N°d'ordre : 01/2009-M/MT

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET

DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE

«HOUARI BOUMEDIEN»

FACULTE DE MATHEMATIQUES

جامعة هواري بومدين
للعلوم والتكنولوجيا
U S T H B

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTER

EN : MATHEMATIQUES

Spécialité : Recherche opérationnelle

Par : MANCER Salem

Sujet

Optimisation multicritère appliquée au domaine des transports

Soutenu le 19/10/2009 , devant le jury composé de :

Mr- MOULAI Mustapha	Professeur	USTHB.	Président
Mr- BERRACHEDI Abdelhafid	Professeur	USTHB.	Dteur de thèse
Mr-BOUROUBI Sadek	Maître de conférence. A	USTHB.	Examinateur
Mr-CHERGUI Mohamed El Amine	Maître assistant.A	USTHB.	Invité
Mr- MEZGHICHE Abdelhak	Maître assistant.A	USTHB.	Invité

A quoi sert aujourd'hui ou demain ton visage est à la mesure de l'éternité
A quoi servent les profits et les pertes les défaites peuvent être des victoires

Mohamed Iqbal

Au terme de ce mémoire je me fais un devoir de rendre hommage à tous ceux qui se sont sacrifiés, et continuent de le faire, pour que le peuple algérien connaisse un jour le bien être.

REMERCIEMENTS

Je souhaite exprimer ma profonde reconnaissance à monsieur Abdelhafid BERRACHEDI, professeur à l'université des sciences et de la technologie Houari Boumediene, pour son soutien et sa disponibilité.

Je remercie les membres du jury qui ont accepté de juger ce travail et d'y apporter leur caution :

- M^{r} MOULAI Mustapha qui m'a fait l'honneur d'accepter la présidence du jury.

- M^{r} BOUROUBI Sadek qui a eu l'amabilité d'accepter de lire et de corriger ce modeste mémoire.

- M^{rs} CHERGUI Mohamed El Amine et Mr MEZGHICHE Abdelhak pour m'avoir honoré de leur participation au jury

- Mes parents, ma famille et particulièrement Ammar Abderraouf.
- Toutes les personnes sincères qui m'ont exprimé leur confiance et sympathie.

Table des matières

Introduction

La recherche opérationnelle dont la vocation est la résolution des problèmes concrets s'est considérablement développée avec la généralisation de l'utilisation des ordinateurs…
Néanmoins les méthodes d'optimisation classiques de la programmation linéaire ou non-linéaire qui ne prennent en compte qu'une seule fonction objectif deviennent inappropriées dés qu'il s'agit de problèmes à objectifs multiples.
Par exemple dans le problème de transport il est naturel de s'intéresser au coût mais aussi à la durée du trajet, aux quantités transportées, à l'itinéraire le plus confortable…etc.

Bien que depuis plus de deux siècles les problèmes à objectifs multiples soient la préoccupation des économistes, qui d'ailleurs nous ont légués les notions de satisfaction, meilleur compromis, fonction d'utilité, paréto-optimal…etc.

Mais depuis trois décennies les mathématiciens ont développé des méthodes d'optimisation multicritère qui génèrent des solutions dites de compromis, qui répondent globalement aux exigences du problème sans pour autant optimiser simultanément tous les critères, ce qui est généralement impossible.

Le mémoire est articulé autour de trois chapitres dont le premier est consacré aux rappels nécessaires de la programmation linéaire, dont plusieurs résultats seront mis à contribution, on a défini le problème de transport comme un programme linéaire puis on a introduit la terminologie de l'optimisation multicritère.

Du fait que le problème de transport peut être ramené à un problème de flot de coût minimum on a abordé au deuxième chapitre les différents résultats de l'optimisation dans les réseaux et la théorie des flots.

Au chapitre trois on a introduit la notion de flot maximal et développé une procédure permettant de générer un flot maximal à valeur minimale.

Ce qui nous a permis de définir et d'étudier certains problèmes de flots bicritères. Puis on a essayé de répondre à deux problèmes de transport bicritère dont le premier consiste à minimiser le nombre d'entrepôts utilisés pour satisfaire la demande tout en minimisant les coûts de transport et le deuxième consiste à vider le maximum d'entrepôts tout en minimisant les coûts de transport.

Dans les deux cas on a trouvé des solutions optimisant chacun des critères et donné des solutions de compromis.

Chapitre I

Rappels ([1]et[8]) :

I-1 Programmation linéaire :

la programmation linéaire a pour objet l'optimisation d'une fonction linéaire qui est soumise à un ensemble de contraintes linéaires au sens large.

On distingue les trois formes de programmes linéaires suivantes :

La forme : $\begin{cases} Ax = b \quad x \geq 0 \\ cx = z(\max) \end{cases}$ appelée la forme standard.

La forme : $\begin{cases} Ax \geq b \quad x \geq 0 \\ cx = z(\min) \end{cases}$ appelée la forme canonique de minimisation.

La forme : $\begin{cases} Ax \leq b \quad x \geq 0 \\ cx = z(\max) \end{cases}$ appelée la forme canonique de maximisation.

Remarque :

En tenant compte du fait que : $a \leq b \Leftrightarrow \begin{cases} a + c = b \\ c \geq 0 \end{cases}$; $a \geq b \Leftrightarrow \begin{cases} a - c = b \\ c \geq 0 \end{cases}$

On peut exprimer une variable x_j en fonction de deux variables x'_j et x''_j de la façon suivante :

Si $x_j \geq 0 \Leftrightarrow x'_j = x_j \geq 0$.

Si $x_j \leq 0 \Leftrightarrow x'_j = -x_j \geq 0$.

Si x_j de signe quelconque $\Leftrightarrow x_j = x'_j - x''_j$, $x_j \geq 0$ *et* $x''_j \geq 0$.

De même que maximiser (Cx) revient à minimiser (- Cx).

Par conséquent tout programme linéaire peut être ramené à une forme canonique de maximisation.

Au programme linéaire (P) suivant écrit sous forme canonique de maximisation, on peut associer un programme linéaire (D) appelé "Problème Dual" et (P) est dit primal.

$$(P)\ \begin{cases} Ax \le b \\ cx = z(\max) \end{cases} \quad x \ge 0\ , \qquad (D)\ \begin{cases} yA \ge c \\ yb = W(Min) \end{cases} \quad y \ge 0\ .$$

Comme tout programme linéaire peut être ramené à une forme canonique de maximisation, alors tout programme linéaire admet un dual. Par conséquent le dual de (D) est (P).

Aussi on peut déduire les règles suivantes pour écrire le programme dual de tout programme linéaire :

Primal (dual)	**Dual (primal)**
Fonction à maximiser	**Fonction à minimiser**
$i^{ème}$ contrainte $\le$	$i^{ème}$ variable ≥ 0
$i^{ème}$ contrainte $\ge$	$i^{ème}$ variable ≤ 0
$i^{ème}$ contrainte $=$	$i^{ème}$ variable quelconque
$j^{ème}$ variable ≤ 0	$j^{ème}$ contrainte $\le$
$j^{ème}$ variable ≥ 0	$j^{ème}$ contrainte $\ge$
$j^{ème}$ variable quelconque	$j^{ème}$ contrainte $=$

Exemple :

$$(P)\begin{cases} 2x_1 - x_2 + 2x_3 = 2 \\ -x_1 + 3x_2 + x_3 \le -3 \\ x_1 - 3x_2 + 5x_3 \ge 4 \\ 5x_1 - 2x_2 + 4x_3 = Z(Max) \\ x_1 \ge 0, x_2 \le 0, x_3 qcq \end{cases} \qquad (D)\begin{cases} 2y_1 - y_2 + y_3 \ge 5. \\ -y_1 + 3y_2 - 3y_3 \le -2. \\ 2y_1 + y_2 + 5y_3 = 4. \\ 2y_1 - 3y_2 + 4y_3 = W(Min). \\ y_1 \, qcq, y_2 \ge 0, y_3 \le 0 \end{cases}$$

L'exemple suivant (adapté de [8]) nous permet d'illustrer la relation entre un programme linéaire et son dual :

Mensuellement une entreprise fabrique 1000 tonnes de phosphate à Skikda et 500 tonnes à Annaba. Elle doit approvisionner ses usines d'engrais à Alger, Oran et Mila dont les besoins mensuels sont respectivement : 600 T, 500 T et 400 T.

Les coûts de transport sont supposés varier proportionnellement aux quantités transportées .Les coûts unitaires de transport sont donnés par le tableau suivant :

	Alger (1)	0ran (2)	Mila (3)
Skikda (1)	4	7	2
Annaba (2)	5	8	3

Notre problème consiste à déterminer un plan de transport optimal .i-e, à déterminer les poids de phosphate à envoyer de chaque ville (i) vers chaque usine (j) donc trouver : $x_{ij} \ge 0$ afin de satisfaire les demandes mensuelles sans dépasser les quantités disponibles tout en minimisant les coûts de transport.

Le programme linéaire s'écrit alors :

$$\begin{cases} x_{11} + x_{21} \ge 600 \\ x_{12} + x_{22} \ge 500 \\ x_{13} + x_{23} \ge 400 \\ x_{11} + x_{12} + x_{13} \le 1000 \\ x_{21} + x_{22} + x_{23} \le 500 \\ x_{ij} \ge 0. \; i = 1,2; \; j = 1,2,3 \\ 4x_{11} + 7x_{12} + 2x_{13} + 5x_{21} + 8x_{22} + 3x_{23} = Z(Min) \end{cases}$$

Sous forme canonique de Max :

$$\begin{cases} x_{11} + x_{12} + x_{13} \le 1000 \\ x_{21} + x_{22} + x_{23} \le 500 \\ -x_{11} - x_{21} \le -600 \\ -x_{12} - x_{22} \le -500 \\ -x_{13} - x_{23} \le -400 \\ -4x_{11} - 7x_{12} - 2x_{13} - 5x_{21} - 8x_{22} - 3x_{23} = Z(Max) \end{cases} \qquad x_{ij} \ge 0$$

Et dont le dual s'écrit:

$$\begin{cases} y_1 - y_3 \ge -4 \\ y_1 - y_4 \ge -7 \\ y_1 - y_5 \ge -2 \\ y_2 - y_3 \ge -5 \\ y_2 - y_4 \ge -8 \\ y_2 - y_5 \ge -3 \\ 1000y_1 + 500y_2 - 600y_3 - 500y_4 - 400y_5 = W(Min) \end{cases} \qquad y_i \ge 0$$

Supposons maintenant qu'un transporteur prenne contact avec la direction de l'entreprise et lui propose de lui acheter le phosphate aux prix M1 et M2 des villes de Skikda et Annaba, de se charger du transport et de lui revendre le phosphate aux prix N1, N2 et N3 à Alger ,Oran et Mila.

Pour convaincre la direction, le transporteur garantit que ses prix seront compétitifs avec les coûts actuels i-e,

$$\begin{cases} N_1 - M_1 \le 4 \\ N_2 - M_1 \le 7 \\ N_3 - M_1 \le 2 \end{cases} \qquad \begin{cases} N_1 - M_2 \le 5 \\ N_2 - M_2 \le 8 \\ N_3 - M_2 \le 3 \end{cases} \qquad \begin{matrix} N_i \ge 0 & i = 1,2,3 \\ M_i \ge 0 & i = 1,2 \end{matrix}$$

Le transporteur ayant obtenu le marché il veut remplir son contrat tout en maximisant son profit, donc maximiser l'expression :

$$600N_1 + 500N_2 + 400N_3 - 1000M_1 - 500M_2$$

En posant :

$y_1 = M_1$, $y_2 = M_2$, $y_3 = N_1$, $y_4 = N_2$, $y_5 = N_3$:

Le programme linéaire sera :

$$\begin{cases} y_1 - y_3 \geq -4 \\ y_1 - y_4 \geq -7 \\ y_1 - y_5 \geq -2 \\ y_2 - y_3 \geq -5 \\ y_2 - y_4 \geq -8 \\ y_2 - y_5 \geq -3 \\ 1000y_1 + 500y_2 - 600y_3 - 500y_4 - 400y_5 = W(\min) \end{cases}$$

Donc le programme linéaire du transporteur est exactement le dual du programme linéaire de l'entreprise . Aussi c'est le même problème vu sous un centre d'intérêt différent.

Théorème 1.1:

$\forall\, \bar{x}$ Solution réalisable pour (P) et $\forall \bar{y}$ solution réalisable pour (D) .Alors $z(\bar{x}) \leq w(\bar{y})$
i-e, $c\bar{x} \leq \bar{y}b$.

Corollaire 1.2 :

Si $\exists x^*$ une solution réalisable pour (P) et y* une solution réalisable pour (D) tel que : c x* = y* b alors : x* optimale pour (P) et y* optimale pour (D).

Théorème des écarts complémentaires :

Introduisons la définition suivante :

Définition 1.3 :

Soit $\bar{x}$ une solution réalisable de (P) alors la i^{éme} contrainte de (P) est dite :

-Serrée si $A_i\ \bar{x} = b_i$

-Lâche si $A_i\ \bar{x} < b_i$

Théorème 1.4:

Une C.N.S. pour qu'un couple de solutions réalisables de deux programmes linéaires duaux (P) et (D) soit un couple de solutions optimales est que :

-Si une contrainte de (P) est lâche alors la variable correspondante de (D) est nulle.
-Si une variable de (P) est positive la contrainte correspondante de (D) est serrée.

Notations :

Un élément de la matrice A sera note A_i^j .

A (m,n) est une matrice à (m) lignes et (n) colonnes.

La i^{ème} ligne sera notée A_i , la j^{ème} colonne sera notée A^j et A_i^j l'élément de la i^{èmé} ligne et la j^{èmé} colonne.

Soit I$\subset \{1,2,...,m\}$; J$\subset \{1,2,...,n\}$:

A_I^J : Sous matrice de A obtenue en supprimant toutes les colonnes n'appartenant pas à J et toutes les lignes n'appartenant pas à I.

A^J : Sous matrice de A obtenue en supprimant toutes les colonnes n'appartenant pas à J.

A_I : Sous matrice de A obtenue en supprimant toutes les lignes n'appartenant pas à I.

Rappel: Soit une matrice A(m,n), on appelle rang de A noté rg (A) le plus grand entier (k) tel que A possède une sous matrice carrée d'ordre (k) non singulière.

Définition 1.5: On appelle matrice augmentée du système Ax = b la matrice M notée $M = [A, b]$ Obtenue en adjoignant la colonne (b) à la suite des colonnes de (A).

Théorème 1.6 : (S) étant le système $Ax = b$, $A : mxn$ alors :

(S) admet des solutions $\Leftrightarrow$ rg (A) = rg (M).

(S) admet une solution unique $\Leftrightarrow$ rg (A) = m.

Notations :

On suppose (sans perte de généralités) que rg (A) = m ;(S) est dit de plein rang

1) Si $m < n$ infinité de solutions.

Si m=n solution unique donnée par les formules de Cramer.

2) Soit B une sous matrice de A carrée d'ordre (m) inversible sans perte de généralité on peut supposer $A = [B, H]$ (les premières colonnes de A sont supposées indépendantes)

donc B : m x m et chaque colonne de $\boldsymbol{B \in IR^m}$. B étant inversible $\Leftrightarrow$ det B $\neq 0 \Leftrightarrow$ toutes les colonnes de B sont indépendantes.

Par conséquent les colonnes forment une base de $\boldsymbol{IR^m}$ (Par abus de langage on dira B base de A).

Soit J l'ensemble des indices des colonnes de B (Par abus de langage on dira J base de A).

x_J : Les variables dont les indices sont dans J, qu'on appellera variables de base (x_B).

$x_{\bar{J}}$: Les variables dont les indices ne sont pas dans J qu'on appellera les variables hors base (x_H).

A s'écrit alors [A^J , $A^{\bar{J}}$] et $x = \begin{pmatrix} x_J \\ x_{\bar{J}} \end{pmatrix}$. Le système $Ax = b$ s'écrira : $A^J x_J + A^{\bar{J}} x_{\bar{J}} = b.$

A^J Étant inversible : $x_J + (A^J)^{-1}.A^{\bar{J}} x_{\bar{J}} = (A^J)^{-1} b$ donc : $x_J = (A^J)^{-1} b - (A^J)^{-1}.A^{\bar{J}} x_{\bar{J}}$

Donc : $x = \begin{pmatrix} x_J \\ x_{\bar{J}} \end{pmatrix} = \begin{pmatrix} (A^J)^{-1} b - (A^J)^{-1} A^{\bar{J}} x_{\bar{J}} \\ x_{\bar{J}} \end{pmatrix}$; $x_{\bar{J}} \in IR^{n-m}$

Donc le système est entièrement résolu.

On dit qu'on a résolu le système par rapport à la base J. L'ensemble des solutions dépend du choix des variables $x_{\bar{J}}$ (i-e, hors base).

La solution obtenue en posant $x_{\bar{J}} = 0$, i-e, $x = \begin{pmatrix} (A^J)^{-1} b \\ 0 \end{pmatrix}$ est appelée solution de base associée à la base J.

Remarques :

- Comme le nombre de bases est inferieur ou égal à C_n^m donc les solutions de base sont en nombre fini.
- A chaque base correspond une solution de base unique.
- Si au moins deux bases ont une même solution de base ($\bar{x}$) on dit que ($\bar{x}$) est une solution dégénérée.
- Dans une solution de base $\bar{x}$ si une variable de base est nulle alors $\bar{x}$ est dégénéré.

De ce qui précède pour un programme linéaire on fait la transformation suivante :

Soit (P) $\begin{cases} Ax = b \quad x \geq 0 \\ cx = z(Max) \end{cases}$ .A : m x n ; rg (A)= m.

Soit $J \subset \{1,2,...,n\}$ une base.

Donc $|J| = m$ et $\{A^j ; j \in J\}$ sont libres.

Donc : $Ax = A^J x_J + A^{\bar{J}} x_{\bar{J}}$ et (P) va s'écrire $(P): \begin{cases} x_J + (A^J)^{-1} A^{\bar{J}} x_{\bar{J}} = (A^J)^{-1} b \\ C^J x_J + C^{\bar{J}} x_{\bar{J}} = z(Max) \end{cases}$ $x \geq 0$.

Qui est équivalent à :

$$(\hat{P}): \begin{cases} x_J + (A^J)^{-1} A^{\bar{J}} x_{\bar{J}} = (A^J)^{-1} b \\ C^J \left[(A^J)^{-1} b - (A^J)^{-1} A^{\bar{J}} . x_{\bar{J}} \right] + C^{\bar{J}} x_{\bar{J}} = z(Max). \end{cases} x \geq 0.$$

$$(\hat{P}): \begin{cases} x_J + (A^J)^{-1} A^{\bar{J}} x_{\bar{J}} = (A^J)^{-1} b \\ (C^{\bar{J}} - C^J (A^J)^{-1} A^{\bar{J}}) x_{\bar{J}} = Z(Max) - C^J (A^J)^{-1} b \end{cases} x \geq 0.$$

$$(\hat{P}): \begin{cases} \hat{A} x = \hat{b} \\ \hat{C} x = Z(Max) - \overset{\wedge}{\alpha} \end{cases} x \geq 0.$$

Avec : $\hat{A} = (A^J)^{-1} . A ; \hat{b} = (A^J)^{-1} b ; \hat{C} = C - \pi A = (0, C^{\bar{J}} - \pi A^{\bar{J}})$

$avec : \pi = C^J (A^J)^{-1} ; \overset{\wedge}{\alpha} = \pi b.$

$(\hat{P})$ est dit écrit sous forme basique (forme canonique par rapport à la base).

Remarques :

- $(\hat{P})$ c'est (P) écrit sous forme canonique par rapport à la base J, parfois $(\hat{P})$ est appelé forme basique de (P).
- π : Vecteur multiplicatif relatif à J.
- $\hat{C}$: Coûts réduits relatifs à J.
- La solution de base $\begin{pmatrix} \hat{b} \\ 0 \end{pmatrix} = x(J)$ est dite solution de base associée à J.
- Si $\hat{b} \geq 0 \Rightarrow x(J) \geq 0$ *donc* $x(J)$ solution réalisable on dira que (J) est une base réalisable.
- Si $x(J)$ est une solution optimale on dira que (J) est une base optimale.

On aboutit au théorème fondamental de la programmation linéaire :

Théorème 1.7 : [8]

a) Si (P) admet une solution réalisable alors (P) admet une solution de base réalisable.
b) Si (P) admet une solution optimale alors (P) admet une solution de base optimale.
c) Si (P) admet une solution réalisable et z borné supérieurement alors (P) admet une solution optimale.

Conséquence : Résoudre un P.L revient à se limiter aux solutions de bases réalisables qui sont en nombre fini comme les bases.

Théorème 1.8 : Si $\hat{C} \leq 0$ alors J optimale.

Algorithme du simplexe :

$$\text{Soit } (P): \begin{cases} Ax = b \\ cx = z(\max) \end{cases} \quad x \geq 0. \quad \begin{matrix} A : mxn \\ rg(A) = m \end{matrix}$$

Initialisation : Soit $J \subset \{1,2,...,n\} \mid J = m$ une base réalisable . On écrit (P) sous forme canonique par rapport à J .On obtient : $(\hat{P}) : \begin{cases} \hat{A}x = \hat{b} \\ \hat{c}x = z(\max) - \hat{\alpha} \end{cases}$ $x \geq 0$ .où :

$$\hat{A} = (A^J)^{-1} A$$

$$\hat{b} = (A^J)^{-1}.b$$

$$\hat{c} = c - \pi A$$

Où : $\pi = c^J (A^J)^{-1}$ solution de base $x(J) = \begin{pmatrix} x_J \\ x_{\bar{J}} \end{pmatrix} = \begin{pmatrix} \hat{b} \\ 0 \end{pmatrix} \geq 0$.

$\hat{\alpha} = z(x(J)) = \pi.b$.

col (i) = indice de la variable de base associée à la i^{ème} ligne ; Aller en (1)

(1) : Soit s tel que : $\hat{c}^s = \underset{j}{Max}\, \hat{c}^j$

si $\hat{c}^s \leq 0$ terminer : J optimale ;

sinon aller en (2).

(2) Soit $I = \{i / \hat{A}_i^S > 0\}$
si $I = \phi$ terminer ; Z non borné supérieurement,
si $I \neq \phi$ aller à (3).

(3) Soit r tel que $\hat{b}_r / \hat{A}_r^s = Min \left\{ \hat{b}_i / \hat{A}_i^s , i \in I \right\}$.

Poser $J = J \cup \{s\} - col(r)$;
Poser col(r) = s ;

Ecrire (P) sous forme canonique par rapport à la nouvelle base ; (pour ce faire effectuer un pivotage autour de $\hat{A}_r^s$) .

Aller en (1).

Remarque [8] :
On dispose d'une autre variante de l'algorithme du simplexe, appelée algorithme révisé du simplexe, basée sur la remarque suivante :

Si on connaît une base réalisable de (P) on peut éviter d'effectuer un pivotage pour revenir à chaque fois à la forme canonique par rapport à la nouvelle base

On peut calculer directement :

$$\hat{A} = (A^J)^{-1} A$$
$$\hat{b} = (A^J)^{-1} b$$
$$\pi = C^J (A^J)^{-1}$$
$$\hat{C} = C - \pi A$$

Puis choisir s tel que $\hat{C}^s > 0$ et $\hat{A}^s = (A^J)^{-1} A^s$

Si $\hat{A}^s \leq 0$ le programme linéaire n'a pas de solution

Sinon soit $r = \underset{\{i / \hat{A}_i^s > 0\}}{Min} \left(\hat{b}_i \Big/ \hat{A}_i^s \right)$

Sans faire de pivotage on a la nouvelle base J' et il suffit de calculer $(A^{J'})^{-1}$ à chaque itération.

L'algorithme révisé du simplexe offre l'avantage de revenir à chaque fois aux données initiales du problème et évite de répéter les erreurs d'arrondi et dans certains problèmes particuliers le calcul de $(A^J)^{-1}$ est facile.

Méthode duale du simplexe :

Illustrons cette méthode par l'exemple suivant :

$$(P)\begin{cases} 2x_1 - x_2 + x_3 = -1 \\ -x_1 - x_2 + x_4 = -2 \\ -x_1 - 3x_2 = z(\max) \end{cases}$$

Ecrivons (P) sous forme canonique par rapport à la base $J = \{3,4\}$, on obtient l'écriture suivante :

	x_1	x_2	x_3	x_4	
x_3	2	-1	1	0	-1
x_4	-1	-1	0	1	-2
	-1	-3	0	0	0

Cette base (J) n'est pas réalisable (car $\hat{b} < 0$), par contre (J) est dite duale réalisable car $\hat{c} \leq 0$, et on sait que (J) serait optimale si $\hat{b} \geq 0$ et $\hat{c} \leq 0$.

- Dans l'algorithme du simplexe on démarre d'une base réalisable, on garde à chaque itération cette propriété et on essaye d'arriver à une base duale réalisable . Dans l'algorithme dual on a une base duale réalisable et on essaye d'arriver à une base réalisable de (P).

Revenons à notre exemple :

$\hat{b}_r = \min \hat{b}_i = -2$ Donc x_4 quitte la base

Soit $\frac{\hat{c}^s}{\hat{A}_2^s} = Min_{\hat{A}_2^j<0}\left\{\frac{\hat{c}_2^j}{\hat{A}_2^j}\right\} = Min\left(\frac{-1}{-1}, \frac{-3}{-1}\right) = 1$. Donc s = 1 : x_1 rentre dans la base.

0	-3	1	2	-5
1	1	0	-1	2
0	-2	0	-1	+2

↓

0	1	$-\frac{1}{3}$	$-\frac{2}{3}$	$\frac{5}{3}$
1	0	$\frac{1}{3}$	$-\frac{1}{3}$	$\frac{1}{3}$
0	0	$-\frac{2}{3}$	$-\frac{7}{3}$	$\frac{16}{3}$

$\hat{b}_r = \min \hat{b}_i = -5$

$\min_{A_1^j<0}\left\{\frac{\hat{c}^j}{A_1^j}\right\} = \frac{c^2}{A_1^2} = \frac{3}{2}$.

Donc la solution $\begin{pmatrix} \frac{1}{3} \\ \frac{5}{3} \\ 0 \\ 0 \end{pmatrix}$ .est optimale car elle est réalisable et $\hat{C} \le 0$: Z* = $-\frac{16}{3}$

Algorithme dual du simplexe :

$$\text{Soit } (P) = \begin{cases} Ax = b \\ cx = z(\max) \end{cases} x \ge 0.$$

Initialisation :

Soit J base duale réalisable, on écrit (P) sous forme canonique par rapport à J.

$$(\hat{P}) = \begin{cases} \hat{A}x = \hat{b} \\ \hat{c}x = z(\max) - \hat{\alpha} \end{cases}, \quad \text{avec : } \pi = c^J (A^J)^{-1}, \qquad \hat{c} = c - \pi A \le 0 \text{ ; } \hat{b} = (A^J)^{-1}.b \text{ ;}$$

$$\hat{A} = (A^J)^{-1} A \text{ ;} \qquad \hat{\alpha} = \pi b \text{ .}$$

Col(i) indice de la variable de base associée à la ligne (i).

Aller en (1) :

(1) Soit (r) tel que $\hat{b}_r = \min \hat{b}_i$.

- Si $\hat{b}_r \ge 0$ terminer (J) réalisable donc optimale car $\hat{c}_i \le 0$.
- Sinon aller à (2).

(2) Soit $L = \left\{ j / \hat{A}_r^j < 0 \right\}$

- Si $L = \phi$ terminer la 2ème ligne est irréalisable (car $0 \le \sum_j A_r^j x_j = b_r \le 0$ et $A_r^j \ge 0$: impossible).
- Sinon aller en (3) .

(3) Soit (s) tel que $\frac{\hat{c}^s}{A_r^s} = \min_{j \in L.} \left\{ \hat{c}^j \Big/ \hat{A}_r^j \right\}$.

Poser J= $J \cup s - col(r)$;

Poser col(r) = s ;

Effectuer un pivotage autour de $\hat{A}_r^s$ pour avoir la forme canonique par rapport à la nouvelle base.

Allez en (1) .

La méthode des deux phases :

Introduction : Pour initialiser l'algorithme du simplexe on doit avoir une base réalisable ce qui n'est pas toujours évident . La méthode des deux phases nous permet de régler ce problème.

$$\text{Soit } \mathbf{(I)} \begin{cases} Ax = b & A : mxn \\ x \ge 0 & b \ge 0 \end{cases}$$

$$A : R^n \to R^m et\ b \in R^m$$

Pour x donné dans IR^n définissons $V_i \ge 0$ tel que : $A_i x + V_i = b_i \quad i = 1,...,m$

Donc : $V_i = |b_i - A_i x|$. Par conséquent : $\sum V_i$ définit une distance dans IR^m . On peut chercher $Min \sum V_i = \psi$.

Le problème (I) admet une solution $\Leftrightarrow Min \sum V_i = 0$.

Donc pour résoudre le problème (I) il suffit de résoudre :

$$(P.A) = \begin{cases} A_1 x + v_1 = b_1 \\ A_2 x + v_2 = b_2 \quad x \ge 0 \\ ..... \\ A_m x + v_m = b_m \quad v \ge 0 \\ \sum v_i = \psi (Min) \end{cases}$$

(0,b) est une solution réalisable pour (P,A).

En plus : $\psi = \sum (v_i) \geq 0 \Rightarrow \psi$ borné inférieurement $\Rightarrow$ (P.A) admet une solution Optimale.

Soit (x*,v*) une solution optimale de (P.A) et ψ * la valeur qui lui correspond :

- Si $\psi^* > 0$ terminer (I) n'a pas de solution réalisable.
- Si $\psi = 0$ Alors (I) admet une solution réalisable.

Comment la trouver : Traitons un exemple simple :

$$(P)\begin{cases} 2x_1 - x_2 + x_3 = 1 \\ -x_1 + 3x_2 + x_3 = 2 \\ x_1 + 2x_2 + 2x_3 = 3 \\ 3x_1 + 4x_2 - 6x_3 = z(\max) \end{cases} \quad ; \quad (PA)\begin{cases} 2x_1 - x_2 + x_3 + v_1 = 1 \\ -x_1 + 3x_2 + x_3 + v_2 = 2 \\ x_1 + 2x_2 + 2x_3 + v_3 = 3 \\ -3x_1 - 4x_2 + 6x_3 + z = 0 \\ v_1 + v_2 + v_3 = w(\max) \end{cases}$$

x_1	x_2	x_3	z	v_1	v_2	v_3	
2	-1	1	0	1	0	0	1
-1	3	1	0	0	1	0	2
1	2	2	0	0	0	1	3
-3	-4	6	1	0	0	0	0
0	0	0	0	1	1	1	

↓

2	-1	1	0	1	0	0	1
-1	3	1	0	0	1	0	2
1	2	2	0	0	0	1	3
-3	-4	6	1	0	0	0	0
-2	-4	-4	0	0	0	0	-6

Forme canonique par rapport à (v_1, v_2, v_3)

↓

$\frac{5}{4}$	0	1	0	$\frac{3}{4}$	$\frac{1}{4}$	0	$\frac{5}{4}$
$-\frac{3}{4}$	1	0	0	$-\frac{1}{4}$	$\frac{1}{4}$	0	$\frac{1}{4}$
0	0	0	0	-1	-1	1	0
$-\frac{27}{2}$	0	0	1	$-\frac{11}{2}$	$-\frac{1}{2}$	0	$-\frac{13}{2}$
0	0	0	0	2	2	0	0

Donc $\psi^* = 0$ donc (P) admet une solution réalisable et (P) s'écrit :

$$(P) = \begin{cases} \frac{5}{4}x_1 + x_3 = \frac{5}{4} \\ -\frac{3}{4}x_1 + x_2 = \frac{1}{4} \\ -\frac{27}{2}x_1 = z(\max) + \frac{13}{2} \end{cases} \qquad x_i \geq 0$$ Fin de la 1ère phase.

Récapitulation :

Si $\psi^* = 0$ on obtient une base optimale (J) de P.A :

- Alors toutes les variables artificielles (qu'elles soient de base ou non). Sont nulles.
- Si $\forall i, v_i \notin J$ alors J est aussi une base réalisable de (P) on peut l'utiliser pour initialiser l'algorithme du simplexe.
- Si $\exists v_r$ tel que $v_r \in J$ comme $v_r = 0$ un pivotage sur $A_r^S (Si\ A_r^S \neq 0)$ permet de passer d'une base optimale à une base optimale, v_r quitte la base et est remplacer par x_S.
- Si un tel pivotage est impossible $(A_r^S = 0, \forall S)$ la ligne (r) était redondante, on la supprime du même coup on supprime v_r (on traitera de la même manière toutes les variables artificielles de base, soit par pivotage, soit en supprimant la ligne correspondante).
- à la fin on obtient une base réalisable de (P).

I-2) Programmation linéaire en nombres entiers :

Définition 1.9: Le programme linéaire :

$$(\text{P.E}) \begin{cases} \text{Ax} = \text{b} \\ Cx = Z(Max) \end{cases} \qquad x_j \in N$$

qui présente la particularité d'avoir la matrice A(m,n), le m-vecteur colonne (b) et le n-vecteur ligne (C) formés par des entiers est appelé : Programme linéaire en nombres entiers.

- Si on remplace la contrainte $x_j \in N \quad par \quad x_j \geq 0$ dans (P.E) on obtient le programme linéaire relaxé noté (P.R)

$$(\text{P.R}) : \begin{cases} Ax = b \\ Cx = Z(Max) \end{cases} \qquad x_j \geq 0$$

Soit (D) l'ensemble des solutions réalisables de (PR) et (S) celui de (PE). Si (D) est borné alors (S) est fini car tout sous-ensemble borné de IN est fini.

Méthode de résolution : on est tenté de résoudre le programme (P.R) et d'associer à la solution optimale obtenue les entiers les plus proches . Mais comme l'indique l'exemple suivant on est souvent loin de la solution optimale de (P.E).

$$(P.E_1)\begin{cases}10x_1+12x_2 \le 59\\ 10x_1+11x_2 = Z(Max)\end{cases} x_1, x_2 \in IN$$

Ce problème admet la solution optimal suivante : $x^* = (1,4); Z^* = 54$.

Le programme relaxé (PR_1) associé admet comme solution optimale $x' = (5,9;0)$ et $Z' = 59$.

- Si on prend les valeurs entières les plus proches de x' , on aura $x'' = (6,0)$ qui n'est même pas une solution réalisable de (P.E).
- Si on prend $x''' = (5,0)$ alors $Z''' = 50$ donc x''' est loin de la solution optimale de (PE_1) .

Les méthodes des coupes introduites par Gomory [8] figurent parmi les méthodes les plus répandues pour résoudre les programmes linéaires en nombres entiers.

Une coupe est une inéquation du type $ax \le d$ qui est vérifiée par tous les points de (S) mais pas par tous les points de (D) . L'idée est d'introduire une coupe à chaque itération du simplexe pour diminuer l'ensemble des solutions réalisables du (P.R) jusqu'à trouver une solution optimale entière.

Ecrivons (P.R) sous forme canonique par rapport à une base J .on obtient :

$$(\hat{P}R)\begin{cases}\hat{A}x = \hat{b}\\ \hat{C}x = Z(Max) - \hat{\alpha}\end{cases} x_j \ge 0$$

(J) est optimale si et seulement si : $(i):\hat{b}\geq 0 \quad ;ii):\hat{c}\leq 0$.

La solution optimale de (PR) est la solution de base associée à J . Donc $x_J=\hat{b}$ et $x_{\bar{J}}=0$, si cette solution optimale est entière (i-e,si $\hat{b}$ est entier) alors elle sera aussi optimale pour (P.E).

Résumé :

(J) sera une base optimale pour (P.E) si les trois conditions suivantes sont vérifiées : $i)\ \hat{b}\geq 0;\quad ii)\ \hat{c}\leq 0;\quad iii)\ \hat{b}$ entier.

Algorithme dual fractionnaire :

Définition 1.10:

On appelle partie entière d'un nombre réel a et on note $[a]$ le plus grand entier relatif inférieur ou égal à a.

Exemple:

$[2{,}15]=2,[-3{,}5]=-4,[4]=4,[-0{,}001]=-1$.

On appelle partie fractionnaire de a et on note $\langle a\rangle$ le nombre $\langle a\rangle=a-[a]$.

L'algorithme dual fractionnaire consiste à maintenir les conditions $i)$ et (ii) de(P.R) satisfaites et de chercher à réaliser (iii).

Pour ce faire on écrit (P.R) sous forme canonique par rapport à une base J en utilisant l'algorithme du simplexe, et on obtient à l'optimum :

$$(\hat{P}.R)\begin{cases}\hat{A}x=\hat{b}\\ \hat{C}x=Z(Max)-\hat{\alpha}\end{cases}$$

- Si $\hat{b}$ est entier, la solution optimale de (P.R) est optimale pour (P.E).
- Sinon soit r un indice tel que $\hat{b}_r$ n'est pas entier, donc $\langle\hat{b}_r\rangle$ est strictement positif.

On ajoute à $(\hat{P}.R)$ la contrainte * : $\sum_{j \notin J} \left\langle \hat{A}_r^j \right\rangle x_j \geq \left\langle \hat{b}_r \right\rangle$ qui s'écrit après adjonction d'une variable d'écart $y \geq 0$.

**. $-\sum_{j \notin J} \left\langle \hat{A}_r^j \right\rangle x_j + y = -\left\langle \hat{b}_r \right\rangle$. On dira que (**) est la coupe engendrée par la r^ième équation.

Remarque :

La somme de la ligne (r) de $(\hat{P}R)$ et (**) donne :

$$x_{col(r)} + \sum_{j \notin J} (\hat{A}_r^j - \left\langle \hat{A}_r^j \right\rangle) x_j + y = \hat{b}_r - \left\langle \hat{b}_r \right\rangle .$$

$\hat{A}_r^j - \left\langle \hat{A}_r^j \right\rangle$ et $\hat{b}_r - \left\langle \hat{b}_r \right\rangle$ sont tous deux entiers.

Le programme obtenu ainsi est écrit sous forme canonique par rapport à la base J' telle que :

$J' = J \cup \{indice\ de\ la\ variable\ d'écart\ y\}$ comme $\hat{c} \leq 0$ on applique à ce dernier l'algorithme dual du simplexe.

- Si ce programme n'a pas de solution réalisable, terminer (P.E) n'a pas de solution réalisable.
- Sinon on écrit ce programme sous forme canonique par rapport à une base optimale et on refait le même processus avec ce nouveau programme.

Algorithme :

Soit à résoudre (P.E) :

0) Résoudre le programme linéaire (P.R) par l'algorithme du simplexe.

- Si (P.R) n'a pas de solution, terminer (P.E) n'a pas de solution réalisable.

- Sinon : baptiser (P_1) le programme linéaire (P.R) écrit sous forme canonique par rapport à la base optimale J_1 ; poser k = 1.Aller en (1).

(1) Si la condition (iii) est vérifiée par (P_k) aller en (3).

Si la condition (iii) n'est pas vérifiée par (P_k), faire choisir un indice r tel que $\left\langle \hat{b}_r \right\rangle > 0$.

Introduire la contrainte $-\sum_{j \notin J} \left\langle \hat{A}_r^j \right\rangle x_j + x_{n+k} = -\left\langle \hat{b}_r \right\rangle$.

On obtient un programme linéaire (P_{k+1}) écrit sous forme canonique par rapport à la base $J_k \cup \{n+k\}$ aller en (2).

(2) résoudre (P_{k+1}) par l'algorithme dual du simplexe :

- Si (P_{k+1}) n'a pas de solution : Terminer (P.E) n'a pas de solution réalisable.
- Sinon écrire (P_{k+1}) sous forme canonique par rapport à une base optimale J_{k+1} ; poser $k = k+1$. Aller en (1).

(4) la solution de base de (k) relative à (P_k) est la solution optimale de (P.E).

Remarque :

Le temps d'exécution de l'algorithme dépend des coupes engendrées à chaque itération. C'est dans ce sens qu'on introduit la notion de coupe profonde pour diminuer le nombre d'itérations.

Définition 1.11:

On dit qu'une coupe (c) est plus profonde qu'une coupe (c') si $(D \cap c) \subset (D \cap c')$ (donc choisir (c) en premier est préférable car cela évitera au moins d'engendrer la coupe (c') et l'algorithme aura moins d'itérations.)

Bien qu'il n'existe pas de méthode systématique pour trouver la coupe la plus profonde, on peut engendrer par des algorithmes particuliers des coupes et choisir parmi celles-ci la plus profonde. C'est dans ce sens qu'on peut utiliser le théorème suivant :

Théorème 1.12:

- $\forall \alpha \in \Re$ l'inéquation : $\sum_{j \in J} \left(\left[\alpha \right] A_i^j - \left[\alpha A_i^j \right] \right) x_j \geq \left[\alpha \right] b - \left[\alpha b_i \right]$ est vérifiée par toute solution réalisable de (P.E) i-e, par tout élément de S.
- Pour certaines valeurs de (α) cette inéquation est une coupe . En particulier pour $\alpha = 1$.

Exemple :

$$(P.E_3) = \begin{cases} 5x_1 + x_2 \leq 20 \\ 2x_1 + 9x_2 \leq 29 \\ 3x_1 + 2x_2 = Z(Max) \end{cases}$$

Dans ce cas simple on peut représenter géométriquement l'ensemble (S) et les coupes engendrées par l'algorithme (fig 2).

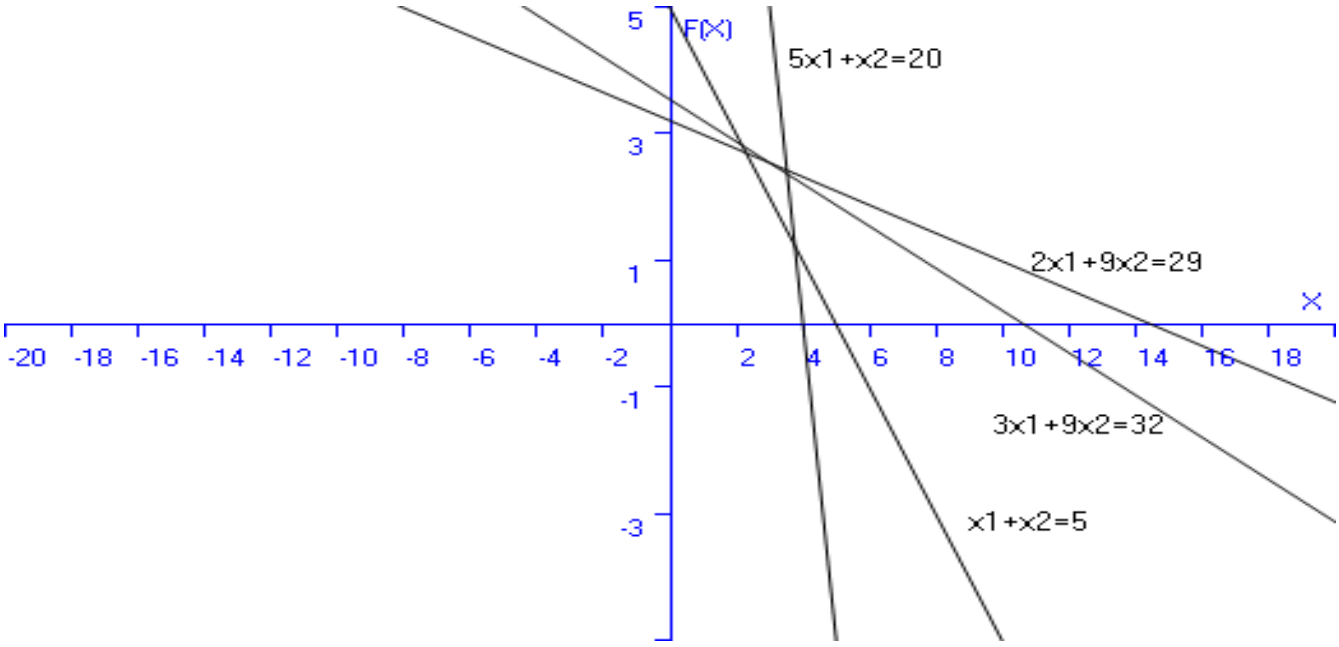


Figure (2)

L'application de l'algorithme de simplexe à $(P.R3)$ permet d'écrire celui-ci sous la forme :

$$(\hat{P}.R3)=\begin{cases} x_1+\frac{9}{43}x_3-\frac{1}{43}x_4=9+\frac{22}{43} \\ x_2-\frac{2}{43}x_3+\frac{5}{43}x_4=2+\frac{19}{43} \\ -\frac{23}{43}x_3-\frac{7}{43}x_4=Z(Max)-(15+\frac{18}{43}) \end{cases} \qquad x_j\geq 0 \quad j=1,2,3,4$$

La coupe engendrée par la 1ème équation est $c_1:\frac{9}{43}x_3+\frac{42}{43}x_4\geq\frac{22}{43}$ qui correspond à : $3x_1+9x_2\leq 32$

Au lieu de cette coupe, choisissons (h) entier tel que $(*):\left\langle h.\frac{9}{43}\right\rangle x_3+\left\langle h.\frac{42}{43}\right\rangle x_4\geq\left\langle h.\frac{22}{43}\right\rangle$ soit la plus profonde coupe possible. Dans ce cas simple on voit géométriquement que la coupe la plus profonde i-e, celle qui garde toutes les solutions entières et diminue le plus le domaine (D) est donnée par l'équation $x_1+x_2\leq 5$ en additionnant les deux contraintes de $(\hat{P}.R3)$ on a :

$$\left.\begin{array}{l} x_1+x_2+\frac{7}{43}x_3+\frac{4}{43}x_4=11+\frac{41}{43} \\ x_1+x_2\leq 5 \end{array}\right\} \Rightarrow \text{la coupe la plus profonde est } \frac{7}{43}x_3+\frac{4}{43}x_4\geq 6+\frac{41}{43}$$

Donc la coupe la plus profonde sera : $\frac{7}{43}x_3+\frac{4}{43}x_4\geq\frac{41}{43}$

Et le théorème nous permet d'écrire :

$\left\langle h\frac{9}{43}\right\rangle=\frac{7}{43}$ *et* $\left\langle h\frac{42}{43}\right\rangle=\frac{4}{43}$ *et* $\left\langle h\frac{22}{43}\right\rangle=\frac{41}{43}$ ce qui donne :h=39 Car :

$\left\langle\frac{39}{43}\text{x}9\right\rangle=\left\langle\frac{43x9}{43}-\frac{36}{43}\right\rangle=\left\langle 8+\frac{7}{43}\right\rangle=\frac{7}{43}$, $\left\langle 39x\frac{42}{43}\right\rangle=\left\langle 39-\frac{39}{43}\right\rangle=\left\langle 38+\frac{4}{43}\right\rangle=\frac{4}{43}$,

$\left\langle 39x\frac{22}{43}\right\rangle=\left\langle\frac{858}{43}\right\rangle=\left\langle 19+\frac{41}{43}\right\rangle=\frac{41}{43}$.

Au lieu d'introduire (C_1) on introduit $(C_2):\frac{7}{43}x_3+\frac{4}{43}x_4\geq\frac{41}{43}$ qui correspond à $x_1+x_2\leq 5$, sur le graphe on voit bien que (C_2) est plus profonde que (C_1).

On ajoute à $(\hat{P}.R3)$ la contrainte $-\frac{7}{43}x_3-\frac{4}{43}x_4+x_5=-\frac{41}{43} \quad x_5\geq 0$

Vérifions que deux itérations seulement suffisent pour résoudre $(P.E3)$.

x_1	x_2	x_3	x_4	x_5	b
1	0	$\frac{9}{43}$	$\frac{-1}{43}$	0	$3+\frac{22}{43}$
0	1	$\frac{-2}{43}$	$\frac{5}{43}$	0	$2+\frac{19}{43}$
0	0	$\frac{-7}{43}$	$\frac{-4}{43}$	1	$\frac{-41}{43}$
0	0	$\frac{-22}{43}$	$\frac{-7}{43}$	0	$-\left(15+\frac{18}{43}\right)$

→

x_1	x_2	x_3	x_4	x_5	b
1	0	$\frac{1}{4}$	0	$\frac{-1}{4}$	$3+\frac{3}{4}$
0	1	$\frac{-1}{4}$	0	$\frac{5}{4}$	$2-\frac{3}{4}$
0	0	$\frac{7}{4}$	1	$\frac{-43}{4}$	$\frac{41}{4}$
0	0	$\frac{-1}{4}$	0	$\frac{-7}{4}$	$-15+\frac{5}{4}$

On prend la première équation, on obtient la coupe $\frac{1}{4}x_3+\frac{3}{4}x_5\geq\frac{3}{4}$ en ajoutant une variable d'écart on obtient $-\frac{1}{4}x_3-\frac{3}{4}x_5+x_6=-\frac{3}{4}$.

x_1	x_2	x_3	x_4	x_5	x_6	b
1	0	$\frac{1}{4}$	0	$\frac{-1}{4}$	0	$\frac{15}{4}$
0	1	$\frac{-1}{4}$	0	$\frac{5}{4}$	0	$\frac{5}{4}$
0	0	$\frac{7}{4}$	1	$\frac{-43}{4}$	0	$\frac{41}{4}$
0	0	$\frac{-1}{4}$	0	$\frac{3}{4}$	1	$\frac{-3}{4}$
0	0	$\frac{-1}{4}$	0	$\frac{-7}{4}$	0	$-15+\frac{5}{4}$

→

x_1	x_2	x_3	x_4	x_5	x_6	b
1	0	0	0	-1	1	3
0	1	0	0	2	-1	2
0	0	0	1	-16	7	5
0	0	1	0	3	-4	3
0	0	0	0	1	-1	-13

Algorithme primal en nombres entiers :

Lorsque le problème relaxé (P.R) de (P.E) est posé sous forme telle que les conditions (i) et (iii) sont satisfaites nous développons une procédure qui nous permet de réaliser la condition (ii)tout en gardant les conditions (i) et (iii) satisfaites.

Soit (P.R) écrit sous forme canonique par rapport à une base (J) telle que :

- (J) réalisable (condition (i) satisfaite)
- Les coefficients de A, b, C restent entiers (condition (iii) satisfaite).

Examinons le déroulement de l'algorithme sur un exemple pour illustrer la façon de satisfaire la condition (ii) :

$$(P.E4)\begin{cases} x_1-4x_2\leq 0 \\ 3x_1+4x_2\leq 15 \qquad x_1,x_2\in N \\ 2x_1+x_2=Z(Max) \end{cases}$$

Le programme relaxé s'écrit sous la forme standard :

$$(P.R4)\begin{cases} x_1 - 4x_2 + x_3 = 0 \\ 3x_1 + 4x_2 + x_4 = 15 \\ 2x_1 + x_2 = Z(Max) \end{cases} \qquad x_j \geq 0, j = 1,2,3,4$$

On voit que $(P.R4)$ est écrit par rapport à la base $\{3,4\}$ qui est réalisable $(car\ \hat{b} \geq 0)$ et tous les coefficients sont entiers.

Si on veut dérouler l'algorithme du simplexe, le pivot qu'on doit choisir doit être non seulement positif mais aussi égal à (1) (si possible), pour ne pas engendrer des termes fractionnaires, sinon on doit introduire une coupe pour avoir cette propriété.

Dans $(P.R4)$ on voit que (x_1) possède un coût réduit égal à (2) donc elle est candidate à rentrer dans la base. A ce stade on doit pivoter autour du coefficient de (x_1) dans la 1ère équation qui est égal à (1), ce qui ne pose pas de problème d'intégrité. Après une itération $(P.R4)$ sera écrit sous forme canonique par rapport à la base $J = \{1,4\}$.

$$(P.R4)\begin{cases} x_1 - 4x_2 + x_3 = 0 \\ 16x_2 - 3x_3 + x_4 = 15 \\ 9x_2 - 2x_3 = Z(Max) \end{cases} \qquad x_j \geq 0, j = 1,2,3,4$$

Pour effectuer une deuxième itération du simplexe : il faut pivoter sur le coefficient de (x_2) dans la deuxième équation qui vaut (16) et cela va engendrer des termes fractionnaires et la condition (iii) ne sera plus satisfaite. Pour remédier à ce problème on applique la règle suivante :

Si le pivot $A_r^S > 1$ on introduit une coupe correspondant à la ligne pivot (r) de la forme :

$\sum_{j \notin J} [h]A_r^j - [hA_r^j]x_j \geq [h]b_r - [hb_r]$ Si on prend $h = \frac{1}{A_r^S}; \hat{c}\ A_r^S > 1 \Rightarrow [h] = 0$ et on aboutit à la coupe $(*) : -\sum_{j \notin J} \left[\frac{A_r^j}{A_r^S} \right] x_j \geq -\left[\frac{b_r}{A_r^S} \right]$.

Les coefficients de cette contrainte sont entiers car ce sont des parties entières.
En multipliant (*) par (-1) et en introduisant une variable d'écart on obtient :

$(**) : \sum_{j \notin J} \left[\frac{A_r^j}{A_r^s} \right] x_j + y = \left[\frac{b_r}{A_r^s} \right]$ $y \geq 0$. On ajoute la contrainte (**) à (P.R) et on effectue dans le nouveau programme un pivotage sur le coefficient de (x_s) qui vaut (1) dans la contrainte ajoutée.

Revenons à notre exemple : $A_2^2 = 16$. On introduit la coupe correspondante à la 2ième équation .
Le nouveau programme est :

$$\begin{cases} x_1 - 4x_2 + x_3 = 0 \\ 16x_2 - 3x_3 + x_4 = 15 \\ x_2 - x_3 + x_5 = 0 \\ 9x_2 - 2x_3 = Z(Max) \end{cases} \quad x_j \geq 0; j = 1,2,3,4,5$$

En pivotant autour de (x_2) dans la 3ième équation et en ajoutant la coupe :
$x_3 - 2x_5 + x_6 = 1$. On aura :

$$\begin{cases} x_1 - 3x_3 + 4x_5 = 0 \\ 13x_3 + x_4 + 6x_5 = 15 \\ x_2 - x_3 + x_5 = 0 \\ x_3 - 2x_5 + x_6 = 1 \\ 7x_3 - 9x_5 = Z(Max) \end{cases}$$

$A_r^S = 13, r = 2, S = 3$

$$\begin{cases} x_1 - 2x_5 + 3x_6 = 3 \\ x_4 + 10x_5 - 13x_6 = 2 \\ x_2 - x_5 + x_6 = 1 \\ x_3 - 2x_5 + x_6 = 1 \\ 5x_5 - 7x_6 = Z(Max) - 7 \end{cases}$$

$A_r^S = 10 ; r = 2 ; S = 5$

La coupe correspondante est : $x_5 - 2x_6 + x_7 = 0 \quad x_7 \geq 0$

$$\begin{cases} x_1 - 2x_5 + 3x_6 = 3 \\ x_4 + 10x_5 - 13x_6 = 2 \\ x_2 - x_5 + x_6 = 1 \\ x_3 - 2x_5 + x_6 = 1 \\ x_5 - 2x_6 + x_7 = 1 \\ 5x_5 - 7x_6 = Z(Max) - 7 \end{cases}$$

$$\begin{cases} x_1 - x_6 + 2x_7 = 3 \\ x_4 + 7x_6 - 10x_7 = 2 \\ x_2 - x_6 + x_7 = 1 \\ x_3 - 3x_6 + 2x_7 = 1 \\ x_5 - 2x_6 + x_7 = 0 \\ x_6 - 2x_7 + x_8 = 0 \\ 3x_6 - 5x_7 = Z(Max) - 7 \end{cases}$$

$$\begin{cases} x_1 + x_8 = 3 \\ x_4 + 4x_7 - 7x_8 = 2 \\ x_2 - x_7 + x_8 = 1 \\ x_3 - 4x_7 3x_8 = 1 \\ x_5 - 3x_7 + 2x_8 = 0 \\ x_6 - 2x_7 + x_8 = 0 \\ x_7 - 2x_8 + x_9 = 0 \\ x_7 - 3x_8 = Z(Max) - 7 \end{cases}$$

$$\begin{cases} x_1 = 3 \\ x_4 + x_8 - 4x_9 = 2 \\ x_2 - x_8 + x_9 = 1 \\ x_3 - 5x_8 + 4x_9 = 1 \\ x_5 - 4x_8 + 3x_9 = 0 \\ x_6 - 3x_8 + 2x_9 = 0 \\ x_7 - 2x_8 + x_9 = 0 \\ - x_8 - x_9 = ZMax) - 7 \end{cases}$$

A cette itération on voit que la condition (iii) est vérifiée, donc on a trouvé une solution optimale.

Algorithme :

Soit à résoudre (P.E).

(0) Baptiser (P_0) le programme linéaire (P.R) écrit sous forme canonique par rapport à une base réalisable (J) telle que tous les coefficients soient entiers.

Poser $J_0 = J; k = 0$

(1) Si $Max\, c^i \leq 0$ aller en (3)

Sinon aller en (2)

(2) Choisir S tel que $c^s > 0$; déterminer r tel que : $\frac{b_r}{A_r^s} = Min_i \left\{ \frac{b_i}{A_i^s}; A_i^s > 0 \right\}$

- Si $A_r^s = 1$ effectuer un pivotage autour de A_r^s poser $J_{k+1} = J_k \cup \{s\} - \{col_{(r)}\}$.

- Sinon on ajoute à (P_k) la contrainte :

$$\sum_{j \in J} \left[\frac{A_r^j}{A_r^s} \right] x_j + y = \left[\frac{b_r}{A_r^s} \right]; \qquad y \geq 0$$

On obtient un programme linéaire (P_{k+1}) écrit sous forme canonique par rapport à la base réalisable $J_k \cup \{indice\ de\ la\ \text{var}iable\ d'écart\}$.

- Effectuer dans (P_{k+1}) un pivotage autour du coefficient de (x_s) dans la contrainte ajoutée, cette opération permet d'écrire (P_{k+1}) sous forme canonique par rapport à une base réalisable J_{k+1}.

Poser $k = k + 1$,

Aller en (1)

(3) la solution de base de (P_k) relative à J_k est une solution optimale de (P.E).

Remarque :

On peut supprimer la contrainte relative à une variable d'écart (y) si celle-ci est de nouveau candidate à rentrer dans la base.

I-3) Problème de transport

I) Définition 1.13:

On dispose de (m) entrepôts et (n) points de vente qu'on doit alimenter à partir des entrepôts . La quantité x_{ij} est acheminée de l'entrepôt (i) au point de vente (j) , $a_1,...,a_m$ représentent les quantités disponibles dans chaque entrepôt et $b_1,...,b_n$ les quantités demandées par chaque point de vente.

- On voit que le problème admet une solution si $\sum_{i=1}^{m} a_i \geq \sum_{j=1}^{n} b_j$.

- Si c_{ij} représente le coût de transport d'une unité de marchandise de l'entrepôt (i) vers le point de vente (j) alors notre programme de transport s'écrit :

$$(P.T)\begin{cases} \sum_{j=1}^{n} x_{ij} \leq a_i \\ \sum_{i=1}^{m} x_{ij} \geq b_j \qquad\qquad x_{ij} \geq 0 \\ \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij} = Z(Min) \end{cases}$$

Remarque :

1. On peut poser $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$ sinon on ajoute un point de vente fictif auquel on affecte la demande $b_0 = \sum_{1}^{m} a_i - \sum_{1}^{n} b_j$ avec un coût de transport $c_{i0} = 0; \forall i = 1,....,m$.
2. La condition $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$ entraîne directement $\sum_{j=1}^{n} x_{ij} = a_i \ \forall i \ et \sum_{i=1}^{m} x_{ij} = b_j \ \forall j$.

Démonstration :

$b_j \le \sum_{i=1}^{m} x_{ij}$ supposons que pour un indice j on a $b_j < \sum_{i=1}^{m} x_{ij}$ alors on a :

$\sum_{j=1}^{n} b_j < \sum_{j=1}^{n}\sum_{i=1}^{m} x_{ij} = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} \le \sum_{i=1}^{m} a_i$. Contradiction car $\sum_{j=1}^{n} b_j = \sum_{i=1}^{m} a_i$

Donc $b_j = \sum_{i=1}^{m} x_{ij} \quad j = 1,...,n$

De même on voit que $a_i = \sum_{i=1}^{m} x_{ij}$.

3. Si $a_i = 0$ donc l'entrepôt (i) ne peut alimenter aucun entrepôt, donc on peut le supprimer.

   Si $b_j = 0$ le point de vente n'a aucune demande, donc on peut le supprimer.

En définitive tout problème de transport s'écrit sous la forme :

$$(T)\begin{cases} \sum_{j=1}^{n} x_{ij} = a_i \quad i = 1,...,m \\ \sum_{i=1}^{m} x_{ij} = b_j \quad j = 1,...,n \quad x_{ij} \ge 0 \\ \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij} = Z(Min) \end{cases} \quad \text{Avec} \quad \begin{array}{l} a_i > 0 : \ i = 1,...,m \\ b_j > 0 : \ j = 1,...,n \\ c_{ij} \ge 0 \end{array}$$

On peut écrire ce problème de transport sous la forme :

$$(\hat{T})\begin{cases} Ax = f \\ Cx = Z(Min) \end{cases} \quad x \ge 0$$

$A : (m+n, mn); f = \begin{pmatrix} a \\ b \end{pmatrix} \in \Re^{m+n}$

$C : m.n$ vecteur ligne et $x : m.n$ vecteur colonne

Si on examine les contraintes du programme (T) :

$$\begin{cases} \sum_{j=1}^{n} x_{ij} = a_i ; i = 1,...,m \\ \sum_{i=1}^{m} x_{ij} = b_j ; j = 1,...,n \end{cases} \qquad x_{ij} \geq 0$$

On peut déduire la structure de la matrice A dont les éléments sont des (0) ou des (1). Chaque variable x_{ij} intervient deux fois et deux seulement ; une fois dans les (m) premières équations et une fois dans les (n) dernières. Donc toute colonne de (A) possède deux composantes différentes de (0) (qui sont égales à 1).

Si on pose : $\mathbf{1}=[1,...,1], \mathbf{0}=[0,...,0]$ des n-vecteur lignes et $\mathbf{1}_k$ le ($m.n$) vecteur ligne tel que :

$\mathbf{1}_k = (\alpha_{11}, \alpha_{12}, ..., \alpha_{1n}, \alpha_{21}, \alpha_{22}, ..., \alpha_{2n}, ..., \alpha_{i1}, \alpha_{i2}, ..., \alpha_{in}, ..., \alpha_{m1}, \alpha_{m2}, ..., \alpha_{mn})$ et

$$\alpha_{ij} = \begin{cases} 1 & si\ j = k \\ 0 & si\ j \neq k \end{cases} \quad i = 1,...,n$$

Soit $A' = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 \end{pmatrix}$ et soit $A'' = \begin{pmatrix} 1_1 \\ 1_2 \\ \vdots \\ 1_j \\ \vdots \\ 1_n \end{pmatrix}$ alors : $A = \begin{pmatrix} A' \\ A'' \end{pmatrix}$.

Introduisons selon la définition récursive suivante la notion modifiée d'une matrice triangulaire [8]:

Définition 1.14:

On dit qu'une matrice carrée non singulière B est ´´ triangulaire´´ si :

i) un scalaire non nul est une matrice ´´triangulaire´´ d'ordre (1).

ii) B possède une ligne ayant exactement un élément non nul, et la sous matrice obtenue à partir de B en supprimant cette ligne et la colonne de cet élément est ´´triangulaire´´ (au sens modifié).

Exemple :

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 0 & 0 \\ 5 & 6 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 3 \\ 6 & 0 \end{bmatrix} \rightarrow [3]$$

Définition 1.15:

Une matrice $A(mxn)$ est dite totalement unimodulaire (noté t-u) si toute sous-matrice carrée de A admet un déterminant égal à : $+1, -1, 0$.

Remarque :

$$\text{Soit } (p)\begin{cases} Ax = b \\ cx = Z(\max) \end{cases} \quad x \geq 0$$

Où A est t-u. Si b et A sont à valeurs entières alors pour toute base J la solution de base associée à J est à composantes entières.

Théorème 1.16:

Soit B une matrice carrée non singulière extraite de A, alors B est ´´triangulaire´´ au sens de la définition.

Corollaire 1.17:

La matrice du problème de transport est t-u.

D'après ce corollaire si la matrice A et le vecteur b du problème de transport sont à valeurs entières alors toutes les solutions seront entières. Donc résoudre un problème de transport en nombres entiers est équivalent à résoudre son problème relaxé.

Théorème 1.18:

Le rang de la matrice (A) des contraintes de $(\hat{T})$ est égal à $(m + n - 1)$.

Remarque :

1) Le concept de matrice triangulaire défini ici coïncide avec celui des matrices triangulaires classiques à une permutation prés des lignes et des colonnes.

2) Si (B) est une matrice ´´triangulaire´´ alors le système (S) : $Bx = f$, se résout par substitution de la manière suivante :

Soit (i) une ligne de (B) ne contenant qu'un élément non nul disons B_i^j , on a

donc : $x_{ij} = f_i / B_i^j$. En remplaçant (x_{ij}) par sa valeur dans les autres équations, on a encore un système de type (S) (i-e, un système dont la matrice est ´´triangulaire´´) à résoudre mais la dimension a diminué d'une unité.

3) La résolution de $(\hat{T})$ se fait par l'algorithme révisé du simplexe, compte tenu de la propriété de triangularité des matrices B, pour calculer :

-Le vecteur multiplicatif : on résout $\pi B = c^B$.

-La solution de base associée : on résout $B\begin{pmatrix}\hat{a}\\ b\end{pmatrix}=\begin{pmatrix}a\\ b\end{pmatrix}$.

-La colonne candidate : on résout $B\hat{A}^S = A^S$.

Ces systèmes sont faciles à résoudre, car l'utilisation de la remarque (2) nous évite d'inverser (B) à chaque itération.

Résolution de problème de transport :

En considérant le problème dual du problème de transport on déduit le théorème suivant :

Théorème 1.19:

Une C.N.S pour qu'une solution réalisable (x) de (T) soit optimale est qu'on puisse trouver : $u_1, u_2, ..., u_m$ et $v_1, v_2, ..., v_n$ tel que :

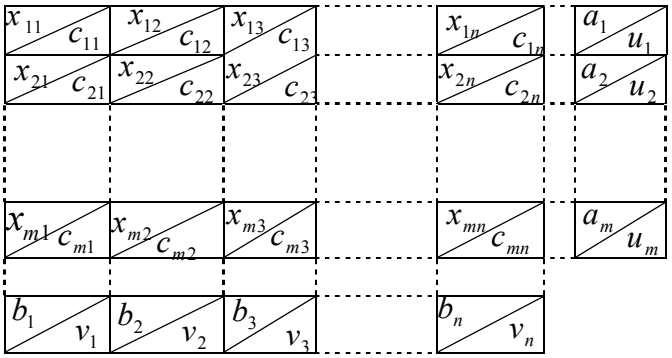
(1) $u_i + v_j \le c_{ij} \quad i = 1,...,m \; et \; j = 1,...,n$

(2) $u_i + v_j < c_{ij} \Rightarrow x_{ij} = 0 \quad i = 1,...,m \; et \; j = 1,...,n$

Remarque :

Si u_i et v_j , qui sont en fait les variables duales, satisfont les contraintes (1) et (2) alors il en est de même pour $u_i - \alpha$ et $v_j + \alpha$ cela est dû au fait que toute contrainte du problème de transport est redondante, aussi pour chercher les u_i et v_j on posera systématiquement $u_1 = 0$ et le théorème précédent nous donnera une solution plus rapidement qu'en utilisant l'algorithme du simplexe.

-On peut représenter un problème de transport (T) par le tableau suivant :



Chaque case de ce tableau représente une colonne de la matrice A.

Une ligne représente une contrainte parmi les (m) premières contraintes de (T) et une colonne représente une contrainte parmi les (n) dernières de (T).

Le problème consiste à trouver $x_{ij}, u_i \; et \; v_j$ tels que :

- La somme des x_{ij} sur la i^{ème} ligne soit égale à u_i.
- La somme des x_{ij} sur la j^{ème} colonne soit égale à v_j.
- u_i et v_j vérifient (1) et (2) du théorème (1.19).

Organigramme [8]:

I- Recherche d'une base réalisable (J) de départ et la solution de base qui lui est associée :

(J) désignera l'ensemble des couples (i, j) appartenant à la base, donc : $|J| = m + n - 1$.

(1) Si $m = 1 \; et \; n \geq 1 \; alors : J = \{(1,1),(1,2),...,(1,n)\}$; $x_{11} = b_1, x_{1,2} = b_2, ..., x_{1n} = b_n$.

(2) Si $m \geq 1 \; et \; n = 1 \; alors \; J = \{(1,1),(2,1),...,(m,1)\}, x_{11} = a_1, x_{21} = a_2, ..., x_{m1} = a_m$

(3) Si m>1 et n>1 on choisit une case du tableau (T) par exemple la case (i,j) telle que $c_{ij} = Min_{\{(r,S)\}} \; c_{r,S}$ (*).

*« Ce choix n'est pas obligatoire mais il consiste à prendre d'abord les liaisons de faible coût ce qui donne en moyenne une diminution du nombre d'itérations ».

On pose alors $J = J \cup \{(i, j)\} \; et \; x_{ij} = Min\{a_i, b_j\}$.

(ie : La quantité maximum qu'on puisse attribuer à x_{ij} tout en réalisant les autres contraintes.)

Si : $a_i \le b_j$ (resp $a_i > b_j$) on aura $x_{ij} = a_i$ (resp $x_{ij} = b_j$)

Et $x_{ij'} = 0.((resp\ x_{i'j} = 0))$.pour $j \neq j'$ (resp pour $i \neq i'$).

La i^{ème} ligne (resp la j^{ème} colonne) de (T) est saturée.

On peut considérer maintenant qu'il nous reste à chercher une solution réalisable d'un nouveau problème de transport déduit du premier en supprimant la i^{ème} ligne (resp la j^{ème} colonne) et en remplaçant b_j (resp a_i) par $b_j - a_i$ (resp par $a_i - b_j$).

II-Procédure de calcul des multiplicateurs :

Déterminer le vecteur multiplicatif de la base J revient à résoudre le système $(S): \pi A^J = c^J$ où π représente le (m+n) vecteur ligne $(u_1, u_2, ..., u_m, v_1, v_2, ..., v_n)$.Comme la matrice A n'est pas de plein rang $(rg(A) = m + n - 1 < m + n)$ et admet une solution réalisable, donc l'ensemble des solutions est défini à une constante prés .Aussi on peut systématiquement poser $u_1 = 0$ .Le système (S) devient $S': u_i + v_j = c_{ij}, \forall (i, j) \in J$,et ce système se résout de proche en proche. Les nouveaux coûts relatifs à la base J sont $\hat{c}_{ij} = c_{ij} - (u_i + v_j)$.

Soit (i', j') un couple hors base tel que : $\delta = c_{i'j'} - u_{i'} - v_{j'} = Min_{(i,j)} \left| c_{ij} - u_i - v_j \right|$

- Si $\delta \ge 0$ alors J est optimale.
- Sinon le couple (i', j') rentre dans la base.

III-Changement de la solution et détermination d'une nouvelle base :

Posant $J' = J \cup \{(i', j')\}$ et appliquons la procédure décrite dans (I) jusqu'à ce que le tableau réduit obtenu ne contienne aucune case de J' seule dans ligne ou seule dans sa colonne. Marquer + la case (i', j') ,les autres cases sont non marquées.

Tant qu'il existe une ligne ou une colonne contenant une case non marquée et une case marquée, marquer la case non marquée du signe opposé de celui de la case marquée.

Fin tant que.

Poser $\theta = \underset{\{(i,j)/(i,j)est\ marqué\ -\}}{Min}\left\lfloor x_{ij} \right\rfloor$

Soit $(\hat{i},\hat{j})$ une case marquée – telle que $x_{\hat{i}\hat{j}} = \theta$

La nouvelle base est $\hat{J} = J \cup \left\{(\hat{i},\hat{j})\right\} - \left\{(i',j')\right\}$

Aller à la procédure de calcul des multiplicateurs (ie: les variables duales) avec $J = \hat{J}$.

Problème d'affectation :

(n) personnes sont à affecter à (n) travaux, chaque personne devant effectuer un travail et un seul, et chaque travail doit être fait par une personne et une seule.
Le coût de l'affectation de la i^{iem} personne au j^{ieme} travail est c_{ij}, le problème consiste à affecter les personnes aux travaux de sorte que la somme des coûts soit minimale.
Ce problème peut se formuler comme un programme linéaire en nombres entiers

$$x_{ij} = \begin{cases} 1 & \text{Si la i}^{eme}\text{ personne est affectée au j}^{ieme}\text{ travail} \\ 0 & \text{Sinon} \end{cases}$$

$$\text{On a alors : } A = \begin{cases} \sum_{i=1}^{n} x_{ij} = 1 \quad j = 1,...,n \\ \sum_{j=1}^{n} x_{ij} = 1 \quad i = 1,...,n \quad x_{ij} \in \{0,1\} \\ \sum_{j=1}^{n}\sum_{i=1}^{n} c_{ij} x_{ij} = z(Min) \end{cases}$$

Soit $(\overline{A})$ le (P.L) obtenu à partir de (A), en relaxant les contraintes d'intégrité sur les variables x_{ij} :

$$(\overline{A}) \begin{cases} \sum_{i=1}^{n} x_{ij} = 1 \quad j = \overline{1,n} \\ \sum_{j=1}^{n} x_{ij} = 1 \quad i = \overline{1,n} \quad x_{ij} \geq 0 \\ \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij} = Z(Min) \end{cases}$$

Toute solution optimale de base $(\overline{A})$ est une solution de (A). En effet $\overline{A}$ est un problème de transport, sa matrice est totalement unimodulaire, i-e, toute sous matrice de A non singulière aura un déterminant égale à 1 ou -1. Par conséquent toute solution de base $\overline{A}$ est entière(quand les seconds membre sont entiers).

Comme : $x_{ij} \geq 0$ et $\sum_{i=1}^{n} x_{ij} = 1$ on a nécessairement $0 \leq x_{ij} \leq 1$ et les seules valeurs entières de l'intervalle $[0,1]$ étant 0 ou 1, par conséquent résoudre A revient à résoudre $\overline{A}$.

Le Programme linéaire $(\overline{A})$ est aussi appelé "problème d'affectation" .

Remarque : Pour toute solution de base du problème d'affectation on aura 2n-1 variables de base et seulement n variables égales à 1(les autres étant nulles), le problème d'affectation sera donc très dégénéré, aussi on se gardera lors de la résolution de conserver à chaque itération une base i-e, de ne faire sortir qu'une variable à la fois.

I-4) Optimisation multicritère :

Un programme multicritère consiste à optimiser plusieurs fonctions objectif soumises à un même ensemble de contraintes, on peut le présenter de la manière suivante :

$$\begin{cases} Max \quad f_1(x) = Z_1 \\ Max \quad f_2(x) = Z_2 \\ \quad\quad \vdots \\ Max \quad f_k(x) = Z_k \\ \\ g_1(x) \le b_1 \\ g_2(x) \le b_2 \qquad\qquad x = (x_1, x_2, ..., x_n) \, ; x_i \ge 0 \\ \quad\quad \vdots \\ g_m(x) \le b_m \end{cases}$$

$f_i(x)$: est la fonction objectif du i^{ème} critère, $i = 1,...,k$.
$g_j(x)$ représente la j^{ème} contrainte. $j = 1,..., m$.
Soit $S = \left\{ x \in IR^n \,/\, g_i(x) \le b_i, i = 1,2,...,m \right\}$ alors (S) est l'ensemble des solutions réalisables.

Définition 1.20:

Soient deux vecteurs Z^1, Z^2 appartenant à R^k ,on dit que :
Z^1 domine Z^2 si $Z_i^1 \ge Z_i^2 ; \forall i$ et il existe j tel que $Z_j^1 > Z_j^2$.

Définition 1.21:

On dit que $\overline{x} \in S$ est une solution efficace si $\forall x \in S \quad f(\overline{x})$ n'est pas dominée par $f(x)$ où f est le vecteur critère : $f = (f_i)_{i=\overline{1,k}}$.

Dans le cas particulier où les fonctions $f_i(x)$ et $g_i(x)$ sont linéaires c'est-à-dire de la forme :

$$f_i(x) = c_i^1 x_1 + c_i^2 x_2 + ... + c_i^n x_n \qquad i = 1,2,...,k$$
$$g_j(x) = a_j^1 x_1 + a_j^2 x_2 + ... + a_j^n x_n \qquad j = 1,2,...,m$$

Alors le programme linéaire multicritère s'écrit sous la forme suivante :

$$\begin{cases} Ax \le b \\ Cx = Z(M\text{ ax}) \end{cases} \quad x = (x_1, x_2, .., x_n)\,, x_i \ge 0\,.$$

- A est la matrice (m,n) de terme a_i^j.
- C la matrice (k,n) de terme C_i^j.
- b le vecteur colonne : $(b_1, b_2, ..., b_m)^T$.

La fonction d'utilité :

Etant donné qu'un programme multicritère admet rarement des solutions qui optimisent simultanément tous les critères, on définit une fonction U dite fonction d'utilité telle que :

$$U : IR^k \to IR \quad .$$

Cette fonction va exprimer une mesure de l'importance ou de la préférence de chaque critère par rapport à l'ensemble des autres du point de vue du décideur. Ainsi le programme devient monocritère.

Par exemple le programme multicritère linéaire précédent devient :

$$MAx\left\{u(z) / z = cx, x \in S\right\}$$

Pratiquement il est souvent impossible de construire mathématiquement une fonction d'utilité à cause déjà de la subjectivité de la notion d''importance'' ou de ''préférence''. En outre on peut rarement appliquer les méthodes de la programmation linéaire car la fonction d'utilité sera généralement non linéaire .

Chapitre II

Problèmes de flots

II-1) Problème de plus court chemin

Définition 2.1: Un réseau $R = (X, U, d)$ est défini par un graphe $G = (X, U)$ et une application $d : U \to R$. $d(u)$ est appelée longueur ou coût de l'arc u.

Définition 2.2: La ''longueur'' d'un chemin (c) dans le réseau $R = (X, U, d)$ est égale à la somme des ''longueurs'' des arcs (comptées avec leur ordre de multiplicité dans c) qui constitue le chemin. $l(c) = \sum_{u \in c} d(u)$.

Notations :

Si $u = (x, y) \in U$ alors on note $I(u) = x$ et $T(u) = y$

Si $A \subset X$ on note :

$W^+(A) = \{u \in U / I(u) \in A\}; W^-(A) = \{u \in U / T(u) \in A\}$ et $W(A) = W^+(A) \cup W^-(A)$.

Soit $W \subset U$ alors s'il existe $A \subset X$ tel que $W = W(A)$ on dit que W est un cocycle.

On appelle la matrice des distances du réseau $R = (X, U, d)$ la matrice D dans l'élément d_{ij} se définit par :

$$d_{ij} = \begin{cases} d_{ij} & si \quad (x_i, x_j) \in U \\ +\infty & si \quad (x_i, x_j) \notin U \\ 0 & si \quad i = j \end{cases}$$

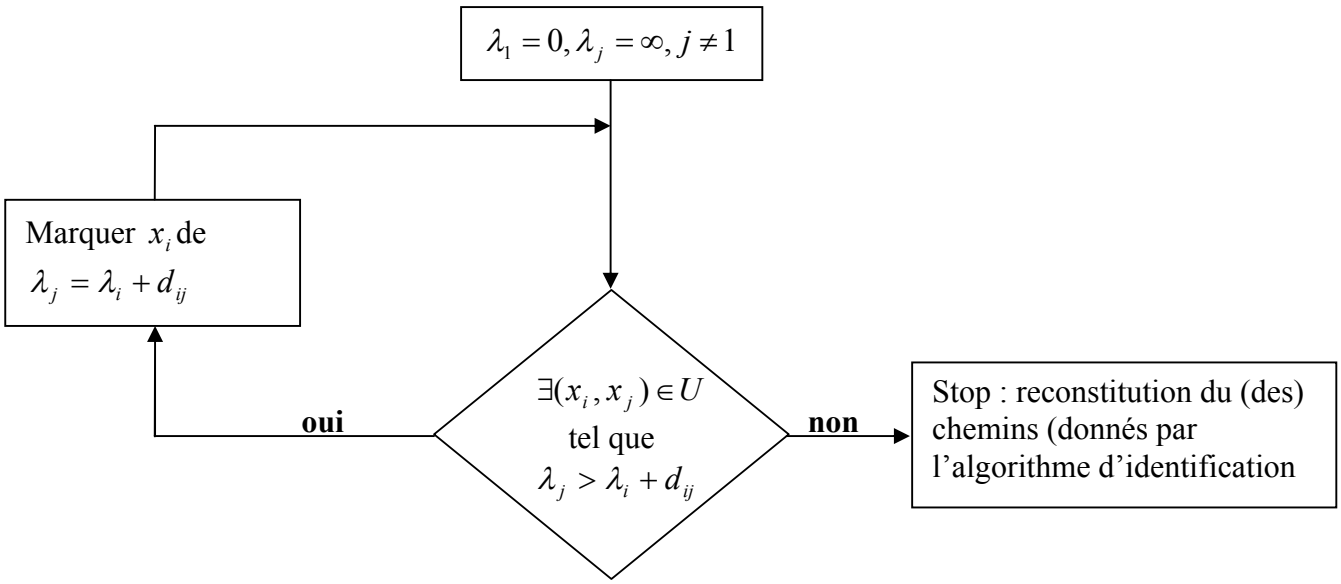
Algorithmes de recherche de plus courts chemins [3]:

On pose $X = \{x_1, x_2, ..., x_n\}$ et on se propose de trouver un plus court chemin de x_1 à x_n.

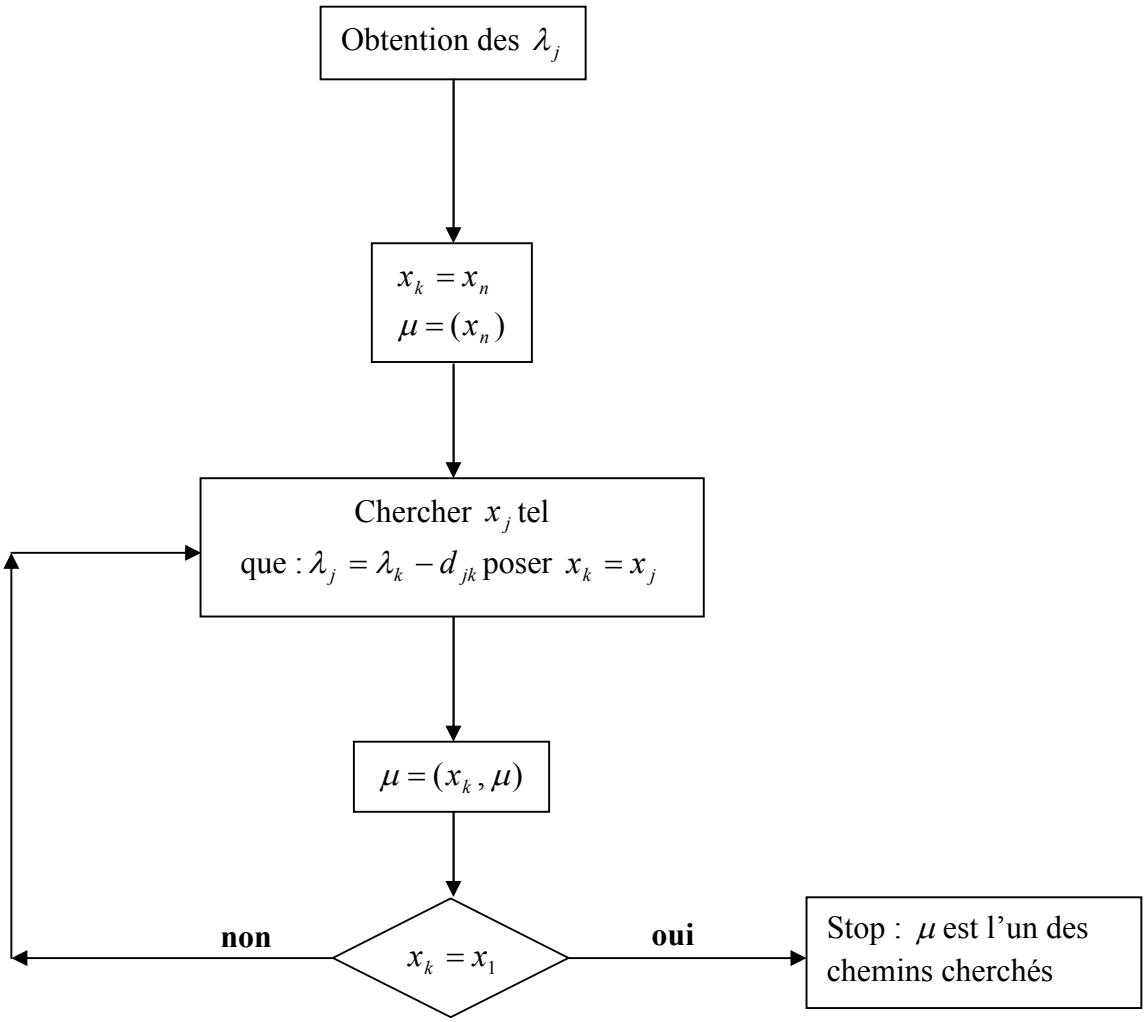
Pour que le problème admette une solution de longueur finie nous supposons que le graphe ne comporte aucun circuit de longueur négative (dit circuit absorbant),et qu'il existe au moins un chemin de x_1 à x_n.

Algorithme général (Algorithme de Ford) :

Bien qu'il offre l'avantage d'être utilisé sans aucune restriction ni sur les signes des longueurs ou la présence d'un circuit, il est très peu utilisé pour son manque d'efficacité.

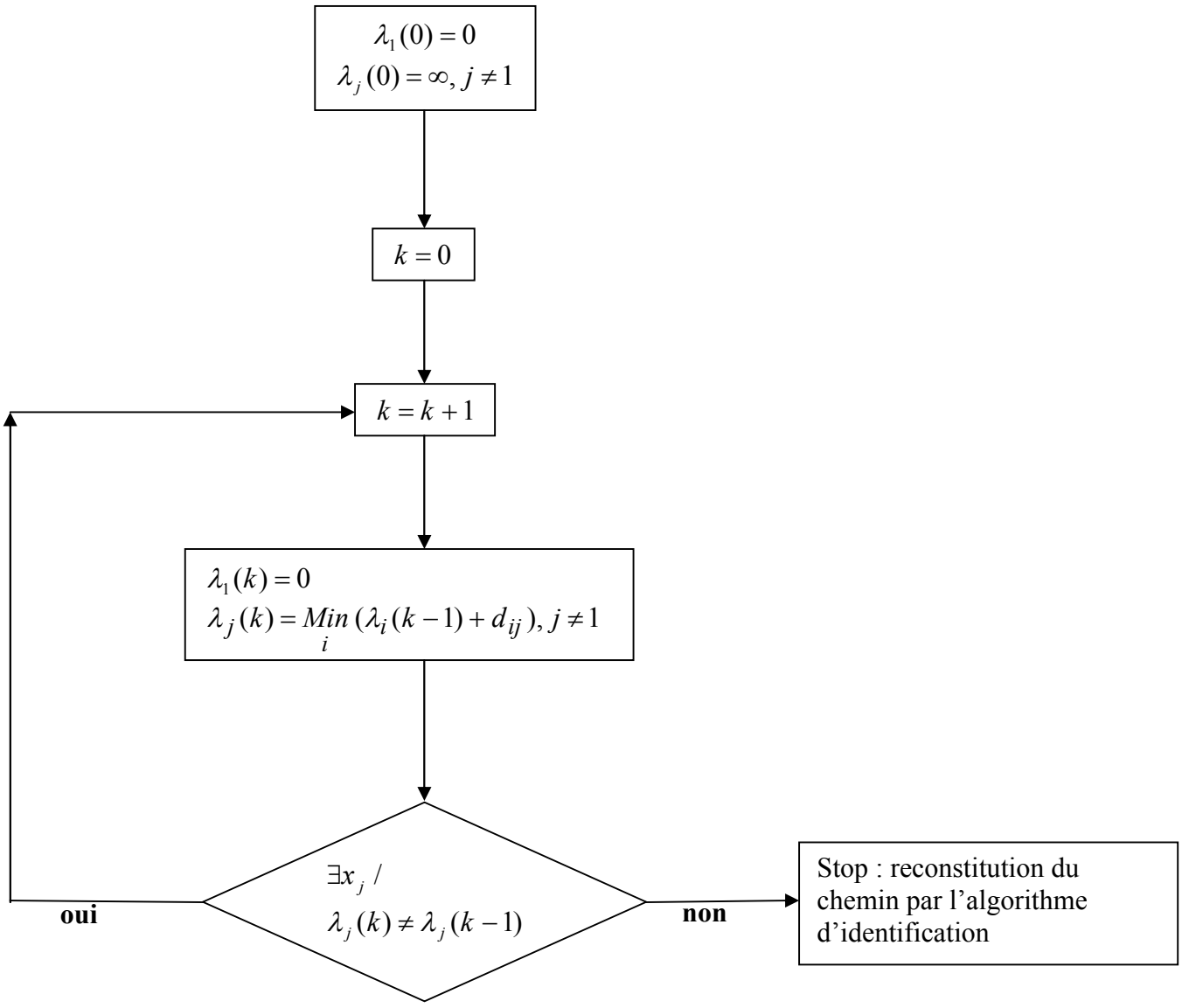


Algorithme d'identification :
Les chemins recherchés se reconstituent à reculons à partir de x_n ,par juxtaposition d'arcs (x_j, x_k) tel que $\lambda_j = \lambda_k - d_{kj}$.



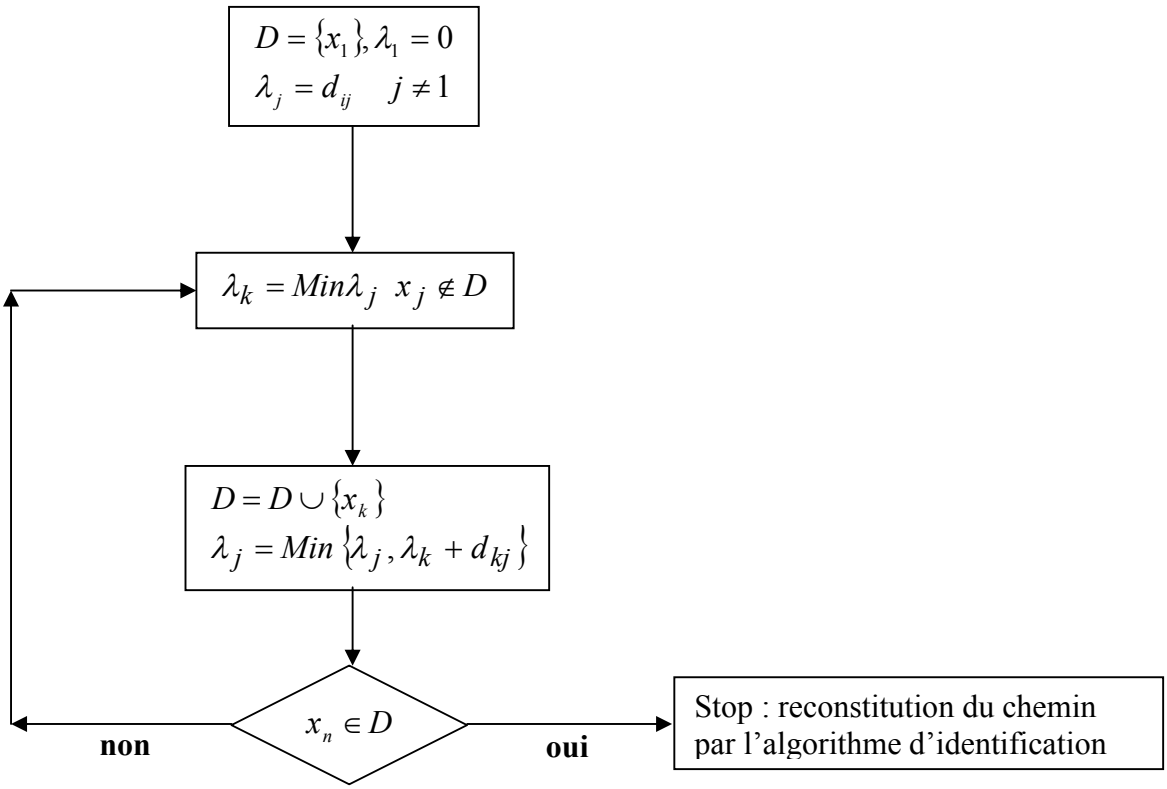
Algorithme de BELLMAN :

Si le réseau ne comporte pas de circuit, on utilise l'algorithme de BELLMAN basé sur le principe de la programmation dynamique de calcul de proche en proche.



Algorithme de Dijkstra :

Si $d(u_i) \geq 0 , \forall i$: alors on utilise l'algorithme de Dijkstra qui consiste à calculer les plus courtes distances de proche en proche par ajustements successifs, en faisant jouer au sommet x_k obtenu à la $k^{ième}$ itération un rôle particulier.



II-2) Problème du flot maximum (Algorithme de Ford et Fulkerson) :

Considérons un réseau $R = (X, U, C)$ avec $X = \{s, a_1, a_2, ..., a_n, p\}, U = \{1,2,3,....,m\}$; C_i est la capacité de l'arc i.

1=(p, s) est appelé l'arc de retour , $C_1 = +\infty$. $W^-(s) = W^+(p) = 1$

alors un flot d'un réseau R est la donnée du vecteur $\zeta = (\zeta_1, \zeta_2, ...\zeta_m)$

Tel que pour tout arc (i) on a :

$$1)\ 0 \le \zeta_i \le C_i$$

$$2)\ \sum_{i \in W^-(x)} \zeta_i = \sum_{i \in W^+(x)} \zeta_i \quad \forall x \in X .$$

Définition 2.3 :

-On désigne par $\zeta_x = \sum_{i \in W^-(x)} \zeta_i$ appelé l'afflux au point x.

-Un arc (i) est dit saturé si $\zeta_i = C_i$.

-Un flot ζ est dit complet si tout chemin allant de (s) à (p) contient au moins un arc saturé.

Algorithme de Ford et Fulkerson :

Cet algorithme nous permet de chercher un flot ζ tel que $0 \le \zeta_i \le C_i$ et ζ_1 maximum.

Pour cela on démarre d'un flot ζ (au jugé) compatible (sinon on prend $\zeta_i = 0, \forall i$).

-Pour diminuer le nombre d'itérations on cherche un flot complet, c'est-à-dire tel que tout chemin allant de s à p contient au moins un arc saturé.

A partir de ce flot utilisons la procédure de marquage suivante :

(1)On marque (s) d'un signe +.

(2)Si a_i est marqué, on marque d'un signe + tout sommet a_j non marqué tel que

$(a_i, a_j) \in U$ et $\zeta(a_i, a_j) < C(a_i, a_j)$.

(3)Si a_i est marqué, on marque d'un signe (-) tout sommet a_j tel que

$(a_j, a_i) \in U$ et $\zeta(a_j, a_i) > 0$.

Si avec cette procédure on parvient à marquer le sommet (p) alors il existera une chaîne élémentaire μ de (s) à (p).

Soient :

$$\mu^+ = \left\{ u \in \mu / u = a_i a_j \text{ et } a_j \text{ marqué d'un signe } + \right\}$$

$$\mu^- = \left\{ u \in \mu / u = a_i a_j \text{ et } a_j \text{ marqué d'un signe } - \right\}$$

$\delta_1 = \underset{u_i \in \mu^+}{Min}(C(u) - \zeta(u))$; $\delta_2 = \underset{u_i \in \mu^-}{Min}(\zeta(u))$ (donc $\delta_1 > 0$ et $\delta_2 > 0$), $\delta = Min(\delta_1, \delta_2)$.

Posons $$\zeta'(u) = \begin{cases} \zeta(u) \text{ si } u \notin \mu \\ \zeta(u) - \delta \text{ si } u \in \mu^- \\ \zeta(u) + \delta \text{ si } u \in \mu^+ \end{cases}$$

Il est évident que ζ' ainsi construit demeure toujours un flot et en plus

$\zeta'(p,s) = \zeta_1' = \zeta_1 + \delta > \zeta_1$ donc on a augmenté la valeur du flot.

(4) on réitère la procédure de marquage avec le nouveau flot obtenu jusqu'à ne plus pouvoir améliorer la valeur du flot, c-à-d on n'arrive pas à marquer le sommet (p) alors le flot obtenu est un flot à valeur maximum .

Cet algorithme peut être justifié par le théorème suivant :

Théorème de la coupe minimum :

Définition 2.4:

On appelle coupe ζ dans le réseau $R = (X, U, C)$ et $(s, p) = u_1$ une partition A et $\overline{A}$ de l'ensemble X telle que : $s \in A$ et $p \in \overline{A}$

La capacité d'une coupe ζ est définie par $C(\zeta) = \sum_{u \in w^+(A)} C(u)$.

Théorème 2.5:

Pour tout flot réalisable f sur $R = (X, U, C)$ (avec $C(u_1) = \infty$) et pour toute coupe ζ séparant p de s on a : $f(u_1) \leq C(\zeta)$.

Ce théorème possède de nombreuses applications, en fait c'est une autre version du théorème 1.1 .

Remarque :

Dans le cas des réseaux canalisés $R = (X, U, b, c)$ on peut appliquer l'algorithme de Ford et Fulkerson en posant $\delta_2 = Min\ \zeta(u) - b(u)$ mais pour l'initialisation on ne dispose pas de flot réalisable car le flot nul ne l'est pas. Le paragraphe suivant va lever cette difficulté.

II-3) Problème du flot réalisable :

Théorème d'existence pour un flot réalisable (A-J-Hoffman) :

Etant donné un réseau $R = (X, U, b, c)$ on ne peut appliquer l'algorithme de Ford et Fulkerson que si on dispose d'un flot compatible au départ pour commencer notre procédure car $\zeta = 0$ n'est pas nécessairement une solution admissible. Le théorème d'Hoffman établit une C.N.S d'existence d'un tel flot.

Théorème 2.6:

Etant donné des nombres b_i et c_i $i=\overrightarrow{1,m}$ la C.N.S pour qu'il existe un flot ζ sur un réseau $R=(X,U,b,c)$ tel que $b_i \le \zeta_i \le c_i$ est que :

Pour tout cocycle élémentaire (W) de (X,U) alors :

$$\sum_{i\in W^+} c_i \ge \sum_{i\in W^-} b_i \quad \text{et} \quad \sum_{i\in W^-} c_i \ge \sum_{i\in W^+} b_i \ .$$

La démonstration repose sur une procédure de recherche d'un flot compatible. On va la reprendre pour expliquer cette procédure.

Démonstration :

La condition est nécessaire car si un tel flot (ζ) existe alors : $\sum_{i\in W^+}\zeta_i = \sum_{i\in W^-}\zeta_i$ et $b_i \le \zeta_i \le c_i \Rightarrow$

$\sum_{i\in W^+} b_i \le \sum_{i\in W^+}\zeta_i = \sum_{i\in W^-}\zeta_i \le \sum_{i\in W^-} c_i$ donc : $\sum_{i\in W^+} b_i \le \sum_{i\in W^-} c_i$.

De même $\sum_{i\in W^-} b_i \le \sum_{i\in W^-}\zeta_i = \sum_{i\in W^+}\zeta_i \le \sum_{i\in W^+} c_i$ donc : $\sum_{i\in W^-} b_i \le \sum_{i\in W^+} c_i$.

Pour montrer que la condition est suffisante, utilisant la procédure suivante qui du même coup nous donne un algorithme de construction d'un flot réalisable s'il existe.

Etant donné le flot $\zeta = 0$ non nécessairement réalisable sur le réseau $R=(X,U,b,c)$ associant à ce flot un réseau $R'=(X,U',b',c')$ tel que $\forall u \in U$:

$$b'(u)=\begin{cases} b(u) & si\ \zeta(u)\ge b(u) \\ \zeta(u) & si\ \zeta(u)< b(u)\end{cases} \text{ et } c'(u)=\begin{cases} c(u) & si\ \zeta(u)\le c(u) \\ \zeta(u) & si\ \zeta(u)> c(u)\end{cases}.$$

Soit $u=(x,y)\in U$ tel que $c(u)\ne c'(u)$ alors on ajoute à l'ensemble U un arc $u'=(y,x)$ avec :

$b'(u')=0$, $c'(u')=\zeta(u)-c(u)$ et $\zeta(u')=\zeta(u)$ (on voit que $\zeta(u')\le c'(u')$).

En ajoutant l'arc (y,x) on va créer un cycle de $(y$ à $x)$ et l'algorithme de Ford Fullkerson va augmenter la valeur du flot sur le cycle mais le diminuer sur l'arc (x,y) car il est en marquage inverse.

On répétera l'opération jusqu'à épuiser tous les arcs où $c(u) \neq c'(u)$ donc on obtient un flot ζ' tel que $\zeta'(u) \leq c(u) \quad \forall u \in U$.

De même si $u = (x, y) \in U$ tel que $b(u) \neq b'(u)$ on ajoute à l'ensemble U un arc $u' = (x, y)$ avec

$$b'(u') = 0\,, \quad c'(u') = c'(u) \text{ et } \zeta(u') = \zeta(u)\,.$$

En ajoutant l'arc u' on va créer un cycle de $(y$ à $x)$ et l'algorithme Ford et Fullkerson va augmenter le flot sur l'arc (x, y) car il est en marquage direct, on répétera l'opération jusqu'à épuiser tous les arcs tel que $b(u) \neq b'(u)$, donc on obtient un flot ζ' tel que $\zeta'(u) \geq b(u) \quad \forall u \in U$ ainsi on obtient un flot ζ dans le réseau $R = (X, U, b, c)$ tel que $b(u) \leq \zeta(u) \leq c(u) \quad \forall u \in U$.

Néanmoins cette procédure peut mener à deux issues :

1) Soit l'algorithme se déroule normalement et on aboutit à un réseau : $R' = (X, U', b', c')$ où $\forall u \in U : b(u) = b'(u)$ et $c(u) = c'(u)$ et dans ce cas le flot généré par l'algorithme est réalisable sur le réseau $R = (X, U, b, c)$.

2) Soit en choisissant un arc $u = (x, y)$ tel que $c(u) \neq c'(u)$ (ou $b(u) \neq b'(u)$) et en ajoutant l'arc $u' = (y, x)$ l'algorithme de Ford et Fulkerson ne permet pas d'augmenter la valeur du flot sur l'arc $u = (x, y)$. Cela implique que la procédure de marquage ne permet pas d'atteindre le sommet (y) en partant de (x) .

Alors soit (A) l'ensemble des sommets marqués donc : $x \in A$ et $y \notin A \Rightarrow u \in W^+(A)$.
$c(u) \neq c'(u) \Rightarrow \zeta'(u) = c'(u) > c(u)$.

$$\forall v \in W^+(A) \quad (v \neq u) \Rightarrow \begin{cases} \zeta'(v) = c'(v) \; l'arc \; (v) \; doit \; être \; en \; marquage \; direct \\ et \\ c'(v) \geq c(v) \; par \; construction \; de \; R' \end{cases}$$

$$\forall_{v \neq u'} v \in W^-(A) \Rightarrow \begin{cases} \zeta'(v) = b'(v) \ marquage \ direct \\ et \\ b'(v) \leq b(v) \ par \ construction \ de \ R' \end{cases}$$

ζ' est un flot sur $R = (X, U, b, c)$ par construction de ζ' mais non réalisable et $W(Y) = W(A) - u'$ est un cocycle sur (X, U) :

$$\sum_{v \in W^+(Y)} \zeta'(v) = \sum_{v \in W^-(Y)} \zeta'(v) \Rightarrow \sum_{v \in W^+(Y)} c'(v) = \sum_{v \in W^-(Y)} b'(v).$$

Mais $\left.\begin{array}{ll} u \in W^+(Y) & c'(u) > c(u) \\ si \ v \in W^+(Y) - u & c'(v) \geq c(v) \end{array}\right\} \Rightarrow \sum_{v \in W^+(Y)} c(v) < \sum_{v \in W^-(Y)} b'(v) \leq \sum_{v \in W^-(Y)} b(v).$

Donc on a mis en évidence un cocycle $(W(Y))$ où la condition d'Hoffman n'est pas vérifiée. Par conséquent la condition d'Hoffman est suffisante.

II-4) Problème de flot de coût minimum :

Définition 2.7 :

Le problème du flot de coût minimum sur un réseau $R = (X, U, a, b, c)$ consiste à trouver un vecteur $(\zeta = \zeta_1, \zeta_2, ..., \zeta_n)$ tel que :

1) $\sum_{u_i \in W^+(j)} \zeta(u_i) = \sum_{u_i \in W^-(j)} \zeta(u_i) \qquad \forall j = 1,2,..., n.$

2) $b_i \leq \zeta_i \leq c_i \qquad i = 1,2,..., m.$

3) $\sum_{i=1}^{m} a_i \zeta_i$ soit minimum.

Remarque :

Le problème du flot de coût minimum peut être formulé comme un programme linéaire en variables bornées qui consiste à trouver un vecteur : $\zeta = (\zeta_1, \zeta_2, ..., \zeta_m)$ tel que :

$$\begin{cases} A\zeta = 0 \quad b_i \le \zeta_i \le c_i \quad i = \overline{1,m} \\ a\zeta = Z(Min) \end{cases}$$

A étant la matrice d'incidence aux arcs du graphe (X,U) définie par :

$$A_{ij} = \begin{cases} 1 \quad si \quad i = I(u_j), & i = 1,...,n \\ -1 \quad si \quad i = T(u_j), & j = 1;...,m \\ 0 \quad \text{sin}\, on \end{cases}$$

Théorème 2.8 :

Une C.N.S pour qu'un flot ζ réalisable sur $R = (X,U,b,c)$ soit une solution optimale du problème du flot de cout minimum est que pour tout cycle Γ de (X,U) tel que :
$\delta = Min\left\lfloor \underset{u\in\Gamma^+}{Min}(c(u) - \zeta(u)), \underset{u\in\Gamma^-}{Min}(\zeta(u) - b(u)) \right\rfloor > 0$ on ait : $\sum_{u\in\Gamma^+} a(u) \ge \sum_{u\in\Gamma^-} a(u)$.

Nous allons reprendre la démonstration de ce théorème car elle constitue une procédure de recherche d'un flot de coût minimum.

Démonstration :

La condition est nécessaire.
Soit (Γ) un cycle tel que $\delta > 0$ .Supposant que $\sum_{u\in\Gamma^+} a(u) < \sum_{u\in\Gamma^-} a(u)$,alors le vecteur ζ' tel que :

$$\zeta'(u) = \begin{cases} \zeta(u) + \delta \quad si\ u \in \Gamma^+ \\ \zeta(u) - \delta \quad si\ u \in \Gamma^- \\ 0 \qquad si\ u \notin \Gamma \end{cases}.$$

On voit que ζ' est par construction un flot réalisable sur $R=(X,U,b,c)$ et on a en plus

$a\zeta' = a\zeta + \delta(\sum_{u\in\Gamma^+} a(u) - \sum_{u\in\Gamma^-} a(u)) < a\zeta$ car $\delta > 0$.

Donc ζ ne peut être optimale par conséquent la condition est nécessaire.

Montrons que la condition est suffisante :

$\delta > 0$ et ζ est un flot réalisable.

Si ζ n'est pas optimale, soit ζ' un flot réalisable tel que $a\zeta' - a\zeta < 0$.

ζ et ζ' étant des flots sur (X,U) donc $\zeta'' = \zeta - \zeta'$ est aussi un flot .

D'après le théorème de décomposition [9] on peut trouver dans (X,U) , (k) cycles $\Gamma_1,\Gamma_2,...,\Gamma_k$ et (k) nombres réels positifs $\alpha_1,\alpha_2,...,\alpha_k \quad (\alpha_i > 0)$ tels que :

$$\forall u \in \Gamma_j \begin{cases} \zeta''(u) > 0 \; si \; u \in \Gamma_j^+ \\ \zeta''(u) < 0 \; si \; u \in \Gamma_j^- \end{cases} \text{ et } \zeta'' = \alpha_1\gamma_1 + \alpha_2\gamma_2 + ... + \alpha_k\gamma_k .$$

γ_j étant le vecteur représentatif de Γ_j c-à-d : $\gamma_j^i = \begin{cases} 1 & u_i \in \Gamma_j^+ \\ -1 & u_i \in \Gamma_j^- \\ 0 & u_i \notin \Gamma_j \end{cases}$.

$a\zeta'' = a(\zeta' - \zeta) = a\zeta' - a\zeta < 0$

et $a\zeta'' = \alpha_1 a\gamma_1 + \alpha_2 a\gamma_2 + ... + \alpha_k a\gamma_k \Rightarrow \alpha_1 a\gamma_1 + ... + \alpha_k a\gamma_k < 0$

$\alpha_i > 0 \; \forall i \Rightarrow$ il existe au moins un cycle Γ_j tel que $a\gamma_j < 0$ donc $\sum_{u\in\Gamma_j^+} a(u) - \sum_{u\in\Gamma_j^-} a(u) < 0$,par conséquent la condition est suffisante.

Algorithme de recherche d'un flot de coût minimum :

Associons d'abord à tout flot (ζ) réalisable sur $R=(X,U,a,b,c)$ le réseau $\hat{R}(\zeta) = (X, U' \cup U'' \cup U''', \hat{d})$ défini comme suit :

Si $u=(x,y)\in U$ alors $u'=(x,y)\in U'$ et $\hat{d}(u') = \begin{cases} +\infty & si \quad \zeta(u) = c(u) \\ a(u) & si \quad \zeta(u) < c(u) \end{cases}$

Si $u=(x,y)\in U$ alors $u''=(y,x)\in U''$ et $\hat{d}(u'')=\begin{cases}+\infty & si \quad \zeta(u)=b(u)\\ a(u) & si \quad \zeta(u)>b(u)\end{cases}$.

$U'''=\{(s,x)/\ x\in X-s\}$; $\hat{d}(u''')=\sum_{u\in U'\cup U''}|a(u)| \quad \forall u'''\in U'''$ (s quelconque).

Algorithme

(0) Si $R=(X,U,a,b,c)$ n'admet pas de flot réalisable ,terminer.

Sinon soit ζ un flot réalisable sur R aller en (1).

(1) Si le problème des plus courtes distances dans $\hat{R}(\zeta)$ admet une solution, terminer.

Sinon aller en (2).

(2) Il existe dans $\hat{R}(f)$ un circuit absorbant (C) ,les arcs de (C) correspondent dans R à un cycle Γ pour lequel $\sum_{\Gamma^-}a(u)>\sum_{\Gamma^+}a(u)$.

Poser $\varepsilon=Min\left\{\underset{\Gamma^-}{Min}(\zeta(u)-b(u)),\underset{\Gamma^+}{Min}(c(u)-\zeta(u))\right\}$

$\zeta:=\zeta+\varepsilon\Gamma$ aller en (1)

Flot de valeur maximum de coût minimum :

On se propose de trouver parmi tous les flots de valeur maximale un qui est de coût minimum. Ce problème est en relation directe avec le problème de transport où le flux sur l'arc (i,j) représente la quantité servie par l'entrepôt (i) au point de vente (j) et le coût sur cet arc est égal au coût de transport de (i) vers (j).
L'algorithme suivant apporte une réponse à ce problème.

Algorithme :

(0) $R=(X,U,a,b,c)$, a_{ij} :coût unitaire, Φ_0 :valeur du flot donné par Ford et Fulkerson.

$\zeta=(\zeta_{ij})_{(ij)\in U}$ / $b_{ij}\le\zeta_{ij}\le c_{ij} \quad \forall(ij)\in U$ et de valeur $\Phi(\zeta)$.

(1) Construire le graphe d'écart $G(\zeta)=(X,U(\zeta))$ tel que :

$$(ij) \in U(\zeta) \text{ Si } \begin{cases} (ij) \in U \quad et\ c_{ij} - \zeta_{ij} > 0 \\ ou \\ (ji) \in U \quad et\ \zeta_{ji} - b_{ij} > 0 \end{cases}$$

- S'il n'existe pas de chemin de s à p dans $G(\zeta)$: ζ est de valeur maximale égal à Φ_0 et de coût minimum. Stop.
- S'il existe un chemin de s à p dans $G(\zeta)$,poser :

  $$l_{ij} = \begin{cases} a_{ij} \quad si \quad (ij) \in U(\zeta) \cap U \\ -a_{ij} \quad si \quad (ij) \in U(\zeta) - U \end{cases}$$

  Construire dans $G(\zeta)$ un chemin μ de s à p de longueur ($\sum l_{ij}$) minimale

  Poser : $r_{ij} = \begin{cases} c_{ij} - \zeta_{ij} \quad si \quad (ij) \in \mu \cap U \\ \zeta_{ij} - b_{ij} \quad si \quad (ij) \in \mu - U \end{cases}$

  $$\alpha = Min\{r_{ij} / (ij) \in \mu\}$$

  $$\Phi(\zeta) = \Phi(\zeta) + \alpha .$$

- Si $\Phi_0 \geq \Phi(\zeta)$ poser $\zeta_{ij} = \begin{cases} \zeta_{ij} + \alpha \quad si \quad (ij) \in \mu \cap U \\ \zeta_{ij} - \alpha \quad si \quad (ij) \in \mu - U \end{cases}$
- Si $(\Phi_0) < \Phi(\zeta)$ poser $\zeta_{ij} = \begin{cases} \zeta_{ij} + \Phi_0 - \Phi(\zeta) \quad si\ (ij) \in \mu \cap U \\ \zeta_{ij} - \Phi_0 + \Phi(\zeta) \quad si\ (ij) \in \mu - U \end{cases}$

  $\zeta = (\zeta_{ij})_{(ij) \in U}$ est de valeur Φ_0 et de coût minimum. Stop.

Si dans un réseau de transport le flot est contraint à ne pas dépasser une valeur Φ_1 au niveau d'un nœud (une intersection),il suffit de poser $\Phi_0 = \Phi_1$ pour utiliser cet algorithme.

Remarque [7] :

On peut montrer que le cas particulier où $0 \leq \zeta_i \leq C_i$ et $C_i \in IN$ est équivalent au cas général où la contrainte est $b_i \leq \zeta_i \leq C_i$ en procédant de la manière suivante :

-On construit un réseau R' en conservant les sommets et les arcs de R et en posant :

$C'_1 = +\infty$; $C'_i = C_i - b_i \quad i = \overline{2,n}$.

On ajoutera une entrée $\bar{s}$ et une sortie $\bar{p}$ que l'on relie aux sommets de G de la façon suivante :

-Si l'arc $i=(a_k,a_l)$ est tel que $b_i \geq 0$ on trace un arc $(a_k,\overline{p})$ de capacité $C'(a_k,\overline{p})=b_i$ et un arc $(\overline{s},a_l)$ de capacité $C'(\overline{s},a_l)=b_i$.

-Si l'arc $j=(a_t,a_r)$ est tel que $b_j<0$ on trace un arc $(\overline{s},a_p)$ de capacité $C'(\overline{s},a_t)=-b_j$ est un arc $(a_r,\overline{p})$ de capacité $C'(a_r,\overline{p})=-b_j$.

Montrons qu'à un flot ζ' dans G' saturant les arcs sortants correspond dans G un flot (ζ) compatible.

On posera $\zeta_i=\zeta_i'+b_i \quad i=2,3,....,m$ et $\zeta_1=\zeta_1'$.

Comme $0\leq\zeta_i'\leq C_i-b_i \;\; i=2,3,...m \Rightarrow b_i\leq\zeta_i\leq C_i \quad i=2,3,...,m$ donc ζ_i construit est compatible.

Pour tout sommet a de G on a :

$$\sum_{i\in W^+(a)}\zeta_i-\sum_{i\in W^-(a)}\zeta_i=\sum_{\substack{i\in W^+(a)\\ b_i\geq 0}}(\zeta_i'+b_i)+\sum_{\substack{j\in W^+(a)\\ b_j<0}}(\zeta_j'+b_j)-\sum_{\substack{i\in W^-(a)\\ b_i\geq 0}}(\zeta_i'+b_i)-\sum_{\substack{j\in W^-(a)\\ b_j<0}}(\zeta_j'+b_j)=0$$

(car ζ' est un flot et par la construction particulière de G').

-Si les capacités sont rationnelles on peut prendre $\lambda=1/D$ comme unité des capacités (où D est le PPCM des dénominateurs des capacités).

Le cas irrationnel ne se pose pas dans la pratique car la densité de l'ensemble des rationnels Q dans l'ensemble des nombres réels R nous permet d'approximer tout irrationnel par un rationnel.

II-5 Modélisation des différents problèmes étudiés en termes de flot de coût minimum[9] :

Le problème du flot de coût minimum est un problème très général qu'on peut utiliser dans la résolution de tous les problèmes d'optimisation déjà cités.

-Rechercher le «le plus court chemin » de (s) à (p) dans un réseau $R=(X,U,d)$ revient à chercher un flot de coût minimum dans le réseau $R'=(X,U',a,b,c)$ Ainsi défini :

$U'=U\cup(p,s)/a(u)=d(u),b(u)=0,c(u)=1 \quad \forall u\in U, a(p,s)=0, b(p,s)=1, c(p,s)=1.$

-Rechercher un « flot maximum » sur $R=(X,U,c)$ avec $(p,s)\in U$ revient à chercher un flot de coût minimum dans le réseau $R'=(X,U,a,b,c)$ tel que :

$a(u)=b(u)=0 \quad si\ u\neq(p,s) \quad , \quad a(p,s)=-1 \ \ et \ \ b(p,s)=0$.

La résolution d'un problème de transport (T) de la forme :

$$(T)\begin{cases}\sum_{j=1}^{n} x_{ij}=a_i & i=1,..,m\\ \sum_{i=1}^{m} x_{ij}=b_i & j=1,...,n \qquad x_{ij}\geq 0.\\ \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}x_{ij}=Z(Max)\end{cases}$$

Peut être ramenée à la recherche d'un flot de coût minimum dans le réseau $R=(X,U,a,b,c)$ construit ainsi :

On ajoute un entrepôt (s) qui va servir tous les entrepôts $\alpha_1,\alpha_2,...,\alpha_m$.

On ajoute un magasin (p) qui sera desservi par tous les magasins $\beta_1,...,\beta_n$.

$U_1=\{(s,\alpha_i)/i=1,...,m\}$ ensemble des arcs qui relient (s) aux entrepôts.

$U_2=\{(\beta_j,p)j=1,...,n\}$ ensemble des arcs qui relient (p) aux magasins.

$U_3=\left\{(\alpha_i,\beta_j)\Big/\begin{matrix}i=1,...,m\\ j=1,...,n\end{matrix}\right\}$ ensemble des arcs qui relient l'entrepôt (i) avec le magasin (j).

Posons :

$X=\{\alpha_1,\alpha_2,...,\alpha_m\}\cup\{\beta_1,\beta_2,...,\beta_m\}\cup\{s,p\}$

$U=U_1\cup U_2\cup U_3$

-Si $u=(s,\alpha_i)$ alors : $a(u)=0 \quad , \quad b(u)=c(u)=a_i$.

-Si $u=(\beta_j,p)$ alors : $a(u)=0 \quad , \quad b(u)=c(u)=b_j$.

-Si $u=(\alpha_i,\beta_j)$ alors : $a(u)=d_{ij}$, $b(u)=0$, $c(u)=\infty$.

-Si $u=(p,s)$ alors : $a(u)=b(u)=0$, $c(u)=\infty$.

Sommet à capacité [5] :

Dans de nombreux problèmes réels, les sommets aussi bien que les arcs possèdent des capacités limitées. Les réseaux routiers sont de toute évidence caractérisés par ce cas ;car aussi bien les tronçons de route(représentés par les arcs)que les intersections(représentées par les sommets)possèdent des capacités limitées.

On peut utiliser l'algorithme de Ford et Fulkerson dans les réseaux de ce type en opérant dans le réseau initial les modifications suivantes :

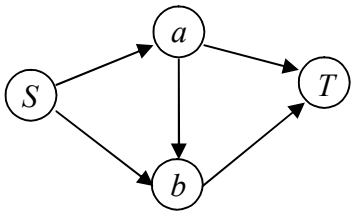
Tout sommet (a) à capacité sera subdivisé en deux sommets a_1 et a_2.

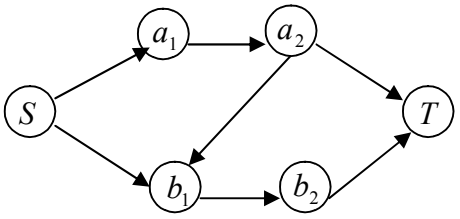
On ajoute l'arc (a_1, a_2) de même capacité que le sommet (a).

Si $u \in w^-(a)$ alors $u \in w^-(a_1)$

Si $u \in w^+(a)$ alors $u \in w^+(a_2)$

Exemple :





Ainsi on revient à un réseau classique et la solution cherchée est valable pour le réseau à sommets à capacité.

Chapitre III

Problèmes de transport bicritère

III-1) définition et caractérisation du flot maximal:

Introduction :

Soit $R = (X, U, C)$ un réseau tel que :

$U = \{u_1, , u_2, ..., u_m, \}, X = \{1,2,..., n\}, (s) = 1, (p) = n, W^-(s) = W^+(p) = \phi$ et $c(u) \in IN^*$.

$U' = U \cup \{u_r\}$ avec $u_r = (p, s)$ et $c(u_r) = \infty$.

Définition 3.1 :

On appelle matrice d'incidence associée au graphe $G = (X, U)$ la matrice $A(m, n)$ tel que

$$\forall i \in X, \forall u_j \in U \begin{cases} a_{ij} = +1 & \text{si} \quad u_j \in W^+(i) \\ a_{ij} = -1 & \text{si} \quad u_j \in W^-(i) \\ 0 & \text{sinon} \end{cases}.$$

Définition 3.2:

Un vecteur colonne (f) de IR^{m+1} est un flot sur R si $Af = 0$ (où A est la matrice d'incidence du graphe $G' = (X, U')$).

On dit que (f) est un flot « réalisable »sur R si de plus on a :

$0 \le f_j \le c(u_j) \quad \forall j = 1,..., m$.

Un flot (f) réalisable sur $R = \{X, U, C\}$ est dit « maximal »si pour tout flot réalisable (f') sur R on a : $f'_j \ge f_j \quad \forall j \Rightarrow f' = f$.

Définition 3.3:

Si (f) est un flot réalisable sur $R=(X,U,C)$ et (μ) un circuit dans $G=(X,U)$ alors μ est dit « saturé »par (f) s'il existe un arc $(v)\in\mu / f(v)=c(v)$.

Le théorème suivant nous permet de donner une caractérisation d'un flot maximal

Théorème 3.4:

(f) est un flot maximal sur $R=(X,U,C)$ si et seulement si tout circuit (μ) dans $G=(X,U)$ est saturé par (f).

Démonstration :

Si (f) est maximal sur R supposons que (μ) est un circuit dans (G) non saturé par f donc : $\forall v\in\mu \quad f(v)<c(v)$ posons $\varepsilon=\underset{v\in\mu}{Min}\left(c(v)-f(v)\right)>0$ car : $\forall v\in\mu \quad f(v)<c(v)$.

Représentons μ par son vecteur dans $G=(X,U)$, $\mu=(\mu_1,\mu_2,...,\mu_m)^T$ tel que :

$$\mu_j=\begin{cases}1 & si\ u_j\in\mu\\ 0 & \text{sinon}\end{cases}.$$

Et soit le vecteur $f'=f+\varepsilon\mu \Rightarrow Af'=Af+\varepsilon A\mu$.

$$\left.\begin{array}{l} Af=o \text{ car } f \text{ est un flot}\\ A\mu=0 \text{ car } \sum_{j=1}^{m}a_{ij}\mu_j=\sum_{u_j\in\mu}a_{ij} \text{ et à tout } u_j\in W^+(i)\\ \text{correspond un et un seul } u_k\in W^-(i)\end{array}\right\}\Rightarrow Af'=0 \text{ donc } f' \text{ est un flot.}$$

De plus f' est réalisable sur $R=(X,U,C)$ car:

- Si $u\notin\mu\Rightarrow f'(u)=f(u)\le c(u)$ car f est réalisable
- Si $u\in\mu\Rightarrow 0<f'(u)=f(u)+\varepsilon\le f(u)+c(u)-f(u)=c(u)$. car $0<\varepsilon\le c(u)-f(u)$ alors f' est réalisable car $0<f'(u)\le c(u)$, f' sature μ et $f'\ge f$ mais f est maximal donc $f=f'\Rightarrow f$ sature μ.

Montrons que la condition est suffisante.

Supposons que chaque circuit dans $G=(X,U)$ est saturé par f

Soit f' un flot réalisable sur $G=(X,U)$ tel que $f' \geq f$.

Posons $f''=f'-f$ donc $f'' \geq 0$ et $Af''=Af'-Af=0-0=0 \Rightarrow f''$ est un flot sur $G=(X,U)$ réalisable.

Soit $U'=\{u \in U / f''(u)>0\}$ alors obligatoirement U' doit contenir un circuit car: s'il n'y avait pas de circuit μ dans U' alors :

Soit $u \in U'$ comme il n'existe pas de circuit μ dans U' donc il existe un cocycle $W(y)$ inclus dans U' tel que : $u \in W^+(y)$ et $W^-(y)=\phi$ (l'absence de circuit implique un cocycle minimal de ce type).

Mais f'' est un flot sur $R=(X,U,C)$ donc $\sum_{v \in W^+(y)} f''(v)=\sum_{v \in W^-(y)} f''(v)$

$W^+(y) \subset U'$ et $W^-(y)=\phi$ alors $0<\sum_{v \in W^+(y)} f''(v)=\sum_{v \in W^-(y)} f''(v)=0$ contradiction :donc il existe obligatoirement un circuit μ dans U' .

Soit $\mu \in U'$ un tel circuit donc : $\forall u \in \mu : f''(u)>0$.

C'est à dire : $\forall u \in \mu : f'(u)-f(u)>0$ mais par hypothèse tout circuit est saturé par $f \Rightarrow \exists v \in \mu$ tel que $f(v)=c(v)$

$v \in \mu$ donc $f'(v)-f(v)>0 \Rightarrow f'(v)-c(v)>0$ Contradiction car $f'(v) \leq c(v)$ (f' est réalisable) par conséquent f est maximal.

Théorème 3.5 :

Si (f) sature un chemin (γ) dans $R=(X,U,C)$ alors pour tout flot réalisable (f') sur R il existe au moins un arc $u \in \gamma$ tel que $f'(u) \leq f(u)$.

Démonstration :

$$\left.\begin{array}{l} f \text{ sature } \gamma \text{ donc } \exists u \in \gamma / f(u) = c(u) \\ f' \text{ réalisable donc } \forall v \in U : f'(v) \le c(v) \end{array}\right\} \Rightarrow f'(u) \le f(u)$$

Essayons de trouver un flot « maximal » de valeur « minimum » dans le réseau $R' = (X, U', C')$ ce qui revient à résoudre :

$$\begin{cases} Min\ z = f(u_r) \\ Af = 0 \\ 0 \le f \le c \\ f \text{ maximal.} \end{cases}.$$

Théorème 3.6 :

Si (f) est un flot « maximal » sur $R'(X, U', C)$ alors tout chemin de (s) à (p) est saturé par f.

Démonstration :

C'est un corollaire du théorème 3.4 .

Posons $f(u_r) = \sum_{u \in W^-(p)} f(u)$ alors f maximal dans $R \Rightarrow f$ maximal dans R'.

Tout chemin (γ) de (s) à (p) dans $R = (X, U, C)$ va créer un circuit $\mu = \gamma \cup \{u_r\}$ dans $R' = (X, U', C')$.

D'après le théorème (1) μ est saturé par f mais u_r n'est jamais saturé (car $c'(u_r) = +\infty$)

Donc $\exists u \in (\gamma)$ tel que $f(u) = c'(u) = c(u)$.

Théorème 3.7:

Si $G = (X, U)$ est sans circuit et si $\exists Y \subset X$ tel que :

$(s) \in Y$, $(p) \notin Y$ et $f(u) = c(u) \quad \forall u \in W^+(Y)$ alors (f) est un flot maximal sur $R = (X, U, C)$.

Démonstration :

Soit (γ) un chemin de (s) à (p) dans $G - \{u_r\}$ alors $W^+(Y) \cap \gamma \neq \phi$

Soit $v \in W^+(Y) \cap \gamma$: $v \in W^+(Y) \Rightarrow v$ est saturé par (f)

$v \in \gamma \Rightarrow \gamma$ est un chemin saturé par (f)

$G - \{u_r\}$ est sans circuit, donc tous les circuits de $R = (X, U, C)$ sont constitués par un chemin de (s) à (p) saturé et de l'arc u_r .Par conséquent tout circuit dans $R = (X, U, C)$ est saturé par (f) et d'après le théorème (1) f est maximal.

Procédure de recherche du flot maximal sur $R = (X, U, C)$:

(On suppose $G(X, U)$ sans cycle)

0) Posons $f = 0$ (flot nul réalisable)

1) $Y = \{s\}$, $k = 1$.

2) Si $p \in Y$ aller en (3).

Si $p \notin Y$ poser $\Omega = \{u \in W^+(Y) / f(u) < c(u)\}$

- Si $\Omega = \phi$ stop : f est maximal.
- Si $\Omega \neq \phi$ poser $v = (x, y) / c(v) = \underset{u \in \Omega}{Min}\, c(u)$

  $\gamma_k = v; Y = Y \cup \{y\}, k = k + 1$ aller en (2).

3) Soi γ le chemin $(\gamma_1, \gamma_2, ..., \gamma_k)$ de (s) à (p) .Poser :

$\varepsilon = \underset{u \in \gamma}{Min}[c(u) - f(u)] > 0$ .Poser $f = f + \varepsilon\mu$ avec $\mu = \gamma \cup \{u_r\}$. Aller en (1)

Justification :

En (1) si $\Omega = \phi$, f est maximal d'après le théorème 3.7 .

En (3) en ajoutant $\varepsilon\mu$ à (f) on obtient un flot réalisable avec au moins un arc saturé .

Procédure de minimisation d'un flot maximal :

1) Soit (f) un flot réalisable maximal sur $R=(X,U,C)$ trouvé par la procédure précédente.

   Poser $k=0$; $f^{(k)}=f$.

2) Soit $R(f^k)=(X,U\cup\{u_r\},C^k)\Big/\begin{matrix}c^k(u)=c(u) & \text{si} & u\neq u_r\\ c^k(u_r)=f^{(k)}(u_r)-1\end{matrix}$.

3)

- Si $R(f^k)$ admet un flot maximal (f') alors :

  poser $k=k+1$; $f^k=f', c^k(u_r)=c^k(u_r)-1$ aller en (2) .

- Sinon : stop $f^{(k)}$ est la solution du problème.

Justification :

Si à l'étape (3) $R(f^k)$ admet un flot maximal alors la valeur du flot aura diminué d'une unité au moins .Comme les $c(u)$ sont des entiers, la suite des flots maximaux générés par la procédure sera finie et de valeurs décroissantes.
Puisque le flot initial a été construit par la procédure du flot maximal en partant du flot nul et en choisissant chaque fois un chemin où les valeurs de $c(u)$ sont minimales, donc le flot généré à l'étape (3) sera de valeur inférieure à tous les flots maximaux déjà construits.

III-2) Problèmes de flots bicritères :

On peut envisager la résolution des problèmes de flots suivants :

1)Premier Problème :

on cherche un flot réalisable de coût minimum et maximal. Ce problème s'écrit donc :

$$\begin{cases} Af = 0 \\ b \le f \le c \\ f \text{ maximal} \\ af = Z(Min) \end{cases}$$

Si on privilégie le premier critère, on choisit un flot de coût minimum parmi tous les flots maximaux.

Si on privilégie le deuxième critère on choisit un flot maximal (s'il existe) parmi tous les flots de coût minimum. S'il n'existe pas, on propose la procédure suivante pour trouver un compromis entre les deux critères.

Soit f un flot de coût minimum, construisons un flot f' maximal à partir de f .

Soit μ un circuit non saturé dans $G(X,U)$ et soit $\varepsilon = \underset{u\in\mu}{Min}\left[c(u) - f(u)\right]$ alors :

$$f_i' = \begin{cases} f_i & \text{si } i \notin \mu \\ f_i + \varepsilon & \text{si } i \in \mu \end{cases}.$$

f' est un flot réalisable qui va saturer le cycle μ .On répète la procédure jusqu'à saturer tous les cycles de $G(X,U)$, ainsi d'après le théorème 3.4 le flot obtenu est maximal.

Remarque : Le problème du flot maximal peut être considéré comme un problème d'optimisation multicritère car un flot maximal est en fait une solution efficace du problème qui consiste à optimiser le flux sur chaque arc du réseau.

2) Deuxième problème :

On fixe un arc $v \in U$,on s'intéresse à un flot réalisable sur R maximal et de valeur minimal sur v .

$$\begin{cases} Af = 0 \\ b \le f \le c \\ f \quad Maximal \\ f(v) = Z(Min) \end{cases}$$

Ce problème a été traité dans l'article de Maiko-S [4].

3) Troisième problème :

Trouver un flot réalisable sur R maximum sur v.

$$\begin{cases} Af = 0 \\ b \le f \le c \\ f(v) = Z(Max) \end{cases}$$

C'est le problème du flot maximum, il suffit d'utiliser l'algorithme de Ford-Fullkerson en posant $u_r = v$.

4) Quatrième problème :

On fixe $v \in U$,on cherche à maximiser le flot sur v et à minimiser le coût du flot.

C'est problème bicritère qui s'écrit :

$$\begin{cases} Af = 0 \\ b \le f \le c \\ f(v) = Z_1(Max) \\ af = Z_2(Min) \end{cases}$$

L'algorithme du flot maximum de coût minimum nous permet de trouver parmi tous les flots maximum celui qui est de coût minimum ainsi on a trouvé une solution qui privilégie le premier critère (voir algorithme page 53).

L'algorithme du flot de coût minimum nous permet d'engendrer tous les flots de coût minimum et de choisir celui qui est maximum sur v.Ainsi on a privilégié le deuxième critère.

Construisons maintenant de la manière suivante un flot qui sera une solution de compromis.

Soit ψ l'ensemble de tous les circuits de $G(X,U)$ passant par v.

Pour tout circuit $\mu \in \psi$ soit $r_\mu = \underset{i \in \mu}{Min}(c_i)$ et $r_{\mu_0} = \underset{\mu \in \psi}{Max}(r_\mu) = \alpha$.

Trouvons un flot de coût minimum dans le réseau $R' = (X, U - \mu_0, a, b, c)$.

Soit $f^{(m)}$ un tel flot, alors soit le flot $f = (f_i)_{i \in U}$ suivant :

$$f_i = \begin{cases} f_i^{(m)} & si\ i \in U - \mu_0 \\ \alpha & si\ i \in \mu_0 \end{cases}$$

Construisons le graphe d'écart correspondant à f, appliquons lui l'algorithme de flot maximum de coût minimum en fixant $\Phi_0 = \alpha$, alors le flot obtenu sature tous les circuits passant par V donc si $U - \psi$ est sans circuit, d'après le théorème 3.4,le flot obtenu est maximal .

III-3) Problèmes de transport bicritère:

Considérons le problème de transport du chapitre (I) avec (n) entrepôts et (m) points de vente avec $\sum_{i=1}^{n} a_i > \sum_{j=1}^{m} b_j$.

On a vu qu'en ajoutant un point de vente artificiel avec une demande $b_{m+1} = \sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j$ et des coûts de transport $c_{i,m+1} = 0$ on peut trouver la solution optimale de ce problème.
On s'intéresse maintenant aux deux problèmes de transport bicritère suivants :

1) Premier problème :

On veut non seulement minimiser les coûts de transport mais aussi pour des raisons stratégiques ou économiques minimiser le nombre d'entrepôts utilisés pour assurer l'opération de transport.

Exemple illustratif :

Le problème de transport donné par le tableau (1) admet les solutions données par les tableaux (2),(3),(4) :

(1)

				a_i
2	3	8	7	6
1	0	4	2	2
6	11	15	9	12
1	6	3	2	

b_i

(2)

2	6 3	0 8	7	6
1	0	2 4	2	2
1 6	11	1 15	2 9	12
1	6	3	2	

(3)

2	3	8	7	6
1	0	4	2	2
1 6	6 11	3 15	2 9	12
1	6	3	2	

(4)

2	6 3	8	7	6
1	0	4	2	2
1 6	11	3 15	2 9	12
1	6	3	2	

La solution du tableau (2) est optimale du point de vue du 1er critère (cout minimum), mais on a utilisé (3) dépôts donc elle est la plus mauvaise du point de vue du 2ème critère.
$Z^1 = (65,3)$.

La solution du tableau (3) en utilisant un seul dépôt est optimale du point de vue du deuxième critère, mais nettement mauvaise pour le premier critère $Z^2 = (135,1)$
La solution du tableau (4) peut représenter une solution de compromis $Z^3 = (87,2)$

$$x_{ij}^3 = \begin{cases} x_{12} = 6, x_{31} = 1, x_{33} = 3, x_{34} = 2 \\ 0 \quad \text{sinon} \end{cases}$$

$$x_{ij}^1 = \begin{cases} x_{12} = 6, x_{23} = 2, x_{31} = 1, x_{33} = 1, x_{34} = 2 \\ 0 \quad \text{sinon} \end{cases}$$

$$x_{ij}^2 = \begin{cases} x_{31} = 1, x_{32} = 6, x_{33} = 3, x_{34} = 2 \\ 0 \quad \text{sinon} \end{cases}$$

Les solutions $x^{(1)}, x^{(2)}, x^{(3)}$ sont des solutions efficaces.

Cas général :

Si on considère le problème bicritère consistant à minimiser le nombre d'entrepôts utilisés et minimiser le coût de transport on peut l'écrire comme suit :

$$\begin{cases} Min \sum_{i=1}^{n} y_i \\ Min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} f_{ij} \\ \sum_{i=1}^{n} y_i f_{ij} = b_j \qquad f_{ij} \in IN; y_i \in \{0,1\} \\ \sum_{j=1}^{m} f_{ij} \leq a_i \\ \sum_{i=1}^{n} a_i y_i \geq \sum_{j=1}^{m} b_j \end{cases}$$

Si on veut optimiser le 2ème critère, sans tenir compte du nombre d'entrepôts utilisés, il suffit de résoudre un problème de transport classique. Soit f une telle solution.

Afin d'en déduire une solution efficace considérons l'ensemble J des entrepôts utilisés : $J = \left\{ e_i / 0 < \sum_{j=1}^{m} f_{ij} \leq a_i \right\}$ et soit $e_i \in J$, essayons de construire une solution de même coût que f qui n'utilise pas e_i.

Si $f_{ij_0} > 0$ trouvons un entrepôt $e_{i'} \in J / c_{ij_0} = c_{i'j_0}$ et $\sum_{j=1}^{m} f_{i'j} < a_{i'}$ alors on peut obtenir la solution f' telle que :

$$f'_{rj} = \begin{cases} f_{ij_0} - 1 & si \quad r = i, j = j_0 \\ f_{i'j_0} + 1 & si \quad r = i', j = j_0 \\ f_{rj} & \text{sinon} \end{cases}$$

Si on peut répéter l'opération jusqu'à obtenir $f'_{ij} = 0 \ , \forall j$ alors le nombre d'entrepôts utilisés aura diminué d'une unité et le coût demeurera toujours optimale. Quand on ne pourra plus diminuer le nombre d'entrepôts la solution trouvée sera efficace.

Si on veut optimiser le premier critère on commence par choisir le minimum d'entrepôts dont l'offre satisfait la demande. Le problème peut se ramener à un problème de sac à dos particulier qui s'exprime de la façon suivante :

$$\begin{cases} Min \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} a_i y_i \geq \sum_{j=1}^{m} b_j \\ y_i \in \{0,1\} \quad i = 1,...,n \end{cases}$$

Nous avons un problème avec des variables bivalentes, trouver une solution efficace qui engage le deuxième critère est généralement difficile. On peut néanmoins utiliser l'heuristique suivante pour donner une solution à ce problème.

On suppose que la suite a_i est décroissante (on ordonne les entrepôts dans l'ordre décroissant de leurs offres s'il le faut).

Algorithme :

(0) $a_1 \geq a_2 \geq ... \geq a_m \;\; b_j \quad j = 1,2,...,m$, e_i = entrepôt $i \quad i = 1,2,...,n$

(1) Poser $S = \{e_1\}, k = 1$

(2) Tant que $\sum_{i=1}^{k} a_i < \sum_{j=1}^{m} b_j$

poser : $k := k + 1$

$S := S \cup \{e_k\}$

Le nombre minimum d'entrepôts à utiliser pour satisfaire la demande est égal à (k).

Pour trouver une solution qui privilégie le premier critère (qui minimise le nombre d'entrepôts utilisés) on résout le problème de transport classique suivant :

$$\begin{cases} \sum_{i=1}^{k} f_{ij} = b_j \quad j = 1,2,...,m \\ \sum_{j=1}^{m} f_{ij} \le a_i \quad i = 1,2,..,k \\ \sum_{i=1}^{k}\sum_{j=1}^{m} c_{ij} f_{ij} = Z(Min) \\ f_{ij} \in IN \end{cases}$$

(k : valeur donnée par l'algorithme précédent)

Remarque :

- Si $\sum_{i=}^{k} a_i = \sum_{j=1}^{m} b_j$ et $a_k > a_{k+1}$ la solution est efficace
- Si $\sum_{i=}^{k} a_i = \sum_{j=1}^{m} b_j$ et $a_k = a_{k+1}$ on doit examiner les solutions obtenues en considérant toutes les combinaisons d'ordre l ($l = card\{i \le k / a_i = a_k\}$) parmi l'ensemble des entrepôts $J' = \{e_i / a_i = a_k\}$.
- Si $\sum_{i=1}^{k} a_i - \sum_{j=1}^{m} b_j = \alpha > 0$ on prend la même précaution en remarquant que :

  -Si $i \le k$ et $a_i > a_{k+1} + \alpha$ on doit obligatoirement utiliser l'entrepôt i

  -Si $i \le k$ et $\exists j > k / a_i \le a_j + \alpha$ on peut inter-changer les entrepôts $a_t (\forall t / i \le t \le k)$ et $a_{t'} (\forall t' / k+1 \le t' \le j)$ et considéré de même toutes les combinaisons formées de k entrepôts pour trouver une solution efficace.

Solution de compromis :

a) Première approche :

On considère la solution précédente $f = (f_{ij})$ avec l'ensemble J des indices des entrepôts utilisés $J = \{1,2,...,k\}$ (k minimum).

On calcule $\underset{i \in J}{Max} \sum_{j=1}^{m} c_{ij} f_{ij} = \sum_{j=1}^{m} c_{ij} f_{ij}$.

Pour tout couple $(r,t)/n \geq r > t > k$ et $a_r + a_t \geq a_i$ on pose $J = J - \{i\} \cup \{r,t\}$ et on résout un nouveau problème de transport avec $k+1$ entrepôts puis on choisira la solution de coût minimal. Cette procédure en diminuant les coûts mais en augmentant le nombre d'entrepôts va engendrer des solutions qui peuvent être des solutions de compromis.

b) Deuxième approche :

A partir de la solution $f = (f_{ij})$ donnée par la résolution du problème de transport, sans tenir compte du nombre d'entrepôts utilisés, on peut trouver une solution de compromis de la manière suivante :

Soit J l'ensemble des entrepôts utilisés pour cette solution.

S'il existe $t \in J / \sum_{i \in J} a_i - \sum_{j=1}^{m} b_j \geq a_t$ alors on résout le problème de transport suivant :

$$\begin{cases} \sum_{\substack{i \in J \\ i \neq t}} f_{ij} = b_j \quad , \quad j = 1,2,...,m \\ \sum_{j=1}^{m} f_{ij} \leq a_i \quad , \quad i \in J, i \neq t \ , \ f_{ij} \in N \\ \sum_{\substack{i \in J \\ i \neq t}} \sum_{j=1}^{m} c_{ij} f_{ij} = Z(Min). \end{cases}$$

Nous aboutissons alors à une solution $f^t = (f_{ij}^t)$ avec le nombre d'entrepôts utilisés qui a diminué d'une unité. Prenons maintenant la solution f^{t_0} qui réalise $\underset{t}{Min} \sum_{\substack{i \in J \\ i \neq t}} \sum_{j=1}^{m} c_{ij} f_{ij}^t$.

f^{t_0} constitue une solution de compromis avec un entrepôt utilisé en moins. Si on veut diminuer encore le nombre d'entrepôts utilisés on a qu'à reprendre la même procédure avec la solution f^{t_0} et $J' = J - \{t_0\}$.

On obtient ainsi des solutions qui diminuent le nombre d'entrepôts utilisés mais qui augmentent le coût. Ces solutions peuvent représenter des solutions de compromis.

2) Deuxième problème :

Dans beaucoup de problèmes de transport on cherche à se retrouver avec le maximum d'entrepôts vidés de leurs contenus après l'opération d'affectation pour libérer ces derniers et éviter de payer des coûts d'occupation ou dans le cas des chambres froides de n'en laisser en marche que le minimum afin de réduire les coûts de fonctionnement, ou tout simplement pour réduire les problèmes de gestion.

Nous sommes devant un problème bicritère qui consiste à vider le maximum d'entrepôts et à minimiser le coût se rapportant à toute l'opération de transport.

Remarque [3] :

Si on a pour chaque entrepôt (e_i) un coût de stockage unitaire (c_i) le problème se résout aisément en ajoutant un magasin fictif avec une demande ($b_0 = \sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j$) et ça revient à résoudre le problème de transport classique suivant :

$$\begin{cases} \sum_{j=0}^{m} f_{ij} \le a_i \quad i = 1,...,n \\ \sum_{i=1}^{n} f_{ij} = b_j \quad j = 0,...,m \quad f_{ij} \in N \qquad (\ c_{i0} = c_i) \\ Z = Min \sum_{i=1}^{n} \sum_{j=0}^{m} c_{ij} f_{ij} \end{cases}$$

a) Première approche :

Si on prend en considération les coûts d'occupation des entrepôts T_i (supposés fixes) on peut modéliser notre problème de la manière suivante :

Pour toute solution f_{ij} du problème de transport posant : $y_i = \begin{cases} 0 \quad si \quad a_i = \sum_{j=1}^{m} f_{ij} \\ 1 \quad si \quad a_i > \sum_{j=1}^{m} f_{ij} \end{cases}$

Alors notre problème peut s'écrire comme suit :

$$\begin{cases} \sum_{j=1}^{m} f_{ij} \le a_i \\ \sum_{i=1}^{n} f_{ij} = b_j \\ Min \sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij} f_{ij} + \sum_{i=1}^{n} y_i T_i \\ Min \sum_{i=1}^{n} y_i \end{cases} \qquad f_{ij} \in IN; \quad y_i \in \{0,1\}.$$

Du fait que ce problème est bicritère et y_i est une variable bivalente, on ne dispose pas de méthode pour le résoudre. Néanmoins si on veut maximiser le nombre d'entrepôts vides on peut utiliser la procédure suivante :

(0) $a_1 \le a_2 \le ... \le a_n$ poser $S = \{e_1\}, k = 1$

(1) Si $\sum_{i=1}^{k} a_i < \sum_{j=1}^{m} b_j$

poser : $k = k + 1$

$S = S \cup \{e_k\}$

Allen en (1)

Sinon Aller en (2)

(2) On peut vider au maximum k entrepôts $(e_1, e_2, ..., e_k)$.

Alors pour trouver une solution efficace à notre problème on peut résoudre le programme linéaire suivant :

$$\begin{cases} \sum_{j=1}^{m} f_{ij} = a_i \qquad i = 1,2,...,k \\ \sum_{j=1}^{m} f_{ij} \le a_i \qquad i = k+1,...,n \\ \sum_{i=1}^{n} f_{ij} = b_j \qquad j = 1,....,m \\ \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij} f_{ij} = Z(\min) \end{cases} \qquad f_{ij} \in IN$$

Sinon on peut décomposer notre problème et développer une heuristique pour trouver une solution qui maximise le nombre d'entrepôts vidés en procédant de la manière suivante :

Inversons le problème de transport où les entrepôts $e_1, e_2, ..., e_k$ deviennent des points de vente avec des demandes $a_1, a_2, ..., a_k$ et les points de vente des entrepôts avec des offres $b_1, b_2, ..., b_m$ .Résolvons le problème de transport suivant :

$$\begin{cases} \sum_{j=1}^{m} f'_{ji} = a_i \quad i = 1,...,k \\ \sum_{i=1}^{k} f'_{ji} \le b_j \quad j = 1,...,m \quad f'_{ji} \in IN \\ \sum_{i=1}^{k} \sum_{j=1}^{m} c_{ij} f'_{ji} = Z(Min) \end{cases}$$

Considérons la solution $f^{(1)} = (f_{ij}^{(1)})$ telle que :

$$f_{ij}^{(1)} = \begin{cases} f'_{ji} & si\ i \le k \\ 0 & si\ i \ge k+1 \end{cases}$$

Puis posons $b'_j = b_j - \sum_{i=1}^{n} f_{ij}^{(1)}$ et résolvons le problème de transport constitué des entrepôts $e_{k+1}, e_{k+2}, .., e_n$ avec des offres $a_{k+1}, .., a_n$ et de m points de vente avec des demandes $b'_1, .., b'_m$:

$$\begin{cases} \sum_{i=k+1}^{n} f''_{ij} = b'_j \quad j = 1,2,..,m \\ \sum_{j=1}^{m} f''_{ij} \le a_i \quad i = k+1,...,n \quad f''_{ij} \in IN \\ \sum_{i=k+1}^{n} \sum_{j=1}^{m} c_{ij} f''_{ij} = Z(Min) \end{cases}$$

Posons $f^{(2)} = (f_{ij}^{(2)})$ telle que : $f_{ij}^{(2)} = \begin{cases} 0 & si\ i \le k \\ f''_{ij} & si\ i \ge k+1 \end{cases}$.

Si : $t \le k$, $i > k$, $j = 1,...,m$ et $l = 1,...,m$ tels que : $f'_{tj} > 0$, $f''_{il} > 0$ et $c_{tl} + c_{ij} < c_{tj} + c_{il}$.

alors poser :

$$\begin{cases} f'_{tj} = f'_{tj} - 1 \\ f'_{tl} = f'_{tl} + 1 \\ f''_{il} = f''_{il} - 1 \\ f''_{ij} = f''_{ij} + 1 \end{cases}$$

Ainsi on peut déduire une solution $f = (f_{ij})_{i=\overline{1,n};\, j=\overline{1,m}}$ qui maximise le nombre d'entrepôts vides telle que : $f_{ij} = f_{ij}^{(1)} + f_{ij}^{(2)}$

Remarque : On doit reprendre les mêmes remarques vues dans le premier problème à savoir si $a_k < a_{k+1}$ la solution trouvée est efficace, sinon on doit reconsidérer toutes les combinaisons décrites précédemment .

b) Deuxième approche :

A partir de la solution du problème de transport (sans tenir compte des coûts d'occupation T_i) on va essayer de trouver une solution qui va augmenter le nombre d'entrepôts vides tout en évitant d'accroitre excessivement le coût de transport.
Cette solution peut représenter une solution de compromis qui privilégie le critère des coûts. A cet effet développons la procédure suivante :

(0) $k = 1$, $F = \{1,2,..,n\}$, a_i quantité disponible dans l'entrepôt (i).

T_i coût d'occupation de l'entrepôt (i), c_{ij} coût de transport de (i) vers (j).

(1)Si $F = \phi$ Aller en (4).

Sinon : poser $I = \{1,2,...,m\}$

Trouver f_{ij} la solution optimale donnée par la résolution du problème de transport. Aller en (2)

(2)Calculer $\underset{i \in I \cap F}{Min}\left[a_i - \sum_{j=1}^{m} f_{ij} > 0 \right] = \varepsilon$

$$J = \left\{ i \in I \cap F \,/\, a_i - \sum_{j=1}^{m} f_{ij} = \varepsilon \right\}.$$

Soit $\hat{\imath} \in J$.

- Trouver $\varepsilon_1, \varepsilon_2, ..., \varepsilon_m$ et λ_{ij} $(i = 1,2,...,n$ et $i \neq \hat{\imath}$, $j = 1,2,...,m)$ tels que:

$$\begin{cases} \sum_{j=1}^{m} \varepsilon_j = \varepsilon \\ \sum_{\substack{i=1 \\ i \neq \hat{\imath}}}^{n} \lambda_{ij} - \varepsilon_j = 0 \qquad \varepsilon_j \in IN\ , \lambda_{ij} \in IN \quad (i \neq \hat{\imath}) \\ \lambda_{ij} \leq f_{ij} \qquad i = 1,2,...,n,\ i \neq \hat{\imath}\ ; j = 1,2,...,m \\ \sum_{j=1}^{m} c_{ij}\varepsilon_j - \sum_{\substack{i=1 \\ i \neq \hat{\imath}}}^{n} \lambda_{ij} c_{ij} = Z(Min) \end{cases}$$

Si $I = \{\hat{\imath}\}$ Aller en (3).

Si $T_{\hat{\imath}} > c_{\hat{\imath}j}\varepsilon_j - \sum_{\substack{i=1 \\ i \neq \hat{\imath}}}^{n} \sum_{j=1}^{m} \lambda_{ij} c_{ij}$ poser :

$$f_{\hat{\imath}j} = f_{\hat{\imath}j} + \varepsilon_j$$

$$f_{ij} = f_{ij} - \lambda_{ij} \quad \text{si } i \neq \hat{\imath}\ .$$

$$I = I - \{\hat{\imath}\}$$

Sinon poser :

$$I = I - \{\hat{\imath}\}\ ; \quad \text{Aller en (2)}$$

(3)Soit $P_i = \begin{cases} 1 & \text{si } a_i - \sum_{j=1}^{m} f_{ij} > 0 \\ 0 & \text{sinon} \end{cases}$

$f^{(k)} = (f_{ij})_{\substack{i=1,..,n \\ j=1,...,m}}$.; $Z^{(k)} = C_{ij} f_{ij}^{(k)} + \sum_{i=1}^{n} T_i P_i$; $P^{(k)} = \sum_{i=1}^{n} P_i$ (nombre d'entrepôts vidés).

$k = k + 1$; $F = F - \{\hat{\imath}\}$; Aller en (1).

(4) Soit $Z^{(t)} = \underset{1 \leq r \leq k}{Min} \left[Z^{(r)} \right]$; $f^{(t)}$ Est une solution minimale avec $P^{(t)}$ entrepôts vidés.

Dans le cas où vider le maximum d'entrepôts est une commodité désirée qui ne s'exprime pas en termes de coût, mais obéit à une stratégie d'Independence ou

d'organisation de la gestion, on peut changer notre procédure de la façon suivante pour générer des solutions de compromis et choisir la plus indiquée.

(0) $J = \phi$, $I = \{1,2,...,n\}, a_1, a_2, ..., a_n$; (f_{ij}) $i = 1,2,...,n$ et $j = 1,2,...,m$, $C_{ij}, k = 1$.

(1)Soit $\underset{i \in I}{Min}\left(a_i - \sum_{j=1}^{m} f_{ij} > 0 \right) = \varepsilon$ $(\varepsilon \in IN^*)$; Soit $\hat{\imath} \in I / a_{\hat{\imath}} - \sum_{j=1}^{m} f_{\hat{\imath}j} = \varepsilon$;

Si $\sum_{i \in J} a_i \geq \sum_{i=1}^{n} \sum_{j=1}^{m} f_{ij}$; Aller en (2)

Sinon trouver $\varepsilon_1, \varepsilon_2, ..., \varepsilon_m$ et λ_{ij} $(i = 1,2,...,n$ et $i \neq \hat{\imath}$, $j = 1,2,...,m)$ tels que:

$$\begin{cases} \sum_{j=1}^{m} \varepsilon_j = \varepsilon \\ \sum_{\substack{i=1 \\ i \neq \hat{\imath}}}^{n} \lambda_{ij} - \varepsilon_j = 0 \qquad \varepsilon_j \in IN \, , \lambda_{ij} \in IN \qquad (i \neq \hat{\imath}) \\ \lambda_{ij} \leq f_{ij} \qquad i = 1,2,...,n, \; i \neq \hat{\imath} \; ; j = 1,2,...,m \\ \sum_{j=1}^{m} c_{ij} \varepsilon_j - \sum_{\substack{i=1 \\ i \neq \hat{\imath}}}^{n} \lambda_{ij} c_{ij} = Z(Min) \end{cases}$$

poser : $f_{\hat{\imath}j} = f_{\hat{\imath}j} + \varepsilon_j$

$f_{ij} = f_{ij} - \lambda_{ij}$ si $i \neq \hat{\imath}$.

$C(k) = \sum_{i=1}^{n} \sum_{j=1}^{m} C_{ij} f_{ij}$

$f_{ij}^{(k)} = f_{ij}$ $i = 1,2,...,n$; $j = 1,2,...,m$

$I = I - \{\hat{\imath}\}$; $J = J \cup \{\hat{\imath}\}$; $k = k + 1$; aller en (1).

(2) J est l'ensemble des entrepôts vidés.

Pour $t = 1,2,...,k$: $f_{ij}^{(t)}$: $i = 1,2,...,n$; $j = 1,2,...,m$; est une solution avec un coût de transport $C(t)$ et (t) entrepôts vidés.

Conclusion

L'accroissement de la population et la multiplication des réseaux routiers et les progrès technologiques nous imposent d'envisager des solutions efficaces à des problèmes de transport multicritère de plus en plus compliqués. Selon les besoins, les moyens et les objectifs fixés la recherche opérationnelle peut développer des méthodes appropriées pour apporter une solution à ces problèmes.

Dans le cadre de notre travail on peut envisager un cas plus général où chaque entrepôt ne peut fournir que certains points de vente. Dans le cas du problème de transport classique il suffit de poser $C_{ij} = \infty$ pour les cas impossibles, mais dans notre problème il faut penser à changer complètement les procédures car on ne tient pas comptes des coûts.

De même on peut considérer le cas où il existe un coût d'occupation des entrepôts qui varie selon une fonction linéaire de la quantité en stock ($T_i = \alpha_i S_i + \beta_i$). Cela impliquera une difficulté supplémentaire à surmonter.

En introduisant les valeurs t_{ij} (t_{ij} = temps nécessaire pour un déplacement de i à j) , on peut envisager un troisième critère qui consiste à minimiser le temps de l'opération de transport.

Le problème de transport peut être utilisé dans de nombreux domaines, c'est pour ça qu'il suscite des centres d'intérêts divers. Toute avancée dans les techniques d'optimisation multicritère lui sera aisément adaptée.

REFERENCES

[1] Acher-J et Gardelle-J : Programmation linéaire ; Dunod-Decision 1978

[2] Desbazeille-G : Exercices et problèmes de recherche opérationnelle ; Dunod 1978

[3] Doesbke-F,Hallim-M,Lefevre-C : Les graphes par l'exemple ; Ellipses 1986

[4] Maiko-S,Ichiro-T,Yoshitsugu-Y: Minimum Maximal Flow problem: an optimization over the efficient set; scientific research c(2) 136500601of the ministry of education, culture, sport, science and technology of Japan 2001

[5] Price -W-L : Introduction aux graphes et aux reseaux ; Masson et c^{ie} 1974

[6] Ralph E-E : Multiple criteria optimisation :Theory, computation and application; . John Willey & sons 1986

[7] Roy-B : Algèbre moderne et théorie des graphes ; Dunod 1970

[8] Sakarovitch-M : Optimisation combinatoire, Programmation discrète ; Hermann 1984

[9] Sakarovitch-M: Optimisation combinatoire, Graphes et programmation ; Herman 1984