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

Mémoire de Master

Spécialité : Modélisation Contrôle et Optimisation

Thème

sur l'Algorithme de Volume Modifie Pour un Problème d'Optimisation Combinatoire

Présenté par

BOUATTOU MILOUD

Soutenu le 2014//2015

Devant le jury

Président	**: AMIR**	.A	**U. MOSTAGANEM.**
Examinateur	**: BELGACEM**	.R	**U. CHLEF**
Encadreur	**: HABLAOUI**	.H	**U. MOSTAGANEM.**

Remerciements

Je remercie tout d'abord notre dieu qui ma donne de la volonté et surtout de la patience pour réalise ce travail.

Je souhaite avant tout remercier mon encadreur de Mémoire Monsieur H.ABLAOUI maître à l'université de Mostaganem pour le temps qu'il a consacré à m'apporter les outils méthodologiques indispensables à la conduite de cette recherche. Son exigence m'a grandement stimulée. L'enseignement de qualité dispensé par le Master a également su nourrir mes réflexions et a représenté une profonde satisfaction intellectuelle, merci donc aux enseignants.

Mes plus sincères remerciements à monsieur A.AMIR d'avoir bien voulu présider mon jury et monsieur R.BELGACEM d'avoir accepté de faire partie de ce jury.

J'aimerais exprimer ma gratitude à tous les enseignements, trop nombreux pour les citer, qui ont pris le temps de discuter de mon sujet. Chacun de ces échanges m'a aidé à faire avancer mon analyse.

Enfin, un grand merci à mes parent qui m'ont apporté une aide précieuse

Je vous souhaite à tous une agréable lecture !.

Table des matières

INTRODUCTION

Dans un problème d'optimisation certaines variables sont astreintes à être entières ou même binaires $x_i = 0$ ou 1, on résout donc un programme linéaire en nombre entiers ILP ou un programme linéaire binaire BIP.

Dans ce travail ,on s'intéresse à un problème particulier d'optimisation combinatoire qui est connu sous le nom de problème de voyageur de commerce.

Le problème du voyageur de commerce (en anglais "traveling salesman problem" noté TSP) a été introduit pour la première fois en 1930 par le mathématicien viennois Karl Menger. Il a intrigué un bon nombre de chercheurs depuis ce temps.

Un voyageur de commerce doit visiter n villes données en passant par chaque ville exactement une fois. Il commence sa tounée par une ville quelconque et la termine en retournant à la ville de départ. Les distances entre les villes sont connues. Quel chemin faut-il choisir afin de minimiser la distance parcourue ?

Le problème du voyageur de commerce (TSP) est un problème d'optimisation combinatoire réputé pour être difficile (il appartient à la classe de problèmes dite NP-complets). Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, mais ils sont généralement difficiles à résoudre.

Plusieurs méthodes ont été proposées pour résoudre le problème du TSP (dont certaines assez récemment) :Ces méthodes peuvent être classées sommairement en deux grandes catégories :

1. Les méthodes exactes qui garantissent la complétude de la résolution telle que la méthode de séparation et évaluation (en anglais Brunch and bound) ou des méthodes quasi-exactes comme la méthode de sous gradient
2. Les méthodes approchées qui sacrifient la complétude pour gagner en efficacité : La méthode du plus proche voisin, de Lin-Kernighan, de l'élastique, du recuit simulé, les réseaux de neurones auto-adaptatifs, les algorithmes génétiques, la procédure de recherche tabu, colonie de fourmis. Certaines méthodes sont très simples et d'autres très complexes.

La notion de distance peut-être remplacée par d'autres notions comme le temps qu'il met ou l'argent qu'il dépense : dans tous les cas, on parle de coût. Donc le problème de voyageur de commerce consiste en la recherche d'un trajet minimal en traversant n villes.

Les domaines d'application sont nombreux : problèmes de logistique, de transport aussi bien de marchandises que de personnes,et plus largement toutes sortes de problèmes d'ordonnancement. Certains problèmes rencontrés dans l'industrie se modélisent sous la forme d'un problème de voyageur de commerce, comme l'optimisation de trajectoires de machines outils : comment percer plusieurs points sur une carte électronique le plus vite possible ?

Un cadre particulièrement intéressant pour la relaxation [1] (mais aussi pour la décomposition) est celui donné par la dualité lagrangienne. Son principe consiste à ajouter des termes d'une inégalité dans la fonction objectif en ajoutant un multiplicateur visant à pénaliser la fonction objectif si la solution trouvée ne vérifie pas cette inégalité. Cette relaxation fournit fréquemment de très bonnes valeurs de relaxation.

La méthode de sous-gradient [3] est fréquemment utilisée pour optimiser la fonction duale lagrangienne pour des problèmes de programmation en nombres entiers.

Dans la méthode du sous-gradient, tous les sous-problèmes doivent être résolus de façon optimale pour obtenir une direction de déplacement.

Dans ce travail, nous adaptons la méthode du sous-gradient de remplacement (en anglais "surrogate subgradient), méthode développée par Zhao [2],où seul un sous-problème est optimisé pour obtenir une bonne direction de déplacement.

Nous avons au départ prévu d'implémenter l'algorithme du volume, procédure qui utilise une méthode du sous-gradient, en remplaçant la procédure du sous-gradient par celle du sous-gradient de remplacement mais nous nous sommes contenté d'étudier la procédure du sous-gradient de remplacement car la seule modification envisagée, de l'algorithme du volume, consiste à remplacer la méthode du sous-gradient par celle du remplacement.

Nous avons divisé notre travail en quatre chapitres :

Le premier chapitre est une introduction à la théorie des graphes où nous donnons quelques définitions que nous allons utiliser par la suite..

Au second chapitre, on présente la formulation mathématique du problème de voyageur de commerce.

Les méthodes de résolution du TSP seront abordées dans le troisième chapitre, nous terminons par un quatrième chapitre qui sera consacré aux tests numériques pour enfin donner une petite conclusion.

Chapitre 1

Notations et rappels

$\mathbb{R}$: L'ensembles des nombres réels

$\mathbb{R}^m$: L'espace des vecteurs a m composantes réelles

$\mathbb{R}^{m \times n}$: L'espace des matrices réelles de dimension $m \times n$

$v(p)$: La valeur optimale de problème (p)

V : $V = \{1, 2, ..., n\}$.L'ensemble des sommets

E : $E = \{e = (i, j) : i, j \in V\}$ .L'ensemble des arêtes

c_e : cout de l'arêtes e

$\delta(i)$: L'ensemble des arêtes incidentes au noeud i

$$\delta(i) = \delta(\{i\}), i \in s$$

$\cap, \cup$: Intersection (respectivement réunion) d'ensembles

$S \subset V$: S est strictement contenu dans V

$|S|$: Cardinale de S : nombre d'éléments de S

$x \in S$: x est élément de l'ensemble S

$\|.\|$: Norme euclidienne

$proj$: L'opérateur de projection $\mathbb{R}^m$

$\emptyset$: L'ensemble vide

Abréviations

- LP : programme linéaire
- ILP : programme linéaire en nombres entiers.
- BIP : programme linéaire à variables binaires.

– TSP : travelling salesman problem.

– $B\&B$: branch and bound.

– SG : Méthode du sous-gradient.

– SSG : Méthode du sous-gradient de remplacement.

Ci-dessous nous donnons quelques notions et rappels sur la théorie des graphes afin de faciliter la lecture de notre travail par un lecteur qui est peu familier avec la théorie des graphes.

Définition 1.0.1 *(un graphe)*

Un graphe G est défini par la donné :

– *D'un ensemble fini X (appelé ensemble des sommets)*

– *D'un ensemble fini E (appelé ensemble des arêtes) .*

– *d'une application e qui associe à chaque arête ses extrémités.*

Définition 1.0.2 *(Une arête)*

Une arête e du graphe est une paire $e = (x, y)$ de sommets. Les sommets x et y sont les extrémités de l'arête. On dit que les sommets x et y sont adjacents , que e est adjacente à x ou bien adjacentes aux sommets x et y.On dit aussi que l'arête e est incidente au sommet x (ou y ou aux deux).

Deux arêtes e et v qui ont une extrémité en commun sont dites aussi adjacentes

Définition 1.0.3 *(degré)*

On définit le degré d'un sommet x (qu'on note $d_G(x)$) dans G par le nombre d'arêtes auquelles x est adjacent.

Définition 1.0.4 *(Une chaine)*

Une chaine C reliant deux sommets x et y est une séquence d'arêtes $\{e_1, e_2, ...e_k\}$ telle que : x est adjacent à e_1 ,les arêtes e_i et e_{i+1} sont adjacentes et enfin y est une extémité de e_k. Un chaine est simple si elle ne passe qu'une seule fois par chacune de ses arêtes.

Définition 1.0.5 *(Un cycle)*

Un cycle est une chaine qui relie un sommet à lui même.

Définition 1.0.6 *(Graphes sans cycles)*

Un graphe non orienté G est sans cycles ou acyclique s'il ne posséde pas de cycles.

Définition 1.0.7 *(Un sous-graphe)*

Un sous-graphe de G consiste à considérer seulement une partie des sommets de V et les arêtes induits par E.

Définition 1.0.8 *(un graphe partiel)*

Un graphe partiel de G consiste à ne considérer qu'une partie des arêtes de E.

Définition 1.0.9 *(Un graphe complet)*

Un graphe est dit complet si pour tout couples de sommets x et y du graphe, il existe une arête du graphe qui a pour extrémités ces sommets.

Définition 1.0.10 *(Un graphe connexe)*

Un graphe G est connexe si pour tout couples de sommets x et y du graphe il existe une chaine dans G qui les relie. Si G n'est pas connexe alors l'ensemble des sommets pour lesquels il existe une chaine d'un sommet a donné vers ces derniers est appelée composant connexe.

Définition 1.0.11 *(Les arbres)*

Un arbre est un graphe connexe et sans cycles.

Une forêt est un graphe sans cycles (chacune de ses composantes connexes est un arbre).

Définition 1.0.12 *(1-arbre d'un graphe $G = (X, U)$)*

On suppose que $X = \{1, 2, 3, ..., n\}$ et soit $G1$ le sous-graphe induit par le sous-ensemble de sommets $X \setminus \{1\}$.

Un $1 - arbre$ est graphe partiel de G constitué par un graphe partiel de $G1$ qui est un arbre auquel on a rajouté le sommet 1 et deux arêtes qui lui sont incidentes.

Définition 1.0.13 *(Un cycle hamiltonien)*

Est un cycle où chaque sommet de G n'est "visité " qu'une et une seule fois.

Un arbre couvrant de G est un graphe partiel de G qui est un arbre : il est dit de poids minimum si son poids (qui représent la somme des poids de ses arêtes) est minimum.

Définition 1.0.14 *(La longueur)*

La longueur d'une chaine correspond au nombre d'arêtes qui la constituent ou à la somme des poids de ces dernières (Si G est valué).

Algorithme de Kruskal Soit $G = (X, U)$ un graphe et soit c une application de $U \rightarrow \mathbb{R}$ (On associe à chaque arête e un poids $c(e)$). Pour déterminer un graphe partiel de G qui soit un arbre et de poids minimum on applique l'algoritme suivant qui est du à Kruskal et qui porte son nom :

Algorithme de Kruskal(X, U, C, n, m)

Trier les arêtes de U par ordre de poids croissant
(i.e : $c(e_1) \leq c(e_2) \leq ..... \leq c(e_m)$)
Initialisation :
  $A \leftarrow \varnothing$
  Pour $j = 1, 2, ..., m$
    Si$((X, A \cup \{e_j\})$ ne contient pas de cycles)
      $A \leftarrow A \cup \{e_j\}$
    fin si
  Fin pour
Fin algorithme

Remarque 1.0.1 *Il est trés facile de montrer que le graphe partiel obtenu par l'algorithme est un arbre de poids minimum.*

Chapitre 2

Problème du voyageur de commerce

2.1 Historique sur le TSP

Des problèmes mathématiques liés au problème du voyageur de commerce (en anglais, TSP : Traveling Salesman Problem) ont été étudiés au dixneuvième siècle par les mathématiciens Irlandais, Sir William Rowan Hamilton, et par le Britannique, Thomas PenyngtonKirkman. Hamilton a créé le jeu Icosian en 1857qui nécessite un lecteur pour déterminer une tournée en utilisant seulement des connecteurs spécifiés par 20 points [CT03].

Dans les années 1930 le TSP est traité plus en profondeur par Karl Menger à Harvard, Hassler Whitney et Merril Flood.

1954 Solution du TSP pour 49 villes par Dantzig, Fulkerson et Johnson par la méthode du cutting-plane, les villes représentent 48 capitale état d'ETATS-UNIS plus Washington.

1975 Solution pour 100 villes par Camerini, Fratta and Maffioli

1987 Solution pour 532, puis 2392 villes par Padberg et Rinaldi

1998 Solution pour les 13.509 villes des Etats-Unis.

2001 Allemagne par Applegate, Bixby, Chvàtal et Cook des universités de Rice et Princeton appliquent une version parallèle de la méthode de Danzig, Fulkerson et de Johnson sur une grande instance de TSP, Ils ont obtenu la solution optimale du TSP qui a 15.112 villes (nœuds) en Allemagne, Le calcul a été exécuté sur un réseau de 110 processeurs et ils ont estimé tout le temps de calcul étaient de $22,6$ ans, mesurés à un processeur d'alpha de Compaq EV6(21264) fonctionnant à 500MH[FKTY03].

En mai 2004, le problème du voyageur de commerce qui consiste à visiter chacune des 24.978 villes en Suède a été résolu : une excursion de longueur approximativement de 72.500 kilomètres a été trouvée et on a montrée l'inéxistence d'une tournée plus courte.
En mars 2005, un problème de taille égale 33.810 villes a été résolu (Cook et al. 2006).

2.2 Formulation :

Le problème du voyageur de commerce est l'un des problèmes de l'optimisation discrète les plus étudié. Bien que sa formulation soit simple, il reste l'un des problèmes les plus difficile à résoudre.
Soit $G = (V; E)$ un graphe non-orienté où V est un ensemble de n sommets (villes) et E est l'ensemble des arêtes qui relient ces villes (des liaisons). On définit $c(e)$ comme le coût associé à l'arête e, pour toute $e \in E$. Le problème du (TSP) consiste à déterminer un cycle passant par chaque ville une et une seule fois : Un tel cycle est dit hamiltonien, on l'appelle aussi tour ou tournée.Ce problème classique a trouvé de nombreuses applications dans des domaines aussi éloignés que la production de lait ou la construction de circuits intégrés.

2.2.1 Formulation sous forme d'un programme linéaire :

Le probème du TSP peut se formuler comme suit :

$$
(\mathrm{P})\begin{cases}
\min \displaystyle\sum_{1\le i<j\le n} c_{ij}x_{ij} & & (1)\\
\displaystyle\sum_{i<j} x_{ij} + \sum_{j<i} x_{ji} = 2 & \text{pour } i \in \{2,3,...,n\} & (2)\\
\displaystyle\sum_{i<j, ij\in S} x_{ij} \le |S| - 1 & \text{pour } S \subset \{2,3,...,n\} & (3)\\
\displaystyle\sum_{j} x_{1j} = 2 & & (4)\\
\displaystyle\sum_{1\le i<j\le n} x_{ij} = n & & (5)\\
x_{ij} \in \{0,1\} & \forall i < j & (6)
\end{cases}
$$

(1) : fonction objectif. on minimise le cout total de la tournée

(2) et (4) : ces contraintes signifient qu'on doit passer une et une seule fois par une ville i

(3) : contrainte d'élimination de sous-tours

(5) : on doit visiter toutes les villes

(6) : contraintes d'intégrité

Chapitre 3

Méthode de résolution

3.1 Relaxation du problème de TSP

Le programme linéaire (P) est un programme linéaire à variables bivalentes donc on ne peut lui appliquer les méthodes classiques de la programmation linéaire telle que la méthode du simplexe. Pour la résolution de (P) on utilise le plus souvent des techniques de relaxation comme la relaxation Lagrangienne.

Les techniques de relaxation sont utilisées souvent pour l'obtention d'une borne inférieure. Appliquées à un problème de minimisation, elles fournissent une bonne évaluation par défaut de l'optimum, en relâchant les contraintes les plus "difficiles" à satisfaire, c'est à dire en les supprimant, ou en les prenant partiellement en compte.

Il existe trois types de relaxation : relaxation linéaire, relaxation lagrangienne, relaxations des contraints.

On s'intéresse ici à la relaxation Lagrangienne.

3.1.1 Relaxation lagrangienne

La relaxation lagrangienne s'articule autour de l'idée qui consiste à relâcher les contraintes difficiles, non pas en les supprimant totalement, mais en les prenant en compte dans la fonction objectif de sorte qu'elles pénalisent la valeur des solutions qui violent ces dernières.

Soit (P1) le programme mathématique suivant :

$$(\text{P1}) : \begin{cases} \min z = f(x) \\ g(x) \leq 0 \\ x \in S \subset \mathbb{R}^n \end{cases}$$

Supposons que la contrainte $g(x) \leq 0$ soit difficile à satisfaire et S un ensemble fini discret ayant une structure spéciale telle que l'on connaisse un "bon" algorithme pour résoudre le problème dit relaxé suivant :

$$(L_\pi) \begin{cases} \min f(x) + \pi.g(x) \\ x \in S \end{cases}$$

On associe à chaque contrainte i ; $g_i(x) \leq 0$ une variable duale $\pi_i \geq 0$ (les coefficients $\pi_1^k, \pi_2^k, ...$et π_n^k sont appellés multiplicateurs de lagrange). Nous définissons la fonction duale :

$$L(\pi) = \min_{x \in S} (f(x) + \pi.g(x)) \tag{3.1.1}$$

la fonction duale L est en général une fonction concave (enveloppe inférieure d'une famille de fonction linéaire affine) continue mais non différentiable.

nous devons résoudre alors le problème duale de (L_π) :

$$(D) \begin{cases} L(\pi^*) = \max L(\pi) \\ \pi \geq 0 \end{cases}$$

3.2 Méthode du sous-gradient

Les méthodes du sous-gradient sont fréquemment utilisées pour résoudre les problèmes d'optimisation non différentiables. Elles ont été introduites par Shor. Les premiers résultats de convergence sont dus à Ermol'ev,Polyak ou Nemirovskii

Shor a présenté un aperçu général des premiers développements sur les méthodes du sous-gradient.

Ces méthodes s'apparentent aux algorithmes du gradient dans le cas différentiable, en effectuant un pas dans la direction opposée au sous-gradient.

Définition 3.2.1 *a- soit un vecteur* $\xi = (\xi_1, \xi_2, ..., \xi_m)^t$. *Il est dit sous-gradient de L au point* π *si :*

$$\forall \pi' \in \mathbb{R}^m \; L(\pi') - L(\pi) \leq (\pi' - \pi).\xi$$

b- l'ensemble des sous-gradients au point π *(noté* $\partial L(\pi)$*) est appelé sous-différentiel de L au point* π.

Le schéma standard de la méthode de sous gradient de base pour résoudre (D) est le suivant :

$$\pi^{k+1} = \text{Proj}\left(\pi^k + s_k.\xi^k\right) \tag{3.2.1}$$

où les coeffficients $\{\pi_j^k\}$ sont les multiplicateurs de Lagrange et ξ^k est le sous gradient de L au point π^k, s_k est le pas, et Proj est l'operateur projection de $\mathbb{R}^m$ sur $\mathbb{R}^m$ c'est-à-dire

$$\text{Proj}(\pi) = \max(0, \pi) = (\max(0, \pi_1), ..., \max(0, \pi_m))^T$$

Algorithme	
Etape 0 :	(initialisation) Choisir π^0
Etape 1 :	à l'itération k $L(\pi^k) = f(y^k) + \pi^k.g(y^k) = \min x \in S\{f(x) + \pi^k.g(x)\}$ calculer ξ^k si $\xi^k = 0$ stop (π^k est la solution optimal de (D)) définir π^{k+1} par : $\pi^{k+1} = \text{Proj}\left(\pi^k + s^k\xi^k\right)$ et aller à l'étape 1

$$\text{où} \qquad s^k = \rho\frac{L(\pi^*) - L(\pi^k)}{\left\|\xi^k\right\|^2} \tag{3.2.2}$$

le coéfficient de relaxation ρ vérifie : $0 < \rho < 2$.

$L(\pi^*)$ n'étant pas connue, si l'on remplace dans 3.2.2 $L(\pi^*)$ par une estimation par défaut $L_1 \leq L(\pi^*)$ polyak 1969 a montré :

– $L(\pi)$ converge vers $L(\pi^*)$

– ou bien, en un nombre fini d'itération on obtien un point π^j vérifiant : $L_1 \leq L(\pi^j) \leq L(\pi^*)$

En pratique, on choisit pour L_1 la valeur de $f(x)$ correspondant à la meilleure solution de (P) obtenue durant les étapes précédentes du calcul.

choix des coefficients ρ plusieure stratégies peuvent être envisagées [1]. Help et al (1974) ont rapporté une expérience satisfaisante avec la règle R1 : legendre et minoux en 1977 ont déterminé les coefficients ρ de façon dynamique en tenant compte de la progression de la fonction L pour proposer la règle R2.

– R1 : Prendre $\rho = 2$ pendant les m (nombre de composantes de π) premières itérations ;puir diviser par 2 la valeur de ρ et le nombre d'itérations jusqu'à atteindre une limite inferieure q (fixée à l'avance) du nombre d'itérations ; enfin diviser par 2 la valeur de ρ toutes les q itérations (généralement $q \simeq 5$) jusqu'à ce que les π^k soient suffisamment petits (ε fixé).

– R2 : au départ, faire $\rho = 2$;

si $L(\pi^k) > \min\{L(\pi^{k-r}), r = 1, 2, ..., k\}$ alors $\rho_{k+1} = \rho_k$

sinon $\rho_{k+1} = \alpha.\rho_k$(où α vérifie : $0 < \alpha < 1$)

Plusieurs remarques sur les limitations de ces méthodes peuvent être faites :

1. On ne dispose pas de test d'arrêt naturel permettant de détecter que l'on est suffisamment proche de la solution.
2. Ces méthodes ne garantissent pas la descente de l'objectif entre deux itères consécutifs, pouvant mener à des phénomènes oscillatoires et à de mauvaises performances numériques.
3. Les méthodes de mise à jour des longueurs de pas les plus efficaces sont parfois dépourvues de résultats de convergence.
4. La vitesse de convergence est souvent faible, en particulier lorsque l'on s'approche de la solution.

Malgré ces limites, ces méthodes restent très largement utilisées par les praticiens, notamment dans la communauté combinatoire.Ce succès est du en partie à leur simplicité d'implémentation, mais également à la qualité de leur comportement numérique pour des problèmes de grande taille.

Application de la méthode du sous- gradient au problème du (TSP) : En relaxant les contraintes de (P) de type (2) on obtient le programme relaxé (L_π) suivant :

$$(L_\pi)\begin{cases} \min L(\pi) = \displaystyle\sum_{1\le i<j\le n} c_{ij}x_{ij} + \sum_{i=2}^{n-1}\pi_i(2 - \sum_{i<j} x_{ij} - \sum_{j<i} x_{ji}) & \\ \displaystyle\sum_{i<j, ij\in S^2} x_{ij} \le |S| - 1 & \text{pour } S \subset \{2, 3, ..., n\} \\ \displaystyle\sum_{j} x_{1j} = 2 & \\ \displaystyle\sum_{1\le i<j\le n} x_{ij} = n & \\ x_{ij} \in \{0, 1\} & \forall i < j \end{cases}$$

La fonction objectif L peut s'écrire :

$$L(\pi) = 2\sum_{i=2}^{n-1}\pi_i + \sum_{1\le i<j\le n}(c_{ij} - \pi_i - \pi_j)x_{ij}$$

On peut facilement voir que le vecteur (ξ^k) suivant est un sous-gradient de L au point π.

$$\xi^k = 2 - \sum_{i<j} x_{ij} - \sum_{j<i} x_{ji}$$

L'algorithme du sous -gradient pour le dual lagrangien est donné par :

L'algorithme de sous gradient pour le dual lagrangien	
Etape 0 :	(initialisation)
	Choisir π^0
Etape 1 :	
	Résoudre le problème lagrangien
	$(L_{\pi^k}) = \min_{x\in X} \sum_{1\le i<j\le n} (c_{ij} - \pi_i^k - \pi_j^k)x_{ij} + 2\sum_{i=2i}^{n-1} \pi^k$
	Calcul de x^k, (on utilise la méthode du 1-arbre)
	calculer $\xi^k = 2 - \sum_{i<j} x_{ij} - \sum_{j<i} x_{ji}$
	si $\xi^k = 0$ stop (π^k est la solution optimal de (D))
Etape 2 :	calculer :
	$\pi^{k+1} = \text{Proj}\left(\pi^k + s^k\xi^k\right)$
	où $s^k > 0$, le pas $\quad s^k = \rho\frac{L(\pi^*)-L(\pi^k)}{\left\|\xi^k\right\|^2}$, $0 < \rho < 2$
Etape 3 :	
	poser : $k = k + 1$ et aller à l'étape 1

Exemple 3.2.1 *On suppose que la solution optimal n'est pas connu.*

On prend l'exemple de 5 villes avec la matrice de distance suivante :

$$C = \begin{bmatrix} - & 11 & 17 & 21 & 12 \\ 11 & - & 11 & 22 & 14 \\ 17 & 11 & - & 19 & 10 \\ 21 & 22 & 19 & - & 9 \\ 12 & 14 & 10 & 9 & - \end{bmatrix}$$

On applique la méthode du plus proche voisin pour obtenir une borne superieure (zub) d'où :

$$zub = 62$$

et on a trouvé le tour $1-2-3-5-4-1$ *de coût* 62

$$\begin{aligned}
\pi^{k+1} &= \pi^k + s^k.\xi^k \\
\pi^{k+1} &= \pi^k + \rho\frac{z_{ub} - L(\pi^k)}{\left\|\xi^k\right\|^2}.\xi^k \\
\pi^{k+1} &= \pi^k + \frac{z_{ub} - L(\pi^k)}{\left\|2 - \sum_{i<j} x_{ij} - \sum_{j<i} x_{ji}\right\|^2}\left(2 - \sum_{i<j} x_{ij} - \sum_{j<i} x_{ji}\right)
\end{aligned}$$

Itération 1 *:* $\pi^1 = (0, 0, 0, 0, 0)$

$$c_{ij} = (c_{ij} - \pi_i^1 - \pi_j^1) = \begin{bmatrix} - & 11 & 17 & 21 & 12 \\ 11 & - & 11 & 22 & 14 \\ 17 & 11 & - & 19 & 10 \\ 21 & 22 & 19 & - & 9 \\ 12 & 14 & 10 & 9 & - \end{bmatrix}$$

puis on applique le kruskal pour obtenir l'arbre couvrant minimal

$$\begin{aligned}
(4, 5) &= 9 \\
(3, 5) &= 10 \\
(2, 3) &= 11
\end{aligned}$$

et on cherche les 2 *minimum de sommet* 1

$$\begin{aligned}
\min 1 &= 11 \\
\min 2 &= 12
\end{aligned}$$

d'où

$$L(\pi^1) = 53$$

$$\begin{aligned}
\xi^1 &= 2 - (2, 2, 2, 1, 3) = (0, 0, 0, 1, -1) \\
\pi^2 &= (0, 0, 0, 0.2222, -0.2222)
\end{aligned}$$

Itération 2 *:* $\pi^2 = (0, 0, 0, 0.2222, -0.2222)$

$$
\begin{aligned}
c_{ij} &= (c_{ij} - \pi_i^2 - \pi_j^2) = \begin{bmatrix} - & 11 & 17 & 20.3218 & 12.6782 \\ 11 & - & 11 & 21.3218 & 14.6782 \\ 17 & 11 & - & 18.3218 & 10.6782 \\ 20.3218 & 21.3218 & 18.3218 & - & 9 \\ 12.6782 & 14.6782 & 10.6782 & 9 & - \end{bmatrix} \\
z_{ub} &= 62 \\
L(\pi^2) &= 54.3564 \\
\pi^3 &= (0, 0, 0, 0.7176, -0.7176)
\end{aligned}
$$

Itération 3 : $\pi^3 = (0, 0, 0, 0.7176, -0.7176)$

$$
\begin{aligned}
c_{ij} &= (c_{ij} - \pi_i^2 - \pi_j^2) = \begin{bmatrix} - & 11 & 17 & 20.3218 & 12.6782 \\ 11 & - & 11 & 21.3218 & 14.6782 \\ 17 & 11 & - & 18.3218 & 10.6782 \\ 20.3218 & 21.3218 & 18.3218 & - & 9 \\ 12.6782 & 14.6782 & 10.6782 & 9 & - \end{bmatrix} \\
z_{ub} &= 62 \\
L(\pi^3) &= 55.7917 \\
\pi^4 &= (0, 0, 0, 1.0398, -1.0398)
\end{aligned}
$$

3.3 Méthode du sous-gradient de remplacement

La méthode de sous-gradient nécessite la minimisation de tous les sous-problèmes pour obtenir une direction de déplacement, et cela peut prendre beaucoup de temps notamment pour des problèmes de grande taille. Il est donc souhaitable d'obtenir une bonne direction au moindre coût.

L'idée principale de la méthode du sous-gradient de remplacement est d'obtenir une direction de déplacement sans optimiser tous les sous-problèmes.

Le dual de remplacement est défini par :

$$\tilde{L}(\pi, x) = \sum_{1 \leq i < j \leq n} c_{ij} x_{ij} + \sum_{i=2}^{n-1} \pi_i (2 - \sum_{i<j} x_{ij} - \sum_{j<i} x_{ji}) \tag{3.3.1}$$

où $\pi = \{\pi_1 \pi_2, ..., \pi_n\} \in \mathbb{R}_+^n$ est un vecteur des multiplicateurs de lagrange.

(Ici on ne minimise pas $\tilde{L}$ pour obtenir plus loin un sous-gradient de remplacement)

Le sous-gradient de remplacement correspondant est défini comme suit :

$$\tilde{\xi}^k = 2 - \sum_{i<j} x_{ij} - \sum_{j<i} x_{ji} \tag{3.3.2}$$

On montrera plus loin que $\tilde{L}(\pi, x) \leq L^*$ (où $L^* = V(D)$)

Proposition 3.3.1 *Étant donné un point (π^k, x^k) , si la valeur de l'objectif du dual de remplacement est inférieur à la valeur optimale du dual, i.e,*

$$\tilde{L}^k = \tilde{L}(\pi^k, x^k) < L^* \tag{3.3.3}$$

alors le sous-gradient de remplacement vérifie :

$$0 \leq L^* - \tilde{L}^k \leq \left(\pi^* - \pi^k\right) \tilde{\xi}^k \tag{3.3.4}$$

Preuve. □

D'après la définition du dual de remplacement on a :

$$\tilde{L}(\pi, x^k) = \sum_{1 \leq i < j \leq n} c_{ij} x_{ij}^k + \sum_{i=2}^{n-1} \pi_i . \tilde{\xi}^k = \tilde{L}^k + \left(\pi^* - \pi^k\right) . \tilde{\xi}^k \tag{3.3.5}$$

En utilisant 3.1.1 on en déduit :

$$L(\pi) \leq \tilde{L}(\pi, x) \tag{3.3.6}$$

Ce qui précède est également vrai au point (π^*, x^k),i.e :

$$L^* = L(\pi^*) \leq \tilde{L}(\pi^*, x^k)$$

3.3.5 entraine :

$$L^* \leq \tilde{L}^k + \left(\pi^* - \pi^k\right) . \tilde{\xi}^k$$

Compte tenu de 3.3.3 et 3.3.2, on obtient

$$0 \leq L^* - \tilde{L}^k \leq \left(\pi^* - \pi^k\right) . \tilde{\xi}^k$$

D'où 3.3.4 est démontré.

Procédure de détermination d'une solution Soit (π^{k+1}, x^k), (X, A) l'arbre donné par l'algorithme du kruskal. Pour déterminer x^{k+1} / $\tilde{\xi}(x^{k+1})$ soit un sous-gradient de déplacement il suffit de permuter une arête u $(c(u) = \max_{v \in A}(c(v)))$de l'arbre qui soit de poids maximum par une arête e adjacente à u / $c(e) < c(u)$

Sinon on prend $x^{k+1} = x^k$

Soit l'algorithme de façon formelle

L'algorithme

début
  (π^{k+1}, x^k), (X, A) l'arbre donné par l'algorithme du kruskal
  $x^{k+1} \leftarrow x^k$
  déterminer u / $c(u) = \max(c(v))$
  trouver $e \in U - A$ / $c(e) < c(u)$
  si (e a été trouvé)
    $A \leftarrow A - \{u\} \cup \{e\}$
    Actualiser x^{k+1} ($x_u = 0$ et $x_e = 1$)
  Fin si
Fin l'algorithme

3.3.1 Méthode du sous-gradient de remplacement

Sur la base de la proposition précédente, si l'optimisation appropriée est effectuée de sorte que 3.3.3 soit satisfaite d'une itération à la suivante, puis 3.3.4 est garanti. Cela implique que l'algorithme peut trouver une bonne direction, et la distance entre les solutions courantes et le multiplicateur optimale peut être diminuée étape par étape. Cette idée conduit à la méthode de sous-gradient de remplacement, où seule une solution approchée est requise pour obtenir un sous-gradient de remplacement. Les étapes de base de la méthode de sous-gradient de remplacement sont :

L'algorithme du sous-gradient de remplacement (SSG)	
Étape 0 :	
	On applique une itération de (SG) pour obtenir x^0,$V(L(\pi^0)$,$\xi(\pi^0)$,π^1.
	$x^0 = \arg\min \left[\sum_{1\leq i<j\leq n} c_{ij}x_{ij} + \sum_{i=2}^{n-1} \pi^0(2 - \sum_{i<j} x_{ij} - \sum_{j<i} x_{ji}) \right]$
Étape 1 :	A l'étape k : soit (π^k, x^k) alors $\tilde{L}^k$ est donné par :
	$\tilde{L}^k = \tilde{L}(\pi^k, x^k) = \sum_{1\leq i<j\leq n} c_{ij}x_{ij} + \sum_{i=2}^{n-1} \pi^k(2 - \sum_{i<j} x_{ij} - \sum_{j<i} x_{ji})$
	calculer π^{k+1} =Proj$(\pi^k + s^k\tilde{\xi}^k)$
	(s^k vérifie : $0 < s^k < \rho\frac{L^* - \tilde{L}^k}{\left\|\tilde{\xi}^k\right\|^2}$ où $\rho \in [0,1]$).
	puis : $\tilde{\xi}^k = 2 - \sum_{i<j} x_{ij} - \sum_{j<i} x_{ji}$
	si $\tilde{\xi}^k = 0$ stop (la solution est optimale)
Étape 2 :	Déterminer x^{k+1} (une solution approchée par la procédure1)
	$\tilde{L}(\pi^{k+1}, x^{k+1}) < \tilde{L}(\pi^{k+1}, x^k)$ avec :
	$\tilde{L}(\pi^{k+1}, x^{k+1}) = \sum_{1\leq i<j\leq n} c_{ij}x_{ij}^{k+1} + \sum_{i=2}^{n-1} \pi^{k+1}(2 - \sum_{i<j} x_{ij}^{k+1} - \sum_{j<i} x_{ji}^{k+1})$
Étape 3 :	
	poser : $k = k + 1$ et aller à l'étape 1

Remarque 3.3.1 *Pour le choix du coefficient ρ, On le prendra égal à 1 pendant les n premières itérations puis nous suivons la même stratégie qui a été utilisée dans la procédure(SG).*

Proposition 3.3.2 *le dual du sous-gradient de remplacement est toujours inférieure au dual optimale*

$$\tilde{L}^k < L^*, \text{ pour tout } k \tag{3.3.7}$$

Théorème 3.3.1 *la suite $\left\{\pi^k\right\}_k$ converge versπ^**

$$\left\|\pi^* - \pi^{k+1}\right\| < \left\|\pi^* - \pi^k\right\|, \text{ pour tout } k \tag{3.3.8}$$

Remarque 3.3.2 *On peut facilement démontrer que si $\pi^k = \pi^{k+1}$ et $x^k = x^{k+1}$, alors $L(\pi^k) = L^*$, et x^k π^k est la solution optimale du problème dual*

Trés souvent, pour pouvoir appliquer une méthode de résolution quelconque et notamment une méthode exacte, la connaissance d'une bonne solution de départ est requise d'où le recours à des méthodes approchées et peu couteuses. C'est dans ce sens que nous présentons la méthode du plus proche voisin qui est une méthode trés simple et peu couteuse.

3.4 Le plus proche voisin

C'est une méthode très simple et facilement programmable. Nous l'avons utilisée pour l'obtention d'une solution de départ afin d'exécuter une méthode du sous-gradient.

On construit la tournée (i.e. le cycle hamiltonien) de manière gloutonne, c'est-à-dire pas à pas : On part d'un sommet quelconque, v_0 , on détermine le sommet le plus proche v_1 , puis le sommet v_2 (non encore exploré) le plus proche de v_1 et on continue le processus jusqu'à ce que l'on ait traité tous les sommets. On obtient ainsi une solution approchée du problème du (TSP) qui peut servir de solution de départ à une autre méthode telle la procédure du sous-gradient. Soit l'algorithme de manière formelle :

Algorithme

Début
- choisir un sommet $v_1 \in V$.
- poser $k \leftarrow 1$
- poser : $U = \{v_1\}$.
- tant que $k < n$
  - $k \leftarrow k + 1$.
  - choisir v_k dans $X = V \backslash U$.qui vérifie :
  - $d(v_{k-1}, v_k) = \min_{x \in X} \{d(v_{k-1}, x)\}$
  - faire :$U \leftarrow U . \cup \{v_k\}$
- fin tant que.

Fin de l'algorithme.

Exemple 3.4.1 *On prend l'exemple de 5 villes avec la matrice des distances suivante :*

$$d = \begin{bmatrix} - & 11 & 17 & 21 & 12 \\ 11 & - & 11 & 22 & 14 \\ 17 & 11 & - & 19 & 10 \\ 21 & 22 & 19 & - & 9 \\ 12 & 14 & 10 & 9 & - \end{bmatrix}$$

On applique la méthode du plus proche voisin pour obtenir une tournée (i.e. le cycle hamiltonien)

Etape 1 : (on choisit le sommet 1 comme sommet de départ et on pose : $U = \{1\}$) On calcule le minimum de la première ligne : $\min d(1, j) = 11 = d(1, 2)$

On actualise U $(U \leftarrow \{1, 2\})$

On calcule $\min_{x \notin U} \{d(2, x)\} = d(2, 3)$

On actualise U $(U \leftarrow \{1, 2, 3\})$

A la fin on trouve la tournée :

$$1 - 2 - 3 - 5 - 4 - 1$$

1. **Exemple 3.4.2** *avec un coût :*

$$Z = 62$$

Chapitre 4

Application

Afin de pouvoir apprécier l'efficacité des différentes méthodes et de comparer les méthodes entre elles nous avons programmé toutes les méthodes citées ci-dessus en langage Matlab et utilisé des instances tirées de OR-library(TSPlib actualisée en 2009) [9], [10] et [11]sur unPC Dell sur Intel (R) le Cœur (TM) i3-2350M CPU @ 2.30 GHz (4CPUs) avec 2048MB RAM. Pour déterminer la matrice des plus courtes distances dans le graphe, nous avons utilisé l'algorithme de Dijkstra (DA). Notre but est de comparer deux méthodes qui nécessitent l'exécution de la même procédure de recherche des plus courtes distances donc nous n'avons pas jugé nécessaire de comptabiliser le temps CPU de l'algorithme (DA) dans le calcul des temps CPU des procédures (SG) et (SSG). Pour toutes les procédures, nous dénotons par :

- Z_{opt} :la valeur optimal
- z_{ub} : borne supérieure
- z_{lb} : borne inferieure
- CPU : temps CPU en secondes
- n :nombre des villes
- $n1$:nombre d'itérations

cas : pour Z_{opt} connu Nous avons testé et comparé la méthode du sous-gradient de remplacement et la méthode du sous-gradient. Nous avons utilisés 10 exemples tirés de 'OR-Library. Les tests numériques confirment l'efficacité de la méthode du sous-gradient de remplacement par rapport à la méthode classique du sous-gradient. La procédure (SSG) nous fournit des valeurs proches des valeurs optimales en des temps raisonnables. excépté

pour l'exemple de gil262, la procédure (SSG) donne de meilleurs résultats que ceux qui sont founiest par la méthode (SG) comme le montre le tableau Tab1

Tab 1: la valeur optimal connue

			"SG"			"SSG"		
Prs	**n**	**solopt**	**n1**	**zlb**	**cpu**	**N1**	**zlb**	**cpu**
Pr1	150	6528	1000	6319.3	1070.72	13	6467	0.880829
Pr2	198	15780	500	14096	815.407	64	15703	2.436162
Pr3	130	6110	500	5900.8	363.386	23	6026.9	0.815970
Pr4	52	7542	100	7146	90.5956	71	7505.5	0.468532
Pr5	42	699	20	634.484	18.0348	6	685.750	0.241949
Pr6	17	2085	500	2034.4	26.2345	118	2076.4	0.764680
Pr7	21	2707	500	2678	42.4339	76	2695.8	0.368068
Pr8	262	2378	100	2245.9	202.176	2	2233.6	4.297126
Pr9	144	58537	500	52547	306.750	606	58411	8.656679
Pr10	159	42080	1000	41183	789.125	83	42021	2.040778

2eme cas : pour Z_{opt} non connu Ici on considère le cas qui nous intéresse le plus à savoir la comparaison entre le sous-gradient et l'algorithme du sous-gradient de remplacement quand la valeur optimale est inconnue. La valeur optimale Z_{opt} n'étant, en général, pas connue au préalable ,en pratique, elle est remplacé par Z_{ub} (on applique la méthode du proche voisin), une évaluation par excès .La difficulté dans la méthode du sous-gradient de remplacement, quand Z_{opt} n'est pas connue à l'avance, réside dans la façon de choisir le coefficient de relaxation ρ (l'étape 2 dans l'Algorithme 2) ; si ρ est choisi assez petit, l'algorithme peut converger vers une mauvaise estimation par défaut de Z_{opt}. Au contraire, si la valeur de ρ est choisie relativement élevée, l'algorithme peut diverger. Cependant si on arrive à trouver un bon choix (ou un compromis) nous aurons l'avantage majeur de pouvoir réduire de manière significative le phénomène de zigzag qui est trés présent dans la méthode du sous-gradient.

Tab 2: pour la zopt non connu

			"SG"				"SSG"			
Prs	**n**	**solopt**	**n1**	**zlb**	**zub**	**cpu**	**N1**	**zlb**	**cpu**	**zub**
Pr1	150	6528	500	6272.6	8194.6	266.5852	28	6498.4	1.60517	8194.6
Pr2	198	15780	500	13952	18761	407.4553	101	15684	3.06532	18761
Pr3	130	6110	500	5825.2	7575.3	194.4681	42	6085.7	0.95454	7575.3
Pr4	52	7542	500	7218.5	8980.9	50.22218	111	7541.9	0.56032	8980.9
Pr5	42	699	500	648.399	862.125	42.81746	10	690.83	0.24469	862.125
Pr6	17	2085	500	2014.9	2184	20.84831	134	2084.5	0.37350	2187
Pr7	21	2707	200	2614.5	3331	13.89143	128	2700	0.38968	3333
Pr8	262	2378	500	2268.8	3241.5	727.7586	5	2302.2	2.49247	3241.5
Pr9	144	58537	500	52591	61651	225.6964	769	58516	9.43246	61651
Pr10	159	42080	400	40369	54669	223.4359	183	42058	3.58290	54669

D'après les tables 1et 2, on remarque que les méthodes du sous-gradient et du sous-gradient de remplacement ont des avantages et des inconvénients :

– Les avantages de la méthode du sous-gradient est :

Facilité de choix de coefficient de relaxation.

– Les inconvénients de la méthode du sous-gradient sont :

1. Zig-Zig.
2. Optimiser à chaque fois.

– Les avantages de la méthode du sous-gradient de remplacement sont :

1. Ne résout pas tous les sous-problèmes.
2. Zig-Zig n'est pas aussi important.

– Les inconvénients de la méthode du sous-gradient de remplacement est :

Choix de coefficient de relaxation est difficile.

3iem cas : d'apres 4 itération pour Z_{opt} non connu Pour palier aux inconvénients des deux méthodes citées ci-dessus et afin de profiter des avantages de l'une et de l'autre nous

avons pensé à combiner les deux méthodes : cela consiste à exécuter quelques itérations de la procédure du sous-gradient de remplacement et de terminer avec la méthode classique du sous-gradient .

Tab 3:

			"SG"				"SSG"			
Prs	**n**	**solopt**	**n1**	**zlb**	**zub**	**cpu**	**n1**	**zlb**	**cpu**	**zub**
Pr1	150	6528	500	6272.6	8194.6	266.585230	25	6505.2	4.179923	7881.5
Pr2	198	15780	500	13952	18761	407.455311	117	15762	16.348790	18416
Pr3	130	6110	500	5819.3	7575.3	199.095127	39	6095.2	3.944920	6986.5
Pr4	52	7542	1000	7279.6	8980.9	73.264877	106	7534.1	2.724815	8863.4
Pr5	42	699	500	645.54	862.13	25.711996	9	686.9	0.565813	822.92
Pr6	17	2085	500	2017.6	2184	5.707631	131	2084.6	1.192866	2187
Pr7	21	2707	500	2611.1	3331	8.293951	121	2702.5	1.247735	2933
Pr8	262	2378	4	2246.3	3241.5	2.125752	4	2376	8.785821	2939.2
Pr9	144	58537	500	52593	61651	235.217075	764	58469	49.992727	61494
Pr10	159	42080	400	40504	54669	286.976078	192	42032	16.745572	51092

CONCLUSION

La méthode du sous-gradient de remplacement (SSG) peut donner de meilleures solutions que la méthode classique du sous-gradient (cela est du principalement à l'existence du phénomène de zigzag dans cette dérnière) pourvu que l'on sache choisir convenablement le coefficient de relaxation.Quoique la méthode (SSG) présente l'inconvénient du choix du coefficient de relaxation. Cependant, la combinaison des méthodes (SSG) et (SG) semble être intéressante et constitue une bonne alternative.

Nous aurions souhaité disposé d'une station de calcul pour pouvoir procéder à des tests numériques sur des problèmes de grandes tailles.

Bibliographie

[1] Michel Gondran et Michel Minoux Graphes et algorithmes.

[2] X. Zhao[3], P. B. Luh[4], and J. Wang[5] Communicated by W.B. Gong and D. D. Ya Surrogate subGradient Algorithm for Lagrangian Relaxation[1,2]

[3] Modelisation et resolution de problemes d'optimisation combinatoire issus d'applications spatiales

[4] https ://tel.archives-ouvertes.fr/tel-00009238

[5] http ://www.isima.fr/f4/projets2005/2005-2006/Boussa%EFdBouraoui.pdf RELAXATION LAGRANGIENNE

[6] www.math.u-psud.fr/~montcouq/Enseignements/Apprentis/arbres

[7] Polyak B.T. (1967), A General Method of Solving Extremum Problems, Soviet Mathematics Doklady 8, 593-597.

[8] Held M., Wolfe P., Crouwder H.P. (1974), Validation of Subgradient Optimization' Mathematical Programming 6, 62-88.

[9] Goffin J.L. (1977), On Convergence Rates of Subgradient Optimization Methods, Mathematical Programming 13, 329-347.

[10] Shor N.Z. (1970), Convergence Rtae of the Gradient Descent Method with Dilatation of the Space, Cybernatic 6, 102-108.