République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieure et de la recherche scientifique
Université Saad Dahlab Blida

Faculté des sciences

Département de Mathématiques

En vue d'obtenir le diplôme de Master

Spécialité : Recherche Opérationnelle

Thème

PROBLÈMES DES FLOTS ET DE TRANSPORT

Mémoire présenté par :

Mariche Ibrahim Messadi Hocine

Soutenu le 17 juillet 2017, devant les jury composé de :

Dr. LABBI Wafaa	(M.A.A)	U. d'Alger 3	Présidente
Dr. BENDRAOUCHE Mohamed	(M.C.A)	USD. Blida 1	Promoteur
Dr. AMROUCHE Karim	(M.A.A)	U. d'Alger 3	Examinateur

2016/2017

Table des matiéres

Remerciements

> Grâce à Allah, nous avons abouti à la concrétisation de ce travail.

En préambule à ce mémoire, nous souhaiterons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.

On remercie notre promoteur Dr BENDRAOUCHE Mohamed pour avoir accepté de nous encadrer tout au long de ce travail, pour son amabilité, sa disponibilité, son aide, ses conseils et suggestions et le temps qu'il a bien voulu nous consacrer malgré ses charges académiques et professionnelles.

Que tout enseignant nous ayant fait bénéficier de son savoir durant tout notre cursus universitaire, trouve ici l'expression de notre profonde gratitude.

Nous remercions les membre de jury :

Dr LABBI WAFAA, Maitre assistante A à l'université d'Alger 3, pour l'honneur qu'il nous fait en acceptant la présidence du jury.

Madame K. AMROUCHE, Maitre de conférences B à l'université d'Alger 3, pour avoir accepté de juger ce travail.

Nous n'oublions pas nos familles, et spécialement nos parents pour leur contribution, leur soutien et leur patience.

Enfin, nous ne saurons oublier dans ces remerciements tous ceux qui nous ont aidé pour mener à bien ce travail dans de bonnes conditions.

RESUME

Dans ce mémoire nous avons étudié deux problèmes d'optimisation. Le premier concerne les problèmes des flots : problème du flot maximum et problème du flot de coût minmum. Le deuxième problème considéré dans ce travail est le problème de transport. Des définitions ainsi que des méthodes de résolution par la théorie des graphes et la programmation linéaire ont été présentées.

ABSTRACT

In this memory we have studied two optimisation problems. The first concerns the flow problems : the maximum flow problem and the minmum cost flow problem.

The second problem presented in this work is the transportation problem. Definitions and methods of resolution based on graph theory and linear programming are provided.

Introduction

>Toute entreprise qu'elle que soit sa taille, son domaine d'activité est amenée à faire face à des problèmes de gestion au quotidien.

Parmi ces problèmes, on cite les problèmes de flot, les problèmes de transport.

On parle du problème de flot maximum quand il s'agit de maximiser la quantité transportée d'une ou de plusieurs sources vers une ou plusieurs destinations. Il est modélisé par un graphe value appelé réseau de transport, que nous définissons. Les arcs représentent les possibilités de transport entre deux sites et sont values par les capacitéscorrespondantes. Le problème du flot maximal est résolu par un algorithme efficace fondamental du à FORD et FULKERSON, que nous présentons et justifions dans ce mémoire.

On entendra par problème de transport tout problème d'optimisation du transfert entre points-origine ou fournisseurs et points – destination ou clients.

Tous ces problèmes, bien qu'appartenant à des domaines de la gestion

très différents peuvent être résolus à l'aide des techniques de la programmation linéaire (simplexe), mais la structure très spécifiques des ces problèmes permettent de recourir à des techniques particulières beaucoup plus légères.

C'est pour cette raison que le but de notre travail est de présenter des méthodes faciles de formulation et de résolution de ce genre de problèmes. Notre travail est divisé en quatre chapitres, où nous allons aborder dans le chapitre 1 les problèmes de flot maximum et celui de coût minimum. Ensuite nous présentons le problèmes de transport ainsi que sa résolution par les flots. Une méthode de résolution du problème de transport par la programmation linéaire est présentée dans le chapitre 3. Dans le dernier chapitre nous proposons un programme codé en C++ pour la résolution de ce problème, ainsi que des implémentations pour quelques exemples.

CHAPITRE I

PROBLEMES DES FLOTS

Dans ce chapitre, nous rappelons d'abord des notions de base de la théorie des graphes nécessaires à une bonne compréhension de la suite du mémoire. En suite nous présontons le problème du flot maximum et l'algorithme de Ford et Fulkerson pour sa résolution. Dans la dèrnière section le problème du flot de coût minimum est etudié. Des formulations en programmation linéaire pour ces deux problèmes des flots sont présentées.

1.1 Notions de base de la théorie des graphe

Définition 1.1.1 *Un graphe orienté $G = (X, U)$ est défini par un ensemble fini $X = \{x_1, x_2, ..., x_n\}$ dont les éléments sont appelés sommets, et par un ensemble fini $U = \{u_1, u_2, ..., u_m\}$ dont les éléments sont appelés arcs. Un arc u de l'ensemble U est défini par une paire ordonnée de sommets. Lorsque $u = (a, b)$ on dit que a est l'extrémité initiale et b l'extrémité finale de u.et on écrit $I(u) = a$, et $T(u) = b$.*

Exemple 1.1.1 *Considérons le graphe orienté suivant:*

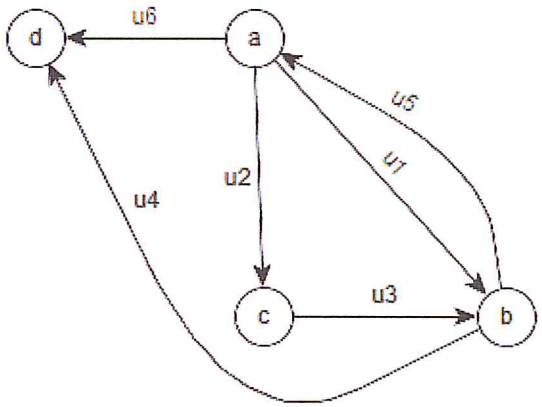


Figure 1.0: Un graphe oriente

$I(u_6) = a, T(u_6) = d.$

1.2 Probléme du flot maximun

Définition 1.2.1 *soit $G = (X, U)$ un graphe où U l'ensemble des arcs ($|U| = m$) et X l'ensemble des sommets. Une application f de U dans $\mathbb{R}$ est dite un flot sur le graphe G si:*

$$\forall x \in X, \sum_{u/I(u)=x} f(u) = \sum_{u/T(u)=x} f(u)$$

Exemple 1.2.1 *Considèrons le réseau décrit par la Figure 1.1*

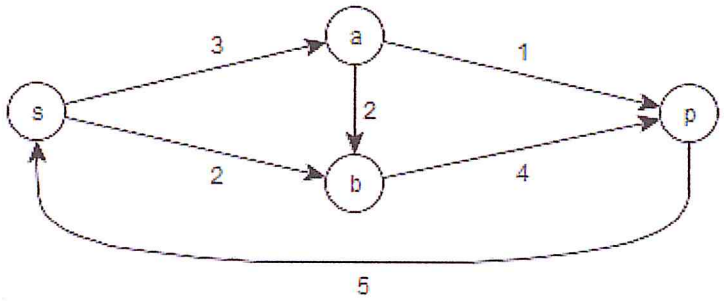


Figure 1.1 Exemple de flot

1.2.1 Position du problème

Définition 1.2.2 *Soit un graphe $G = (X,U)$ dans lequel un arc spécial dit «de retour» $u_r = (ps)$ a été distigué. Le sommet s est appelé la source et le sommet p est dit sommet puits. On suppose donnée une application:*

$$c : U \to \mathbb{R}^+ \cup \{+\infty\}$$

qui à chaque arc u fait correspondre sa «capacité» $c(u)$. Le problème du flot maximum de s à p dans le réseau $R = (X, U, c)$ consiste à trouver une application $f : U \to \mathbb{R}^+$ tel que:

(i) f soit un flot sur G.

(ii) $\forall u \in U \quad 0 \leq f(u) \leq c(u)$.

(iii) $f(u_r)$ soit maximum.

Remarque 1.2.1 *L'application f verifiant (i) et (ii) s'appelle un flot réalisable.*

En d'autres termes, il s'agit de trouver un flot non négatif sur les arcs du réseau R tel que la valeur de $f(u_r)$ (appelée dans la suite «valeur de flot f») soit la plus grande possible sous réserve que le flot sur chaque arc soit inférieur ou égal à la capacité de cet arc.

1.2.2 Application

On considère un réseau de conduits (tuyaux, câbles électriques, etc ...) ayant chacun une «capacité» (limite supérieure de la quantité de flux qui peut passer dans le conduit par unité de temps) et on cherche à envoyer la plus grande quantité de flux Q du sommet s au sommet p sachant que le flot est consevatif (la somme des flux entrant en un sommet intermédiaire du réseau est égale à la somme des flux sortant de ce sommet).

1.2.3 Modélisation

On associe un graphe au réseau de conduit $\breve{G} = (X, \breve{U})$ où X correspond à l'ensemble des noeuds des conduits et l'ensemble des arcs $\breve{U}$ correspond aux conduits. A chaque arc u de $\breve{G}$ on associe sa capcité $c(u)$ qui est la capacité du conduit correspondant (si un conduit peut transporter les mêmes quantités dans les deux sens, on associe la même valeur de la capacité aux deux arcs en sens opposé représentant ce conduit). Les équations de conservations du flux aux sommets s'écrivent:

$$\Sigma_{\{u\in\breve{U}|I(u)=x\}} f(u) - \Sigma_{\{u\in\breve{U}|T(u)=x\}} f(u) = \begin{cases} Q & \text{si} \quad x = s \\ 0 & \text{si} \quad x \neq s, p \\ -Q & \text{si} \quad x = p \end{cases} \tag{1}$$

Par adjonction de l'arc de retour $u_r = (ps)$ de capacité infinie et en posant $U = \breve{U} \cup \{u_r\}$, et $f(u_r) = Q$ les équations (1.2) deviennent:

$$\Sigma_{\{u\in U|I(u)=x\}} f(u) - \Sigma_{\{u\in U|T(u)=x\}} f(u) = 0 \qquad \forall \ x \in X \tag{2}$$

et le problème revient donc à chercher un flot réalisable maximum de s à p dans le réseau $R = (X, U, c)$.

1.2.4 Une généralisation du problème de flot maximum de s à p.

On suppose qu'il existe un ensemble S de sommets sources et un ensemble P de sommets puits. Ce cas se réduit au précédent en ajoutant au réseau initial un sommet s dit «supersource» et un sommet p dit «superpuits» ainsi que les arcs (sx) pour tout $x \in S$ et (xp) pour tout $x \in P$. Les arcs ajoutés ont des capacités infinies. Ainsi dans le réseau de la Figure 1.2

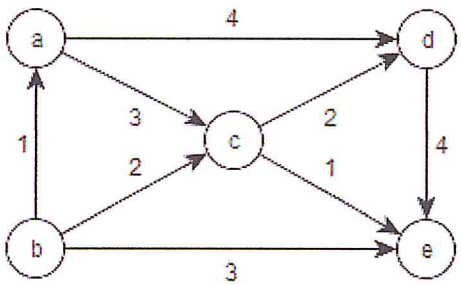


Figure 1.2: Sources et puits multiples

si a et b sont des sources ,e et d des puits. On se ramène à la formulation générale sur le réseau de la Figure 1.3.

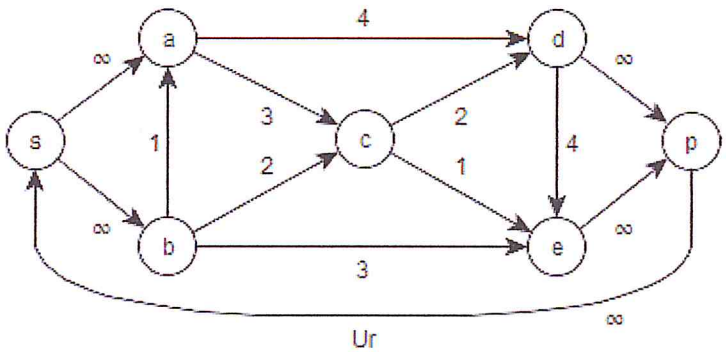
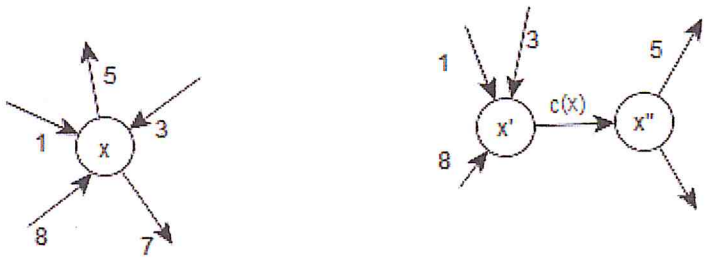


Figure 1.3: Passage vers: une source et puits uniques

De même, on se convaincra facilement qu'on peut se ramener au modèle de la définition(1.2.1) s'il y a des capacité définies au nœuds du réseau (stations de pompage, par exemple). Dans ce cas on dédoublera chacun des sommets x sur les quels il existe des contraintes de capacité en les sommets x′et x″. Tous les arcs qui sont dans le réseau initial et qui ont x comme sommet terminal (respectivement initial) aurant dans le réseau modifié x′comme sommet terminal (respectivement x″comme sommet initial) et on rajoute un arc (x′x″) auquel est affecté la capacité du sommet x. Cette transformation est schématisée sur la Figure 1.4.



Figur 1.4: Capacite c(x) en x.

1.2.5 Formulation du problème de flot maximum en programmation linéaire

Le problème du flot maximum de s à p dans le réseau R peut se formuler comme un programme linéaire en variables bornées

$$(P.L) \begin{cases} Max(z) = f(u_r) \\ f \leq c \\ Af = 0 \qquad f \geq 0 \end{cases} \tag{3}$$

où A est la $n \times m$-matrice d'incidence aux arcs du graphe $G = (X, U)$ ($n = |X|, m = |U|$); la $i^{éme}$ contrainte $A_i f = 0$ exprime la conservation du flot au sommet x_i.

1.2.6 Algorithme de Ford et Fulkerson pour la recherche d'un flot maximum:

L'algorithme de Ford et Fulkerson (Ref [5,6]) est basé sur une procédure de marquage.

On part d'un flot initial réalisable . Puis on marque le sommet s. Deux types de marquage sont alors possibles:

Marquage direct:

S'il existe un arc $u = (x, y)$ tel que $c(u) > f(u)$, avec x marqué et y non marqué, on marque le sommet y.

Figure 1.5: Marquage direct

Marquage indirect:

S'il existe un arc $u = (x, y)$ tel que $f(u) > 0$, avec x marqué et y non marqué, on marque le sommet y.

Figure 1.6: Marquage indirect

1.2.7 Principe de l'algorithme de Ford et Fulkerson

L'algorithme de Ford et Fulkerson permet de trouver un flot maximum de s à p dans le réseau $R = (X, U, c)$, en partant d'un flot réalisable f.

L'idée de cet algorithme est chercher une chaine élémentaire C joignant s et p qui avec l'arc u_r définit un cycle $\gamma = C \cup \{u_r\}$.

Soient:

C^+ est l'ensemble des arcs de C orientés dans le sens de s à p.

C^-est l'ensemble des arcs de C orientés dans le sens contraire.

On peut alors améliorer le flot:

par

$$f(u) := \begin{cases} f(u) + \varepsilon & \text{si } u \in C^+ \cup \{u_r\} \\ f(u) - \varepsilon & \text{si } u \in C^- \\ f(u) & \text{si } u \notin C \end{cases}$$

où

$$\begin{aligned} \varepsilon_1 &= \min_{u \in C^+}\{c(u) - f(u)\} \\ \varepsilon_2 &= \min_{u \in C^-}\{f(u)\} \\ \varepsilon &= \min(\varepsilon_1, \varepsilon_2) \end{aligned}$$

Algorithme de Ford et Fulkerson

(0) On part d'un flot de départ réalisable f , par exemple un flot nul.

(1) On marque le sommet s .

(2) On essaye de marquer le sommet p avec la procédure de marquage.

Le sommet p est-il marqué?

(2.a) si oui : amélioration du flot f;

Soit C la chaine le long laquelle le marquage de p à été éffectué.

C^+ est l'ensemble des arcs de C orientés dans le sens de s à p.

C^-est l'ensemble des arcs de C orientés dans le sens contraire.

(il faut écrire une phrase ici)

$$\begin{aligned} \varepsilon_1 &= \min_{u \in C^+}\{c(u) - f(u)\} \\ \varepsilon_2 &= \min_{u \in C^-}\{f(u)\} \\ \varepsilon &= \min(\varepsilon_1, \varepsilon_2) \end{aligned}$$

Le nouveau flot est donné par:

$$f(u) := \begin{cases} f(u) + \varepsilon & \text{si } u \in C^+ \cup \{u_r\} \\ f(u) - \varepsilon & \text{si } u \in C^- \\ f(u) & \text{si } u \notin C \end{cases}$$

-Effacer toutes les marques sauf la marque de s.

-Aller à (1).

-(2) sinon: p n'a pas pu etre marqué.

-Terminé le flot courant est maximum.

Fin Algorithme

Remarque 1.2.2 *Dans le cas où* $C^+ = \emptyset$ *on prend* $\varepsilon = \varepsilon_2$. *Si* $C^- = \emptyset$ *on prend* $\varepsilon = \varepsilon_1$.

1.2.8 Application:

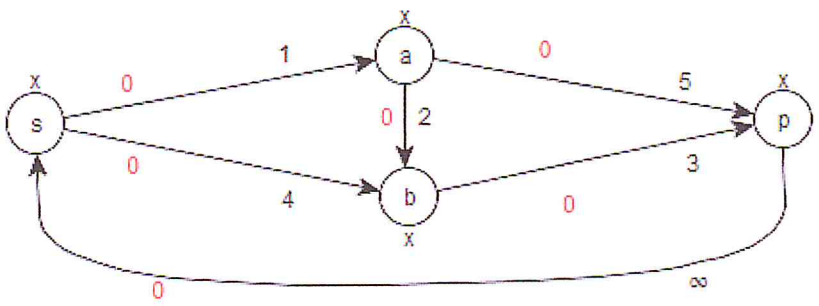


Figure 1.7: exemple de flot

-On part du flot nul comme il est indiqué sur la Figure 1.7.

-1ère itération: on marque le sommet s.

Par le marquage direct on marque a (car $1 > 0$), puis on marque le sommet b (car $2 > 0$)

On marque le sommet p (car $3 > 0$)

$$C = \{(s,a),(a,b),(b,p)\} \tag{4}$$

$$C^+ = C \ , \ C^- = \emptyset \tag{5}$$

$$\epsilon = \epsilon_1 = \min_{u \in C^+} \{c(u) - f(u)\} = \min\{1-0, 2-0, 3-0\} = 1.$$

Le nouveau flot: est représenter par la Figure 1.8:

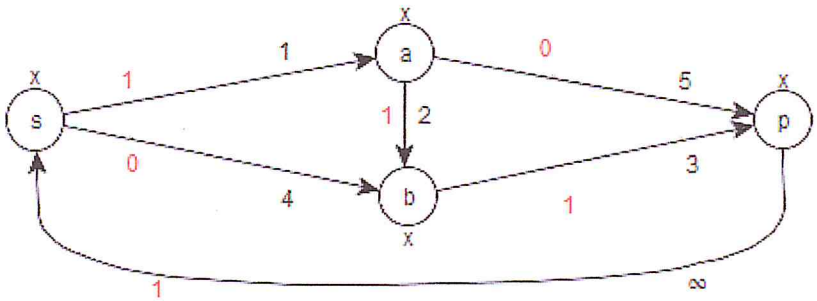


Figure 1.8 1ere-iteration

-Itération 2:

On marque le sommet s.

On ne peut pas marquer le sommet a par le marquage directe. Marquons le sommet b (car $4 > 0$)

On marque a par marquage indirecte car $f(a,b) = 1 > 0$,puis on marque p (car $5 > 0$).

$$
\begin{aligned}
C &= \{(s,b),(a,b),(a,p)\} \\
C^+ &= \{(s,b),(a,p)\} \ , \ C^- = \{(a,b)\} \\
\epsilon_1 &= \min\{4-0, 5-0\} = 4, \epsilon_2 = 2-1 = 1. \\
\epsilon &= 1.
\end{aligned}
$$

Le nouveau flot est représenté sur la Figure 1.9

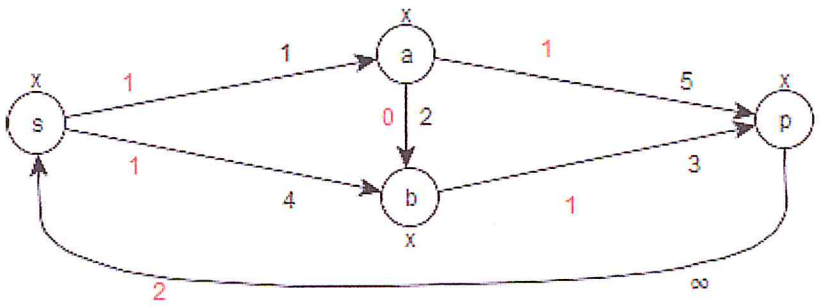


Figure 1.9: Iteration 2

-Itération 3:

On marque le sommet s.

L'arc (s,a) est saturé.

Marquage de a à partir de s n'est pas possible.

On marque le sommet b (car $4 > 1$).

On ne peut marquer a par marquage indirecte car $f(a,b) = 0$.

On marque le sommet p .

$$
\begin{aligned}
C &= \{(s,b),(b,p)\} \\
C^+ &= C \\
\epsilon &= \epsilon_1 = \min(4-1, 3-1) = 2.
\end{aligned}
$$

Le nouveau flot est sur la Figure 1.10:

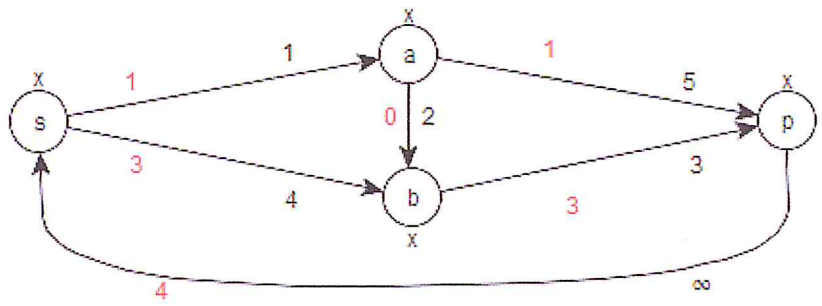


Figure 1.10: Iteration 3

-Itération 4:

On marque s .

a ne peut pas être marqué à partir de s.

On marque b par le marquage direct.

On ne peut pas marquer a à partir de b par le marquage indirect, et on ne peut pas marquer p à partir de b par le marquage direct. Donc on ne peut pas mérquer p.

Le flot actuel est maximum qui est de valeur $f(u_r) = 4$.

1.2.9 Théorème de la coupe minimum

Définition 1.2.3 *Etant donné un graphe $G = (X, U)$ et $Y \subset X$.On considère les ensembles d'arcs:*

$$\Omega^+(Y) = \{u \in U \mid I(u) \in Y, T(u) \notin Y\}$$
$$\Omega^-(Y) = \{u \in U \mid I(u) \notin Y, T(u) \in Y\}$$

Le couple d'ensemble d'arc $(\Omega^+(Y), \Omega^-(Y))$ est appelé le «cocycle relatif à Y dans G.

Définition 1.2.4 *Dans le réseau $R = (X, U, c)$ où l'arc de retour $u_r = (ps)$ a été distingué un ensemble d'arcs C est appelé "coupe séparant p de s" si on peut trouver $Y \subset X$ avec $s \in Y, p \notin Y$ tel que:*

$$C = \Omega^+(Y)$$

Par exemple dans le graphe de la Figure (1.11)

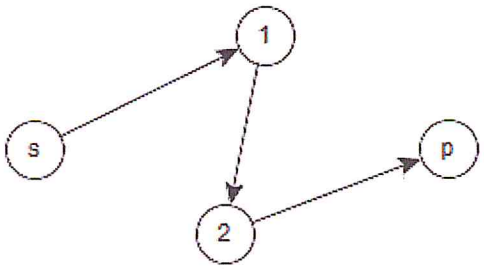


Figure 1.11: Exemple de coupe

$$C = \{(s, 1), (2, p)\}$$

On a $Y = \{s, 2\}$et $C = \Omega^+(Y)$ alors l'ensemble C est une coupe séparant p de s.

Si on prend

$C_1 = \{(2, p)\}$. Montrons que C_1 n'est pas une coupe séparant p de s?

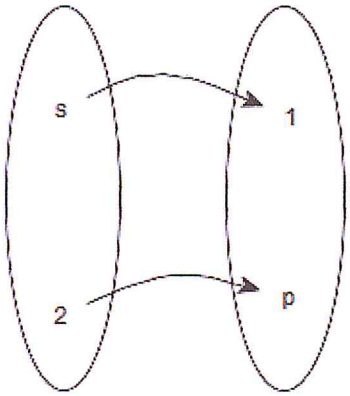
Supposons qu'il existe $Y \subset \{s, 1, 2, p\}$ tel que:

$\cdot s \in Y, p \notin Y.$

$\cdot C_1 = \Omega^+(Y).$

Ceci implique que $s \in Y$ et $2 \in Y$.

Il existe donc 2 arcs qui sortent de Y.



Ceci implique que $\Omega^+(Y)$ contient au moins 2 arcs $(s,1)$ et $(2,p)$

Donc C_1 ne peut pas être une coupe.

Capacité d'une coupe

Définition 1.2.5 *Dans un réseau $G=(X,U,c)$,$u_r=(ps)$ l'arc de retour.*

Soit C une coupe séparant p de s.

La capacité de C est

$$c(C)=\sum_{u\in C} c(u)$$

Théorème 1.2.1 *(de la coupe minimum)(Ref [5,6])*

La valeur d'un flot réalisable maximum sur le réseau $R=(X,U,c)$ est égale à la capacité d'une coupe de capacité minimum séparant p de s dans R.

1.3 Problème du flot de coût minimum

Le problème du flot de coût minimum est un modèle trés général de la recherche opérationnelle.

Définition 1.3.1 *Etant donné un réseau $R=(X,U,a,b,c)$ où :*

$$\begin{cases} a : U \to \mathbb{R} \\ b : U \to \mathbb{R} \cup \{-\infty\} \\ c : U \to \mathbb{R} \cup \{+\infty\} \end{cases}$$

avec :

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

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

$$(i)\ \ f \text{ soit un flot sur } G = (X, U)$$
$$(ii)\ b(u) \leq f(u) \leq c(u) \quad \forall\, u \in U$$
$$(iii)\ \sum_{u \in U} a(u) f(u) \quad \text{soit minimum}$$

Formulation du problème de flot de coût minimum en programmation linéaire

Le problème du flot de coût minmum sur $R = (X, U, a, b, c)$ peut être résolu par la programmaion linéaire sa formulation est donné par le programme linéaire suivant:

$$(F) \begin{cases} Ef = 0 \\ b \leq f \leq c \\ af = z(Min) \end{cases}$$

Où E est la matrice d'incidence aux arcs du graphe $G = (X, U)$.

Remarque 1.3.1 *Le problème du flot maximum de s à p sur* $R' = (X, U \cup \{u_r\}, c)$ *avec* $u_r = (ps)$ *se ramène à celui de la recherche d'un flot de coût minimum sur* $R' = (X, U \cup \{u_r\}, a, b, c)$ *avec:*

$$\begin{cases} a(u) = 0 & u \in U \\ a(u_r) = -1 & \\ b(u) = 0 & u \in U \cup \{u_r\} \end{cases}$$

$x_{ij} \geq 0$ pour tout i, j.

(3) Le problème de transport s'écrit alors:

$$(T)\begin{cases} \text{Minimiser } z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \sum_{j=1}^{n} x_{ij} \leq a_i \qquad i = 1, 2, ..., m \\ \sum_{i=1}^{m} x_{ij} \geq b_j \qquad j = 1, 2, ..., n \\ x_{ij} \geq 0 \quad i = 1, ..., m \text{ et } j = 1, ..., n \end{cases}.$$

2.2 Résolution du problème de transport par les flots:

Le problème de transport (T) se modélise en tant que problème de flot du coût minimum sur le réseau $R = (X, U, a, b, c)$ avec:

$$.X = \{y_1, y_2, ..., y_m\} \cup \{z_1, z_2, ..., z_n\} \cup \{s, p\}$$

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

$$
\begin{aligned}
.u_r &= (ps) \\
.U_1 &= \{(y_i z_j) | i = 1, 2, ..., m; j = 1, 2, ..., n\} \\
.U_2 &= \{(s y_i) | i = 1, 2, ..., m\} \\
U_3 &= \{(z_j p) | j = 1, 2, ..., n\} \\
a(u) &= \begin{cases} a_{ij} & \text{si } u = (y_i z_j) \in U_1 \\ 0 & \text{dans les autres cas} \end{cases} \\
b(u) &= \begin{cases} b_j & \text{si } u = (z_j p) \in U_3 \\ 0 & \text{dans les autres cas} \end{cases} \\
c(u) &= \begin{cases} c_i & \text{si } u = (s y_i) \in U_2 \\ 0 & \text{dans les autres cas} \end{cases}
\end{aligned}
$$

Considérons le problème de transport donné par le tableau suivant:

	z_1	z_2	z_3	z_4	Disponibilités
y_1	8	5	7	25	25
y_2	8	2	7	6	25
y_3	9	3	4	8	50
Demandes	15	20	30	35	

Tableau du probleme de transport

Pour illustrer la réduction précédente on a représenté sur la Figure 2.1 le réseau correspondant au problème de transport défini par le Tableau ci-dessus

On a porté en regard de chaque arc u $(a(u), b(u), c(u))$

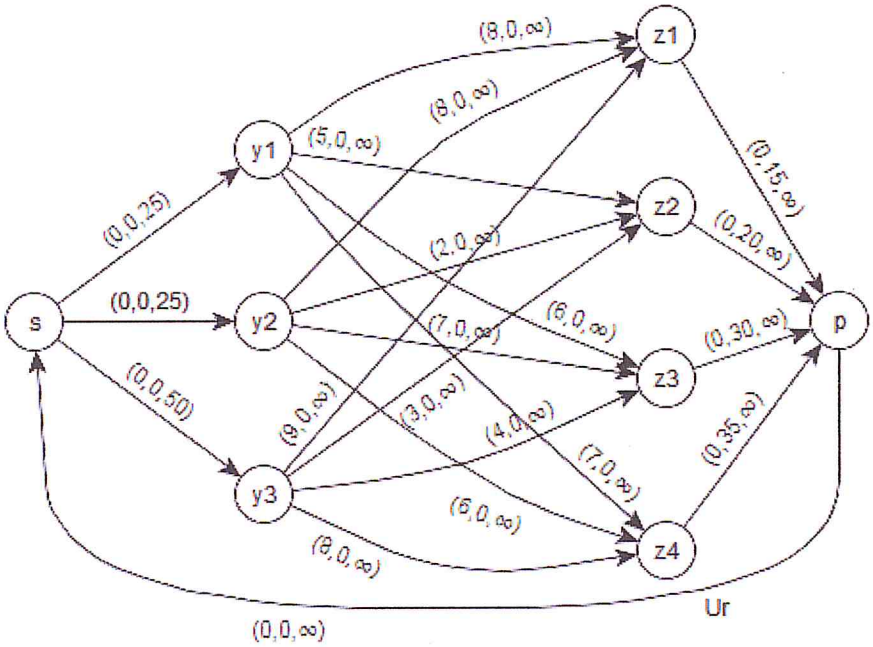


Figure 2.1 graphe correpond au probleme de transport

2.3 *Problème de transport généralisé:*

Dans le problème de Transport le réseau sous jacent est biparti et les capacités des arcs sont infinies. Si on lève ces deux restrictions on obtient un modèle trés général et trés utile dans la pratique, appelé "Problème de transport généralisé". Dans ce cas on considère que S est un ensemble de sources et T est un ensemble de puits. Dans la littérature anglo-saxone on l'appelle "Transhipment problem ".(Ref [1,8])

Formulons ce problème par la théorie des graphes. On considère un réseau $R = (X, U, S, T, a, b, c)$ avec :

$$
\begin{aligned}
.S,T &\subset X \ : \ S \cap T = \emptyset \\
.a &: U \rightarrow \mathbb{R}^+ \\
.b &: T \rightarrow \mathbb{R}^+ \\
.c &: S \rightarrow \mathbb{R}^+ \cup \{u_r\} \\
.d &: U \rightarrow \mathbb{R}^+ \cup \{u_r\}
\end{aligned}
$$

Le problème de transport généralisé noté (T_g) consiste à trouver un vecteur $f \in \mathbb{R}^m$ (où $m = |U|$.) tel que:

$$(i) \quad \Sigma_{\{u|I(u)=x\}} f(u) - \Sigma_{\{u|T(u)=x\}} f(u) = \begin{cases} \leq c(x) & \text{si} \quad x \in S \\ 0 & \text{si} \quad x \in X - \{S \cup T\} \\ \leq -b(x) & \text{si} \quad x \in T \end{cases}$$

$(ii) \quad 0 \leq f(u) \leq d(u) \qquad u \in U$

$(iii) \quad \sum_{u \in U} a(u) f(u)$ soit minimum.

En pratique les sommets x de S sont des points où certain bien est disponible en quantité $c(x)$, les sommets x de T sont des points de consommation ($b(x)$ est la demmande correspondante), le flux devant être conservatif pour les sommets intèrmédiaires. Il s'agit de trouver une manière de satisfaire la demande aux points de T sans prendre plus de biens qu'il y'en a de disponible aux points de S, sans violer les contraintes de capacité sur les arcs et ceci au moindre coût.

2.3.1 Modélisation du problème de transport généralisé (T_g) par les flots

Le problème de tranport généralisé (T_g) peut être vu comme un problème de flot de coût minimum sur le réseau $R' = (X', U', a', b', c')$ avec:

$$
\begin{aligned}
.X' &= X \cup \{s, p\} \\
.U' &= U_1 \cup U_2 \cup U_3 \cup \{u_r\} \\
.u_r &= (ps) \\
.U_1 &= U \\
.U_2 &= \{(sx) | x \in S\} \\
U_3 &= \{(xp) | x \in T\} \\
.a'(u) &= \begin{cases} a(u) & \text{si} \quad u \in U_1 \\ 0 & \text{si } u \in U' - U_1 \end{cases} \\
.b'(u) &= \begin{cases} b(x) & \text{si} \quad u = (xp) \in U_3 \\ 0 & \text{si} \quad u \in U' - U_3 \end{cases} \\
.c'(u) &= \begin{cases} d(u) & \text{si} \quad u \in U_1 \\ c(x) & \text{si} \quad u = (sx) \in U_2 \\ \infty & \text{si} \quad u \in U' - (U_1 \cup U_2) \end{cases}
\end{aligned}
$$

2.4 Problème de transport équilibré

Dans le cas où

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

Les contraintes d'inégalités du problème de transport (T) deviennent des égalités.

Le problème est alors équivalent au problème suivant:

$$
(T_e) \begin{cases} \text{Minimiser } z = \sum_{j=1}^{n} \sum_{i=1}^{m} c_{ij} x_{ij} \\ \sum_{j=1}^{n} x_{ij} = a_i \quad i = 1, ..., m \\ \sum_{i=1}^{m} x_{ij} = b_j \quad j = 1, ..., n \\ x_{ij} \geq 0 \quad i = 1, ..., m \text{ et } j = 1, ..., n \end{cases}
$$

Le problème (T_e) s'appelle le problème de transport équilibré. Dans la suite on s'interessera à la résolution du problème équilibré (T_e).

Dans le cas où le problème de transport n'est pas équilibré, on ajoutera une déstination fictive de demande égale à:

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

Les coûts de transport d'une unité du produit depuis les origines vers la déstination fictive sont nuls. Et on est donc ramené à la résolution du problème de transport équilibré.

CHAPITRE III

RÉSOLUTION DU PROBLÈME DE TRANSPORT ÉQUILIBRÉ PAR LA PROGRAMMATION LINÉAIRE

3.1 Exemple illustratif

Une entrprise de fabrication de conserves expédie des caisses des usines vers des dépôts. Elle veut que le programme d'expédition des caisses minimise le coût de transport total des usines vers les dépôts. Pour simplifier, nous supposerons qu'il y a deux usines 1 et 2 et trois dépôts A, B et C. Les disponibilités en caisses dans les usines a_i et les besoins des dépôts b_j sont représentées dans le Tableau ci-dessous.

a_i	b_j
350 caisses à l'usine 1	300 caisses au depôt A
550 caisses à l'usine 2	300 caisses au depôt B
	300 caisses au depôt C

Tableau: disponibilies des usines

Le coût d'expédition, par caisse, entre chaque usine et chaque dépôt est consigné dans le Tableau ci-dessous.

Origines	Déstinations		
	A	B	C
1	25	17	16
2	24	18	14

Tableau: coûts d'expédition (en dinars par caisse)

Le problème consiste à déterminer le nombre de caisses que chacune des usines doit expédier vers chacun des dépôts de façon à ce que le coût de transport total soit minimum. La fonction objectif à minimiser est :

$$z = 25x_{11} + 17x_{12} + 16x_{13} + 24x_{21} + 18x_{22} + 14x_{23}$$

sous les contraintes

$$\begin{aligned} x_{11} + x_{12} + x_{13} &= 350 \\ x_{21} + x_{22} + x_{23} &= 550 \qquad \text{(dues aux origines)} \end{aligned}$$

et

$$\begin{aligned} x_{11} + x_{21} &= 300 \\ x_{12} + x_{22} &= 300 \\ x_{13} + x_{12} &= 300 \qquad \text{(dues aux destinations)} \end{aligned}$$

3.2 Résolution du problème de transport équilibré:

Nous rappelons que le problème de transport équilibré est le problème de transport suivant :

$$(T_e) \begin{cases} \text{Minimiser } z = \sum_{j=1}^{n} \sum_{i=1}^{m} c_{ij} x_{ij} \\ \sum_{j=1}^{n} x_{ij} = a_i \quad i = 1, ..., m \\ \sum_{i=1}^{m} x_{ij} = b_j \quad j = 1, ..., n \\ x_{ij} \geq 0 \quad i = 1, ..., m \text{ et } j = 1, ..., n \end{cases}$$

Pour la résolution on suppose aussi que $a_i > 0, b_j > 0$. Notons que le nombre de contraintes est égal à $m+n$ et que le nombre de variables est $m.n$.

La bonne structure de ce problème nous permettent de calculer une solution de base réalisable de ce problème par des techniques très simples sans passer par la résolution de la phase I de la méthode du simplexe. Ces techniques nous permettent aussi de calculer la solution du problème dual et les coûts relatifs correspondant à n'importe quelle solution de base, ainsi que la nouvelle solution de base.

Notons encore que le problème de transport (T_e) possède toujours une solution réalisable, le lecteur pourra vérifier qu'elle est donnée par :

$$x_{ij} = \frac{a_i b_j}{2} \quad \forall \quad i \text{ et } j$$

(T_e) peut être écrit:

$$(T_e) \begin{cases} \min \ z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \sum_{j=1}^{n} -x_{ij} = -a_i \quad i = 1...m \quad (1) \\ \sum_{i=1}^{m} x_{ij} = b_j \quad j = 1...n \quad (2) \\ x_{ij} \geq 0 \quad \forall \, i, j \end{cases}$$

3.2.1 Redondance dans les contraintes:

Dans toute la suite on considérons le problème (T_e) .

Théorème 3.2.1 *(Ref [2]) Parmi les containtes de (T_e) , il existe une contrainte redondante.*

Preuve. Soit E la matrice des contraintes

$E : (m+n) * (m * n)$

Le système de (T_e)

$$(T_e) \begin{cases} \min z = CX \\ EX = B \\ X \geq 0 \end{cases} \tag{6}$$

$$X^t = (x_{11}, x_{12}, ..., x_{1n}, x_{21}, x_{22}...x_{2n}, ..., x_{m1}x_{m2}...x_{mn}) \tag{7}$$

$$B = \begin{pmatrix} a_1 \\ \vdots \\ a_m \\ b_1 \\ \vdots \\ b_n \end{pmatrix} \tag{8}$$

Le vecteur colonne de la matrice E associé à une variable x_{ij} contient seulement 2 éléments non nuls $(+1)$ et (-1), les autres sont des zéros.

Le -1 correspond à la ième ligne et $+1$ correspond à la $(m+j)$ ième ligne.

Explication: les contraintes contenant x_{ij} sont:

-La ième contrainte

$$-x_{i1} - x_{i2} - ..... - x_{ij} - .... - x_{in} = -a_i \tag{9}$$

-La (m+j) ième contrainte

$$x_{1j} + x_{2j} + ... + x_{ij} + ...0... + x_{nj} = b_j \tag{10}$$

Puisque chaque variable est repéteé 2 fois avec des coéfficients $+1$ et -1, en sommant toutes les variables, on obtient $0 = 0$.

Exemple illustratif: $m = 2, n = 3$

Les contraintes sont:

$$\begin{array}{cccccccc}
-x_{11} & -x_{12} & -x_{13} & & & & & = -a_1 \\
 & & & -x_{21} & -x_{22} & -x_{23} & & = -a_2 \\
x_{11} & & & +x_{21} & & & & = b_1 \\
 & x_{12} & & & +x_{22} & & & = b_2 \\
 & & x_{13} & & & +x_{23} & & = b_3
\end{array}$$

Ceci implique qu'il ya redendance.

Chaque contrainte de (T_e) est redondante. ■

Soit $\bar{E}$ la matrice obtenue de E par suppression de la dernière ligne.

$\bar{E}$ est une matrice $((m+n-1)*(m*n))$

Les contraintes correspondantes à $\bar{E}$ deviennent:

$$\begin{cases} \sum_{j=1}^{n} -x_{ij} = -a_i \quad i = 1...m \quad (1)' \\ \sum_{i=1}^{m} x_{ij} = b_j \quad j = 1...n-1 \quad (2)' \end{cases}$$

Proposition 3.2.1 *Le rang de $\bar{E}$ est égal à $m+n-1$.*

Corollaire 3.2.1 *Une solution de base du problème (T_e) contient $m+n-1$ variables de base,et $m*n-(m+n-1)$ variables hors base.*

3.2.2 Tableau de transport:

On présente les variables par le Tableau suivant:

La variable x_{ij} correspond à la cellule (i,j) et vice versa

Ce Tableau sra noté par x

$$x = \begin{array}{c|ccc} & j=1 & \dots & j=n \\ i=1 & x_{11} & \dots & x_{1n} \\ \vdots & \vdots & & \vdots \\ i=m & x_{m1} & \dots & x_{mn} \end{array} = (x_{ij})$$

Tableau de transport (11)

Si x satisfait toutes les contraintes, donc x est une solution réalisable du problème de transport (T_e) .

Définition 3.2.1 *Un sous ensemble S de cellules du tableau de transport est dit linéairement indépondant si les vecteurs colonnes de la matrice des contraintes du problème (T_e) correspondant aux variables associeés aux cellules de S sont linéairemnt indépendants.*

Définition 3.2.2 *Un sous ensemble de $m+n-1$ cellules du tableau de transport (à dimension $n*m$) est dit de base s'il est linéairement indépendant.*

Définition 3.2.3 *d'une $\theta-$ boucle*

Une $\theta-$boucle est un sous ensemble S de cellules du tableau de transport vérifiant les propriétés suivantes:

(i) S est non vide

(ii) S contient soit 0 soit 2 cellules de chaque ligne et colonne du tableau.

(iii)Aucun sous ensemble propre de S vérifie les conditions (i) et (ii).

Exemple 3.2.1

y				y
	y	y		
y		y		
	y			y

Une θ – boucle Les cellules qui ne sont pas dans la θ – boucle correspondent aux cellules vides .

Exemple 3.2.2

		y	y		
		y	y		
				y	y
				y	y

Le sous ensemble des cellules marquées avec y dans le tableau ci-dessus vérifie les propriétés (i) et (ii) mais non pas (iii),

donc ce sous ensemple n'est pas une θ – boucle.

Remarque 3.2.1 *Si S est une θ – boucle alors les éléments $+\theta$ et $-\theta$ pouvent être mis alternativement dans les cellules de S de sorte que si une ligne ou une colonne du tableau contient une cellule de S avec élément $+\theta$,alors elle contient aussi une autre cellule de S avec élément $-\theta$.*

Pour illustrer cette remarque on reprend l'exemple 3.2.1

La θ – *boucle* sera représentée de la façon suivante:

Exemple 3.2.3

$+\theta$				$-\theta$
	$+\theta$	$-\theta$		
$-\theta$		$+\theta$		
	$-\theta$			$+\theta$

Théorème 3.2.2 (Dépendance linéaire d'une θ – *boucle*) *(Ref [2]) Un sous ensemble de cellules du tableau de transport formant une θ – boucle est lineairement dépendant.*

Théorème 3.2.3 *(Ref[2]) Soit S un sous ensemble de cellules du tableau si S ne contient pas une θ – boucle alors S est lineairement indépendants.*

Théorème 3.2.4 *(Ref[2]) Soit S une base de $m+n-1$ cellules du tableau de transport et (p,q) une cellule hors base .Alors $S \cup \{p,q\}$ contient exactement une unique θ – boucle.*

3.3 Obtention d'une solution de base réalisable de départ

3.3.1 Méthode de la matrice minimale (Minimum Entry Method)

Étape 1 Trouver la cellule $(p;q)$, telle que c_{pq} est le plus petit coût de tout le tableau.

Étape 2 Envoyer le maximum d'unités pour la cellule $(p;q)$. Ainsi x_{pq} est initialisé comme étant le $min\{a_p;b_q\}$. Ajuster ensuite a_p et b_q, en tenant compte du montant x_{pq} à expédier. Exprimons cette phrase à l'aide d'inégalités :

$$\begin{aligned} x_{pq} &= \min\{a_p, b_q\} \\ a'_p &= a_p - x_{pq} \\ b'_q &= b_q - x_{pq} \end{aligned}$$

Entourer (ou mettre en évidence d'une autre manière) le coût c_{pq}. À la fin de cette étape, soit a'_p, soit b'_q est nul, soit les deux.

Étape 3

(a) Si $a'_p = 0$ et $b'_q > 0$, cela signifie que l'origine p a été "vidée". Il faut donc éliminer la ligne p du tableau.

(b) Si $b'_q = 0$ et $a'_p > 0$, cela signifie que la destination q est entièrement satisfaite et qu'il reste des marchandises dans le dépôt p. Il faut donc éliminer la colonne q du tableau.

(c) Si $a'_p = 0$ et $b'_q = 0$, nous nous trouvons dans un cas dégénéré.

On élimine alors la ligne p, à moins qu'elle ne soit la seule ligne restante du tableau ; auquel cas il faut éliminer la colonne q.

Étape 4

(a) S'il reste un total de deux ou plusieurs lignes et colonnes non encore éliminées, reprendre l'étape 1.

(b) S'il ne reste qu'une ligne non éliminée, la solution réalisable de base initiale est déterminée par les cellules entourées.

Appliquons cette méthode à notre problème de référence. Le tableau de transport initial est :

	Déstinations			Offre
Origines	1	2	3	
1	25	17	16	350
2	24	18	14	550
Demande	300	300	300	900

Tableau de transport

Pour des raisons pratiques dues à l'utilisation d'un traitement de texte, nous utiliserons les notations suivantes pour la suite de ce travail :

· ♣ à gauche d'une ligne ou en haut d'une colonne pour montrer que cette ligne ou cette colonne a été éliminée ;

· □ pour montrer qu'une cellule $(p; q)$ fait partie de la base ;

· ♭ pour mettre en évidence la nouvelle valeur de a_p ou b_q.

Étape 1 La cellule $(p; q)$ choisie est la cellule $(2; 3)$ dont le coût (14) est le plus petit de l'ensemble du tableau.

Étape 2 Calculons x_{23}, a'_2 et b'_3.

$$\begin{aligned} x_{23} &= \min\{550; 300\} = 300 \\ a'_2 &= a_2 - x_{23} = 550 - 300 = 250 \\ b'_3 &= b_3 - x_{23} = 300 - 300 = 0 \end{aligned}$$

Étape 3 Comme $a_2 = 250$ et $b_3 = 0$, il faut éliminer la colonne 3.

Étape 4 Il reste deux lignes (1 et 2) et deux colonnes (1 et 2), il faut donc choisir un nouvel x_{pq} qui entrera à son tour dans la solution réalisable de base initiale.

Rédigeons d'abord le tableau après cette première itération.

Origines	Déstinations			Offre
	1	2	3 ♣	
1	25	17	16	350
2	24	18	14 □ 300	550 ♭ 250
Demande	300	300	300 ♭ 0	900

Reprenons nos calculs à l'étape (1) avec un tableau réduit aux lignes 1 et 2 et aux colonnes 1 et 2.

Étape 1 Le coût minimum sur ce tableau est 17, soit celui de la cellule $(1;2)$; on va donc initialiser x_{12}.

Étape 2

$$\begin{aligned} x_{21} &= \min\{a_1; b_2\} = \min\{350; 300\} = 300 \\ a'_2 &= a_2 - x_{21} = 250 - 300 = 50 \\ b'_1 &= b_1 - x_{21} = 300 - 300 = 0 \end{aligned}$$

Origines	Déstinations			Offre
	1	2 ♣	3 ♣	
1	25	17 □ 300	16	350 ♭ 50
2	24	18	14 □ 300	550 ♭ 250
Demande	300	300 ♭ 0	300 ♭ 0	900

Étape 3

$a'_1 = 50$ et $b'_2 = 0$, il faut donc éliminer la colonne 2. Comme il reste deux lignes et une colonne, nous n'avons pas terminé, nous cherchons alors le troisième x_{pq} à entrer dans la base.

Étape 1 Le coût minimum sur la ligne 1, la ligne 2 et la colonne 1 est 24,soit le coût de la cellule $(2;1)$.

Étape 2

$$\begin{aligned} x_{21} &= \min\{a_2; b_1\} = \min\{250; 300\} = 250 \\ a'_2 &= a_2 - x_{21} = 250 - 250 = 0 \\ b'_1 &= b_1 - x_{21} = 300 - 250 = 50 \end{aligned}$$

Étape 3 $a'_2 = 0$ et $b'_1 = 50$, on élimine donc la ligne 2

Étape 4 Il reste une ligne (1) et une colonne (1), il faut donc procéder à une quatrième itération.

Origines	Déstinations			Offre
	1	2 ♣	3 ♣	
1	25	17 □ 300	16	350 ♭ 50
2 ♣	24 □ 250	18	14 □ 300	550 ♭ 0
Demande	300 ♭ 50	300 ♭ 0	300 ♭ 0	900

Étape1 Le seul x_{pq} qui peut encore entrer dans la base est x_{11}.

Étape2

$$\begin{aligned} x_{11} &= \min\{a_1; b_1\} = \min\{50; 50\} = 50 \\ a'_1 &= a_1 - x_{11} = 50 - 50 = 0 \\ b'_1 &= b_1 - x_{11} = 50 - 50 = 0 \end{aligned}$$

Étape3 $a'_1 = 0$ et $b'_1 = 0$, la ligne 1 est la dernière des lignes, on élimine donc la colonne 1.

Étape4 l ne reste que la ligne 1, nous avons donc terminé !

Origines	Déstinations			Offre
	1 ♣	2 ♣	3 ♣	
1 ♣	25 □ 50	17 □ 300	16	350 ♭ 0
2 ♣	24 □ 250	18	14 □ 300	550 ♭ 0
Demande	300 ♭ 50	300 ♭ 0	300 ♭ 0	900

À l'aide de cet exemple, nous pouvons vérifier les affirmations formulées précédemment. Tout d'abord, en raison de la redondance, on trouve une solution réalisable de base initiale avec $(m + n - 1)$ variables, ici $2 + 3 - 1 = 4$. Ensuite, à chaque itération, une seule contrainte est satisfaite, ce qui se voit facilement sur le tableau. La dernière contrainte est automatiquement satisfaite à la dernière itération ; donc, si après la quatrième itération l'une des origines ou des destinations avait eu une valeur non-nulle, cela aurait signifié qu'il n'existe pas de solution réalisable.

Nous pouvons maintenant calculer le coût de transport total de ce plan d'expédition:

$$z = \sum_{j=1}^{n} \sum_{i=1}^{m} c_{ij} x_{ij} = 25(50) + 17(300) + 24(250) + 14(300) = 16550 \qquad (12)$$

Il ne s'agit pas encore du coût minimum ; il sera déterminé lors de la recherche de la solution réalisable optimale.

3.3.2 Méthode du coin nord-ouest (Northwest Corner Rule)

De même que la méthode que nous verrons plus loin, cette méthode ne diffère de la précédente que par le critère appliqué à l'étape (1) exposée ici, les étapes (2), (3) et (4) restant les mêmes. Nous présenterons les calculs complets des différentes

itérations, ainsi que les tableaux obtenus. De cette manière, le lecteur pourra juger des différences entre chacune des méthodes.

Étape1 Localiser la cellule $(p;q)$ qui se trouve dans le coin nord-ouest, c'est-à-dire en haut à gauche, de la partie non-éliminée du tableau de transport.

Voyons immédiatement à l'aide de notre exemple quels sont les résultats obtenus. Nous repartons du tableau initial.

	Déstinations			Offre
Origines	1	2	3	
1	25	17	16	350
2	24	18	14	550
Demande	300	300	300	900

Tableau de coûts

Étape 1 Le tableau restant se compose de deux lignes et de trois colonnes ;son coin nord-ouest est la cellule $(1;1)$, x_{11} entre donc dans la base.

Étape 2

$$\begin{aligned} x_{11} &= \min\{a_1;b_1\} = \min\{350;300\} = 300 \\ a'_1 &= a_1 - x_{11} = 350 - 300 = 50 \\ b'_1 &= b_1 - x_{11} = 300 - 300 = 0 \end{aligned}$$

Étape 3 $a'_1 = 50$ et $b'_1 = 0$, on élimine donc la colonne 1.

Étape 4 Il nous reste deux lignes et deux colonnes, nous n'avons donc pas terminé.

Origines	Déstinations			Offre
	1 ♣	2	3	
1	25 □ 300	17	16	350 ♭ 50
2	24	18	14	550
Demande	300 ♭ 0	300	300	900

Tableau aprés la 1ere Iteration

Étape 1 Le tableau restant se compose des lignes 1 et 2 ainsi que des colonnes 2 et 3 ; son coin nord-ouest est donc le cellule $(1;2)$.

Étape 2

$$\begin{aligned} x_{12} &= \min\{a_1; b2\} = \min\{50; 300\} = 50 \\ a'_1 &= a_1 - x_{12} = 50 - 50 = 0 \\ b'_2 &= b_2 - x_{12} = 300 - 50 = 250 \end{aligned}$$

Étape 3 $a'_1 = 0$ et $b'_2 = 250$, on élimine donc la ligne 1.

Étape 4 Il reste une ligne et deux colonnes, il faut donc reprendre la démarche à l'étape (1) afin d'opérer une troisième itération.

Origines	Déstinations			Offre
	1 ♣	2	3	
1 ♣	25 □ 300	17 □ 50	16	350 ♭ 0
2	24	18	14	550
Demande	300 ♭ 0	300 ♭ 250	300	900

Étape 1 Le tableau est maintenant constitué des colonnes 2 et 3 et de la ligne 2, son coin nord-ouest est donc la cellule $(2;2)$.

Étape 2

$$\begin{aligned} x_{22} &= \min\{a_2; b2\} = \min\{550; 250\} = 250 \\ a'_2 &= a_2 - x_{22} = 550 - 250 = 300 \\ b'_2 &= b_2 - x_{22} = 250 - 250 = 0 \end{aligned}$$

Étape 3 $a'_2 = 300$ et $b'_2 = 0$, on élimine donc la colonne 2.

Étape 4 Comme il reste une ligne et une colonne, une quatrième itération est nécessaire.

Origines	Déstinations			Offre
	1 ♣	2 ♣	3	
1 ♣	25 □ 300	17 □ 50	16	350 ♭ 0
2	24	18 □ 250	14	550 ♭ 300
Demande	300 ♭ 0	300 ♭ 0	300	900

Étape 1 Il ne reste que la ligne 2 et la colonne 3, donc le coin nord-ouest est la cellule $(2;3)$.

Étape 2

$$\begin{aligned} x_{23} &= \min\{a_2; b_3\} = \min\{300; 300\} = 300 \\ a'_2 &= a_2 - x_{23} = 300 - 300 = 0 \\ b'_3 &= b_3 - x_{22} = 300 - 300 = 0 \end{aligned}$$

Étape 3 $a'_2 = 0$ et $b'_3 = 0$, la ligne 2 est la dernière ligne, on élimine donc la colonne 3.

Étape 4 Il ne reste que la ligne 2, la solution réalisable de base initiale est trouvée.

Origines	Déstinations			Offre
	1 ♣	2 ♣	3 ♣	
1 ♣	25 □ 300	17 □ 50	16	350 ♭ 0
2	24	18 □ 250	14 □ 300	550 ♭ 0
Demande	300 ♭ 0	300 ♭ 0	300 ♭ 0	900

Nous pouvons calculer le coût de transport total de ce plan d'expédition :

$$z = 25(300) + 17(50) + 18(250) + 14(300) = 17050$$

On voit tout de suite que la solution trouvée n'est pas optimale. En effet, le coût total de ce plan de transport est plus élevé que celui trouvé à l'aide de la méthode de la matrice minimale, qui était de 16 550.

3.4 Problème dual:

Le problème dual du problème de tranport original (T_e) est:

$$Max\ w(u,v) = \sum_{i=1}^{m} a_i u_i + \sum_{j=1}^{n-1} b_j v_j \quad (13)$$

$$u_i + v_j \leq c_{ij} \quad \forall\ i,j \quad (14)$$

soit $X(B)$ une solution de base réalisable de départ.

3.4.1 Obtention de la solution du dual correspondant à une solution de base .

Proposition 3.4.1 *La solution du dual correspondant à une solution de base $X(B)$ est obtenue par le système*

Alors $X(\theta)$ est la solution obtenu en donnant la valeur θ à la variable $x_{p,q}$, gardant toutes les autres variables hors base nulles et révaluant les valeurs des variables de base.

La valeur maximale de θ telle que $X(\theta) \geq 0$ est donnée par:

$$\theta_1 = \min\{x_{ij} \ / \ (i,j) \ \text{ est une cellule correspondant à } \ -\theta\} = x_{rs} \qquad (18)$$

La variable de sortie est la variable x_{rs}.

Donc la nouvelle solution de base est: $X(\theta_1)$, la valeur de la fonction objectif correspondante est:

$$z(\bar{X}) + \bar{c}_{pq}.\theta_1 \qquad (19)$$

Cette procédure est répétée jusqu'à l'obtention d'une solution réalisable optimale.

3.5 Exemple

Considérons le problème de transport avec la matrice des coûts suivante:

	$j=1$	2	3	4	5	6
$i=0$	$c_{11}=10$	4	12	□ 1	12	□ 8
2	□ −3	2	6	2	□ 3	□ −1
3	□ 2	□ 4	2	10	2	1
4	10	−2	□ 5	□ −5	7	5

Exemple 3.5.1 *Les c_{ij} sont représentés dans le coin inférieur à droite de la cellule. Les cellules correspondant aux cases encadrées représentent une base. Il est demandé de trouver la solution du dual correspondant à cette base. On pose $v_6 = 0$, en regardant les équations* (3.10) *correspondant aux cellules de base* $(2,6)$ *et* $(1,6)$ *on obtient:*

$$u_2 + v_6 = -1 \text{ donc } u_2 = -1$$

$$u_1 + v_6 = 8 \text{ donc } u_1 = 8$$

Maintenant en utilisant les valeurs de u_1, u_2 *dans les équations* (3.10) *correspondant aux cellules de base de* $(2,1),(1,4)$ *et* $(2,5)$*(on obtient* $v_1 = -2$, $v_4 = -7$ *et* $v_5 = 4$. *En continuant de cette façon on obtient la solution dual demandée, qui est représenté sur le tableau suivant. Les coûts relatifs* $\bar{c}_{ij}$ *sont donnés dans le coin supèrieur à gauche de la cellule:*

	$j=1$	2	3	4	5	6	u_i
$i=1$	$4=\bar{c}_{11}$ $c_{11}=10$	-4 4	1 12	□ 1	0 12	□ 8	8
2	□ -3	3 2	4 6	10 2	□ 3	□ -1	-1
3	□ 2	□ 4	-5 2	13 10	-6 2	-3 1	4
4	10 10	-4 -2	□ 5	□ -5	1 7	3 5	2
c_j	-2	0	3	-7	4	0	

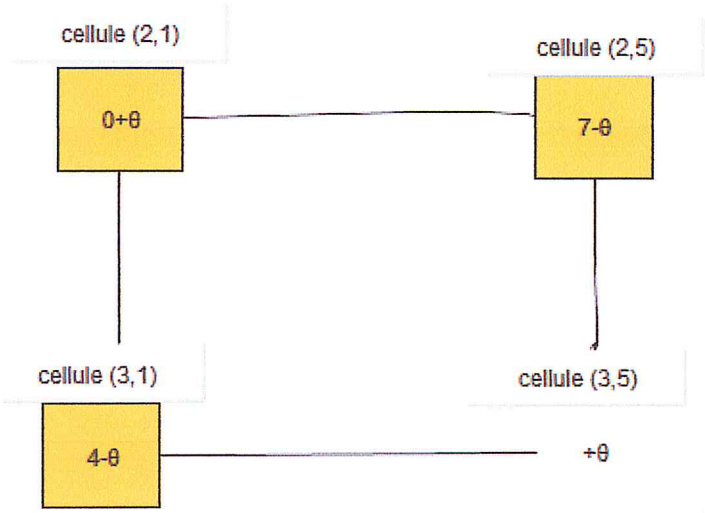
Dans chaque cellule de base le coût relatif est égal à zéro. Ceci n'est pas représenté sur le tableau.

Cette solution de base n'est pas optimale car les coûts relatifs ne sont tous positifs ou nuls.

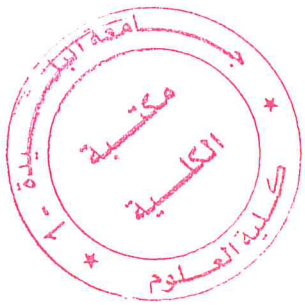
On fait un changement de base.

La variable d'entrée est la variable x_{35} car $\bar{c}_{3,5} = \max_{i,j/\bar{c}_{i,j}<0} |\bar{c}_{i,j}|$

La $\theta - boucle$ est formée par la suite des cellules suivantes: (3,5), (2,5),(2,1),(3,1) et est représentée sur la Figure suivante:



La $\theta - boucle$



Détermination de la variable de sortie:

$$\theta_1 = \min\{x_{ij} \ /(i,j) \text{ est une cellule correspondant à } -\theta\} = \min\{4,7\} = 4$$

θ_1 est atteinte pour la cellules (3,1) donc la varaible de sortie est x_{31}. La nouvelle solution de base $\overline{X}$ est definie par:

$\overline{x}_{35} = 4, \overline{x}_{25} = 7 - 4 = 3, \overline{x}_{21} = 4.$

les autres varaibles sont inchangées.

On poursuit de cette façon jusqu'à l'optimalité.

Exemple 3.5.2

3.6 Annexe

Dans cette partie nous proposons un programme en c++ qui résout le probléme de transport équilibré .

Nous implémentons ce programme sur deux exemples . Le premier exemple est l'exemple 1 donné au chapitre 3.le 2 exemple sera représenté par un tableau.

3.6.1 Programme de transport:

```
#include <stdio.h>
    #include <math.h>
    #define NMAX 10
    double C[NMAX][NMAX], XM[NMAX][NMAX];
    double R[NMAX][NMAX];
    double DAO[NMAX], RAD[NMAX];
    int P[5][2*NMAX+5];
    int ID,IDES,IF1,IOPTIMAL,ISOU,IX,IY,LT,NR;
    double CC,CT,QT,TT,XINFCC;
    void Init() {
    int I,J;
    printf("\n MODEL DE TRANSPORT\n\n");
    printf(" LE NOMBRE DES SOURCES EST:"); scanf("%d", &ISOU);
    printf("\n NOMBRE DESTINATIONS EST: "); scanf("%d", &IDES);
    printf("\n ENTRER LA QUANTITE DISPONIBLES:\n");
    for (I=1; I<=ISOU; I++) {
    printf(" SOURCE #%d = ", I); scanf("%lf", &DAO[I]);
    }
    printf("\n ENTRER LA QUANTITE DEMANDE:\n");
    for (I=1; I<=IDES; I++) {
```

```
printf(" DESTINATION #%d = ", I); scanf("%lf", &RAD[I]);
}
printf("\n ENTRER LES COUTS DE LA MATRICE:\n");
for (I=1; I<=ISOU; I++)
    for (J=1; J<=IDES; J++) {
printf(" DE LA SOURCE #%d VERS DESTINATION #%d = ", I, J);
scanf("%lf", &C[I][J]);
}
printf("\n\n LE MODELE DE TRANSPORT\n");
}
void Corner();
void Optimal();
void TCost();
void SK(int I,int J);
void INCO(int I,int J);
void SubMain(){
Corner();
Optimal();
TCost();
}
void Corner() { //METHODE COIN-NORD
//liGNES: e20, e50, e60
int I,J;
I=1; J=1;
e20:if (DAO[I]<=RAD[J]) goto e50;
XM[I][J] += RAD[J]; DAO[I] -= RAD[J];
RAD[J]=0; J++;
```

```
goto e60; //SINON
e50:XM[I][J] += DAO[I]; RAD[J] -= DAO[I];
DAO[I]=0; I++;
e60:if (I<=ISOU && J<=IDES) goto e20;
}
void Optimal() {
//liGNES: e10, e70, e140, e150
int I,J;
e10: XINFCC=0.0;
for (I=1; I<=ISOU; I++)
for (J=1; J<=IDES; J++) {
if (XM[I][J] != 0.0) goto e70;
SK(I,J);
INCO(I,J);
e70:; }
    if (XINFCC>=0.0) {
IOPTIMAL=1;
goto e150;
}
for (I=1; I<=LT; I++) {
IX=P[3][I]; IY=P[4][I];
if (I % 2 == 0) {
XM[IX][IY] -= TT;
goto e140;
}
XM[IX][IY] += TT;
e140:;}
```

```
e150:if (IOPTIMAL==0) goto e10;
}
void SK(int I, int J) {
//liGNES: e70, e160, e260
int I1,I2;
for (I1=1; I1<=ISOU; I1++)
for (I2=1; I2<=IDES; I2++)
R[I1][I2]=XM[I1][I2];
for (I1=1; I1<=ISOU; I1++) R[I1][0]=0.0;
for (I2=1; I2<=IDES; I2++) R[0][I2]=0.0;
R[I][J]=1.0;
e70: for (I2=1; I2<=IDES; I2++) {
if (R[0][I2]==1.0) goto e160;
NR=0;
for (I1=1; I1<=ISOU; I1++)
if (R[I1][I2] != 0.0) NR++;
    if (NR!=1) goto e160;
for (I1=1; I1<=ISOU; I1++) R[I1][I2]=0.0;
R[0][I2]=1.0; IF1=1;
e160:;}
for (I1=1; I1<=ISOU; I1++) {
if (R[I1][0]==1.0) goto e260;
NR=0;
for (I2=1; I2<=IDES; I2++)
if (R[I1][I2] != 0.0) NR++;
if (NR!=1) goto e260;
for (I2=1; I2<=IDES; I2++) R[I1][I2]=0.0;
```

```
R[I1][0]=1.0; IF1=1;
e260:;}
if (IF1==1) {
IF1=0; goto e70;
}
}
void INCO(int I, int J) {
//liGNES: e20,e70,e130,e170,e180,e230
int I1,I2;
P[1][1]=I; P[2][1]=J; IX=I; IY=J; ID=1; CC=0.0; QT=999999.0;
e20: ID++; IF1=0;
    for (I1=1; I1<=ISOU; I1++) {
if (R[I1][IY]==0.0 || I1==IX) goto e70;
P[1][ID]=I1; P[2][ID]=IY; IX=I1; CC -= C[IX][IY];
IF1=1; I1=ISOU;
if (XM[IX][IY] < QT && XM[IX][IY] > 0.0) QT=XM[IX][IY];
e70:;}
if (IF1==0) goto e170;
ID++; IF1=0;
    for (I2=1; I2<=IDES; I2++) {
if (R[IX][I2]==0.0 || I2==IY) goto e130;
P[1][ID]=IX; P[2][ID]=I2; IY=I2; CC += C[IX][IY];
IF1=1; I2=IDES;
e130:;}
if (IF1==0) goto e170;
if (IX!=I || IY!=J) goto e20;
goto e180;
```

```
e170:printf(" SOLUTION REALISABLE!\n");
return;
e180:if (CC>0.0 || CC>XINFCC) goto e230;
TT=QT; XINFCC=CC; ID–; LT=ID;
for (I1=1; I1<=ID; I1++) {
P[3][I1]=P[1][I1]; P[4][I1]=P[2][I1];
}
e230:;}
void TCost() {
int I,J;
CT=0.0;
printf("\n TRANSPORTER:\n");
for (I=1; I<=ISOU; I++)
for (J=1; J<=IDES; J++) {
CT += XM[I][J]*C[I][J];
if (XM[I][J]==0.0) goto e10;
printf(" DE LA SOURCE #%d VERS DESTINATION #%d= %8.2f\n", I, J,
XM[I][J]);
e10:;}
printf("\n LE COUT TOTAL DE TRANSPORT EST = %10.1f\n\n", CT);
}
int main() {
Init();
SubMain();
}
```

3.6.2 Implémentation sur deux exemples:

Exemple 3.6.1 *Aprés l'exécution du programme sur l'exemple 1(chapitre3) on obtient les resultats suivante:*

```
 DESTINATION #1 = 300
 DESTINATION #2 = 300
 DESTINATION #3 = 300

ENTRER LES COUTS DE LA MATRICE:
 DE LA SOURCE #1 VERS DESTINATION #1 = 25
 DE LA SOURCE #1 VERS DESTINATION #2 = 17
 DE LA SOURCE #1 VERS DESTINATION #3 = 16
 DE LA SOURCE #2 VERS DESTINATION #1 = 24
 DE LA SOURCE #2 VERS DESTINATION #2 = 18
 DE LA SOURCE #2 VERS DESTINATION #3 = 14


LE MODELE DE TRANSPORT

TRANSPORTER:
   DE LA SOURCE #1 VERS DESTINATION #1=    50.00
   DE LA SOURCE #1 VERS DESTINATION #2=   300.00
   DE LA SOURCE #2 VERS DESTINATION #1=   250.00
   DE LA SOURCE #2 VERS DESTINATION #3=   300.00

 LE COUT TOTAL DE TRANSPORT EST =    16550.0
```

Exemple 3.6.2 *Soit la société X possède trois abattoirs en trois villes différentes et fornit trois autres villes en de première qualité comme.*

Abatoire	*Stok*	*Ville*	*Demande*
Chlef	*120*	*Alger*	*150*
Sétif	*80*	*Blida*	*70*
Oran	*80*	*Tiaret*	*60*

Les coûts de transport (en DA/tonne) dépondent des contrats particuliéres avec les compagnies de chemin de fer et fret aérien sont résumés dans le tableau ci-dessous:

	Alger	*Blida*	*Tiaret*
Chlef	*8*	*5*	*6*
Sétif	*15*	*10*	*12*
Oran	*3*	*9*	*10*

et donc on obtient le tableau suivant

	Alger	*Blida*	*Tiaret*	*Disponibilité*
Chlef	*8*	*15*	*6*	*120*
Sétif	*15*	*10*	*12*	*80*
Oran	*3*	*9*	*10*	*80*
Demande	*150*	*70*	*60*	*280*

Aprés l'exécution du programme, on obtient les résultats suivants:

```
ENTRER LES COUTS DE LA MATRICE:
 DE LA SOURCE #1 VERS DESTINATION #1 = 8
 DE LA SOURCE #1 VERS DESTINATION #2 = 5
 DE LA SOURCE #1 VERS DESTINATION #3 = 6
 DE LA SOURCE #2 VERS DESTINATION #1 = 15
 DE LA SOURCE #2 VERS DESTINATION #2 = 10
 DE LA SOURCE #2 VERS DESTINATION #3 = 12
 DE LA SOURCE #3 VERS DESTINATION #1 = 3
 DE LA SOURCE #3 VERS DESTINATION #2 = 9
 DE LA SOURCE #3 VERS DESTINATION #3 = 10

LE MODELE DE TRANSPORT

TRANSPORTER:
   DE LA SOURCE #1 VERS DESTINATION #1=     70.00
   DE LA SOURCE #1 VERS DESTINATION #3=     50.00
   DE LA SOURCE #2 VERS DESTINATION #2=     70.00
   DE LA SOURCE #2 VERS DESTINATION #3=     10.00
   DE LA SOURCE #3 VERS DESTINATION #1=     80.00

 LE COUT TOTAL DE TRANSPORT EST =     1920.0
```

Conclusion

Dans ce travail, nous avons considéré les problèmes des flots et le problème de transport . Après avoir donné la définition d'un flot, le problème du flot maximum a été défini.

Ensuite nous avons donné une méthode de résolution du problème de flot maximum qui est l'algorithme de Ford et Fulkerson . Après nous avons défini le problème du flot de coût minimum. Des exemples d'éclaircissement et des formulation en programmation linéaire des problèmes précédents ont été présentées.

Dans la deuxième partie de ce mémoire, le problème de transport est introduit. Des définitions et des exemples ont été présentés pour une bonne compréhension de ce problème.

Deux approches sont établies pour la résolution de ce problème. La première est basée sur la modélisation par les flots. La deuxième est une méthode de résolution utilisant la programmation linéaire .

En fin, programme codé en C++ a été élaboré pour la résolution de problème de transport.

Concernant les problèmes de transport notons que dans ce mémoire le problème du flot de cout de minimum a été modélisé par la programmation linéaire . Un travail intéressant dans le futur est de proposer la méthode primale duale pour la résolution du problème de flot de cout minimum et comparer avec celle basée sur la programmation linéaire.

Quant aux perspectives du problème de transport on pourra calculer la complexité de l'algorithme proposé et de l'améliorer.

REFERENCES

[1] C.Berge, Graphe et hypergraphe, 3ème édition. Dunod 1983.

[2] Linear and combinatiorial programing Katt G.Murty

John Wily and Sons.INC

New York . London. Sydney Toronto.

[1] L.R.Ford,Jr.and D.R.Fulkerson.Maximal Flow Through a Network.Canadian Journal of Mathematics,Vol.8,pages 399-404,1956.

[2] L R Ford et D R.Fulkerson,Flows in Network,princeton Nj,princeton University Press,1962.

[3] Optimisation Combinatoire: Graphes et programmation linéaire.

Michel Sakarovich.

Hermann, editeurs des sciences et des arts.

[4] Optimisation Combinatoire: programmation discrête.

Michel Sakarovich.

Hermann, editeurs des sciences et des arts.

[5] Optimisation appliqué.

Yadolah Dodage/Sylvie Gonano Weber et jean_Pierre Renfer

Spring_Verlag France 2005.