UNIVERSITÉ, Dr. YAHIA FARÈS DE MÉDÉA

Faculté des Sciences

Département de Mathématiques et de l'Informatique

MÉMOIRE DE MASTER

en mathématiques

Spécialité : Recherche Opérationnelle

THÈME

IMPLÉMENTATION D'UN ALGORITHME DE TYPE POINT INTÉRIEUR POUR LA RÉSOLUTION DU PROBLÈME DE FLOT DE COÛT MINIMUM

présenté

par

GHERIBI Dalila et DJOUABRI Zina

soutenu devant le Jury composé de :

N. IKHLEF ESCHOUF	M.C.A	U.YF.M	Président
S. KOUIDER AIAD	M.A.A	U.YF.M	Examinatrice
A. BENDALI BRAHAM	M.A.B	U.YF.M	Examinatrice
A. BENKACI	M.A.A	U.YF.M	Encadreur

08 – juillet – 2019

Résumé

Dans ce mémoire, nous nous intéressons au problème de flot de coût minimum. Ce problème est un modèle très général, plusieurs problèmes peuvent être mis sous cette forme par exemple, on cite : le problème de la recherche d'un plus court chemin, et le problème de transport généralisé.

Le problème de flot de coût minimum est un problème linéaire en variables bornées. Pour sa résolution, une méthode efficace de type point intérieur « affine-scaling » est appliquée.

Nous élaborons, un code de calcul en langage Matlab, permettant de résoudre un exemple du problème de transport généralisé en temps réel.

Nous reportons aussi des résultats numériques qui sont obtenus en testant notre programme sur un exemple type. Ces résultats confirment l'efficacité de notre approche.

Mots−clés : flot de coût minimum, programme linéaire en variables bornées, problème de transport généralisé, algorithme de point intérieur.

Remerciements

Tout d'abord, nous tenons à remercier, ALLAH le Tout Puissant, pour nous avoir données le courage et la force pour accomplir ce modeste travail.

Nous tenons à exprimer toute notre gratitude à notre enseignant Monsieur BENKACI Azzedine, Maître-assistant à la faculté des sciences de l'Université de Médéa, pour nous avoir encadrées et guidées tout au long de ce mémoire. Il nous a permis de réaliser ce travail en nous manifestant sa confiance et en nous encourageant dans notre travail. Il a été toujours à notre disposition.

Nous souhaitons également exprimer toute notre reconnaissance aux enseignants du département MI, pour leur immense effort consacré à la formation du Master mathématique au sein de la faculté des sciences. Nous remercions aussi tous ceux qui ont contribué à la rédaction de ce manuscrit.

Nous ne pourrons terminer sans mentionner tout ce que nous devons à nos parents et à nos frères et sœurs. Ils nous ont toujours soutenus et supportés. Sans eux ce travail n'aurait pu voir le jour.

Enfin, nous remercions toute personne qui nous a aidé par sont soutient et sa confiance afin d'accomplir ce modeste travail.

Table des matières

Résumé i

Remerciements ii

Table des matières iii

Liste des figures v

Liste des tableaux vi

Liste des algorithmes vii

Introduction 8

1 Notions préliminaires de Graphes 11

1.1 Introduction 11
1.2 Graphes orientés 11
1.2.1 Extrémité initiale et extrémité terminale d'un arc 11
1.2.2 Graphe partiel et sous-graphe 12
1.2.3 Sommet adjacent et arc adjacent 12
1.2.4 Successeurs et prédécesseurs - degré d'un sommet 13
1.2.5 Matrice d'incidence aux arcs (sommet – arc) 13
1.3 Graphe valué et réseau 14
1.4 Notions de flot dans un réseau 15
1.4.1 Définitions de base 15
1.4.2 Exemple de flot 16
1.4.3 Valeur d'un flot et flot d'un sommet. 17
1.5 Conclusion 18

2 Méthode de point intérieur pour la programmation linéaire 19

2.1 Introduction 19
2.2 Définitions et notions préliminaires 19
2.3 Le problème de la programmation linéaire 20
2.4 Méthode de point intérieur «l'Affine Scaling » 21

2.4.1 Principe de la méthode 21
2.4.2 Algorithme « Affine Scaling » 23
2.4.3 Performances pratiques de l'algorithme 24
2.5 Exemple numérique 24
2.6 Conclusion 26

3 Problème de flot de coût minimum et applications 27

3.1 Introduction 27
3.2 Position du problème 27
3.3 Modèle mathématique 28
3.4 Applications au problème de transport généralisé 28
3.4.1 Position du problème et modélisation mathématique 28
3.4.2 Problème équilibré 29
3.4.3 Structure algébrique du modèle 29
3.4.4 Propriétés de la matrice A 30
3.5 Représentation du problème de transport sous forme d'un problème de flot de coût minimum 31
3.6 Processus de résolution 33
3.7 Conclusion 33

4 Implémentation et tests numériques 35

4.1 Introduction 35
4.2 Adaptation de l'algorithme pour la résolution des programmes linéaire en variables bornées 35
4.2.1 Réglage des longueurs des demis axes de l'ellipsoïde 35
4.2.2 Détermination du point intérieur de départ 36
4.2.3 Algorithme de « l'Affine Scaling » modifié 38
4.3 Exemple test de transport généralisé 38
4.4 Résultats numériques 39
4.5 Comparaison concise avec l'algorithme du simplexe 42
4.6 Conclusion 42

Conclusion 43

A. Codes Matab et Excel 45

A.1 Implémentation et mise en œuvre avec Matlab 46
A.2 Implémentation et mise en œuvre avec Excel 47

Notations et abréviations 48

Références 49

Liste des figures

Figure 1.1 – Exemple de graphe simple. 12

Figure 1.2 – Exemple de réseau simple. 15

Figure 1.3 – Exemple de flot sur le graphe $G = (X, U)$. 16

Figure 2.1 – Comportement de la méthode de point intérieur sur l'exemple pour différentes valeurs de beta (la tolérance ϵ est fixée à10^{-8}). 25

Figure 3.1 – Représentation du problème de transport généralisé sous forme d'un problème de flot de coût minimum. 32

Figure 3.2 – Schéma conceptuel de résolution d'un problème réel. 33

Figure 4.1 – La forme de l'ellipse $E(y, \beta r)$ en fonction de la position du centre y dans le pavé $\left[l_1, u_1\right] \times \left[l_2, u_2\right]$. 37

Figure 4.2 – Représentation de l'exemple test sous forme d'un réseau. 40

Figure 4.3 – Représentation de la solution optimale de l'exemple. 40

Figure A.1 – Représentation et résolution de l'exemple test avec MATLAB. 46

Figure A.2 – Représentation et résolution de l'exemple test avec EXCEL. 47

Liste des tableaux

Tableau 2.1 – Les 6 premières itérations issues de l'algorithme. 24
Tableau 4.1 – Coûts unitaires en millier de dollar $. 39
Tableau 4.2 – Quantités disponibles (en tonnes). 39
Tableau 4.3 – Quantités requises (en tonnes). 39
Tableau 4.4 – Solution initiale (point intérieur) et coût total de transport. 41
Tableau 4.5 – Solution et coût total de transport après une itération. 41
Tableau 4.6 – Solution et coût total de transport après 2 itérations. 41
Tableau 4.7 – Solution optimale et coût total de transport après 9 itérations. 41

Liste des algorithmes

Algorithme 2.1 – Méthode de point intérieur « Affine Scaling » 23

Algorithme 4.1 – Méthode modifiée de point intérieur « Affine Scaling » 38

Introduction

La Recherche Opérationnelle (en abrégé R.O) a pour but d'apporter une démarche scientifique à la résolution de problèmes de décision complexes issus du monde réel. Sa tendance scientifique est de construire des modèles mathématiques d'aide à la décision, en particulier les modèles liés à des problèmes d'optimisation et de proposer des méthodes efficaces pour les résoudre. La RO intervient de manière cruciale dans des secteurs nombreux et diversifiés comme les transports, la production industrielle, la planification et la logistique.

Dans ce mémoire, nous nous intéressons au problème de flot de coût minimum. Ce problème est issu du domaine de l'optimisation dans les réseaux, de la logistique et de la gestion des ressources.

Le problème de flot de coût minimum se formule comme un programme linéaire en variables continues bornées. Pour le résoudre, nous appliquons une méthode de type point intérieur, plus précisément « Affine-scaling »

Le principe général d'une méthode de point intérieur consiste à se déplacer le long de la direction admissible qui donne la meilleure amélioration de l'objectif après projection sur un sous-espace linéaire, tout en restant à l'intérieur du domaine réalisable. Il s'agit d'un problème de la programmation non linéaire ([3],[4],[7]).

Le principe de la méthode du simplexe consiste à se déplacer d'un point extrême en un autre le long de la frontière du polyèdre. Il y a un nombre fini de points extrêmes, il s'agit d'un problème combinatoire. Dans le mauvais cas, l'algorithme du simplexe n'est pas un algorithme polynomial pour la programmation linéaire ([4], [5], [12]).

En 1979, Khachiyan a développé le premier algorithme polynomial pour la programmation linéaire, à savoir la méthode des ellipsoïdes ([21],[22],[24]). Bien que son impact théorique important, ses performances pratiques s'avèrent mauvaises.

En 1984, Karmarkar propose les méthodes de points intérieurs polynomiales et efficaces en pratique sur des problèmes creux de grande taille ([2],[11],[23]).

Dans ce travail, notre objectif est double :

- Appliquer la méthode de type point intérieur baptisé « Affine Scaling » pour la résolution du problème de flot de coût minimum.
- Implémenter un programme informatique permettant la résolution effective d'un exemple du problème de transport généralisé.

Ce mémoire est composé de quatre chapitres, d'une conclusion générale et d'une annexe.

Le premier chapitre est consacré à la présentation de quelques définitions sur les graphes simples orientés. Les principales définitions et quelques propriétés intéressantes sont rappelées. Nous exposons aussi la notion de flot défini sur un réseau. Ces notions sont indispensables par la suite, pour étudier le problème de flot de coût minimum.

Dans le deuxième chapitre, nous présentons d'abord, quelques définitions géométriques telles que : polyèdre, point extrême et point intérieur, puis nous rappelons la position du problème de la Programmation Linéaire. Nous exposons brièvement, le principe de la méthode de point intérieur, plus précisément, le « Affine Scaling » ainsi que leur développement mathématique pour la résolution d'un programme linéaire. Au final, nous donnons un algorithme traduisant cette méthode, et nous le testons sur un exemple simple.

Dans le chapitre trois, nous posons le problème de flot de coût minimum. En premier lieu, nous exposons la formulation mathématique du problème. Il s'agit d'un programme linéaire en variables bornées. Ensuite, nous présentons le

problème de transport généralisé ainsi que son développement mathématique, et sa représentation graphique.

Le chapitre quatre est consacré à l'implémentation de notre approche de résolution. Nous reportons aussi des résultats numériques qui ont été obtenus en testant notre programme Matlab sur un exemple réel du problème de transport généralisé.

Dans l'annexe-A, nous présentons les scripts de nos programmes informatiques, codés et adaptés en langage Matlab, ainsi que leur mise en œuvre pour résoudre un exemple donné du problème de transport généralisé.

Nous terminons notre travail par une conclusion générale.

Chapitre 1

Notions préliminaires de Graphes

1.1 Introduction

Dans ce chapitre, nous présentons quelques définitions sur les graphes simples orientés. Les principales définitions et quelques propriétés intéressantes sont rappelées. Nous exposons aussi la notion de flot défini sur un réseau ([19],[20]). Ces notions sont indispensables par la suite, pour étudier le problème de flot de coût minimum.

1.2 Graphes orientés

Avant d'exposé la notion de flot, tout d'abord nous avons besoin d'introduire quelques notations et terminologie.

Définition 1.1. Un graphe simple $G = (X, U)$ est défini par la donnée :

- d'un ensemble $X := \{x_1, x_2, \dots, x_n\}$ dont les éléments sont appelés « sommets », si $|X| = n$ alors le graphe G est dit d'ordre n.
- d'un ensemble $U := \{u_1, u_2, \dots, u_m\}$ dont les éléments sont des couples ordonnés de sommets (x, y) appelés « arcs ».

Graphiquement, les sommets sont représentes par des points, et l'arc $u = (x, y)$ sera représenté par une flèche allant du point x vers le point y.

1.2.1 Extrémité initiale et extrémité terminale d'un arc

Soit $G = (X, U)$ un graphe orienté. À tout arc $u = (x, y) \in U$, on associe :

- $I(u) = x$ est l'extrémité initiale de l'arc u.
- $T(u) = y$ est l'extrémité terminale de l'arc u.

Un arc $v = (x, x)$ dont les extrémités coïncident est appelé une boucle.

La figure 1.1 présente un exemple de graphe orienté.

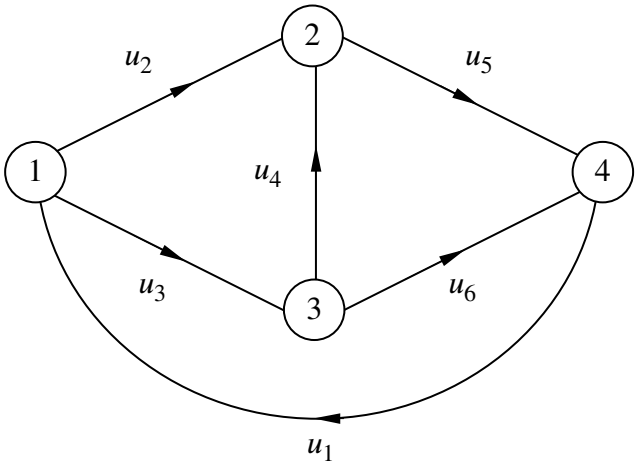


Figure 1.1 – Exemple de graphe simple.

1.2.2 Graphe partiel et sous-graphe

Beaucoup de problèmes d'optimisation consistent à chercher un graphe partiel ou un sous-graphe vérifiant certaines propriétés, dans un graphe G. On donne les deux définitions suivantes :

Définition 1.2 (*Graphe partiel*)**.** Soit $G = (X, U)$ un graphe. Le graphe $G' = (X, U')$ est un graphe partiel de G, si U' est inclus dans U.

Définition 1.3 (*Sous graphe*)**.** Un sous-graphe de G est un graphe $H = \left(X', U\left[X'\right]\right)$ tel que X' est un sous ensemble de X, et $U\left[X'\right]$ sont les arcs induits par U sur X'.

Définition 1.4 (*Sous graphe partiel*)**.** Un sous-graphe partiel de G est, à la fois un graphe partiel et un sous graphe.

1.2.3 Sommet adjacent et arc adjacent

- Deux sommets sont adjacents (ou voisins) s'ils sont joints par un arc.
- Deux arcs sont adjacents s'ils ont au moins une extrémité commune.

1.2.4 Successeurs et prédécesseurs - degré d'un sommet

Soit $G = (X, U)$ un graphe simple. À tout sommet $x \in X$, on associe.

- $\Omega^+(x) := \{u \in U : I(u) = x\}$ est l'ensemble des arcs sortant de x.
- $\Omega^-(x) := \{u \in U : T(u) = x\}$ est l'ensemble des arcs entrant de x.
- $\Omega(x) := \Omega^+(x) \cup \Omega^-(x)$ est l'ensemble des arcs incidents au sommet x.

- $\Gamma^+(x) = \{s \in X : (x, s) \in \Omega^+(x)\}$ est l'ensemble des successeurs de x.
- $\Gamma^-(x) = \{s \in X : (s, x) \in \Omega^-(x)\}$ est l'ensemble des prédécesseurs de x.
- $\Gamma(x) = \Gamma^+(x) \cup \Gamma^-(x)$ est l'ensemble des voisins du sommet x.

- $d_G^+(x)$ demi-degré extérieur de x, est le nombre d'arcs ayant x comme extrémité initiale et on écrit $d_G^+(x) := |\Omega^+(x)|$.
- $d_G^-(x)$ demi-degré intérieur de x, est le nombre d'arcs ayant x comme extrémité terminale, et on écrit $d_G^-(x) := |\Omega^-(x)|$.
- $d_G(x) := d_G^+(x) + d_G^-(x)$ degré de x, est le nombre d'arcs ayant le sommet x comme extrémité, et on écrit $d_G(x) := |\Omega(x)|$.

On remarque que : $\sum_{x \in X} d_G^+(x) = \sum_{x \in X} d_G^-(x) = |U|$.

1.2.5 Matrice d'incidence aux arcs (sommet – arc)

Soit $G = (X, U)$ un graphe orienté avec $|X| = n$ et $|U| = m$, la matrice d'incidence aux arcs est une matrice $n \times m$, telle que chaque ligne i associée à un sommet et chaque colonne j associée à un arc et qui prennent les valeurs suivantes :

$$A_{ij} := \begin{cases} 1 & \text{si } x_i \text{ est l'extrémité initiale de } u_j \\ -1 & \text{si } x_i \text{ est l'extrémité terminale de } u_j \\ 0 & \text{sinon} \end{cases}$$

Les égalités suivantes sont vérifiées :

$$\sum_{i=1}^{n} A_{ij} = 0 \qquad \forall j \in \{1, \dots, m\} \tag{1.1}$$

$$\sum_{i=1}^{n} |A_{ij}| = 2 \qquad \forall j \in \{1, \dots, m\} \tag{1.2}$$

$$\sum_{j=1}^{m} |A_{ij}| = d_G(x_i) \qquad \forall i \in \{1, \dots, n\} \tag{1.3}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m} |A_{ij}| = 2|U| \tag{1.4}$$

La matrice d'incidence de graphe de la figure 1.1 est donnée par :

$$A := \begin{pmatrix} -1 & 1 & 1 & & & \\ & -1 & & -1 & 1 & \\ & & -1 & 1 & & 1 \\ 1 & & & & -1 & -1 \end{pmatrix}$$

1.3 Graphe valué et réseau

Sans perte de généralité, on suppose dans la suite, que le graphe $G = (X, U)$ est connexe. Dans cette section, Nous exposons la définition de graphe valué et de réseau.

Définition 1.5 (*Graphe valué*)**.** Un graphe simple $G = (X, U)$ est dit valué ou pondéré, si à chaque arc u on associe une valeur numérique, notée $c(u)$ représentant en général, le coût de passage sur cet arc. Il est noté $G = (X, U, c)$, ou c est une application telle que $c: U \longrightarrow \mathbb{R}$.

Définition 1.6 (*Réseau*)**.** Un graphe valué $G = (X, U, c)$ connexe est un réseau si :

- il possède deux sommets particuliers s et p appelés respectivement, source et puits.
- les arcs sont munis de capacités inférieures $a(u)$ et supérieures $b(u)$ avec $a(u) \leq b(u)$ pour tout $u \in U$.
- l'arc (p, s) existe, il est appelé arc de retour, et se note $u_r = (p, s)$.

Il est noté $R = (X, U, a, b, c)$, ou a, b et c sont respectivement la capacité inférieure, supérieure et le coût d'utilisation de l'arc u.

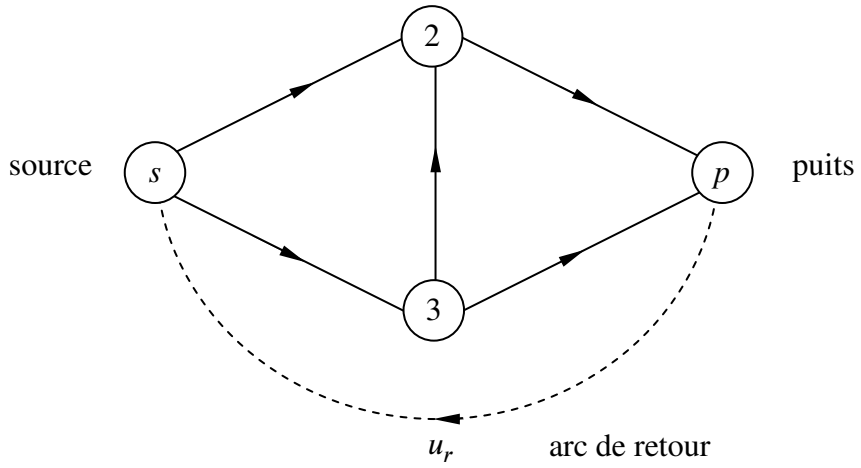


Figure 1.2 – Exemple de réseau simple.

1.4 Notions de flot dans un réseau

Dans cette section, nous rappelons la notion de flot d'un arc et de flot compatible en présentant un exemple simple. Nous présentons aussi la définition de la valeur d'un flot et flot d'un sommet.

1.4.1 Définitions de base

Dans le langage courant, un flot (ou flux) est défini comme la quantité transférée du point A à un autre B. Par la suite, on donne des définitions plus formelles sur le sujet.

Définition 1.7 (*Flot*)**.** Un flot défini sur un réseau est une application $f : U \longrightarrow \mathbb{R}$ notée $f(u)$, ou $f(x, y)$ vérifiant la condition suivante :

$$\sum_{u \in \Omega^+(x)} f(u) = \sum_{u \in \Omega^-(x)} f(u) \qquad \forall\, x \in X$$

Il s'agit de la première loi de Kirchhoff[1], ou encore la loi de conservation.

Dans la pratique, le flot est borné, alors on donne la définition suivante :

[1] La loi est énoncée par le physicien Allemand Gustav Kirchhoff en 1845.

Définition 1.8 (*Flot compatible*)**.** Un flot $f \in \mathbb{R}^{|U|}$ défini dans un réseau $R = (X, U, a, b, c)$ est dit compatible s'il vérifie la condition suivante :

$$a(u) \leq f(u) \leq b(u) \qquad \forall\, u \in U$$

où $a(u)$ et $b(u)$ sont respectivement, la capacité minimum et maximum de l'arc u.

1.4.2 Exemple de flot

Le courant dans un réseau électrique est un exemple classique de flot sur un graphe. Soit $G = (\{1, 2, 3, 4\}, \{u_1, u_2, u_3, u_4, u_5, u_6\})$ le graphe à 4 sommets et 6 arcs de la figure 1.2. Le vecteur $f = (5, 3,\ 2, -2, 1, 4)$ est un flot sur G. En effet, on peut s'assurer que la loi de conservation aux nœuds est bien vérifiée.

Dans cet exemple, l'ensemble des sommets est $X \coloneqq \{1, 2, 3, 4\}$, et l'ensemble des arcs est $U \coloneqq \{u_1, u_2, u_3, u_4, u_5, u_6\}$.

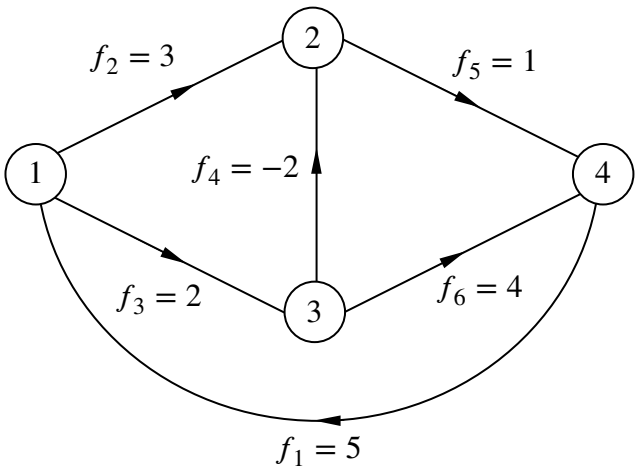


Figure 1.3 – Exemple de flot sur le graphe $G = (X, U)$.

Remarque 1.1. L'orientation choisie pour les arcs est arbitraire, mais la négativité de la composante f_4 par exemple, indique que le courant électrique va du sommet 2 au sommet 3 sur l'arc u_4.

On voit bien que le flot représenté dans la figure 1.3, vérifie la loi de conservation :

$$Af := \begin{pmatrix} -1 & 1 & 1 & & & \\ & -1 & & -1 & 1 & \\ & & -1 & 1 & & 1 \\ 1 & & & & -1 & -1 \end{pmatrix} \begin{pmatrix} 5 \\ 3 \\ 2 \\ -2 \\ 1 \\ 4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

1.4.3 Valeur d'un flot et flot d'un sommet.

Le problème classique du flot maximal consiste à transporter la quantité maximale possible d'une origine (source) à une destination (puits) données, sans dépasser les capacités des arcs. Cette quantité maximale est appelée valeur de flot.

Définition 1.9 (*Valeur d'un flot*)**.** La valeur d'un flot compatible f entre s (source) et p (puits) est la somme des flux partant de la source s, est aussi égale à la somme des flux des arcs arrivant sur le puits p, et on note :

$$V(f) = \sum_{u \in \Omega^+(s)} f(u) = \sum_{u \in \Omega^-(p)} f(u)$$

On présente aussi, une autre définition qui est très utile dans la pratique :

Définition 1.10 (*Flot d'un sommet*). le flot d'un sommet est défini comme suit :

$$f(x) = \sum_{u \in \Omega^+(x)} f(u) - \sum_{u \in \Omega^-(x)} f(u) \qquad \forall\, x \in X$$

en particulier :

$$f(s) = \sum_{u \in \Omega^+(s)} f(u) - f(u_r)$$

et

$$f(p) = f(u_r) - \sum_{u \in \Omega^-(p)} f(u)$$

1.5 Conclusion

Dans ce chapitre, nous avons présenté quelques définitions sur les graphes simples orientés. Les principales définitions et quelques propriétés intéressantes ont été rappelées. Nous avons exposé aussi la notion de flot défini dans un graphe pondéré ou un réseau. Ces notions sont essentielles et indispensables pour aborder le problème de flot de coût minimum dans les chapitres ultérieurs.

Chapitre 2

Méthode de point intérieur pour la programmation linéaire

2.1 Introduction

Dans ce chapitre, nous présentons d'abord, quelques définitions géométriques telles que : polyèdre, point extrême et point intérieur, puis nous rappelons la position du problème de la Programmation Linéaire. Nous exposons brièvement, le principe de la méthode de point intérieur, plus précisément le « Affine Scaling » ainsi que leur développement mathématique pour la résolution d'un programme linéaire. Au final, nous donnons un algorithme traduisant cette méthode, et nous le testons sur un exemple simple.

2.2 Définitions et notions préliminaires

Dans cette section, nous énonçons les définitions de certains points particuliers tels que le point extrême et le point intérieur d'un ensemble convexe.

Définition 2.1 (Ensemble convexe)**.** Un ensemble S est appelé convexe si et seulement si deux points quelconques x et y pris dans X, le segment $[x, y]$ est contenu dans S.

Définition 2.2 (Polytope convexe)**.** Un ensemble convexe de la forme $S \coloneqq \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$ est appelé un polytope convexe.

Définition 2.3 (Polyèdre). Un polytope borné est appelé un polyèdre convexe.

Définition 2.4 (Point extrême)**.** Un point x d'un polyèdre S est dit point extrême ou sommet, s'il n'existe pas deux points x_1 et x_2 de S, avec $(x_1 \neq x_2)$, tels que $x = (x_1 + x_2)/2$.

Définition 2.5 (Point intérieur)**.** Un point $x \in S$ est dit point intérieur de S s'il existe $\epsilon > 0$, tel que : $\|y - x\| < \epsilon \Longrightarrow y \in S$, autrement dit, s'il existe une boule centrée en x contenue dans S.

Définition 2.6 (Ensemble intérieur)**.** L'ensemble de points intérieurs à S est appelé l'intérieur de S et noté int(S), ou S^0.

2.3 Le problème de la programmation linéaire

Un problème de programmation linéaire consiste à minimiser (ou maximiser) une fonction linéaire sous contraintes linéaires. Il s'agit d'un programme mathématique de la forme :

$$\begin{aligned} \text{Minimiser} \quad & z := c'x \\ \text{Sujet à} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{2.1}$$

où c est le vecteur des coûts de taille n; x est la variable de décision de taille n; z est la fonction économique (l'objectif); A est la matrice des contraintes de taille $m \times n$ avec $m \leq n$, supposée de plein rang (rang(A) = m) et b est le second membre de taille m.

Par opposition, le problème dual s'écrit :

$$\begin{aligned} \text{Maximiser} \quad & w := \lambda' b \\ \text{Sujet à} \quad & A'\lambda + r = c \\ & r \geq 0 \end{aligned} \tag{2.2}$$

où la variable positive r est appelée vecteur des coûts réduits ; et λ est la variable duale de décision de taille m.

2.4 Méthode de point intérieur «l'Affine Scaling »

Dans cette section, nous exposons le principe de la méthode de point intérieur « Affine scaling » appliquée sur le problème primal (2.1). Nous présentons aussi un algorithme pour résoudre des exemples de programme linéaire.

2.4.1 Principe de la méthode

Considérons le programme (2.1). Soit S l'ensemble de solutions réalisables :

$$S := \{x \in \mathbb{R}^n : Ax = b, \text{ et } x \geq 0\}$$

On note S^0 l'intérieur de S, c'est-à-dire : $S^0 := \{x \in \mathbb{R}^n : Ax = b, \text{ et } x > 0\}$.

On suppose que S^0, n'est pas vide, ses éléments sont des points intérieurs.

On considère aussi, l'ellipsoïde[2] E définie par :

$$E(y, \beta) := \left\{ x \in \mathbb{R}^n : \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{y_i^2} \leq \beta^2 \right\} \tag{2.3}$$

où y est le centre le l'ellipsoïde, et βy_i avec $(0 < \beta \leq 1)$ est la longueur des demi-axes, pout tout $(i = 1, \dots, n)$.

Posons $Y := \text{diag}(y_1, y_2, \dots, y_n)$, l'ellipsoïde s'écrit sous la forme matricielle, ainsi :

$$E(y, \beta) := \left\{ x \in \mathbb{R}^n : \left\| Y^{-1}(x - y) \right\|_2 \leq \beta \right\} \tag{2.4}$$

Idée principale :

Minimiser $z := c'x$ pour $x \in S$ peut être difficile, par contre il est facile de minimiser z pour x appartenant à l'ellipsoïde E, puisque la solution possède une forme analytique. Donc, au lieu de minimiser l'objectif directement sur S, nous allons minimiser une suite de problèmes d'optimisation sur des ellipsoïdes.

À une étape donnée, le problème qu'il faut résoudre est le suivant :

[2] L'ellipsoïde admet un centre et au moins trois plans de symétrie. L'intersection d'un ellipsoïde avec un plan est une ellipse, un point ou l'ensemble vide.

$$\begin{aligned} \text{Minimiser} \quad & z := c'x \\ \text{Sujet à} \quad & Ax = b \\ & \left\| Y^{-1}(x-y) \right\|_2 \leq \beta \end{aligned} \tag{2.5}$$

Posons : $d = x - y$, le programme (2.5) devient :

$$\begin{aligned} \text{Minimiser} \quad & c'd \\ \text{Sujet à} \quad & Ad = 0 \\ & \left\| Y^{-1}d \right\|_2 \leq \beta \end{aligned} \tag{2.6}$$

Le modèle mathématique (2.6) est un programme non linéaire. La direction admissible de descente $\hat{d}$ est calculée comme suit :

La fonction de Lagrange est définie pour tout $d \in \mathbb{R}^n$, $\lambda \in \mathbb{R}^m$, et $\mu \geq 0$, comme suit :

$$\mathscr{L}(d; \lambda, \mu) := c'd - \lambda' Ad + \mu \tfrac{1}{2}(d'Y^{-2}d - \beta^2)$$

Les conditions d'optimalité de Karush-Kuhn-Tucker s'écrivent :

$$(\text{KKT}) \equiv \begin{cases} c - A'\lambda + \mu Y^{-2}d = 0 & \text{(i)} \\ Ad = 0 & \text{(ii)} \\ \mu(d'Y^{-2}d - \beta^2) = 0 & \text{(iii)} \end{cases}$$

de l'égalité (i) on tire,

$$\mu d = Y^2(A'\lambda - c)$$

on multiplie par A, on trouve

$$\mu Ad = AY^2(A'\lambda - c) = 0$$

de l'égalité (ii) on a $Ad = 0$, donc le multiplicateur λ s'obtient en résolvant le système linéaire suivant :

$$(AY^2A')\lambda = AY^2c$$

Il nous reste qu'à régler le paramètre μ.

$$\mu Y^{-1}d = Y(A'\lambda - c)$$

de l'égalité (iii), on a $\left\| Y^{-1}d \right\| = \beta$, donc le paramètre β s'obtient par la formule :

$$\mu = \frac{\|Y(A'\lambda - c)\|}{\beta}$$

Au final, la direction admissible de descente se calcule par :

$$\hat{d} = -\frac{1}{\hat{\mu}} Y^2(c - A'\hat{\lambda}) = -\beta\, Y.\frac{Y(c - A'\hat{\lambda})}{\left\|Y\left(c - A'\hat{\lambda}\right)\right\|}$$

Le vecteur $\hat{r} := c - A'\hat{\lambda}$ joue un rôle important, c'est le vecteur approximatif des couts réduits du simplexe. La direction $\hat{d}$ peut s'écrire ainsi :

$$\hat{d} = -\beta\, Y.\frac{Y\hat{r}}{\|Y\hat{r}\|} \tag{2.7}$$

2.4.2 Algorithme « Affine Scaling »

La méthode que nous avons exposée précédemment, peut se traduire par l'algorithme suivant :

Algorithme 2.1 – Méthode de point intérieur « Affine Scaling »

0)	Les données : (A, b, c), et une solution de départ $x^0 > 0$, Une tolérance $\epsilon > 0$; et le paramètre $\beta \in]0\ ,1[$. Soit, e $:= (1,1,\dots,1)'$.
1)	Calcul des valeurs approchées du dual et des couts réduits : $X_k := \mathrm{diag}(x_1^k, x_2^k, \dots, x_n^k)$ $\left(AX_k^2A'\right)\lambda_k := AX_k^2c$ $r_k := c - A'\lambda_k$
2)	Calcul de la direction admissible de descente : $d_k := -X_k\frac{X_k r_k}{\|X_k r_k\|}$
3)	Test d'optimalité : si $r^k \geq 0$, et $e'X_k r^k < \epsilon$, alors STOP. x^k et λ^k sont les solutions ϵ-optimales du primal et du dual.
4)	Test de finitude : si $d_k \geq 0$, alors STOP. Solution non finie.
5)	Mise à jour de la solution primale : $x^{k+1} = x^k + \beta d_k$

Remarque 2.1. Les deux programmes mathématiques ci-dessous, sont équivalents.

$$(AY^2A')\lambda = AY^2c \iff \min_{\lambda \in \mathbb{R}^m} \|Y(A'\lambda - c)\|^2$$

2.4.3 Performances pratiques de l'algorithme

Le point critique dans l'algorithme (2-1) est le calcul, à chaque itération de la matrice AX_k^2A', qui nécessite $O(m^2n)$ opérations, puis il faut résoudre un système linéaire de matrice AX_k^2A', ce qui prend $O(m^3)$ opérations, alors chaque itération nécessite $O(m^2n + m^3)$, et comme $m \leq n$, au total, on a au plus $O(n^3)$ opérations par itération. Alors, plus la matrice A devient creuse, plus les performances de l'algorithme augmentes.

Remarque 2.2. Pour les résultats de convergence de la méthode, n'hésitez pas à consulter, par exemple [24].

2.5 Exemple numérique

Dans cette section, nous présentons le comportement de la méthode de point intérieur « l'Affine Scaling » sur un exemple simple. Pour ce faire, nous considérons le programme linéaire suivant :

$$(P) \equiv \begin{cases} (\max)\, z := 7x_1 + 9x_2 \\ -x_1 + x_2 \leq 2 \\ x_1 + 2x_2 \leq 7 \\ 2x_1 + x_2 \leq 8 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$$

Après addition des variables d'écart et transformation en un problème de minimisation, l'exemple s'écrit sous la forme standard comme suit :

$$(PS) \equiv \begin{cases} (\min)\, z := -7x_1 - 9x_2 + 0x_3 + 0x_4 + 0x_5 \\ -x_1 + x_2 + x_3 = 2 \\ x_1 + 2x_2 + x_4 = 7 \\ 2x_1 + x_2 + x_5 = 8 \\ x_i \geq 0, (i = 1, \dots, 5) \end{cases}$$

Tableau 2.1 – Les 6 premières itérations issues de l'algorithme.

$\beta := 0.995$

Itération (k)	0	1	2	3	4	5	6
Solution x^k	0.1000	1.8929	2.7669	2.9935	2.9955	2.9999	2.9999
	0.1000	2.4028	2.1098	1.9971	2.0020	1.9998	2.0001
	2.0000	1.4901	2.6571	2.9964	2.9935	3.0001	2.9998
	6.7000	0.3015	0.0136	0.0124	0.0006	0.0005	0.0000
	7.7000	1.8114	0.3564	0.0160	0.0071	0.0003	0.0002
Coût réduit r^k	-6.9917	-0.3540	-0.0179	0.0000	0.0000	0.0000	0.0000
	-8.9800	0.0737	0.0153	0.0001	0.0000	0.0000	0.0000
	0.0050	-0.7627	-0.0291	0.0000	0.0000	0.0000	0.0000
	0.0056	4.5964	3.7119	3.6667	3.6667	3.6667	3.6667
	0.0039	0.6435	1.6206	1.6667	1.6667	1.6667	1.6667
$r^{k\prime}x^k$	-1.5202	0.9219	0.5336	0.0722	0.0022	0.0004	0.0001
$c^\prime x^k$	-1.6000	-34.876	-38.356	-38.928	-38.986	-38.998	-38.999

La figure 2.1 représente le comportement de l'algorithme sur l'exemple.

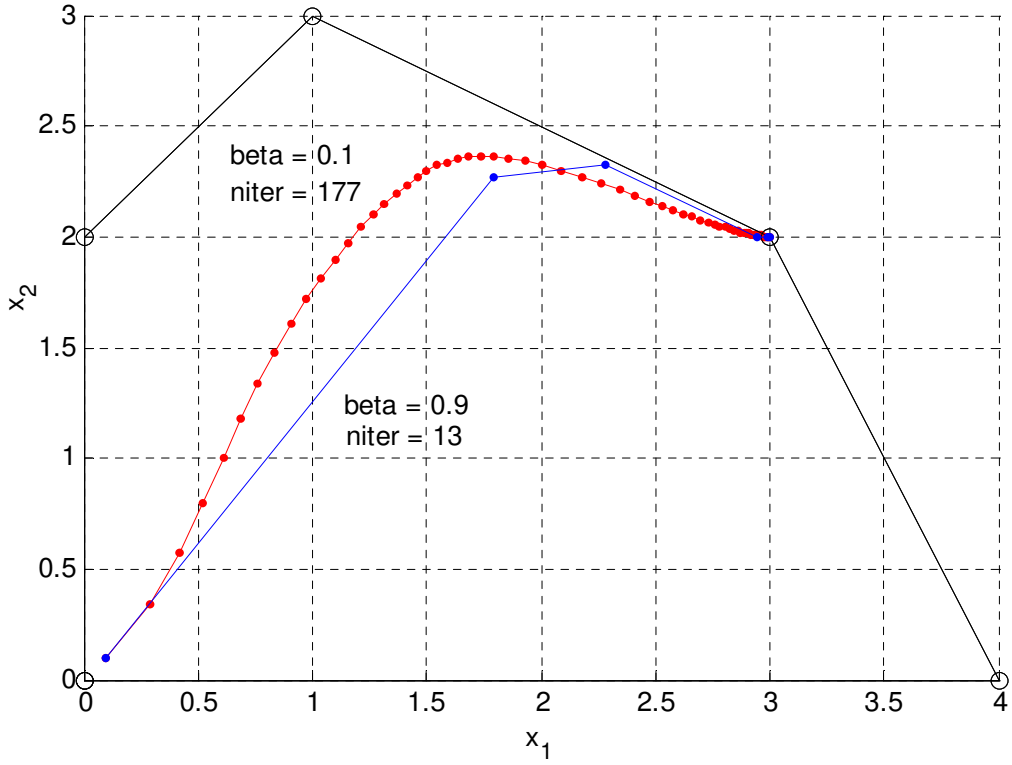


Figure 2.1 – Comportement de la méthode de point intérieur sur l'exemple pour différentes valeurs de beta (la tolérance ϵ est fixée à10^{-8}).

Sous la forme matricielle l'exemple s'écrit :

$$A = \begin{pmatrix} -1 & 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 & 1 \end{pmatrix} \qquad b = \begin{pmatrix} 2 \\ 7 \\ 8 \end{pmatrix} \qquad c = \begin{pmatrix} -7 \\ -9 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Le tableau 2.1, représente les 6 premières itérations de l'algorithme appliqué à l'exemple.

2.6 Conclusion

Dans ce chapitre, nous avons présenté, quelques notions géométriques telles que : polyèdre, point extrême et point intérieur, puis nous avons rappelé la position du problème de la programmation linéaire. Ensuite, nous avons exposé brièvement, le principe de la méthode de point intérieur, plus précisément, le « Affine Scaling » ainsi que leur développement mathématique pour la résolution d'un programme linéaire. Au final, nous avons donné un algorithme traduisant cette méthode, et nous l'avons testé sur un exemple simple.

Chapitre 3

Problème de flot de coût minimum et applications

3.1 Introduction

Le problème de flot de coût minimum est un modèle très général. En effet plusieurs problèmes peuvent être mis sous cette forme par exemple on cite : le problème de la recherche d'un plus court chemin et le problème de transport généralisé. Ce chapitre est consacré à la position du problème de flot de coût minimum. Tout d'abord, nous exposons la formulation mathématique du problème. Il s'agit d'un programme linéaire en variables bornées. Ensuite, nous présentons le problème de transport généralisé ainsi que son développement mathématique, et sa représentation graphique.

3.2 Position du problème

Dans cette section, nous posons le problème de flot de coût minimum. Il est défini comme suit :

Étant donné un réseau $R = (X, U, \alpha, \beta, c)$ où $c : U \longrightarrow \mathbb{R}^m$, $\alpha : U \longrightarrow \mathbb{R}^m \cup \{-\infty\}$ et $\beta : U \longrightarrow \mathbb{R}^m \cup \{+\infty\}$ avec $\alpha(u) \leq \beta(u)$ pour tout arc et $|U| = m$.

Le problème du flot de coût minimum sur R consiste à chercher un vecteur $f \in \mathbb{R}^m$ tel que :

i) f soit un flot sur le graphe $G = (X, U)$ (loi de conservation)

ii) $\alpha(u) \leq f(u) \leq \beta(u)$ $\forall\, u \in U$ (compatible)

iii) $\sum_{u \in U} c(u) f(u)$ soit minimum sous les conditions i) et ii).

3.3 Modèle mathématique

D'après la section précédente, le problème de flot de coût minimum sur $R = (X, U, \alpha, \beta, c)$ se formule comme un programme linéaire en variables bornées :

$$\text{Minimiser} \qquad z := c' f \tag{3.1}$$

$$\text{Sujet à} \qquad Af = 0 \tag{3.2}$$

$$\alpha \leq f \leq \beta \tag{3.3}$$

où A est la matrice incidente aux arcs du graphe $G = (X, U)$; α, β et c sont respectivement, la borne inferieure, la borne supérieure et le vecteur coût du flot f.

3.4 Applications au problème de transport généralisé

Plusieurs problèmes peuvent être mis sous la forme du problème de flot de coût minimum. Dans cette section, nous nous intéressons au problème de transport généralisé.

3.4.1 Position du problème et modélisation mathématique

Étant donné m dépôts capables d'offrir les quantités a_i, $(i = 1, \ldots, m)$ de certaines matières premières pour alimenter n clients sollicitant des quantités b_j, $(j = 1, \ldots, n)$. Le coût unitaire de transport c_{ij} engendré lors de l'affectation d'un client j à un dépôt i, est estimé et connu à priori. Les quantités transportées x_{ij} du dépôt i vers le client j, sont bornées. L'objectif est de minimiser le coût de transport total. Le problème se modélise par le programme linéaire en nombres bornés (PLB), suivant :

$$\text{Minimiser} \qquad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{3.4}$$

$$\text{Sujet à} \qquad \sum_{i=1}^{m} x_{ij} = b_j \qquad (j = 1, \ldots, n) \tag{3.5}$$

$$\sum_{j=1}^{n} x_{ij} = a_i \qquad (i = 1, \ldots, m) \tag{3.6}$$

$$l_{ij} \leq x_{ij} \leq u_{ij} \qquad (i = 1, \ldots, m);\ (j = 1, \ldots, n) \tag{3.7}$$

Description du modèle : Le critère (3.1) désigne le coût total qu'on cherche à minimiser (c'est l'objectif). Le groupe de contraintes (3.2) désigne que chaque client j nécessite une quantité b_j. Le groupe de contraintes (3.3) désigne que chaque dépôt i doit offrir une quantité a_i. Les variables de décision (3.4) indiquent, respectivement, les quantités minimales l_{ij} et maximales u_{ij}, transportées du dépôt i vers le client j.

3.4.2 Problème équilibré

Le problème de transport est dit équilibré si :

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j \tag{3.8}$$

Dans les contextes où l'équilibre entre l'offre et la demande n'est pas respecté, on peut toujours modifier le problème pour le rendre artificiellement équilibré.

3.4.3 Structure algébrique du modèle

La structure particulière du problème permet d'apercevoir des algorithmes efficaces pour le résoudre. Elle peut être vue clairement en écrivant les contraintes sous la forme standard ci-dessous.

Le premier groupe de contraintes (la demande) :

$$\begin{array}{cccccc}
x_{11} + x_{21} + \cdots + x_{m1} & & & & & = b_1 \\
& x_{12} + x_{22} + \cdots + x_{m2} & & & & = b_2 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \\
& & & x_{1n} + x_{2n} + \cdots + x_{mn} & & = b_n
\end{array}$$

Le deuxième groupe de contraintes (l'offre) :

$$\begin{array}{ccccccccc}
x_{11} & & x_{12} & & & & x_{1n} & & = a_1 \\
& x_{21} & & x_{22} & & & & x_{2n} & = a_2 \\
& \ddots & + & \ddots & + & & \ddots & & \vdots \\
& x_{m1} & & x_{m2} & & & & x_{mn} & = a_m
\end{array}$$

Le vecteur des coûts s'écrit :

$$c_{11} + c_{21} + \cdots + c_{m1} \quad , \quad c_{12} + c_{22} + \cdots + c_{m2} \quad , \ldots , \quad c_{1n} + c_{2n} + \cdots + c_{mn}$$

Le programme mathématique (3.1)-(3.4) s'écrit :

$$\text{Minimiser} \qquad z \coloneqq c'x \tag{3.9}$$

$$\text{Sujet à} \qquad Ax = b \tag{3.10}$$

$$l \leq x \leq u \tag{3.11}$$

où A est une matrice de taille $(m+n) \times (m \times n)$, b est un vecteur de taille $m+n$, et c est un vecteur de taille $m \times n$, l et u sont respectivement, la borne minimale et maximale du vecteur x.

$$A \coloneqq \begin{pmatrix} \mathrm{e}_m & & & \\ & \mathrm{e}_m & & \\ & & \ddots & \\ & & & \mathrm{e}_m \\ I_m & I_m & \cdots & I_m \end{pmatrix}, \quad b \coloneqq \begin{pmatrix} b_1 \\ \vdots \\ b_n \\ a_1 \\ \vdots \\ a_m \end{pmatrix}, \quad c \coloneqq \begin{pmatrix} c_{11} \\ \vdots \\ c_{m1} \\ c_{12} \\ \vdots \\ c_{m2} \\ \vdots \\ c_{1n} \\ \vdots \\ c_{mn} \end{pmatrix}, \quad \text{et} \quad x \coloneqq \begin{pmatrix} x_{11} \\ \vdots \\ x_{m1} \\ x_{12} \\ \vdots \\ x_{m2} \\ \vdots \\ x_{1n} \\ \vdots \\ x_{mn} \end{pmatrix}$$

où, e_m est un vecteur ligne de taille m, et I_m est une matrice d'identité de taille $m \times m$.

3.4.4 Propriétés de la matrice *A*

La matrice A possède des propriétés captivantes, permettant d'appliquer une méthode adéquate telle que la méthode du point intérieur, pour la résolution du problème de manière efficace.

Propriété 3.1. Le nombre de lignes de A est $m+n$. Il n'y a que $m+n-1$ lignes qui sont linéairement indépendantes. Donc on peut éliminer une contrainte.

Propriété 3.2. La matrice A est creuse, sa densité est définie par le pourcentage suivant :

$$d(A) \coloneqq \frac{\text{nombre de valeurs non nulles}}{\text{nombre total}} = \frac{2mn}{(m+n-1) \times (mn)} = \frac{2}{(m+n-1)} \ll 1$$

3.5 Représentation du problème de transport sous forme d'un problème de flot de coût minimum

Dans cette section, nous allons voir comment le problème de transport généralisé peut se mettre sous la forme d'un problème de flot de coût minimum.

On rappelle que, le problème de transport généralisé est défini à partir :

- d'un graphe $G = (X, U)$
- des applications $\alpha, \beta, c : U \longrightarrow \mathbb{R}$ avec $\alpha(u) \leq \beta(u),\ \forall\, u \in U$.

Il s'agit de déterminer sur le réseau $R = (X, U, \alpha, \beta, c)$, un flot compatible $f \in \mathbb{R}^{|U|}$ tels que : $\sum_{u \in U} c(u) f(u)$ soit minimum. Le programme mathématique associé est donné

$$(\mathscr{F}) \begin{cases} c' f = z(\text{Min}) \\ Af = 0 \\ \alpha \leq f \leq \beta \end{cases}$$

On construit le réseau $R = (X, U, \alpha, \beta, c)$ avec :

a) L'ensemble des sommets X.

$$X = D \cup C \cup \{S, P\}$$

où

$$D = \{D_1, D_2, \ldots, D_m\} \qquad \text{et} \qquad C = \{C_1, C_2, \ldots, C_n\}$$

b) L'ensemble des arcs U.

$$U = U_1 \cup U_2 \cup U_3 \cup \{u_r\}$$

où

$$U_1 = \{(SD_i), i = 1, \ldots, m\} \quad , \quad U_2 = \left\{\left(D_i C_j\right), i = 1, \ldots, m \text{ et } j = 1, \ldots, n\right\}$$
$$U_3 = \left\{\left(C_j P\right), j = 1, \ldots, n\right\} \quad \text{et} \quad u_r = (PS)$$

c) Capacité inferieure et supérieure, ainsi que le coût de chaque arc $u \in U$.

	$u = (SD_i)$	$u = (C_jP)$	$u = (D_iC_j)$	$u = (SP)$
Coût : $c(u)$	0	0	c_{ij}	0
Capacité inf : $\alpha(u)$	a_i	b_j	l_{ij}	0
Capacité sup : $\beta(u)$	a_i	b_j	u_{ij}	∞

Cet exemple, peut être schématisé par la figure 3.1, ci-dessous.

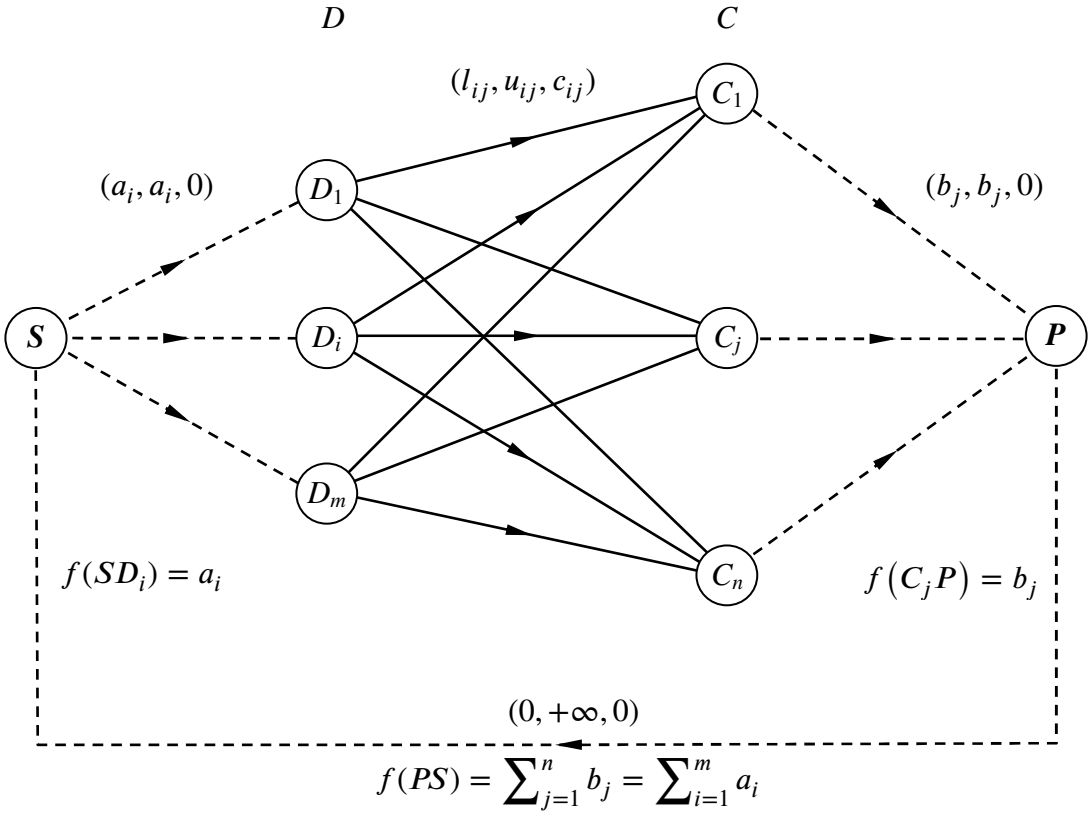


Figure 3.1 – Représentation du problème de transport généralisé sous forme d'un problème de flot de coût minimum.

Ainsi, le problème de transport généralisé se ramène à un problème du flot de coût minimum, et nous allons voir comment on peut le résoudre en appliquant une méthode de type point intérieur (voir chapitre 2).

3.6 Processus de résolution

La résolution de notre problème passe par plusieurs étapes. Ces dernières peuvent être schématisées par le diagramme représenté dans la figure 3.2

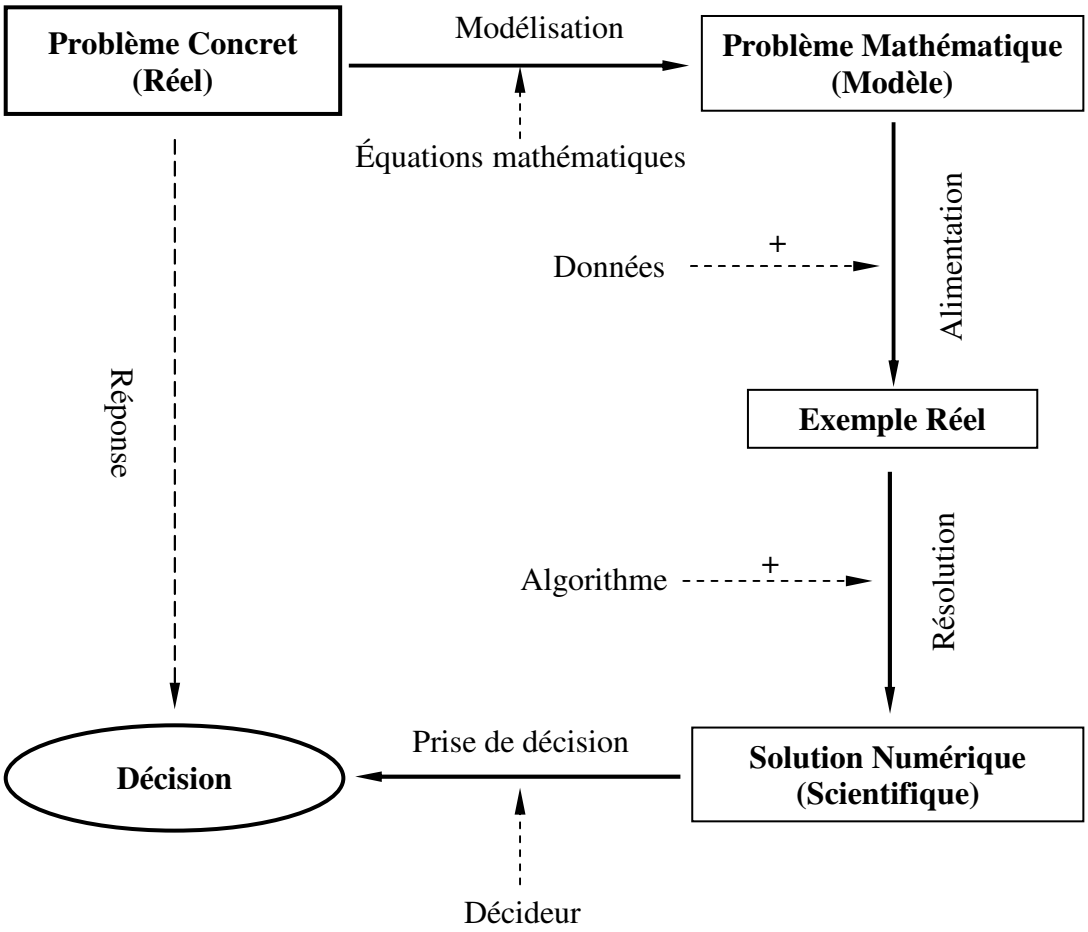


Figure 3.2 – Schéma conceptuel de résolution d'un problème réel.

3.7 Conclusion

Le problème de flot de coût minimum constitue une classe générale permettant de modéliser et de résoudre de nombreux problèmes pratiques, à titre d'exemple on cite : le problème de la recherche d'un plus court chemin et le problème de transport généralisé.

Dans ce chapitre, nous avons posé le problème de flot de coût minimum, puis, nous avons exposé la formulation mathématique du problème, Il s'agit d'un

programme linéaire en variables bornées. Ensuite, nous avons présenté le problème de transport généralisé ainsi que son développement mathématique et sa représentation graphique. Dans le chapitre 4, nous allons voir comment nous pouvons résoudre une instance du problème de transport généralisé, en appliquant la méthode de type point intérieur, qui est exposée dans le chapitre 2.

Chapitre 4

Implémentation et tests numériques

4.1 Introduction

Dans ce chapitre, nous présentons l'implémentation de notre approche de résolution. Nous reportons aussi des résultats numériques obtenus en testant notre programme Matlab sur un exemple réel de transport généralisé qui est un cas particulier du problème de flot.

4.2 Adaptation de l'algorithme pour la résolution des programmes linéaire en variables bornées

Dans cette section, nous présentons une adaptation de l'algorithme de « l'affine scaling » classique (vu en chapitre 2), pour la résolution du programme linéaire en variables bornées suivant :

$$\text{Minimiser} \qquad z := c'x \tag{4.1}$$

$$\text{Sujet à} \qquad Ax = b \tag{4.2}$$

$$l \leq x \leq u \tag{4.3}$$

4.2.1 Réglage des longueurs des demis axes de l'ellipsoïde

L'algorithme classique de « l'affine scaling », consiste à minimiser la fonction objective sur l'ellipsoïde suivant :

$$E(y, \beta y) := \left\{ x \in \mathbb{R}^n : \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{(\beta y_i)^2} \leq 1 \right\} \tag{4.4}$$

Les demi-longueurs des axes de ellipsoïde E sont βy_i, pour tout $(i = 1, \ldots, n)$.

Pour qu'une solution x soit toujours localisée dans le domaine défini par : $l \leq x \leq u$, nous proposons de chercher à optimiser l'objectif sur l'ellipsoïde suivant :

$$\tilde{E}(y, \beta r) \coloneqq \left\{ x \in \mathbb{R}^n \,:\, \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{(\beta r_i)^2} \leq 1 \right\} \tag{4.5}$$

Les demi-longueurs des axes de ellipsoïde $\tilde{E}$ sont βr_i, pour tout $(i = 1, \ldots, n)$.
où

$$r_i \coloneqq \min(y_i - l_i, u_i - y_i) \qquad (i = 1, \ldots, n) \tag{4.6}$$

Dans ce cas, l'écriture matricielle de l'ellipsoïde $\tilde{E}(y, \beta r)$ est :

$$\tilde{E}(y, \beta r) \coloneqq \left\{ x \in \mathbb{R}^n \,:\, \left\| Y^{-1}(x - y) \right\|_2^2 \leq 1 \right\} \tag{4.7}$$

où Y est la matrice diagonale définie par :

$$Y \coloneqq \operatorname{diag}\{\beta r_1, \beta r_2, \ldots, \beta r_n\} \qquad (i = 1, \ldots, n) \tag{4.8}$$

4.2.2 Détermination du point intérieur de départ

Pour son démarrage, l'algorithme « affine-scaling » nécessite une solution réalisable initiale. Ici, la solution est cherchée sous la forme de produit suivant :

$$x_{ij} = \alpha_i \times \beta_j \qquad \begin{matrix} (i = 1, \ldots, m) \\ (j = 1, \ldots, n) \end{matrix} \tag{4.9}$$

on trouve :

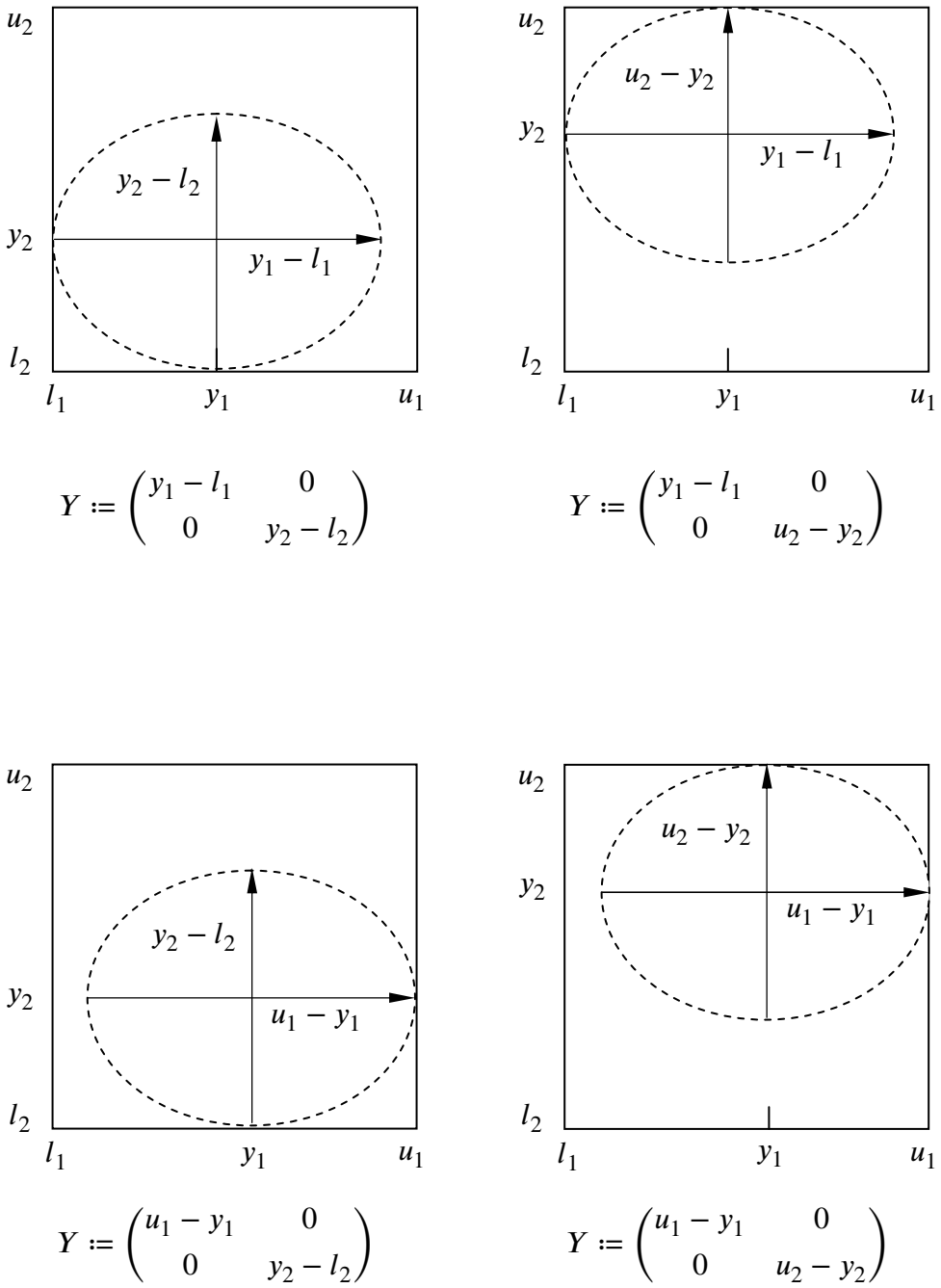
$$\alpha_i = \frac{a_i}{\sum_{j=1}^{n} \beta_j} \qquad (i = 1, \ldots, m)$$

$$\beta_j = \frac{b_j}{\sum_{j=1}^{m} \alpha_i} \qquad (j = 1, \ldots, n)$$

Au final, on obtient :

$$x_{ij} = \frac{a_i \times b_j}{S} \qquad \begin{matrix} (i = 1, \ldots, m) \\ (j = 1, \ldots, n) \end{matrix} \tag{4.10}$$

où $S = \sum_{i=1}^{m} \alpha_i \times \sum_{j=1}^{n} \beta_j = \sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$.



$$Y := \begin{pmatrix} y_1 - l_1 & 0 \\ 0 & y_2 - l_2 \end{pmatrix} \qquad Y := \begin{pmatrix} y_1 - l_1 & 0 \\ 0 & u_2 - y_2 \end{pmatrix}$$

$$Y := \begin{pmatrix} u_1 - y_1 & 0 \\ 0 & y_2 - l_2 \end{pmatrix} \qquad Y := \begin{pmatrix} u_1 - y_1 & 0 \\ 0 & u_2 - y_2 \end{pmatrix}$$

Figure 4.1 – La forme de l'ellipse $E(y, \beta r)$ en fonction de la position du centre y dans le pavé $\left[l_1\ ,\ u_1\right] \times \left[l_2\ ,\ u_2\right]$.

4.2.3 Algorithme de « l'Affine Scaling » modifié

La méthode adaptée, que nous avons exposée précédemment, peut se traduire par l'algorithme suivant :

Algorithme 4.1 – Méthode modifiée de point intérieur « Affine Scaling »

0) Les données : (A, b, l, u, c), et une solution de départ $x^0 > 0$,

Une tolérance $\epsilon > 0$; et le paramètre $\beta \in]0\ ,\ 1[$.

1) Calcul des multiplicateurs λ_k, en résolvant le système linéaire suivant :

$$X_k := \operatorname{diag}\left\{\min(x_i^k - l_i, u_i - x_i^k), \qquad i = 1, \dots, n\right\}$$

$$\left(AX_k^2A'\right)\lambda_k := AX_k^2c$$

2) Calcul de la norme du vecteur z_k:

$$z_k = X_k(c - A'\lambda_k), \text{ et } \mu = \|z_k\|$$

3) Si $\mu < \epsilon$, alors STOP et x^k est la solutions ϵ-optimale du primal.

4) Calcul de la direction admissible de descente :

$$d_k := -X_k\left(\frac{z_k}{\mu}\right)$$

5) Mise à jour de la solution primale : $x^{k+1} = x^k + \beta d_k$

4.3 Exemple test de transport généralisé

Dans cette section, nous considérons l'exemple de problème de transport généralisé suivant :

On cherche à expédier un produit directement de $m = 3$ origines à $n = 5$ destinations. Sur les routes empruntées, des conditions quant au poids maximal ou minimal permis, ont été imposées. Les origines sont les dépôts, et les destinations sont les clients. Le but est de minimiser le coût total de cette opération.

Le tableau 4.1 représente le coût unitaire de transport entre les dépôts et les clients.

Tableau 4.1 – Coûts unitaires en millier de dollar (M$).

	Clients				
Dépôts	C_1	C_2	C_3	C_4	C_5
D_1	55	50	45	53	60
D_2	45	43	43	51	45
D_3	40	53	36	60	57

Le tableau 4.2 et le tableau 4.3 donnent respectivement, le nombre d'unités disponibles en chaque dépôt, et la demande de chaque client.

Tableau 4.2 – Quantités disponibles (en tonnes).

Dépôts	D_1	D_2	D_3	Total
Disponibilités	100	120	130	350

Tableau 4.3 – Quantités requises (en tonnes).

Clients	C_1	C_2	C_3	C_4	C_5	Total
Demandes	50	60	70	80	90	350

Un tel problème de transport où l'offre totale est égale à la demande totale est dit équilibré. Dans les contextes où l'équilibre entre l'offre et la demande n'est pas respecté, on peut toujours modifier le problème pour le rendre artificiellement équilibré. Ce problème peut se ramener à chercher d'un flot de cout minimum dans un réseau, comme l'indique la figure 4.2.

4.4 Résultats numériques

Dans cette section, nous présentons les résultats numériques que nous avons obtenus sur l'exemple test cité dans la section précédente.

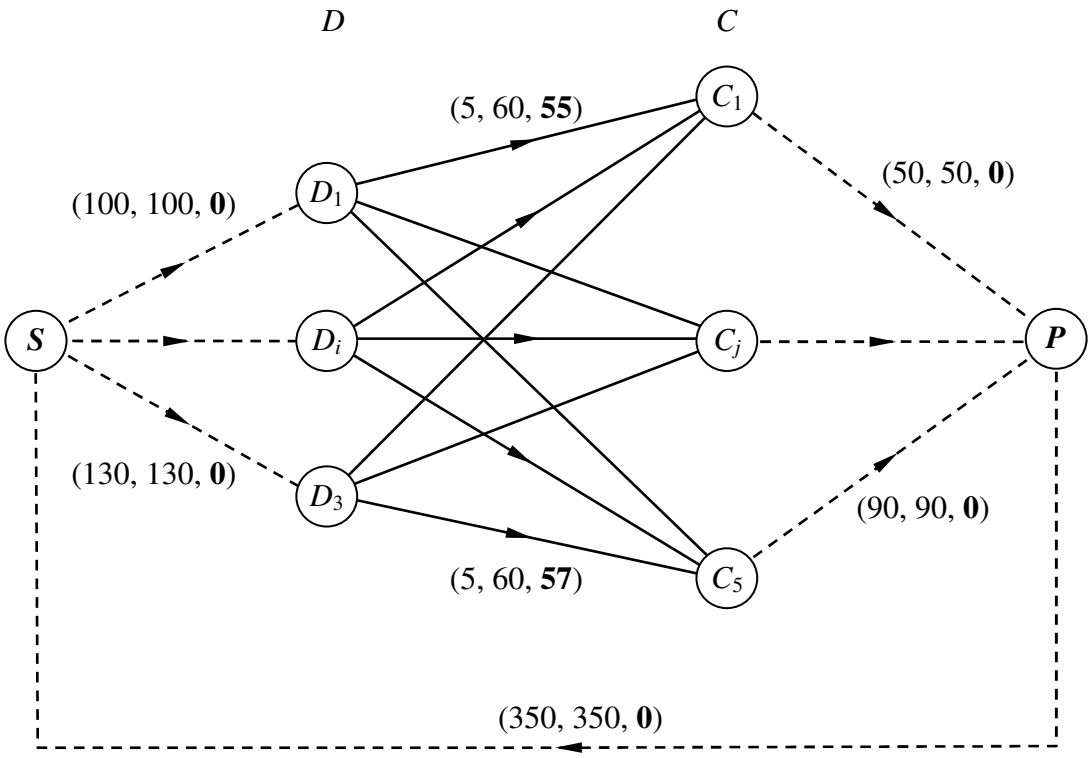


Figure 4.2 – Représentation de l'exemple test sous forme d'un réseau.

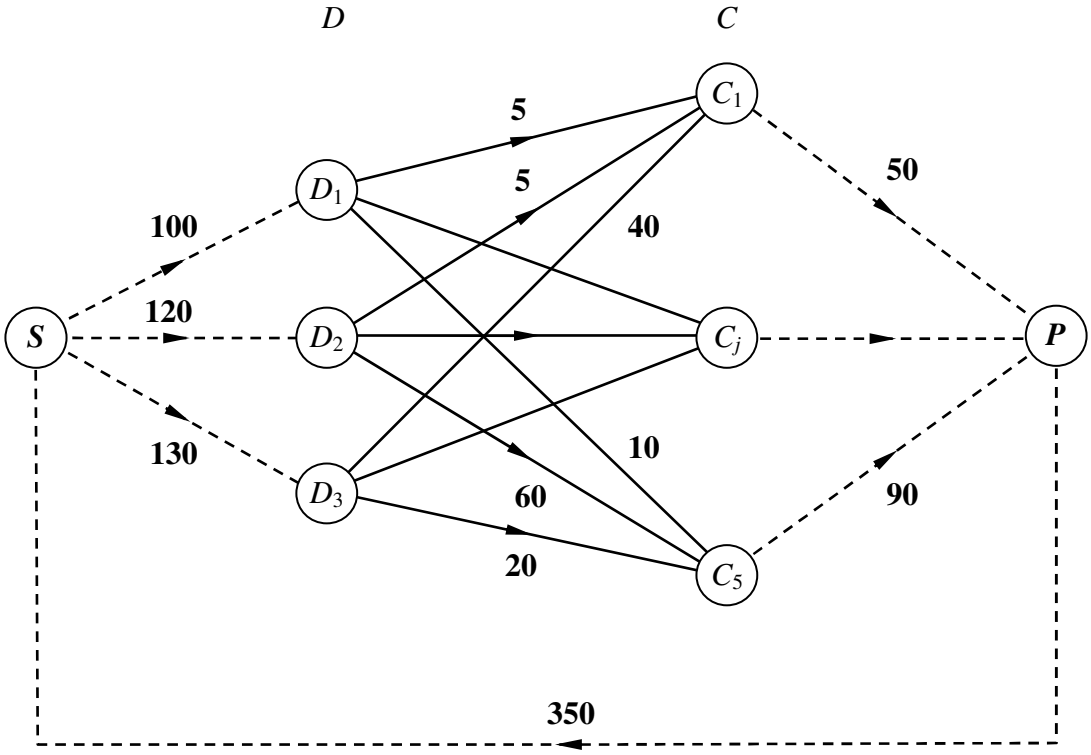


Figure 4.3 – Représentation de la solution optimale de l'exemple.

Tableau 4.4 – Solution initiale (point intérieur) et coût total de transport.

Itération N°0	C_1	C_2	C_3	C_4	C_5	a_i
D_1	14.2857	17.1429	20.0000	22.8571	25.7143	**100**
D_2	17.1429	20.5714	24.0000	27.4286	30.8571	**120**
D_3	18.5714	22.2857	26.0000	29.7143	33.4286	**130**
b_j	**50**	**60**	**70**	**80**	**90**	**350**
Coût	17 321					

Tableau 4.5 – Solution et coût total de transport après une itération.

Itération N°1	C_1	C_2	C_3	C_4	C_5	a_i
D_1	9.8841	19.7868	18.1153	36.9822	9.8841	**100**
D_2	12.1727	23.5230	5.0000	27.4493	12.1727	**120**
D_3	27.9432	16.6901	46.8847	15.5685	27.9432	**130**
b_j	**50**	**60**	**70**	**80**	**90**	**350**
Coût	16 660					

Tableau 4.6 – Solution et coût total de transport après 2 itérations.

Itération N°2	C_1	C_2	C_3	C_4	C_5	a_i
D_1	5.0000	19.9962	7.0213	57.0671	10.9155	**100**
D_2	5.4231	32.3000	5.0000	17.8680	59.4089	**120**
D_3	39.5769	7.7039	57.9787	5.0649	19.6756	**130**
b_j	**50**	**60**	**70**	**80**	**90**	**350**
Coût	16 207					

Tableau 4.7 – Solution optimale et coût total de transport après 9 itérations.

Itération N°9	C_1	C_2	C_3	C_4	C_5	a_i
D_1	5	20	5	60	10	**100**
D_2	5	35	5	15	60	**120**
D_3	40	5	60	5	20	**130**
b_j	**50**	**60**	**70**	**80**	**90**	**350**
Coût	16 155					

4.5 Comparaison concise avec l'algorithme du simplexe

Pour la validation de notre approche de résolution, nous comparons nos résultats obtenus, en utilisant notre algorithme, avec ceux obtenus par l'algorithme du simplexe qui est implémenté dans la composante logicielle, nommée SOLVER et qui est installée dans EXCEL.

L'outil appelé SOLVER est un outil permettant de modéliser et de résoudre des problèmes d'optimisation de petite taille. (Pour plus de détails nous renvoyons le lecteur à [6]).

En passant en revue les résultats numériques obtenus, en appliquant les deux approches précédentes, nous constatons qu'ils sont les mêmes comme le montre la figure A.2.

4.6 Conclusion

Dans ce chapitre, nous avons présenté l'implémentation de notre approche de résolution. Nous avons reporté aussi, des résultats numériques obtenus, en testant notre programme MATLAB, sur un exemple réel du problème.

Durant notre étude, nous avons constaté que de nombreux tests numériques effectués pour la résolution du programme linéaire en variables bornées, ont permis d'avérer les avantages de l'utilisation de la méthode de point intérieur. Les résultats obtenus affirment l'efficacité de cette approche.

Conclusion

Dans la course à la performance exigée par le contexte économique actuel, la rapidité de la prise de décision peut s'avérer primordiale. C'est la raison pour laquelle les gestionnaires et les ingénieurs cherchent toujours à résoudre leurs problèmes par des méthodes efficaces (au sens de précision et de rapidité).

Dans ce mémoire, nous nous sommes intéressés au problème de flot de coût minimum qui est un problème algorithmique de théorie des graphes, consistant à trouver la manière la plus économe d'utiliser un réseau de transport tout en satisfaisant les contraintes de production et de demande des nœuds du réseau. Il permet de modéliser tout un ensemble de problèmes pratiques dans lesquels il s'agit de trouver une manière optimale d'acheminer une ressource par exemple, un fluide ou de l'électricité, d'un ensemble de sources à un ensemble de puits.

Ce problème optimisation provient du domaine pratique très vaste, tel que la logistique, la modélisation de transports et les réseaux de communication. Il se formule comme un programme linéaire en variables continues et bornés.

Le problème du flot de coût minimum est fondamental dans la mesure où la plupart des autres problèmes de flots peuvent en être vus comme des cas particuliers. De plus, il est possible de résoudre le problème dans certains cas de manière efficace en utilisant l'algorithme du simplexe adapté pour les réseaux (l'algorithme de Ford et Fulkerson). Dans ce mémoire, nous avons utilisé une méthode de type point intérieur, plus précisément « l'Affine Scaling ».

La démarche scientifique que nous avons adoptée durant ce travail, se résume comme suit.

En premier lieu, nous avons posé le problème de flot de coût minimum ainsi que sa formulation mathématique, puis nous avons présenté une formulation mathématique modélisant le problème de transport généralisé, et nous avons montré aussi qu'il est possible de le ramener au problème de flot de coût minimum.

Ensuite, nous avons exposé le principe de la méthode de type point intérieur en présentant un algorithme baptisé « l'Affine Scaling », et nous l'avons adaptée pour la résolution du problème de flot de coût minimum.

Au final, nous avons élaboré et implémenté un programme informatique codé en langage Matlab pour la résolution d'une instance du problème. Les résultats numériques obtenus ont été apportés et commentés.

Durant notre étude, nous avons constaté que de nombreux tests numériques effectués pour la résolution du programme linéaire, ont permis d'avérer les avantages de l'utilisation de l'algorithme « l'Affine Scaling ». Les résultats obtenus affirment l'efficacité de cette approche.

Ces résultats nous conduisent à envisager quelques perspectives. Ces idées s'articulent autour les points suivant :

- Étudier le comportement de l'algorithme sur des instances de grande taille du problème, en estimant sa complexité algorithmique.
- Penser à appliquer une autre variante de l'algorithme baptisé « Réduction du potentiel », pour la résolution du problème de flot de coût minimum.

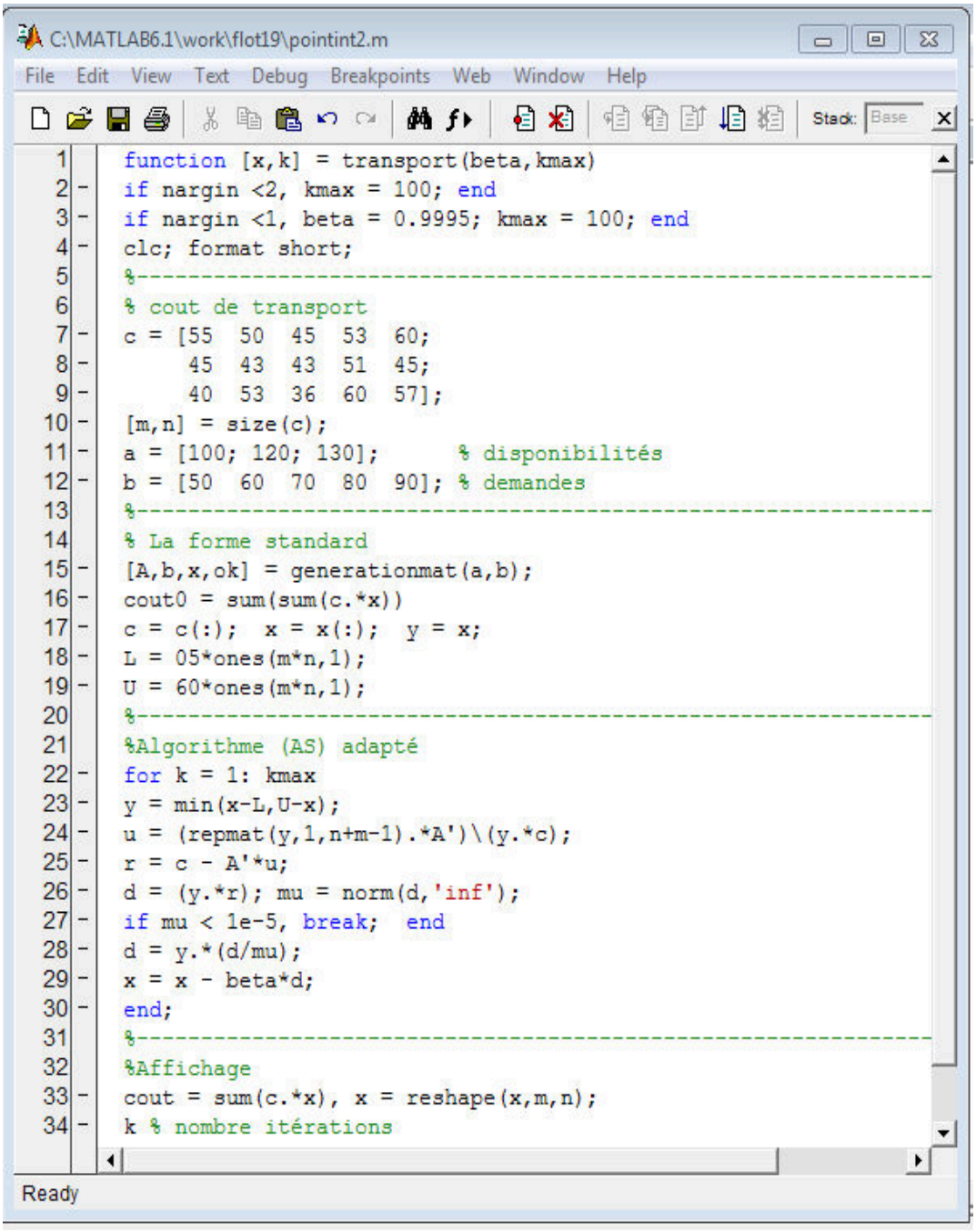
Nous espérons que notre but est atteint, et le travail que nous avons réalisé dans ce mémoire, nous a permis d'approfondir nos connaissances théoriques et pratiques dans le domaine de l'optimisation et d'acquérir des connaissances dans le domaine de l'informatique.

Annexe A

Codes Matab et Excel

A.1 Implémentation et mise en œuvre avec Matlab

Ce fichier consiste à regrouper les données caractérisant un exemple du problème de transport généralisé, et à écrire le modèle mathématique sous la forme standard, et aussi, à exécuter notre programme en affichant de résultats numériques relatifs à notre exemple test.

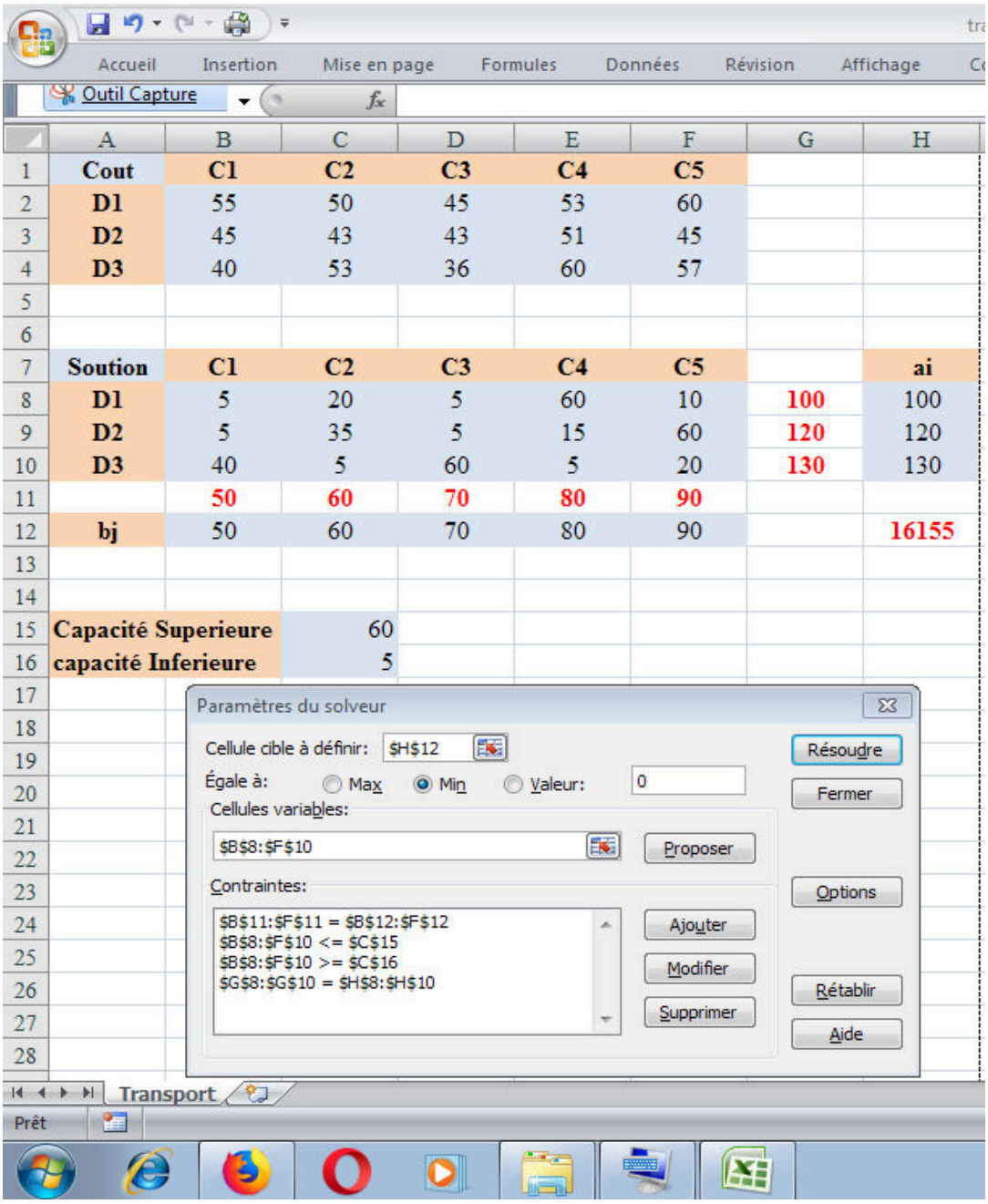



```
1  function [x,k] = transport(beta,kmax)
2  if nargin <2, kmax = 100; end
3  if nargin <1, beta = 0.9995; kmax = 100; end
4  clc; format short;
5  %-------------------------------------------------------------
6  % cout de transport
7  c = [55  50  45  53  60;
8       45  43  43  51  45;
9       40  53  36  60  57];
10 [m,n] = size(c);
11 a = [100; 120; 130];      % disponibilités
12 b = [50  60  70  80  90]; % demandes
13 %-------------------------------------------------------------
14 % La forme standard
15 [A,b,x,ok] = generationmat(a,b);
16 cout0 = sum(sum(c.*x))
17 c = c(:);  x = x(:);  y = x;
18 L = 05*ones(m*n,1);
19 U = 60*ones(m*n,1);
20 %-------------------------------------------------------------
21 %Algorithme (AS) adapté
22 for k = 1: kmax
23 y = min(x-L,U-x);
24 u = (repmat(y,1,n+m-1).*A')\(y.*c);
25 r = c - A'*u;
26 d = (y.*r); mu = norm(d,'inf');
27 if mu < 1e-5, break;  end
28 d = y.*(d/mu);
29 x = x - beta*d;
30 end;
31 %-------------------------------------------------------------
32 %Affichage
33 cout = sum(c.*x), x = reshape(x,m,n);
34 k % nombre itérations
```

Figure A.1 – Représentation et résolution de l'exemple test avec MATLAB.

A.2 Implémentation et mise en œuvre avec Excel

Ce fichier est un programme informatique qui permet la résolution du problème en temps réel, en utilisant la composante logicielle appelée SOLVEUR qui est intégrée sous l'application EXCEL. Cette composante utilise la méthode du simplexe pour résoudre les programmes linéaires. Pour plus de détails sur le sujet, nous renvoyons le lecteur à ([9],[15],[23]).



	A	B	C	D	E	F	G	H
1	**Cout**	**C1**	**C2**	**C3**	**C4**	**C5**		
2	**D1**	55	50	45	53	60		
3	**D2**	45	43	43	51	45		
4	**D3**	40	53	36	60	57		
5								
6								
7	**Soution**	**C1**	**C2**	**C3**	**C4**	**C5**		**ai**
8	**D1**	5	20	5	60	10	**100**	100
9	**D2**	5	35	5	15	60	**120**	120
10	**D3**	40	5	60	5	20	**130**	130
11		**50**	**60**	**70**	**80**	**90**		
12	**bj**	50	60	70	80	90		**16155**
13								
14								
15	**Capacité Superieure**		60					
16	**capacité Inferieure**		5					
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								
27								
28								

Figure A.2 – Représentation et résolution de l'exemple test avec EXCEL.

Notations et abréviations

Symboles

$\mathbb{N}$	L'ensemble des nombres entiers.
$\mathbb{R}$	L'ensemble des nombres réels.
$\mathbb{R}^n$	Le produit cartésien : $\underbrace{\mathbb{R} \times \cdots \times \mathbb{R}}_{n \text{ fois}}$.
x	Vecteur de taille n : ($x_1, x_2, \ldots, x_n$).
$x \geq 0$	Vecteur positif : $x \in \mathbb{R}^n_+$
$\sum_{i=1}^{n} x_i$	La somme : $x_1 + x_2 + \cdots + x_n$.
$\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}$	La somme double : $(x_{11} + \cdots + x_{1n}) + (x_{21} + \cdots + x_{2n}) + \cdots + (x_{m1} + \cdots + x_{mn})$.
x'	Le vecteur transposé de x.
$\sum_{i=1}^{n} x_i\, y_i$, ou $x'y$	Le produit scalaire du vecteur x et y.
$\|x\|_2$, ou $\sqrt{x'x}$	La norme euclidienne.
max z ; min z	Maximiser la valeur z ; Minimiser la valeur z.

Abréviations

PL	Programme linéaire.
PLB	Programme linéaire en variables bornées.
KKT	Les conditions d'optimalité locale du 1er ordre de Karush, Khun et Tuker.

Références

[1] A. Antoniou, W. Lu. Practical optimization, Algorithms and Engineering Applications, Springer, 2007.

[2] E. R. Barnes. "A variation on Karmarkar's algorithm for solving linear programming problems". Mathematical programming 36 (2): 174–182. 1986.

[3] D. P. Bertsekas. Constrained optimization and Lagrange multiplier methods. Academic press, New-York, 1982.

[4] J.C. Culioli. Introduction à l'optimisation, École Nationale Supérieure des Mines de Paris, Centre Automatique et système, Fontainebleau, Ellipse, 1994.

[5] G.B. Dantzing, Linear Programming and Extensions. Princeton. University Press, 1964.

[6] Y. Dodge, S. Gonano-Weber et J.P. Renfer, Optimisation Appliquée, Springer-Verlag France, 2005.

[7] A. V. Fiacco and G. P. McCormick. Nonlinear programming : Sequential unconstrained minimization techniques. S.I.A.M., 1990.

[8] J. Frédéric Bonnans, Claude Lemaréchal, J. Charles Gilbert. Numerical Optimization, Theoretical and Practical Aspects, Springer, 2003.

[9] P. E. Gill, W. Murray, and M. H. Wright. Practical Optimization, Academic Press, 1981.

[10] M. Gondran, M. Minoux. Programmation mathématique, Théorie et Algorithmes, Dunod, 1983.

[11] N. K. Karmarkar. A new polynomial time algorithm for linear programming, Combinatorica, 4 : 373-395, 1984.

[12] D. G. Luenberger, Y. Ye. Linear and Nonlinear Programming, Springer, 2008.

[13] M. Minoux. Programmation mathématique, Théorie et algorithmes, Collection Technique et Scientifique des Télécommunications, Dunod, Paris, 1983.

[14] W. Murray. Second derivative methods. In W. Murray, editor. Numerical methods for unconstrained optimization, Academic press, London, 1972.

[15] J. Nocedal, S. J. Wright. Numerical Optimization, Springer, 1999.

[16] C. H. Papadimitriou, K. Steiglitz. Combinatorial optimization. Prentice Hall, 1982.

[17] E. Polak. Computational methods in optimization : a unified approach. Academic press, New-York, 1971.

[18] M. J. D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, Optimization, Academic Press, London, 1969.

[19] M. Sakarovitch, Graphes et Programmation linéaire : Méthodes Mathématiques et Algorithmiques, Programmation Discrète, Hermann, Paris, 1984.

[20] M. Sakarovitch, Optimisation Combinatoire : Méthodes Mathématiques et Algorithmiques, Programmation Discrète, Hermann, Paris, 1984.

[21] S.M. Sinha. Mathematical Programming, Theory and Methods, Elsevier, 2006.

[22] W. Sun, Y. Yuan. Optimization Theory and Methods Nonlinear Programming, Springer, 2006.

[23] R. J. Vanderbei. Linear Programming : Foundations and Extensions. Springer Verlag. pp. 333–347. 2001.

[24] R. J. Vanderbei, J. C. Lagarias, ”I. I. Dikin’s Convergence Result for the Affine-Scaling Algorithm. Contemporary Mathematics. 114. 1990.