(23041)	$f(s') < f(s^*)$	$s^* = (23041)$ $f(s^*) = 72$
(23041)	72	(20341) (24031) (23401) (23140) (23014)	76 75 76 68 75	(23140)	$f(s') < f(s^*)$	$s^* = (23140)$ $f(s^*) = 68$
(23140)	68	(21340) (24130) (23410) (23041) (23104)	74 73 74 72 73	(23041)	$f(s') > f(s^*)$ Stop	$s^* = (23140)$ $f(s^*) = 68$

Tableau 3.2: Résultats de la méthode descente pour le PVC pour des data dans $[11, 20]$

Solution initaile $s^* = s_0$	$f(s^*)$	$V(s^*)$	$f(V(s^*))$	$s' \in V(s^*)$	comparaison	s^* et $f(s^*)$
(23401)	125	(24301) (20431) (23041) (23104) (23410)	131 132 121 125 115	(23410)	$f(s') < f(s^*)$	$s^* = (23410)$ $f(s^*) = 115$
(23410)	115	(24310) (21430) (23140) (23014) (23401)	130 123 120 119 125	(23014)	$f(s') > f(s^*)$ Stop	$s^* = (23410)$ $f(s^*) = 115$

Tableau 3.3: Résultats de la méthode descente pour le PVC pour des data dans $[21, 30]$

Solution initaile $s^* = s_0$	$f(s^*)$	$V(s^*)$	$f(V(s^*))$	$s' \in V(s^*)$	comparaison	s^* et $f(s^*)$
(23401)	181	(24301) (20431) (23041) (23104) (23410)	181 186 176 180 172	(23410)	$f(s') < f(s^*)$	$s^* = (23410)$ $f(s^*) = 172$
(23410)	172	(24310) (21430) (23140) (23014) (23401)	181 177 176 171 181	(23014)	$f(s') < f(s^*)$	$s^* = (23014)$ $f(s^*) = 171$
(23014)	171	(20314) (21034) (23104) (23410) (23041)	176 181 180 172 176	(23410)	$f(s') > f(s^*)$ Stop	$s^* = (23014)$ $f(s^*) = 171$

Tableau 3.4: Résultats de la méthode descente pour le PVC pour des data dans [31, 40]

Solution initaile $s^* = s_0$	$f(s^*)$	$V(s^*)$	$f(V(s^*))$	$s' \in V(s^*)$	comparaison	s^* *et* $f(s^*)$
(23401)	227	(24301) (20431) (23041) (23104) (23410)	228 235 227 220 223	(23104)	$f(s') < f(s^*)$	$s^* = (23104)$ $f(s^*) = 220$
(23104)	220	(21304) (20134) (23014) (23401) (23140)	230 226 218 227 225	(23014)	$f(s') < f(s^*)$	$s^* = (23014)$ $f(s^*) = 218$
(23014)	218	(20314) (21034) (23104) (23410) (23041)	226 228 220 223 227	(23104)	$f(s') > f(s^*)$ Stop	$s^* = (23014)$ $f(s^*) = 218$

Tableau 3.5: Résultats de la méthode descente pour le PVC pour des data dans [41, 50]

Solution initaile $s^* = s_0$	$f(s^*)$	$V(s^*)$	$f(V(s^*))$	$s' \in V(s^*)$	comparaison	s^* et $f(s^*)$
(23401)	271	(24301) (20431) (23041) (23104) (23410)	282 277 272 271 277	(23104)	$f(s') = f(s^*)$ Stop	$s^* = (23401)$ $f(s^*) = 271$

Tableau 3.6: Résultats de la méthode descente pour le PVC pour des data dans [51, 60]

Solution initaile $s^* = s_0$	$f(s^*)$	$V(s^*)$	$f(V(s^*))$	$s' \in V(s^*)$	comparaison	s^* et $f(s^*)$
(23401)	329	(24301) (20431) (23041) (23104) (23410)	326 337 332 324 326	(23104)	$f(s') < f(s^*)$	$s^* = (23104)$ $f(s^*) = 324$
(23104)	324	(21304) (20134) (23014) (23401) (23140)	332 326 321 329 332	(23014)	$f(s') < f(s^*)$	$s^* = (23014)$ $f(s^*) = 321$
(23014)	321	(20314) (21034) (23104) (23410) (23041)	329 326 324 326 332	(23104)	$f(s') > f(s^*)$ Stop	$s^* = (23014)$ $f(s^*) = 321$

Tableau 3.7: Résultats de la méthode Descente pour le PVC pour des data dans [61, 70]

Solution initaile $s^* = s_0$	$f(s^*)$	$V(s^*)$	$f(V(s^*))$	$s' \in V(s^*)$	comparaison	s^* et $f(s^*)$
(23401)	377	(24301) (20431) (23041) (23104) (23410)	377 382 379 375 378	(23104)	$f(s') < f(s^*)$	$s^* = (23104)$ $f(s^*) = 375$
(23104)	375	(21304) (20134) (23014) (23401) (23140)	379 378 375 377 380	(23014)	$f(s') = f(s^*)$ Stop	$s^* = (23104)$ $f(s^*) = 375$

Tableau 3.8: Résultats de la méthode descente pour le PVC pour des data dans $[71, 80]$

Solution initaile $s^* = s_0$	$f(s^*)$	$V(s^*)$	$f(V(s^*))$	$s' \in V(s^*)$	comparaison	s^* et $f(s^*)$
(23401)	428	(24301) (20431) (23041) (23104) (23410)	424 431 424 429 426	(24301)	$f(s') < f(s^*)$	$s^* = (24301)$ $f(s^*) = 424$
(24301)	424	(23401) (20341) (24031) (24103) (24310)	428 424 427 422 429	(24103)	$f(s') < f(s^*)$	$s^* = (24103)$ $f(s^*) = 422$
(24103)	422	(21403) (20143) (24013) (24301) (24130)	424 426 429 424 425	(21403)	$f(s') = f(s^*)$ Stop	$s^* = (24103)$ $f(s^*) = 422$

Tableau 3.9: Résultats de la méthode descente pour le PVC pour des data dans $[81, 90]$

Solution initaile $s^* = s_0$	$f(s^*)$	$V(s^*)$	$f(V(s^*))$	$s' \in V(s^*)$	comparaison	s^* et $f(s^*)$
(23401)	481	(24301) (20431) (23041) (23104) (23410)	478 482 482 483 480	(24301)	$f(s') < f(s^*)$	$s^* = (24301)$ $f(s^*) = 478$
(24301)	478	(23401) (20341) (24031) (24103) (24310)	481 479 482 480 480	(20341)	$f(s') > f(s^*)$ Stop	$s^* = (24301)$ $f(s^*) = 478$

Tableau 3.10: Résultats de la méthode descente pour le PVC pour des data dans [91, 100]

3.5.2 Discussion les résultats numériques

En observant les tableaux des résultats numériques on remarque que 60% des cas la méthode descente s'échappe du minimum local ce qui est raisonnable. Il semble aussi que la valeur de la meilleure solution locale par la méthode descente est très proche de la valeur de la solution optimale. Ceci montre que, malgré que les méthodes exactes telles que la séparation et évaluation et la programmation dynamique donne une solution optimale exacte dans un temps raisonnable (car n est petit), alors la méthode descente donne une solution très proche de la solution optimale mais reste plus rapide.

Il est tout à fait clair que l'étude qui a été faite est préliminaire car on a considéré que les problèmes des petites tailles, alors que les métaheuristiques sont applicables aux problèmes de grandes tailles. Aussi il faut essayer d'utiliser un benchmark (TSPLIB) connu et utilisé par les chercheurs dans ce domaine pour pouvoir juger l'efficacité de la méthode descente. Aussi le programme pourrait être amélioré et des paramètres tels que λ doit être changé et voir quel est la valeur de λ pour lequel l'algorithme performe mieux. De plus, et puisque la méthode descente est la base des métaheuristiques, le programme pourrait être développé pour inclure les autres méthodes telles que tabou, recuit simulé, Kangourou. Par conséquent une comparaison globale des quatre méthodes pourrait être faite.

3.6 Conclusion

Dans ce chapitre on a appliqué une des métaheuristiques connue sous le nom méthode descente sur le problème du voyageur de commerce. Etant donné que ces méthodes sont très rapides pour déterminer une solution approchée et d'ailleurs ne nécessitent pas beaucoup de mémorisation.

On a choisi cette méthode car elle représente le core de toutes les métaheuristiques. On a choisi le problème du voyageur de commerce car c'est le problème le plus renommé des problèmes d'optimisation combinatoire.

En tout état de cause on a fait qu'une étude préliminaire où on a générer une dizaine de problèmes chacun de 5 villes. On a programmé l'algorithme de solution en langage Java.

Les résultats numériques finaux indiquent que 90% l'algorithme donne une solution optimale (car l'instance n est très petit).

De plus 60% des cas le nombre d'itérations est au maximum égale à 3.

Conclusion générale

Dans ce manuscrit on a considéré le problème du voyageur de commerce (PVC): Un voyageur de commerce désire visiter un certain nombre de villes, débutant et finissant son parcours dans la même ville, en visitant chacune des autres villes une et une seule fois. Il désire sélectionner la tournée qui minimise la distance totale. On a donné des cas généraux: Le problème du voyageur de commerce généralisé, le problème de tournées de véhicules, le problème de m voyageur et des cas particuliers: Le problème du voyageur de commerce métrique et le problème du voyageur de commerce euclidien. Le PVC est un modèle si important de problèmes d'optimisation combinatoire. C'est d'ailleurs le plus utilisé par les chercheurs pour juger l'efficacité d'une méthode de solution quelconque. Malgré sa simplicité apparente, le problème du voyageur de commerce est l'un des problèmes les plus difficiles de sa classe. Il est NP-Difficile au sens fort. Pour sa solution on a considéré une méthode exacte célèbre dite: Par séparation et évaluation. Par un exemple illustratif on a montré le mécanisme et le fonctionnement de cette méthode, on a parlé des facteurs déterminants son efficacité: l'évaluation, la séparation, la stratégie de recherche. On a mentionné son inconvénient qui réside dans le fait qu'elle échoue de résoudre les problèmes de ce genre de grande taille. C'est pour cela qu'on a introduit ce qu'on appelle les métaheuristiques qui déterminent une solution approchée mais dans un temps très court. Citant la méthode descente et la méthode descente stochastique, recuit simulé, tabou, et la méthode Kangourou. Via des exemples on a démontré leurs fonctionnements et leurs mécanismes.

A la fin une étude numérique est donné pour la performance de l'algorithme de la méthode descente applique au PVC. Cet algorithme a été programmé en Java.

Bibliographie

[1] M. Akli. Problème de tournées de véhicules avec contraintes et fenêtre de temps. Thèse de Magister. Université de Mouloud Mammeri, Tizi Ouzou, (2013).

[2] D. Appligate, W. Cook. Acomputational study of the job shop scheduling problem. ORSA jornal on computing, n°2, (1991), 149-156.

[3] B. Autin. Les métaheuristiques en optimisation combinatoire. Mémoire. Conservation national des arts et metiers, Paris. (2006).

[4] S. Ben Ismail. Introduction à l'optimisation combinatoire. Telecom Bretagne, (2012).

[5] N. L. Biggs, E. Lloyd Keith, J. Wilson Robin. Graph theory. Clarendon press, (1986), 1736-1936.

[6] B. Bontoux. Techniques hybrides de recherche exacte et approchée: application à des problèmes de transport. Thèse de doctorat. Université d'Avignon et des pays de vaucluse, France, (2008).

[7] P. M. Camerini, L. Fratta, F. Maffioli. On inproving relaxation methodes by modified gradient techniques. Mathematical programming study, n°3. (1974), 26-34.

[8] G. B. Dantzig, D. R. Fulkerson, S. M. Johnson. Solution of a large scale traveling salesman problem. Operations Research, n°4, (1954), 393-410.

[9] S. M. Douiri, S. Elbevoussi, H. Lakhbab. Cours des méthodes de résolution exactes, heuristiques et métaheuristiques. Université Mohammed V, Faculté des Sciences de Rabat. Laboratoire de Recherche Mathématiques,

[10] M. R. Garey, D. S. Johnson. Computers and Intractability. A guide to the theory of NP-Completeness. W. H. Freeman and company, New York, (1979).

[11] J. K. Hao, P. Galinier, M. Habib. Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. Revue d'Intellgence Artificielle, n°2, (1999), 283-324.

[12] A. Herby. Apprenez à programmation en Java . Imprim'vert, France, (2001).

[13] Hellouin. Benjamin. Problème du voyageur de commerce: cas métrique. Université Andrés Bello. Département de Mathématiques

[14] P. Lacomme, C. Prins, M. Sevaux. Algorithmes de graphes. Eyrolles, France, (2003).

[15] M. Padberg, G. Rinaldi. A branch and Cut algorithm four the resolution of alarge-scale symetric traveling salesman problems. SIAM Review, n°1, (1991), 60-100.

[16] C. Rego, C. Rouciarol. Le probleme de tournées de véhicules: étude et reolution approchee. Rapport de recherche, n°2179, France, (1994).

[17] M. Sakarovitch. Optimisation combinatoire. Méthodes mathématiques et algorithmiques. Programation Discrète. Hermann, France, (1984).

[18] M. W. P. Savelsberg. Local search in routing problems with times windows. Annals of operations research 4, n°1, (1985), 285-305.

[19] A. Serbencu, V. Minzu, A. Serbencu. An ant colony system based metaheuristic for solving singlemachine scheduling problem. Revue dunarea, University of galati fascill III, (2007), 19-24.

[20] A. Schrijise. On the history of combinatorial optimization, (1960).

[21] B. Tadunfock Teti, L. P. Fotso. Heuristique du problème du voyageur de commerce. Université de yaouandé 1,vol 1, (2006).

[22] http://www.uqac.ca/rebaine/8INF806/techniquedebranchandboundcourshiver2005.pdf.

Annexe