MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ ABDELHAMID BEN BADIS DE MOSTAGANEM
FACULTÉ DES SCIENCES EXACTES ET DE L'INFORMATIQUE
DÉPARTEMENT DE MATHÉMATIQUES ET INFORMATIQUE

MÉMOIRE

Master Académique

pour obtenir le diplôme de Master délivré par

Université de Mostaganem

Spécialité "Modélisation, Contrôle et Optimisation"

présenté et soutenu publiquement par

Naima BOUKSARA

le 14 Juin 2018

Etude de la méthode du sous gradient

Encadeur : **Hocine ABLAOUI (MAA, UNIVERSITÉ DE MOSTAGANEM, ALGÉRIE)**

Jury

Behenes,	MAA	Président (Université de Mostaganem, Algérie)
Amir Abdessamad ,	Professeur	Examinateur (Université de Mostaganem, Algérie)

LABORATOIRE DE MATHÉMATIQUES PURES ET APPLIQUÉES
FACULTÉ DES SCIENCES EXACTES ET DE L'INFORMATIQUE (FSEI)
Chemin des Crêtes (Ex-INES), 27000 Mostaganem, Algérie

MASTER

Dédicaces

Je dédie ce travail aux personnes les plus proches et les plus chères à moi : Mes parents, qui ont été toujours là à mes côtés dans le meilleur et dans le pire, je leur dis : Merci pour votre soutien, vos prières et votre bénédiction m'ont été d'un grand secours pour mener à bien mes études, aucune dédicace n'égalera vos sacrifices que vous m'avez donnés. Je le dédie aussi à mes sœurs Nabila, Hafida, Ikram, Marwa et mon frère Wahid.

Je vous remercie pour votre soutien en témoignage de l'attachement, de l'amour et de l'affection que je porte pour vous.

Mes dédicaces sont adressées ainsi à mon mari qui m'a soutenue.

Je n'oublie pas aussi mes chères amies Saliha, Badra, Leila, Nora, Hanane et Zahia, je vous dédie ce travail en témoignage de l'amitié qui nous unit.

Remerciements

Ce modeste travail n'aurait pu voir le jour sans l'aide d'Allah le tout puissant et l'aide d'un grand nombre de personnes.

Je saisis cette occasion pour présenter mes remerciements les plus sincères à mon encadreur **M . Ablaoui** qui m'a apportée son aide et a contribué à l'élaboration de ce mémoire et également lui témoigner ma gratitude pour sa patience et son soutien qui m'a été précieux.

Mes remerciements s'étendent aussi aux membres du jury et je remercie aussi tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce travail.

Table des matières

Liste des abréviations

P : Programme primal
PL : Programme linéaire .
GC : Méthode du gradient conjugué.
SG : Méthode du sous-gradient.
SSG : Méthode du sous-gradient de remplacement.

Introduction

C'est dans les années soixante que les méthodes du sous gradient sont apparues. Elles ont été utilisées pour la résolution des problèmes de minimisation de fonction non continûment différentiables . Les fonctions à minimiser sont convexes (ou concaves dans le cas d'un problème de maximisation) et linéaires par morceaux. En optimisation non différentiable ou plus communément appelée optimisation discrète, les méthodes du sous gradient jouent le rôle des méthodes du gradient en optimisation différentiable. C'est pour cela que nous avons dédié le chapitre 2 à un rappel sur les méthodes du gradient afin de pouvoir, ultérieurement, faire des parallèles avec les méthodes du sous gradient. L'étude d'une méthode du sous gradient est faite, dans ce mémoire, dans le cas particulier du problème du p-médian dans un graphe : un premier chapitre a été consacré à un bref rappel de quelques notions sur les graphes pour faciliter la lecture à un lecteur occasionnel, et tout un chapitre (le chapitre 4) est dédié au problème du p-médian dans un graphe. Cette approche se justifie par le fait que bien que des résultats théoriques sur les méthodes du sous gradient existent, ils n'ont qu'un intérêt théorique : plusieurs paramètres du problème doivent être déterminés de façon expérimentale et cela dépend du problème particulier à résoudre. On y trouve au chapitre 3 une présentation de la méthode du sous gradient et une très brève introduction de la méthode du sous gradient de remplacement. Enfin au chapitre 5, on y dresse quelques résultats numériques de la méthode du sous gradient avec une étude comparative de règles (R1 et R2) de choix de coefficients de relaxation et surtout faire des tests sur une règle R3 que nous proposons et c'est là que réside l'originalité de notre travail. Nous terminons par une brève conclusion et quelques perspectives de travail.

Chapitre 1

Notations et Rappels

1 Introduction

Dans ce chapitre, nous donnons quelques rappels sur les graphes.

2 Théorie des graphes

Définition 1.1 [6] ***graphes non orientés***
graphe $G = (V, E, e)$ *est défini par la donnée :*

— *D'un ensemble fini* $V = \{v_1, v_2, ..., v_n\}$ *dont les éléments sont appelés* ***sommets*** *(Vertices en anglais).*

— *Par un ensemble fini* $E = \{e_1, e_2, ..., e_m\}$ *dont les éléments sont appelés* ***arêtes*** *(Edges en anglais).*

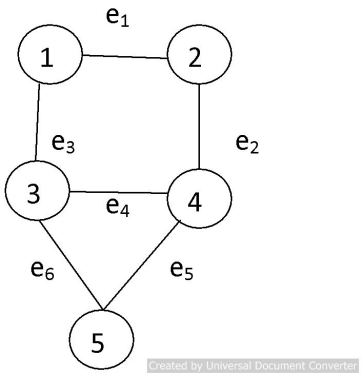
— *Et d'une application :*

$$e : V \rightarrow \{\{a, b\} \setminus a \in X \ \textbf{\textit{et}} \ \ b \in X\}$$

$$e(u) = \{a, b\}; \quad a, b \ \textit{sont applées les extrémités de l'arête } u$$

Soit $e = (a, b)$ *une arête de* E. *Les extrémités* a *et* b *de* E *sont dits* ***adjacents****, ou* ***incidents*** *ou encore adjacents avec* e*, ou bien que l'arête* e *est incidente avec les sommets* a *et* b
On appelle ***ordre*** *du graphe* G *le cardinal de l'ensemble des sommets* V .

Exemple 1.1 *:* [6] *Soit le graphe* $G = (V, E)$.
Soient $V = \{1, 2, 3, 4\}$ *et* $E = \{e_1, e_2, e_3, e_4, e_5\}$



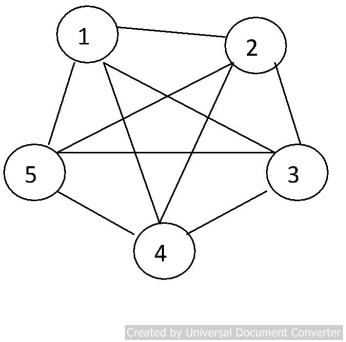
Définition 1.2 [4] ***Chaines et cycles***
Une chaine C *dans* G *reliant deux sommets x et y de* E *est une séquence ordonnée d'arête (ou arcs)* $C = (e_1, e_2, ..., e_\alpha)$ *telle que :*

- *x est adjacent à* e_1.
- e_i *et* e_{i+1} *sont adjacents pour tout* $i = 1, ..., \alpha - 1$.
- *y est adjacent a* e_α.
- *Un cycle* C *est une séquence ordonnée d'arêtes toutes distinctes telles que le sommet initial de la séquence est adjacent au dernier sommet de la séquence.*

Définition 1.3 [6] ***Quelques types de graphes***

1. *Un graphe est dit* ***simple*** *si entre toute paire de sommets il existe au plus une arête qui relie ces deux sommets.*
2. *Un graphe est dit* ***complet*** *si chaque sommet x du graphe est adjacent à tous les autres sommets.*

Exemple 1.2 *Graphe complet*
$V = \{1, 2, 3, 4, 5\}$
$E = \{1,2, 1,3, 1,4, 1,5, 2,3, 2,4, 2,5, 3,4, 3,5, 4,5\}$



Définition 1.4 [6] ***(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 1.5 [6] *(**Un sous-graphe**))*
Un sous-graphe de G consiste à considérer seulement une partie des sommets de V et les arêtes qui relient ces sommets.

Chapitre 2

La méthode du gradient

La méthode du gradient est un outil très puissant qui est utilisé en optimisation "différentiable". Elle fait partie d'une classe plus grande de méthodes numériques appelées méthodes de descente*[17]*.

1 Le principe de la méthode

[17] Soit à résoudre un problème d'optimisation qui se formule comme suit :

$$min_{x\in \mathrm{X}}(f(x))$$

On se donne X^0 comme un point de départ.
On construit l'itérè x^1 pour se rapprocher le plus possible de l'optimum x^*.
On détermine x^1 par la formule : $x^1 = x^0 + \alpha_0 d^0$, où α_0 (pas) est un réel positif et d^0 est le gradient de f au point x_0.
x_2 est déterminé de la même manière que précédemment : $x^2 = x^1 + \alpha_1 d^1$ etc..., et on continue le processus.
Le schéma général d'une méthode du gradient est le suivant :

$$\begin{cases} x^0 \text{ donnée} \\ x^{k+1} = x^k + \alpha_k d^k \end{cases}$$

α_k : Pas de déplacement (qui peut être fixe ou variable)
d^k : Gradient de f au point x_k.

Soit l'algorithme de façon formelle :

Algorithme du gradient

Poser : $k \leftarrow 0$
Choisir : x_0
Tant que $\| x_{k+1} - x_k \| > \epsilon$ et ($k \leq k_{\max}$) faire
  calculer $d^k = -\nabla f(x_k)$
    $d^k = -\nabla f(x_k)$
    α_k (si α_k est variable)
    $x_{k+1} \leftarrow x_k + \alpha_k d^k$
  Faire $k \leftarrow k+1$
Fin tant que
Fin algorithme

Remarque 2.1 *Il y a plusieurs variantes de la méthode du gradient :*

— *Algorithme à pas constant*

— *Algorithme à pas optimal*

1.1 Choix du pas

1. **Gradient à pas constant** *[7]* : On choisit $\alpha_k = \alpha$ pour tout k, avec α suffisamment petit pour garantir la faisabilité de x_{k+1}.
2. **Gradient à pas optimal** *[7]* : On choisit α_k de sorte que la valeur de $f(x_{k+1})$ soit la diminue le plus possible. Autrement dit, α_k est choisi comme solution du problème :

$$inf_{x\in\mathbb{R}_+} q(\alpha) \quad \text{Avec} \quad q(\alpha) = f(x_k + \alpha d^k);$$

avec $d^k = -\bigtriangledown f(x_k)$.

Nous présentons ci-dessous, l'algorithme du gradient dans le cas où notre fonction à optimiser est quadratique.

2 Le cas quadratique

2.1 Gradient à pas fixe

[7] On considère la fonction f de $\mathbb{R}^n$ dans $\mathbb{R}$ définie par :

$$f(x) = \frac{1}{2}x^{\mathrm{T}}\mathrm{Q}x - x^{\mathrm{T}}b$$

où Q une matrice symétrique définie positive d'ordre n et un vecteur de $\mathbb{R}^n$
On considère l'algorithme du gradient à pas fixe :

Algorithm 1 : Gradient à pas fixe

Input :x^0 (première approximation de la solution cherchée), Q, b, α (pas), ϵ
n_{max}(nombre maximal d'itérations)
output : x^*, k
Initialisation
$x^k \leftarrow x^0$
Pour :$k = 1 : n_{max}$ do
  $d^k \leftarrow b$ - $\mathrm{Q}x^k$ (direction de descente) ;
  $x^{k+1} \leftarrow x^k + \alpha d^k$;
  Si $\|d^k\| < \epsilon$ & $k < n_{max}$ (critère d'arrêt) Faire
    La méthode converge après k itérations ;
  Sinon
    $x^k \leftarrow x^{k+1}$;
  Fin
Fin

2.2 Gradient à pas optimal

[7] La méthode du gradient à pas optimal s'écrit :

$$\begin{cases} x^0 & \text{donnée} \\ x^{k+1} = x^k + \alpha_k d^k \end{cases}$$

où $\alpha_k = Argmin(q(\alpha)),\ -$**et** $q(\alpha) = f(x_k + \alpha d^k)$- réalise le minimum sur $\mathbb{R}^n$de la fonction q. Contrairement à la méthode du gradient à pas fixe où le pas est choisi une fois pour toute, dans la méthode du gradient à pas optimal nous avons à résoudre, à chaque itération, un problème d'optimisation pour déterminer le pas de déplacement. Cette manière de procéder est couteuse par rapport à la méthode du gradient à pas fixe mais elle fait diminuer la fonction de descente au maximum.

Soit l'algorithme de manière formelle :

Algorithme 2 : Gradient à pas optimal

```
    input : x_0, ε, Q, b, n_max
    output : x*
1   Initialisation ;
2   x_k ← x_0
3   d_k = b − Q x_k
4   α_k ← ‖d_k‖² / (d_k^T Q d_k)
5   k ← 0
6   tant que : k < n_max & ‖d_k‖ > ε
7   |  k ← k + 1
8   |  x_{k+1} ← x_k + α_k d_k
9   |  d_{k+1} ← b − Q x_{k+1}
10  |  α_{k+1} ← ‖d_{k+1}‖² / (d_{k+1}^T Q d_{k+1})
11  End
```

Théorème 2.1 *Soit* C *un ensemble convexe de* $\mathbb{R}^n$ *et* f *une fonction convexe de* $C \in \mathbb{R}$
Alors tout minimum local est minimum global.

Preuve. Raisonnons par l'absurde
Supposons : $\exists x^* \in C \diagup x^*$ est un minimum local de f sur C et x^* n'est pas un minimum global.
Donc : $\exists r > 0 \diagup \forall x \in B(x^*, r) \cap C, f(x^*) \leqslant f(x)$.
et

$$\exists y \in C : f(y) < f(x^*) \qquad (*)$$

Alors : $\forall \lambda \in [0,1]$

$$(1-\lambda)x^* + \lambda y \in C$$

comme f est convexe donc :

$$f((1-\lambda)x^* + \lambda y) \leqslant (1-\lambda)f(x^*) + \lambda f(y)$$

d'après (*) on tire

$$f((1-\lambda)x^* + \lambda y) < (1-\lambda)f(x^*) + \lambda f(x^*)$$

D'où

$$f((1-\lambda)x^* + \lambda y) < f(x^*)$$

Pour λ suffisamment petit on a :

$$(1-\lambda)x^* + \lambda y \in B(x^*, r) \cap C$$

et donc x^* n'est pas un minimum local. Cela contredit l'hypothèse.
■

2.3 La méthode du gradient conjugué

La méthode du gradient conjugué est utilisée, surtout, pour résoudre des systèmes d'équations linéaires dont la matrice est symétrique définie positive. Cette méthode, imaginée en 1950 simultanément par Cornelius Lanczos, Eduard Stiefel et Magnus Hestenes1, est une méthode itérative qui converge en un nombre fini d'itérations (au plus égal à la dimension du système linéaire).
L'idée de la méthode est de construire itérativement des directions $d_0, ..., d_k$ mutuellement conjuguées. A chaque étape k la direction d_k est obtenue comme combinaison linéaire du gradient en x^k et de la direction précédente d_{k-1}, les coefficients étant choisis de telle manière que d_k soit conjuguée avec toutes les directions précédentes.

Soit l'algorithme de manière formelle : :

Algorihme GC

1 **Initialisation** ; $k = 0, x^{(0)}$
2 **Itération** : $g^{(0)} = \nabla f(x^{(0)}) = \mathrm{Q}x^{(0)} - b$
Si $g^{(0)} = 0$ **Stop**
Si non $d^{(0)} = -g^{(0)}$
3 $\alpha_k = \frac{-g^{(k)\mathrm{T}} d^{(k)}}{d^{(k)\mathrm{T}} \mathrm{Q} d^{(k)}}$
4 $x^{k+1} = x^k + \alpha_k d^{(k)}$
5 $g^{(k+1)} = \nabla f(x^{k+1})$
6 **Si** $g^{(k+1)} = 0$ **stop**
7 **Si non**
8 $\beta_k = \frac{g^{(k)\mathrm{T}} \mathrm{Q} d^{(k)}}{d^{(k)\mathrm{T}} \mathrm{Q} d^{(k)}}$
9 $d^{(k+1)} = -g^{(k+1)} + \beta_k d^{(k)}$
10 $k = k + 1$, **aller en 3**
11 **Fin Si**
12 **Fin algorithme**

Exemple 2.1 *1 Appliquons l'algorithme du gradient conjugué dans le cas quadratique au problème quadratique défini par :*

$$minf(x) = \frac{1}{2} x^{\mathrm{T}} \mathrm{Q} x - x^{\mathrm{T}} b$$

où

$$\mathrm{Q} = \begin{pmatrix} 3/2 & 0 & 1/2 \\ 0 & 2 & 1 \\ 1/2 & 1 & 3/2 \end{pmatrix} \quad b = \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix}$$

$k = 0$

— $x_0 = (0,0,0)^{\mathrm{T}}$, $g^{(0)} = \mathrm{Q}x - b = -b$

— *on a* $g^0 \neq (0,0,0)^{\mathrm{T}}$, $d^0 = -g^{(0)}$, $\alpha_0 = 0.2778$

— $x^{(1)} = x^{(0)} + \alpha_0 d^{(0)} = (0.8333, 0, 0.2778)^{\mathrm{T}}$

— $g^1 = \mathrm{Q}x^1 - b$,$\beta_0 = 0.08025$

— $d^{(1)} = (0.463, -0.5556, -0.5864)^{\mathrm{T}}$

$k = 1$

— $\alpha_1 = 0.2187$, $x^{(2)} = (0.9346, -0.1295, 0.1435)^{\mathrm{T}}$

— $g^2 = (-0.0467, -0.1869, 0.1402)^{\mathrm{T}}$

— $\beta_1 = 0.7075$

— $d^{(2)} = (-0.07948, 0.1476, -0.1817)^{\mathrm{T}}$

$k = 2$

— $\alpha_2 = 0.8231$, $x^{()} = (1, 0, 0)^{\mathrm{T}}$

— $g^3 = (0, 0, 0)^{\mathrm{T}}$

Alors $x^* = (1, 0, 0)$

Dans le cas où la fonction à optimiser n'est pas différentiable, c'est qui est le cas le plus souvent dans la pratique (les données sont souvent le résultat d'un nombre fini d'expériences), les méthodes du gradient ne peuvent pas être utilisées. Dans le prochain chapitre nous allons nous intéress au cas non différentielle pour introduire les méthodes du sous gradient.

Chapitre 3

Optimisation non différentiable

Soit *[10]* à résoudre le programme mathématique (p) suivant :

$$(p)\begin{cases} minf(x) \\ g_i(x) \leqslant 0 \qquad (i = 1.....m) \\ x \in \mathrm{S} \end{cases}$$

où : f et $g_i \quad (i = 1, ..., m)$ sont des fonctions quelconques et non nécessairement différentiables. $\mathrm{S} = \{a_1, a^2, ...a^k\}$ est un ensemble fini discret .
S est appelé ensemble des solutions du problème. Une solution $x \in \mathrm{S}$, est dite réalisable si elle vérifie de plus les contraintes du problème (p).
Pour résoudre (p), en peut s'intéresser aux méthodes approchées pour déterminer des "bonnes solutions" du problème. Des méthoded spécifiques à chaque type particulier de problèmes peuvent être développées (problème de recherche de chemins Euleriens, ou Hamiltomiens,...etc) et très souvent lorsque les contraintes sont linéaires, on utilise les méthodes de relaxations lagrangiennes que l'on présente ci-dessous.

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 objective de sorte qu'elles pénalisent la valeur des solutions qui violent ces dernières *[10]*.
On définit la fonction de Lagrange L par :

$$\mathrm{L}(\pi, x) = f(x) + \pi * g(x)$$

où : $\pi = (\pi_1, \pi_2, ..., \pi_m)$ et $g(x) = [g_1(x), g_2(x), ..., g_m(x)]^{\mathrm{T}}$.

$$(\mathrm{L}_\pi) : \mathrm{L}(\pi) = min_{x \in \mathrm{S}} \mathrm{L}(\pi, x)$$

La fonction duale L est en général une fonction concave continue mais non différentiable.
Nous devons résoudre alors le problème duale de (L_π) :

$$(\mathrm{D})\begin{cases} max\ \mathrm{L}(\pi) \\ \pi \geqslant 0 \end{cases}$$

Proposition 3.1 [10] *Pour tout* $\pi \in \mathbb{R}^+$, *et pour tout x solution réalisable de (p) on a :*

$$\mathrm{L}(\pi) \leqslant f(x)$$

Preuve. Par définition

$$L(\pi) \leqslant f(x) + \sum_{i=1}^{m} \pi_i g_i(x) \qquad \forall \pi \geqslant 0, \qquad \forall x \in S.$$

Comme $\pi_i \geqslant 0$ et $g_i(x) \leqslant 0$, pour tout x solution réalisable de (p) on a :

$$L(\pi) \leqslant f(x)$$

Pour x^* minimum absolu de (p) et pour π^* optimum de (D) :

$$L(\pi) \leqslant L(\pi^*) \leqslant f(x^*)$$

■

2 Résolution du problème Dual par une méthode du sous-gradient

La méthode du sous gradient est une généralisation naturelle de la méthode du gradient au cas non différentiable en remplaçant le gradient par un sous gradient arbitraire. Ces méthodes ont été introduites dans les années 60 par Shor *[10]* pour la minimisation sans contrainte de fonction convexe.

Définition 3.1 [10]

1. *Soit* $\gamma = (\gamma_1, \gamma_2, ..., \gamma_m)^T$ *un vecteur de* $\mathbb{R}^m$. *Il est dit sous-gradient de* L *au point* π *si :*

   — $\forall \pi' \in \mathbb{R}^{m+}$ *on a :*
   $$L(\pi') - L(\pi) \leqslant (\pi' - \pi)\gamma \quad \textbf{\textit{si f concave}}$$
   — $\forall \pi' \in \mathbb{R}^{m+}$ *on a :*
   $$L(\pi') - L(\pi) \geqslant (\pi' - \pi)\gamma \quad \textbf{\textit{si f convexe}}$$

2. *L'ensemble des sous-gradients de* L *au point* π *est un ensemble convexe noté* $\partial L(\pi)$. *Il est appelé sous-différentiel de* L *en* π

Théorème 3.1 [10] *Pour* $\pi \in \mathbb{R}^{m+}$ *soit :*

$$Y(\pi) = \left\{ y^k \in S, f(y^k) + \pi g(y^k) = L(\pi) \right\}$$

Alors :
$\forall y^k \in Y(\pi), g(y^k) \in \partial L(\pi)$ ($g(y^k)$ *est un sous-gradient de* L *en* π

Preuve. Par définitions de L, on a pour tout $\pi' \in \mathbb{R}^m$:

$$L(\pi') \leqslant f(x) + \pi' g(x) \qquad \forall x \in S$$

En particulier, pour $y^k \in Y(\pi)$:

$$L(\pi') \leqslant f(y^k) + \pi' g(y^k) \qquad \forall x \in S$$

$$y^k \in Y(\pi) \Rightarrow L(\pi) = f(y^k) + \pi g(y^k)$$

Par soustraction on en déduit :

$$L(\pi') - L(\pi) \leqslant (\pi' - \pi) g(y^k) \qquad \forall \pi' \in \mathbb{R}^{m+}$$

Ce qui prouve que $g(y^k) \in \partial L(\pi)$ ■

Remarque 3.1 [10]

1. *La fonction* L *est concave, donc on sait que tout optimum local est un optimum global.*
2. *La fonction* L *n'est pas partout différentiable, on ne peut lui appliquer les méthodes classiques de la programmation non linéaire.*

2.1 Algorithme du sous-gradient *[10]*

On cherche à généraliser les méthodes du gradient classiques en prenant, à chaque itération, comme direction de déplacement $\gamma(\pi)$: le gradient, lorsque la fonction est différentiable au point π où l'on se trouve et un sous-gradient (quelconque) $\gamma(\pi) \in \partial L(\pi)$ lorsque la fonction L n'est pas différentiable en π. Une telle procédure a au moins le mérite de la simplicité puisque le calcul de $L(\pi)$, en tout point, procure un sous-gradient de L en π.
Elle se justifie, de plus, par le fait que $\gamma(\pi)$ est une direction de diminution stricte de la fonction $d(\pi)$, distance de π à l'ensemble Q des points optimaux :

$$Q = \{\pi \in \mathbb{R}^m \setminus L(\pi) = L(\pi^*)\}$$

Cependant, l'utilisation de la direction de déplacement $\gamma(\pi)$ ne garantit pas l'augmentation de la fonction L elle-même.
Pour cette raison il est inutile de chercher à optimaliser L dans la direction $\gamma(\pi)$, et à chaque étape j, la valeur du déplacement λ_j sera choisie à priori.

L'algorithme est le suivant *[10]* :

1. **A l'étape 0** : A partir de π^0 ($\pi^0 \in \mathbb{R}^{m+}$) définir une suite $(\lambda_j)_{j\geqslant 0}$ telle que $lim(\lambda_j)_{j\to+\infty} = 0$
2. **A l'étape j :**
   On est au point π^j.
   Calculer :
   $$L(\pi^j) = f(y^j) + \pi^j * g(y^j) = Min_{x\in S}\{f(x) + \pi^j g(x)\}$$
   $y(\pi^j) = g(y^j)$ est un sous-gradient de L en π^j
3. Définir π^{j+1} par :
   Si $\pi^{j+1} \in \mathbb{R}^{m+1}$
   | projet π^{j+1} sur $\mathbb{R}^{m+1}$
   Fin si
   Aller en (2)

2.2 Convergence

Théorème 3.2 [10] *Si* $(\lambda_j)_j$ *converge vers* 0 *et si* $\sum_{j=1}^{\infty} \lambda_j = +\infty$
Alors : $L(\pi^j)$ *converge vers* $L(\pi^*)$ *(Polyak 1987)*[13]. *Et si on choisit, à chaque étape,* λ_j *définie par :*

$$\lambda_j = \rho_j \frac{L(\pi^*) - L(\pi^j)}{\| y(\pi^j) \|} \tag{3.1}$$

Avec $\varepsilon < \rho_j < 2 \ \forall j$ $(\epsilon > 0)$ *alors la convergence est géométrique (Polyak 1969)*[14].

En pratique la valeur de $L(\pi^*)$ *n'est pas connue et si dans (3.1),* $L(\pi^*)$ *est remplacée par* $L_1 (L_1 < L(\pi^*))$ *(Polyak 1969)* [14] *a montré que : soit* $L(\pi^j)$ *converge vers* $L(\pi^*)$ *ou bien, en un nombre fini d'itérations, on obtient un point* π^j *qui vérifie :* $L_1 \leqslant L(\pi^j) \leqslant L(\pi*)$. *Ceci se produit en particulier si* $\rho_j = 2, \forall\, j$.

2.3 Choix de ρ

Plusieurs stratégies peuvent être envisagées *[10]*. Help et Al (1974) ont rapporté une expérience satisfaisante avec la règle R1 : Legendre et Minoux ont (1977) *[11]* déterminé les coefficients ρ de façon dynamique en tenant compte de la progression de la fonction L pour proposer la règle R2.

Règle R1 :
Prendre $\rho = 2$ pendant les m (nombre de composantes de π) premières itérations ; puis diviser par 2 la valeur de ρ et le nombre d'itérations jusqu'à atteindre une limite inférieure 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 (fixés).

Règle R2 :
Au départ, on prend $\rho = 2$;
Si $L(\pi^j) > Min\{L(\pi^{j-r}), r = 1, 2, .., j-1.\}$ Alors $\rho_{j+1} = \rho_j$
Sinon $\rho_{j+1} = \alpha\rho_j (\alpha < 1)$

Comme nous allons le voir plus loin, au chapitre 5, la règle R1 s'avère être beaucoup plus appropriée que la règle R2 dans le cas du problème du p-médian dans un graphe. Nous avons pensé à une autre règle, et c'est là que réside l'originalité de notre travail, qui consiste à "combiner" les deux idées contenues dans les règles R1 et R2 pour proposer la règle R3 suivante :

Règle R3 :
Initialisation :
Choisir une valeur du nombre d'itérations q ($q \simeq 10$).
Poser $\rho = 2$ pendant q itérations.
Détermination de ρ_{j+1} et du nombre d'itérations :
A l'étape j on a : le nombre d'itérations q et ρ_j
Si $L(\pi^j) > \min_{1 \leq r \leq j-1} L(\pi^{j-r})$ alors
  $\rho_{j+1} \leftarrow \rho_j$
  $q \leftarrow q - q_1$ $(q_1 \simeq 5)$
Sinon
  $\rho_{j+1} \leftarrow \rho_j * \alpha$;$\alpha \in]0, 1[$ $(\alpha \simeq 0.2)$
  $q \leftarrow q + q_1$ $(q_1 \simeq 5)$
Fin si

3 Méthode du sous-gradient de remplacement

La méthode du sous-gradient nécessite la résolution de tous les sous-problèmes pour obtenir une direction de recherche. Cela peut nécessiter beaucoup de temps pour des problèmes de grande taille. Par conséquent, il est souhaitable d'obtenir une direction appropriée avec moins d'efforts. L'idée principale de la méthode du sous-gradient de rem-

placement est d'obtenir une bonne direction de déplacement sans avoir à résoudre tous les sous-problèmes *[16]*.
Comme une extension du dual dans (p), le dual de substitution est introduit,

$$\widetilde{L}(\lambda, x) = \sum_{i=1}^{m} f_i(x) + \lambda^T g(x); \qquad x \in S$$

Comparé au dual, le dual de substitution ne nécessite pas la solution de tous les sous-problèmes. Le sous-gradient de substitution correspondant est défini par :

$$\widetilde{g}(x)$$

La proposition suivante stipule que, lorsque la valeur du dual de remplacement est inférieure à la valeur optimale du dual L^*, le sous-gradient de remplacement et la direction de déplacement λ^* forment un angle aigu. on obtient ainsi une bonne direction.

Proposition 3.2 [16] *Pour* (λ^k, x^k) *donnés, si :*

$$\widetilde{L}^k = \widetilde{L}(\lambda^k, x^k) < L^*$$

Alors le sous-gradient de remplacement satisfait :

$$0 \leqslant L^* - \widetilde{L}^k \leqslant (\lambda^* - \lambda^{(k)})^T \widetilde{g}(x^k)$$

L'algorithme du sous gradient de remplacement s'énonce comme suit *[16]* :

— **Etape 0** : Initialiser. Etant donné λ^0 et résolvons les sous-problèmes pour obtenir x^0,

$$x^0 = argmin_{x \in S}(f(x) + (\lambda^{(0)})g(x))$$

Poser : le dual de remplacement égal au dual :

$$\widetilde{L}(\lambda^0, x^0) = L(\lambda^0)$$

Et poser le sous-gradient de remplacement égal le sous gradient :

$$\widetilde{g}(x^{(0)}) = g(\lambda^0)$$

— **Etape 1** : Actualisation des multiplicateurs :
A l'itération k, nous avons le point courant $(L^{(k)}, x^{(k)})$, le dual de remplacement est donné par :

$$\widetilde{L}^{(k)} = \widetilde{L}(\lambda^{(k)}, x^{(k)}) = f(x^{(k)}) + (\lambda^{(k)})\widetilde{g}(x^{(k)})$$

avec

$$x^{(k)} \in S$$

Les multiplicateurs lagrangiens sont actualisés de la façon suivante :

$$\lambda^{(k+1)} = \lambda^{(k)} + \theta^{(k)}\widetilde{g}^{(k)}$$

où :

$$\widetilde{g}^{(k)} = \widetilde{g}^{(k)}(x^{(k)}) = g(x^{(k)})$$

$\theta^{(k)}$ est le pas de déplacement qui doit satisfaire :

$$0 < \theta^{(k)} < \frac{(L^* - \widetilde{L}^{(k)})}{\| \widetilde{g}^{(k)} \|^2}$$

— **Etape 2** :
Etant donné $\lambda^{(k+1)}$, trouver une solution $x^{(k+1)}$ qui vérifie :

$$\widetilde{L}(\lambda^{(k+1)}, x^{(k+1)}) < \widetilde{L}(\lambda^{(k+1)}, x^{(k)}) = f(x^{(k)}) + \lambda^{(k+1)}\widetilde{g}(x^{(k)})$$

Si $x^{(k+1)}$ ne peut pas être obtenu, poser :

$$x^{(k+1)} \leftarrow x^{(k)}$$

Critères d'arrêt :
L'algorithme s'arrête si :

$$\| \lambda^{(k+1)} - \lambda^{(k)} \| \leqslant \epsilon_1$$
$$\| x^{(k+1)} - x^{(k)} \| \leqslant \epsilon_2$$

Le temps CPU est supérieur à une valeur maximum CPU_{max}, où le nombre d'itérations dépasse une valeur maximum k_{max}.

L'application de la méthode du sous gradient de remplacement nous garantit que :

$$\widetilde{L}^{(k)} < L^* \quad pour\ tout\ itération\ k$$

Nous avons donc une bonne direction de déplacement.
Comme nous avons le théorème de convergence suivant :

Théorème 3.3 [16] *Pour tout itération k :*

$$\| \lambda^* - \lambda^{(k+1)} \| < \| \lambda^* - \lambda^{(k)} \|$$

Choix du coefficient de relaxation : *[10]*
Dans la méthode du sous gradient de remplacement, le pas de déplacement $\theta^{(k)}$ est donné par :

$$\theta^k = \rho \frac{(L^* - \widetilde{L}^{(k)})}{\| \widetilde{g}^{(k)} \|^2} \tag{3.2}$$

avec $0 < \rho < 1$.
En pratique la valeur de $L(\pi^*)$ n'est pas connue, elle sera remplacée par Z_{ub} dans (3.2) et plus exactement, $\theta^{(k)}$ dans (3.2) sera remplacée par :

$$\theta^k = \rho \frac{(Z_{ub} - Z_{lb})}{\| \widetilde{g}^{(k)} \|^2} \tag{3.3}$$

où :

— Z_{ub} est une borne supérieur de L^*.
— $Z_{lb} = max_{0 \leqslant j \leqslant k}(\widetilde{L}^j)$: meilleure valeur trouvée de $(\widetilde{L}^j)_{0 \leqslant j \leqslant k}$ jusqu'à l'itération k.

Pour le choix du coefficient de relaxation ρ, nous pouvons adapter la même démarche que pour la méthode du sous gradient, à savoir les règles R1 et R2 et surtout R3.

Chapitre4

Chapitre 4

Un problème d'optimisation non différentiable : Problème du p-médian

1 Problème du p-médian

1.1 Présentation du problème*[2]*

On suppose qu'on ait à satisfaire N demandes localisées pour un certain service.
Ce service est effectué par des points qu'on appellera "centre de services". Le problème consiste à déterminer la localisation optimale de p centres de services "p variant de 1 à N de sorte que le "coût" de satisfaction de l'ensemble des demandes soit le plus petit possible.
Le problème du p-médian possède énormément d'applications pratiques. Que ce soit en informatique (routiers, aériens ou téléphoniques). Handler et Mirchandani ont dressé une liste très variée des applications potentielles du modèle comme les décisions de localisation pour les **services d'urgence** (police, pompiers, médicales), les réseaux **informatiques** et de **communication** (localisation des fichiers informatiques sur une série de serveurs identifiés), les **applications militaires** (centres stratégiques), les **activités de service public ou privé** (les magasins, centres commerciaux, potes), les **les activités de transport** (arrêts de transport en commun, entrepôts), **l'intelligence artificielle et les modèles statistiques** (partition de nuage de points).

1.2 Formulation mathématique du p-médian *[5]*.

Soit G = (V; E) un graphe non orienté qu'on supposera, sans perte de généralité, simple et sans boucles.
d^{ij} : Longueur minimale entre les sommets i et j
w_{ij} : Le poids du sommet j
En général, on pose $c_{ij} = w_{ij}d_{ij}$ coût de "satisfaction" de la demande totale du sommet j par le médian i.
On définit les variables par :

$$x_{ij} = \begin{cases} 1 & \text{si } j \text{ servi par le sommet } i, i \in \mathrm{V}, j \in \mathrm{V} \\ 0 & \text{si non} \end{cases}$$

$$x_{ii} = \begin{cases} 1 & \text{si } i \text{ est choisi comme médian }, i \in \mathrm{V}, j \in \mathrm{V} \\ 0 & \text{si non} \end{cases}$$

Le problème p-médian peut être formulé en un programme linéaire $(p.l)$ suivant :*[5]*

$$Min(Z) = \sum_{i \in V} \sum_{j \in V} W_j d_{ij} x_{ij} \tag{4.1}$$

$$\sum_{i \in V} x_{ij} = 1 \quad \forall j \in V \tag{4.2}$$

$$\sum_{i \in V} x_{ii} = p \tag{4.3}$$

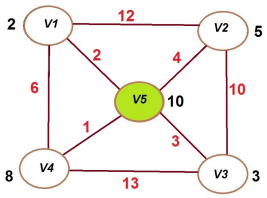
$$x_{ij} - x_{ii} \leq 0 \qquad \forall i, \forall j \in V \tag{4.4}$$

$$x_{ij} \in \{0,1\} \tag{4.5}$$

Interprétation des contraintes :*[5]*

1. (4.1) Fonction objectif du problème du p-médian .
2. (4.2) Signifie qu'un sommet j n'est servi que par un médian.
3. (4.3) Signifie que le nombre de médians est fixé à p
4. (4.4) Signifie qu'un sommet j n'est servi par un sommet i que si i est choisi comme médian

Exemple 4.1 *On a 5 villes qui sont reliées par 8 routes.*
Le problème consiste à déterminer un lieu où l'on doit implanter un centre de service (dispensaire, école ou autres) tel que le coût total de transport soit le plus petit.



Z(V1) = 89; Z(V2) = 113; Z(V3) = 107; Z(V4) = 53; Z(V5) = 41

2 Algorithme de Dijsktra

Cet algorithme nous donne la matrice des plus courtes distances d'un sommet de départ s vers tous les autres sommets du graphe *[4]*. C'est une procédure itérative qui détermine de proche en proche une plus courte distance du sommet s aux autres sommets du graphe : On démarre d'un vecteur $\pi(x)$ (plus courte distance "provisoire" du sommet s au sommet x) que l'on actualise à chaque itération jusqu'à ce que l'on obtienne le vecteur des plus courtes distances de s à tous les sommets x du graphe. Pour obtenir la matrice des plus courtes distances entre deux sommets quelconques on fera varier le sommet de départ s de x_1 à x_n (ensemble de tous les sommets du graphe).

Soit de manière formelle l'algorithme de Dijsktra :

ALGORITME DE DIJKSTRA

Entrée : G = (X, U), n, d
Initialisation :
  S ← {s}, $\pi(s) = 0, \widetilde{A}(s) = \epsilon, k \leftarrow 1, x_1 \leftarrow s$
Pour tout : $x \in X; x \neq s$, pour tout $\pi(x) = +\infty$ fin pour
  Tant que ($k < n$ et $\pi(x_k) < \infty$)
    Pour tout $u \in U / I(u) = x_k$ et $T(u) = 2S$:
      $x \leftarrow T(u)$
      Actualisation de :π
      Si $\pi(x) > \pi(x_k) + d(u)$
        $\pi(x) \leftarrow \pi(x_k) + d(u), \widetilde{A}(x) \leftarrow u$
      Fin
    Fin pour.
    Choisir $x \notin s / \pi(x) = \min_{y \in S}(\pi(y))$
    $k \leftarrow k + 1, x_k = x; S \leftarrow S \cup \{x_k\}$
  Fin tant que.
Fin algorithme.

3 Méthode exacte

3.1 Dualité lagrangienne *[1]*

Soit $\lambda = (\lambda_j > 0, j = 1, .., n)$ le multiplicateur de Lagrange associé aux contraintes (4.2) de type . Le problème lagrangien est donné par :

$$(L_\lambda) = \begin{cases} V(L_\lambda) = min L(\lambda; x) \\ x \in X \end{cases} \tag{4.6}$$

où L est la fonction lagrangienne

$$L(\lambda, x) = \sum_{i=1}^{n} \sum_{j=1}^{n} (d_{ij} - \lambda_j) x_{ij} + \sum_{j=1}^{n} \lambda_j \tag{4.7}$$

Le dual Lagrangien (D) est donné par :

$$(D) = \begin{cases} V(D) = V(\lambda_x) = max V(L_\lambda) \\ \lambda \geqslant 0 \end{cases} \tag{4.8}$$

Pour $\lambda \geqslant 0$ donné, le problème L_λ est résolu en décomposant pour chaque indice i, (la contrainte (4.3) est considérée implicitement obtenir les n problèmes suivants $(L_\lambda)_{i=1,n}$:

$$(L_\lambda)_{i=1,n} = \begin{cases} V(L_\lambda)_i = min \sum_{j=1}^{n} (d_{ij} - \lambda_j) x_{ij} \\ sous\ les\ contraintes\ (4.4),\ (4.5) \end{cases} \tag{4.9}$$

Pour chaque i, le problème $(L_\lambda)_i$ est résolu par :

$$\delta_i = \sum_{j=1}^{n} \{min(0, d_{ij} - \lambda_j)\} \tag{4.10}$$

et en choisissant I comme l'ensemble des indices correspondants aux p petites valeurs de δ_i :

$$I^* = \{ \text{l'indice des p plus petites valeurs de} \delta_i \} \tag{4.11}$$

La solution x_{ij} du problème $(L_\lambda)_i$ est donnée par :

$$x^*_{ij} = \begin{cases} 1 & sii \in I^* \\ 0 & sinon \end{cases} \tag{4.12}$$

et pour tout indice i, j

$$x^*_{ij} = \begin{cases} 1 & si \quad i \in I^* et \quad d_{ij} - \lambda_{ij} \leqslant 0 \\ 0 & sinon \end{cases} \tag{4.13}$$

Il s'ensuit que $x^* \in X, \delta_i = V((L_\lambda)_i)$ et

$$V(L_\lambda) = \sum_{i \in I^*} \delta_i + \sum_{j=1}^{n} \lambda_j \tag{4.14}$$

3.2 La méthode du sous-gradient *[1]*

La méthode du sous-gradient est fréquemment utilisée lorsqu'un sous-gradient peut facilement être calculé : ce qui est vérifié dans notre cas. En effet si $x^* = (x^*_{ij})_{ij}$ est une solution optimale du problème (L_λ) alors un sous gradient de L au point λ, $g(x) = (g_1(x), g_2(x), ..., g_n(x))$, est donné par :

$$g_j(x^*) = 1 - \sum_{i=1}^{n} x^*_{ij}; \qquad j = 1, 2, ..., n \tag{4.15}$$

La méthode du sous gradient est une méthode itérative où les multiplicateurs de Lagrange, λ_i, sont actualisés comme suit :

$$\lambda_j^{(k+1)} = max(0, \lambda_j^{(k)} + \theta^{(k)} g_j^{(k)}) \qquad j = 1, .., n \tag{4.16}$$

$\lambda^{(k+1)}$ est la projection de $\lambda^{(k)} + \theta^{(k)} g^{(k)}$ sur $\mathbb{R}^n_+$; où $g^{(k)}$ est un sous-gradient de L.
et $\theta^{(k)}$ est le pas de déplacement qui est défini par :

$$\theta^{(k)} = \frac{\pi(V(D) - V(L_{(\lambda^{(k)})}))}{\| g^{(k)} \|^2} \tag{4.17}$$

Où le paramètre $\pi \in]0, 2]$
Soit l'algorithme du Sous-gradient *[1]* de façon formelle :

Algorithme1 : sous-gradient

Initialisation :

Déterminer une valeur initiale de Zub (borne supérieure de (P)).
(On peut utiliser l'heuristique de Tietz Bart ou autre).
poser :$Z_{lb} = -\infty$
(Z_{lb} : Meilleure borne inférieure trouvée durant la procédure.)
Choisir une valeur initiale du multiplicateur Lagrangien $\lambda^{(0)}$
$\lambda_j^{(0)} = \min_{i \neq j}(d_{ij})$ pour tout $j = 1, 2, ..., n$
Choisir $\epsilon > 0$ et le nombre d'itérations maximum $k_{max} > 0$:
Soit $k = 0$:
Tant que $(k < k_{max})$:
  Calculer :$I^*_{(k)}, (x^*_{ij})^{(k)}$ et $V(L_{(\lambda^{(k)})})$
  Donnés par les relations (4.10), (4.11),(4.12), (4.13) et (4.14).
  Actualiser Z_{lb} en posant : $Z_{lb} = \max(Z_{lb}, V(L_{(\lambda^{(k)})})$
  Calculer : $Z = \sum_{j=1}^{n} \min_{i \in I^*_{(k)}}(d_{ij})$
  Mettre à jour la valeur de Z_{ub} par : $Z_{ub} = \min(Z; Z_{ub})$
  Si $((Z_{ub} - Z_{lb}) > \epsilon$
    Calculer le sous-gradient $g^{(k)}$ par la relation (**??**).
    Calculer $\theta^{(k)}$ par la relation (4.17).
    Mettre à jour le multiplicateur Lagrangien par :
    $\lambda_j^{(k+1)} = \max(0, \lambda_j^{(k)} + \theta^{(k)} g_j^{(k)})$ pour tout $j = 1, 2, ..., n$
    $k = k + 1$
  Sinon
    Soit $k = k_{max}$ (La solution courante est optimale).
  Fin Si.
Fin tant que
Fin de l'Algorithme.

3.3 Sous-gradient de remplacement *[16]*

Comme pour la méthode du gradient, on peut facilement envisager des variantes de l'algorithme du sous gradient, et une des variantes que nous présentons, ici, est la méthode du sous gradient de remplacement (surrogate subgradient) qui a été introduite par X.Zhao ; P.B. Luh ; J. Wang (1997) *[15]*. Son principe réside dans le fait de trouver un sous-gradient "approximatif" de L en tout point.
Dans la méthode du sous gradient on est obligé d'optimiser tous les problèmes (pour le cas du problème du p-médian on résout tous les sous problèmes $(L_\lambda)_i$ $i = 1, 2, ..., n$) alors que dans la méthode du sous gradient de remplacement, la résolution d'un seul sous problème suffit (dans notre cas, il suffit de résoudre un sous problème $(L_\lambda)_{i0}$ pour $i0$ quelconque).
Comme une extension de la fonction de Lagrange ($L(\lambda) = min_x L(\lambda, x)$).
La fonction de Lagrange de remplacement est donnée par :

$$\widetilde{L}(\lambda, \overline{x}) = f(\overline{x}) + \lambda g(\overline{x})$$

Pour un point $\overline{x}$ qui vérifie certaines conditions que l'on verra ci-dessous.
Le sous gradient de remplacement, $\widetilde{g}(\overline{x}) = (\widetilde{g}_1(\overline{x}), \widetilde{g}_2(\overline{x}), ..., \widetilde{g}_n(\overline{x}))$, est donné par :

$$\widetilde{g}_j(\overline{x}) = 1 - \sum_{i=1}^{n} \overline{x}_{ij} \qquad j = 1, 2, ..., n$$

La proposition suivante montre que si $\widetilde{L}(\lambda, \overline{x}) < L^*$. Alors le sous gradient de remplacement fait un angle aigu avec la direction de déplacement λ^* et donc on a une bonne direction.

Proposition 4.1 [1] *Soit* $(\lambda^{(k)}, x^{(k)})$ *un point courant si :*

$$\widetilde{L}^{(k)} = L(\lambda^{(k)}, x^{(k)}) \leqslant L^*$$

Alors :

$$0 \leqslant L^* - \widetilde{L}^{(k)} \leqslant (\lambda^* - \lambda^{(k)})^t \widetilde{g}(x^{(k)})$$

De la dernière inégalité on peut dire que, pour $\lambda^{(k)}$ donné, un bon choix de $x^{(k)}$ vérifiant les hypothèses de la propositions, fait de $\widetilde{g}(x^{(k)})$ une bonne direction de déplacement (l'angle entre $\widetilde{g}(x^{(k)})$ et $\lambda^* - \lambda^{(k)}$ est aigu) la distance entre le multiplicateur courant $\lambda^{(k)}$ et l'optimal λ^* diminue étape par étape.
L'avantage majeur de cette méthode réside dans la rapidité d'obtention d'une solution $x^{(k+1)}$ contrairement à la méthode du sous gradient.

Procédure de recherche d'une solution *[1]* Soit $\lambda^{(k+1)} \in \mathbb{R}^n_+$, $x^{(k)}$ une solution courante et $I^*_{(k)}$ un ensemble de médians. Pour déterminer une solution $x^{(k+1)}$ telle que $g(x^{(k+1)})$ soit un sous-gradient de remplacement, nous appliquons la procédure suivante :

Algorithme 2 : Déterminer une solution appoximative
Choisir n'importe quel indice k_1 dans $I^*_{(k)}$ et n'importe quel indice $k_0 \in \widetilde{I}^*_{(k)}$
$\sum_{j=0}^{n} \min(d_{k_0 j} - \lambda_j^{(k+1)}, 0) < \sum_{j=0}^{n} \min(d_{k_0 j} - \lambda_j^{(k+1)}) x_{k_1 j}^{(k)}$
Actualiser l'ensemble $I^*_{(k+1)} = I^*_{(k)} \cup \{k_0\} - \{k_1\}$
pose :

$$x_{ij}^{(k+1)} = \begin{cases} x_{ij}^{(k)} \quad \text{pour tout } i \in I^*_{(k+1)} - \{k_0\} \\ 1 \text{ si } i = k_0 \text{ et } d_{k_0 j} - \lambda_j^{(k+1)} \leq 0 \quad j = 1, 2, ..., n \\ 0 \quad \text{sinon} \end{cases}$$

Si (comme un indice k_0 ne peut pas être obtenu) alors :
$I^*_{(k+1)} = I^*_{(k)}$
choisir un indice k_1 dans $I^*_{(k+1)}$ tel que
$\sum_{j=0}^{n} \min(d_{k_1 j} - \lambda_j^{(k+1)}, 0) < \sum_{j=0}^{n} \min(d_{k_1 j} - \lambda_j^{(k+1)}) x_{k_1 j}^{(k)}$
$k_0 = 0$.

$$x_{ij}^{(k+1)} = \begin{cases} x_{ij}^{(k)} \quad \text{pour tout } i \in I^*_{(k+1)} - \{k_1\} \\ 1 \text{ si } i = k_1 \text{ et } d_{k_1 j} - \lambda_j^{(k+1)} \leq 0 \quad j = 1, 2, ..., n \\ 0 \quad \text{sinon} \end{cases}$$

Si (comme un indice K_1 ne peut pas être obtenu) alors :
$k_0 \leftarrow n + 1$
$x^{(k+1)} \leftarrow x^{(k)}$
Fin si
Fin si
Fin algorithme

Proposition 4.2 [1] *soit* $\lambda^{(k+1)} \in \mathbb{R}_+^n$, $x^{(k)}$ *la solution courante et* $x^{(k+1)}$ *la solution donnée par l'algorithme 2 (quand elle peut être obtenue), alors :*

$$L(\lambda^{(k+1)}, x^{(k+1)}) < L(\lambda^{(k+1)}, x^{(k)}) \tag{4.18}$$

Nous donnons ci-dessous l'algorithme du sous gradient de remplacement *[1]* :

Algorithme3 : Procédure de sous-gradient de remplacement (SSG)

Appliquer une itération de l'algorithme1 pour obtenir
$g^{(0)}, x_{ij}^{(0)}, L_{\lambda^{(0)}}, I_{(0)}, \lambda^{(1)}$
z_{ub} et z_{lb} sont aussi déterminés
Choisir, un nombre maximum d'itération $kmax$ et une valeur de ρ dans $]0,1[$.
pose : $k \leftarrow 0$
Tant que : $(k < kmax)$
déterminer $x^{(k+1)}$ et $I^{(k+1)}$ par algorithme2
Calculer $Z = \sum_{j=1}^{n} \min_{i \in I_{(k+1)}}(d_{ij})$ puis actualiser $Z_{ub} = \min(Z, Z_{ub})$
Calculer :
$L_{(\lambda^{(k+1)})} = \sum_{i \in I^*_{(k+1)}} \sum_{j=1}^{n} (d_{ij} - \lambda_j^{(k+1)}) x_{ij}^{(k+1)} + \sum_{j=1}^{n} \lambda_j^{(k+1)}$.
actualiser $Z_{lb} = \max(Z_{lb}, L_{(\lambda^{(k+1)})})$
Calculer $g^{(k+1)} = 1 - \sum_{i=1}^{n} x_{ij}$
Calculer la taille du pas $\theta^{(k+1)}$ par :
$\theta^{(k+1)} = \frac{\rho(L_{(\lambda^*)} - L_{(\lambda^{(k+1)})})}{\| g^{(k+1)} \|^2}$
$k \leftarrow k+1$
actualiser $\lambda_j^{(k+1)} = \max(0, \lambda_j^{(k)} + \theta^{(k)} g_j^{(k)})$ pour tout $j = 1, 2, ..., n$
Fin tant que
Fin algorithme

Théorème 4.1 [1] *Soit* $L(\lambda^{(k)}, x^{(k)})$ *donné par l'algorithme 4 alors :*

$$L(\lambda^{(k)}, x^{(k)}) < V(D) \quad \textit{pour tout } k \tag{4.19}$$

$$\| \lambda^{(k+1)} - \lambda^{(*)} \| < \| \lambda^{(k)} - \lambda^{(*)} \| \tag{4.20}$$

Comme dans la méthode du sous gradient la valeur optimale de la fonction objectif n'est pas connue au préalable, en pratique les valeurs de $L(\lambda^*)$ sont remplacées, respectivement par une borne supérieure Z_{ub} et une une borne inférieure Z_{lb} dans la formule du calcul de

$$\theta^{(k+1)} = \frac{\rho(L_{(\lambda^*)} - L_{(\lambda^{(k+1)})})}{\| g^{(k+1)} \|^2}$$

. Cette méthode présente l'avantage de réduire sensiblement le phénomène de Zig-Zag qui est présent dans la méthode du sous gradient classique, mais elle présente l'inconvénient majeur de la convergence (la solution donnée par cette méthode peut ne pas converger vers la solution optimale si le coefficient ρ n'est pas bien choisi). Dans l'article *[1]*, une méthode qui combine la méthode du sous gradient classique et la méthode du sous gradient de remplacement est proposée. Elle consiste à exécuter d'abord quelques itérations de la méthode du sous gradient de remplacement et à terminer par la méthode du sous gradient classique.

4 Méthode approchée

Le recours aux méthodes heuristiques est utilisé dans le cas où seules des solutions approchées sont désirées, ou simplement pour l'obtention de "bonnes" solutions réalisables (qui serviront, éventuellement, de points de départ à d'autres procédures telles que les procédures (S.E.P), celles du sous-gradient ou autres).
Plusieurs heuristiques ont été proposées *[2]* :

1. Heuristique de Maranzana
2. Heuristique de Teitz et Bart

4.1 Méthode du Maranzana *[2]*

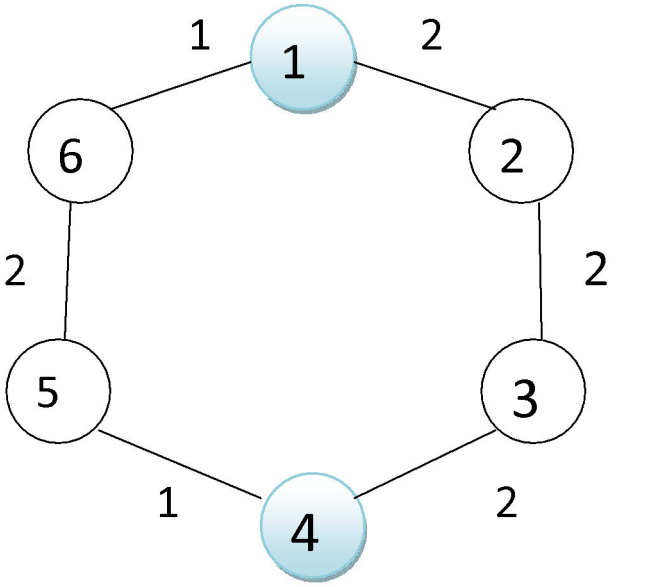
C'est une procédure itérative qui est basée sur le partitionnement de l'ensemble des sommets du graphe.
Elle a été développée par Marazana en (1964). L'algorithme débute par un choix, a priori arbitraire, de P sommets (un choix judicieuxet nécessaire pour l'obtention d'une "bonne" solution si elle n'est pas optimale) et la partition associée (*) de l'ensemble des sommets du graphe. Ces p sommets sont, en fait, pris comme solutions de départ du problème. Dans chaque sous-ensemble (plus exactement sous-graphe engendré) composant la partition, on détermine le 1-médian, et l'ensemble de ces 1-médians constituent la nouvelle solution approchée du problème du p-médian dans le graphe. On répète le processus jusqu'à ce que l'on ne puisse plus changer (et donc améliorer) la solution.

Soit l'algorithme de manière formelle *[2]* :

ALGORITHME DE MARANZANA

Entrée : G = (X; U); W; p : d
Initialisation
  Choisir, aléatoirement, l'ensemble M = $\{x_1, x_2, ..., x_p\}$ de p sommets de X.
  Poser :S = $\{x_1, x_2, ..., x_p\}$
Tant que(M $\neq$ S)
  Poser : S $\leftarrow$ M
  Pour tout $i = 1, 2, ..., p$, faire :
    Calculer l'ensemble P_i par :
    $p_i = \{x \in X / d(x_i, x) \leq d(x_k, x)$ pour tout $k = 1, 2, ..., p\}$
  Fin pour tout
  Pour tout i = 1 ; 2 ; : : : ; p
    Déterminer le médian m_i de P_i par :
    $m_i = \arg\min_{y_i \in p_i} \sum_{x \in p_i} W(x) d(y_i, x)$
  Fin pour tout
  poser : M = $\{m_1, m_2, ..., m_p\}$
Fin tant que
Fin de l'algorithme

Exemple 4.2



$P = 2$

$S_0 = \{2, 4\}$

$1^{ère}$ *itération* :

$p_1 = \{2, 1, 6\}$

$J(1) = 0 + 2 + 1 = 3$

$J(2) = 2 + 0 + 3 = 5$

$J(6) = 1 + 3 + 0 = 4$

$p_2 = \{4, 3, 5\}$

$J(3) = 0 + 2 + 3 = 5$

$J(4) = 2 + 0 + 1 = 3$

$J(5) = 3 + 1 + 0 = 4$

$S_1 = \{1, 4\}$

$2^{ème}$ *itération* :

$S = \{1, 4\}$

$p_1 = \{1,2,6\}$

J(1) = 0 + 2 + 1 = 3

J(2) = 2 + 0 + 3 = 5

J(6) = 1 + 3 + 0 = 4

$p_2 = \{4,3,5\}$

J(3) = 0 + 2 + 3 = 5

J(4) = 2 + 0 + 1 = 3

J(5) = 3 + 1 + 0 = 4

$S_2 = \{1,4\}$

$S_2 = S_1$. *Stop* S = {1,4} *est la solution cherchée et son cout est égal à 6.*

4.2 Heuristique de Teitz et Bart *[2]*

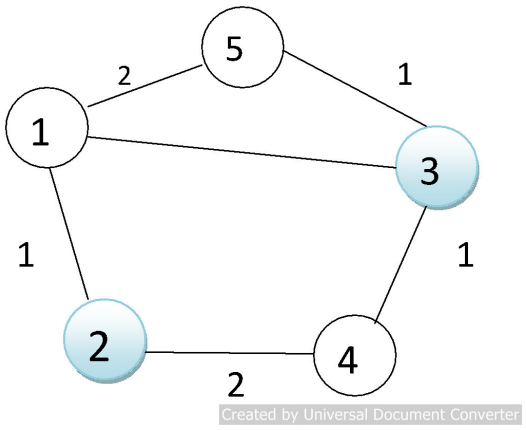
L'heuristique de Teitz et Bart est une procédure itérative qui est basée sur la substitution des sommets : On démarre à partir d'une solution quelconque (ou d'une solution qui a été déterminée par une autre procédure approchée), on procède à la substitution d'un sommet médian par un autre sommet non médian. La solution est acceptée si le coût engendré est meilleur, sinon on rejette la solution. On continue le processus jusqu'à ce que l'on constate qu'aucune substitution n'est possible.

Soit l'algorithme de Teitz et Bart de façon formelle *[2]* :

ALGORITHME : DE TEITZ ET BART

Entrée : $G = (X;U), W, p, d$:
Initialisation
  Choisir, aléatoirement, l'ensemble $S = \{x_1, x_2, ..., x_p\}$ de p sommets de X.
  Calculer : $Q(S) = \sum_{k=1}^{n} W(x_k) d(x_k, S)$
  Poser : $S_0 = \{x_{p+1}, x_2, ..., x_p\}$
Tant que $(S \neq S_0)$
  Poser :$S \leftarrow S$.
  Pour tout $j = p+1, p+2, ..., n$:
    Pour tout $i = 1, 2, ..., p$:
      Calculer : $Q(S_0 \cup \{x_j\} \setminus \{x_i\} = \sum_{k=1}^{n} W(x_k) d(x_k, (S_0 \cup \{x_j\} \setminus \{x_i\}))$
      Calculer : $\triangle_{ij=} Q(S_0) - Q(S_0 \cup \{x_j\} \setminus \{x_i\}$
    Fin pour tout
    Déterminer $\triangle_{ioj}$ par $\triangle_{ioj} = \max_{x_i \in S_0}(\triangle_{ij})$
    Si $\triangle_{ioj} \leq 0$
      $S \leftarrow S_0$
    Sinon
      $S \leftarrow S_0 \cup \{x_j\} \setminus \{x_i\}$
      $Q(S) \leftarrow Q(S_0 \cup \{x_j\} \setminus \{x_i\})$
    Fin Si
  Fin pour tout
Fin tant que
Fin de l'algorithme

Exemple 4.3



Initialisation

$p = 2$

Si on prend $S = \{1, 2\}$

$V - S = \{3, 4, 5\}$

$Q(S) = 0 + 0 + 3 + 2 + 2 = 7$

$1^{ère}$ *itération*

$x_i = 1$

$x_j = 3$

$Q(S \cup \{x_j\} \setminus \{x_i\}) = Q(\{3,2\} = 1+0+0+1+1 = 3$

$\triangle_{1,3} = 7 - 3 = 4$

$x_i = 2$

$x_j = 3$

$Q(S \cup \{x_j\} \setminus \{x_i\}) = Q(\{1,3\} = 0+1+0+1+1 = 3$

$\triangle_{2,3} = 7 - 3 = 4$

$\triangle_{i03} = \triangle_{13} = \triangle_{23} = 4$

marquer "3" *avec* " + "

_ $2^{ème}$ *itération*

$x_i = 3$

$x_j = 4$

$Q(S \cup \{x_j\} \setminus \{x_i\}) = Q(\{4,2\} = 1+0+1+0+2 = 4$

$\triangle_{3,4} = 3 - 4 = -1$

$x_i = 2$

$x_j = 4$

$Q(S \cup \{x_j\} \setminus \{x_i\}) = Q(\{3,4\} = 3+2+0+0+1 = 6$

$\triangle_{2,4} = 3 - 6 = -3$

$\triangle_{i04} = -1$

marquer "4" *avec* " + "

$3^{ème}$ *itération*

$x_i = 3$

$x_j = 5$

$Q(S \cup \{x_j\} \setminus \{x_i\}) = Q(\{5,2\} = 1+0+1+2+0 = 4$

$\triangle_{3,5} = 3 - 4 = -1$

$x_i = 2$

$x_j = 5$

$Q(S \cup \{x_j\} \setminus \{x_i\}) = Q(\{3,5\} = 2 + 3 + 0 + 1 + 0 = 6$

$\triangle_{2,5} = 3 - 6 = -3$

$\triangle_{i05} = -1$

marquer "5" *avec* " + "

On redémarre avec $S0 = \{3,2\}$ *et le coût* $= 3$

$V - S = \{1,4,5\}$

1ère *itération*

$x_i = 3$

$x_j = 1$

$Q(S \cup \{x_j\} \setminus \{x_i\}) = Q(\{1,2\} = 0 + 0 + 3 + 2 + 2 = 7$

$\triangle_{3,1} = 3 - 7 = -4$

$x_i = 2$

$x_j = 1$

$Q(S \cup \{x_j\} \setminus \{x_i\}) = Q(\{3,1\} = 0 + 1 + 0 + 1 + 1 = 3$

$\triangle_{2,1} = 3 - 3 = 0$

$\triangle_{i01} = 0$

marquer "1" *avec* " + "

2ème *itération*

$x_i = 3$

$x_j = 4$

$Q(S \cup \{x_j\} \setminus \{x_i\}) = Q(\{4,2\} = 1 + 0 + 1 + 0 + 2 = 4$

$\triangle_{3,4} = 3 - 4 = -1$

$x_i = 2$

$x_j = 4$

$Q(S \cup \{x_j\} \setminus \{x_i\}) = Q(\{3,4\} = 3 + 2 + 0 + 0 + 1 = 6$

$\triangle_{2,4} = 3 - 6 = -3$

$\triangle_{i04} = -1$

marquer "4" *avec* " + "

3ème *itération*

$x_i = 3$

$x_j = 5$

$Q(S \cup \{x_j\} \setminus \{x_i\}) = Q(\{5,2\} = 1 + 0 + 1 + 2 + 0 = 4$

$\triangle_{3,5} = 3 - 4 = -1$

$x_i = 2$

$x_j = 5$

$Q(S \cup \{x_j\} \setminus \{x_i\}) = Q(\{3,5\} = 2 + 3 + 0 + 1 + 0 = 6$

$\triangle_{2,5} = 3 - 6 = -3$

$\triangle_{i05} = -1$

marquer "4" *avec* " + "

Donc on s'arrête avec la solution :

$S0 = \{3,2\}$ *et le coût* $= 3$.

Chapitre 5

Résultats numériques

Les méthodes, utilisées dans ce mémoire (méthode du sous gradient avec différentes variantes et la méthode de sous gradient de remplacement), ont toutes été programmées en langage Matlab sur un PC **acer** avec les références suivantes :
Processeur : Intel(R) core(TM) $i3-3110MCPU@2.40GHz$.
Mémoire installée 4.00 G0 RAM.

Nous avons testé les différentes stratégies de choix du coefficient de relaxation sur des exemples tirés de la base de donnée OR-Library introduite par Beaseley *[3]*. Nous avons utilisé les 40 exemples (pmed01, pmd02, ...,pmd40). Pour déterminer la matrice des plus courtes distances dans le graphe, nous avons utilisé l'algorithme de Dijkstra (DA).
On note par :

- Z_{opt} : La valeur optimale de l'objectif.
- z_{ub} : Borne supérieure du problème dual.
- z_{lb} : Borne inférieure du problème dual.
- ti : Temps exprimé en secondes.
- n : Taille du problème (ordre du graphe).
- $iter$: Nombre d'itérations.
- $kmax1$: Nombre d'itérations durant lequel on observe, dans les règles R2 et R3, si une amélioration de la valeur de la fonction objectif a eu lieu.

Dans ce chapitre, nous commençons d'abord par exécuter la méthode du sous gradient, sur les 40 exemples de la base de données OR-Library *[3]*, en utilisant les règles R1 et R2. Comme le montrent les tableaux (Tab1.1 et Tab1.2), nous obtenons de meilleurs résultats avec la règle R1 : meilleur temps CPU et meilleur qualité des solutions. Nous avons fait plusieurs choix du coefficient α (coefficient, compris entre 0 et 1, qui doit être multiplié par le coefficient de relaxation). On rappelle que $\rho_{j+1} = \alpha\rho_j$ $(\alpha < 1)$. Dans le tableau Tab1.1, nous avons choisi $kmax1 = 5$ et deux valeurs du coefficient α qui sont 0.2 et 0.7. $\alpha = 0.2$ donne, dans la majorité des cas, de meilleurs résultats.

Dans le tableau Tab1.2, nous avons choisi $kmax1 = 10$ et deux valeurs du coefficient α qui sont 0.2 et 0.7.Comme précédemment, $\alpha = 0.2$ donne, dans la majorité des cas, de meilleurs résultats.

En conclusion, nous constatons que la meilleure valeur de $kamx1$ est égale à 5 et la meilleure valeur du coefficient α est 0.2.

Pour comparer les règles R1 et R3, nous avons trouvé que pour $kamx1 = 10$ et $\alpha = 0.2$,

la règle R3 donne de meilleurs temps CPU pour quasiment la même qualité des solutions que la règle R1. Le tableau Tab2 illustre parfaitement notre propos.

On conclut ce chapitre par le tableau Tab3 qui résume les résultats des tests effectuées pour comparer la méthode combinée du sous gradient classique et du sous gradient de remplacement. Cette dernière s'avère être plus efficace que la méthode du sous gradient.

			R1			**R2 Avec** $\alpha = 0.2$			**R2 avec** $\alpha = 0.7$		
problème	n	z_{opt}	$iter$	Z_{lb}	ti	$iter$	z_{lb}	ti	$iter$	Z_{lb}	ti
pmed01	100	5819	139	5818.1	0.1054	365	5818	0.2146	162	5818.1	0.1041
pmed02	100	4093	440	4088.4	0.3404	315	4087.4	0.2715	1000	4087.4	1.3336
pmed03	100	4250	440	4240.8	0.3574	340	4238	0.32045	900	4240	1.4786
pmed04	100	3034	440	3034	0.4888	110	3034	0.3213	300	3034	5.7558
pmed05	100	1355	440	1355	0.6842	120	1355	0.5182	300	1355	4.2652
pmed06	200	7824	835	7783.5	1.1083	295	7782.7	1.0012	900	7783.5	5.7970
pmed07	200	5631	779	5630.9	1.5886	780	5630.7	1.9287	800	5630	3.1240
pmed08	200	4445	800	4445	2.8438	140	4445	1.5012	400	44445	6.3563
pmed09	200	2735	835	2734	4.84438	645	2732.7	3.7138	700	2732.7	13.1615
pmed10	200	1255	835	1255	7.57276	175	1255	6.3896	600	1255	4163
pmed11	300	7696	1232	7692.9	3.1048	295	7688.7	0.9061	1200	7688.7	7.9411
pmed12	300	6634	1232	6625.7	3.6901	330	6625.5	1.4757	800	6625.7	10.8774
pmed13	300	4374	63	4373	6.3510	63	4373	6.3238	63	4373	7.0478
pmed14	300	2968	1232	2971.7	26.2014	1000	2967	22.432	2000	2967.2	36.2605
pmed15	300	1729	1232	1729	33.5864	1000	1729	30.3432	1000	1729	41.4577
pmed16	400	8162	1630	8091.9	7.1876	365	8090.5	1.8296	690	8091.9	8.0233
pmed17	400	6999	1630	6968.7	8.0747	960	6968	2.7584	1100	6968	16.8413
pmed18	400	4809	1630	4808.5	39.5448	1700	4808.4	69.8393	2000	4808.4	78.9024
pmed19	400	2845	1630	2845	80.1294	1000	2845	69.8393	3000	2845	139.44
pmed20	400	1789	1630	1789	134.1919	415	1788.6	178.86	2100	1789	162.348
pmed21	500	9138	39	9137.28	0.9940	39	9137.28	0.9967	39	9137.28	1.0499
pmed22	500	8579	2029	8543.5	17.6072	190	8538.2	4.4772	2135	8543.4	20.6379
pmed23	500	4619	2030	4619	74.8379	260	4619	60.9951	2200	4619	85.0847
pmed24	500	2961	2029	2961	143.7927	3000	2960.8	154.5397	3040	2961	171.4673
pmed25	500	1828	2029	1827.8	243.937	2200	1827.5	244.9015	2200	1827.7	245.8473
pmed26	600	9917	2427	9853.6	28.3363	2750	9847.9	23.9823	2100	9853.1	30.657
pmed27	600	8307	2427	8301.6	32.1909	2500	8299	33.4808	2510	8300.4	34.8289
pmed28	600	4498	2427	4498	158.4120	2500	4497.5	159.6922	2496	4498	178.4510
pmed29	600	3033	2427	3033	398.6419	2764	3033	399.3081	2700	3033	482.9274
pmed30	600	1989	2300	1989	663.1566	2500	1989	664.7548	3000	1989	805.462
pmed31	700	10086	2826	10026	44.4468	2850	10020	44.8614	2950	10026	44.0995
pmed32	700	9297	2826	9292.23	49.9655	2700	9282.9	47.9062	3000	9290.1	57.8010
pmed33	700	4700	2826	4700	252.6668	2900	4699	260.8995	3330	4699.6	330.9129
pmed34	700	3013	2826	3012.9	1069.807	2000	3012.9	1000.1259	2900	3012.8	1024.3
pmed35	800	10400	3225	10302	63.3161	210	10294.4	67.247	3250	10301	76.5305
pmed36	800	9934	3225	9833	70.2596	900	9828.1	51.4456	3310	9826.8	52.3780
pmed37	800	5057	3100	5057	617.9622	1000	5056	581.5794	3200	5057	643.8342
pmed38	900	11060	3627	10947	94.10478	200	10938	60.2677	3700	10947	96.7532
pmed39	900	9423	3627	9360.9	94.5193	2300	9360.1	88.7445	3700	9360	95.0665
pmed40	900	5128	3500	5127.9	1255.922	2900	5127.9	1188.8	3500	5127.9	1492.2

Tab1.1 : comparaison entre les règles R1 et R2 avec $kmax1 = 5$

			R1			R2 Avec α = 0.2			R2 avec α = 0.7		
problème	n	z_{opt}	$iter$	Z_{lb}	ti	$iter$	z_{lb}	ti	$iter$	Z_{lb}	ti
pmed01	100	5819	139	5818.1	0.1054	365	5818	0.2233	139	5818	0.1041
pmed02	100	4093	440	4088.4	0.3404	315	4088.4	0.2562	2270	4088.4	1.3336
pmed03	100	4250	440	4240.8	0.3574	340	4239	0.3208	2830	4240.03	1.4786
pmed04	100	3034	440	3034	0.4888	110	3033	0.3300	3000	3033	5.7558
pmed05	100	1355	440	1355	0.6842	460	1355	0.52788	2900	1353.5	4.2652
pmed06	200	7824	835	7783.5	1.1083	900	7783.5	1.4856	3100	7783	5.7970
pmed07	200	5631	779	5630.9	1.5886	800	5630.7	1.9171	1665	5630.7	3.1240
pmed08	200	4445	800	4445	2.8438	140	4444.3	1.5066	3050	4444.3	6.3563
pmed09	200	2735	835	2734	4.84438	145	2732.7	3.7198	2865	2724.7	13.1615
pmed10	200	1255	835	1255	7.57276	175	1255	6.3912	3114	1254.5	11.4163
pmed11	300	7696	1232	7692.9	3.1048	295	7688.71	0.9081	2810	7692	7.9411
pmed12	300	6634	1232	6625.7	3.6901	330	6625.4	1.5154	3690	6625.7	10.8774
pmed13	300	4374	63	4373	6.3510	63	4373	6.3248	63	4373	7.0478
pmed14	300	2968	1232	2971.7	26.2014	1300	2971.6	28.7336	2900	2967.2	36.2605
pmed15	300	1729	1232	1729	33.5864	200	1729	30.4360	2610	1729	41.4577
pmed16	400	8162	1630	8091.9	7.1876	365	8090.5	1.8297	1690	8091.9	8.0233
pmed17	400	6999	1630	6968.7	8.0747	1700	6968.6	8.7221	2950	6968.6	16.8413
pmed18	400	4809	1630	4808.5	39.5448	1900	4808.36	40.2141	3256	4807.7	78.9024
pmed19	400	2845	1630	2845	80.1294	1200	2845	69.6267	3400	2843.3	139.44
pmed20	400	1789	1630	1789	134.1919	415	1788	126.562	2260	1789	162.348
pmed21	500	9138	39	9137.28	0.9940	39	9137.28	0.9149	39	9137.28	1.0499
pmed22	500	8579	2029	8543.5	17.6072	190	8538.22	4.4772	2470	8543.5	20.6379
pmed23	500	4619	2030	4619	74.8379	2600	4619	80.9951	3200	4617.7	85.0847
pmed24	500	2961	2029	2961	143.7927	325	2960.1	127.525	3840	2961	171.4673
pmed25	500	1828	2029	1827.8	243.937	2200	1827.5	246.085	1770	1827	242.8473
pmed26	600	9917	2427	9853.6	28.3363	275	9848.9	4.0518	2500	9853.5	30.657
pmed27	600	8307	2427	8301.6	32.1909	360	8299	8.5508	1290	8300.5	18.8289
pmed28	600	4498	2427	4498	158.4120	350	4497.5	140.496	2800	4498	178.4510
pmed29	600	3033	2427	3033	398.6419	2530	3033	399.0081	2440	3033	482.9274
pmed30	600	1989	2300	1989	663.1566	2500	1989	627.076	4000	1985.7	805.462
pmed31	700	10086	2826	10026	44.4468	225	10019.8	4.7704	2950	10026	44.0995
pmed32	700	9297	2826	9292.23	49.9655	2900	9291.19	53.8062	3070	9291.6	57.8010
pmed33	700	4700	2826	4700	252.6668	2900	4699	258.8451	3530	4700	330.9129
pmed34	700	3013	2826	3012.9	1069.807	1600	3012.9	970.2700	2140	3012.8	1024.3
pmed35	800	10400	3225	10302	63.3161	2100	10294.14	67.8640	2180	10302	76.5305
pmed36	800	9934	3225	9833	70.2596	2850	9828.06	71.4456	2310	9826.8	52.3780
pmed37	800	5057	3100	5057	617.9622	3000	5056.09	622.2348	3200	5057	643.8342
pmed38	900	11060	3627	10947	94.10478	2400	10937	80.1678	3900	10947	96.7532
pmed39	900	9423	3627	9360.9	94.5193	3890	9360.1	100.2927	3700	9360.1	95.0665
pmed40	900	5128	3500	5127.9	1255.922	3520	5127.9	1461.8	3520	5127.9	1492.2

Tab1.2 : comparaison entre les règles R1 et R2 avec $kmax1 = 10$

			R1			R3 Avec α = 0.2			R3 avec α = 0.5		
problème	n	z_{opt}	$iter$	Z_{lb}	ti	$iter$	z_{lb}	ti	$iter$	Z_{lb}	ti
pmed01	100	5819	139	5818.1	0.1054	365	5818	0.2233	162	5818	0.1102
pmed02	100	4093	440	4088.4	0.3404	315	4088.4	0.2562	430	4088.4	0.3025
pmed03	100	4250	440	4240.08	0.3574	340	4239	0.3208	585	4240.3	0.4060
pmed04	100	3034	440	3034	0.4888	110	3033.99	0.3300	185	3033.99	0.3542
pmed05	100	1355	440	1355	0.6842	120	1355	0.52788	190	1355	0.5395
pmed06	200	7824	835	7783.5	1.1083	295	7783.5	0.4856	565	7783.47	0.7840
pmed07	200	5631	779	5630.9	1.5886	265	5630.7	0.9171	455	5630.8	1.5886
pmed08	200	4445	800	4445	2.8438	140	4444.9	1.5066	215	4444.9	1.6126
pmed09	200	2735	835	2734	4.84438	145	2732.7	3.7198	340	2733.14	4.0394
pmed10	200	1255	835	1255	7.57276	175	1255	6.3912	225	1255	6.5016
pmed11	300	7696	1232	7692.9	3.1048	295	7688.71	0.9081	435	7692.7	2.8497
pmed12	300	6634	1232	6625.7	3.6901	330	6625.4	1.5154	630	6625.7	2.1927
pmed13	300	4374	63	4373	6.3510	63	4373	6.3248	63	4373	6.3669
pmed14	300	2968	1232	2971.7	26.2014	180	2971.6	22.7336	460	2971.7	23.9047
pmed15	300	1729	1232	1729	33.5864	200	1729	30.4360	680	1729	32.4241
pmed16	400	8162	1630	8091.9	7.1876	365	8090.9	1.8297	525	8091.9	2.5226
pmed17	400	6999	1630	6968.7	8.0747	360	6968.6	2.7221	410	6968.6	2.9908
pmed18	400	4809	1630	4808.5	39.5448	275	4808.36	32.2141	305	4808.4	33.7082
pmed19	400	2845	1630	2845	80.1294	120	2845	69.6267	375	2845	73.5991
pmed20	400	1789	1630	1789	134.1919	415	1788.6	126.562	455	1789	126.6261
pmed21	500	9138	39	9137.28	0.9940	39	9137.28	0.9149	39	9137.28	0.95850
pmed22	500	8579	2029	8543.5	17.6072	190	8538.22	4.4772	350	8543.5	5.88591
pmed23	500	4619	2030	4619	74.8379	260	4619	59.9951	285	4618.99	90.394
pmed24	500	2961	2029	2961	143.7927	325	2960.7	127.525	410	2961	128.2728
pmed25	500	1828	2029	1827.8	243.937	220	1827.5	226.085	350	1827.8	226.0681
pmed26	600	9917	2427	9853.6	28.3363	275	9848.9	4.0518	440	9853.5	6.04611
pmed27	600	8307	2427	8301.6	32.1909	360	8299	8.5508	685	8301.6	12.289
pmed28	600	4498	2427	4498	158.4120	350	4497.5	140.496	635	4498	139.005
pmed29	600	3033	2427	3033	398.6419	195	3033	367.0081	390	3033	369.890
pmed30	600	1989	2300	1989	663.1566	250	1989	627.076	480	1989	637.3668
pmed31	700	10086	2826	10026	44.4468	225	10019.8	4.7704	450	10026	8.1810
pmed32	700	9297	2826	9292.23	49.9655	415	9292.19	12.8062	585	9292.23	15.3829
pmed33	700	4700	2826	4700	252.6668	260	4699	198.8451	565	4699.7	204.1775
pmed34	700	3013	2826	3012.9	1069.807	160	3012.2	970.2700	675	3012.8	1053.2839
pmed35	800	10400	3225	10302	63.3161	210	10294.14	6.8640	520	10301.5	13.2103
pmed36	800	9934	3225	9833	70.2596	285	9829.06	11.4456	355	9832.1	12.6851
pmed37	800	5057	3100	5057	617.9622	415	5056.09	547.2348	510	5057	547.2348
pmed38	900	11060	3627	10947	94.10478	240	10937	10.1678	385	10947	27.5435
pmed39	900	9423	3627	9360.9	94.5193	470	9360.6	20.2927	465	9360.6	18.4927
pmed40	900	5128	3500	5127.9	1255.922	285	5127.9	1069.596	430	5127.9	1163.75

Tab2 : comparaison entre les règles R1 et R3 avec $kmax1 = 10$

			SG			SGR		
problème	n	z_{opt}	$iter$	Z_{lb}	ti	$iter$	Z_{lb}	ti
pmed01	100	5819	139	5818.1	0.1372	23	5801.3	0.0401
pmed02	100	4093	440	4088.4	0.3404	12	4077.8	0.1214
pmed03	100	4250	440	4240	0.3741	08	4247.8	0.1541
pmed04	100	3034	440	3034	0.4885	06	2938.9	0.2521
pmed05	100	1355	440	1355	0.6841	06	1289	0.4442
pmed06	200	7824	835	7783.5	1.1534	17	7818.8	0.0941
pmed07	200	5631	779	5630.9	1.5831	13	5597.5	0.5748
pmed08	200	4445	800	4445	2.4479	9	4391	1.3017
pmed09	200	2735	835	2734	8.8444	8	2672.9	3.4791
pmed10	200	1255	835	1255	7.5836	5	1206.4	6.0528
pmed11	300	7696	1232	7692.9	3.1122	24	7688	0.2031
pmed12	300	6634	1232	6625.7	3.6901	20	6615.8	0.6642
pmed13	300	4374	63	4373	6.3183	10	4321.6	6.1471
pmed14	300	2968	1232	2967.1	26.2014	9	2896.5	22.2382
pmed15	300	1729	1200	1729	33.9817	6	1658.6	25.5319
pmed16	400	8162	1610	8119	6.6702	27	8139	0.3674
pmed17	400	6999	1630	6968.7	7.8114	13	6406.0	1.2618
pmed18	400	4809	1600	4808.4	38.3226	12	4764.6	25.0526
pmed19	400	2845	1630	2845	77.6084	9	2765.1	68.6884
pmed20	400	1789	1600	1789	133.6862	7	1776.3	123.5555
pmed21	500	9138	100	9137.3	0.9000	21	9068.5	0.6192
pmed22	500	8579	1938	8543.6	19.3836	31	8563.3	4.1761
pmed23	500	4619	2000	4619	74.2159	14	4550.9	58.1650
pmed24	500	2961	2000	2961	142.2737	10	2958.3	123.6628
pmed25	500	1828	2000	1827.8	242.6240	07	1780.8	221.0302
pmed26	600	9917	2427	9853.6	27.6062	23	9916.2	0.9214
pmed27	600	8307	2400	8301.6	30.8652	21	8300.5	4.4158
pmed28	600	4498	2427	4498	158.4120	12	44630	129.3004
pmed29	600	3033	2300	3033	404.0906	9	2952.2	381.3697
pmed30	600	1989	2300	1989	663.1566	7	1984.5	631.7933
pmed31	700	10086	2500	10026	39.7590	21	10059	1.2674
pmed32	700	9297	2500	9297	45.4305	19	9270.4	6.3258
pmed33	700	4700	2700	4700	252.7150	14	4693.9	190.3413
pmed34	700	3013	2700	3012.9	20.6785	8	2949.7	752.3367
pmed35	800	10400	3100	10302	64.6784	23	10372	2.5819
pmed36	800	9934	3200	9833	69.8520	18	9819.8	5.4762
pmed37	800	5057	3100	5057	617.9622	11	4952.2	518.2730
pmed38	900	11060	3500	10947	91.6504	22	10998	4.2450
pmed39	900	9423	3500	9423	94.5193	20	9388.4	6.8658
pmed40	900	5128	3500	5153	1255.9	12	5122	911.697

Tab3 : Etude comparative entre les méthodes SG et SGR

Conclusion

Nous avons étudié la méthode du sous gradient dans le cas du problème du p-médian dans un graphe.
Ce problème est un cas particulier de l'optimisation combinatoire. Bien que des résultats théoriques de convergence existent mais ils ont un intérêt purement théorique. En pratique, la détermination optimale des différents paramètres (coefficients de relaxation, nombre d'itérations $kmax1$ dans notre cas, ... etc.) se fait de manière expérimentale.
Nous avons proposé la règle R3, que nous avons testé sur les exemples tirées de OR-Library avec succes.
Pour confirmer les performances de cette règle, nous pensons et nous devons faire des tests sur des instances beaucoup plus importantes. Nous pensons, dans de futures travaux, appliquer cette règle au cas de la méthode du sous gradient de remplacement.

Bibliographie

[1] **Ablaoui Hocine and Amir Abdessamad.** : Surrogat subgradient for p-median problems "Avril 2018 19, 20, 22, 23

[2] **Ayad Fatima and Belouihat Fatiha. :** *Résolution du problème p-médian dans un graphe* : mémoire de fin d'étude, master université de mostaganem, 2008. 17, 24, 26

[3] **Beasley, J.E. OR-Library** : Distributing test problems by electronic mail, Journal of Operational Research Society, vol. 41, no. 11 (1990) 1069-1072. 31

[4] **Benbernous Saadia :** .Résolution du problème p-médian dans un arbre , mémoire de fin d'étude, master université de mostaganem, 2017. 3, 18

[5] **Christofides N and beasley, J. E** : " A tree search algorithm for the p-median problems" European Journal of Operational Research vol 10 (1982) 196-204. 17, 18

[6] **Didier M.** : *Introduction à la théorie des graphe.* 2, 3, 4

[7] https ://www.ljll.math.upmc.fr/privat/cours/ensem.php. 6

[8] **Held M, Wolfe P end Crowder H.P (1974)** :, Validation de la notion dualité en programmation discrète : sélection et affectation optimales d'une flotte d'avions, R.A.I.R.O.

[9] **M.Bergonniaux** : *optimisation sans contraintes.* 5

[10] **Michel Gondram, and Michel Minoux** : *graphe et algorithme.* 10, 11, 12, 13, 15

[11] **Minoux M .(1977)** : Résolution des problèmes de voyageur de commerce orientés de grandes dimensions : méthodes exactes et approchée. Non publie. 13

[12] **M. Bergounioux** : *Optimisation sans contraintes.*

[13] **POLYAK B.T(1967)** : A general method for solving extremal problems, Sovient Math. Doklady, 12

[14] **Polyak B.T (1969)** : Minimization of unsmooth functionals, USSR Computational mathematics and Mathematical physics. 12, 13

[15] **Wang,J, Luh, P.B,ZHAO, X, and Wang, J (1997)** : An Optimization-Based Algorithm for job shop scheduling, sadhana. 21

[16] **X. ZHAO, P. B. LUH, and J. WANG5** : *Communicated by W. B. Gong and D. D. Yao,*Surrogate Gradient Algorithm for Lagrangian Relaxation. 14, 15, 21

[17] **M.Bergonniaux** : *optimisation sans contraintes.* 5

Résumé

Dans ce mémoire, on étudie la méthode du sous gradient dans le cas particulier du problème du p-médian dans un graphe où une nouvelle règle de choix des coefficients de relaxation (règle R3) est proposée : les tests numériques effectués sur la base de donnée OR-library , ont été très concluants pour cette règle.

Abstract

In this thesis, we study the sub-gradient method in the particular case of the p-median problem in a graph where a new choice rule of coefficients of relaxation (rule R3) is proposed : The numeral tests made and based on OR- library data, were very decisive for this rule.